

TongTech °

TongWeb 7 用户使用手册

东方通科技

1	TongWeb7 应用服务器概述.....	13
1.1	概述	13
1.2	JavaEE7 Web Profile 的新特性.....	13
1.3	TongWeb7 的体系结构	14
1.4	应用开发工具.....	14
1.5	TongWeb7 的特性	15
1.6	集成的第三方产品.....	15
1.7	规范支持	16
2	TongWeb7 安装指南.....	16
2.1	系统配置要求.....	16
2.2	安装 TongWeb7 应用服务器	17
2.2.1	Windows 平台上界面安装	17
2.2.2	Linux 平台命令行安装	22
2.2.3	Linux 平台上静默安装	27
2.2.4	安装 License	27
2.3	TongWeb7 开始向导	28
2.3.1	TongWeb7 应用服务器目录说明	28
2.3.2	启动服务器	28
2.3.2.1	在 Windows 上启动服务器.....	28
2.3.2.2	在 Linux 平台上启动服务器.....	28
2.3.2.3	安全启动.....	28
2.3.3	宕机重启模式	29
2.3.3.1	宕机重启模式用法.....	29
2.3.3.2	宕机重启使用场景.....	29
2.3.4	管理控制台	30
2.3.4.1	登录.....	30
2.3.4.2	注销.....	31
2.3.4.3	导航栏.....	31
2.3.4.4	首页.....	31
2.3.5	JConsole	32
2.3.6	停止服务器	33
2.3.6.1	Windows.....	33
2.3.6.2	Unix/Linux.....	34
2.4	卸载 TongWeb7 应用服务器	34
2.4.1	Windows 平台上卸载	34
2.4.2	Linux 平台上卸载	35
3	应用管理	35
3.1	应用管理概述.....	35
3.1.1	应用类型支持	35
3.1.2	应用部署方式	35
3.1.3	应用管理方式	35
3.1.4	应用形态结构	36
3.1.4.1	Web 应用结构.....	36
3.1.4.2	EJB 应用结构	37
3.1.4.3	EAR 应用结构.....	38
3.1.4.4	RAR 应用结构.....	39
3.1.5	应用部署附加属性	39
3.1.5.1	应用名称.....	40
3.1.5.2	应用前缀.....	40
3.1.5.3	JSP 预编译.....	40
3.1.5.4	共享库.....	40

3.1.5.5	类加载顺序.....	41
3.1.5.6	静态资源缓存最大值.....	42
3.1.5.7	应用描述.....	42
3.1.5.8	虚拟主机.....	42
3.1.6	应用自定义部署描述文件.....	42
3.2	管理控制台应用管理.....	42
3.2.1	查看已部署的应用.....	42
3.2.2	应用部署.....	43
3.2.3	应用查看与编辑.....	48
3.2.4	查看应用子模块.....	50
3.2.5	应用解部署.....	51
3.2.6	应用重部署.....	51
3.2.7	应用访问.....	52
3.2.8	应用停止.....	53
3.2.9	应用启动.....	53
3.2.10	应用更新.....	53
3.2.11	应用版本管理.....	55
3.2.12	Connector 应用.....	55
3.2.12.1	Connector 应用部署.....	55
3.2.12.2	应用查看与编辑.....	56
3.2.12.3	应用重部署.....	57
3.2.12.4	应用解部署.....	57
3.3	自动部署.....	57
3.3.1	设置自动部署目录.....	57
3.3.2	自动部署支持的应用类型.....	58
3.3.3	自动部署支持的部署方式.....	58
3.3.4	默认的自动部署目录.....	58
3.3.5	自动部署应用.....	58
3.3.6	自动解部署应用.....	59
3.3.7	自动重部署应用.....	59
3.4	热部署.....	59
3.4.1	热部署配置.....	59
3.4.2	热部署应用.....	59
4	Web 容器.....	60
4.1	Web 容器说明.....	60
4.1.1	Web 容器配置.....	60
4.2	会话高可用.....	61
4.2.1	会话高可用提供的功能.....	61
4.2.2	会话高可用的使用.....	61
4.3	访问日志.....	62
4.3.1	访问日志提供的功能.....	62
4.3.1.1	访问日志基本功能.....	62
4.3.2	访问日志的使用.....	63
4.3.3	访问日志格式.....	64
4.3.4	访问日志使用示例.....	65
4.4	虚拟主机.....	65
4.4.1	虚拟主机提供的功能.....	65
4.4.1.1	虚拟主机的基本功能.....	65
4.4.1.2	缺省虚拟主机.....	65
4.4.1.3	单点登陆.....	66
4.4.1.4	远程访问过滤.....	66

4.4.2	虚拟主机的使用	66
4.4.2.1	创建虚拟主机	66
4.4.2.2	查看/编辑虚拟主机	68
4.4.2.3	启动/停止虚拟主机	69
4.4.2.4	删除虚拟主机	69
4.4.3	虚拟主机使用示例	69
4.4.3.1	远程访问过滤	69
4.5	通道	70
4.5.1	通道提供的功能	70
4.5.1.1	通道的基本功能	70
4.5.1.2	长连接	70
4.5.1.3	通道的工作模式	70
4.5.1.4	传输压缩(HTTP&HTTPS)	71
4.5.1.5	SSL(HTTPS)	71
4.5.2	通道的使用	71
4.5.2.1	创建通道	71
4.5.2.2	查看/编辑通道	75
4.5.2.3	启动/停止通道	78
4.5.2.4	删除通道	78
4.5.3	配置使用说明	78
4.5.3.1	通道重定向	78
4.5.3.2	HTTP 通道的重定向	79
4.5.3.3	Proxy 转发	79
4.6	虚拟主机与通道的关系	80
4.7	资源	80
4.7.1	文件集	80
4.7.2	共享库	81
4.7.2.1	共享库类型	82
4.8	类加载	83
4.8.1	类加载机制	83
4.8.2	类加载模式	84
4.8.3	类加载推荐策略	85
4.8.4	类加载参数	85
4.9	其他	85
4.9.1	设置 Session Cookie 的名字	85
4.9.2	应用上下文共用 Session	86
4.9.3	多应用 Session 共享	86
4.9.4	虚拟目录	86
5	JDBC 配置	86
5.1	JDBC 数据源概述	86
5.2	TongWeb7 中的 JDBC 数据源概述	86
5.2.1	连接池管理功能描述	87
5.2.2	空闲超时的处理	87
5.2.3	泄露连接的处理	87
5.2.4	获取连接的处理	87
5.2.5	连接有效性检查	87
5.2.6	动态驱动路径加载	88
5.2.7	语句缓存	88
5.2.8	故障转移	88
5.2.9	负载均衡	88
5.2.10	多数据源状态检查	88

5.3	JDBC 数据源的使用.....	89
5.3.1	创建数据源连接池.....	89
5.3.2	查看/编辑连接池.....	94
5.3.3	测试连接.....	96
5.3.4	删除 JDBC 连接池.....	97
5.4	多数据源 JDBC 的使用.....	97
5.4.1	创建多数据源连接池.....	97
5.4.2	查看/编辑多数据源连接池.....	98
5.4.3	测试连接.....	98
5.4.4	删除 JDBC 连接池.....	98
6	EJB.....	99
6.1	EJB2.X 特性.....	99
6.2	EJB3.0 特性.....	100
6.3	EJB3.1 特性.....	100
6.4	EJB3.2 新特性.....	101
6.5	EJB2.x 应用结构.....	101
6.6	EJB2.x 使用说明.....	102
6.6.1	配置说明.....	102
6.6.2	CMP 使用步骤.....	102
6.6.3	BMP 使用步骤.....	102
6.7	JPA.....	102
6.7.1	EclipseLink.....	103
6.7.2	使用说明.....	104
6.7.3	JPA 事物支持说明.....	106
6.8	EJB 容器.....	106
6.8.1	EJB3.2 特性支持.....	106
6.8.2	EJB 实例管理.....	106
6.8.2.1	无状态会话 Bean 的实例池.....	106
6.8.2.2	有状态会话 Bean 的实例缓存.....	107
6.8.2.3	消息驱动 Bean 的实例池.....	107
6.8.3	查看/编辑 EJB 配置属性.....	107
6.8.3.1	查看/编辑无状态会话 Bean 实例池属性.....	107
6.8.3.2	查看/编辑有状态会话 Bean 属性.....	108
6.8.3.3	查看/编辑单例会话 bean 属性.....	109
6.8.3.4	查看/编辑消息驱动 Bean 实例池属性.....	109
6.8.4	EJB 远程调用.....	109
6.8.4.1	远程调用协议及方式.....	109
6.8.4.2	远程调用配置.....	109
6.8.5	EJB 集群.....	110
6.8.5.1	支持故障转移.....	110
6.8.5.2	支持故障隔离和恢复.....	110
6.8.5.3	支持负载均衡.....	110
6.8.5.4	使用说明.....	110
6.9	全局事务.....	111
6.9.1	全局事务概述.....	111
6.9.2	全局事务场景描述.....	111
6.9.3	全局事务传播策略.....	111
6.9.4	全局事务配置.....	112
7	基础服务配置.....	112
7.1	JNDI.....	112
7.1.1	TongWeb7 中的 JNDI 概述.....	112

7.1.2	InitialContext 的环境属性	112
7.1.2.1	访问本地资源的初始化上下文环境属性	112
7.1.2.2	访问远程资源的初始化上下文环境属性	112
7.1.3	JNDI 命名空间	112
7.1.3.1	全局命名空间	112
7.1.3.2	应用命名空间	114
7.1.3.3	模块命名空间	114
7.1.3.4	组件命名空间	114
7.1.4	JNDI 树展示	115
7.1.4.1	JNDI 展示	115
7.1.5	代码中使用 JNDI 的示例	117
7.1.6	应用移植	117
7.1.6.1	全局 JNDI 名应用移植	117
7.1.6.2	组件 JNDI 名应用移植	118
7.2	安全服务	118
7.2.1	安全服务概述	118
7.2.2	TongWeb7 中的安全服务	118
7.2.2.1	安全域	118
7.2.3	安全服务的使用	119
7.2.3.1	安全域概述	119
7.2.3.2	安全域基本属性	120
7.2.3.3	创建文件安全域	121
7.2.3.4	创建 LDAP 安全域	122
7.2.3.5	创建 JDBC 安全域	123
7.2.3.6	创建 Jaas 安全域	124
7.2.3.7	创建脚本安全域	125
7.2.3.8	创建服务扩展安全域	126
7.2.3.9	SSL 证书方式认证和授权	126
7.2.3.10	安全管理器	128
7.2.4	管理控制台三员分立	128
7.2.4.1	系统管理员	128
7.2.4.2	安全保密管理员	130
7.2.4.3	安全审计员	131
7.3	诊断	131
7.3.1	系统日志	131
7.3.1.1	查看系统日志	131
7.3.1.2	下载日志	132
7.3.1.3	搜索日志	132
7.3.2	SQL 日志	133
7.3.3	访问日志	133
7.3.4	快照	134
7.3.4.1	生成快照和下载	134
7.3.4.2	删除快照	136
7.3.4.3	设置快照路径	136
7.3.4.4	快照回放	136
7.3.4.5	快照下载	136
7.3.5	SNMP (简单网络协议)	136
7.3.5.1	SNMP 代理服务配置	136
7.3.5.2	SNMP 代理服务示例	138
7.4	监视	138
7.4.1	监视概述	138

7.4.2	监视配置	138
7.4.3	监视明细	140
7.4.4	模块监视属性	141
7.4.4.1	JVM 内存	141
7.4.4.2	JVM 内存池	141
7.4.4.3	JVM 垃圾收集器	142
7.4.4.4	JVM 线程	142
7.4.4.5	JVM 类加载信息	142
7.4.4.6	JVM 运行时	142
7.4.4.7	操作系统	142
7.4.4.8	TongWeb 信息	142
7.4.4.9	通道信息	142
7.4.4.10	数据源信息	142
7.4.4.11	JVM 编译器信息	142
7.4.4.12	事务信息	143
7.4.4.13	JCA	143
7.4.4.14	应用细节信息	143
7.4.4.15	应用会话信息	143
7.4.4.16	应用类加载器	143
7.4.4.17	应用资源缓存	143
7.4.5	监视概览	143
7.4.6	监视回放	144
7.4.7	hung 线程	145
7.4.8	阈值配置	146
7.5	日志服务	148
7.5.1	日志服务概述	148
7.5.2	模块日志级别配置	148
7.5.3	系统日志配置	150
7.5.4	压缩日志配置	150
7.5.5	日志路径配置	151
7.5.6	审计日志	152
7.5.6.1	操作页面	152
7.5.6.2	配置	153
7.6	JavaMail 资源	153
7.6.1	JavaMail 介绍	153
7.6.2	TongWeb7 中的 JavaMail	153
7.6.3	JavaMail 资源的使用	154
7.6.3.1	创建 JavaMail 资源	154
7.6.3.2	查看/编辑 JavaMail 资源	155
7.6.3.3	删除 JavaMail 资源	155
7.6.4	使用 JavaMail 的示例	155
8	启动参数配置	156
8.1	概述	156
8.2	参数配置	156
8.3	参数格式	157
8.4	使用限制	157
8.5	外部 JVM 参数配置	158
9	JCA	158
9.1	概述	158
9.2	TongWeb7 中的 Connector	158
9.3	线程池	159

9.3.1	连接池概述	159
9.3.2	TongWeb7 中的线程池	159
9.3.3	创建线程池	159
9.3.4	查看/编辑线程池	160
9.3.5	删除线程池	161
9.3.6	Connector 应用中指定线程池	161
9.4	JCA 连接池	161
9.4.1	创建连接池	161
9.4.2	查看/编辑连接池	162
9.4.3	删除连接池资源	163
9.4.4	创建安全映射	163
9.4.5	查看/编辑安全映射	164
9.4.6	删除安全映射	165
9.5	托管对象资源	165
9.5.1	创建托管资源对象	165
9.5.2	查看/编辑托管资源对象	166
9.5.3	删除托管资源对象	167
10	工作管理器	167
10.1	概述	167
10.2	创建工作管理器	167
10.3	查看/编辑工作管理器	168
10.4	删除工作管理器资源	169
10.5	使用工作管理器	169
11	类加载分析工具	169
11.1	概述	169
11.2	名词定义	170
11.2.1	冗余	170
11.2.2	潜在冲突	170
11.3	类加载器树	170
11.3.1	类加载器树概述	170
11.3.2	类加载器树使用示例	170
11.4	类资源分析	170
11.4.1	类资源分析概述	170
11.4.2	类资源分析使用示例	170
11.4.3	类资源分析结果	171
11.5	类冲突检测	171
11.5.1	类冲突检测概述	171
11.5.2	类冲突检测使用示例	171
11.5.3	类冲突检测结果	172
11.5.4	类冲突检测报告	172
12	JMS 服务	172
12.1	概述	172
12.2	JMS 主要功能	172
12.3	JMS 提供的主要接口	173
12.4	TongWeb7 中的 JMS	174
12.4.1	与 ActiveMQ 集成	176
12.4.2	与 TongLINKQ8.1 集成	178
12.4.3	查看/编辑集成属性	179
12.5	JMS 资源的使用	179
12.5.1	创建连接工厂资源	180
12.5.2	查看/编辑连接工厂资源	182

12.5.3	删除连接工厂资源.....	183
12.5.4	创建目的地资源.....	183
12.5.5	查看/编辑目的地资源.....	185
12.5.6	删除目的地资源.....	185
12.6	JMS 使用示例.....	186
	示例一：使用 ActiveMQ——JNDI 集成方式（使用默认提供的名字服务）.....	186
	示例二：使用 ActiveMQ——JavaBean 集成方式.....	188
	示例三：使用 TongLINK/Q8.1.....	191
13	TongWeb 域.....	193
13.1	概述.....	193
13.2	创建 TongWeb 域.....	193
13.3	删除 TongWeb 域.....	193
13.4	启动 TongWeb 域.....	193
13.5	停止 TongWeb 域.....	194
13.6	注意事项.....	194
14	安全防护.....	194
14.1	HttpOnly 支持.....	194
14.2	HTTP trace 支持.....	194
14.2.1	启用 trace 功能.....	194
14.2.2	禁用 trace 功能.....	195
14.3	X-Frame-Options.....	195
14.4	防 SDOS 攻击（慢攻击）.....	196
14.5	支持国密算法的负载均衡软件 THS.....	197
15	工具使用指南.....	197
15.1	Eclipse 中 TongWeb7 插件.....	197
15.1.1	功能概述.....	197
15.1.2	安装 Eclipse.....	197
15.1.3	安装 Eclipse 插件.....	197
15.1.4	Eclipse 插件的使用.....	197
15.1.4.1	添加 TongWeb7 服务器.....	197
15.1.4.2	修改服务器实例配置.....	201
15.1.4.3	添加和移除工程.....	201
15.1.4.4	启动和停止服务器.....	203
15.1.4.5	使用 Debug 功能.....	204
15.1.5	Eclipse 插件的卸载.....	204
15.2	Eclipse 中混淆器使用.....	204
15.3	IDEA 插件.....	205
15.3.1	IDEA 插件功能概述.....	205
15.3.2	IDEA 插件的安装与卸载.....	206
15.3.2.1	安装 IDEA.....	206
15.3.2.2	安装 IDEA 插件.....	206
15.3.2.3	卸载 IDEA 插件.....	206
15.3.3	IDEA 中 TongWeb7 插件的使用.....	206
15.3.3.1	在 IDEA 中添加 TongWeb7.....	206
15.3.3.2	部署工程.....	209
15.3.3.3	使用 Debug 功能.....	212
15.3.3.4	解部署工程.....	212
15.4	Patch 补丁工具使用说明.....	212
15.4.1	工具简介.....	212
15.4.2	环境需求.....	212
15.4.3	配置说明.....	212

15.4.4	制作补丁	214
15.4.5	升级补丁	214
15.5	TongAPM 工具使用说明	215
15.5.1	安装 TongAPM 工具	215
15.5.2	APM 配置	216
15.5.3	慢请求分析首页	217
15.5.4	慢请求追踪	217
15.5.5	类方法分析	218
15.5.6	线程刨析	218
15.5.7	JDBC-TOP SQL	219
15.5.8	JDBC-JDBC 资源泄漏	219
15.5.9	内存分析-潜在内存泄漏	220
15.5.10	大对象分析	220
15.6	applient 工具	221
15.6.1	安装 Eclipse	221
15.6.2	在 Eclipse 中创建客户端应用	221
15.6.3	Applient 工具调用 EJB	226
1.	附录 1 应用配置说明	226
1.1	.tongweb-web.xml	226
1.2	.tongweb-ejb-jar.xml	230
1.3	.default-web.xml	231
1.4	.ejb-jar.xml	232
1.5	.persistence.xml	242
2.	附录 2 TongWeb7 配置说明	243
2.1.	EJB 配置说明	243
2.1.1.	无状态会话 bean 配置	243
2.1.2.	有状态会话 bean 配置	244
2.1.3.	单例会话 bean 配置	244
2.1.4.	消息驱动 bean 配置	244
2.2.	JDBC 配置说明	245
2.2.1.	连接池基本配置	245
2.2.2.	连接池池设置	246
2.2.3.	连接池验证连接属性配置	247
2.2.4.	连接池的高级属性配置	248
2.3.	Web 容器配置说明	250
2.3.1.	容器配置	250
2.3.2.	access-log	250
2.3.3.	virtual-host	253
2.3.3.1.	Remote-filter	253
2.3.4.	http-listener	254
2.3.4.1.	ssl	255
2.3.4.2.	protocol	255
2.3.4.3.	http-options	256
2.3.4.4.	advance	257
2.3.5.	ajp-listner	258
2.3.5.1.	ajp-options	259
2.4.	日志服务配置	259
2.4.1.	日志服务配置属性	259
2.5.	事务配置	260
2.5.1.	事务配置属性	260
2.6.	JSF 配置	260

2.6.1.	JSF 配置属性.....	260
3.	附录 3 常见问题说明.....	261
3.1.	跨域访问技术 CORS 配置说明	261
3.2.	TongWeb 及其集中管理工具对 IPv6 支持.....	263
3.3.	TongWeb7 支持 Invoker Servlet.....	263
3.4.	TongWeb7 对 JDK9 以上版本支持说明	263
4.	附录 4 开机自启动服务安装工具说明.....	263
5.	附录 5 TongWeb7 使用 jmxmp 协议.....	265
6.	附录 6 使用命令行安装 TongWeb.....	265
	installTongWebs.sh num	265
	installTongWebs.sh num pathPrefix.....	266
	installTongWebs.sh configFilePath.....	266
7.	附录 7 jmstool 命令使用说明.....	267
8.	附录 8 参数说明.....	268
9.	附录 9 命令行使用说明.....	275
9.1	基本使用说明.....	275
9.1.1	执行方式.....	275
9.1.2	安全认证.....	275
9.1.3	参数格式.....	275
9.1.4	错误提示.....	275
9.1.5	参数默认值.....	276
9.2	基本参数.....	276
9.3	Commandstool 使用命令	276
9.3.1	change-admin-password.....	276
9.3.2	create-auth-realm.....	276
9.3.3	delete-auth-realm.....	277
9.3.4	create-file-user.....	277
9.3.5	delete-file-user.....	278
9.3.6	create-http-listener.....	278
9.3.7	update-http-listener.....	278
9.3.8	delete-http-listener.....	279
9.3.9	start-http-listener.....	279
9.3.10	stop-http-listener.....	279
9.3.11	create-ajp-listener.....	280
9.3.12	delete-ajp-listener.....	280
9.3.13	start-ajp-listener.....	280
9.3.14	stop-ajp-listener.....	280
9.3.15	create-virtual-server.....	280
9.3.16	delete-virtual-server.....	281
9.3.17	start-virtual-server.....	281
9.3.18	stop-virtual-server.....	281
9.3.19	create-jdbc-connection-pool.....	281
9.3.20	update-jdbc-connection-pool.....	282
9.3.21	delete-jdbc-connection-pool.....	283
9.3.22	deploy.....	283
9.3.23	undeploy.....	284
9.3.24	redeploy.....	284
9.3.25	ping-jdbc-connection-pool.....	285
9.3.26	help.....	285
9.3.27	list-apps.....	285
9.3.28	list-file-users.....	285

9.3.29	list-http-listeners.....	285
9.3.30	list-ajp-listeners.....	285
9.3.31	list-jdbc-connection-pools.....	285
9.3.32	list-virtual-servers.....	286
9.3.33	list-auth-realms.....	286
9.3.34	list-sys-properties.....	286
9.3.35	list-jms-connection-factory.....	286
9.3.36	list-jms-destination.....	286
9.3.37	create-jms-connection-factory.....	286
9.3.38	create-jms-destination.....	287
9.3.39	delete-jms-connection-factory.....	287
9.3.40	delete-jms-destination.....	288
9.3.41	list-adapter-adminobject.....	288
9.3.42	list-connector-connection-pools.....	288
9.3.43	create-connector-connection-pool.....	288
9.3.44	delete-connector-connection-pool.....	289
9.3.45	list-threadpools.....	289
9.3.46	create-threadpool.....	289
9.3.47	delete-threadpool.....	289
9.3.48	create-adapter-adminobject.....	290
9.3.49	delete-adapter-adminobject.....	290
9.3.50	list-connector-security-maps.....	290
9.3.51	create-connector-security-map.....	290
9.3.52	delete-connector-security-map.....	291
9.3.53	version.....	291
9.3.54	web-container-config.....	291
9.3.55	update-web-container-config.....	291
9.3.56	server-log-config.....	292
9.3.57	update-server-log-config.....	292
9.3.58	set-jvm-arg.....	293
9.3.59	delete-jvm-arg.....	293
9.3.60	set-server-arg.....	293
9.3.61	delete-server-arg.....	294
9.3.62	update-auto-deploy-config.....	294
10.	附录 10 提供 NativeJdbcExtractor 工具类.....	294
11.	附录 11 数据库密码加密脚本.....	295
12.	附录 12 通过 JCA 集成 Tuxedo 配置说明.....	295
12.1	Tuxedo 远端服务配置简介.....	295
12.2	com.oracle.tuxedo.TuxedoAdapter.rar 配置.....	296
12.3	TongWeb 相关部署及配置.....	296
13.	附录 13 Apache 作为 Proxy Server 配置说明.....	296
14.	附录 14 TongWeb 提供等同 Apache 的 rewrite 重写机制功能配置说明.....	298
14.1	RewriteCond.....	299
14.2	RewriteRule.....	299

1 TongWeb7 应用服务器概述

1.1 概述

TongWeb7 是遵循 JavaEE7 Web Profile 规范的企业级应用服务器，它为企业应用提供了可靠、可伸缩、可管理和高安全的基础平台。同时具有功能完善、支持开放标准和基于组件开发、多层架构、轻量等特点，为开发和部署企业应用提供了必需的底层核心功能。用户通过 TongWeb7 的管理控制台可方便的对应用进行管理，同时能够监控系统组件和应用运行时的状态及调优。因此 TongWeb7 适用于高度可用、可靠、可伸缩，稳定的业务领域。

1.2 JavaEE7 Web Profile 的新特性

JavaEE7 Web Profile 与 JavaEE6 相比，提供了一些新特性：CDI1.1、EJB3.2、Servlet3.1、JPA2.1、JSF2.2、JCA1.7 和 Bean Validation1.1 等。

1) CDI1.1

依赖注入是一个开发企业应用时越来越流行的技术，CDI 将其扩展到了应用服务器内部的各容器，如 EJB 容器、Web 容器，该规范可以使普通 JavaBean、SessionBean 和 JSF Backing Bean 通过 DI 的方式在应用中使用，并且可以关联到一个特定范围，例如 request 范围、session 范围等。

2) EJB3.2

EJB3.2 与 JavaEE 6 提供的 EJB3.1 版本相比，支持本地异步会话 Bean 调用；有状态会话 bean 的生命周期回调拦截方法，现在可以在一个事务环境中执行；可以完全禁用特定的有状态会话 bean 的钝化；TimerService API 已被扩展，现在可以在同一个 EJB 模块中查询所有活动计时器等等。

3) Servlet 3.1

Servlet3.1 是 JavaEE 7 中重点聚焦的功能之一，它使得开发 web 应用变得更加简单，增加了更多便利的注解支持、提供可插拔设计、提供异步处理的支持等。

4) JPA 2.1

JPA(Java Persistence API)是 JavaEE 5 和 Java SE 共有的有关对象持久化的接口，即对象持久化的。JPA 在 JavaEE7 里得到了进一步的增强，使得 JPA 技术变得更加有效和可靠，对应用提供了更有效的性能提升。

5) JCA1.7

TongWeb7 中的 JCA 架构实现了 JCA1.7 规范，JCA(Java Connector Architecture)提供了一个应用服务器和企业信息系统(EIS)连接的标准 Java 解决方案，以及把这些系统整合起来的方法。JCA 简化了异构系统的集成，用户只要构造一个基于 JCA 规范连接器应用，并将该连接器应用部署到 J2EE 服务器上，这样不用编写任何代码就可以实现 EIS 与 J2EE 应用服务器的集成。

6) JSF 2.2

JSF 是 JavaEE 5 规范中提出的关于 Web 层的开发框架，与其他 Web 框架不同的是 JSF 以用户界面为核心，它将控制粒度细化到页面的“组件”一级，即 JSF 将各类页面元素抽象成 UI(User Interface 即用户界面)组件，这些 UI 组件可以灵活的组装生成页面，并被方便的定制和重用。JSF 使得开发人员摆脱了细碎的 HTML 代码和 JavaScript 脚本调试，可以应用面向对象的思想开发 Web 应用程序。JSF2.2 进一步增强了对 HTML5 开发的支持。

7) Bean Validation1.1

数据验证是贯穿企业应用各处的一个公共任务，从表示层到持久层，每一层都需要数据验证，Bean Validation 避免了每一层重复的验证代码，提供了统一的注解和验证框架。

1.3 TongWeb7 的体系结构

TongWeb7 体系结构图如下：

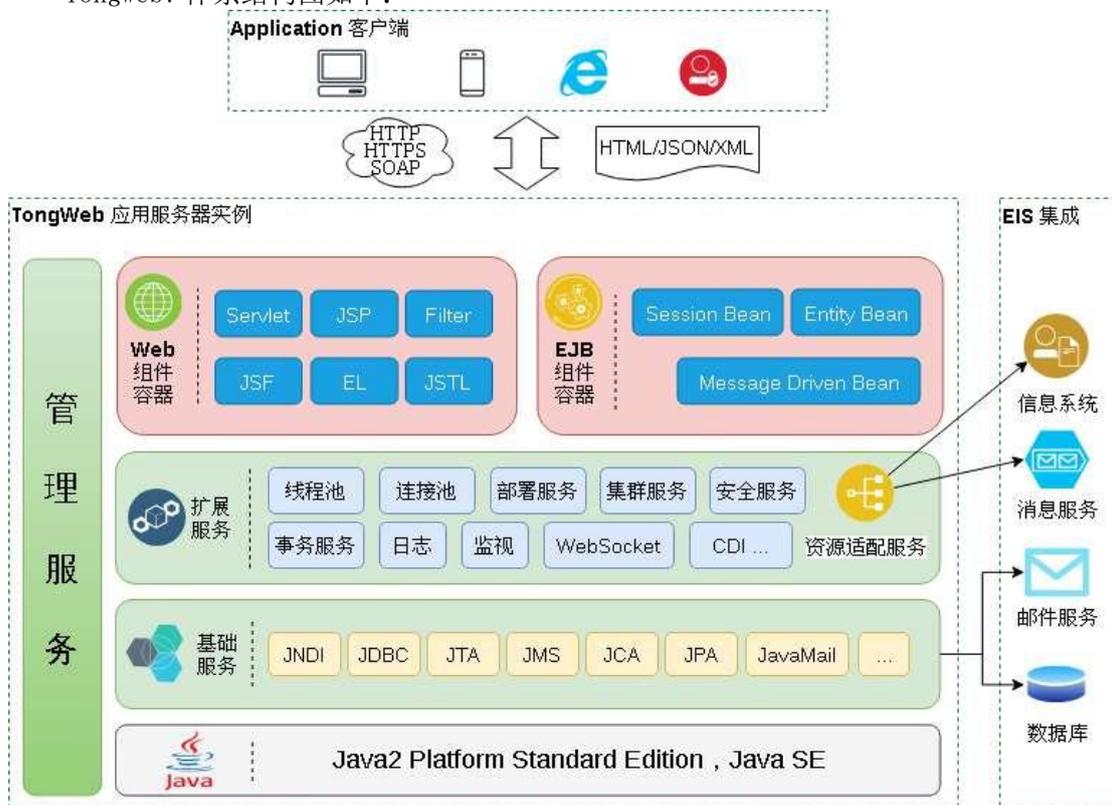


图 1.1 TongWeb7 体系结构图

1.4 应用开发工具

提供 Web 应用、EJB 应用和企业应用的开发工具，并能够部署到 TongWeb7 应用服务器上。

1) 管理控制台

通过管理控制台用户可以方便进行各类型应用的管理，以及各种资源与服务的使用和调优。

2) Web 容器

提供 Web 应用的运行环境，如支持 HTTP 协议，支持 SSL 协议上的 HTTP 协议 (HTTPS) 等。

3) 生命周期事件订阅层

该层接受 Web 容器发来的生命周期事件，注册到该层的监听器收到事件后控制上层服务和容器的生命周期。Web 容器核心和上层服务及容器通过该层进行解耦，同时生命周期层还支持定制和扩展，这种松散耦合的架构使得应用服务器可以灵活扩展新的功能。

4) EJB 容器

提供 JPA 和 EJB 应用的运行环境。

5) 数据源

提供 JDBC 数据源，用于管理数据库连接。

6) 命名服务

提供本地和远程的名字服务。

7) 配置管理

可以管理各个容器和服务的配置信息，并支持实时配置变更。

8) 安全服务

提供基于 Java EE 标准的安全服务。

9) 交易服务

支持 JTA1.2 规范，保证数据和业务逻辑的正确性和完整性。

10) 交易服务

提供日志服务，用户可以对不同的模块设置不同的日志获取粒度，以达到根据需要选择性的获取不同粒度的日志信息或关闭日志信息。

11) 部署服务

提供应用部署功能的核心服务，可以部署各种类型应用，包括 web 应用，ejb 应用，企业应用等。

12) 监视服务

提供监视服务，用户可以控制监视量的采集，并能够获取服务器提供的监控量的值。

1.5 TongWeb7 的特性

1) JavaEE 7 Web Profile 规范

TongWeb7 遵循 JavaEE 7 Web Profile 规范，因此支持 JavaEE 7 Web Profile 规范中的新特性，CDI1.1、EJB3.2、Servlet3.1、JPA2.1、JSF2.2 和 Bean Validation 等。

2) 高可靠性、高伸缩性、灵活扩展的集群

TongWeb7 的集群采用集中式的缓存集群解决方案，提供极高的可靠性，不存在任何单点问题，同时拥有很高的伸缩性；缓存集群可以在运行时支持动态扩展，为整个集群提供灵活的扩展性。

2) 基于 JMX 的管理机制

JMX 技术是 Java 关于应用和资源管理的标准技术，它为开发标准化、集中式的、安全的远程管理应用提供了方案。TongWeb7 采用 JMX 作为管理框架的基础，清晰简洁。

4) 管理工具

TongWeb7 提供三种管理工具，分别是管理控制台和第三方 JMX 工具 JConsole 还有命令行。管理控制台和命令行提供应用组件和资源的管理等功能，JConsole 是基于 JMX 的 GUI 工具，提供 JVM、MBeans 等信息。

5) 调优辅助工具

TongWeb7 提供日志服务，快照服务，监视服务，便于用户解决功能或者性能的问题。

1.6 集成的第三方产品

1) ActiveMQ

默认提供开源的 JMS Server ActiveMQ，用于 JMS 资源。

1.7 规范支持

类型	支持内容
组件	JSP2.3 Servlet3.1 WebSocket1.0 JSF2.2 JSTL1.2 EJB3.2 EL3.0 JCA1.7 Debugging Support for Other Languages 1.0 Common Annotations for the Java Platform 1.2 JPA2.1 Bean Validation 1.1 CDI 1.1 JCA1.7 Dependency Injection for Java 1.0
资源和服务	JTA1.2 JDBC 4.0
协议	HTTP1.1 RMI
安全	JAAS1.0

2 TongWeb7 安装指南

本章节主要介绍在 Windows、Linux、Unix 系统中安装 TongWeb7 应用服务器的注意事项，及在各平台中如何安装 TongWeb7 应用服务器。

2.1 系统配置要求

安装 TongWeb7 应用服务器最低系统要求见以下内容：

系统组件	系统要求
操作系统	Windows 平台： Microsoft Windows 系列 Linux 平台： RedHat 系列 RedFlag 系列 Suse Linux 系列 国产平台： 龙芯系列 飞腾系列 华为系列

	申威系列 海光系列
Java 环境	JDK1.7 以上
物理内存	512MB 及以上
硬盘空间	可用空间 500MB 及以上
监视器	图形界面安装需要 256 色，字符界面安装没有色彩要求
浏览器	Microsoft IE8 或 Firefox3.0 及以上版本浏览器

2.2 安装 TongWeb7 应用服务器

2.2.1 Windows 平台上界面安装

- 1) 运行 TongWeb7 产品光盘中提供的 Install_TW7.0.*.*_*.exe，出现图 2.2.1 所示安装界面：（注：在 Linux 环境下，如果当前平台编码 LANG 不为 zh_CN.UTF-8，则直接跳过安装界面语言选择界面）



图 2.2.1 进入安装界面

- 2) 选择安装界面语言（中文简体/English），点击 OK，出现图 2.2.2 所示界面。

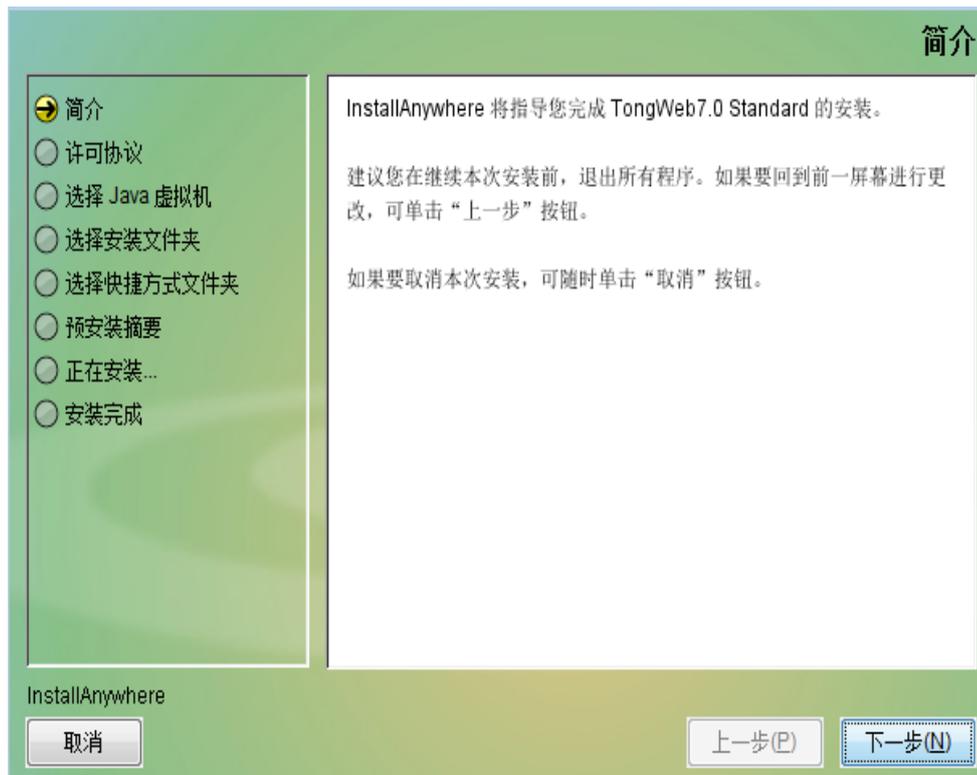


图 2.2.2 windows 安装简介页面

3) 点击“下一步”，进入图 2.2.3 所示“许可协议”界面。

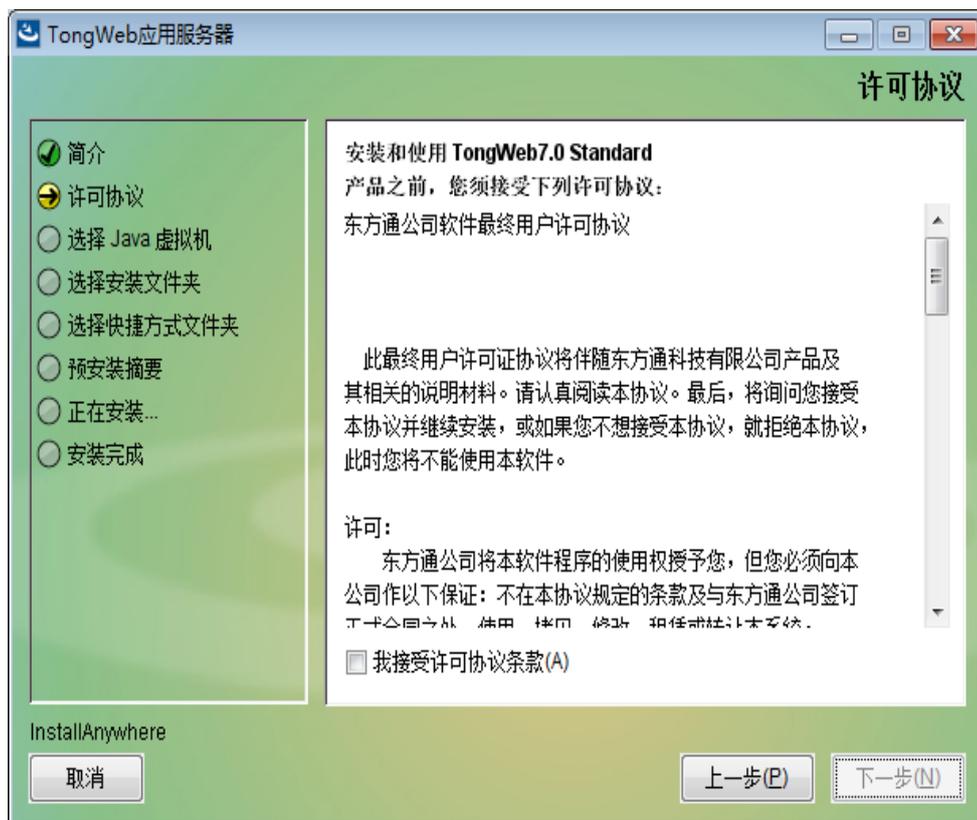


图 2.2.3 许可协议界面

4) 选择“本人接受许可协议条款”后，点击“下一步”，出现图 2.2.4 所示“选择 Java VM”界面。

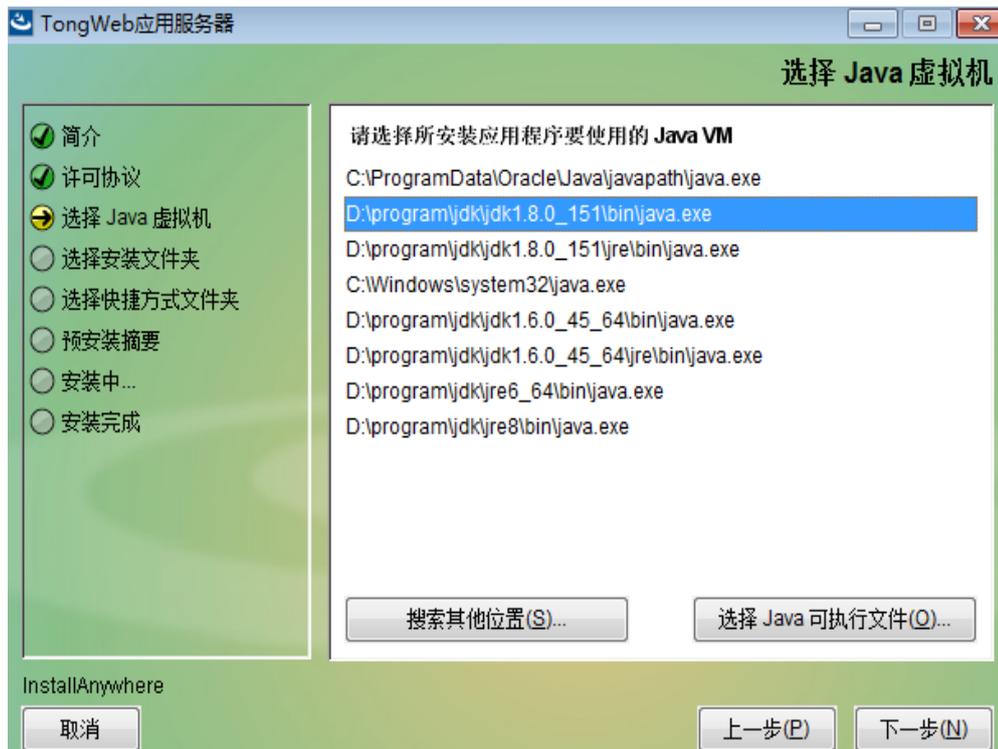


图 2.2.4 选择 Java 虚拟机界面

5) 选择 Java 虚拟机，出现图 2.2.5 所示“选择安装文件夹”界面。

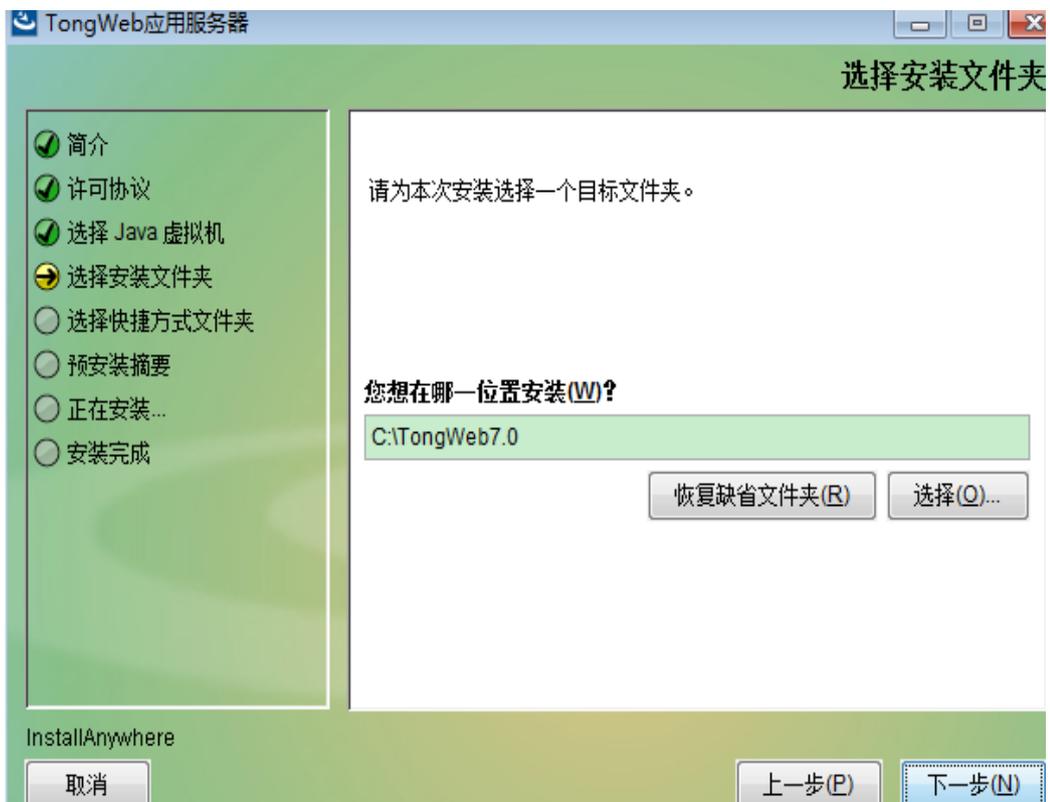


图 2.2.5 选择安装路径

6) 选择安装文件夹后，点击“下一步”，出现图 2.2.6 所示“选择快捷文件夹”界面。

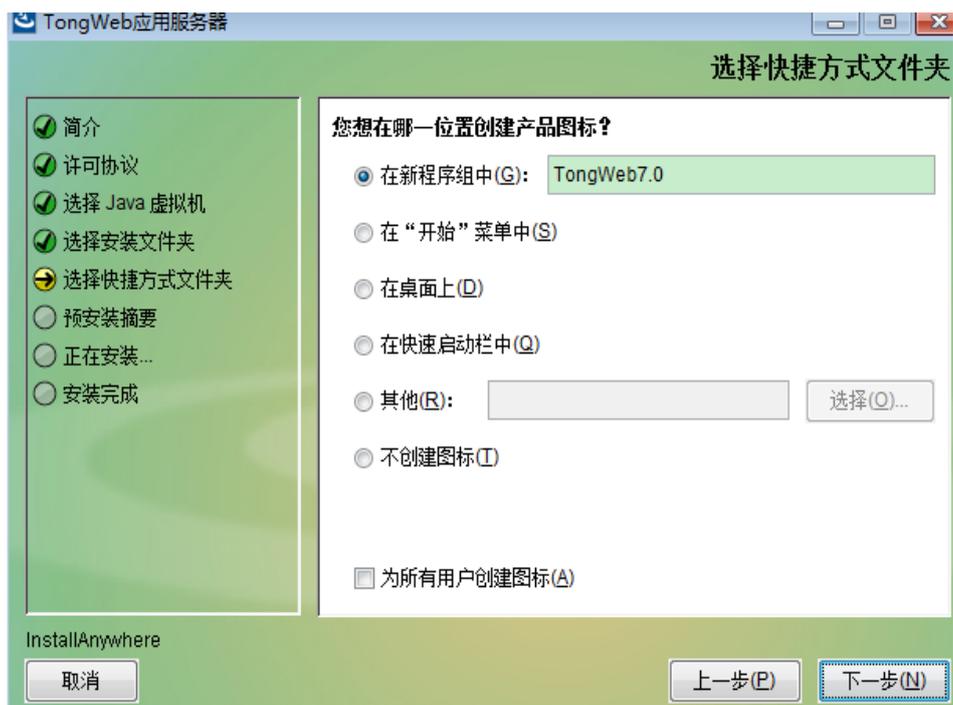


图 2.2.6 选择快捷产品图标

- 7) 选择是否创建快捷图标，点击“下一步”，出现图 2.2.7 所示“预安装摘要”界面。



图 2.2.7 预安装界面

- 8) 查看并审核预安装摘要后，点击“安装”，进入如图 2.2.8 安装界面。



图 2.2.8 安装界面

- 9) 安装滚动条完成后，进入端口设置界面，可以修改 http-listener、jmx-connector、shutdown-port 端口号；也如图 2.2.9 所示，当前界面显示为默认端口。



图 2.2.9 windows 安装界面八

- 10) 设置端口后，点击“下一步”，进入安装完成界面。如图 2.2.10 所示的“安装完毕”界面。

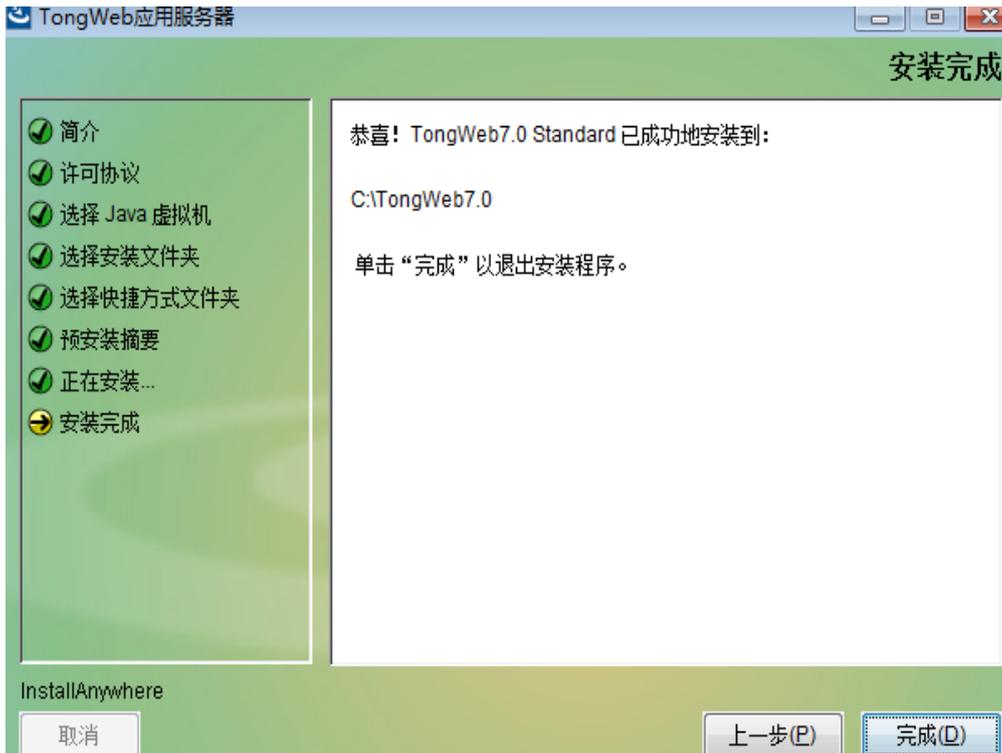


图 2.2.10 windows 安装界面九

11) 出现了成功安装的提示信息后，表示安装成功。点击“完成”退出安装程序。

2.2.2 Linux 平台命令行安装

Linux 平台图形界面安装，直接执行安装程序：`$sh Install_TW7.*.*_*.bin`；安装过程和 Windows 平台一致。如果没有开启图形界面功能，需要通过命令行安装。过程如下：

1) 运行`$sh Install_TW7.*.*_*.bin -i console`，出现如下信息：

```
[root@Config123VM0 mengal]# sh Install_TW7.*.*_Standard_Linux.bin -i
console
Preparing to install...
Extracting the installation resources from the installer archive...
Configuring the installer for this system's environment...

Launching installer...

Preparing CONSOLE Mode Installation...

=====
==
Choose Locale...
-----

1- English
->2- 中文简体
```

CHOOSE LOCALE BY NUMBER: 1

2) 出现如上信息的后，选择安装界面语言，进入下图：（注：如果当前平台编码 LANG 不为 zh_CN.UTF-8）

```
=====
==
TongWeb7.0 Standard                               (created with InstallAnywhere)
-----
--
```

```
=====
==
License Agreement
-----
```

Installation and Use of TongWeb7 Requires Acceptance of the Following License

Agreement:

End user license agreement for Tongtech co., LTD software

The End user license agreement will be accompanied with the products and

related documents of Tongtech co., LTD. Please read it carefully. You will

be asked to accept this license and continue the installation. If you do not accept this license, you should refuse it and quit the installation.

Grant of license:

Tongtech co., LTD grants you the license to use the software program, but you

must make such assurance as following to our company: Do not

use、copy、modify、rent or convey this system besides the terms listed in this

license and the formal contact signed with Tongtech co., LTD.

You guarantee:

1. Using this software only on a single computer;

2. For the purpose of backup or archival management for the use on one computer, making copy of this system by machine-reading format.

You guarantee not:

1. Transfer license of this system again.

2. Getting source codes of this system by altering, modifying,

translating,

reversing, anti-editing, anti-compiling or any other methods.

PRESS <ENTER> TO CONTINUE:

3) 出现如上信息后, 按回车键继续安装。

3. Copy or transfer this software in whole or in part.

When you transfer this software in part or in whole to any third part,
your

right to use the software shall terminate immediately and without notice.

The copyright and ownership of this software:

The copyright of this software is owned by Tongtech co., LTD. The
structures,

tissues and codes are the most valuable commercial secrets of Tongtech co.,
LTD. This software and documents are protected by national copyright laws
and

international treaty provisions. You are not allowed to delete the
copyright

notice from this software. You must agree to prohibit any kind of illegal
copy

of this software and documents.

Limited warranty:

In the largest permitting area of the law, In no situation shall Tongtech
co., LTD be liable for any special, unexpected, direct or indirect damages
(including, without limitation, damages for loss of business profits,
business

interruption, loss of business information, or any other pecuniary loss)
arising out of the use of or inability to use this product and the
providing or

inability to provide supporting services, even if Tongtech co., LTD has
been

advised of the possibility of such damages.

PRESS <ENTER> TO CONTINUE:

4) 出现如上信息后, 按回车键继续安装。

Termination:

Tongtech co., LTD may terminate the license at any time if you violate any
term or condition of the license. When the license is terminated, you must
destroy all copies of the software and all of its documents immediately, or
return them to Tongtech co., LTD.

Law:
"Intelligent Property Protection Regulation", "Copyright Law", "Exclusive Law"

Now, you must have already carefully read and understand this license, and agreed to obey all the terms and conditions strictly.

DO YOU ACCEPT THE TERMS OF THIS LICENSE AGREEMENT? (Y/N): y

- 5) 出现如上信息后, 请选择是否接受许可条款, 若接受请输入 y。进入选择 Java VM, 默认为当前系统正在使用 Java VM。(注意, 只有 windows、linux 和 aix 才自带 JDK, 其它平台不带。)

=====
==

Choose Java Virtual Machine

Please Choose a Java VM for Use by the Installed Application
->1- /usr/java/jdk1.8.0_144/bin/java //当前系统正使用的 JDK
2- /usr/bin/java
3- Choose a Java VM already installed on this system//使用其它 JDK
ENTER THE NUMBER FOR THE JAVA VM, OR PRESS <ENTER> TO ACCEPT THE
CURRENT SELECTION: 3
ENTER THE ABSOLUTE PATH TO THE JAVA VM EXECUTABLE OF YOUR CHOICE
: /home/tong/jdk/jdk1.7.0_67/bin/java
PATH TO THE JAVA EXECUTABLE IS:
/home/tong/jdk/jdk1.7.0_67/bin/java
IS THIS CORRECT? (Y/N): y

- 6) 完成上面步骤后, 按回车, 进入选择安装路径。

=====
==

Choose Install Folder

Where would you like to install?

Default Install Folder: /root/TongWeb7.0

ENTER AN ABSOLUTE PATH, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
:

- 7) 出现如上信息后, 请输入安装路径, 若同意使用给出的默认安装路径, 请按回车键继续安装。

=====
==

Choose Link Location

Where would you like to create links?

- >1- Default: /root
- 2- In your home folder
- 3- Choose another location...

- 4- Don't create links

ENTER THE NUMBER OF AN OPTION ABOVE, OR PRESS <ENTER> TO ACCEPT THE DEFAULT
:

8) 出现如上信息后，请输入链接选项，选择后，按回车键继续安装。

=====
==
Pre-Installation Summary

Please Review the Following Before Continuing:

Product Name:
TongWeb7 Standard

Install Folder:
/root/TongWeb7.0

Link Folder:
/root

Disk Space Information (for Installation Target):
Required: 107,191,640 bytes
Available: 230,416,384 bytes

PRESS <ENTER> TO CONTINUE:

9) 出现如上信息后，请确认预安装信息是否正确，若正确请按回车键继续安装。

=====
==
Installing...

[=====|=====|=====|=====]
[-----|-----|-----|-----]

10) 安装完成后，进入端口修改界面，缺省值为默认端口。

```
=====
=
Set Ports
-----

tong-http-listener (DEFAULT: 8088): 8089
system-http-listener (DEFAULT: 9060): 9061
ejb-server-listener (DEFAULT:5100): 5100
jms-service (DEFAULT:7200): 7201
shutdown-port (DEFAULT:8005): 8005
```

11) 输入端口后，按回车键结束安装。

```
Installation Complete
-----

Congratulations. TongWeb7 has been successfully installed to:

/root/TongWeb7.0

PRESS <ENTER> TO EXIT THE INSTALLER:
```

2.2.3 Linux 平台上静默安装

制作一个 install.properties 的属性配置文件，该配置文件放到安装程序同级目录下，以下是参数解析：

- INSTALL_UI ##这个参数表示的是安装模式，此处介绍的是 silent 模式，则为 INSTALL_UI=silent
- USER_INSTALL_DIR ##这个参数表示的是 TongWeb 的安装路径，例如：
USER_INSTALL_DIR=/home/tong/twns
- SILENT_JDK_HOME ##设置 JDK 路径，该 jdk 路径优先。例如：
SILENT_JDK_HOME=/home/jdk/jdk1.7.0_67
- USER_INPUT_PORTS_RESULTS ##TongWeb 端口配置：

格式为：USER_INPUT_PORTS_RESULTS=

"tong-http-listener", "system-http-listener", "ejb-server-listener ", "jms-service", "shutdown-port ", 例如：

USER_INPUT_PORTS_RESULTS="8081", "9061", "5101", "7201", "8006"

安装命令：

sh 安装程序 -i silent -f install.properties 配置文件，例如：

sh Install_TW7.*.*.*_*.bin -i silent -f install.properties

2.2.4 安装 License

购买 TongWeb7 产品后，在 TongWeb7 产品光盘中有提供有 license 文件。将 TongWeb7 产品光盘中的 license.dat 文件复制到安装完成的 TongWeb7 根目录下。

2.3 TongWeb7 开始向导

2.3.1 TongWeb7 应用服务器目录说明

目录名称	说明
Agent	代理服务器注册节点（标准版不存在该目录）
applications	系统应用所在目录。
asdp	asdp 防御攻击目录（安全版本提供）
autodeploy	服务器默认提供的自动部署监听目录。
bin	服务器启动，停止等脚本文件所在目录。
conf	服务器的配置文件所在目录。
deployment	已部署应用的应用程序目录。
TongDataGrid	分布式缓存目录(标准版不存在该目录)
lib	服务器运行所需的类文件所在目录，主要以 Jar 文件形式存在。
logs	服务器存放日志文件的目录，日志文件包括访问日志文件和服务器日志文件等。
native	Apr native 在不同平台所需要的库文件。
samples	应用服务器的示例目录，示例包括 EJB、WEB 等模块。
service	自启动服务器目录
persistence	存放各类监视量的持久化文件。
snapshot	存放服务器生成的快照文件。
temp	服务器产生的临时文件以及应用预编译文件所在的目录。
tools	应用服务器所带的工具目录

2.3.2 启动服务器

2.3.2.1 在 Windows 上启动服务器

1) 快捷方式启动

TongWeb7 在安装过程中提供有多种快捷方式，通过选择快捷方式启动应用服务器。
例如：开始->所有程序->TongWeb7->Start TongWeb7。

2) 通过命令行启动

TongWeb7 安装成功后，使用`{TW7_HOME}/bin`目录下的 `startserver.bat` 启动应用服务器。

2.3.2.2 在 Linux 平台上启动服务器

在 Linux 平台上，直接在`{TW7_HOME}/bin`目录下，通过 `startserver.sh` 启动应用服务器。也可以通过 `startservernohup.sh` 以后台运行的方式启动应用服务器。

2.3.2.3 安全启动

安全启动功能可以防止通过停止脚本非法停止应用服务器。开启安全启动功能需要在`{TW7_HOME}/bin/external.vmoptions` 脚本中，把 `-Dstartup.secure=false` 改成 `-Dstartup.secure=true`。如果脚本中没有 `-Dstartup.secure` 参数，需要添加该参数；

此时，通过\${TW7_HOME}/bin/stopserver 来停止 TongWeb7 应用服务器时，需要在调用停止脚本时输入登录用户认证信息，用户名、密码同管理控制台用户名、密码一样。如下图所示：

```
thanos@master:~/TongWeb7/bin$ sh stopserver.sh
[root@master bin]# sh stopserver.sh
[2020-07-30 17:17:51 248] [INFO] [main] [core] [input username: ]
thanos
[2020-07-30 17:17:54 190] [INFO] [main] [core] [input password: ]
thanos123.com
[root@master bin]#
```

2.3.3 宕机重启模式

宕机重启模式是 TongWeb7 运行中因为出错宕机（进程异常退出）时，可以自动重新启动 TongWeb7。以宕机重启模式运行的 TongWeb7 共有两个 java 进程：主进程（TongWeb7 重启监控进程）和子进程（TongWeb7 应用服务器进程）。主进程只能监控同目录下启动的 TongWeb7 子进程。

2.3.3.1 宕机重启模式用法

- 用法

`startserver.sh(startserver.bat) restart`

- 宕机重启参数说明

1) `-Ddongweb.restart.interval=1`: 设置宕机后重启的时间间隔，以秒为单位。如果不设置这个参数，默认为 1 秒

- 内存溢出重启参数说明

在 Linux 系统下会提供监测内存溢出以及 HTTP 通道线程挂起数功能。如扫描日志文件发现有内存溢出时，就用 jmap 生成内存转储文件；发现线程挂起数超过配置值，则输出堆栈信息。文件存储路径为 \${TW7_HOME}/snapshot/restart。同时根据配置是否重启服务器。可在启动脚本中通过增加如下参数配置相关信息：

- 1) `-Dmonitor.abnormal.restart=false`: 设置服务器非正常状态时是否重启，如果不设置这个参数或者参数值不为 true，表示不重启，默认为 false。
- 2) `-Dmonitor.interval=10`: 设置监测时间间隔，以秒为单位。如果不设置这个参数，默认为 10 秒。
- 3) `-Dmonitor.hang.max=0`: 设置允许的最大挂起线程数，如果不设置这个参数，默认为-1, 表示不监测线程挂起数。设置为 0 表示，只要发现有一个挂起则输出堆栈。
- 4) `-Dmonitor.hang.name=tong-http-listene`: 设置监测某个 HTTP 通道名称下的线程挂起数，如果不设置这个参数，默认为空，表示监测所有的 HTTP 通道线程挂起数。
- 5) `-Dmonitor.hang.name=tong-http-listene`: 设置监测某个 HTTP 通道名称下的线程挂起数，如果不设置这个参数，默认为空，表示监测所有的 HTTP 通道线程挂起数。

2.3.3.2 宕机重启使用场景

只有服务器子进程发生异常宕机时监控主进程才会重启 TongWeb7 应用服务器。

- 异常宕机

- 1) Kill -9 子进程 id: 认为是 TongWeb7 服务器宕机, 重启应用服务器;
- 2) Kill -2 子进程 id: 认为是 TongWeb7 服务器宕机, 重启应用服务器;
- 3) Kill -15 子进程 id: 认为是 TongWeb7 服务器宕机, 重启应用服务器;
- 4) 其它方式: 可以通过监控主进程的监控日志或控制台日志可以看到应用服务器子进程是如何退出的, 退出码为 0 的为正常退出, 其他都属于异常退出。例如下图中, 应用服务器子进程因为异常退出 (退出码 137) 而重启了。

```

2020-07-31 09:02:47 [INFO] find current readLine:208,date:2020-07-31 09:02:46
2020-07-31 09:02:57 [INFO] MemMonitorThread is set to stop
2020-07-31 09:02:57 [INFO] TongWeb server process exit (exitcode=130).
2020-07-31 09:02:57 [INFO] MemMonitorThread is stopped
2020-07-31 09:02:58 [INFO] ----- Start TongWeb ----- number = 2
2020-07-31 09:03:03 [info] [2020-07-31 09:03:03 669] [INFO] [main] [systemout] [License e
xpires 2020-09-30]
2020-07-31 09:03:03 [info] [2020-07-31 09:03:03 674] [WARNING] [main] [core] [License信息
:客户名称=测试用户;项目名称=测试项目;到期时间=2020-09-30]
2020-07-31 09:03:06 [info] [2020-07-31 09:03:06 673] [INFO] [main] [lca] [Creating JCA th

```

● 不属于异常宕机:

- 1) 通过\${TW7_HOME}/bin的 stopserver.bat (stopserver.sh), 正常停止服务;
- 2) 2) ctrl+c 停止了重启监控主进程(通过操作系统查看进程命令可看字符串 com.tongweb.launcher.monitor.LauncherMonitor), 这会导致同时停止了监控主进程和应用服务器子进程;
- 3) kill -3 子进程 PID: 输出子进程的 jvm 堆栈信息, 输入到 server.log 日志中, 应用服务器子进程不会宕机;
- 4) kill -3 主进程 PID: 输出主进程的 jvm 堆栈信息, 服务器主进程不会宕机;

2.3.4 管理控制台

TongWeb7 管理控制台是应用服务器提供的图形管理工具, 它允许系统管理员以 Web 方式管理系统服务、应用程序、监控系统信息等。

2.3.4.1 登录

在浏览器中输入 <http://IP:9060/console> 进行访问, 访问成功后出现如图 2.3.1 所示登陆页面, 默认登录用户名密码为: thanos/thanos123.com。



图 2.3.1 登录页面

注意: 如果管理控制台已被某用户登录, 使用该用户登陆时会出现如下图的提示信息:



图 2.3.2 登录冲突

2.3.4.2 注销

TongWeb7 管理控制台中提供用户退出登录的功能，在管理控制台右上角用户名下的选择“退出登录”按钮，便可退出当前用户的登录。

2.3.4.3 导航栏

管理控制台左侧导航树上展示了控制台提供的所有功能，如图 2.3.3 所示：



2.3.3 所有任务

2.3.4.4 首页

登录成功以后，进入应用服务器控制台首页界面，显示 TongWeb7 安装信息、JDK 及

License 信息, 如下图 2.3.4 所示:

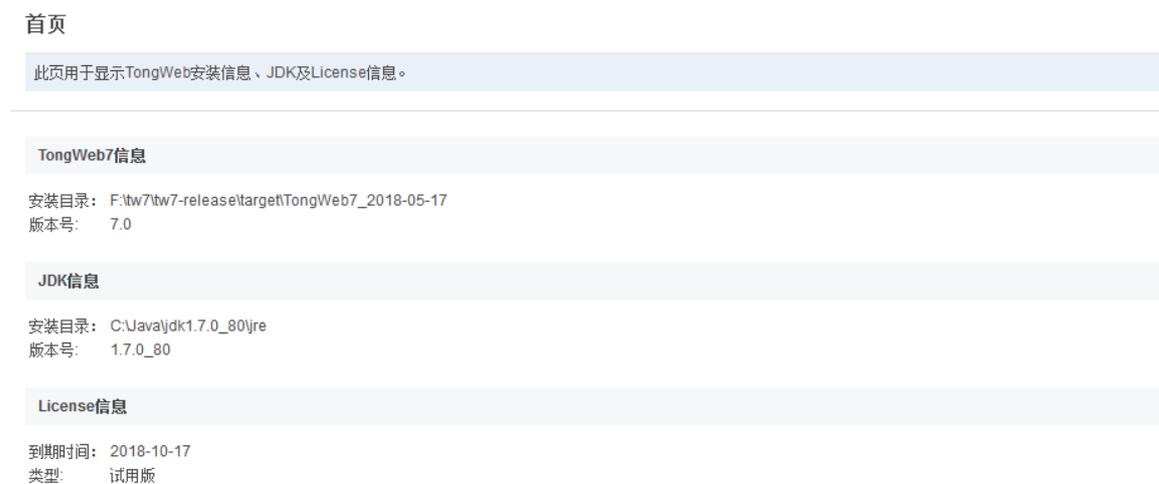


图 2.3.5 首页

在右上角界面, 提供了帮助、中英文切换功能。点击“查看帮助”, 进入文档查看界面; 点击“中文”, 可以进行选择“English”或“中文”, 如图 2.3.6 所示:

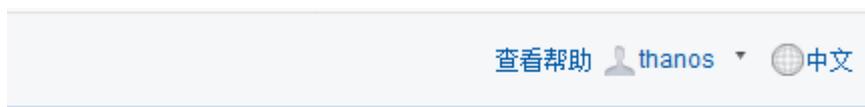


图 2.3.6

2.3.5 JConsole

JConsole 是第三方的 JMX 工具, 它是基于 JMX 的 GUI 工具, 提供 JVM、MBeans 等信息。具体使用方法如下:

- 1) 查看 JMX URL: 启动应用服务后, 可在日志文件中查找 ($\{\text{TW7_HOME}\}/\text{logs}/\text{server.log}$)。如 URL for the Standard JMXConnectorServer: [service:jmx:rmi:///jndi/rmi://testserver:7200/jmxrmi]. 说明: 中括号中的信息即为连接服务器的 JMX URL, 其中 testserver 为服务器所在机器的主机名或 IP 地址(本地或者远程)。
- 2) 启动 JConsole: 通过运行 $\text{JDK_HOME}/\text{bin}/\text{jconsole}$ 来启动 JConsole。
- 3) 连接到正在运行的服务器, 登录用户名密码与控制台登录用户相同。



图 2.3.5 JConsole 登录界面

4) 点击链接，进入 jconsole 管理控制台界面，如下图 2.3.6 所示：

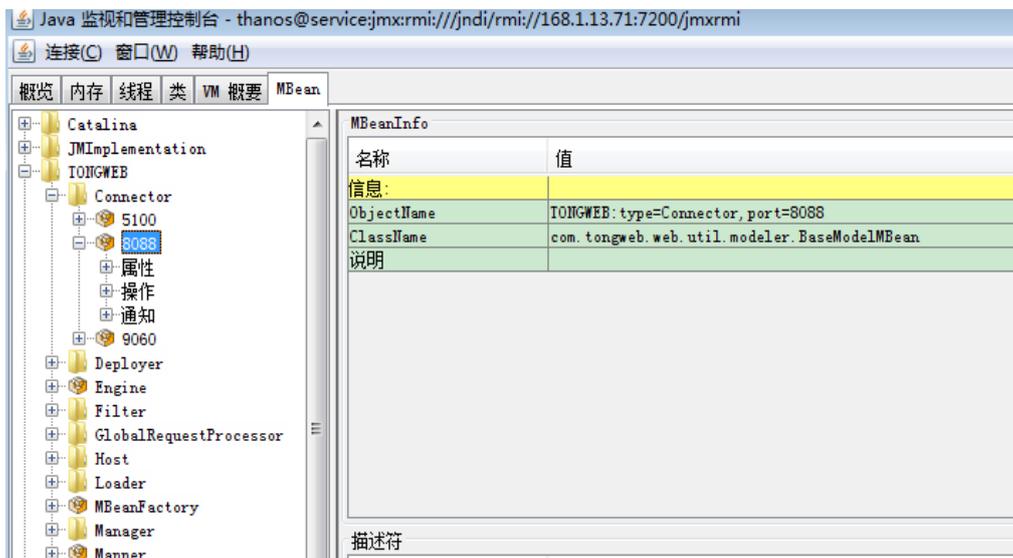


图 2.3.6 MBean 界面

2.3.6 停止服务器

2.3.6.1 Windows

TongWeb7 提供几种方式停止服务器，分别是使用 Ctrl+C 强行停止服务器、通过停止脚本停止服务器。

1. 使用 Ctrl+C 停止

在 TongWeb7 的运行窗口直接按下 Ctrl+C，即可终止 TongWeb7 的运行。

2. 通过停止脚本停止

TongWeb7 安装成功后，使用 TW_HOME/bin 目录下的 stopserver.bat 停止 TongWeb7 应用服务器。

2.3.6.2 Unix/Linux

1. 使用 Ctrl+C 停止

在 TongWeb7 的运行窗口直接按下 Ctrl+C，即可终止 TongWeb7 的运行。

2. 通过停止脚本停止

在 TW_HOME/bin 下运行 stopserver.sh 以停止 TongWeb7 应用服务器。

2.4 卸载 TongWeb7 应用服务器

2.4.1 Windows 平台上卸载

- 1) 通过选择快捷方式卸载 TongWeb7 应用服务器。例如：开始->所有程序->TongWeb7->Uninstall TongWeb7，执行卸载程序后，出现如图 2.4.1 所示的“卸载界面”。



图 2.4.1 进入卸载界面

- 2) 确认卸载后，点击“卸载”。卸载完成后，出现如图 2.4.2 的“卸载完毕”界面。

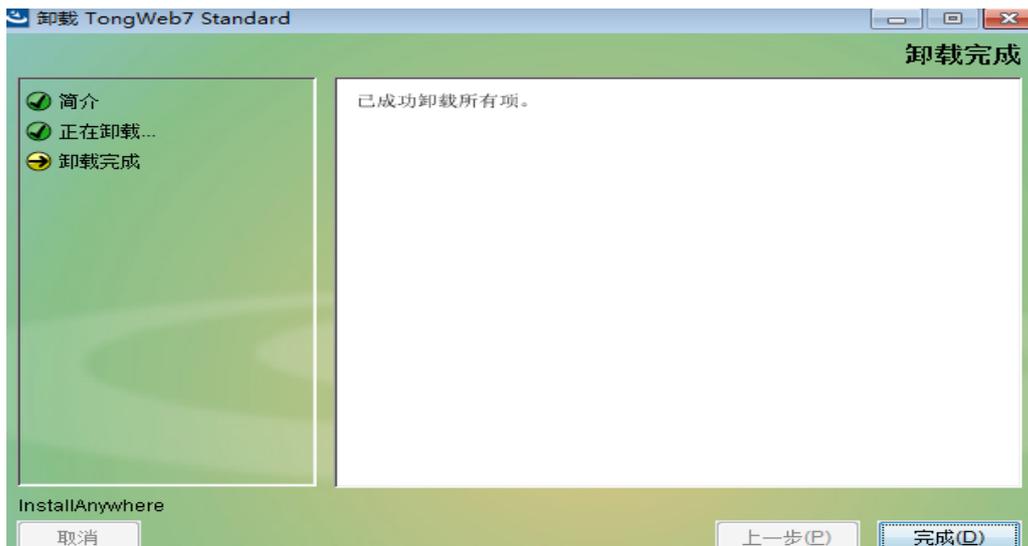


图 2.4.2 卸载完成

- 3) 确认信息后，点击“完成”退出卸载程序。

说明：如果文件在使用过程中被修改或者新生成的文件夹，则卸载时认为无法删除。

2.4.2 Linux 平台上卸载

直接删除安装目录。

3 应用管理

3.1 应用管理概述

TongWeb7 提供应用的管理。从应用类型上分为：web 应用，ejb 应用，企业应用这三种应用类型；从应用部署的方式上可以分为：目录部署，文件方式部署；从应用的管理方式分为：命令行部署，热部署，自动部署，管理控制台部署等；在应用部署的时候，可以在管理控制台中添加一些附加的属性，如部署描述信息，虚拟主机等部署附加属性部署；在应用中可以通过配置 TongWeb7 的自定义部署描述文件，配合 JAVA EE 应用中标准描述文件的 web.xml 和 ejb-jar.xml 等文件，更进一步的自定义应用。

3.1.1 应用类型支持

TongWeb7 支持 Java EE 应用文件的类型如下表，见表 3-1-1。

类型	扩展名	用途和构成
Web 应用	.war	包含 Servlet 和 JSP 等 Web 组件，EJB 组件以及静态 HTML 页面、Jar 文件、标记库等
EJB 应用	.jar	包含 EJB 实现以及 EJB 实现所需的类
连接器应用	.rar	包含连接器(资源适配器)的实现类
企业应用	.ear	包含上述三种应用类型
其它	任意	支持任意后缀(指标准格式 jar war ear rar 之外的后缀，如 car)，上传或者选择应用，如果后缀为标准格式则直接判断后缀为应用类型，其他则通过后台获取。

3.1.2 应用部署方式

TongWeb7 支持两种部署方式：文件方式部署和目录方式部署。

1) 文件方式

文件方式部署即应用以应用包（如*.war，*.ear 等）的方式进行部署，该方式支持所有类型的应用。部署后的信息存放在 TW_HOME/deployment 下的目录中。

2) 目录方式

目录部署即应用以展开的目录方式进行部署，该方式支持 war，jar，ear，rar 等各种类型的应用。目录部署的优点是方便应用的修改，当应用包含了需要频繁修改的文件时，使用目录方式部署相对方便。缺点是需要目录部署的应用需与服务器在同一台机器上。

说明：企业应用在目录部署时，如果子模块中包含 WEB 应用，需要将其解压为以.war 或 _war 结尾的目录，否则不识别该子模块。子模块中的 EJB 应用可以不解压，如果也要解压，其目录需要以.jar 或 _jar 结尾。

3.1.3 应用管理方式

TongWeb7 的应用管理，一共分为 4 种方式：

1) 控制台管理应用

可以通过基于 WEB 的管理控制台，通过浏览器终端的操作，上传应用并将应用进行部署，在部署完成之后，可以对已经部署的应用进行启动，停止，和解部署等相应的管

理。

2) 自动扫描

自动扫描为应用管理的一种形式，分为自动部署和热部署。

可以通过将应用放入 TongWeb 的自动部署目录中，实现应用自动部署。TongWeb 的自动部署目录默认位于 TW_HOME/autodeploy 目录下，也可以通过配置重新指定为其它目录。

热部署是当热部署开关开启了之后，如果修改了应用部署目录下的文件，那么 TongWeb 将会对应用进行热部署；热部署需要注意的是，热部署的流程与重部署的流程不相同，重部署相当于重新销毁之后再创建，而热部署是对应用重新加载，其流程不一样。

3) 命令行管理

可以通过应用服务器的命令行工具，对应用进行本地的部署和远程的部署，命令还包括启动，停止，更新等。

4) 接口管理

TongWeb7 应用服务器提供了 JMX 和 RestFul 两种类型的接口，两类接口中都有应用管理的接口，可以通过开发程序调用接口进行应用管理。

3.1.4 应用形态结构

3.1.4.1 Web 应用结构

web 应用由动态资源（如 JSP、Servlet、JSP Tag 库等）和静态资源（如 HTML 页面和图片文件等）组成。具体 Web 应用的结构图如下：

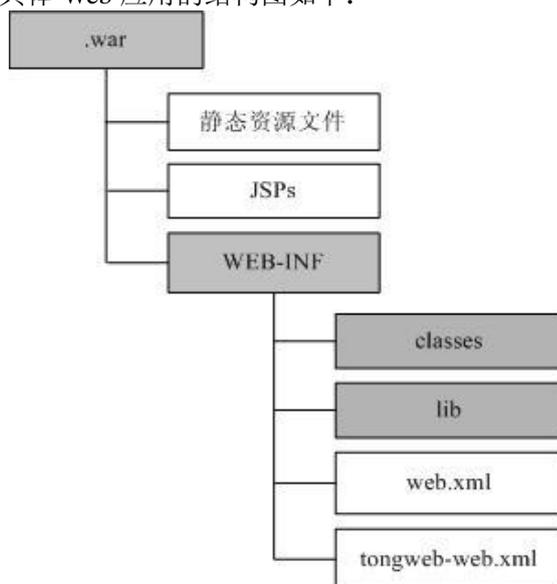


图 3.1.1 Web 应用的结构图

说明：classes 目录下存放的是 web 应用所需的类和资源，lib 目录下存放的是 web 应用所需的类包。web.xml 是 J2EE 标准的部署描述文件，tongweb-web.xml 是 TongWeb7 自定义的部署描述文件。

从 JAVA EE6 开始，由于应用的配置可以通过原注释的方式写到类中，因此可以允许没有 web.xml 文件，同样对于 tongweb-web.xml 也是一样。且 EJB 首次可以编写到 web 应用中，如图 3.1.2 Web 应用，test.war，是一个典型的例子：

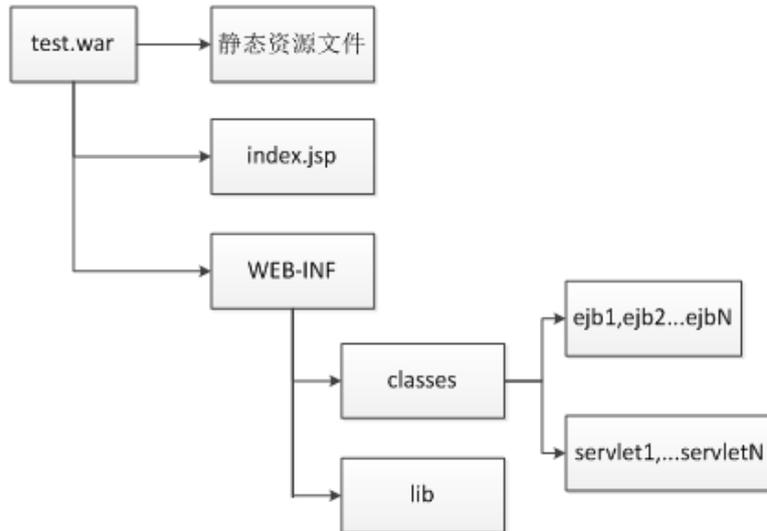


图 3.1.2 Web 应用示例

上述的实例 test.war 中，classes 中可以编写传统的 servlet，也可以有 ejb，而对于 JAVA EE5 等以前版本的传统部署描述文件 web.xml 已经省略掉，JAVA EE 6 以后就通过原注释定义在 servlet 类中。

3.1.4.2 EJB 应用结构

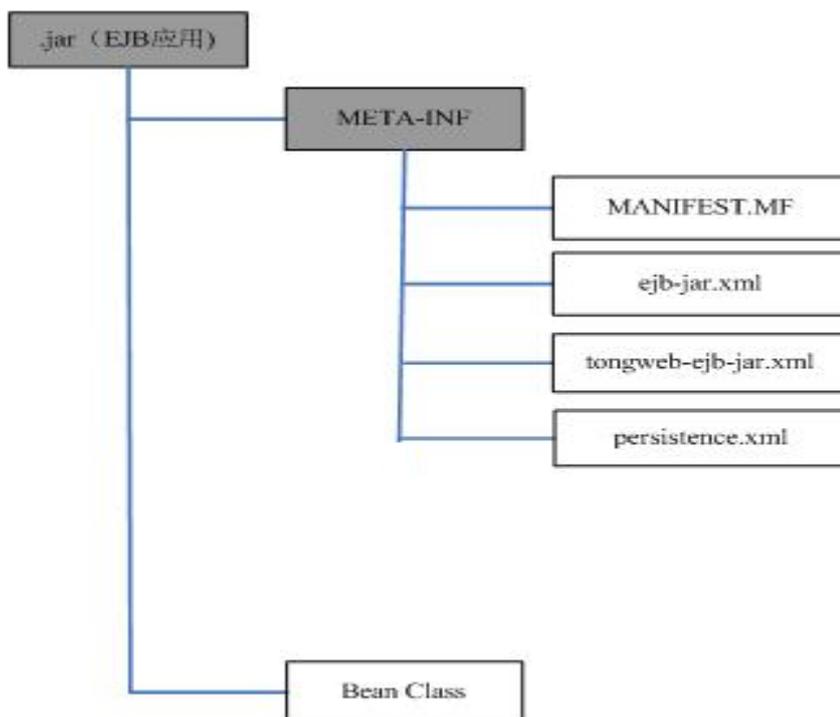


图 3.1.3 EJB 应用的结构

说明：ejb-jar.xml 是 JavaEE 标准部署描述文件，tongweb-ejb-jar.xml 是 TongWeb7 自定义的部署描述文件，persistence.xml 是 Entity 使用 JPA 时所需的持久化配置文件。和 web 应用结构同样类似，JAVA EE6 以后 ejb 的 jar 文件的配置定义完全可以通过原注释的方式，定义在 ejb 的类中，如图 3.1.4 的 test.jar 所示：

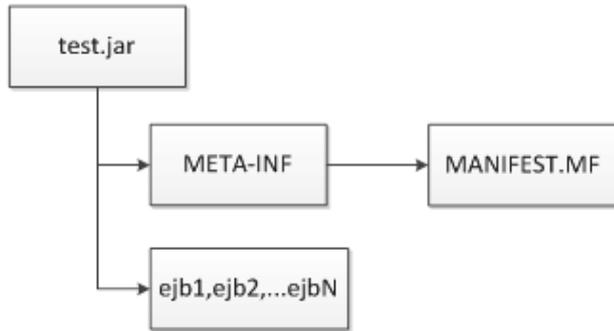


图 3.1.4 EJB 应用示例

test.jar 中是一个典型的 EJB3.1 规范的 ejb 的 jar 包，其 ejb-jar.xml 已经不再是必选的部分。

3.1.4.3 EAR 应用结构

企业应用以 EAR 包的形式封装应用，EAR 内可以包含 Web 应用（WAR）、EJB 应用（EJB JAR）、连接器应用（RAR）和公用包。一个企业应用可包含一个或多个 Java EE 应用或者组件。

具体企业应用的结构如下：

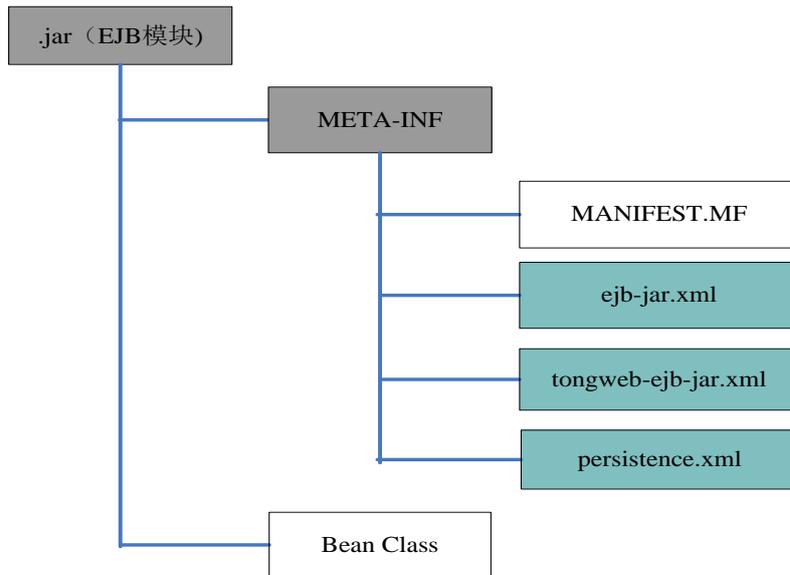


图 3.1.5 企业应用的结构

说明：application.xml 是 JavaEE 标准的部署描述文件。APP-INF 下存在的是企业应用的公共类。

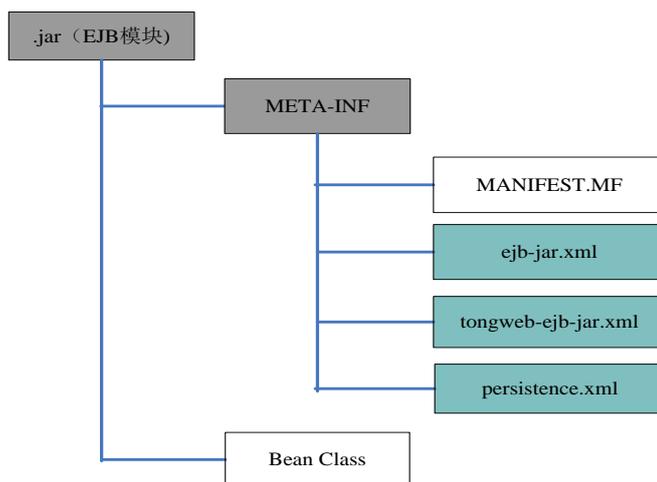


图 3.1.6 企业应用的示例

图 3.1.6 中所示的 test.ear 是典型的企业级应用的示例。

3.1.4.4 RAR 应用结构

Connector 应用即资源适配器，它主要由符合 JCA 规范的 Connector 应用类(jar 包形式)和 JavaEE 标准的部署描述文件 ra.xml 组成。

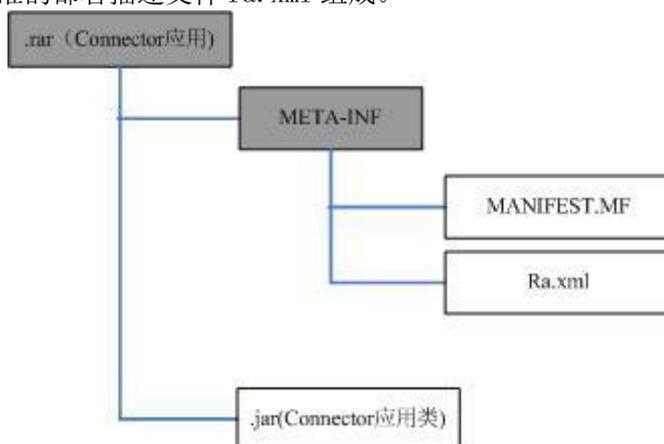


图 3.1.7 Connector 应用的结构

3.1.5 应用部署附加属性

在 TongWeb7 的部署过程中，可以设置应用部署的附加的属性：应用的名称，应用前缀，部署顺序，JSP 预编译，类加载顺序，应用描述等。上述的属性见图 3.1.8。



图 3.1.8 应用部署

3.1.5.1 应用名称

默认情况下，应用名称就是部署时选择的应用文件或者目录名，这个名称在部署阶段可以修改。

3.1.5.2 应用前缀

应用前缀的附加属性，对应的是 web 应用。

每个 Web 应用对应一个访问前缀，当访问具体 Web 应用时作为“应用名”出现在请求 URI 中。例如，如果部署 Web 应用时指定了访问前缀为“test”，则对该应用下 JSP 页面和 Servlet 的请求必须是如下形式：`http(或 https)://服务器 ip: 监听端口 /test/.....`。如果访问前缀为“/”表示访问前缀为空。

每个企业应用 (EAR) 中 Web 模块 (WAR) 的访问前缀，由 Web 应用的自定义部署描述文件 (`tongweb-web.xml`) 中的属性 “`context-root`” 决定，或者由 `META-INF/application.xml` 中 “`web-uri`” 决定（应用前缀在 `application.xml` 中优先级高于 `tongweb-web.xml`），如果没有上述相应的配置，则使用 Web 应用包名（文件部署方式，不含 `.war` 后缀）或目录名（目录部署方式）作为访问前缀。在企业应用中的每一个 Web 应用都要有一个唯一的访问前缀，任何两个 Web 应用不能有相同的访问前缀。

3.1.5.3 JSP 预编译

在管理控制台部署的时候，可以设置当前的应用 JSP 预编译，设置完成之后，对于 Web 应用和 EAR 应用中的 Web 模块，都会先调用应用服务器内置的 JSPC 工具编译，应用部署完成后，JSP 不会在请求访问的时候再进行编译了。

这个属性的作用是提高 JSP 的请求访问的速度，用在含有大量的 JSP 页面的生产模式中。

3.1.5.4 共享库

当应用类型为 War、Ear 时，则可以给当前应用配置共享库，支持多选，选择共享库之后，可以手动检测出所选共享库中的多版本以及冗余类，同时可以排除一些加载包，这样就能有效的避免应用发生某些运行时错误，共享库如果引用的是文件夹，则会加载文件夹下面的所有 JAR 包进行检测。如果未排除任何冗余或者多版本，应用将加载所选共享库下的全部加载包（按路径去除重复）。当应用类型为 Ear 时，配置的共享库引用会作用于该应用的所有子模块。

共享库需要预先配置，请参见具体章节 [4.6 资源](#)。具体操作步骤如下：

1. 当应用部署类型为 War、Ear，在部署时，可以选择共享库。共享库中分为私有库和公有库，如果选择公有库，公有库为所有应用共享的，所在不能对其进行排除。如果只有这个应用自己加载，选择私有库。如下图：

1 基本属性

2 虚拟主机设置

应用名称	<input type="text" value="classloaderTestParno"/>	✓
应用前缀	<input type="text" value="/classloaderTestParno"/>	应用前缀
部署顺序	<input type="text" value="100"/>	默认的部署顺序是 100，如果需要调整部署顺序的话，可以指定
JSP预编译	<input type="checkbox"/> 支持	JSP预编译
共享库	请选择	应用运行时使用的共享类库 多版本或冗余类检测
类加载顺序	<input checked="" type="checkbox"/> 全选 <input type="checkbox"/> 全不选	类加载顺序，默认为子优先，可调整
描述	<input type="checkbox"/> private <input type="checkbox"/> 公有 <input type="checkbox"/> twlibs	该应用的描述信息

2. 选择完所共享的库后，点击后面的“多版本或冗余类检测”，进入类检测界面，如下图，如果不需要的类点击后面的“排除”，都选择好以后点击“确定”，对应用进行部署。

多版本或冗余类检测						
多版本(0)		冗余类(3)		已排除(0)		运行时(3)
来源	类型	名称	版本	路径	操作	
com.mengal.test.SimpleBean						
private1	私有	--	--	/home/zhuhq/tw7/common/dir1/HelloWorldJar.jar	排除	
private2	私有	--	--	/home/zhuhq/tw7/common/dir2/HelloWorldJar2.jar	依赖	
com.mengal.test.HelloWorldBeanLocal						
private1	私有	--	--	/home/zhuhq/tw7/common/dir1/HelloWorldJar.jar	排除	
private2	私有	--	--	/home/zhuhq/tw7/common/dir2/HelloWorldJar2.jar	依赖	
com.mengal.test.HelloWorldBean						
private1	私有	--	--	/home/zhuhq/tw7/common/dir1/HelloWorldJar.jar	排除	
private2	私有	--	--	/home/zhuhq/tw7/common/dir2/HelloWorldJar2.jar	依赖	

多版本：名称一样，版本不同，名称和版本取值于加载包中的 META-INF/MANIFEST.MF 内容。

冗余类：同一个类(包名+类名)在多个加载包中存在，请参见具体章节 [11.2.1 冗余](#)。

已排除：表示排除的多版本或者冗余类，新的排除项需要点击重新检测才能显示，已排除的包支持重新依赖。不能排除应用依赖以及公有库依赖包，只能排除私有库里面的包。

运行时：显示当前应用运行时加载的依赖包，不能执行操作，新的排除项需要点击重新检测才能显示。

重新检测：排除某些多版本或者冗余类加载包之后，点击重新检测，系统根据当前排除项重新计算多版本或者冗余，每个类别上都有数字提示当前类别的数量，可根据数量直观的判断出是否存在多版本或者冗余类。

3.1.5.5 类加载顺序

类加载顺序是指定 web 应用、企业应用中 web 模块和 ejb 模块的类加载顺序，分为父优先和子优先两种。对于这两种加载顺序的说明，请参见具体章节 [4.6 类加载](#)。

注：通过控制台部署企业应用时，为企业应用中所有的 web 模块指定同样的类加载顺序，编辑时可以分别为每个 web 模块设置各自的类加载顺序。默认为子优先。

控制台指定的类加载顺序优先于应用 tongweb-web.xml 部署描述文件指定的类加载顺序，且 tongweb-web.xml 中的类加载顺序优先其它配置文件的类加载顺序。部署描述文件格式内容具体参见章节 [tongweb-web.xml](#)。

3.1.5.6 静态资源缓存最大值

用于应用服务器中 html, css, js 与 image 等静态资源文件的存储。以 KB 为单位，默认值为 10240，具体使用参考 [4.3.2.1](#) 章节

3.1.5.7 应用描述

对于每一个应用可以通过控制台部署时可以设置应用描述，应用描述存储在 TW_HOME/conf/tongweb.xml 配置文件中

3.1.5.8 虚拟主机

虚拟主机是对于 Web 应用与含有 Web 模块的 EAR 应用来说的，同一个应用可以部署到多个虚拟主机上，通过在 DNS 配置的主机名称在浏览器中进行指定的访问。

在通过管理控制台的应用部署中，可以在第二个步骤中，设置应用的虚拟主机，如图 3.1.9 所示。



图 3.1.9 虚拟主机设置

3.1.6 应用自定义部署描述文件

应用中可以配置自定义部署描述文件，在 web 应用中可以配置 tongweb-web.xml，在含有 ejb 的应用中可以配置 tongweb-ejb-jar.xml。

上述的两个配置文件的细节见 [tongweb-web.xml](#) 和 [tongweb-ejb-jar.xml](#) 章节。

3.2 管理控制台应用管理

应用管理最方便的途径是通过管理控制台来进行应用管理，包括内容为：查看已经部署的应用；部署应用；查看 EAR 应用的子模块；应用的解部署、重部署、编辑、访问；应用的启动和停止。

3.2.1 查看已部署的应用

查看已经部署的应用步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，见图 3.2.1。



图 3.2.1 应用列表

在图 3.2.1 中，所有类型的应用都在这一个列表中进行展示。在列表中的字段为：

- 名称：应用的名称。
- 前缀：Web 应用才有的应用前缀，Web 应用和 EAR 应用都有访问链接。
- 应用类型：war, ejb, ear, rar
- 部署源类型：文件部署，目录部署
- 部署方式：控制台部署，自动部署
- 虚拟主机：Web 应用和 EAR 应用才会有，可以将应用部署在多个虚拟主机中
- 状态：正在部署，启动中，停止中，已启动，已停止，错误

操作：访问（分 http 访问和 https 访问，点击可以访问应用，若应用部署的虚拟主机所关联的通道均为空，则该超链接为灰色不可用），重部署

3.2.2 应用部署

查看完已经部署的应用，可以在控制台中点击“部署应用”的蓝色按钮，进行部署的操作，步骤如下：

- 依次展开管理控制台左侧导航树中的“应用管理”节点；
- 点击应用管理左上角的“部署应用”按钮；
- 展现上传页面，如图 3.2.2 所示。

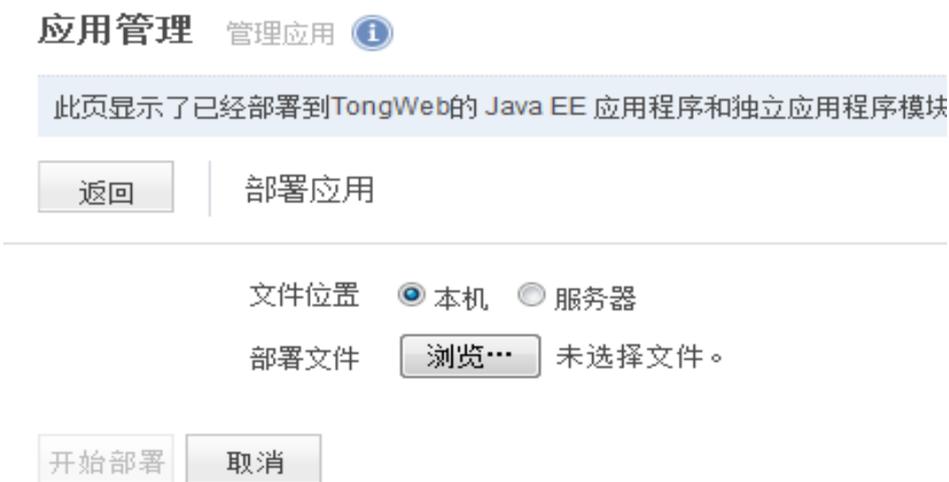


图 3.2.2 上传页面

- 文件位置分为本机和服务器，对于本机来说，即是客户端的机器，其客户端并不一定和服务器在同一台机器上，点击选择文件，将文件上传到服务器上，如图 3.2.3。

应用管理 管理应用 ?

此页显示了已经部署到TongWeb的Java EE 应用程序和独立应用程序模块列表。可以对所安装的应用程序和模

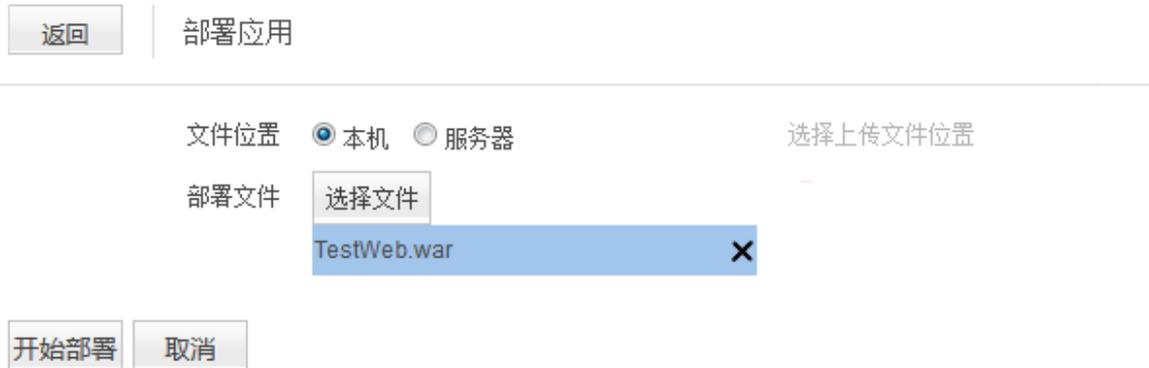


图 3.2.3 上传应用成功

4. 如果不想使用本机的应用进行部署，可以选择服务器上的对应的文件或者目录进行部署，那么如图 3.2.4，点击选中的文件，直接进行服务器文件部署。

应用管理 管理应用 ?

此页显示了已经部署到TongWeb的 Java EE 应用程序和独立应用程序模块列表。可以对所安装



图 3.2.4 服务器部署

5. 无论是本机远程部署应用，还是在服务端直接部署，前面的几个步骤已经准备就绪，点击开始部署以后，进入第一步填写应用部署的附加属性的页面，见图 3.2.5。

返回 | 部署应用--TestWeb

1 基本属性

2 虚拟主机设置

应用名称	<input type="text" value="TestWeb"/>	✔	
应用前缀	<input type="text" value="/TestWeb"/>		应用前缀
部署顺序	<input type="text" value="100"/>		默认的部署顺序是 100，如果需要调整部署顺序的话，可以指定
JSP预编译	<input type="checkbox"/> 支持		JSP预编译
共享库	<input type="button" value="请选择"/>		应用运行时使用的共享类库 多版本或冗余类检测
类加载顺序	<input type="radio"/> 父优先 <input checked="" type="radio"/> 子优先		类加载顺序，默认为子优先，可调整
描述	<input style="width: 100%;" type="text"/>		该应用的描述信息

图 3.2.5 基本属性

6. 根据 [3.1.5 应用部署附加属性](#) 中的应用部署附加属性，填写完应用的附加属性后，点击下一步，进入应用部署的第二步虚拟主机设置页面（不包括 EJB 应用），如图 3.2.6。

应用管理 管理你的应用 ⓘ

此页显示了已经部署到TongWeb的Java EE 应用程序和独立应用程序模块列表和模块进行启动、停止、更新(重新部署)、删除操作或者部署新的应用或模块

返回 | 部署应用--testweb

1 基本属性

2 虚拟主机设置

从列表中选择应用的虚拟主机

或者跳过剩下步骤，直接 [完成](#)

©2012-东方通科技股份有限公司 条款 |

图 3.2.6 虚拟主机设置

7. 选择完成虚拟主机，最后一步将前面填写的信息进行汇总，图如 3.2.7。



图 3.2.7 部署信息展示

8. 最后点击蓝色的“完成”按钮，页面中弹出一个蒙版，在蒙版的中间可以实时的看到部署的进度信息；应用部署完成后跳转到应用列表页面。如图 3.2.8。

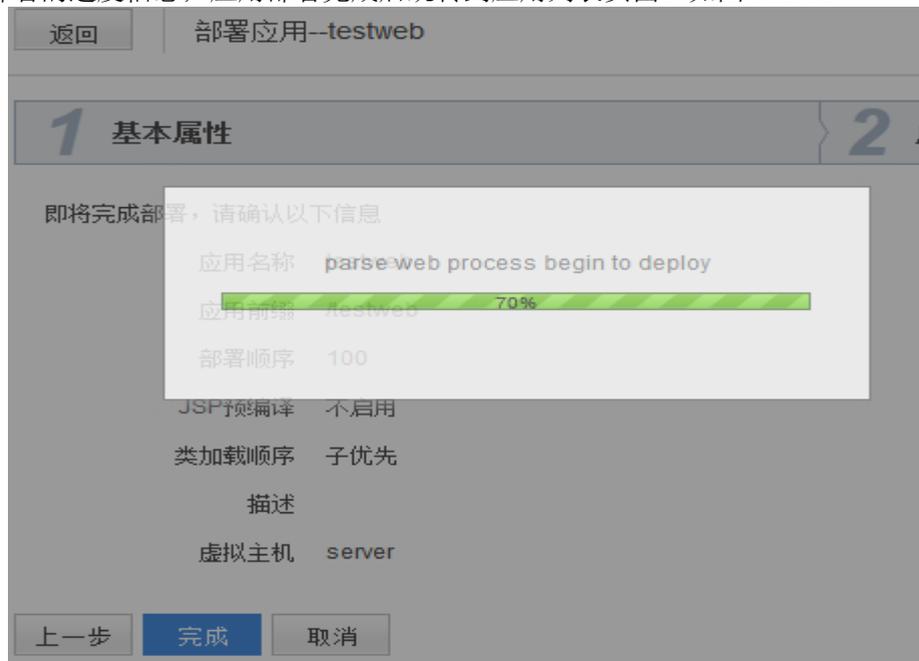


图 3.2.8 部署进度展示

9. 在实时进度不断变化的过程中，若出现图 3.2.9 中所示，则为应用部署出错了，页面跳转到应用列表页面，在列表页面的顶端部分出现应用部署失败的提示框以及有了解详情的链接（应用部署失败提示框背景色为红色），单击“了解详情”链接，弹出日志文件下载框，该附件是应用部署详细异常的压缩包（压缩包名称及包内日志文件名称以应用名称命名），打开后会显示完整的应用部署堆栈，用以分析本次部署失败的原因，如图 3.2.10 所示；若出现图 3.2.11 中所示，在应用部署列表页面上端部分出现应用部署成功的提示框，说明应用部署成功。（应用部署成功提示框背景色为绿色）。



图 3.2.9 部署失败提示框

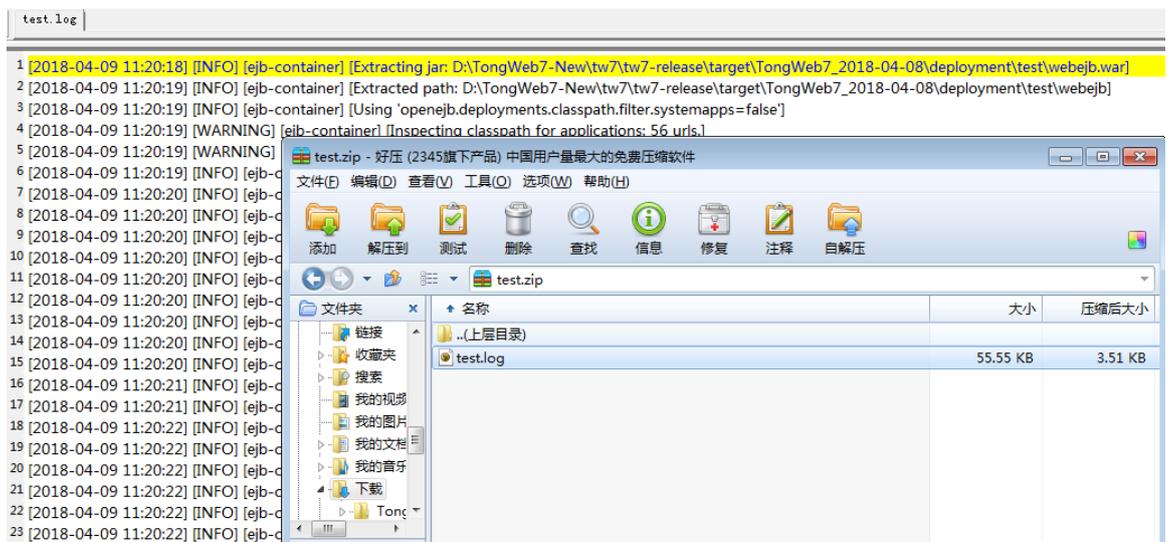
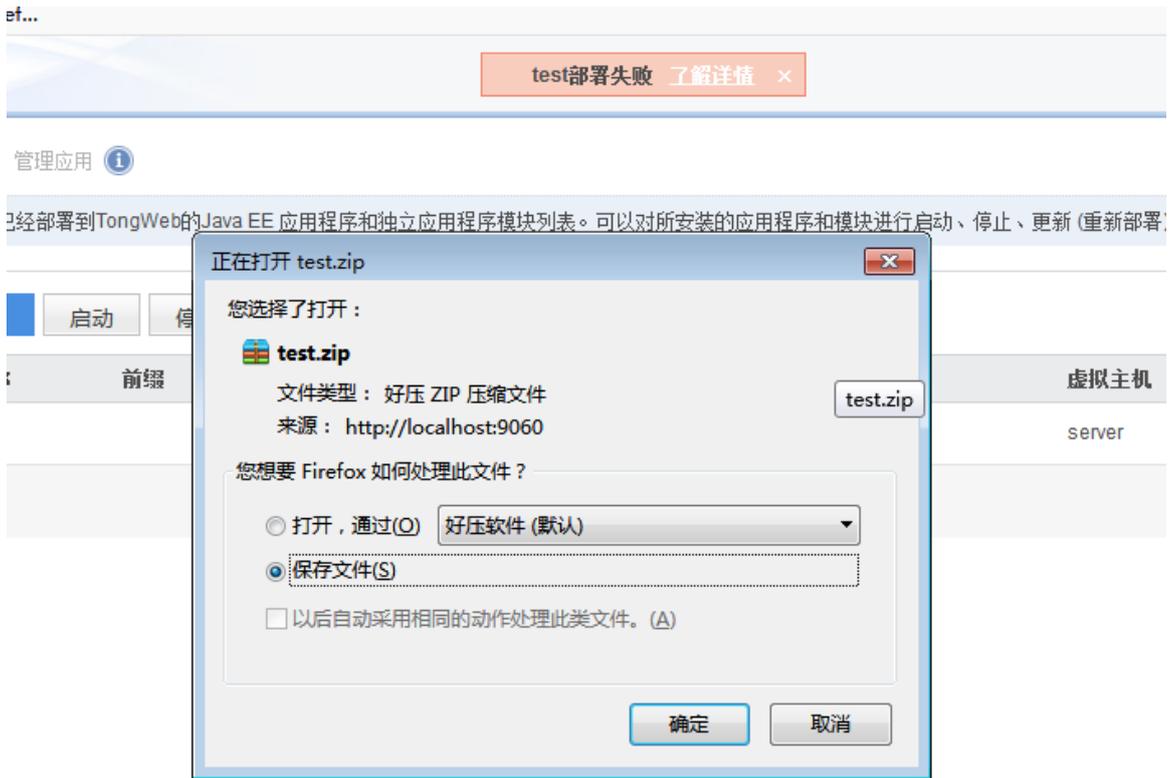


图 3.2.10 部署失败详细信息页面



图 3.2.11 部署成功图

10. 到此为止，应用部署完成，已部署的应用页面中已经出现对应的应用。如图 3.2.12 所示。

应用管理 管理应用 ?

此页显示了已经部署到TongWeb的Java EE 应用程序和独立应用程序模块列表。可以对所安装的应用程序和模块进行启动、停止、更新(重新部署)、删除操作或者

部署应用 启动 停止 解部署							
<input type="checkbox"/>	名称	前缀	应用类型	部署源类型	部署方式	虚拟主机	状态
<input type="checkbox"/>	test		ear	文件部署	控制台部署	server	出错
<input type="checkbox"/>	testweb	/testweb	war	文件部署	控制台部署	server	已启动

图 3.2.12 应用列表

3.2.3 应用查看与编辑

在应用列表中显示的是已经部署完成的应用，可以在列表中对当前的应用进行查看，步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，如图 3.2.13 所示。

应用管理 管理应用 ?

此页显示了已经部署到TongWeb的Java EE 应用程序和独立应用程序模块列表。可以对所安装的应用程序和模块进行启动、停止、更新(重新部署)、删除操作或者

部署应用 启动 停止 解部署							
<input type="checkbox"/>	名称	前缀	应用类型	部署源类型	部署方式	虚拟主机	状态
<input type="checkbox"/>	test		ear	文件部署	控制台部署	server	出错
<input type="checkbox"/>	testweb	/testweb	war	文件部署	控制台部署	server	已启动

图 3.2.13 应用列表

3. 点击“名称”列下面的链接，进入应用查看和编辑页面，如图 3.2.14 所示。

返回 | 编辑应用-- testweb

基本属性

应用名称	testweb	应用名称
应用位置	F:\twnt\bug\tongweb-webprofile-1.5.0\deployment\testweb	应用位置
应用前缀	<input type="text" value="/testweb"/>	应用前缀
部署顺序	<input type="text" value="100"/>	默认的部署顺序
JSP预编译	<input checked="" type="checkbox"/> 支持	JSP预编译
类加载顺序	<input type="radio"/> 父优先 <input checked="" type="radio"/> 子优先	类加载顺序，
描述	<input type="text"/>	应用的描述信

虚拟主机设置

从列表中选择应用的虚拟主机

图 3.2.14 编辑应用

4. 在这个页面中，可以进行编辑，然后保存来更新应用的信息。
5. 对于 EAR 应用来讲，起编辑页面的类加载顺序以树状形式展示、编辑 web 模块的类加载顺序，然后保存来更新应用的信息。如图 3.2.15 所示。

返回
编辑应用-- ejb2

基本属性

应用名称	ejb2		
应用位置	F:\twnt\bug\tongweb-webprofile-1.5.0\deployment\ejb2		
部署顺序	<input style="width: 100%;" type="text" value="100"/>		
JSP预编译	<input type="checkbox"/> 支持		
web类加载父优先	<input checked="" type="checkbox"/> ejb2 <input checked="" type="checkbox"/> ejb2_war		
描述	<div style="background-color: #d9ead3; height: 30px; width: 100%;"></div>		

虚拟主机设置

从列表中选择应用的虚拟主机

server ▾

保存

图 3.2.15 编辑 EAR 应用

3.2.4 查看应用子模块

对于 EAR 应用来讲，企业应用存在子模块，可以对子模块进行查看，步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，在企业应用的“操作”列，存在“子模块”的链接，如 3.2.16 所示。

应用类型	部署源类型	部署方式	虚拟主机	状态	操作
ear	文件部署	控制台部署	server	已启动	重部署子模块

图 3.2.16 应用列表

3. 点击“子模块”的链接，弹出子模块的列表页面，其中包含一个 Web 模块和一个 EJB 模块。如图 3.2.17 所示。

应用类型	部署源类型	部署方式	虚拟主机
子模块 ✕			
名称	类型	操作	
QuerySessionBean	ejb		
TransactionTest1-war	war	http访问 https访问	
<input type="button" value="取消"/>			

图 3.2.17 企业应用的子模块信息

3.2.5 应用解部署

在应用列表中显示的是已经部署完成的应用，选中想要进行解部署的应用，点击列表最上方的“解部署”按钮，可以进行“解部署”的操作。步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，如图 3.2.18 所示。



图 3.2.18 应用列表

3. 点击左侧的复选框，选中对应的应用，然后再点击上面的“解部署”按钮，弹出确认对话框，点击删除，则应用进行解部署操作。如图 3.2.19 所示。



图 3.2.19 删除对话框

3.2.6 应用重部署

应用部署完成后，发现应用中某些类或者配置不正确，修改后则需要重新部署才能生效。步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，如图 3.2.20 所示。

应用管理 管理应用 i



图 3.2.20 应用列表

- 修改应用部署目录下的应用，然后点击“操作”列下面的“重部署”按钮，弹出重部署的页面。见图 3.2.21。

应用管理 管理应用 i



图 3.2.21 应用重部署

- 直接点击“确定”按钮后，对应的应用即进行重部署。
- TongWeb7 中可以提供使用新的部署文件进行重部署。选择复选框，见图 3.2.22。

应用管理 管理应用 i



图 3.2.22 选择重部署文件

- 但是这种重部署的方式，相当于销毁旧应用，重新部署一个新应用。这一步的操作和部署的页面类似，但是重部署的应用的部署附加属性在新应用中被继承下来，且重部署过程中不能修改，从这一点可以看到，如果当前重部署的新应用与原应用从应用的形态上相差过大，可能会导致应用重部署失败，这个是需要特别注意的。另外，新旧应用的应用类型必须保持一致。

注：重部署无法更改类加载顺序。

3.2.7 应用访问

应用已经部署完成，可以对应用进行访问。步骤如下：

- 依次展开管理控制台左侧导航树中的“应用管理”节点；

2. 所有已部署的应用以列表的形式展现，点击操作列下面的“http 访问”，即可弹出新窗口对该应用进行访问。如图 3.2.23 所示。

前缀	应用类型	部署源类型	部署方式	虚拟主机	状态	操作
/TestWeb	war	文件部署	控制台部署	server	已启动	重部署 http访问 https访问
	ear	文件部署	控制台部署	server	已启动	重部署 子模块
/TC_examples	war	文件部署	控制台部署	server	已启动	重部署 http访问 https访问

图 3.2.23 应用列表

值得注意的一点是，对企业级应用，访问的链接在子模块窗口的操作栏中。

3.2.8 应用停止

应用已经部署完成，那么应用默认即为可以访问的状态，这个时候可以对应用进行停止。步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 列表中的应用状态为“已启动”状态，选中对应的应用，点击列表上的停止按钮，即可将对应的应用进行停止。
3. 应用停止仅是对应用禁止访问，并没有真正停止应用；



图 3.2.24 应用停止

3.2.9 应用启动

应用已经部署完成，当应用停止的时候，可以将应用进行启动。步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 所有已部署的应用以列表的形式展现，点击列表上的启动按钮，即可将对应的应用进行启动操作。
3. 应用启动是允许访问访问；

3.2.10 应用更新

应用更新是带有版本的应用，对自身版本升级的一种手段。当前应用更新仅支持 war 类型应用的更新，应用更新的内容包括，跟随着应用发布的 JNDI 对应资源节点，Web 的对应的资源。

首先需要准备应用，在应用目录/包下的 META-INF/MANIFEST.MF 文件中定义属性名为：tongweb-App-Version；属性值为版本号，采用 X.Y 的形式，其中 X 和 Y 都是数字（比如 1.0、1.1、1.2 等等诸如此类）。应用准备完成，按照以下的几个步骤在管理控制台中进行应用更新：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 点击应用管理左上角的“部署应用”按钮，首先将带版本的应用进行部署，部署的步骤按照 3.2.2 节中的部署，如图 3.2.25 所示。

<input type="button" value="部署应用"/> <input type="button" value="启动"/> <input type="button" value="停止"/> <input type="button" value="解部署"/>				
<input type="checkbox"/>	名称	前缀	应用类型	部署源类型
<input type="checkbox"/>	TestWeb	/TestWeb	war	文件部署
<input type="checkbox"/>	stateless		ear	文件部署
<input type="checkbox"/>	TC_examples	/TC_examples	war	文件部署
<input type="checkbox"/>	TestPoolSize(versioned)	/TestPoolSize	war	---

图 3.2.25 应用列表

3. 更新带版本的应用，点击“子版本”，进入应用版本管理页面。该页面显示带版本应用的名称、前缀、应用类型、虚拟主机、退休状态和版本号等信息，如图 3.2.26 所示。

应用 TestPoolSize 的所有版本 [应用版本管理](#) ⓘ

此页显示了该应用的所有版本

<input type="checkbox"/>	名称	前缀	应用类型	虚拟主机	退休状态	版本号
<input type="checkbox"/>	TestPoolSize	/TestPoolSize	war	server	未退休	1.1

图 3.2.26 子版本列表

4. 选择要更新的应用，点“更新”按钮，如图 3.2.27 所示。

应用管理 [管理应用](#) ⓘ

此页显示了已经部署到TongWeb的 Java EE 应用程序和

文件位置 本机 服务器

部署文件 未选择文件。

图 3.2.27 更新应用

5. 将文件上传到服务器端，点击“下一步”按钮，填写应用附加属性页面，在附加属性页面中，增加了更新策略，自然退休和强制退休。对于自然退休来说，即为当前应用不再接受新的请求，等待老的请求（已经建立了会话的请求）的会话超时之后，再进行退休；而强制退休是立即退休，选中“强制退休”选项之后，可以加上一个超时退休时间作为备选选项。如图 3.2.28 所示。

返回 | 应用版本更新--TestPoolSize

应用版本(旧) 1.1

应用版本(新) 1.2

应用名称 TestPoolSize

应用前缀 /TestPoolSize

部署顺序 100

JSP预编译 否

类加载顺序 子优先

描述

更新策略 自然退休 强制退休

完成 取消

图 3.2.28 更新应用

6. 应用更新结束后，页面跳转到应用版本管理页面，如图 3.2.29 所示，应用已经更新完成，列表中存在两个应用，名称和前缀相同，但是版本号不同。被更新的应用的状态为已退休或者正在退休，而更新的应用的状态为未退休。

应用 TestPoolSize 的所有版本 [应用版本管理](#) i

此页显示了该应用的所有版本

返回 更新

名称	前缀	应用类型	虚拟主机	退休状态	版本号	操作
TestPoolSize	/TestPoolSize	war	server	未退休	1.2	
TestPoolSize	/TestPoolSize	war	server	已退休	1.1	切换到该版本 删除该版本

图 3.2.29 子版本列表

3.2.11 应用版本管理

应用更新之后，查看应用版本管理页面的应用列表中，存在已经退休的应用和目前正在提供服务的应用。TongWeb7 提供了应用版本切换和版本删除功能，点击“切换到该版本”的链接，可以将当前的应用切换到不同的版本，点击“删除该版本”可以将对应版本的应用删除，见图 3.2.29 所示。

3.2.12 Connector 应用

3.2.12.1 Connector 应用部署

部署功能请见 [3.2.2 部署](#)，其中上传 rar 应用后进入到应用配置页面，可设置应用名称、部署顺序、选择 connector 应用关联的线程池（默认为 default-thread-pool）及描述。见图 3.2.30:

此页显示了已经安装到此域的 Java EE 应用程序和独立应用程序模块列表。通过首先选择应用程序名, 然后使用此页中的控件, 可以从此域中启动, 停止, 更新 (重新部署)

返回 | 部署应用-genericra

1 基本属性
2 完成部署!

应用名称	<input type="text" value="genericra"/>	✔	
部署顺序	<input type="text" value="100"/>		默认的部署顺序是 100, 如果需要调整部署顺序的话, 可以指定
线程池	<input type="text" value="default_thread_pool"/>		默认为 default_thread_pool
描述	<div style="border: 1px solid #ccc; height: 20px; background-color: #e0ffe0;"></div>		该应用的描述信息

图 3.2.30 connector 应用部署

点击下一步, 确认配置点击“完成”进行部署。返回应用部署列表。见图 3.2.31。

应用管理 管理应用 ①

此页显示了已经部署到TongWeb的Java EE 应用程序和独立应用程序模块列表。可以对所安装的应用程序和模块进行启动、停止、更新 (重新部署)、删除操作或者部署新的应用或模块。

<input type="checkbox"/>	名称	前缀	应用类型	部署源类型	部署方式	虚拟主机	状态	操作
<input type="checkbox"/>	jms		rar	文件部署	控制台部署	server	已启动	重部署
<input type="checkbox"/>	genericra		rar	目录部署	控制台部署	server	已启动	

图 3.2.31 connector 应用列表

3.2.12.2 应用查看与编辑

在应用列表中显示的是已经部署完成的应用, 选择 rar 应用进行查看, 见图 3.2.32。

1. 编辑功能可参考 [3.2.3 应用查看与编辑](#)。
2. 可以修改线程池的关联。
3. 点击“已设属性”查看/编辑已经设置属性值的属性 (通过管理控制台设置的属性, 其中控制台设置的属性应用优先级大于 ra.xml 配置);
4. 点击“全部属性”查看/编辑全部属性 (Connector 应用的 resourceadapter 支持的所有属性);
5. 点击“保存”按钮 (如果应用关联了连接池等资源则保存后重启生效)。

返回
编辑应用-- genericra

基本属性

应用名称	genericra	应用名称	
应用位置	F:\jee6+\products\apache-tomee\target	应用位置	
	/tongweb-webprofile-1.5.0/applications		
	/genericra		
部署顺序	<input type="text" value="100"/>		✔
线程池	<input type="text" value="default-thread-pool"/>		如果不选择，默认为default-thread-pool
描述	<div style="background-color: #d9ead3; height: 20px; width: 100%;"></div>		
			应用的描述信息

属性设置

全部属性

属性名称	值
<input type="text" value="LogLevel"/>	<input type="text" value="INFO"/>
<input type="text" value="JndiProperties"/>	<input "="" type="text" value="java.naming.factory.initial="/>
<input type="text" value="RMPolicy"/>	<input type="text" value="OnePerPhysicalConnectic"/>
<input type="text" value="SupportsXA"/>	<input type="text" value="false"/>
<input type="text" value="ProviderIntegrationMode"/>	<input type="text" value="jndi"/>

图 3.2.32 connector 应用部署

3.2.12.3 应用重部署

具体操作参见 [3.2.6 应用重部署](#)，如果此应用关联了连接池、托管对象等资源则不允许重部署操作。

3.2.12.4 应用解部署

具体操作参见 [3.2.5 应用解部署](#)，如果此应用关联了连接池、托管对象等资源则不允许解部署操作。

3.3 自动部署

TongWeb7 支持文件和目录方式的自动部署，当自动部署功能启动后，TongWeb7 会对自动部署目录进行监控，自动执行应用的部署和解部署操作。

3.3.1 设置自动部署目录

自动部署的启动和停止可通过 TongWeb7 管理控制台进行配置。进入管理控制台，选中“服务”页面下的自动部署属性，则启动自动部署功能，取消选中，则停止自动部署功能。注：在修改了该属性后，需要重启服务器才可以生效。如图 3.3.1 所示。



图 3.3.1 服务配置

在服务配置页面中，可以设置自动部署如下的三个选项：

- 自动部署功能是否开启
- 选择自动部署的目录，可以选择在服务器的任何一个目录上
- 自动部署目录的监控线程扫描时间

设置完成上述的选项后，点击保存，重启服务器后，自动部署的功能开启/关闭。

3.3.2 自动部署支持的应用类型

自动部署支持的应用类型如表 3.3.2 所示。

应用类型	文件扩展名称
企业应用	.ear
Web 应用	.war
EJB 应用	.jar

表 3.3.2

3.3.3 自动部署支持的部署方式

TongWeb7 支持文件和目录两种方式的自动部署。

文件形式：表 3.3.2 中描述的扩展类型文件以及结构符合标准应用的其他扩展。

目录形式：直接将要部署的应用放入到自动部署目录即可。例如：企业应用的目录名称 enterprise，那么直接将 enterprise 放入自动部署目录进行自动部署。

3.3.4 默认的自动部署目录

TW_HOME/autodeploy 是 TongWeb7 的默认自动部署目录。此自动部署目录可以通过 TongWeb7 管理控制台进行配置。进入管理控制台，设置“服务配置”页面下的自动部署目录属性为需要设置的路径即可。

3.3.5 自动部署应用

将要部署的应用文件或目录复制到自动部署目录中即可完成自动部署，部署操作结束后，在自动部署目录中的 .autodeploystatus 文件夹下，会出现一个对应于部署应用的部署状态文件夹，这个文件夹是空的，是用来标记自动部署操作成功与否的状态的。

当出现上述文件夹并且文件夹名称与应用名称相同的时候，说明自动部署应用成功，当 .autodeploystatus 文件夹出现“应用名.failed”的时候，说明当前自动部署的应用部署失败。

注：自动部署应用的类加载顺序使用应用的自定义部署描述文件中配置的顺序。如果应用中无自定义部署描述文件或该文件未指定类加载顺序，默认使用子优先。

3.3.6 自动解部署应用

要将已经自动部署的应用进行解部署，直接删除部署时复制到自动部署目录下的应用文件即可。解部署操作结束后会自动删除自动部署目录中的 .autodeploystatus 文件夹下面的同名文件夹。

如果上述文件夹被删除了之后，说明应用已经成功进行解部署了；当自动部署应用失败之后，进行解部署应用，过一段时间查看 .autodeploystatus 文件夹的文件名 .failed 会自动被清除掉。

3.3.7 自动重部署应用

当自动部署应用出现错误，即当 .autodeploystatus 文件夹出现“应用名.failed”的时候，这个时候经过分析，对此应用可以进行重部署。

自动部署的重部署为直接替换掉自动部署目录下的对应的应用文件和目录，将修改过的相同应用文件或目录重新复制到自动部署目录下。

因为是否进行重部署是根据文件的大小和修改时间进行判定的，若和之前已经部署的应用文件大小和修改时间都相同，则不会进行重部署。（针对目录部署的情况，是根据一级文件夹的大小和修改时间判定的）。

3.4 热部署

热部署同样是应用部署的一种方式，其主要的作用是在应用已经部署之后，需要在线实时对应用进行修改，结果立刻展现的一种方式。

3.4.1 热部署配置

热部署的开启可通过 TongWeb7 管理控制台进行配置。进入管理控制台，选中“服务”页面下的热部署的选项，选中，保存后即生效。如图 3.4.1 所示。

自动部署	<input checked="" type="checkbox"/> 开启	开启之后监听的是目录中应用的增加和删除																										
自动部署目录	<table border="1"><tr><td>C:/</td><td>\$RECYCLE.BIN</td></tr><tr><td>D:/</td><td>\$WINDOWS.~BT</td></tr><tr><td>E:/</td><td>360Downloads</td></tr><tr><td>F:/</td><td>Apache2.2</td></tr><tr><td>G:/</td><td>apache编译_多平台</td></tr><tr><td></td><td>cts</td></tr><tr><td></td><td>download</td></tr><tr><td></td><td>eclipse-reviewboard</td></tr><tr><td></td><td>eclipse-reviewboard_Tongweb5</td></tr><tr><td></td><td>Oracle</td></tr><tr><td></td><td>OtherServer</td></tr><tr><td></td><td>recovery</td></tr><tr><td></td><td>System Volume Information</td></tr></table>	C:/	\$RECYCLE.BIN	D:/	\$WINDOWS.~BT	E:/	360Downloads	F:/	Apache2.2	G:/	apache编译_多平台		cts		download		eclipse-reviewboard		eclipse-reviewboard_Tongweb5		Oracle		OtherServer		recovery		System Volume Information	已选择文件：F:\twnt\bug\tongweb-webprofile-1.5.0\autodeploy
C:/	\$RECYCLE.BIN																											
D:/	\$WINDOWS.~BT																											
E:/	360Downloads																											
F:/	Apache2.2																											
G:/	apache编译_多平台																											
	cts																											
	download																											
	eclipse-reviewboard																											
	eclipse-reviewboard_Tongweb5																											
	Oracle																											
	OtherServer																											
	recovery																											
	System Volume Information																											
时间间隔	<input type="text" value="3000"/>	毫秒 检查自动部署目录的时间间隔，以毫秒为单位。																										
热部署	<input checked="" type="checkbox"/> 开启	是否开启部署目录资源修改后实时生效的开关																										

图 3.4.1 服务配置

3.4.2 热部署应用

当热部署的开关开启之后，修改 TW_HOME/deployment 下面的应用的类文件和应用的配置文件 web.xml 的时候，应用会发生热部署，应用重新进行加载，将最新的内容提

供给用户。

如果热部署的是 ear 应用，则需要替换 deployment/ear 应用目录下未解压的子模块（web 模块或 ejb 模块），该 ear 应用就会发生热部署。（替换 ear 目录下已经解压的子模块中的内容，不能触发热部署）。

4 Web 容器

4.1 Web 容器说明

在 Java EE 平台上，Web 应用运行在 Web 容器中。Web 容器提供了 Web 应用的运行时环境，包括生命周期管理、安全、请求转发等。同时 Web 容器为 Web 应用提供访问其他 API 的能力，如命名服务。

Web 应用通过 XML 格式的部署描述文件来定义自身的行为。借助 Web 容器的帮助，Web 应用不需要自己实现很多和它自身业务逻辑不相关的复杂逻辑，如数据库连接池等。当它需要时，只需要在部署描述文件中声明或者在应用中通过 annotation 将资源注入即可。Web 容器通过读取应用的部署描述文件，了解该应用需要什么样的服务，在应用程序部署后自动实现。

一个容器可以同时运行多个 Web 应用，它们一般通过不同的 URI 来进行区分和访问，如 <http://host:port/contextroot/servletname>（说明：http(或 https)://虚拟主机名或别名:虚拟主机关联的通道的监听端口/虚拟主机上部署应用的访问前缀/应用中的 Servlet 名）。

4.1.1 Web 容器配置

查看/编辑全局 Web 容器配置的属性

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“容器配置”节点，出现如图 4.1.1 所示的“容器配置”页面；

容器配置 配置容器属性 ⓘ

此页显示了web容器配置属性，可编辑这些属性。

jvmRoute	<input type="text"/>	在负载均衡场景中所必须的唯一标识，以支持s
JSP开发模式	<input checked="" type="checkbox"/> 开启	启用JSP开发模式
默认请求参数解码字符集	<input type="text" value="GBK"/>	默认请求参数解码字符集，此属性修改后需要重
默认应答编码字符集	<input type="text" value="GBK"/>	默认应答编码字符集，此属性修改后需要重启服
session超时时间	<input type="text" value="30"/>	session超时时间，以分钟为单位
session复制日志	<input type="checkbox"/>	开启或关闭记载session复制日志
超时线程日志	<input type="checkbox"/>	开启或关闭超时线程日志
超时线程阈值	<input type="text" value="0"/>	超时线程阈值，以秒为单位
http慢攻击检测	<input type="checkbox"/> 开启	启用http慢攻击检测，相关属性修改需要重启服
主机名验证器	<input type="text" value="无"/>	用于验证SSL连接的主机名是否被篡改
防host头攻击	<input type="checkbox"/>	开启或关闭防host头(主机名)攻击
应用退休超时时间	<input type="text" value="2"/>	等待web应用程序卸载的秒数

图 4.1.1 web 容器配置属性

- 属性：

- `jvmRoute`: 是在负载均衡场景中所必须提供的唯一标识, Apache 通过该属性来区分不同的 TongWeb 容器, 用以实现 session 亲和。
 - JSP 开发模式: 开启 JSP 开发模式后, 每次访问 JSP 页面都会重新编译该页面, 对页面做的修改能够实时生效。
 - 默认请求参数解码字符集: 默认的请求参数解码字符集。此属性修改后需要重启服务器才能生效。
 - 默认应答编码字符集: 默认的应答编码字符集。此属性修改后需要重启服务器才能生效。
 - Session 超时时间: session 超时的全局配置。优先级小于应用的 session 超时时间。
 - Session 复制日志: 开启或关闭 session 复制日志信息。
 - 超时线程日志: 开启或关闭记录超时线程日志。
 - 超时线程阈值: 超时线程的阈值, 单位为秒, 0 表示不开启超时线程检测任务。
 - http 慢攻击检测: 具体说明详见[防 SDOS 攻击 \(慢攻击\)](#)
 - 主机名验证器: 客户端应用程序访问服务器时, 即 `url.openConnection()`, 是否利用验证器验证。当选择“无”时, 表示客户端应用程序访问服务器时, 没有验证器验证; 当选择“TW 主机名验证器”时, 表示无论客户端应用程序是否加入验证器干扰, 都没用, 始终会经过 TW 主机名验证器去验证。如果访问的链接精确匹配, 则验证通过; 否则, 验证不通过。当选择“自定义主机名验证器”时, 显示主机名验证器类文本框。表示无论客户端应用程序是否加入验证器干扰, 都没用, 始终会经过自定义主机名验证器去验证。用户自定义的验证器类必须实现 `jdk` 的接口 `HostnameVerifier`, 并且打成 `jar` 包, 放入启动工程的 `lib` 目录中, 并重启。然后在文档框中填入自定义验证器类的全路径, 包括包名。文本框主机名验证器类为空或者输入错误, 则表示没有加入自定义验证器。
 - 防 Host 头攻击: 开启或关闭防 Host 头攻击, 开启时, 需要配置正确的主机名白名单, 否则可能导致无法访问应用。
 - 应用退休超时时间: 等待 web 应用卸载程序的秒数, 默认是 2 秒。在此期间服务器将继续处理未处理完毕的请求, 处理完毕后卸载。
3. 编辑属性完成后, 点击“保存”按钮。
- 说明: 默认请求参数解码字符集和默认应答编码字符集所提供的可选的内容是可扩展的, 目前下拉列表中提供四种常用的字符集, 如果需要添加其他的字符集可以在 `console\WEB-INF\classes\webcontainer_charset.xml` 配置文件中添加需要的字符集。

4.2 会话高可用

4.2.1 会话高可用提供的功能

会话高可用的特性: 在 web 集群中某些节点故障后 session 数据不会丢失, web 集群中其它可用节点仍然可以使用此 session 数据为用户请求提供服务, 从而使得 web 集群节点的失败和故障对用户请求透明化。

全局会话高可用配置功能, 可以为所有部署的应用启用会话高可用特性, 需要注意的是如果在 TongWeb 运行期间, 在打开全局会话高可用配置之前部署的应用, 都无法使用会话高可用特性, 需要对应用进行重部署操作。

4.2.2 会话高可用的使用

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点;
2. 点击“会话高可用”节点, 出现如图 4.2.2 所示的“会话高可用配置”页面;

此页用于配置全局会话高可用。

功能开关	<input checked="" type="radio"/> 开启 <input type="radio"/> 关闭	是否启用全局应用的会话高
缓存集群地址	<input style="width: 90%;" type="text" value="127.0.0.1"/>	要连接到的缓存集群地址， 冒号后面的部分是该缓存节 127.0.0.1,10.10.4.50:5701,
缓存集群组名称	<input style="width: 90%;" type="text" value="test"/>	要连接到的缓存集群组名称
缓存集群组密码	<input style="width: 90%;" type="text" value="dev-pass"/>	要连接到的缓存集群组密码
异步写入开关	<input type="radio"/> 开启 <input checked="" type="radio"/> 关闭	异步写入开关是否开启异步
超时时间	<input style="width: 90%;" type="text" value="500"/>	会话备份超时时间（单位：
亲和性会话开关	<input checked="" type="radio"/> 开启 <input type="radio"/> 关闭	是否使用亲和性会话

图 4.2.2 编辑会话高可用属性

- 基本属性
 - 功能开关：是否启用全局应用的会话高可用特性。
 - 缓存集群地址：要连接到的缓存集群地址，默认值是“127.0.0.1”，完整格式如“192.168.0.1:5701”，其中冒号前面的部分是缓存集群内任意一个缓存节点所在机器的 IP 地址如 192.168.0.1，冒号后面的部分是该缓存节点在此机器上的监听端口如 5701（注：端口配置不是必需的，但建议配置）。如果有多个缓存节点要用英文逗号分割，如 127.0.0.1,10.10.4.50:5701,192.168.0.1:5702,...
 - 缓存集群组名称：要连接到的缓存集群组名称，默认值“dev”，用于连接到特定的缓存集群。
 - 缓存集群组密码：要连接到的缓存集群组密码，默认值“dev-pass”，用于连接到特定的缓存集群。
 - 异步写入开关：异步写入开关是否开启异步存储会话数据的功能，默认值是 false 表示不开启。
 - 超时时间：会话备份超时时间（单位：毫秒），默认值为 500。
 - 亲和性会话开关：是否使用亲和性会话。

4.3 访问日志

4.3.1 访问日志提供的功能

4.3.1.1 访问日志基本功能

记录访问 Web 应用时 http 请求的相关信息，包括请求的方法，请求的协议，请求头中的信息，请求响应的状态码等，不包括 Web 应用本身输出的日志信息。

访问日志默认使用的消息格式为“%{yyyyMMddHHmmssSSS}t %U %m %a %D”，包含了应用访问中最重要的一项信息：访问时间，请求链接，请求方式，远程 ip 以及请求处理时间。使用默认格式生成的访问日志信息如：“20131231120000001 /test.jsp GET 127.0.0.1 0”。

每个虚拟主机控制是否生成访问日志文件，且访问日志可以存放在不同的目录上。默认的访问日志文件名形如 access_log.admin.13.12.31.12.txt，即 access_log 作为文件名前缀，txt 作为文件名后缀，中间为虚拟主机名称和日志时间，默认以小时进行轮转。

4.3.2 访问日志的使用

查看/编辑访问日志的属性

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“访问日志”节点，出现如图 4.2.1 所示的“访问日志”页面；
此页显示了访问日志的配置属性，可编辑访问日志属性。

The screenshot shows the 'Access Log' configuration interface. It includes the following fields and options:

- 文件前缀:
- 文件后缀:
- 扩展日志格式: 开启扩展
- 日志格式:
- 是否轮转: 轮转
- 轮转周期: 按天 按小时
- 容量限制: 最大天数 天; 最大个数 个
- 文件日期格式:
- 保存按钮: 位于配置项下方，蓝色背景，白色文字。

图 4.2.1 编辑访问日志属性

- 基本属性
 - 文件前缀：生成的日志文件前缀，默认为“access_log.”。
 - 文件后缀：生成的日志文件后缀，默认为“.txt”。
 - 扩展日志格式：是否开启扩展日志。
 - 日志格式：单条日志记录的格式，默认为“%{yyyyMMddHHmmssSSS}t %U %m %a %D”。
 - 是否轮转：访问日志轮转开关，默认为选中，即开启状态。
 - 轮转周期：访问日志的轮转间隔时间，提供了按天和按小时两种方式。按天方式将每天的访问信息记录在一个文件中，按小时方式则每小时生成一个日志文件记录该小时的访问信息。
 - 容量限制：限制访问日志的个数，有两种方式，可同时生效，一种是限制日志文件存在的天数，超过这个天数的日志将被删除，一种仅限制日志文件个数，当日志文件个数超过设置的个数，多余的日志将会被删除，两种方式配置-1 或 0 将不会生效，另外只有访问日志功能被启用并且访问日志轮转开发开启时，功能才会生效。另外，这里的配置均是针对每个虚拟主机下的日志文件的，非访问日志目录下总的日志文件。
 - 文件日期格式：日期格式，影响到文件名，默认为“yy.MM.dd.HH”。支持基本的时间格式 (<http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html> )，但是需要与轮转周期一致。
 - 3. 编辑属性完成后，点击“保存”按钮。
- 说明：访问日志属性配置的修改可能会影响管理控制台的诊断功能中对于访问日志的查看检索。

4.3.3 访问日志格式

TongWeb7 默认使用传统类型的日志，并且默认的日志格式为“%{yyyyMMddHHmmssSSS}t %U %m %a %D”。传统日志和扩展日志的日志格式都可以进行自定义配置，日志格式字符串中的各配置定义如下。

传统日志：

- %a - 远程主机的 IP 地址。
- %A - 本地 IP 地址。
- %b - 除了 HTTP 头文件外所发送的字节数，如果是 0 则记录为“-”。
- %B - 除了 HTTP 头文件外所发送的字节数。
- %h - 远程主机名，如果 connector 没有开启 DNS 反向查找，则为远程主机的 IP 地址。
- %H - 请求的协议。
- %m - 请求方法，如（GET、POST 等）。
- %p - 接受请求的本地端口。
- %q - 查询字符串，包含“?”。
- %r - 请求头的第一行。
- %s - HTTP 状态码。
- %S - 用户的 session id。
- %t - 访问时间。
- %u - 被授权登陆的用户，如果没有则记录“-”。
- %U - 请求链接地址。
- %v - 本地服务器名。
- %D - 请求处理时间，以毫秒表示。
- %T - 请求处理时间，以秒表示。
- %F - 提交响应消耗的时间，以毫秒表示。
- %I - 当前请求的线程名称。
- %{xxx} - 请求头的 xxx 属性。
- %{xxx}o - 响应头的 xxx 属性。
- %{xxx}c - 名为 xxx 的 cookie 值。
- %{xxx}r - ServletRequest 的 xxx 属性。
- %{xxx}s - HttpSession 的 xxx 属性。
- %{xxx}t - 以 xxx 格式的 SimpleDateFormat 记录请求时间。

扩展日志：

- bytes -除了 HTTP 头文件外所发送的字节数，如果是 0 则记录为“-”。
- c-dns -远程主机名，如果 connector 没有开启 DNS 反向查找，则为远程主机的 IP 地址。
- c-ip - 远程 IP 地址。
- cs-method -请求方法，如（GET、POST 等）。
- cs-uri -请求的 URI。
- cs-uri-query -查询字符串，包含“?”。
- cs-uri-stem - 请求的 URL 地址。
- date - “yyyy-mm-dd”格式的时间。
- s-dns - 本地主机名。
- s-ip - 本地 IP 地址。
- sc-status - HTTP 状态码。
- time - “HH:mm:ss”格式的请求处理时间。
- time-taken - 请求处理时间。
- x-threadname - 当前请求的线程名称。
- x-H(authType) - HttpServletRequest 对象中 getAuthType 方法返回值。
- x-H(characterEncoding) - HttpServletRequest 对象中 getCharacterEncoding 方法返

返回值。

x-H(contentLength) - HttpServletRequest 对象中 getContentLength 方法返回值。

x-H(locale) - HttpServletRequest 对象中 getLocale 方法返回值。

x-H(protocol) - HttpServletRequest 对象中 getProtocol 方法返回值。

x-H(remoteUser) - HttpServletRequest 对象中 getRemoteUser 方法返回值。

x-H(requestedSessionId) - HttpServletRequest 对象中 getRequestedSessionId 方法返回值。

x-H(requestedSessionIdFromCookie) - HttpServletRequest 对象中 isRequestedSessionIdFromCookie 方法返回值。

x-H(requestedSessionIdValid) - HttpServletRequest 对象中 isRequestedSessionIdValid 方法返回值。

x-H(scheme) - HttpServletRequest 对象中 getScheme 方法返回值。

x-H(secure) - HttpServletRequest 对象中 isSecure 方法返回值。

cs(XXX) - 请求头的 XXX 属性。

sc(XXX) - 响应头的 XXX 属性。

x-A(XXX) - servlet context 的 XXX 属性。

x-C(XXX) - 名为 XXX 的 cookie 值。

x-O(XXX) - 响应头中与 XXX 相关的一系列属性。

x-P(XXX) - 使用 UTF-8 编码的 URL 请求参数。

x-R(XXX) - request 的 XXX 属性。

x-S(XXX) - session 的 XXX 属性。

4.3.4 访问日志使用示例

示例目标：默认虚拟主机 server 生成访问日志文件。

示例步骤如下：

1. 配置访问日志属性：具体操作步骤见“[访问日志的使用](#)”。
2. 开启默认虚拟主机 server 的访问日志开关，具体操作步骤见“[查看/编辑虚拟主机](#)”。
3. 部署 Web 应用 test1.war 到默认虚拟主机 Server 上。
4. 通过 `http://localhost:8088/test1/jspa.jsp` 访问应用 test1.war 中的 `jspa.jsp`。
5. 在 `TW_HOME/logs/access_log.server.14.03.31.12.txt` 中查看访问日志信息，出现如下访问日志信息。

```
20140331120000031/test1/jspa.jsp GET 127.0.0.1 0
```

说明：管理控制台的诊断功能中提供了对于访问日志的查看和检索功能，具体使用查看相关章节。

4.4 虚拟主机

4.4.1 虚拟主机提供的功能

4.4.1.1 虚拟主机的基本功能

虚拟主机是将单个物理主机分成多个“虚拟”的主机，即虚拟主机间可共享一台物理主机的资源。每个虚拟主机通过“虚拟主机的唯一标识(id)”来区分，但是每个虚拟主机可以使用多个主机名(虚拟主机的唯一标识和虚拟主机别名)。

一个 Web 应用可以部署在多个虚拟主机上，一个虚拟主机可以与多个通道关联。

虚拟主机提供“启用/停用”开关，默认为启用。如果为“启用”，则虚拟主机接收请求并进行处理，如果为“停用”，则虚拟主机不接收请求(返回状态码 404)。

4.4.1.2 缺省虚拟主机

服务器提供缺省的虚拟主机，如果入站请求中未指定特定的虚拟主机，则入站请求会发送到缺省虚拟主机，如果部署应用时未指定特定的虚拟主机，则应用程序会部署到缺省虚拟主机中。缺省虚拟主机的名称为“server”。

4.4.1.3 单点登陆

如果开启单点登陆功能，用户只需认证一次就可以访问部署在该虚拟主机上所有使用相同安全域的 Web 应用。

如果关闭单点登陆功能，则用户访问部署在同一虚拟主机上的使用相同安全域的不同应用，需要分别认证。

注意：单点登录功能是根据应用使用的安全域进行匹配的，部署在同一虚拟主机上并使用相同安全域的多个应用，能够通过单点登录进行认证访问。当使用单点登录功能进行认证访问后，若再次进行认证访问该虚拟主机上使用其它安全域的应用时，之前通过单点登录功能进行认证访问的应用，将无法通过单点登录功能进行访问，再次访问时需再次进行认证；而与新认证访问的应用具有相同安全域的应用，将可以通过单点登录功能进行访问。

例如：虚拟主机 vs1 开启单点登录功能，部署应用 app1、app2、app3 和 app4，app1 和 app2 应用使用相同安全域 realm1，app3 和 app4 应用使用相同安全域 realm2。当访问 app1 并进行认证后，可通过单点登录进行访问 app2，而不用再次认证。此时 app1 和 app2 均能正常访问。若此时访问 app3 并进行认证，则 app4 可通过单点登录功能进行访问，而不用再次认证，此时 app3 和 app4 均能正常访问。若此时再次访问 app2，则 app2 需要再次认证。

4.4.1.4 远程访问过滤

通过配置虚拟主机上的远程过滤功能，可以允许或拒绝来自某些地址或主机对该虚拟主机上 Web 应用的请求。当用户不配置此功能时，则服务器不进行任何访问过滤。

具体配置见“查看/编辑虚拟主机”中“[创建虚拟主机](#)”的基本属性。

4.4.2 虚拟主机的使用

4.4.2.1 创建虚拟主机

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“虚拟主机管理”节点；



The screenshot shows a management interface with a table of virtual hosts and a set of control buttons. The buttons include '创建虚拟主机' (Create Virtual Host), '启动' (Start), '停止' (Stop), and '删除' (Delete). The table has columns for '名称' (Name), '别名' (Alias), '状态' (Status), and '关联的通道' (Associated Channels).

<input type="checkbox"/> 名称	别名	状态	关联的通道
admin		已启动	system-http-listener
ejb		已启动	ejb-server-listener
server		已启动	tong-http-listener,tong-https-listener

图 4.3.1 虚拟主机列表

3. 在出现的“虚拟主机管理”页面中，点击“创建虚拟主机”按钮，出现如图 4.3.2 所示的“创建虚拟主机”页面；

图 4.3.2 创建虚拟主机

- 基本属性
 - 虚拟主机名称：虚拟主机的唯一标识，新建虚拟主机名称不能与已使用的 HTTP、AJP 通道名称相同
 - 虚拟主机别名：虚拟主机的别名，可以定义多个别名，当定义多个别名时用逗号分隔。
 - 通道列表：与虚拟主机关联的通道名称的列表。
 - 访问日志：是否开启访问日志，默认开启。
 - 访问日志目录：可以单独制定此虚拟主机存储其访问日志的目录，默认为 logs/access。
 - SSO：是否开启单点登陆，默认关闭。
 - 远程过滤：是否开启远程过滤，默认关闭，选择开启有如下属性可以填写：

允许的 ip 地址：用逗号分隔的一个正则表达式，客户端的 IP 地址与正则表达式进行匹配。如果指定了这个属性，客户端的地址必须匹配这个表达式，其请求才会被处理。如果没有指定这个属性，所有的请求都被接受，除非客户端地址被“拒绝访问的远程地址”所匹配。

禁止的 ip 地址：用逗号分隔的一个正则表达式，客户端的 IP 地址与正则表达式进行匹配。如果指定了这个属性，客户端的地址只有不匹配这个正则表达式，其请求才会被处理。

例如：使用通配符的正则表达式表示 168.1.103.0 到 168.1.103.99 的远程地址：
 168\.\.1\.\.103\.\.([0-9]|[1-9][0-9]) (“\.” 为 “.” 的转义字符，“|” 为 “或” 选择符)

允许的主机名：用逗号分隔的一个正则表达式，客户端的主机名与正则表达式进行匹配。如果指定了这个属性，客户端的主机名必须匹配这个表达式，其请求才会被处理。如果没有指定这个属性，所有的请求都被接受，除非客户端主机名被“拒绝访问的远程主机”所匹配。“DNS 反向查找”功能打开时该配置生效，该功能在创建或编辑通道时进行配置。

禁止的主机名：用逗号分隔的一个正则表达式，客户端的主机名与正则表达式进行匹配。如果指定了这个属性，客户端的主机名只有不匹配这个正则表达式，其请求才会被处理。“DNS 反向查找”功能打开时该配置生效，该功能在创建或编辑通道时进行配置。

例如：使用通配符的正则表达式表示以 TW 开头的任意远程主机：TW.*。（“.”为单个任意字符，“.*”为“.”出现 0 次或多次）编辑属性完成后，点击“保存”按钮：

- 自定义 valve：自定义 valve 的类名，自定义的 valve 必须实现 com.tongweb.catalina.Valve 接口。自定义的 Valve 会添加到虚拟主机的 pipeline (valve 链) 里。访问应用时执行自定义的操作。如果要为该 valve 属性设置值，可以在类名后面用“,”间隔进行追加，例如 com.tongweb.catalina.valves.RemoteIpValve,protocolHeader=X-Forwarded-Proto, 则为 RemoteIpValve 的对象属性“protocolHeader”设值为字符串“X-Forwarded-Proto”。

注意：创建虚拟主机时，虚拟主机名和虚拟主机别名彼此不能存在相同的名称，并且不能和已经创建的虚拟主机名以及别名相同。

虚拟主机 server 会绑定所有 AJP 通道，且 AJP 通道只会和 server 主机绑定，具体说明见 [AJP 通道中的说明](#)。在虚拟主机的创建和编辑时，可绑定的通道列表中，不包含 AJP 通道。

4. 虚拟主机同时提供了用户自定义配置其他属性的功能，可配置的属性包括：
 - autoDeploy：运行时，周期性地检查新加的或更新的 web 应用。默认为 true。
 - deployOnStartup：服务器启动时，自动部署本 host 的 web 应用。
 - deployIgnore：正则表达式，指定忽略的部署目录。如“/.svn”目录。
 - workDir：临时文件产生目录。
 - appBase：应用部署目录
 - xmlBase：应用 xml 描述符目录。
 - allowLinking：配置为 true 时，新部署在该虚拟主机上的应用可以通过操作系统的软连接功能，使用应用子路径访问应用路径之外的资源。使用该功能属性配置后，需要重启应用服务器。
 - X-Forwarded-Proto：配置为 true 时，如果请求由代理 Server 转发过来时，可获得真实的客户端访问协议（https 或 http），代理 Server（例如 nginx）需设置请求头名称为“X-Forwarded-Proto”。
 - cacheMaxSize：为部署在此虚拟机下的应用设置静态资源缓存的最大值，单位为 K。
 - cachingAllowed：为部署在此虚拟机下的应用设置是否允许启用静态资源（HTML、图片、声音等）的缓存。默认值为 true。
 - cacheObjectMaxSize：为部署在此虚拟机下的应用设置允许缓存的最大文件容量，大小此容量的文件将不被 Cache。cacheObjectMaxSize 会被限定在 cacheMaxSize/20 以下。
 - cacheTTL：为部署在此虚拟机下的应用设置 Cache 有内容的检查间隔。默认 5000, 单位 ms。

5. 点击“保存”按钮；

4.4.2 查看/编辑虚拟主机

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“虚拟主机管理”节点；
3. 在出现的“虚拟主机管理”页面中，点击需要查看/编辑的虚拟主机名称，出现如图 4.3.3 所示内容；
4. 各属性与创建虚拟主机时相同。



图 4.3.3 查看/编辑虚拟主机

说明：系统级的虚拟主机 admin 不提供“虚拟主机别名”、“通道列表”的编辑。

4.4.2.3 启动/停止虚拟主机

启动/停止一个或者多个虚拟主机：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“虚拟主机管理”节点；
3. 选中一个或多个虚拟主机；
4. 点击“启用”/“停用”按钮；

4.4.2.4 删除虚拟主机

1. 依次展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“虚拟主机管理”节点；
3. 选中待删除的虚拟主机；
4. 点击“删除”按钮。

4.4.3 虚拟主机使用示例

4.4.3.1 远程访问过滤

示例目标：拒绝机器(IP 地址为 168.1.103.13)访问应用服务器(位于 168.1.103.13 机器上)中虚拟主机 Server 上部署的 Web 应用；

使用步骤：

1. 在 IP 地址为 168.1.103.13 的机器上启动 TongWeb7 应用服务器。
2. 在管理控制台中编辑名为 server 的“虚拟主机”，开启远程过滤，并设置禁止的 ip 地址为“168.1.103.13”，如图 4.3.4 所示。编辑虚拟主机的步骤参见“[4.3.2.2 查看/编辑虚拟主机](#)”。



图 4.3.4 远程主机过滤

3. 在 168.1.103.13 的 TongWeb7 上部署任意 Web 应用(其中包含若干个 jsp)到默认虚拟主机 Server 上，应用访问前缀为 test.。
4. 通过 168.1.103.13 机器访问该 Web 应用(请求 URI 为 http://168.1.103.13:8088/test/具体_jsp_页面)，访问失败。
5. 通过 168.1.103.94 的机器(或任意 IP 地址不是 168.1.103.13 的其他机器)访问该 Web 应用(请求 URI 为 http://168.1.103.13:8088/test/具体_jsp_页面)，访问成功。

4.5 通道

4.5.1 通道提供的功能

4.5.1.1 通道的基本功能

Web 容器使用通道接收用户请求(每个通道提供自己的监听地址和监听端口)，根据传输协议的不同，提供三种类型的通道：HTTP(HTTP1.1)，HTTPS 和 AJP(AJP1.3)。同时为通道提供了 4 种不同的 io 模式：bio，nio，nio2 与 apr。服务器为通道提供开关，只有通道开启才能接收用户请求。

4.5.1.2 长连接

HTTP1.0 规定浏览器与服务器只保持短暂的连接，浏览器的每次请求都需要与服务器建立一个 TCP 连接，服务器完成请求处理后立即断开 TCP 连接(短连接)。为了克服 HTTP1.0 的这个缺陷，HTTP1.1 支持持久连接，在一个 TCP 连接上可以传送多个 HTTP 请求和响应，减少了建立和关闭连接的消耗和延迟(长连接)。TongWeb7 服务器支持 HTTP1.1，因此默认的连接均为长连接。

用户可以控制长连接中最大请求个数以及长连接的超时时间。

4.5.1.3 通道的工作模式

通道提供四种 io 工作模型：bio，nio，nio2 与 apr。

其中 bio 同步并阻塞模式，服务器实现模式为一个连接一个线程，即客户端有连接请求时服务器端就需要启动一个线程进行处理

其中 nio 与 nio2 都是非阻塞 io 模式，使用线程池技术来保证对请求接收的不阻塞。

使用传统 IO 的阻塞工作模式的主线程使用 ServerSocket.accept() 接收请求。nio 的非阻塞工作模式处理请求都是将接收到的 socket 请求交由在线程池中的处理线程处理的。

因此，接收和处理请求是在不同的线程中进行，不会占用主线程，即使线程池中也没有空闲的工作线程，也不会拒绝接收请求。

Apr 即 Apache Portable Runtime，是从操作系统层面解决 io 阻塞问题。通道使用 apr io 模式需要操作系统对 apr 的支持。

在 TongWeb7 安装目录中 native 目录下存在对各操作系统支持的 apr 链接库。如果是在 unix 或者 linux 操作系统下，启动脚本 startserver.sh 会将 apr 链接库添加到系统环境变量 LD_LIBRARY_PATH 中。如果是在 windows 操作系统下，则需要根据不同的平台(32 位或 64 位)将 apr.dll 放到 TongWeb7 安装目录的 bin 目录下。

4.5.1.4 传输压缩(HTTP&HTTPS)

在 HTTP1.1 中支持 gzip 压缩，这样可以缩小页面的容量从而加快页面的显示速度。

服务器提供是否压缩选项:不使用，压缩文本数据，强制压缩。同时服务器还提供“压缩类型”和“压缩的内容最小值”配置项。

具体配置见“查看/编辑通道”中“[传输压缩](#)”的相关属性。

4.5.1.5 SSL(HTTPS)

支持服务端单向认证，服务端/客户端双向认证。支持 SSL/SSL3/TLS 协议。

如果 HTTPS 通道 io 模式选择 apr，则必须开启 openssl 的支持。

4.5.2 通道的使用

4.5.2.1 创建通道

如果创建 HTTP/HTTPS 类型的通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“HTTP 通道管理”节点；



The screenshot shows a management interface with a table of channels. At the top, there are four buttons: '创建HTTP通道' (Create HTTP Channel), '启动' (Start), '停止' (Stop), and '删除' (Delete). The table has five columns: '名称' (Name), '状态' (Status), '监听端口' (Listening Port), '类型' (Type), and '默认虚拟主机' (Default Virtual Host). The table contains four rows of channel information.

名称	状态	监听端口	类型	默认虚拟主机
system-http-listener	已启动	9060	http	admin
ejb-server-listener	已启动	5100	http	ejb
tong-http-listener	已启动	8080	http	server
tong-https-listener	已启动	8443	https	server

图 4.4.1 通道列表

3. 在出现的“HTTP 通道管理”页面中点击“创建 HTTP 通道”按钮，出现如图 4.4.2 所示“创建 HTTP 通道”页面；

4. 依次填写通道的基本属性、高级属性、线程池属性以及压缩属性。

返回
创建HTTP通道

1 基本属性
2 高级属性
3 线程池属性

*** http通道名称**

http通道类型 http https

监听地址 全部 指定IP

*** 监听端口**

*** 默认虚拟主机**

重定向端口

io模式

代理服务器URL

X-Powered-By 开启

通道名称, 唯一的标识

通道类型, 选择https后需设置SSL属性

指定IP

监听端口号

默认虚拟主机

重定向端口

io模式

代理服务器的url, 格式为ip:port

启用后, 在响应header中有如, X-Powered-By:

上一步
下一步
取消

图 4.4.2 创建 HTTP 通道

- 基本属性
 - 通道名称: HTTP 通道的唯一标识, 新建通道名称不能与虚拟主机名称、已使用的 HTTP、AJP 通道名称相同。
 - 通道类型: HTTP/HTTPS。
 - 监听地址: HTTP 通道的监听地址, 可以为全部或其它指定的 IP 地址。
 - 监听端口: HTTP 通道的监听端口号。
 - 默认虚拟主机: 通道对应的默认虚拟主机。
 - 重定向端口: 非 ssl 到 ssl 的重定向端口。
 - io 模式: bio, nio 与 nio2, 如果当前系统支持 apr, 则可选 apr。默认为 nio 模式。
 - 代理服务器的 URL: 代理服务器名和端口组成的 URL 为 http(s)://proxyName:proxyPort, 其中, proxyName 代表 Proxy 转发模式下, Proxy 对外提供服务的地址。当设置此配置, ServletRequest 的 getServerName 方法返回此 Proxy 地址, 否则返回通道的监听地址。proxyPort 代表 Proxy 转发模式下, Proxy 对外提供服务的端口。当设置此配置, ServletRequest 的 getServerPort 方法返回此 Proxy 端口, 否则返回通道的监听端口。
 - xpowered-by: 用于设置是否在 response 的 http 头里生成 X-Powered-By 信息。
- SSL 属性(选择通道类型为“HTTPS”时显示)
 - SSL 协议版本: https 采用的 SSL 协议版本。
 - 客户端认证: 是否使用客户端数字证书来认证, 默认为不选中, 即不使用。
 - 证书类型: SSL 使用的证书类型, 默认为 JKS。
 - RC4 加密算法: 默认不开启 RC4 协议。
 - SSL&TLS Ciphers: 可以使用的加密算法列表, 用逗号分开。
 - 证书路径: 证书所在路径, 默认为 conf/server.keystore。

- 证书密码：证书的密码，默认为一个常量，该常量没有具体意义，只是用来保护证书密码不在页面显示还有用来判断是否用户输入了新的密码。
- 信任证书类型：信任证书的类型，用来认证客户端证书。
- 信任证书路径：信任证书的路径。
- 信任证书密码：信任证书的密码。
- 如果通道 io 模式选择 apr 时，会显示
 - openssl：是否开启 openssl 选项。
 - http2：是否使用 http2 协议，该选项只有在开启 openssl 选项下才可选择。使用
- 高级属性：
 - 连接超时：socket 超时时间，默认 60 秒。
 - TCP_NODELAY：设置 ServerSocket 的 TCP_NO_DELAY 属性，多数场景下可提高性能。
 - 内存释放空间：java 堆内存溢出时可以释放空间，默认 1MB。
 - 异步超时时间：servlet3.0 新特性，支持 servlet 的异步处理，默认 10 秒。
 - 请求超时时间：keep-alive 下的超时时间，默认 60000ms。在这个时间内没有新的请求，则断开连接。
 - 最大长连接请求数：keep-alive 模式下允许的最大请求数，默认为 100。
 - 最大连接数：服务器在任何给定时间将接受和处理的最大的连接数，默认为 10000。
 - 处理器缓存数量：协议处理器通过缓存处理器对象来提高性能，表示有多少对象被缓存，如果为-1 则无限制，默认为 200。
- 线程池属性：
 - 最大线程数：连接器可创建的最大线程数，一个线程处理一个请求，默认为 200。
 - 初始线程数：最小备用线程，即启动时初始化的线程，默认是 10。
 - 等待队列：指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个数的请求将拒绝连接。默认为 100。
 - 线程优先级：JVM 中请求处理线程的优先级，默认为 5。
- 压缩属性：
 - 压缩：是否开启压缩，默认为不使用 压缩。
 - 压缩类型：压缩时需要用到的 MIME 类型列表。
 - 压缩内容最小值：启用压缩的输出内容大小，默认 2048 (Byte)。
 - 排除的浏览器：正则表达式，用于匹配 user-agent Header 指定哪些 HTTP clients 不使用压缩。
- 其他属性：
 - 禁用的 HTTP 请求方法：要禁用的 HTTP 请求方法。
 - 上传超时时间：是否为数据上传指定更长的连接超时时间。
 - URL 编码格式：用于解码 URI 字符的编码格式，默认 GBK。
 - parse-body-methods：用于 rest，默认支持 GET, POST。
 - uri 处理：如果 ContentType 中指定了编码规范，则可以不使用 URL 编码格式，默认不开启。
 - 虚拟主机：基于 ip 的虚拟主机。
 - DNS 反向查找：通过 ip 查找主机名称。
 - Referer 头验证：开启验证 HTTP Rererer 请求头，不被允许的 Referer 请求将被禁止，返回 HTTP 状态码 403，默认不开启。开启后可填写允许的主机和允许的 IP 地址。如果允许的主机和允许的 IP 地址都为空，将禁止所有的 Referer 请求，但是来自服务器本机的 Referer 请求仍然可以处理。
 - 允许的主机：开启 Referer 头验证时，允许的主机名称，支持通配符*和?，可以逗号分隔。

- 允许的 IP 地址：开启 Referer 头验证时，允许的 IP 名称，支持通配符 *，可以逗号分隔。
- 5. Http(s)通道同时提供了用户自定义配置其他属性的功能，可配置的属性包括：
 - maxHeaderCount 容器允许的请求头个数的最大值。如果一个请求包含的请求头大于指定的最大值，则此请求将被拒绝。
 - maxConnections: 服务器在给定时间内接收并处理的最大连接数。当连接数达到上限，服务器不再接收新的连接。操作系统可以根据 acceptCount 接收连接。Bio 默认为最大线程数 10000，Nio 为 10000。
 - maxTrailerSize: 限制一个分块的 HTTP 请求中的最后一个块的尾随标头的总长度。如果该值是-1，没有限制的被强加。如果没有指定，默认值是 8192。
 - restrictedUserAgents 值为正则表达式，用于匹配 user-agent Header 指定哪些 HTTP clients 不使用 keep alive。默认为空字符串。
 - server: http 响应的 server 头，如不指定则使用应用指定的，如应用也没有指定则使用 webservice。
 - socketBuffer: 设 Socket 输出缓冲区的大小（以字节为单位），-1 表示禁止缓冲，默认值为 9000 字节。
- 6. 设置属性完成后，点击“完成”按钮；或者在设置完高级属性后，直接点击“完成”按钮。

如果创建 AJP 通道，具体的操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“AJP 通道管理”节点；



图 4. 4. 3 AJP 通道列表

3. 在出现的“AJP 通道管理”页面中点击“创建 AJP 通道”按钮，出现如图 4. 4. 4 所示的“创建 AJP 通道”的页面。



图 4.4.4 创建 AJP 通道

- 基本属性
 - ajp 通道名称: AJP 通道的唯一标识, 新建通道名称不能与虚拟主机名称、已使用的 HTTP、AJP 通道名称相同。
 - 监听地址: AJP 通道的监听地址。
 - 监听端口: AJP 通道的监听端口。
 - Ajp secret: AJP 通道的协议密码。
 - 重定向端口: AJP 通道的重定向端口。
 - io 模式: nio 与 nio2, 如果当前系统支持 apr, 则可选 apr。默认为 nio 模式。
 - 代理服务器的 URL: 代理服务器名和端口组成的 URL。
 - xpowered-by: 用于设置是否在 response 的 http 头里生成 X-Powered-By 信息。
 - 高级属性
 - 连接超时: socket 超时时间, 默认 60 秒。
 - TCP_NODELAY: 设置 ServerSocket 的 TCP_NO_DELAY 属性, 多数场景下可提高性能。
 - 异步超时时间: servlet3.0 新特性, 支持 servlet 的异步处理, 默认 10 秒。
 - 请求超时时间: keep-alive 下的超时时间, 默认 60000ms。在这个时间内没有新的请求, 则断开连接。
 - 处理器缓存数量: 协议处理器通过缓存处理器对象来提高性能, 表示有多少对象被缓存, 如果为-1 则无限制, 默认为 200。
 - 线程池属性:
 - 最大线程数: 连接器可创建的最大线程数, 一个线程处理一个请求, 默认为 200。
 - 初始线程数: 最小备用线程, 即启动时初始化的线程, 默认是 10。
 - 等待队列: 指定当所有可以使用的处理请求的线程数都被使用时, 可以放到处理队列中的请求数, 超过这个数的请求将拒绝连接。默认为 100。
 - 线程优先级: JVM 中请求处理线程的优先级, 默认为 5。
 - 其他属性:
 - URL 编码格式: 用于解码 URI 字符的编码格式, 默认 GBK。
 - parse-body-methods: 用于 rest, 默认支持 GET, POST。
 - uri 处理: 如果 ContentType 中指定了编码规范, 则可以不使用 URL 编码格式, 默认不开启。
 - 虚拟主机: 基于 ip 的虚拟主机。
 - DNS 反向查找: 通过 ip 查找主机名称。
4. 设置属性完成后, 点击“完成”按钮; 或者在设置完高级属性后, 直接点击“完成”按钮。

说明:

在 TongWeb 中, 所有 AJP 通道的默认虚拟主机都为 server, 同时虚拟主机 server 会绑定所有的 AJP 通道。在 AJP 通道的创建与编辑界面, 不提供设置其默认虚拟主机的入口。在虚拟主机创建与编辑界面, 在设置要绑定的通道时, 可选的通道列表不包含 AJP 通道。即不可以修改这两者的对应关系。同时, 在虚拟主机列表中, 虚拟主机 server 关联的通道中也不显示绑定的 AJP 通道, 请注意。

4.5.2.2 查看/编辑通道

查看/编辑 HTTP/HTTPS 通道, 具体操作步骤如下:

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点;
2. 点击“HTTP 通道管理”节点;

3. 在出现的“HTTP 通道管理”页面中点击需要查看/编辑的 HTTP 或 HTTPS 通道名称，出现如图 4.4.5 所示页面；
4. 各属性与创建 HTTP/HTTPS 通道时相同。
5. 编辑属性完成后，点击“保存”按钮。

返回
编辑HTTP通道 tong-http-listener

http通道名称	<input type="text" value="tong-http-listener"/>	通道名称，唯一的标识
http通道类型	<input checked="" type="radio"/> http <input type="radio"/> https	通道类型，选择https后需设置SSL属性
监听地址	<input checked="" type="radio"/> 全部 <input type="radio"/> 指定IP	指定IP
	<input type="text"/>	
* 监听端口	<input type="text" value="8080"/>	监听端口号
* 默认虚拟主机	<input type="text" value="server"/>	默认虚拟主机
重定向端口	<input type="text" value="8443"/>	重定向端口
io模式	<input type="text" value="nio"/>	io模式
代理服务器URL	<input type="text"/>	代理服务器的url，格式为ip:port
X-Powered-By	<input type="checkbox"/> 开启	启用后，在响应header中有如，X-Powered-By: Servlet/3.0 JSP/2.2

高级属性

连接超时	<input type="text" value="60000"/>	socket超时（网络连接超时），默认60000ms，以毫秒为单位
TCP_NODELAY	<input checked="" type="checkbox"/> 开启	设置ServerSocket的TCP_NO_DELAY属性，多数场景下可提高性能
内存释放空间	<input type="text" value="1048576"/>	java堆内存溢出时可以释放空间，默认1048576(1MB),以字节为单位
异步超时时间	<input type="text" value="10000"/>	servlet3.0新特性，支持servlet的异步处理，默认10000ms，以毫秒为单位
请求超时时间	<input type="text" value="60000"/>	keep-alive下的超时时间，默认60000ms，以毫秒为单位。在这个时间内没有新的请求，则断开连接
最大长连接请求数	<input type="text" value="100"/>	keep-alive模式下允许的最大请求数，默认为100
处理器缓存数量	<input type="text" value="200"/>	协议处理器通过缓存处理器对象来提高性能，表示有多少对象被缓存，如果为-1则无限制，默认为200

线程池属性

最大线程数	<input type="text" value="200"/>	连接器可创建的最大线程数，默认为200
初始线程数	<input type="text" value="10"/>	最小备用线程，默认是10
等待队列	<input type="text" value="100"/>	指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个数的请

压缩属性

压缩	<input type="radio"/> 不使用 <input checked="" type="radio"/> 压缩文本数据 <input type="radio"/> 强制压缩	压缩，默认为压缩文本数据
压缩类型	<input type="text" value="3个被选中"/>	压缩时需要用到的MIME类型，列表默认为text/html, text/xml, text/plain
压缩内容最小值	<input type="text" value="2048"/>	启用压缩的输出内容大小，默认2048字节(2k)，以字节为单位
排除的浏览器	<input type="text"/>	正则表达式，用于匹配user-agent Header指定哪些HTTP clients不使用compression

其他设置

禁用HTTP请求方法	<input type="text" value="请选择"/>	要禁用的HTTP请求方法
上传超时时间	<input type="checkbox"/> 开启 <input type="text"/>	超时时间，以毫秒为单位
URL编码格式	<input type="text" value="GBK"/>	用于解码URI字符的编码格式，默认GBK
parse-body-methods	<input type="text" value="POST, PUT, DELETE"/>	用于rest，默认支持POST
uri处理	<input type="checkbox"/> 开启	如果ContentType中指定了编码规范，则可以不使用URL编码格式
虚拟主机	<input type="checkbox"/> 开启	基于IP的虚拟主机
DNS反向查找	<input type="checkbox"/> 开启	DNS反向查找
Referer头验证	<input checked="" type="checkbox"/> 开启	开启验证HTTP Referer请求头，不被允许的Referer请求将被禁止
允许的主机名	<input type="text"/>	Referer请求头中允许的主机名，可以逗号分隔，支持通配符*和?
允许的ip地址	<input type="text"/>	Referer请求头中允许的ip地址，可以逗号分隔，支持通配符*

其他Property属性

属性	值	操作
没有可显示的数据		

图 4. 4. 5 查看/编辑 HTTP 通道

查看/编辑 AJP 通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“AJP 通道管理”节点；
3. 在出现的“AJP 通道管理”页面中点击需要查看/编辑的 AJP 通道名称，出现如图 4. 4. 6 所示页面；
4. 各属性与创建 AJP 通道时相同。
5. 编辑属性完成后，点击“保存”按钮。

AJP通道管理 管理通道

此页显示了已创建的AJP通道，可以对这些通道执行编辑、启动、停止、删除操作或者创建新的通道。

ajp通道名称	<input type="text" value="Testexample1"/>	通道名称，唯一的标识
监听地址	<input checked="" type="radio"/> 全部 <input type="radio"/> 指定IP <input type="text"/>	指定IP
* 监听端口	<input type="text" value="8080"/>	<input checked="" type="checkbox"/>
* ajp secret	<input type="text" value="tw"/>	<input checked="" type="checkbox"/>
重定向端口	<input type="text" value="443"/>	重定向端口
io模式	<input type="text" value="nio"/>	io模式
代理服务器URL	<input type="text"/>	代理服务器的url，格式为ip.port
X-Powered-By	<input type="checkbox"/> 开启	启用后，在响应header中有如，X-Powered-By: Servlet/3.0 JSP/2.2

高级属性	
连接超时	60000 socket超时（网络连接超时），默认60000ms，以毫秒为单位
TCP_NODELAY	<input checked="" type="checkbox"/> 开启 设置ServerSocket的TCP_NODELAY属性，多数场景下可提高性能
异步超时时间	10000 servlet3.0新特性，支持servlet的异步处理，默认10000ms，以毫秒为单位
请求超时时间	60000 keep-alive下的超时时间，默认60000ms，以毫秒为单位。在这个时间内没有新的请求，则断开
处理器缓存数量	200 协议处理器通过缓存处理器对象来提高性能，表示有多少对象被缓存，如果为-1则无限制，默认

线程池属性	
最大线程数	200 连接器可创建的最大线程数，默认为200
初始线程数	10 最小备用线程，默认是10
等待队列	100 指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，超过这个

其他设置	
URL编码格式	GBK 用于解码URI字符的编码格式，默认GBK
parse-body-methods	POST 用于rest，默认支持POST
uri处理	<input type="checkbox"/> 开启 如果ContentType中指定了编码规范，则可以不使用URL编码格式
虚拟主机	<input type="checkbox"/> 开启 基于IP的虚拟主机
DNS反向查找	<input type="checkbox"/> 开启 DNS反向查找

图 4.4.6 查看/编辑 AJP 通道

4.5.2.3 启动/停止通道

启动/停止 HTTP/HTTPS 通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“HTTP 通道管理”节点；
3. 选中需要启动/停止的通道，点击“启动”/“停用”按钮。

启动/停止 AJP 通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“AJP 通道管理”节点；
3. 选中需要启动/停止的通道，点击“启动”/“停用”按钮。

4.5.2.4 删除通道

删除 HTTP/HTTPS 通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“HTTP 通道管理”节点；
3. 选中待删除的通道；
4. 点击“删除”按钮。

删除 AJP 通道，具体操作步骤如下：

1. 展开管理控制台左侧导航树中的“Web 容器配置”节点；
2. 点击“AJP 通道管理”节点；
3. 选中待删除的通道；
4. 点击“删除”按钮。

4.5.3 配置使用说明

4.5.3.1 通道重定向

通道重定向功能是当通道接收到非 SSL 的请求，且应用的 web.xml 中 security-constraint 部分定义该请求应通过 SSL 通道访问，则将请求重定向到 HTTPS 通道。

说明：如果通道接收到非 SSL 请求，应用的 web.xml 中 security-constraint 部分定义该请求应通过 SSL 通道访问，且未配置通道重定向端口，此时服务器默认提供 HTTPS 通道的端口用于重定向（即使没有配置通道重定向端口）。默认提供的 HTTPS 通道为第一个创建的 HTTPS 通道。如果没有 HTTPS 通道可用则返回 403（服务器拒绝请求）错误。

4.5.3.2 HTTP 通道的重定向

1. 分别创建一个 HTTP 通道（假设端口为 8008）和 HTTPS 通道（假设端口为 8443）。要求两个通道具有相同的监听地址，且 HTTP 通道的重定向端口与 HTTPS 通道的监听端口相同。创建 HTTP/HTTPS 通道的步骤见“[创建通道](#)”。
2. 将步骤 1 中创建的 HTTP 通道和 HTTPS 通道均绑定到虚拟主机上（如默认虚拟主机 server）。
3. 将应用部署到虚拟主机上（需要与通道绑定的虚拟主机一致）。
4. 通过 HTTP 通道（`http://localhost:8008/sec`）访问该应用，则请求被重定向到 HTTPS 通道（`https://localhost:8443/sec`）。

4.5.3.3 Proxy 转发

1. Proxy 转发概述

Proxy 转发模式下，TongWeb7 能够接收 Web Server 作为 proxy 转发给其的请求，从而获取更优的性能和安全性。使用 Proxy 转发的场景可以使用 THS、Apache 处理静态内容，而 Servlet 和 JSP 等动态请求则交给 TongWeb7 应用服务器处理。对于最终用户（浏览器）而言，转发给 TongWeb7 的请求的应答就如同来自 THS、Apache 发出的应答一样。

1. TongWeb7 支持以下 Proxy 转发方式
 - 基于 HTTP (S) 协议的 Proxy 转发
 - 基于 AJP 协议的 Proxy 转发
2. TongWeb7 支持 THS、Apache 作为 Proxy Server

将 THS 作为 Proxy Server 的使用步骤

1. 启动 TongWeb7 服务器；
2. 通过管理控制台创建 HTTP 或 HTTPS 通道，配置“代理服务器的 URL”为 `http(s)://proxyName:proxyPort`（proxyName 为 THS 的地址，proxyPort 为 THS 的监听端口），创建 HTTP/HTTPS 通道的步骤见“[创建通道](#)”；
3. 将步骤 2 中创建的通道绑定到缺省虚拟主机（server）；
4. 将应用部署到缺省虚拟主机 server 上。
5. 在 THS 中配置 Proxy 转发的信息，配置方法如下：

2. 配置基于 HTTP 的 proxy 转发

前提：THS 使用 HTTP 协议的默认配置；（具体用法参考《TongHttpServer 用户手册》）

1. 将 `<THS_Install_Dir>/bin/https.conf` 中 ProxyPass 和 ProxyPassReverse 注释掉；

```
#ProxyPass / balancer://tongSSLCluster/ stickysession=ROUTEID growth=100
#ProxyPassReverse / balancer://tongSSLCluster/
```

2. 在 `<THS_Install_Dir>/bin/https.conf` 中相应配置内容如下：

```
<VirtualHost _default_:8080>
//8080 是 THS HTTP 监听端口
.....
ProxyPass /test1 http://168.1.103.13:9003/test1
ProxyPassReverse /test1 http://168.1.103.13:9003/test1
//其中：168.1.103.13 是 TongWeb7 HTTP 通道的监听地址，9003 为 HTTP 通道的监听端口。
```

3. 重启 THS（`sh start.sh restart`）
4. 通过“`http://THSAddress:Port/test1`”（THSAddress 是 THS 的 IP 地址，Port 是 THS 的 HTTP 监听端口）成功访问 test1 应用。

3. 配置基于 HTTPS 的 Proxy 转发

前提：THS 使用 HTTPS 协议的默认配置；（具体用法参考《TongHttpServer 用户手册》）

1. 将 `<THS_Install_Dir>/bin/https.conf` 中 ProxyPass 和 ProxyPassReverse 注释掉；

```
#ProxyPass / balancer://tongSSLCluster/ stickysession=ROUTEID growth=100
```

```
#ProxyPassReverse / balancer://tongSSLCluster/
```

2. 在<THS_Install_Dir>/bin/https.conf 中相应配置内容如下:

```
<VirtualHost _default_:443>
```

```
//443 是 THS 中 HTTPS 监听端口
```

```
.....
```

```
ProxyPass /test1 https://168.1.103.13:9003/test1
```

```
ProxyPassReverse /test1 https://168.1.103.13:9003/test1
```

```
//其中: 168.1.103.13 是 TongWeb7 HTTPS 通道的监听地址, 9003 为 HTTPS 通道的监听端口。
```

3. 重启 THS (sh start.sh restart);

4. 通过 “https://THSAddress:Port/test1” (THSAddress 是 THS 的 IP 地址, Port 是 THS 的 HTTPS 监听端口) 成功访问 test1 应用

4.6 虚拟主机与通道的关系

虚拟主机与通道之间是多对多的关系, 即一个虚拟主机可以与多个通道关联, 一个通道可以与多个虚拟主机关联。

注意:

在使用虚拟主机时, 首先应该确认操作系统的 hosts 表中, 是否正确的配置了虚拟主机名与 IP 地址的映射, 例如: admin 10.10.4.10

如果未配置主机名与 IP 地址的映射, 则需要手工添加。要注意 hosts 表中配置的主机名必须和在管理控制台中配置虚拟主机时填写的“主机名”一致。

使用虚拟主机访问应用时, 除了要在 URL 中指定应用的访问前缀 (contextRoot), 还要指定虚拟主机 (virtualServer) 和通道 (port), 形如:

http://virtualServer:port/contextRoot。在使用上述方式访问应用时, 存在以下两种使用情况:

1. 使用指定虚拟主机 virtualServer 访问应用:

访问形式: http://virtualServer:port/contextRoot

成功访问应用条件:

- 1) 虚拟主机 virtualServer 和通道 port 关联 (所谓关联, 就是指创建虚拟主机 virtualServer 时指定了通道 port 为访问通道)。
- 2) 访问的应用 (contextRoot 代表的) 部署在虚拟主机 virtualServer 上。

2. 使用默认虚拟主机 (defaultVs) 访问应用:

访问形式: http://virtualServer:port/contextRoot

创建通道 port 时, 可以指定一个虚拟主机 (defaultVs) 作为默认虚拟主机。

成功访问应用条件:

- 1) 虚拟主机 virtualServer 和通道 port 不关联, 这种情况下便会使用默认虚拟主机 defaultVs

访问的应用 (contextRoot 代表的) 部署在虚拟主机 defaultVs 上。

4.7 资源

4.7.1 文件集

文件集功能是维护管理共享库需要的 JAR 文件, 支持客户端上传或服务器选择, 服务器选择时可为 JAR 文件或文件夹, 如图:

文件集管理 显示隐藏详细信息

此页显示了已添加的文件列表(系统内置文件列表consolejars除外)，可以更新、删除所添加文件或者添加新的文件。

<input type="checkbox"/>	文件集名称	文件来源	文件状态	上传时间
<input type="checkbox"/>	ant-1.9.7.jar	客户端	正常	2020-01-02 10:54:33
<input type="checkbox"/>	ant-1.9.4.jar	客户端	正常	2020-01-02 10:54:27
<input type="checkbox"/>	spring-boot	服务器	正常	2020-01-02 14:24:47
<input type="checkbox"/>	filet-kafka-2.3.0.jar	服务器	正常	2020-01-02 14:41:01
<input type="checkbox"/>	settings-eclipse.jar	服务器	正常	2020-01-02 14:42:12

列表：包含文件集名称，文件来源，文件状态，上传时间，其中文件来源取值“客户端”和“服务器”，文件状态表示文件集对应的文件是否存在，取值“正常”和“文件不存在”。

新增：文件集名称不能重复，默认取文件名称，客户端方式不支持修改名称，服务器方式支持修改，使用服务器方式如果选择文件夹且包含 class 文件，请确保文件路径和包名一致，客户端方式上传的 JAR 文件位于 lib/s1 目录下。

查看：显示资源文件的相关信息，如果是文件夹，则可以查看文件夹下面的所有 JAR 包或 class 文件根目录。

修改：只能修改描述信息。

此页显示了已添加的文件列表，可以更新、删除所添加文件或者添加新的文件。

返回

文件上传

文件来源 服务器

文件集名称 classes

文件位置 /Users/yincrc/Desktop/classes/ [隐藏文件](#)

共2个文件

- 1 /Users/yincrc/Desktop/classes/test.jar
- 2 /Users/yincrc/Desktop/classes(.class文件)

描述

文件集的备注信息

删除：不能删除被共享库引用的文件集，应先接触引用关系。如果文件来源是客户端，则会同时删除上传的文件，如果是服务器，则只会删除引用，不删除服务器源文件。

4.7.2 共享库

共享库功能是维护管理应用需要的类库集合，依赖于文件集，可以配置公有库和私有库，一个文件集可以被多个共享库引用，一个共享库也能被多个应用使用，如图：

此页显示了已经创建的共享库(系统内置共享库consolejars除外)，可以对这些共享库进行更新、删除操作或者创建新的共享库。

共享库名称	共享类型	更新时间	创建时间
ant194	私有	2020-01-02 14:23:21	2020-01-02 10:54:58
boot	私有	2020-01-02 14:24:56	2020-01-02 14:24:56
ant197	私有	2020-01-02 14:40:26	2020-01-02 10:54:48

列表：包含共享库名称，共享类型，更新时间，创建时间。

新增：共享库名称不能重复，资源文件必选，左侧为全部文件集，右侧为所选文件集，可以双击列表中的文件来选择文件，也可以通过中间的按钮来操作。

返回
创建共享库

* 共享库名称

共享类型 公有

资源文件

未选择资源

搜索

- ant-1.9.7.jar
- ant-1.9.4.jar
- spring-boot
- fillet-kafka-2.3.0.jar
- settings-eclipse.jar

→

→→

←

←←

已选择资源

搜索

-

↑

↓

共享库名称，唯一标识名

共享库的共享类型

共享库包含的资源文件

共享库的描述信息

保存
取消

查看：共享库的相关信息。

修改：修改共享库类型、资源文件以及描述，如果修改了类型或者资源文件且应用被应用，则会提示是否确认修改，受影响的应用需要重新部署才能使用新的共享库。如果引用的文件集为文件夹，增加或者修改 JAR，则需要修改共享库才能生效。

删除：不能删除被应用引用的共享库，应先接触引用关系。

4.7.2.1 共享库类型

a) 公共共享库：

新建公共共享库后，内存中会创建一个包含此文件列表的 `ClassLoader`。多个应用依赖此公共共享库时，对同一个 `class` 只会加载一次。并且应用对共享库中可访问的静态变量修改是相互可见的。

b) 私有共享库：

新建私有共享库后，内存中会创建一个包含此文件列表的 `ClassLoader`。在应用依赖此私有共享库后，会把 `ClassLoader` 中包含的 `jar` 添加到应用级的 `ClassLoader` 中。多个应用同时依赖一个私有共享库时，对同一个 `class` 会加载多次。并且共享库中可访问的静态变量是相互独立的。

4.8 类加载

4.8.1 类加载机制

Java 应用程序运行时，在 class 执行和被访问之前，它必须通过类加载器加载使之有效，类加载器是 JVM 代码的一部分，负责在 JVM 虚拟机中查找和加载所有的 Java 类和本地的 lib 库。JVM 和 TongWeb7 应用程序服务器提供了多种不同的类加载器，形成一个具有父子关系的分层结构。

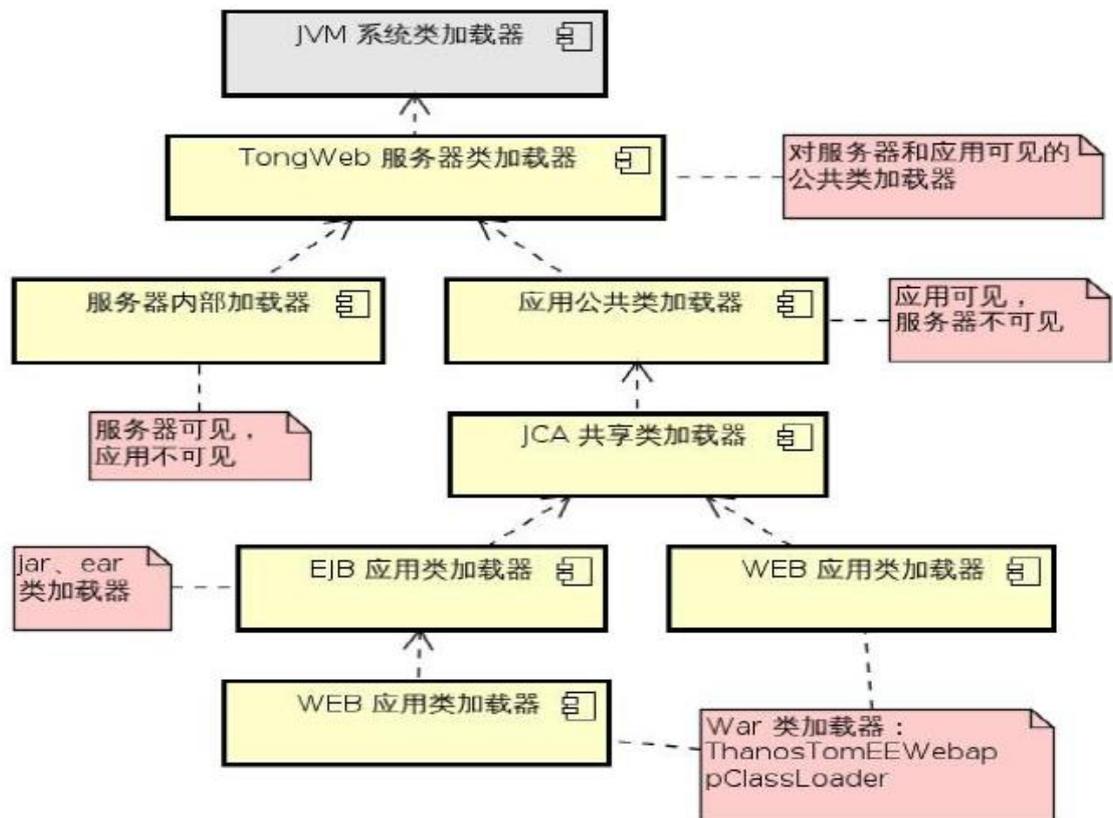


图 4.6.1 类加载器分层结构图

如上图所示，TongWeb7 中类加载器被组织成一个自上而下的层次结构，最上层是系统的运行环境 JVM 类加载器，最下层是具体的应用程序类加载器，上下层之间形成父子关系。

TW_HOME/conf/tongweb.properties 配置文件中定义了类加载器所需要的类路径，服务器启动时根据此配置文件创建好各个层级的 ClassLoader。

- Bootstrap ClassLoader: JVM 自带的系统类加载器，负责加载核心 Java 类库
- Launcher\$ExtClassLoader: 负责加载用户需要扩展 JVM 核心平台时所引用的类，查找的 url 由 java.ext.dirs 指定，默认为<JAVA_HOME>/jre/lib/ext
- Launcher\$AppClassLoader: 负责加载服务器启动所需的类，查找的 url 由-classpath 指定。
- TongWeb 服务器类加载器: 加载类的路径对应为 TW_HOME/lib 目录，是 TongWeb server 及应用都可以加载到的公共的类加载器。
- 服务器内部类加载器: 服务器内部 Server、Service 等核心类的类加载器。
- 应用公共类加载器: 加载类的路径对应为 TW_HOME/lib/common、及 TW_HOME/lib/classes 目录，部署的应用可以共享此目录中的类。
- JCA 共享类加载器: 资源适配器共享，对所有应用可见。
- WEB 应用类加载器: 负责加载独立的 web 模块中的 servlets 和其它类以及 web 模块中包含的 ejb 类。每个 web 模块会创建一个 Web 类加载器，用于加载 WEB-INF/classes 中的类和 WEB-INF/lib 中的 jar 包。

- EJB 应用类加载器：负责加载 EAR 应用中所包含的公共类、ejb 模块和独立 ejb 应用下所有的类。
- EAR 应用中的 WEB 应用类加载器：负责加载 EAR 应用中所包含的 web 模块中的 servlets 和其它类，加载路径是 WEB-INF/classes 中的类和 WEB-INF/lib 中的 jar 包。

说明：

- 每个类加载器负责在自身定义的路径上查找和加载类。
- 一个子类加载器能够委托它的父类加载器查找和加载类，一个加载类的请求会从子类加载器发送到父类加载器，但是从来不会从父类加载器发送到子类加载器。
- 一旦一个类被成功加载，JVM 会缓存这个类直至其生命周期结束，并把它和相应的类加载器关联在一起，这意味着不同的类加载器可以加载相同名字的类。
- 如果一个加载的类依赖于另一个或一些类，那么这些被依赖的类必须存在于这个类的类加载器查找路径上，或者父类加载器查找路径上。
- 如果一个类加载器以及它所有的父类加载器都无法找到所需的类，系统就会抛出 ClassNotFoundException 异常或者 NoClassDefFoundError 的错误。

4.8.2 类加载模式

TongWeb7 应用服务器提供两种类加载模式：父优先和子优先。该模式决定了类加载器在查找一个类的时候，是先查找类加载器自身指定的类路径还是先查找父类加载器上的类路径。

web 应用和 ear 下的 ejb 应用支持两种类加载模式，tongweb-web.xml 中的 delegate 属性用于决定应用使用子优先模式还是父优先模式。delegate 默认使用子优先模式。配置的方式为，在应用部署时，可以选择父优先还是子优先模式，控制台的加载模式优先于 tongweb-web.xml 中 delegate 配置。tongweb-web.xml 中 delegate 属性配置优先于其它文件中的 delegate 配置。如下：

```
<tongweb-web-app>
  <loader delegate="true" />
</tongweb-web-app>
```

当 delegate 设置为 true 时是父优先，设置为 false 时是子优先。

各 ClassLoader 默认使用父优先的类加载模式，即当父类加载器找不到所加载的资源时，才使用当前类加载器(子类加载器)。

- 父优先：在加载类的时候，在从类加载器自身的类路径上查找加载类之前，首先尝试在父类加载器的类路径上查找和加载类。

当 ClassLoader 被请求加载某个类时，它首先委托自己的父类加载器去加载，若父类加载器能加载，则返回这个类所对应的类对象，若父类加载器不能加载，才由当前类加载器去加载。若都不能成功，则系统就会抛出 ClassNotFoundException 异常或者 NoClassDefFoundError 的错误。

例如：在 web 应用(.war)中 lib 下的 utility.jar 中存在类 A.class，TW_HOME/lib 目录下的 package.jar 中也存在一个同名的类 A.class。若采用父优先的类加载策略，则当应用级的 ClassLoader 被请求加载类 A.class 时，返回的是其父类加载器加载的 package.jar 中的 A.class 对象，而不是 utility.jar 中的类 A.class。

- 子优先：在加载类的时候，首先尝试从自己的类路径上查找加载类，在找不到的情况下，再尝试父类加载器类路径。

当 ClassLoader 被请求加载某个类时，它首先试图自己去加载，若能加载成功，则返回这个类所对应的 Class 对象，若不能加载成功，才由该类加载器的父类加载器去加载。若都不能成功，则系统就会抛出 ClassNotFoundException 异常或者 NoClassDefFoundError 的错误。

例如：对于前面类加载的示例，使用子优先的类加载策略，则最终被加载的类为 utility.jar 中的 A.class，这个对象由 Application ClassLoader 加载。

4.8.3 类加载推荐策略

在组装企业应用时，唯一性是处理类加载问题的一个很好的指导原则，即相同的类只放置在一个地方，从一种途径加载。以下是 TongWeb7 推荐的策略：

1. EJB 应用自身依赖的类，封装在自身的 EJB JAR 中。
2. Web 应用自身依赖的类，或者以 JAR 包的形式放置于 WAR 中 WEB-INF/lib 目录下，或者以类的形式放置于 WEB-INF/classes 目录下。
3. 在一个 EAR 应用中，多个 EJB 或 Web 模块共同使用的公用包，直接以普通 JAR 包方式放置于 EAR 根目录下。

在一个 EAR 应用中，多个 web 模块中包含类名相同，功能（代码）不同的类，使用子优先模式。避免 EAR 的类加载器选择 Web 模块中的某一个类（取决于 web 模块在部署文件中的顺序）共享给所有 Web 模块使用。

4. 在部署多个应用，这些应用要共享相同类且不需要更多服务配置时，可以选择放在 TongWeb7 默认提供的类共享目录 lib/common 和 lib/classes 目录下。

其中 lib/common 下放 jar 包，classes 下放 class 类。启动服务器后在会把这两个目录的文件加载并创建共享的 ClassLoader。功能方面同 4.6.2 公共共享库类型功能实际是一致的，可以实现多个应用共享类，但所有应用对同一个 class 只会加载一次。并且应用对共享库中可访问的静态变量修改是相互可见的。所以可能会出现不同应用的冲突问题。所以如果想实现每个应用使用共享类文件且 classloader 相互独立应采用私有类型共享库。

5. 部署的应用希望依赖非服务器默认提供的 javaee api 版本，如 annotation-api。可以选择将期望版本的 jar 包放置在 lib\endorsed 中实现 api 替换。

此功能是基于 JAVA 提供的 endorsed 技术，关于 endorsed：可以简单理解为 -Djava.endorsed.dirs 指定的目录面放置的 jar 文件，将有覆盖系统 API 的功能。但是能够覆盖的类是有限制的，其中不包括 java.lang 包中的类（出于安全的考虑）。

使用此参数是因为 java 是采用双亲委派机制进行加载 class 类的。而 jdk 提供的类只能由类加载器 Bootstrap 进行加载。如果你想要在应用程序中替换掉 jdk 中的某个类是无法做到的，所以 java 提供了 endorsed 来达到你想要替换到系统中的类。

4.8.4 类加载参数

在服务器启动和应用部署过程中，有一些参数可供参考。

exclusions.list

位置 TW_HOME/bin/conf 【默认无】

default.exclusion

位置 TW_HOME/lib/tw7.jar 【默认有】

tongweb.properties

位置 TW_HOME/conf 【默认有】

生效范围：

应用服务器会取 **exclusions.list** 中的内容作为启动过程中 jar 包等的过滤列表，如果该列表不存在，则取默认列表即 **default.exclusion** 来作为基本的过滤列表。在启动过程中，还会读取 **tongweb.properties**，取其 **tongweb.util.scan.StandardJarScanFilter.jarsToSkip** 的值加入到过滤列表中。在随后的加载过程中，会将环境变量中的 **tongweb.util.scan.StandardJarScanFilter.jarsToSkip** 设置为上述列表值。

上述参数描述了应用服务器在启动和应用部署过程中过滤掉的类和 jar 包的列表

4.9 其他

4.9.1 设置 Session Cookie 的名字

可以对每个 web 应用设置单独的 Session Cookie 名字，方法如下：

在 tongweb-web.xml 文件的根节点下加入如下内容：

```
<session>
```

```
  <cookie-properties>
```

```
    <property name="cookieName" value="MySESSIONID" />
```

```
</cookie-properties>
</session>
```

其中 cookieName 为固定的属性名，其属性值就是 Session Cookie 的名字，如果不设置这个属性，将使用默认值：JSESSIONID。

4.9.2 应用上下文共用 Session

对于跨上下文的访问，在 TongWeb7 中需要在启动脚本中配置参数 -Dopenejb.crosscontext=true 即可。

4.9.3 多应用 Session 共享

在某些应用场景中，需要服务器提供多应用间 Session 共享机制，在 TongWeb7 中需要在启动脚本中配置参数 -DSharedSessionEnabled=true 即可。

4.9.4 虚拟目录

功能说明：

应用中的 JSP,html 和静态资源可以在放在虚拟目录中(本地任意目录)，对于 JSP,html 和静态资源的加载优先级，如下：

- 1、war 中不存在，虚拟目录下存在，用的是虚拟目录下的；
- 2、war 中存在，虚拟目录下不存在，用 war 中的；
- 3、war 包和虚拟目录下都存在并且同名的话，用的是虚拟目录下的文件；

该功能仅限 JSP,html 和静态资源，JSP 引用的 class 需要在应用的类路径下；

使用方式：

在 tongweb-web.xml 文件的根节点下加入如下内容：

```
<property name="aliases" value="/aliasPath1=docBase1,/aliasPath2=docBase2"/>
```

如果应用前缀为“/”，则配置如下：

```
<property name="aliases" value="/=D:\virtualdir"/>
```

说明：aliasPath1 指 http 请求 URL 中该资源的访问路径；docBase1 是资源所在的绝对目录。

如果有多个虚拟目录需要指定，将多个/aliasPathN=docBaseN 用逗号隔开即可。

举例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<tongweb-web-app>
```

```
<property name="aliases" value="/images=D:\Work\mdir\images,/script=D:\Work\mdir\script,/pages=D:\Work\mdir\pages,/css=D:\Work\mdir\css"/>
```

```
</tongweb-web-app>
```

比如某应用的静态图片的访问 url 为 <http://ip:port/appname/images/code.gif>，那么其虚拟目录可以配置为/images=D:\Work\mdir\images，其中/images 是请求 URL 中该资源的访问路径，D:\Work\mdir\images 是存放该资源的绝对路径。同理/script 下可以放置 js 资源，/pages 下可以放置 jsp 资源，/css 下可以放置 css 文件。

5 JDBC 配置

5.1 JDBC 数据源概述

JDBC 数据源的主要功能是为应用程序提供数据库连接。JDBC 数据源基于数据库连接池技术，因此连接复用便是 JDBC 数据源的基本功能。每个 JDBC 数据源使用一个连接池维护一定数量的连接。连接池预先建立多个数据库连接对象，然后将连接对象保存到连接池中，当客户请求到来时，从池中取出一个连接对象为客户服务，当请求完成时，客户程序调用 close() 方法将连接对象放回池中。

通过连接复用减少了创建数据库连接的次数，提高了服务器的性能。

5.2 TongWeb7 中的 JDBC 数据源概述

TongWeb7 中的 JDBC 数据源于易于适配到多种数据库上。JDBC 连接池连接池维护特定

数据库的一组可重复使用的连接。由于每创建一个新的物理连接都会耗费时间，因此服务器维护了可用连接池以提高性能。应用程序请求连接时可以从池中获取一个连接。应用程序关闭连接时，连接将返回到池中。

TongWeb7 分别提供 `javax.sql.DataSource`、`javax.sql.XADataSource` 接口的数据源实现。用户创建 JDBC 资源后，服务器会创建一定数量的连接，并将其放到连接池中，应用通过 JDBC 资源的 JNDI 获取数据源对象，再通过数据源对象从连接池中获取数据库连接，然后使用数据库连接进行一系列数据库操作，连接使用完成后应用调用连接对象的 `close()` 方法将连接返回连接池中供下次使用。用户可创建单数据源和多数据源两类 JDBC。

JDBC 单数据源与多数据源区别：

单数据源用单一数据库为用户提供数据库服务，多数据源为用户提供单一接入点，这样用户访问这个单一入口，即可对多个数据库进行协作管理，实现负载均衡与故障转移。

XA 与非 XA 的区别：

JTA(Java Transaction API) 为 J2EE 平台提供了分布式事务服务。如果计划使用 JTA 来划分事务，用户将需要一个实现了 `javax.sql.XADataSource`、`javax.sql.XAConnection` 和 `javax.sql.XAResource` 接口的 JDBC 驱动。实现了这些接口的驱动将有能力参与到 JTA 事务中。一个 `XADataSource` 对象是一个 `XAConnection` 对象的工厂。`XAConnection` 是参与到 JTA 事务中的连接。

XA 连接与非 XA 连接的区别是：XA 连接是一个 JTA 事务中的参与者，即 XA 连接不支持 JDBC 的自动提交特性，应用程序不必在 XA 连接上调用 `java.sql.Connection.commit()` 或 `java.sql.Connection.rollback()`，此时，应用程序应该使用 `UserTransaction.begin()`、`UserTransaction.commit()` 和 `UserTransaction.rollback()`。

5.2.1 连接池管理功能描述

为保证连接池的性能，服务器提供连接池的管理功能，主要包括定时处理空闲连接、泄漏连接、检测连接的有效性、负载均衡、故障转移等。

5.2.2 空闲超时的处理

空闲连接即连接池中没有被使用的连接。开启空闲超时功能，当连接池中存在空闲的连接数大于连接池的最小连接数时（初始化连接数）时，服务器以用户配置的“检查连接的周期”时间为周期对连接池中的空闲连接进行检查，主要检查空闲连接是否超时。

具体检查步骤如下：

1. 如果连接空闲的时间超过用户配置的“空闲超时”时间，将从连接池中删除。
2. 检查连接池中的连接数，如果连接数小于用户配置的“最小连接数”，则创建新连接并放到连接池中，使连接池中的连接数达到用户配置的最小值。

5.2.3 泄露连接的处理

泄露连接即被应用占用时间过长的连接（具体时间为用户配置的“泄漏超时”时间）。服务器提供泄露连接回收的功能。具体回收步骤如下：

开启泄漏回收功能，如果某个连接被应用使用时间已经超过用户配置的“泄漏超时时间”，则将泄露连接销毁。

连接销毁之后，检查连接池中的连接数，如果连接数小于用户配置的“最小连接数”，则创建新连接并放到连接池中，使连接池中的连接数达到用户配置的最小值。

5.2.4 获取连接的处理

如果在用户配置的“等待超时时间”内没有获取到连接，将打印用户获取连接超时信息。

5.2.5 连接有效性检查

服务器在将连接提供给应用程序之前验证连接的有效性，如果由于网络出现故障或数据库服务器崩溃等原因造成无法获取数据库连接，应用服务器将自动重新建立数据库

连接。该功能会带来一定的性能开销，因此不是每次获取连接时都进行有效性检查。为了减少开销，服务器将按照一定的周期对连接进行有效性检查，除了对连接进行定期的有效性检查外，还提供获取连接时对连接进行有效性检查、创建连接时对连接进行有效性检查、归还连接时进行有效性检查。（获取连接时对连接进行有效性检查、创建连接时对连接进行有效性检查、归还连接时进行有效性检查的使用[见附录 2.2.1 连接池基本配置](#)）

连接有效性检查的条件如下：

- 开启连接有效性检查功能；
- 配置“测试连接的 SQL 语句”或者用户自定义的验证类；
- 用户配置的“检查连接的周期”大于 0，且从上一次检查至今，已经超过了配置的“连接验证时间间隔”；
- 打印验证信息（可选条件），验证失败后打印失败信息；

具体检查步骤如下：

1. 检查连接是否有效，如果连接无效，将从连接池中删除。
2. 检查连接池中的连接数，如果连接数小于用户配置的“最小连接数”，则创建新连接并放到连接池中，使连接池中的连接数达到用户配置的最小值。

5.2.6 动态驱动路径加载

用户可以自定义数据库驱动路径，无需将驱动包路径放在服务器 lib 包下也能被加载到。

5.2.7 语句缓存

连接池支持为 PreparedStatement 和 CallableStatement 语句进行缓存，当程序再次调用一个连接的 PreparedStatement 或 CallableStatement 方法时，直接从缓存中获取，无需再创建新的 PreparedStatement 或 CallableStatement 对象，可以提高性能。

5.2.8 故障转移

多数据源提供了一个用于满足连接请求的数据源有序列表。通常情况下，每一个对多数据源发出的连接请求，都由该列表中的第一个数据源提供服务。如果数据源出现连接异常、未能通过连接测试，并且该连接无法被替换，或者该数据源已挂起，该数据源将会被标记为疑似异常状态，禁止提供服务，待服务器执行多数据源状态检查。若被禁止的数据源为最高优先级数据源，则会从该列表中选举出下一个优先级较高且有效的数据源提供服务。当故障数据源恢复连接时，将恢复到由优先级较高的第一个数据源提供服务。

5.2.9 负载均衡

从单数据源列表中，选定多个数据源，组成多数据源，对用户请求进行负载均衡服务。用户发出连接请求时，多数据源使用循环法，依次选择列表中的数据源提供服务。当连接出现异常、未能通过测试，或者该数据源已挂起时，该数据源会被标记为待检查状态，禁止提供服务。切换至列表中的其他数据源循环提供服务。

5.2.10 多数据源状态检查

服务器定时对多数据源包含的单数据源进行连接有效性检查，当检查到其中某个数据源发生故障时，服务器会对数据源中的所有闲置连接进行验证，当验证结果为所有闲置连接无效，且数据源中无活动连接时，将对数据库进行连接重置。重置失败，或者重置后的连接认证为无效，则被标记为禁用，待下次检查。

服务器会定期多次对多数据源中禁用的数据源进行状态检查，如果该数据源通过检查，则该数据源将变为可用，并恢复向该数据源路由连接请求。检查频率是由该多数据源的“测试频率”特性控制，默认 120 秒。

5.3 JDBC 数据源的使用

5.3.1 创建数据源连接池

1. 依次展开管理控制台左侧导航树中的“JDBC 配置”节点，然后点击“JDBC 配置”节点；如图 5.3.1 所示：



图 5.3.1 连接池列表

2. 在出现的 JDBC 连接池列表页中点击“创建连接池”按钮，出现如图 5.3.2 所示的创建 JDBC 连接池基本属性页面；



图 5.3.2 创建 JDBC 连接池-基本属性

- 属性设置
- 名称：连接池（数据源 JNDI）的名称，连接池的唯一标识。
- Ip 类型：ipv4, ipv6
- 资源类型：可选值为 `javax.sql.DataSource`、`javax.sql.XADataSource`。
- 数据库类型：数据库 Driver 类型名称。
- 如果资源类型为 `javax.sql.DataSource`，则数据库类型的下拉菜单中可选的数据库 Driver 类型名有：
 - Sybase15 JDBC Driver
 - dameng JDBC Driver
 - Kingbase JDBC Driver
 - Oscar JDBC Driver
 - HSQL JDBC Driver
 - Sybase type 4 Driver for Sybase 11.x
 - postgresSQL type 4 Driver for postgresQ
 - Oracle OCI Driver for Oracle
 - Oracle Type 4 Driver for Oracle
 - MM.MySQL Type 4 Driver for MySQL
 - MySQL Connector/J Type 4 Driver for MySQL

- Microsoft Type 4 Driver for MS SQL Server
- Derby Client Driver
- Derby Embedded Driver
- IBM Type 4 Driver for Informix
- IBM Type 4 Driver for DB2
- IBM Type 2 Driver for DB2
- Microsoft Type 6 Driver for MS SQL Server
- GaussDB 100
- 如果资源类型为 javax.sql.XADataSource，则数据库类型的下拉菜单中可选的数据库 Driver 类型名有：
 - Sybase type 4 XA Driver for Sybase 12.x
 - Oracle OCI XA Driver for Oracle
 - Oracle Type 4 XA Driver for Oracle
 - Microsoft Type 4 XA Driver for MS SQL Server
 - IBM Type 4 XA Driver for Informix
 - IBM Type 2 XA Driver for DB2
 - Microsoft Type 6 XA Driver for MS SQL Server

- 数据库驱动类名：数据库驱动类的类名，该类位于 JDBC 驱动程序中。
 - 连接 url：连接数据库的 URL，默认提供连接 URL 的格式。
 - 用户名：连接数据库所需的用户名。
 - 密码：连接数据库所需的密码。
 - 连接参数：驱动的连接参数，格式必须是 propertyName=property，多个参数用分号分隔。
 - 驱动路径：数据库驱动的位置路径，TongWeb7 提供数据库驱动程序的动态加载功能。
3. 设置好上面的信息后，点击“下一步”按钮，这时会进行测试连接，如果上面信息填写正确，在页面顶部会提示“数据库连接成功”信息，并进入“创建 JDBC 连接池”池设置页面。如果信息填写不正确，在页面顶部会提示失败信息，点击详情可查看失败原因。

测试连接成功页面如图 5.3.3 所示：

数据库连接成功

管理JDBC连接池 ⓘ

创建连接池

1 池属性 **2 池设置** 3 验证连接属性

初始化连接数	<input type="text" value="10"/>	连接池启动时创建的初始化连接数量，默认10
最大连接数	<input type="text" value="10"/>	连接池在同一时间能够分配的最大活动连接的数量，默认10
空闲连接数	<input type="text" value="10"/>	最小空闲连接数
超时时间	<input type="text" value="30000"/>	获取连接的最大等待时间，以毫秒为单位，默认为30000ms，不能小于2

或者跳过剩下步骤，直接

图 5.3.3 创建 JDBC 连接池-池设置

测试连接失败页面如图 5.3.4 所示：



图 5.3.4 测试失败页面

- 连接池设置
 - 初始化连接数：池中连接的最小数目。该值确定了首次创建池或应用服务器启动时，置于池中的连接的数目，默认值为 10。
 - 最大连接数：连接池在同一时间能够分配的最大活动连接的数量，当池中连接数大于等于连接的最大数时，不再为请求创建连接，而是等待空闲连接，默认 100 。
 - 最小空闲连接数：连接池可以保持的空闲连接最少有多少个，默认值是 10
 - 等待超时时间：获取连接的最大等待时间，以毫秒为单位，默认为 30000ms，配置为 0 表示一直等待。
4. 设置好以上信息之后，点击“下一步”按钮，进入“创建连接池”验证连接属性页面，如果不进入下一步而是在本页面点击“直接完成”，那么验证连接属性的配置全部采用默认值。“创建连接池”验证连接属性页面如图 5.3.5 所示：

返回

创建连接池

1 基本属性	2 池设置	3 验证连接属性
默认测试SQL语句	SELECT 1 from DUAL	用来验证从连接池取出的连接是否能够正常的
重新定义测试SQL表名	<input type="text"/>	不指定则采用默认SQL语句，若指定则按照新
创建连接时验证	<input type="checkbox"/> 开启	指明是否在创建连接时时进行检验
获取连接时验证	<input type="checkbox"/> 开启	指明是否在从池中取出连接前进行检验
归还连接时验证	<input type="checkbox"/> 开启	指明是否在归还到池中前进行检验
验证超时	<input type="text" value="5"/>	测试连接活动的最长时间，以秒为单位，不能
连接验证时间间隔	<input type="text" value="30000"/>	避免过度验证，保证验证不超过这个频率，以 再次验证，默认30000

或者跳过剩下步骤，直接 [完成](#)

图 5. 3. 5 创建连接池-验证连接属性

- 验证连接属性

- 默认测试 SQL 语句：验证连接时使用的 SQL 语句。应用服务器将使用用户指定的 SQL 语句进行验证。默认提供相应数据库的测试 SQL 语句。

- 重新定义测试 SQL 表名：不指定则采用默认 SQL 语句，若指定则按照新的自定义表名来构建查询语句进行校验。

- 获取连接时验证：是否启用获取连接时对连接进行有效性检查。默认为 false。

- 创建连接时验证：是否在创建连接时对连接进行有效性检查。默认为 false。

- 归还连接时验证：是否在归还到池中前对连接进行有效性检查。默认为 false。

- 验证超时：测试连接活动的最长时间，以秒为单位，不能小于 1s。默认为 5。

- 连接验证时间间隔：避免过度验证，保证验证不超过这个频率，以毫秒为单位。如果一个连接应该被验证，但上次验证未达到指定间隔，将不再次验证，默认 30000。

验证连接失败后，应用服务器将关闭当前连接，然后重新建立连接。

5. 设置好以上信息之后，点击“下一步”按钮，进入“创建连接池”高级属性页面，如果不进入下一步而是在本页面点击“直接完成”，那么高级属性的配置全部采用默认值。“创建连接池”高级属性页面如图 5. 3. 6 所示

高级属性	
auto-commit <input type="checkbox"/> 开启	连接池创建的连接的默认的auto-commit 状态
释放连接时提交 <input checked="" type="checkbox"/> 开启	连接返回时提交 不选则默认回滚
打印初始化连接异常 <input checked="" type="checkbox"/> 开启	是否打印初始化连接异常
线程连接关联 <input type="checkbox"/> 开启	JDBC连接与线程绑定
是否开启事务连接 <input checked="" type="checkbox"/> 开启	使用JTA事务管理的开关
空闲回收 <input checked="" type="checkbox"/> 开启	标记是否删除空闲超时的连接
空闲超时时间 <input type="text" value="60000"/>	连接在池中保持空闲而不被连接回收器线程回收的最小时间值，
空闲检查周期 <input type="text" value="60000"/>	空闲连接回收器线程运行期间休眠的时间值，以毫秒为单位，默
泄露回收 <input type="checkbox"/> 开启	如果设置为true，则当完成泄露连接跟踪后，将泄露的连接销毁。
泄露超时时间 <input type="text" value="2"/>	在这段时间内跟踪连接池中的连接泄露，并将获取连接的调用栈
连接泄露时打印日志 <input type="checkbox"/> 开启	在检测到泄露连接的时候打印程序的stack traces 日志
sql日志 <input type="checkbox"/> 开启	是否启用SQL日志功能，选中为启用
语句跟踪 <input type="checkbox"/> 开启	语句跟踪
语句超时 <input type="text" value="0"/>	在该时间后将终止运行时间异常长的查询（以秒为单位）。应用
语句缓存 <input type="checkbox"/> 开启	语句缓存
客户端信息 <input type="text"/>	JDBC连接的ClientInfo属性信息，格式为分号分隔的键值对，如：
连接最大寿命 <input type="text" value="1800000"/>	保持连接的最大毫秒数，不能低于30000ms
即时泄露回收 <input checked="" type="checkbox"/> 开启	该功能与数据源自身的泄露回收功能相比，在每次请求结束后都

图 5.3.6 创建连接池-高级属性

- 高级属性
 - 释放连接提交：释放连接的时候是否自动提交，默认：自动提交
 - 线程连接关联：用户线程与数据库连接绑定（若希望数据库连接不上释放连接则需要开启获取连接时验证），用户线程被 GC 才释放连接。默认：不开启。
 - 是否开启事务连接：如果设置为 true，表示支持资源管理器本地事务或 JTA 事务，如果设为 false，表示不支持 JTA 事务。默认为 true。
 - XA 事务由事务管理器在资源管理器外部进行控制和调整。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。
 - auto-commit：连接池所创建的连接的默认的 auto-commit 状态。默认为 true。
 - 空闲回收：标记是否删除空闲超时的连接。
 - 空闲超时时间：连接在池中保持空闲而不被连接回收器线程回收的最小时间值，单位毫秒，默认为 60000ms，不能超过连接最大寿命。
 - 空闲检查周期：空闲连接回收器线程运行期间休眠的时间值，以毫秒为单位，默认为 60000ms。
 - 泄露回收：泄露回收功能。设为 true 时，如果连接发生泄露超时（连接占用的时间超过泄露超时时间），将连接丢弃。默认为 false。
 - 泄露超时时间：在这段时间内跟踪连接池中的连接泄露，并将获取连接的调用栈（堆栈）记录下来，当超时后，以 warning 级别将该堆栈信息输出到日志。以秒为单位。
 - 连接泄露时打印日志：在检测到泄露连接的时候打印程序的 stack traces 日志。
 - sql 日志：是否启用 SQL 日志功能，选中为启用
 - 连接最大寿命：保持连接的最大毫秒数。当一个连接被归还时，如果从连接被创建到当前时间的的时间差大于该值，连接将被关闭而不是回到池中，以毫秒为单位。
 - 语句跟踪：是否开启语句跟踪，开启语句跟踪功能，当连接关闭的时候连接池会检查

应用中是否有遗忘关闭的 `statement`，如果存在的话，由连接池帮助关闭

o 语句超时：在该时间后将终止运行时间异常长的查询（以秒为单位）。应用服务器将在所创建的语句上设置“`QueryTimeout`”。默认值为 0，表示未启用该属性。需要注意的是，某些数据库驱动，例如 MySQL，启用该特性时，会创建大量的 `timer` 线程，占用过多的资源，需谨慎使用。

o 语句缓存：是否开启语句缓存

o 即时泄漏回收：数据源连接回收功能，与数据源自身的泄露回收功能相比，该功能在每次请求结束后都进行回收，回收效率更高。一般情况下，没有主动关闭连接的应用很大概率也没有主动关闭 `Statement`，此时需要配合数据源的语句跟踪功能一起使用。

6. 设置好上述相应属性后，点击“完成”按钮，创建成功之后返回连接池列表，在页面顶端提示创建成功信息。如果创建失败，提示创建失败信息，点击详情可以查看失败原因。如图 5.3.7 所示：



图 5.3.7 连接池创建成功页面

5.3.2 查看/编辑连接池

1. 展开管理控制台左侧导航树中的“JDBC 配置”，进入连接池管理页面；
2. 点击需要查看/编辑的连接池名，页面显示如下图 5.3.8 所示：

名称	a	唯一标识符，连接池名称
资源类型	<input checked="" type="radio"/> DataSource <input type="radio"/> XADataSource	资源类型
* 数据库驱动类名	<input type="text" value="com.mysql.jdbc.Driver"/>	驱动类名
* 连接url	<input type="text" value="jdbc:mysql://10.10.40.103:10000/test?"/>	连接url
用户名	<input type="text" value="root"/>	用户名
密码	<input type="password" value="*****"/>	密码
连接参数	<input type="text"/>	当用Driver建立新连接时被发送转
驱动路径	<input type="button" value="取消"/>	驱动路径

/	<ul style="list-style-type: none">devetchome<ul style="list-style-type: none">alex<ul style="list-style-type: none">apache-maven-3.3.9mavenRepositorytw7twDoc下载<ul style="list-style-type: none">idea-IU-173.4548.28TongWeb7_2018-02-27
	已选择文件: /home/alex/下载/mysql-connector-java-5.1.27-bin.jar

池设置

初始化连接数	<input type="text" value="5"/>	连接池启动时
最大连接数	<input type="text" value="80"/>	连接池在同一
最小空闲连接数	<input type="text" value="10"/>	最小空闲连接
等待超时时间	<input type="text" value="40000"/>	获取连接的最

验证连接属性		
默认测试SQL语句	SELECT 1 from DUAL	用来验证从连接池取出的连接是否能够正常的
重新定义测试SQL表名	<input type="text"/>	不指定则采用默认SQL语句，若指定则按照新
创建连接时验证	<input type="checkbox"/> 开启	指明是否在创建连接时时进行检验
获取连接时验证	<input type="checkbox"/> 开启	指明是否在从池中取出连接前进行检验
归还连接时验证	<input type="checkbox"/> 开启	指明是否在归还到池中前进行检验
验证超时	<input type="text" value="5"/>	测试连接活动的最长时间，以秒为单位，不能
连接验证时间间隔	<input type="text" value="30000"/>	避免过度验证，保证验证不超过这个频率，以秒为 再次验证，默认30000
高级属性		
auto-commit	<input type="checkbox"/> 开启	连接池创建的连接的默认的auto-commit 状态
释放连接时提交	<input checked="" type="checkbox"/> 开启	连接返回时提交 不选则默认回滚
打印初始化连接异常	<input checked="" type="checkbox"/> 开启	是否打印初始化连接异常
线程连接关联	<input type="checkbox"/> 开启	JDBC连接与线程绑定
是否开启事务连接	<input checked="" type="checkbox"/> 开启	使用JTA事务管理的开关
空闲回收	<input checked="" type="checkbox"/> 开启	标记是否删除空闲超时的连接
空闲超时时间	<input type="text" value="60000"/>	连接在池中保持空闲而不被连接回收器线程回收的最小时间值，
空闲检查周期	<input type="text" value="60000"/>	空闲连接回收器线程运行期间休眠的时间值，以毫秒为单位，影
泄露回收	<input type="checkbox"/> 开启	如果设置为true，则当完成泄露连接跟踪后，将泄露的连接销毁。
泄露超时时间	<input type="text" value="2"/>	在这段时间内跟踪连接池中的连接泄露，并将获取连接的调用栈
连接泄露时打印日志	<input type="checkbox"/> 开启	在检测到泄露连接的时候打印程序的stack traces 日志
sql日志	<input type="checkbox"/> 开启	是否启用SQL日志功能，选中为启用
语句跟踪	<input type="checkbox"/> 开启	语句跟踪
语句超时	<input type="text" value="0"/>	在该时间后将终止运行时间异常长的查询（以秒为单位）。应用
语句缓存	<input type="checkbox"/> 开启	语句缓存
客户端信息	<input type="text"/>	JDBC连接的ClientInfo属性信息，格式为分号分隔的键值对，如：
连接最大寿命	<input type="text" value="1800000"/>	保持连接的最大毫秒数，不能低于30000ms
即时泄露回收	<input checked="" type="checkbox"/> 开启	该功能与数据源自身的泄露回收功能相比，在每次请求结束后都

图 5.3.8 查看/编辑 JDBC 连接池

- 基本属性
同“创建 JDBC 连接池”中属性
 - 池设置
同“创建 JDBC 连接池”中属性
 - 验证连接属性
同“创建 JDBC 连接池”中属性
 - 高级属性
同“创建 JDBC 连接池”中属性
3. 编辑 JDBC 连接池属性后，点击“保存”按钮。

5.3.3 测试连接

1. 展开管理控制台左侧导航树中的“JDBC 配置”节点，进入连接池管理页面；

2. 在连接池列表中点击需要测试连接的测试连接按钮，在页面上方显示测试连接结果。

5.3.4 删除 JDBC 连接池

1. 展开管理控制台左侧导航树中的“JDBC 配置”节点，进入连接池管理页面；
2. 选中要删除的 JDBC 连接池；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

5.4 多数据源 JDBC 的使用

5.4.1 创建多数据源连接池

1. 依次展开管理控制台左侧导航树中的“JDBC 配置”节点，然后点击“JDBC 配置”节点；如图 5.4.1 所示：



图 5.4.1 连接池列表

2. 在出现的 JDBC 连接池列表页中点击“创建多数据源连接池”按钮，出现如图 5.4.2 所示的创建 JDBC 多数据源连接池-基本属性页面；



图 5.4.2 创建 JDBC 多数据源连接池-基本属性

- 属性设置
- 名称：连接池（数据源 JNDI）的名称，连接池的唯一标识。
- 算法类型：故障转移、负载均衡二选一
- 资源类型：可选值为 `javax.sql.DataSource`、`javax.sql.XADataSource`。
- 数据源列表：提供服务的单数据源列表。
- 故障转移：当数据源繁忙时，是否进行故障转移，繁忙定义可以连接已被全部占用。

- 测试频率：多数据源定时状态检查频率。
3. 设置好上述相应属性后，点击“完成”按钮，创建成功之后返回连接池列表，在页面顶端提示创建成功信息。如果创建失败，提示创建失败信息，点击详情可以查看失败原因。如图 5.4.3 所示：



图 5.4.3 连接池创建成功页面

5.4.2 查看/编辑多数据源连接池

1. 展开管理控制台左侧导航树中的“JDBC 配置”，进入连接池管理页面；
2. 点击需要查看/编辑的连接池名，页面显示如下图 5.4.4 所示：

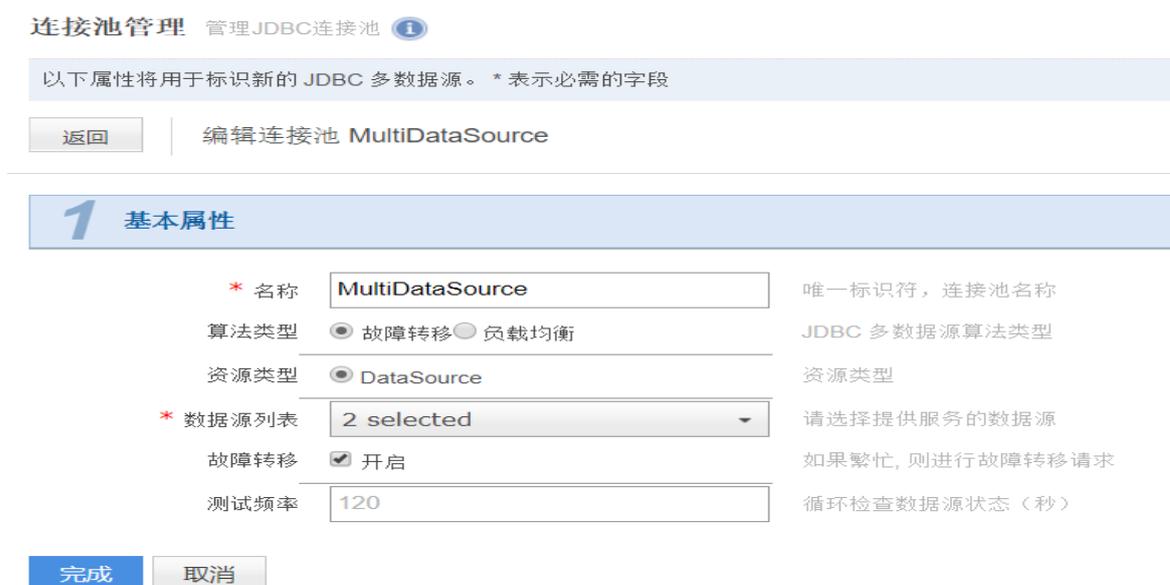


图 5.4.4 查看/编辑 JDBC 多数据源连接池

- 基本属性
同“创建 JDBC 多数据源连接池”中属性
4. 编辑 JDBC 连接池属性后，点击“完成”按钮。

5.4.3 测试连接

1. 展开管理控制台左侧导航树中的“JDBC 配置”节点，进入连接池管理页面；
2. 在连接池列表中点击需要测试连接的测试连接按钮，在页面上方显示测试连接结果。

5.4.4 删除 JDBC 连接池

1. 展开管理控制台左侧导航树中的“JDBC 配置”节点，进入连接池管理页面；
2. 选中要删除的 JDBC 连接池；

3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

6 EJB

EJB3.2 是 Java EE7 中关于 EJB 的新规范，其向下兼容 EJB3.1, EJB3.0 以及 EJB2.x。

6.1 EJB2.X 特性

2.0 增加的特性包括 Local 接口、CMP EntityBean,它们是对 1.x 缺陷的重大更正。Enterprise JavaBeans 体系结构是用于开发和部署基于组件的分布式业务应用程序的组件体系结构。使用 Enterprise JavaBeans 架构编写的应用程序具有可伸缩性，事务性和多用户安全性。这些应用程序可以编写一次，然后部署在支持 Enterprise JavaBeans 规范的任何服务器平台上。

Enterprise JavaBeans 2.0 规范的目的是解决 Enterprise JavaBeans 1.1 规范中的许多开放区域。

1. 将 Enterprise JavaBean 与 Java Message Service TM (JMS) 集成。JMS 是用于从 Java 程序访问企业消息传递系统的 API。需要进行 JMS 集成，以允许从客户端异步调用 Enterprise JavaBean，允许它们与使用 JMS 进行集成的旧系统进行互操作，支持将断开连接的客户端与 Enterprise JavaBean 一起使用以及允许在发布/订阅中配置和使用 Enterprise JavaBean。
2. 改进了对实体 Bean 持久性的支持。EJB 规范当前未定义数据访问对象或策略的体系结构。当前支持两种持久性形式：bean 管理的和容器管理的。Bean 管理的持久性在定制使用的数据访问组件方面为 Bean 开发人员提供了高度的灵活性。容器管理的持久性使开发人员不必编写数据访问调用的代码，并允许 Bean 类独立于存储实体的数据源。对于具有容器管理的持久性的实体 bean，容器提供程序依赖于工具来在 bean 部署时生成持久性存储访问层。对于 Bean 管理的持久性，EJB 规范假定数据访问调用或者直接编码到企业 bean 类中，或者封装在作为实体 bean 一部分的数据访问组件中。但是，如果将数据访问调用直接编码到 Bean 类中，则可能很难使该组件适应与具有不同架构的数据库一起使用。在使用封装的数据访问组件的情况下，我们希望这些数据访问组件由工具生成，因此并非对所有工具都一样。为了更好地支持使用工具来为 Bean 管理的持久性或容器管理的持久性生成数据访问组件，我们需要研究在何种程度上可以为 bean 提供程序定义描述复杂内部结构的标准方式为了确保在一个 Enterprise JavaBeans 服务器上开发的 bean（具有一组特定的工具）可以被可移植地部署在具有不同持久存储工具和一组不同工具的不同服务器环境中，和另一个数据库。另外，在容器管理的持久性的情况下，容器与持久性存储机制之间的标准接口将使工具能够在多个供应商的容器之间运行。
3. 支持企业 JavaBean 之间的关系。在 Enterprise JavaBeans 1.1 规范中，企业 Bean 之间的 1-1、1-n 和 mn 关联的管理以及企业 Bean 及其依赖对象之间的关系的管理留给 Bean 提供程序。Enterprise JavaBeans 2.0 中定义一种方法来捕获关于 bean 开发人员已知的这种关系的信息，以便可以在部署时以及在运行时使此信息可用。
4. 支持 Enterprise JavaBeans 的继承和子类化。Enterprise JavaBeans 规范当前不提供组件的子类。EJB 2.0 中为此添加支持。
5. 实体 Bean Finder 方法的查询语法。Enterprise JavaBeans 1.1 规范假定具有容器管理的持久性的实体 bean 的查找器方法是使用容器提供程序的工具生成的。但是，仅此类方法的名称和签名没有指定足够的信息来生成除 `ejbFindByPrimaryKey` 之外的 finder 方法的实现。因此，Bean 提供者必须以特定于容器及其关联工具的方式传达此类方法的描述，从而妨碍 Bean 的可移植性。为了支持可移植 finder 方法的定义，EJB2.0 定义一种格式，用于指定 finder 方法实现将使用的查询条件或选择谓词。
6. 支持主页界面中的其他方法。Bean 提供程序当前无法添加方法来提供独立于单个 bean 实例的存在的行为（Home 接口上的 `create` 和 `finder` 方法除外）。我们计划研究增加对 bean 提供程序的支持，以在 Home 接口上定义其他方法，例如，批量更新操作。
7. 容器扩展机制，拦截器是通常由客户现场的系统程序员提供的方法，容器在 bean 调

用协议期间，通过标准接口在定义明确的点（并在定义明确的安全性和事务上下文内）调用此方法。EJB2.0 计划引入一种拦截器机制，以提供一种可移植的方式来针对特定操作环境专门化容器的行为，从而减少对专用容器的需求。

8. EJB 服务器网络互操作性协议。为了支持来自多个供应商的基于 CORBA 的 EJB 服务器实现之间的网络互操作性，有必要通过指定对可互操作性的安全性和命名的支持来完成 RMI/IIOP 完成 EJB 的映射。

6.2 EJB3.0 特性

EJB 规范的主要目的是定义一个面向对象的分布式组件架构，这种架构的主要思想是向 EJB 开发人员和使用人员隐藏底层的操作细节，使他们专注于业务的开发和调用。EJB3.0 以前的规范更多地从 EJB 容器的角度来设计 EJB 架构，从 EJB3.0 开始，EJB 规范把重点放在从 EJB 开发人员的角度来简化 EJB 的架构。EJB3.0 的主要特点如下：

1. 简化了接口定义要求

在 EJB3.0 中，开发一个 EJB 组件不需要再定义 Home 接口、远程或本地接口。EJB 开发人员只需要把 EJB 的业务方法定义在普通的 Java 接口中即可。该 Java 接口被称作 EJB 组件的业务接口 (Business Interface)。

2. 简化 EJB 对环境的查找 (依赖注入)

在 Java EE 环境中，EJB 组件通常要查找某些资源来完成某功能。例如引用另外一个 EJB，引用数据源等。在 EJB3.0 之前，一个 EJB 组件获取环境中的资源需要进行 JNDI 查找。从 EJB3.0 开始，通过依赖注入的方式，EJB 组件查找 Java EE 中的资源更加简单，只需要通过声明的方式就可以完成。

3. 引入 Java 持久化机制 (JPA)

EJB3.0 规范中引入了 JPA，这套机制定义了如何进行 Java 对象与关系数据库之间的映射关系。通过该机制可以简化 EJB2.x 的 Entity Bean 的开发。同时该机制还可以运行在 Java SE 的环境中，因此在 JDBC 的基础上，增加了另一个 Java 持久化的选择。（在 EJB3.0 规范中使用 JPA 实现持久化的应用称作 Entity，而 CMP 和 BMP 统称为 Entity Bean。）

6.3 EJB3.1 特性

EJB3.0 给 JavaEE5 带来了由重量级向轻量级的转变，而 EJB3.1 的目标更是在此基础上再接再厉，增加许多必须的特性。

1. 无接口的 EJB

EJB 3.0 移除了复杂的本地和远程接口，EJB 3.1 更进了一步，接口也不再是必须的。EJB 3.1 中，Session Beans 的定义可以不需要任何的接口，只要给一个 POJO 标记上 @Stateless 或者 @Stateful 就能得到所有的 EJB 功能。示例如下：

```
@Stateless public class PlaceBidBean {
    @PersistenceContext
    private EntityManager entityManager;
    public void placeBid (Bid bid) {
        entityManager.persist(bid);
    }
}
```

2. 引入了 Singleton Beans

如果需要缓存一些数据在内存中而不是重复的从数据库中反复的查询，Stateless Session Beans 和 Stateful Session Beans 都不能达到这种需求。因此在 EJB3.1 中引入了 Singleton Beans 的概念。当一个 Bean 被标记为 Singleton 时，在整个应用层容器只能保证每个 JVM 共享一个实例，对于缓存这一规定是行之有效的。在并发访问的控制上，使用了 Readandwrite 机制，通过标记 @Lock (READ) @Lock (WRITE) 来控制并发访问。

3. 异步调用

在 EJB 3.1 之前，在会话 Bean 上的任何函数调用都是同步的。而在 EJB3.1 中，会

话 Bean 的方法使用@Asynchronous 注解后便成为是异步调用的方法。异步调用的方法可以返回一个 java.util.concurrent API 的 Future 对象，客户端可以通过这个 Future 对象获取调用的状态。

4. 统一的 jndi 命名

之前的 EJB 规范中，对于 EJB 的 jndi 命名并没有要求。而在 EJB3.1 中规范了 jndi 的命名。根据调用 EJB 的访问级别：global、app、module，EJB 对应的统一 jndi 名分别为：

```
java:global[/<app-name>]/<module-name>/<bean-name>
java:app/<module-name>/<bean-name>[!<fully-qualified-interface-name>]
java:module/<bean-name>[!<fully-qualified-interface-name>]
```

5. 直接用 WAR 文件打包 EJB 组件

EJB 3.1 中一个重要的改进是可以直接将 EJB 组件打包到 WAR 文件中，不用再独立创建 jar。如下图所示：

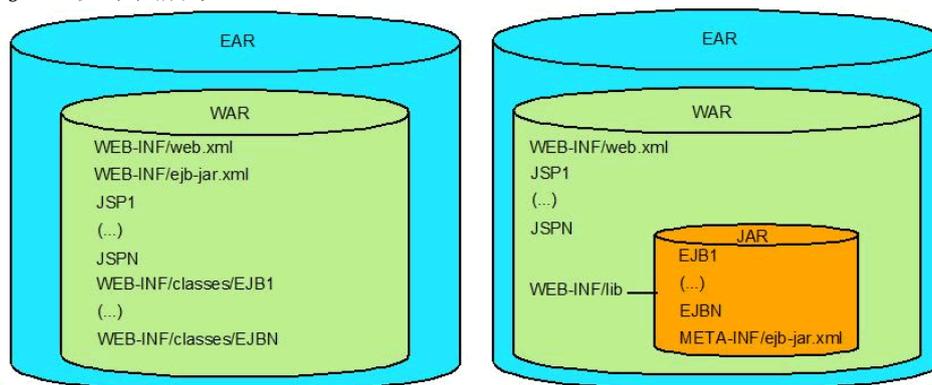


图 6.2.1 EAR

6. EJB Lite

许多企业级应用不需要 EJB 完整的功能，因此在 EJB 3.1 中引入了 EJB Lite，它是 EJB API 的一个子集，EJB Lite 包括了创建一个企业级应用的所有功能，但不包括专业的 API。EJB Lite 提供了厂家选项，让厂家可以在它们自己的产品中实施 EJB API 的子集，使用 EJB Lite 创建应用程序可以部署到任何支持 EJB 的服务器上，不管它是完整的 EJB 还是 EJB Lite。

6.4 EJB3.2 新特性

EJB3.2 规范相对于 EJB3.1 及以前的规范没有革命性的变化，仅仅是对以前具有的功能的扩展和补充，以及放宽了某些实现限制，朝着使 EJB 轻量化和易用的目标更进了一步，例如：

1. 扩展了 EJB3.2Lite 功能集，增加了支持本地异步会话 Bean 调用和非持久性 EJB Timer Service。
2. 会话 bean 指定实现接口（作为本地或远程业务接口）的默认规则已经放宽，可以包含多个接口。
3. 可以完全禁用特定的有状态会话 bean 的钝化（passivation）。
4. 有状态会话 bean 的生命周期回调拦截方法，现在可以在一个事务环境中执行（由生命周期回调方法的事务属性决定）。
5. 扩展了 TimerService API，可以在同一个 EJB 模块中查询所有活动计时器。
6. JMS 消息驱动 bean 的标准激活属性名单已经扩展，以与 JMS 2.0 规范中的变化相匹配。

6.5 EJB2.x 应用结构

EJB2.x 结构下图如所示：

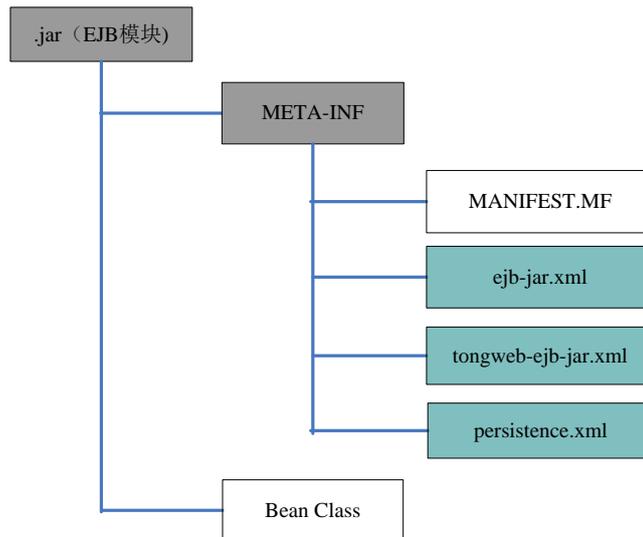


图 EJB2.x 应用结构

- **ejb-jar.xml**: Java EE 标准部署描述文件。
- **tongweb-ejb-jar.xml**: 描述 TongWeb 中特定 EJB 属性的部署描述文件。
- **persistence.xml**: Entity 使用 JPA 的 JDBC 映射信息。

6.6 EJB2. x 使用说明

EJB2. x 实体 Bean，有两种主要类型 BMP 和 CMP。

BMP 是在 Bean 中完成对数据库 JDBC 的各种调用，在实体 bean(entity bean)中，明确写入了 SQL 语句，如“insert .. ”或“select .. ”,并且使用 Datasource 获得一个数据库资源以及连接(connection)从而对数据库直接进行增加删除修改。

CMP 是由 EJB 容器自动完成对数据库的操作。在实体 bean 重写入 setXXX 或 getXXX 方法，然后在 ejb-jar.xml 中定义 cmp-field，根据方法所需，配置 query 元素，将所需 Sql 在配置文件中实现。

6.6.1 配置说明

- 1、[ejb-jar.xml](#) 配置见附件说明。
- 2、[persistence.xml](#) 配置见附件说明：

6.6.2 CMP 使用步骤

- 1、编写类继承 javax.ejb.EJBObject，这个类，作为远程接口。
- 2、编写接口继承 javax.ejb.EJBHome。
- 3、编写抽象类继承 javax.ejb.EntityBean，并实现 ejbCreate 方法。
- 4、配置 ejb-jar.xml 文件，具体含义，请查看[配置说明](#)。
- 5、在 persistence.xml 中配置数据源。

6.6.3 BMP 使用步骤

- 1、编写类继承 javax.ejb.EJBObject，这个类，作为远程接口。
- 2、编写接口继承 javax.ejb.EJBHome, 这个类里有一个方法是 create () 它返回接口类型。并定义业务方法。
- 3、编写类，继承 javax.ejb.EntityBean，实现 ejbCreate 和相关业务方法。
- 4、配置 ejb-jar.xml 文件，具体含义，请查看[配置说明](#)。
- 5、在 persistence.xml 中配置数据源。

6.7 JPA

早在 Java EE 5 平台就引入了 Java 持久化 API (Java Persistence API, JPA)，它为 Java EE 和 Java SE 应用程序提供了一个基于 POJO 的持久化模块。JPA 处理关系数据与 Java 对象之间的映射，它使对象/关系 (O/R) 映射标准化，JPA 已经被广泛采用，已经成为事实上的 O/R 持久化企业标准。

其总体结构如图 6.4.1 所示:

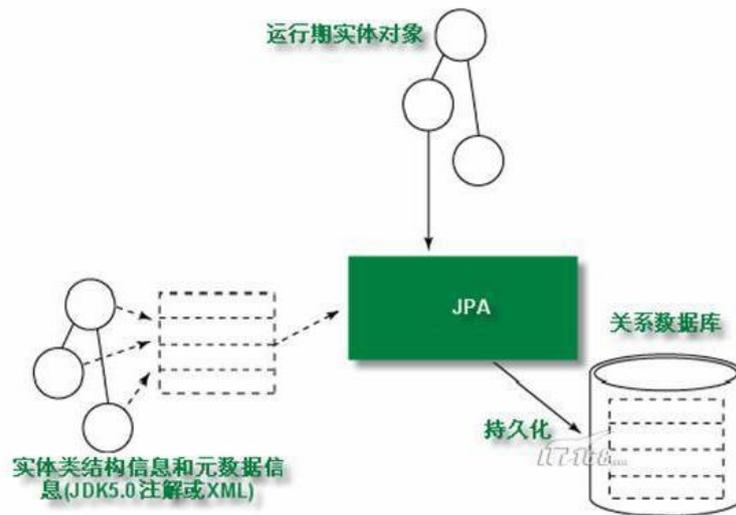


图 6.4.1 JPA 总体结构图

元数据信息用来描述实体和数据库表的映射关系，JPA 框架根据元数据信息将实体对象持久化到数据库中。描述映射关系的元数据信息有 2 种方式，Annotation 和基于注入的 xml 配置文件(META-INF/orm.xml)，xml 配置文件的优先级高于 Annotation。

Java EE 6 带来了 JPA 2.0，它有了许多新特性和增强，包括：

- 1、对象/关系映射增强；
- 2、Java 持久化查询语言增强；
- 3、一种新的基于标准的查询 API；
- 4、支持悲观锁定。

Java EE7 带来了 JPA 2.1，它也包含了许多新特性和改进，包括：

- 1、以标准化的方式支持存储过程。
- 2、支持属性转换器，实现基本值在数据库表示和相应对象表示之间进行转换。
- 3、支持实体图，可以用于定义一个实体及其子元素的加载方式。
- 4、支持通过 Criteria 批量的更新和删除数据。
- 5、支持通过 CDI 实现实体类生命周期事件监听。
- 6、支持通过 SynchronizationType 控制当前事务持久化上下文的同步。
- 7、支持借助于@ConstructorResult 注解能够使用从一个 SQL 查询返回的参数值构造对象。

目前 TongWeb7 的 JPA 默认实现框架是 Eclipse 的 Eclipse 2.6.3。

6.7.1 EclipseLink

EclipseLink 实现了 Java EE 中的 Jsr388: java persistence2.1 规范，为开发者提供功能强大、使用简单的持久化数据管理框架。EclipseLink 封装了和关系型数据库交互的操作，让开发者把注意力集中在编写业务逻辑上。EclipseLink 可以作为独立的持久层框架发挥作用，也可以轻松的与其它 Java EE 应用框架或者符合 EJB 3.2 标准的容器集成。

EclipseLink 支持的数据库

- Apache Derby
- Borland JDataStore
- IBM's DB2
- Informix
- ingres
- empress
- HSQL
- Microsoft Access

- Microsoft Visual FoxPro
- Microsoft's SQL Server
- MySQL
- Oracle
- Pointbase
- Postgres
- Sybase

6.7.2 使用说明

TongWeb7 中 EclipseLink 的使用方式如下：

建立持久化配置文件

首先需要在应用的\META-INF\下建立一个 jpa 规范对应的配置文件 persistence.xml。

如：

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- suppress ALL -->
- <persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd" version="2.1">
- <persistence-unit name="CTS-EM" transaction-type="JTA">
  <provider>com.tongweb.eclipse.persistence.jpa.PersistenceProvider</provider>
  <jta-data-source>jdbc/DB1</jta-data-source>
  <class>com.sun.ts.tests.jpa.core.persistenceUtil.Employee</class>
- <properties>
  <property name="eclipselink.target-database" value="com.tongweb.eclipse.persistence.platform.database.HSQLPlatform" />
  <property name="eclipselink.ddl-generation" value="create-tables" />
  <property name="eclipselink.ddl-generation.output-mode" value="database" />
</properties>
</persistence-unit>
</persistence>
```

图 6.4.2 persistence.xml 配置

对于配置文件中所需要的数据源可以通过 TongWeb7 的管理控制台来创建。关于 EclipseLink 更多特有属性配置可参考 [EclipseLink 官网](#)。

EclipseLink 根据该持久化文件的配置生成实体 manager factory。之后创建 EntityManager 提供了相应的操作实体、事物、查询等方面的 API。具体应用代码中的调用方式举例如下图：

<http://openjpa.apache.org/builds/2.4.0/apache-openjpa/docs/main.html>

另外还提供一个 -D 开关，可以关闭 TongWeb7 自带的 JPA 实现：-DenableJPA=false，不配置该参数则使用默认值 true，即表示默认开启 TongWeb7 自带的 JPA。

6.7.3 JPA 事物支持说明

对于 JPA 提供的特色功能，一般通过 persistence.xml 的 properties 中进行配置。TongWeb7 对 hibernate 的 JPA 事务支持，通过以下配置实现：

```
<property name="hibernate.transaction.jta.platform"
value="com.tongweb.tongejb.hibernate.OpenEJBjtaPlatform" />
```

6.8 EJB 容器

EJB 容器是可供 EJB 运行的环境，它负责 EJB Bean 实例的生命周期管理，Bean 实例状态管理，并发(线程)管理，事务服务，安全服务，命名服务，资源(数据源等)管理，使得 EJB 客户端程序能进行远程调用或本地调用。EJB 容器支持 EJB3.2 规范，提供定时服务，依赖注入，拦截器等功能。

6.8.1 EJB3.2 特性支持

EJB 3.2 规范中定义了两类 EJB 规范：支持部分 EJB 特性的轻量级规范 EJB3.2 Lite 以及支持全部 EJB 特性的 EJB3.2 Full。TongWeb7 完全支持 EJB3.2 Lite 定义的全部特性以及 EJB3.2 Full 中定义的部分其他特性。

6.8.2 EJB 实例管理

6.8.2.1 无状态会话 Bean 的实例池

为提高无状态会话 Bean 的响应性能，EJB 容器提供了实例池 (Pool) 机制进行 Bean 实例管理。为了避免 EJB 容器为用户的每次方法调用都进行实例的创建与销毁，因而影响系统性能，减少系统吞吐量，EJB 容器提供了 EJB 池 (实例池) 来应对大量用户的访问。

在应用服务器刚刚启动成功时，根据容器属性“最小实例数”来创建实例，如果“最小实例数”设置为 0 那么池中是没有实例的。

当有客户端调用时，容器会检查实例池状态，已经分配出去调用的实例是否已经达到“最大实例数”。

1. 如果没达到，那么容器会去实例池中获取实例，如果存在空闲实例那么选择一个空闲实例直接返回，如果没有空闲实例，那么创建一个新实例给调用请求，等调用结束后把新创建的实例放到池中，成为空闲实例。
2. 如果达到，并且“池溢出策略”为等待池空闲，那么当前请求就阻塞住等待空闲实例，直到之前分配的实例调用结束返回池中成为空闲实例，容器将空闲实例返回给此次调用的请求。
3. 如果达到，并且“池溢出策略”为创建临时实例，那么创建一个新实例给调用请求，调用结束后，这个新创建的实例不放到实例池中。

容器运行时，会周期性的扫描实例池，从而管理实例池中的所有实例。每次进行实例池的扫描时，会检查每个空闲实例：

1. 实例的存活时间是否超过“实例超时时间”。如果超过，则认定这个实例为超时实例。如果设置了“实例超时替换”，那么创建一个新的实例来替换这个超时实例。如果没有设置“实例超时替换”，并且当前池中实例数大于“最小实例数”，那么将删除这个超时实例。
2. 实例的空闲时间是否超过“实例空闲超时时间”。如果超过，则认定这个实例为空闲超时实例。如果当前池中实例数大于“最小实例数”，那么将删除这个空闲超时实例。

例。

- 实例池是否设置了允许刷新，如果设置了，那么判断容器是否进行过刷新操作，如果调用过，那么将实例中的每一个实例都创建一个新的实例来替换。

6.8.2.2 有状态会话 Bean 的实例缓存

对于 EJB 池，因为 EJB 容器不为每个客户端分别维护相应的 Bean 实例，同时还重复使用这些 Bean 实例，所以实例池不适用于有状态的会话 Bean。对于有状态的会话 Bean，因为 EJB 容器需要为每个客户端分别维护相应的 Bean 实例，所以 EJB 池不适用，但是 EJB 容器为其提供了 EJB 缓存、钝化和激活机制来管理大量 Bean 实例。

对有状态的会话 Bean 来说，由于每个用户都有自己的 Bean 实例，那么有多少个用户就有多少个 Bean 实例为之服务，如果不采用任何机制，势必会导致服务器资源的严重浪费。

EJB 容器提供了 EJB 缓存来维护使用过的 EJB，这使得对 EJB 的请求能够被更快的响应，而当缓存中的 Bean 实例数量达到“缓存最大容量”后，会按照“钝化实例数”将一些实例对象保存到硬盘，并从实例缓存中删除此实例，释放缓存，这一过程称为钝化。EJB 调用时，从硬盘中找到钝化的实例并读取成实例对象的过程称之为激活。而当缓存中的 Bean 实例的空闲时间超过“空闲超时时间”仍然没有被再次访问的话，那么将从实例缓存中删除实例。

6.8.2.3 消息驱动 Bean 的实例池

为提高消息驱动 Bean 的响应性能，EJB 容器和无状态会话 Bean 一样，提供了实例池 (Pool) 机制进行 Bean 实例管理。实例池的使用和配置完全和无状态会话 Bean 相同，详细说明请参考 [6.5.2.1 章节无状态会话 Bean 实例池](#) 使用说明。

6.8.3 查看/编辑 EJB 配置属性

6.8.3.1 查看/编辑无状态会话 Bean 实例池属性

依次展开管理控制台左侧导航树中的“EJB-> 无状态会话 bean 配置管理”节点，出现如图 6.5.1 所示的无状态会话 bean 实例池配置管理页面，页面中提供池管理的属性：

此页显示了无状态会话Bean的相关属性，可以执行修改保存操作

等待超时	<input type="text" value="30"/>	秒	等待超时
最大实例数	<input type="text" value="10"/>		指定无状态session bean
最小实例数	<input type="text" value="0"/>		池初始化后容器中的实例
池溢出策略	<input checked="" type="radio"/> 等待池空闲 <input type="radio"/> 创建临时实例		选择等待池空闲，请求将
实例超时时间	<input type="text" value="0"/>	小时	设定实例的存活时间
实例超时替换	<input checked="" type="checkbox"/> 允许		实例超时后，是否可以替
刷新	<input type="checkbox"/> 允许		是否允许刷新
创建实例延迟参数	<input type="text" value="0"/>	%	延迟创建实例
实例空闲超时时间	<input type="text" value="0"/>	分钟	设置实例的最大空闲时间
实例垃圾回收	<input type="checkbox"/> 开启		实例是否回收
扫描频率	<input type="text" value="5"/>	分钟	设置扫描周期
执行替换线程数	<input type="text" value="5"/>		这个属性设置了执行实例
实例关闭超时时间	<input type="text" value="5"/>	分钟	关闭池操作的超时时间

图 6.5.1 查看/编辑实例池属性
具体说明如下：

- 等待超时：从池中获取实例等待的超时时间。默认为 30 秒。
- 最小实例数：池中 Bean 实例的初始值和最小值，默认值为 0。
- 最大实例数：池中 Bean 实例的最大值，默认值为 10。
- 池溢出策略：当池中正在使用中的实例数量达到最大实例数，又有请求申请池分配实例时的实例分配策略。默认为“等待池空闲”。
- 实例超时时间：实例在池中允许存活的最大的时间。默认是 0 小时，也就是没有超时时间。
- 实例超时替换：当池中的实例的存活时间超过“实例超时时间”后的策略，是替换还是删除。默认是“允许”，也就是替换。
- 刷新：当调用池的刷新操作时，是否更新池中的实例。默认是“不允许”。
- 创建实例延迟参数：当多个实例同时创建时，避免实例同时退休，每个实例按一定的时间比例延迟退休。默认是 0%。
- 实例空闲超时时间：实例的空闲时间超过该时间，则从实例池中删除该实例。默认值是 0 分钟，也就是没有空闲超时时间。
- 实例垃圾回收：开启实例垃圾回收，那么池中的实例在 jvm 中以软引用方式保存。当 jvm 内存紧张时就会回收池中的实例。默认为“不开启”。
- 扫描频率：实例池周期扫描池中实例，清理或者替换超时、空闲超时、刷新实例。这个属性配置了实例池多久进行一次扫描。默认是 5 分钟。
- 执行替换线程数：替换池中的实例时，会用线程池来进行替换操作。这个属性配置了线程池的线程数。默认是 5。
- 实例关闭超时时间：关闭池操作的超时时间。默认是 5 分钟。
编辑属性完成后，点击“保存”按钮；实例池属性即刻生效。

说明：该步骤编辑的实例池属性针对无状态会话 Bean。

6.8.3.2 查看/编辑有状态会话 Bean 属性

依次展开管理控制台左侧导航树中的“EJB-> 有状态会话 bean 配置管理”节点，出现如图 6.5.2 所示的有状态会话 bean 实例缓存配置管理页面：

有状态会话Bean配置 会话Bean配置 ⓘ

此页显示了有状态会话Bean的相关属性，可以执行修改保存操作

等待超时	<input type="text" value="30"/>	秒	每次调用等待一个可用的bean实例的最大时间，
会话空闲超时时间	<input type="text" value="20"/>	分钟	指的是一个bean在两次调用之间能够等待的最大
扫描频率	<input type="text" value="60"/>	秒	检查的频率
缓存最大容量	<input type="text" value="1000"/>		指定了bean容器中缓存的最大容量，默认1000
钝化实例数	<input type="text" value="100"/>		定义每次进行钝化处理的实例数量，默认100

图 6.5.2 查看/编辑实例缓存属性

具体说明如下：

- 等待超时：每次调用等待一个可用的 bean 实例的最大时间，超过这个时间将报错。
单位是秒，默认为 30 秒。
 - 会话空闲超时：指的是一个 bean 在两次调用之间能够等待的最大时间，默认是 20 分钟。
 - 扫描频率：实例缓存周期扫描的时间频率。默认是 60 秒。
 - 缓存最大容量：指定了 bean 容器中缓存的最大容量，默认 1000。
 - 钝化实例数：定义每次进行钝化处理的实例数量，默认 100。
- 编辑属性完成后，点击“保存”按钮，使实例缓存属性即刻生效。

说明：该步骤编辑的实例缓存属性针对有状态的会话 Bean。

6.8.3.3 查看/编辑单例会话 bean 属性

依次展开管理控制台左侧导航树中的“EJB-> 单例会话 Bean 配置管理”节点，出现如图 6.5.3 所示的单例会话 bean 配置管理页面：



图 6.5.3 查看/编辑实例缓存属性

- 等待超时：每次调用等待一个可用的 bean 实例的最大时间，超过这个时间将报错。单位是秒，默认为 30 秒。

编辑属性完成后，点击“保存”按钮，使实例缓存属性即刻生效。

6.8.3.4 查看/编辑消息驱动 Bean 实例池属性

依次展开管理控制台左侧导航树中的“EJB-> 消息驱动 Bean 配置管理”节点，出现类似如图 6.5.2 所示的无状态会话 bean 实例池配置管理页面，页面中提供池管理的属性；这里提供消息驱动 Bean 实例池的管理，使用说明和无状态会话 Bean 实例池配置完全相同。参看 6.5.2.1 章节。

6.8.4 EJB 远程调用

6.8.4.1 远程调用协议及方式

EJB 远程调用使用的数据传输协议仅支持 HTTP 协议，因此其调用的方式如下：

```
Properties p = new Properties();
p.put("java.naming.factory.initial", "com.tongweb.tongejb.client.RemoteInitialContextFactory");
p.put("java.naming.provider.url", "http://127.0.0.1:5100/ejbserver/ejb");
//url 类型是 http://IP:5100，其中 IP 是部署 ejb 的 TongWeb 服务器 IP 地址。
Context c1 = new InitialContext(p);
ManagerBeanRemote mbr = (ManagerBeanRemote)c1.lookup("ManagerBeanRemote");
mbr.remoteCall()
```

当远程调用 EJB 的客户端在非 JavaEE 容器环境运行时环境（servlet、EJB 等）中时，例如在一个普通 java 类中的远程调用 TongWeb7 中部署的 EJB，需要在执行 java 程序的 CLASSPATH 中加入客户端依赖 jar 包，客户端依赖 jar 包路径为 TW_HOME/lib/client/client.jar。

6.8.4.2 远程调用配置

EJB 远程调用使用的是 HTTP 通道，通道名称固定为“ejb-server-listener”，如图 6.5.4 所示 EJB 远程调用配置页面。

此页显示了已创建的HTTP通道，可以对这些通道执行编辑、启动、停止、删除操作或者创建新的通道。

[返回](#) | 编辑HTTP通道 ejb-server-listener

http通道名称	ejb-server-listener	通道名称，唯一的标识
http通道类型	<input checked="" type="radio"/> http <input type="radio"/> https	通道类型，选择https后需设置SSL属性
监听地址	<input checked="" type="radio"/> 全部 <input type="radio"/> 指定IP	指定IP
	<input type="text"/>	
* 监听端口	<input type="text" value="5100"/>	监听端口号
* 默认虚拟主机	<input type="text" value="admin"/>	默认虚拟主机
重定向端口	<input type="text" value="443"/>	重定向端口
io模式	<input type="text" value="nio"/>	io模式
代理服务器URL	<input type="text"/>	代理服务器的url，格式为ip:port
X-Powered-By	<input type="checkbox"/> 开启	启用后，在响应header中有如，X-Powered-By : Servlet/3.0 JSP/2.2

高级属性

图 6.5.4 EJB 远程调用通道配置

6.8.5 EJB 集群

6.8.5.1 支持故障转移

TongWeb7 可以支持 EJB 集群的故障转移功能。当 EJB 集群的某个应用服务器节点意外宕机或者其他原因无法访问，会将 EJB 调用的请求转发到集群中的其它节点。

6.8.5.2 支持故障隔离和恢复

EJB 集群中的某个应用服务器宕机或者其他原因无法访问，会被暂时隔离，之后的一段时间内，访问 EJB 集群的请求将不会尝试访问被隔离的应用服务器。直到间隔时间后，才会允许再次访问 EJB 集群中被隔离的应用服务器，如果该应用服务器可以正常访问，那么解除对它的隔离，如果依然无法访问，那么继续隔离。

这个间隔时间可以在访问 EJB 集群的客户端 JVM 参数中通过 `-Dcluster.isolation.interval=300000` 这样的方式进行配置，属性单位是毫秒，如果不配置这个属性，默认是 300000，也就是 5 分钟。

6.8.5.3 支持负载均衡

TongWeb7 应用服务器的 EJB 集群支持负载均衡，并提供了三种访问集群的负载均衡策略。具体介绍如下：

- 亲和：集群第一次访问使用随机的方式从集群中选择一个节点，如果节点访问成功，那么下次访问还使用这个节点。
- 轮转：按照集群配置的节点顺序，依次访问集群中的节点。
- 随机：每次访问从集群的多个节点中随机找出一个节点进行访问。
默认情况下 TongWeb7 使用轮询策略。

6.8.5.4 使用说明

调用 EJB 集群和 EJB 远程调用类似，需要在 `jndi` 中设置属性，不同的是在远程地址属性中配置多个节点，并用逗号分隔。具体使用方式如下：

```
Properties p = new Properties();
p.put("java.naming.factory.initial", "com.tongweb.tongejb.client.RemoteInitialContextFactory");
p.put("java.naming.provider.url",
"http://127.0.0.1:5100/ejbserver/ejb?readTimeout=60000,http://127.0.0.1:5101/
```

```
ejbserver/ejb?readTimeout=60000");//url 类型是 http://IP:5100, 其中 IP 是部署  
ejb 的 TongWeb 服务器 IP 地址。  
p.put("remote.loadbalance", "random");//设置负载均衡策略, 可以设置 roundrobin  
(轮转)、random(随机)、sticky(亲和)三个属性, 默认是 roundrobin  
readTimeout 是读超时时间, 毫秒为单位。默认为 30s  
Context c1 = new InitialContext(p);  
ManagerBeanRemote mbr = (ManagerBeanRemote)c1.lookup("ManagerBeanRemote");  
mbr.remoteCall();
```

6.9 全局事务

6.9.1 全局事务概述

JTA(Java Transaction API)为 J2EE 平台提供了分布式事务服务, 通过 JTA 可以使不同的资源(如支持 XA 协议的数据库)加入到同一个 JTA 事务中。TongWeb7 的全局事务是基于 JTA、XA 协议以及 EJB 远程调用协议而扩展出的跨应用服务器的事务服务, 此全局事务的事务管理器仍然是一个 JTA 事务管理器, 而加入到 JTA 事务中资源不再局限于 XA 资源, 还包括了远端应用服务器上部署的 EJB 资源。全局事务可以使得在一个 JTA 事务中调用到的某远端应用服务器上的 EJB 也将加入到这个 JTA 事务中, 同样地, 如果该 EJB 还调用了其它远端应用服务器上的 EJB, 那么这些 EJB 也会加入到这个 JTA 事务中, 这样的 JTA 事务称为全局事务。

6.9.2 全局事务场景描述

全局事务的场景涉及到多个 TongWeb7, 假设有四个 TongWeb7, 名称分别为 TW1、TW2、TW3、TW4, 每个 TongWeb7 上都部署了一个 EJB, 名称分别为 EJB1、EJB2、EJB3、EJB4, 每个 EJB 都使用了一个数据源, 名称分别为 DS1、DS2、DS3、DS4, 此外, EJB1 依次调用了 EJB2、EJB3, EJB3 又调用了 EJB4, 以上场景即是一个典型的全局事务场景, 在这种场景下四个数据源 DS1、DS2、DS3 和 DS4 将加入到同一个事务中, 统一提交或回滚。

全局事务是在 JTA 事务的基础之上进行的功能增强, 使得 JTA 事务可跨越 TongWeb 节点进行传播, 可应用于更广泛的分布式场景下的事务实施, 其具体特性如下:

1. 跨 TongWeb 节点的事务传播

从一个 TongWeb 节点上的 EJB 事务方法通过 EJB 远程调用进入另一个 TongWeb 节点上的 EJB 事务方法, 那么在第一个 TongWeb 节点上未提交的临时状态数据则对第二个 TongWeb 节点上 EJB 事务方法可见, 同理该事务可以继续传播到更远的 TongWeb 节点上。

2. 全局事务的事务性保证

根据上述“跨 TongWeb 节点的事务传播”, 当其中任何一个 TongWeb 节点上的 EJB 事务方法发生异常时, 则所有参与的 TongWeb 节点上的临时状态数据全部回滚, 即保证数据的原子性, 同时保证数据的一致性、持久性、隔离性等事务特性。

3. 全局事务的容错性

全局事务的容错性是指当 TongWeb 节点在事务处理过程中发生宕机, 事务可以通过记录的事务日志得以恢复, 而不破坏数据一致性, 同时也支持 EJB 集群下部分节点宕机的事务容错性。

6.9.3 全局事务传播策略

TongWeb7 的全局事务传播策略完全遵循 EJB 规范中定义的六种事务传播策略, 即:

- MANDATORY
- REQUIRED
- REQUIRES_NEW
- SUPPORTS
- NOT_SUPPORTED
- NEVER

其中，根据规范的定义，符合全局事务传播的策略有：MANDATORY、REQUIRED、SUPPORTS。

6.9.4 全局事务配置

全局事务目前对外开放两个 -D 配置参数，说明如下：

1. 参数名 GT_ENABLED，表示是否开启全局事务支持功能，类型为 Boolean，默认值为 false。
2. 参数名 TX_RECOVERY，表示是否开启事务日志和恢复功能，类型为 Boolean，默认值为 false。

7 基础服务配置

7.1 JNDI

7.1.1 TongWeb7 中的 JNDI 概述

JNDI 名称是便于用户使用的对象名称，这些名称通过服务器提供的命名和目录服务绑定到其他对象，JNDI 客户端通过 JNDI API 访问此服务。例如，当创建一个 JNDI 名为 jdbc/oracleds 的数据源时，会在全局的 JNDI 命名空间中，绑定名为 jdbc/oracleds 的数据源对象，JNDI 客户端都可以通过该 JNDI 名称获取数据源对象。

目前 TongWeb7 不支持集成第三方的 JNDI 服务。

7.1.2 InitialContext 的环境属性

JNDI 系统需要设置环境变量，来配置 JNDI 系统的初始化上下文工厂，对象转换工厂等。对于 JNDI 的环境属性配置来说，一般分为本地 JNDI 环境属性和远程 JNDI 环境属性两种，在这些属性中以初始化上下文工厂的配置尤为重要。

7.1.2.1 访问本地资源的初始化上下文环境属性

默认的本地 InitialContext 所需的环境属性不需要设置。如果存在特殊的使用场景下要求必须配置 java.naming.factory.initial 等属性时，可以配置为如下方式，以访问本地 JNDI 资源。

```
java.naming.factory.initial=com.tongweb.naming.java.javaURLContextFactory
```

7.1.2.2 访问远程资源的初始化上下文环境属性

远程的 InitialContext 所需的环境属性为：

```
java.naming.factory.initial=com.tongweb.tongejb.client.RemoteInitialContextFactory
```

```
java.naming.provider.url=http://ip:port/ejbserver/ejb
```

注意：“ip”需要修改为 TongWeb7 服务器所在的 ip。“port”为 ejb 远程调用端口，默认是“5100”。

设置上下文环境属性的代码如下：

```
Properties p = new Properties();
p.put("java.naming.factory.initial",
    "com.tongweb.tongejb.client.RemoteInitialContextFactory");
p.put("java.naming.provider.url", "http://10.10.4.28:5100/ejbserver/ejb");
InitialContext ic = new InitialContext(p);
```

7.1.3 JNDI 命名空间

全局命名空间，即本机或其它机器都可以访问的 JNDI 命名空间，其对应一个 TongWeb 实例。组件命名空间即只能在本组件内部访问的名称空间，即组件之间是相互隔离的。JAVA EE7 规范根据应用部署包的结构，又定义了两种命名空间，模块命名空间和应用命名空间这两种新的命名空间。

7.1.3.1 全局命名空间

java:global - 这个命名空间中的名称被部署在同一个应用程序服务器实例中的所

有应用程序共享。可以看到 JAVA EE7 的规范中的定义，以 java:global 为例，见下面的图 7.1.1。

```
java:global[/<app-name>]/<module-name>/<bean-name>[!<fully-qualified-interface-name>]
```

<app-name> only applies if the session bean is packaged within an .ear file. It defaults to the base name of the .ear file with no filename extension, unless specified by the application.xml deployment descriptor.

<module-name> is the name of the module in which the session bean is packaged. In a stand-alone ejb-jar file or .war file, the <module-name> defaults to the base name of the module with any filename extension removed. In an ear file, the <module-name> defaults to the pathname of the module with any filename extension removed, but with any directory names included. The default <module-name> can be overridden using the module-name element of ejb-jar.xml (for ejb-jar files) or web.xml (for .war files).

<bean-name> is the ejb-name of the enterprise bean. For enterprise beans defined via annotation, it defaults to the unqualified name of the session bean class, unless specified in the contents of the Stateless/Stateful/Singleton annotation name() attribute. For enterprise beans defined via ejb-jar.xml, it's specified in the <ejb-name> deployment descriptor element.

图 7.1.1 EJB 全局名空间的 JNDI 规范

app-name: 指的就是应用名称，所谓的应用名称就是当前应用去掉后缀名的名称。

module-name: 这个 module-name 主要针对的是部署格式为 EAR 的时候，EAR 中子模块的名称。

bean-name: 指的是当前 EJB 的 bean 实现的类名称。

full-qualified-interface-name: 后面为!全包名，这部分是可选的。

全局命名空间示例 1:

当前部署的 war 应用名: myweb.war; EJB name: MyBean; EJB 实现接口为: com.tw.IHello

那么可以通过 lookup 以下 JNDI 名来调用 EJB:

1. java:global/myweb/MyBean
2. java:global/myweb/MyBean!com.tw.IHello(也可以这么进行访问)

全局命名空间示例 2:

当前部署的 EAR 应用名: myear.ear; EJB 子模块的名称为: myejb.jar; EJB name: MyBean; EJB 实现接口为: com.tw.IHello. 可以通过 lookup 以下 JNDI 名调用 EJB: (可以看到其中加了 moduleName 这一个部分)

1. java:global/myear/myejb/MyBean
2. java:global/myear/myejb/MyBean!com.tw.IHello

上述的 JNDI 的路径查找比较麻烦，需要写上全路径的名称，而基于 TongWeb7 来讲，默认针对于每一个部署的 ejb，分配了一个简单的全局 JNDI 名称，但是 JNDI 的名称针对于用例的写法不同对应的 JNDI 名称也不同，区别如下:

1. 如果用例遵循 EJB2 规范，那么其默认 JNDI 的查找名称为 EJB 名称(在 ejb-jar.xml 中的配置的名称) + Local/Remote + Home。

例如: 某 EJB 的名称为 User，并且该 EJB 实现了 Local 接口，那么该 EJB 的 JNDI 名为: UserLocalHome。

2. 如果用例遵循 EJB3 规范，并且该 EJB 实现了 Local 接口，那么其默认 JNDI 名为: EJB 名称 + Local; 如果该 EJB 实现了 Remote 接口，那么 JNDI 名为: EJB 名称 + Remote。

3. 如果某 EJB 是一个无接口的 Bean，那么其默认的 JNDI 名为: EJB 名称+LocalBean。

上述的三种不同的 EJB，其对应的 JNDI 的简写名称是不同的，这块需要注意。

针对于数据源，其实质为 JAVA EE 的资源在 JNDI 树上的映射，对于这种资源来说，

没有像 EJB 这种这么复杂，例如通过管理控制台，或者在 tongweb.xml 中配置一个数据源：

```
<jdbc-connection-pool
  name="testtongweb"
  jdbc-driver="com.mysql.jdbc.Driver"
  jdbc-url="jdbc:mysql://127.0.0.1:3306/test"
  user-name="root"
  password="root"
  jta-managed="true"
  default-auto-commit="true"
  initial-size="10"
  max-active="100"
  max-wait-time="30000"
  validation-query="SELECT 1"
  time-between- eviction-runs="5000"
  min-evictable-idle-time="0"
  validation-interval="30000"
  max-age="0" />
```

可以看到针对这个 testtongweb 数据源，TongWeb7 会在全局名空间中创建 testtongweb 节点，代码中可以直接通过 JNDI 进行查找：

```
InitialContext ctx = new InitialContext();
ctx.lookup("testtongweb");
```

需要注意数据源和 EJB 一样，同一个命名空间下不允许绑定两个 JNDI 名相同的对象。

7.1.3.2 应用命名空间

应用命名空间其意义在于规范 JAVA EE 组件，在不同组件范围下的访问。

java:app - 这个命名空间中的名称被单个应用程序中的所有模块的所有组件共享，“单个应用程序”意思是一个单独的部署单元，比如单个.ear 文件，单个单机部署的模块等等。例如，在同一个.ear 文件中的一个.war 文件和一个 EJB 的.jar 文件都能访问 java:app 命名空间中的资源。

应用命名空间示例 1：

当前部署的 EAR 应用名：myear.ear；EJB name：MyBean；EJB 实现接口为：com.tw.IHello

那么在同一个 EAR 下面，其它的子模块可以通过 lookup 以下 JNDI 名调用该 EJB：

1. java :app/myejb/MyBean
2. java:app/myejb/MyBean!com.tw.IHello

7.1.3.3 模块命名空间

模块命名空间和应用命名空间差不多，同样是 JAVA EE6 的新规范，只不过其范围小于应用命名空间；

java:module - 这个命名空间中的名称被一个模块的所有组件共享(例如，单个 EJB 模块中的所有企业 Bean，或一个 Web 模块中的所有组件)。

模块命名空间示例 1：

当前部署的 EAR 应用名：myear.ear；EJB name：MyBean；EJB 实现接口为：com.tw.IHello.

那么在同一个 EAR 下面的同一个 myejb.jar 下面，其 JAVA EE 组件可以通过 lookup 以下 JNDI 名调用该 EJB：

1. java:module/MyBean
2. java:module/MyBean!com.tw.IHello

7.1.3.4 组件命名空间

TongWeb7 中的 JNDI 不仅支持全局命名空间，应用命名空间，模块命名空间，还支持

组件命名空间。对于组件命名空间来说，组件命名空间的标准上下文环境是 java:comp/env，应用组件使用 java:comp/env 查找对象时，不必关注对象真正的 JNDI 名字，只需要在部署描述文件中配置该对象的引用名到 JNDI 名的映射，降低应用组件代码与 JNDI 名称的耦合性。

说明：访问组件命名空间中 JNDI 名前缀为“java:comp/env”。

7.1.4 JNDI 树展示

TongWeb7 的管理控制台提供 JNDI 树，便于用户查找现有的 JNDI 信息。TongWeb7 管理控制台分为四个视图进行展现。

7.1.4.1 JNDI 展示

1. 展开管理控制台左侧导航树中的 JNDI 节点，出现如下图所示



图 7.1.1 JNDI 导航树

点击服务器资源域显示如下：



图 7.1.2 服务器资源域展示

点击远程 EJB 域显示如下：



图 7.1.3 远程 EJB 域展示

点击本地 EJB 域显示如下：



图 7.1.4 本地 EJB 域展示

点击应用 global 域显示如下



图 7.1.5 应用 global 域展示

点击应用域，如下图显示：



图 7.1.6 应用域展示

global 命名空间是所有应用的资源绑定空间，按照应用名分类存放，通过该空间可访问任何应用定义的资源，访问时需要指定应用名（ear 应用需要，web 和 ejb 则不需要）来区分查找哪个应用。

app 命名空间是某应用内部资源的绑定空间，按照模块名分类存放，通过该空间可访问到同一个应用下任何模块定义的资源，访问时需要指定模块名来区分查找哪个模块。

module 命名空间是某模块（ejb 模块、web 模块）内部资源的绑定空间，通过该空间可访问到同一个模块下的其它资源。（为兼容旧规范，java:module 命名空间下绑定了应用中所有模块定义的资源），访问时只需要指定资源名即可（无需应用名和模块

名)。

`comp` 命名空间绑定了对任何一个组件可见的资源，包括服务提供的资源和应用自定义的资源，服务提供的资源绑定在 `java:comp/命名空间` 下，应用自定义的资源绑定在 `java:comp/env` 命名空间下。

7.1.5 代码中使用 JNDI 的示例

示例目标：Java 客户端通过 `InitialContext` 的环境属性访问 TongWeb7 服务器中已部署 EJB 的 JNDI 名。

示例步骤如下：

(一) 部署 EJB

具体步骤见“应用管理”中的[应用部署](#)；

(二) 编写 Java 客户端访问 EJB 应用

```
public static void main(String[] args) {
    Properties p = new Properties();
    p.put("java.naming.factory.initial",
"com.tongweb.tongejb.client.RemoteInitialContextFactory");
    p.put("java.naming.provider.url",
"http://10.10.4.28:5100/ejbserver/ejb");
    StatelessSessionRemote ClientA;
    Context ctx;
    try {
        ctx = new InitialContext(p);
        ClientA = (StatelessSessionRemote)
ctx.lookup("StatelessSessionRemote ");
        ClientA.SetName("ClientA");
        System.out.print("Invoke the remote interface of SLSB, result is="
+ ClientA.getName() + "<br>");
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

7.1.6 应用移植

JNDI 的调用，其主要使用在应用移植中，应用中已经写好了 JNDI 查询语句，最大程度的减少代码的修改。

7.1.6.1 全局 JNDI 名应用移植

针对于全局的 JNDI 名称，以 EJB 为例，TongWeb7 针对于不同的 EJB 定义类型，其默认绑定的全局 JNDI 名称是不同的；([见全局命名空间对应章节](#))

全局 JNDI 名应用移植示例：

当前部署的 war 应用名：`myweb.war`；EJB name：`MyBean`；EJB 实现接口为：`com.tw.IHello`；对应的本地 EJB 版本为 EJB3；可以将原有代码修改为：

```
1. context.lookup("java:global/myweb/MyBean");
2. context.lookup("java:global/myweb/MyBean!com.tw.IHello");
3. context.lookup("java:app/myweb/MyBean");
4. context.lookup("java:app/myweb/MyBean!com.tw.IHello");
5. context.lookup("java:module/MyBean");
6. context.lookup("java: module/MyBean!com.tw.IHello");
7. context.lookup("MyBeanLocal");
```

而如果想要不修改代码的前提下，可以配置自定义部署描述文件：

举例来说，部署在某 JavaEE 应用服务器上的一个应用想要移植到 TongWeb7 上，该应用包含了一个实现了 `com.test.ejb3examples.ejb.ManagerLocal` 接口的 EJB，并且在

该 JavaEE 应用服务器上定义了这个 EJB 的 JNDI 名为 myManager ， 在应用中调用这个 EJB 的代码如下：

```
Context context = new InitialContext();
context.lookup("myManager");
```

在将该应用移植到 TongWeb7 时，无需修改代码，只需要在 TongWeb7 的自定义部署描述文件 tongweb-ejb-jar.xml 中定义该 EJB 的 JNDI 名为 myManager，例如：

```
<tongweb-ejb-jar>
  <ejb-deployment ejb-name="ManagerRBean">
    <jndi name="myManager"
      interface="com.test.ejb3examples.ejb.ManagerLocal"/>
  </ejb-deployment>
</tongweb-ejb-jar>
```

然后将 tongweb-ejb-jar.xml 拷贝到该 EJB 所在 jar 包的 META-INF 目录下

7.1.6.2 组件 JNDI 名应用移植

应用通过组件命名空间查找，需要配置 web.xml 中 ejb-ref 或者 resource-ref 与自定义部署描述文件 jndi 节点或者 resource-link 的映射。这里以数据源为例(EJB 配置 ejb-ref 与数据源同理)，针对于数据源的资源映射，在 web.xml 中配置数据源 myDS，如下：

```
<resource-ref>
  <res-ref-name>myDS</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
</resource-ref>
```

并且在 tongweb-web.xml 自定义部署描述文件中，元素 resource-link 中的 name 与 web.xml 中的 res-ref-name 一致，如下：

```
<tongweb-web-app>
  <resource-links>
    <resource-link
      name="myDS"//需要与上面 web.xml 中的 res-ref-name 一
      type="javax.sql.DataSource"
      global="mypool" /> //此处需与全局数据源 id 一致
    </resource-links>
  </tongweb-web-app>
```

调用数据源的代码如下：

```
Context context = new InitialContext();
context.lookup("java:comp/env/myDS");
```

7.2 安全服务

7.2.1 安全服务概述

安全服务用于对数据进行保护：在存储和传输数据时，防止对数据进行未经授权的访问。认证是一种实体（用户、应用程序或组件）用来确定另一个实体是否是其声明的实体的方法，实体使用安全凭证对其自身进行验证。授权是确定使用凭证的实体是否有权对所访问的资源进行操作的方法。

7.2.2 TongWeb7 中的安全服务

7.2.2.1 安全域

安全域是服务器定义和强制执行通用安全策略的范围，是存储用户和组信息及其关联的安全凭证的系统信息库。服务器支持六种类型的安全域：

- 文件安全域：服务器将用户凭证存储在本地文件中。

- LDAP 安全域：服务器从轻量目录访问协议 (LDAP) 服务器中获取用户凭证。支持的 ldap 服务器有 OpenLDAP 等。
- JDBC 安全域：服务器从数据库中获取用户凭证。支持的数据库为数据源所支持的数据库。
- Jaas 安全域：服务器从自定义的 LoginModule 中获取用户凭证，支持用户使用自定义的 LoginModule 灵活的进行安全域的验证。
- 脚本安全域：服务器按照 JSR223 规范从脚本（如 js 脚本）中获取用户凭证。
- 服务扩展安全域：认证过程由用户具体实现类完成，用户类必需实现 TongWeb 服务器指定的接口“LoginProvider”。

7.2.3 安全服务的使用

7.2.3.1 安全域概述

为应用程序绑定特定的安全域，需要在部署描述文件中定义该安全域的名称。如 web.xml 中作如下描述：

```
<security-constraint>
  <display-name>Example Security Constraint</display-name>
  <web-resource-collection>
    <web-resource-name>Protected Area</web-resource-name>
    <!-- 指定需要保护的资源的 url（相对于应用前缀） -->
    <url-pattern>/jsp/security/protected/*</url-pattern>
    <!-- 指定需要保护的方法 -->
    <http-method>DELETE</http-method>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
    <http-method>PUT</http-method>
  </web-resource-collection>
  <auth-constraint>
    <!-- 指定能访问受保护资源的角色 -->
    <role-name>userRole</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <!-- 验证类型，BASIC、FORM、DIGEST 或 CLIENT-CERT -->
  <auth-method>FORM</auth-method>
  <!-- 应用所使用的安全域名，对应于在服务器中配置的安全域 -->
  <realm-name>myAuthRealm</realm-name>
  <form-login-config>
    <form-login-page>/jsp/security/protected/login.jsp</form-
login-page>
    <form-error-page>/jsp/security/protected/error.jsp</form-
error-page>
  </form-login-config>
</login-config>
```

还需要在服务器自定义的部署描述文件中定义所关联的用户或组。

如 tongweb-web.xml 中作如下描述：

```
<security-role-mapping>
  <!-- 将 web.xml 中描述的角色 userRole 与服务器中相应的用户关联起来 -->
  <role-name>userRole</role-name>
  <!-- 服务器安全域中定义的用户 -->
  <principal-name>admin:guest</principal-name>
</security-role-mapping>
```

或者:

```
<security-role-mapping>
  <!-- 将 web.xml 中描述的角色 userRole 与服务器中相应的用户组关联起来 -->
  <role-name>userRole</role-name>
  <!-- 服务器安全域中定义的组 -->
  <group-name>twnt</group-name>
</security-role-mapping>
```

当多个角色需要关联多个用户或者组时，多个角色、用户或者组都可以使用冒号分隔，从而实现一对多、多对一或者多对多的角色关联。

7.2.3.2 安全域基本属性

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。
2. 点击“创建安全域”，打开“创建安全域”页面，配置属性如下：

返回	编辑安全域	
安全域名称	defaultRealm	安全域名称
角色模型	<input checked="" type="radio"/> strict <input type="radio"/> authOnly <input type="radio"/> strictAuthOnly	角色模型
指定加载方式	<input checked="" type="radio"/> 应用类加载器 <input type="radio"/> 服务器类加载器	指定加载方式
X509证书解析器实现类	<input type="text" value="com.tongweb.catalina.realm.X509St"/>	X509证书解析器实现类名，默认值为com.tongweb.catalina.realm.X509St
是否使用锁定机制	<input checked="" type="radio"/> 是 <input type="radio"/> 否	是否使用锁定机制
允许错误次数	<input type="text" value="5"/> 次	允许错误次数，默认为5次
锁定超时	<input type="text" value="300"/> 秒	用户被锁定的时间，默认为300秒
缓存大小	<input type="text" value="1000"/> 个	认证失败的用户缓存数量，默认为1000个
最小缓存时间	<input type="text" value="3600"/> 秒	被锁定的用户在缓存中的时间少于此值，则
安全域类型	<input checked="" type="radio"/> File <input type="radio"/> SQL <input type="radio"/> LDAP <input type="radio"/> JAAS <input type="radio"/> Script <input type="radio"/> SPI	安全域类型

图 7.2.1 创建安全域

基本属性:

- 安全域名称：必填项，安全域的唯一标识。
- 角色模型：是指处理 web.xml 中通配符*角色的方式，strict 模式是指用户必须被认证且必需至少具备 web.xml 预定义的角色中的一种；authOnly 模式是指只需要用户被认证，不必需用户具备任意角色；strictAuthOnly 模式是指用户必须被认证，如果 web.xml 中没有预定义任何角色则不需要检查用户角色，否则，用户必需至少具备 web.xml 预定义的角色中的一种。
- 指定加载方式：如果选择应用类加载则 LoginModule 等相关类可以放在 web/lib 下，而选择服务器类加载，类将从 TW_HOME/lib 下加载。
- X509 证书解析器实现类名：当使用 CLIENT-CERT 认证时，指定这个类名，用来从证书中获取用户名，默认以证书的 Owner 描述（如 CN=tongtech, OU=tongweb, O=tongweb, L=beijing, ST=bj, C=cn）作为用户名，注意：SSL 机制只负责认证，即获取到客户端证书里面的用户名，而该用户名对应哪些角色需要由

具体的安全域来实现。另外要使用该认证强制要求服务器端开启 https 协议连接器，并且开启客户端认证（服务器端配置的 truststoreFile 必需导入客户端证书，即必需有 Entry type: trustedCertEntry 而不能只有 Entry type: PrivateKeyEntry），应用的 web.xml 最佳实践是配置<transport-guarantee>INTEGRAL 或 CONFIDENTIAL</transport-guarantee>使用，这样即使使用 http 协议访问也会强制重定向到 https 协议。

- 是否使用锁定机制：如果选择是，则可以配置下面的四个属性从而实现锁定机制。
- 允许错误次数：允许用户输入的用户名、密码或者其他凭证连续错误的次数，如果达到该次数，则在锁定超时配置的时间范围内，该用户无法再次尝试登陆，均返回登陆失败。
- 锁定超时：达到允许错误次数后，用户被锁定的时间，默认为 300 秒。
- 缓存大小：认证失败的用户会被存放在一个缓存中，当记录的失败用户太多，则移除缓存中存放时间最长的用户。此值为缓存的初始值，默认为 1000。
- 最小缓存时间：当认证失败用户数超过缓存集合，判断将被移除的用户的缓存时间是否小于此值。如果小于，写警告日志，表示该用户被移除的太快了，默认 3600 秒。
- 安全域类型：提供文件、JDBC、LDAP、Jaas、服务扩展、脚本六种类型安全域。

7.2.3.3 创建文件安全域

1. 创建文件安全域

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。
2. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“File”来创建文件安全域，配置基本属性以及文件安全域特有属性：

The screenshot shows a configuration form for creating a security domain. At the top, there are two labels '安全域类型' (Security Domain Type) with a set of radio buttons: File (selected), SQL, LDAP, JAAS, Script, and SPI. Below this, there are two input fields: '用户文件名' (User filename) with the value 'users.properties' and a description '用户文件名，默认值为 安全域名称.users.properties'; and '组文件名' (Group filename) with the value 'groups.properties' and a description '组文件名，默认值为 安全域名称.groups.properties'. At the bottom left, there are two buttons: '保存' (Save) and '取消' (Cancel).

图 7.2.2 创建文件安全域

文件安全域属性：

- 用户文件名：指定一个存放用户名/密码信息的文件名，缺省存放在 conf/security 目录下，默认为 users.properties。
 - 组文件名：指定一个存放用户/组信息的文件名文件路径，缺省存放在 conf/security 目录下，默认为 groups.properties。
3. 设置属性完成后，点击“保存”按钮。
- #### 2. 为文件安全域创建用户
1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。
 2. 直接点击对应文件安全域右侧的“管理用户”，进入用户管理页面，如图所示：

此页显示了已创建的用户列表，通过首先选择用户名，然后使用此页中的控件，可以从此域中更新或删除所创建的用户。

用户名称	用户所在组名称
<input type="checkbox"/> thanos	tongweb
<input type="checkbox"/> manager	tongweb
<input type="checkbox"/> twns	tongweb
<input type="checkbox"/> twnt	tongweb
<input type="checkbox"/> root	cli
<input type="checkbox"/> tongweb	tongweb

图 7.2.3 用户、组信息

1. 点击“创建用户”按钮，即出现如图 7.2.4 所示的创建用户界面：

返回
创建用户

用户名称

密码加密摘要算法 SM3 MD5 SHA-1
 SHA-256 SHA-512 DIGEST

加强字节长度

迭代加密次数

密码长度

密码

确认密码

密码最短使用期限 天

密码最长使用期限 天

用户名称

密码加密摘要算法

将密码和一些随机数字混合加密，增加强度

迭代多次加密，增加强度

密码长度限制

密码

确认密码

密码可修改前最短使用期限，默认值0，表示可立即修改

密码过期时间，默认值2147483647天

保存
取消

图 7.2.4 创建用户

创建用户属性：

- 用户名称：必须创建，新建用户的名称。
- 所属组名：必须创建，用户所在的用户组。
- 密码加密摘要算法：用户密码的加密方式，NONE 表示不加密，当 web.xml 中使用验证类型为 DIGEST 的时候，密码的加密方式应选择 DIGEST。
- 加强字节长度：将密码和一些随机数字混合加密，增加强度。
- 迭代加密次数：迭代多次加密，增加强度。
- 密码长度：密码最小位数
- 密码：用户对应的登陆密码。
- 确认密码：再次输入密码，以防止密码输入错误。

7.2.3.4 创建 LDAP 安全域

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。
2. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“LDAP”来创建 LDAP 安全域，配置如下属性：

安全域类型	<input type="radio"/> File <input type="radio"/> SQL <input checked="" type="radio"/> LDAP <input type="radio"/> JAAS <input type="radio"/> Script <input type="radio"/> SPI	安全域类型
服务器主机名	<input type="text" value="ldap://127.0.0.1:389"/>	服务器主机名，形式如：ldap://127.0.0.1:389
LDAP baseDN	<input type="text"/>	连接ldap服务器的root域
服务器密码	<input type="text"/>	ldap服务器密码
用户基本域名	<input type="text" value="ou=people,dc=tongweb,dc=com"/>	用户基本域名
用户名属性	<input type="text" value="uid"/>	用户名属性，默认值为uid
密码属性	<input type="text" value="userPassword"/>	密码属性，默认值为userPassword
角色的基本域名	<input type="text" value="ou=groups,dc=tongweb,dc=com"/>	角色的基本域名
角色名属性	<input type="text" value="cn"/>	角色名属性，默认值为cn
角色对应用户属性	<input type="text" value="uniqueMember"/>	角色对应用户属性，默认值为uniqueMember

图 7.2.5 创建 LDAP 安全域

LDAP 安全域属性：

- 服务器主机名：必填项，登录 ldap 服务器的主机名。默认是 ldap://127.0.0.1:389
 - LDAP baseDN：访问 LDAP 目录的基准 DN。例如 ou=myou, o=myorg.com，其中 ou 为组织单元，o 为组织。
 - 服务器密码：连接 LDAP 服务器的用户口令，缺省为匿名连接。
 - 用户基本域名：必填项，用于查询用户对应目录的基本 DN。默认为 ou=people, dc=tongweb, dc=com
 - 用户名属性：通过该属性可以查询到用户名，默认是 uid。例如：userIdAttribute=uid，在 LDAP 中 uid=zhangsan，那么，通过查询 uid，就可以得到用户名 zhangsan。
 - 密码属性：定义查询密码的属性。
 - 角色的基本域名：用于查询角色对应目录的基本 DN，如 ou=groups, dc=mycompany, dc=com。默认是 ou=groups, dc=tongweb, dc=com
 - 角色名属性：通过该属性可以查询到角色名。例如：roleNameAttribute=cn。在 LDAP 中 cn=admin，那么，通过查询 cn，就可以得到角色名 admin。
 - 角色对应用户属性：通过该属性可以查询到用户。
3. 设置属性完成后，点击“保存”按钮。

7.2.3.5 创建 JDBC 安全域

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。

2. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“SQL”来创建 JDBC 安全域，配置如下属性：

安全域类型 File SQL LDAP 安全域类型

JAAS Script SPI

jdbc数据源名称	<input type="text" value="不使用"/>	jdbc数据源名称
jdbcURI	<input type="text"/>	jdbcURI
jdbc驱动名	<input type="text"/>	jdbc驱动名
数据库用户名	<input type="text"/>	数据库用户名
数据库密码	<input type="text"/>	数据库密码
userSelect	<input type="text" value="SELECT user_name, user_pass FROM users WHERE user_name = ?"/>	默认值为SELECT
groupSelect	<input type="text" value="SELECT user_name, role_name FROM user_roles WHERE user_name = ?"/>	默认值为SELECT

图 7.2.6 创建 JDBC 安全域

JDBC 安全域属性：

- jdbc 数据源名称：选择使用数据源后，可以直接通过配置的数据源进行数据库的连接，而不再需要配置下面的数据库连接信息。
 - jdbcURI：必填项，连接数据库的 URL。
 - jdbc 驱动名：必填项，数据库驱动的类型名。
 - 数据库用户名：必填项，连接数据库所需的用户名。
 - 数据库密码：连接数据库所需的密码。
 - userSelect：sql 语句，查询结果为满足条件的用户和密码。
 - groupSelect：sql 语句，查询结果为用户名和用户所在组。
3. 设置属性完成后，点击“保存”按钮。

说明：用于存放用户信息和组信息的数据库表需要用户手动创建，例如可以在数据库中配置用户表和组表如下：

JDBC 表结构：

用户表 usertable

列名	约束
userid	主键, not null
password	not null

组表 grouptable

列名	约束
userid	主键, not null
groupid	not null

注意：用户表 usertable 的主键 userid 是组表 grouptable 的外键，其中 usertable 和 grouptable 的 userid 必须是相同的。

7.2.3.6 创建 Jaas 安全域

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。

1. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“JAAS”来创建自定义安全域，配置如下属性：

安全域类型 File SQL LDAP JAAS Script SPI

LoginModule名称 Login

自定义用户principal 自定义

自定义组principal 自定义

图 7.2.7 创建自定义安全域

自定义安全域属性：

- LoginModule 名称：必填项，LoginModule 的名称。安全域通过此 loginModule 进行安全验证。
 - 自定义用户 principal：用户自定义的用户权限封装类。
 - 自定义组 principal：用户自定义的角色权限封装类。
2. 设置属性完成后，点击“保存”按钮。

说明：用户自定义的 LoginModule 实现类、用户 principal 实现类和组 principal 实现类的 jar 包需要根据配置的加载方式放在 TongWeb7 安装路径下的 lib 文件夹内或者应用的 web/lib 文件夹下。

7.2.3.7 创建脚本安全域

展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。

1. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“Script”来创建脚本安全域，配置如下属性：

安全域类型 File SQL LDAP JAAS Script SPI

脚本文件 脚本文件的路径

脚本引擎 脚本引擎的类型

图 7.2.7 创建脚本安全域

自定义安全域属性：

- 脚本引擎：脚本引擎的名称，默认使用 Javascript 引擎。
 - 脚本文件：脚本文件名称，默认为 loginscript.js。
2. 设置属性完成后，点击“保存”按钮。

说明：脚本编写需要符合服务器要求，在脚本中需要接收并处理用户输入的 user 和 password 两个参数，并必需于最后返回 List<String> 类型的对象，其中包含了指定用户的角色列表，脚本执行过程抛出异常或者返回 List<String> 类型的对象为空，则视为不为用户分配角色。

以下是示例的 js 脚本：

```
var users = [
    {"user": "admin", "password": "admin", "roles": "admin"}
];
```

```

var list = new java.util.ArrayList();

var loginOk = false;
for (var i = 0; i < users.length; i++) {
    var u = users[i];
    if (u.user == user && u.password == password) {
        var roles = u.roles.split(",");
        for (var j = 0; j < roles.length; j++) {
            list.add(roles[j]);
        }
        loginOk = true;
        break;
    }
}
if (!loginOk) {
    throw new Error("User or Password does not match");
}
// return to tongweb, type: List<String>
list;

```

上述示例脚本中在 `users` 中预设了 `admin` 用户，脚本接收并验证服务器提供的 `user` 和 `password` 两个参数，最终返回的 `List<String>` 类型的对象 `list` 包含了该 `user` 用户的角色列表。

7.2.3.8 创建服务扩展安全域

1. 展开管理控制台左侧导航树中的“安全服务”节点，并点击“安全域管理”节点，可以在该页面看到已配置的安全域。
1. 点击“创建安全域”，打开“创建安全域”页面，在安全域类型属性一栏选择“SPI”来创建服务扩展安全域，配置如下属性：

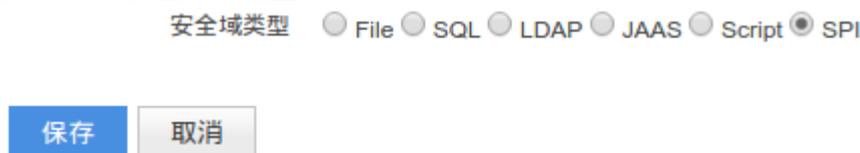


图 7.2.8 创建服务扩展安全域

2. 设置属性完成后，点击“保存”按钮。

说明：服务扩展安全域是指用户可以通过实现服务器提供的接口来提供用户凭证，该接口为 `LoginProvider`，需要实现一个方法：`List<String> authenticate(String user, String password)`，该方法的实现逻辑大致为接收并处理用户输入的 `user` 和 `password` 两个参数，并返回 `List<String>` 类型的对象，其中包含了指定用户的角色列表。

用户的实现类需要安装 `JAVA SPI` 机制在 `jar` 包内的 `META-INF/services` 文件夹下进行配置。

7.2.3.9 SSL 证书方式认证和授权

`web.xml` 中 `security-constraint` 节点的 `user-data-constraint` 节点也可以指定使用传输层保护，当配置其 `transport-guarantee` 元素的属性为 `INTEGRAL` 或者 `CONFIDENTIAL` 时，要求应用的受保护资源必须通过 `https` 协议访问。

同时，在 `web.xml` 中，`login-config` 节点可以配置 `BASIC`、`DIGEST`、`FORM` 和 `CLIENT-CERT` 四种认证方式，当使用 `CLIENT-CERT` 方式时，要求应用的受保护资源必须通过 `https` 协议访问。该情况下安全域的认证过程就由 `SSL` 证书认证完成，而安全域只

完成授权过程。

传输层安全

在应用的 web.xml 中配置 security-constraint 节点如下，则要求应用下的受保护资源通过 https 才能够访问。

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      My App
    </web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>CONFIDENTIAL</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

此时用户访问受保护的资源则需要 SSL 证书认证。如果 https 通道配置为单项认证则只需要正确配置服务器端证书；如果是双向认证则在客户端也需要正确配置相关证书。

SSL 证书认证

在应用的 web.xml 中配置 login-config 节点如下，则要求应用下的所有资源通过 https 才能够访问，安全域的认证过程由 SSL 证书认证完成，而安全域只完成授权过程。

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>
      My App
    </web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>test</role-name>
  </auth-constraint>
</security-constraint>
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>twnt-realm</realm-name>
</login-config>
```

在上面的配置中，认证方式为 CLIENT-CERT，所用安全域为 twnt-realm，只有 test 角色的用户才能够访问受限资源。此时用户使用 SSL 证书认证方式需要正确配置以下几点：

1. 根据 https 通道是单向认证还是双向认证正确配置好服务器端和客户端的 ssl 证书（单向认证不需要配置客户端证书）。
2. 用户名依赖于配置的 X509 证书解析器实现类，默认为 X509SubjectDnRetriever，该类将证书的域名称作为用户名，然后进行授权过程。因为授权与角色映射相关，因此可以通过用户名-角色映射和组-角色映射两种方式。用户名-角色映射可以直接通过配置 tongweb-web.xml 实现；组-角色映射需要在组文件中分配相应证书解析器解析的用户名对应组名，例如，证书的 dname 为 CN=tongtech, OU=tongweb, O=tongweb, L=beijing, ST=beijing, C=cn，在文件安全域的组 properties 文件中添加一行：CN\=tongtech, \ OU\=tongweb, \ O\=tongweb, \ L\=beijing, \ ST\=beijing, \ C\=cn=tongweb，分配该认证用户的组为 TongWeb，然后通过配置 TongWeb 组与所需角色（上述配置文件为 test）的映射即可。

7.2.3.10 安全管理器

服务器使用 Java 安全管理器对服务器资源的访问作进一步控制。JVM 的安全机制需要一个安全策略文件 (.policy) 来定义访问权限，这些权限定义了运行在一个 JVM 实例中的类是否可以执行某项运行时操作。

使用安全管理器能提高系统安全性，但会对某些应用的性能造成较大影响。当服务器存在如下条件时，并不推荐使用安全管理器：

- 对性能有较高要求
- 较好地控制了应用的部署，包括对各应用的访问策略文件的控制
- 只部署受信赖的应用
- 所部署的应用不要求强制执行访问策略控制

安全管理器由一个安全策略文件来定义权限。在启动服务器时用 `-Djava.security.policy` 属性指定安全策略的全路径名。

服务器提供了缺省的安全策略文件 `TW_HOME/conf/tongweb.policy`，该文件提供了一组缺省的权限，用户可以在此基础上创建自己的安全策略文件。

如果需要这个安全策略文件生效，需要在 `java` 启动参数中增加 `-Djava.security.manager`。比如在启动参数配置文件 `external.vmoptions` 为例：增加 `-Djava.security.manager`

```
-Djava.security.policy=${TongWeb_Home}/conf/tongweb.policy
```

例如，如果需要修改或增加对目录权限的控制，可以通过修改 `tongweb.policy` 进行控制。以在 windows 下增加 C: 盘的权限为例，在 `tongweb.policy` 这个配置文件 `grant {}` 里面添加：

```
permission java.io.FilePermission "c:${/}-", "read,write,delete,execute";
```

`read` 对应读取权限，`write` 对应写入权限，`delete` 对应删除权限，`execute` 对应执行权限，可以根据自己需要配置权限。

`TongWeb7` 的缺省的安全策略文件对管理控制台提供了缺省的权限，以便能够在保证安全的情况下对应用服务器进行监视、管理等。例如，当创建数据源时，如果指定的驱动路径并不在 `TW_HOME/applications/console` 或其子目录下，则该驱动 `jar` 可能由于缺少相应的权限而创建失败。如果信任或者必须使用该驱动 `jar`，可以将该 `jar` 文件拷贝至 `TW_HOME/lib` 下，或者在 `tongweb.policy` 文件中添加该驱动 `jar` 所需的相应权限。

当安全策略生效时，用户自定义部署的应用也需要配置相应的权限才能正确访问。

7.2.4 管理控制台三员分立

涉密信息系统中的“三员”是指系统管理员、安全保密管理员、安全审计员。“三员分立”要求系统管理员、安全保密管理员、安全审计员三者之间的关系相互独立、互相制约，加强涉密信息系统保密管理，减少泄密风险。而三员分立权限控制模型就是基于“三员分立”的思想提出来的权限管理方法。

`TongWeb` 管理控制台便采用了此种方法的设计和实现来确保系统的安全保密。

7.2.4.1 系统管理员

系统管理员拥有的角色权限是可以对 `TongWeb` 进行各种维护、部署、监控及配置操作。

系统管理员在默认系统安全域中可以创建用户、编辑用户、不能分配角色权限及删除用户。

默认系统管理员用户名为 `thanos`，密码为 `thanos123.com`。

系统管理员对用户安全方面的操作如下：

1. 进入【安全服务】安全域列表页面。
2. 选择【defaultRealm】--【管理用户】（只有这个安全域是针对控制台的三员分立）。

3. 用户列表：列表页面显示已创建的所有三员用户。

列表字段显示包括用户名称、用户所属角色（系统管理员、安全保密管理员、安全审计员，若还未分配角色则显示为空）。



图 7.2.9 系统管理员管理默认安全域用户列表页

4. 用户创建：

支持输入用户名、密码、加密方式、密码使用期限设置。



图 7.2.10 系统管理员创建用户页面

5. 用户编辑：

支持修改密码长度、密码、密码使用期限（用户名由安全保密管理员修改）。



图 7.2.11 系统管理员编辑用户页面

7.2.4.2 安全保密管理员

安全保密管理员拥有的角色权限是可以对 TongWeb 三员用户进行角色的分配、删除用户。默认用户名为 security，密码为 security123.com，角色为安全保密管理员。

安全保密管理员可以做的操作如下：

1. 进入【管理用户】三员用户管理列表页面
2. 用户列表：列表页面显示已创建的所有三员用户。

列表字段显示包括用户名称、用户所属角色（系统管理员、安全保密管理员、安全审计员，若还未分配角色则显示为空）。



图 7.2.12 安全保密管理员用户列表页面

3. 用户编辑（即分配角色）：

支持用户名称、分配角色（下拉列表的方式展示无角色、系统管理员、安全保密管理员、安全审计员）以及角色内的用户权限分配



图 7.2.13 安全保密管理员用户编辑页面

4. 用户删除。

7.2.4.3 安全审计员

系统管理员及安全保密管理员进行的日常操作都会记录审计日志。主要负责对系统管理员和安全保密员的操作行为进行审计跟踪、分析和监督检查，及时发现违规行为，并定期向系统安全保密管理机构汇报情况。

TongWeb 中的安全审计员功能是查询已经生成的审计日志。默认用户名为 auditor，密码为 auditor123.com，角色为安全审计员。

具体安全审计员登陆后的使用方式请见 [7.5.6 审计日志](#)。

7.3 诊断

诊断服务用于诊断 TongWeb7 服务器运行时可能出现的问题。通过诊断服务，可以创建、收集和访问由正在运行的服务器及其部署的应用程序生成的诊断数据。通过这些数据来诊断和剖析服务器运行中出现的问题，如：异常、性能问题及其它故障。

诊断服务的功能模块有，系统日志、SQL 日志、访问日志、快照、snmp。

7.3.1 系统日志

系统日志记录了 TongWeb7 服务器的运行状态。通过分析系统日志的错误信息，可帮助查找系统出错原因。以及通过日志的时间间隔找到耗时较多的系统操作，以便诊断系统故障原因和性能瓶颈。

7.3.1.1 查看系统日志

1) 展开左侧导航栏点击“诊断”节点，双击“系统日志”，即展现出所有的系统日志，时间最近的排在前面；如下图 7.3.1 所示：



图 7.3.1 查看系统日志

7.3.1.2 下载日志

- 1) 在系统日志界面中，点击“下载日志”，弹出下拉框列表，列表中的文件包括当前的系统日志文件以及已经轮转的日志文件。可以选择一个或多个日志文件进行下载，如下图 7.3.2 所示：



图 7.3.2 选择下载的系统日志

- 2) 在下拉框中选择完所要下载的日志文件，点击右侧的“下载日志”，会弹出 log.zip 的下载文件，如下图 7.3.3 所示：

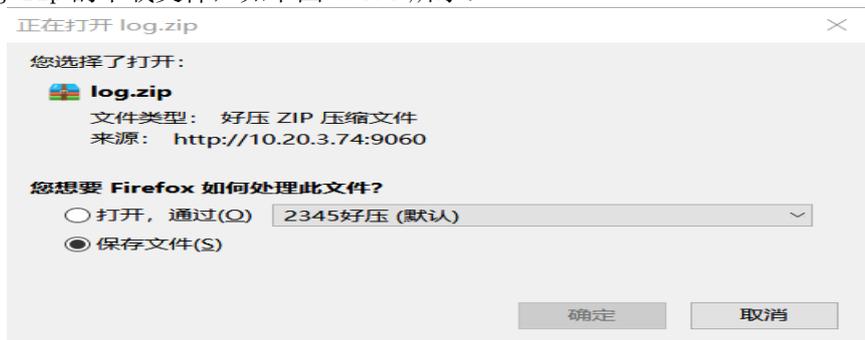


图 7.3.3 下载系统日志

7.3.1.3 搜索日志

在系统日志界面中，点击“搜索日志”，即弹出时间范围的下拉选择框。

- 1) 该时间范围是通过分析系统日志文件中记录的日志时间而动态生成的，从而确保每次的日志搜索在较短的耗时内完成。
- 2) 当单个系统日志文件大于 50M 时，仅提供该日志的下载功能；

- 3) 进行搜索日志时，可以通过自定义时间段、日志级别、日志来源、日志信息来对要查找的内容进行过滤。如下图 7.3.4 所示：

The screenshot shows the 'System Log' interface. At the top, there's a header '系统日志' and a sub-header '系统日志信息'. Below this is a message: '此页默认显示最近1000条系统日志记录，可以通过搜索日志功能查找满足条件的日志'. There are two tabs: '下载日志' and '搜索日志'. Under '搜索日志', there are several filters: '时间范围' (Time Range) set to '2020-04-10 11:15:43--2020-04-17 14:18:09', '自定义时间段' (Custom Time Range) with start '2020-04-10 11:15:43' and end '2020-04-17 14:18:09', '日志级别' (Log Level) set to 'ALL', '日志来源' (Log Source) set to 'ALL', and a '日志信息' (Log Message) search box. A blue '搜索' (Search) button is below. Below the filters is a table with columns: '时间', '日志级别', '日志来源', and '日志信息'. The table contains three rows of log entries.

时间	日志级别	日志来源	日志信息
2020-04-17 14:12:09	SEVERE	data-source	Unable to create initial connections of pool.
2020-04-17 14:12:09	WARNING	System.out	at java.util.TimerThread.run(Timer.java:505)
2020-04-17 14:12:09	WARNING	System.out	at java.util.TimerThread.mainLoop(Timer.java:555)

图 7.3.4 日志

7.3.2 SQL 日志

SQL 日志记录了 SQL 语句处理所耗费的时间信息。通过分析 SQL 日志可以找出处理耗时多的 SQL 语句，来诊断系统性能瓶颈。

- 1) 展开左侧导航栏点击“诊断”节点，双击“SQL 日志”，即展现出所有的 SQL 日志信息，如下图 7.3.5 所示：

The screenshot shows the 'SQL Access Log' interface. At the top, there's a header 'SQL访问日志' and a sub-header 'SQL日志信息'. Below this is a message: '此页显示了由TongWeb中JDBC连接池生成的并且满足过滤条件的SQL日志信息。'. There are two tabs: '1小时内' and '自定义'. Below the tabs are filters: '处理时长' (Processing Time) with options '500ms', '2s', '自定义', and a value '0' in a box, followed by '(ms)' and a '确定' (Confirm) button. Below the filters is a table with columns: '时间', 'SQL语句', and '请求处理时间'. The table contains two rows of log entries.

时间	SQL语句	请求处理时间
2015-05-04 18:11:59	SQL execut 47 ms : drop table account0	47
2015-05-04 18:11:59	SQL execut 18 ms : create table account0(accno varchar2(20),customer varchar2(30),balance number(18, 2))	18

图 7.3.5 查看 SQL 日志

- 2) 在 SQL 访问日志界面中，可以对 SQL 访问日志信息内容过滤，分别通过时间、处理时长、自定义时间来过滤 SQL 访问日志内容。

7.3.3 访问日志

访问日志记录的是虚拟主机的访问日志，默认虚拟主机 admin 记录的是控制台上操作的相关信息，server 以及新建虚拟主机记录的是访问 Web 应用时 HTTP 请求的相关信息，包括访问处理时长，访问连接，访问 IP，请求方式等。通过分析访问日志，可以找出处理耗时多的请求，以便诊断系统性能瓶颈。

- 1) 展开左侧导航栏点击“诊断”节点，双击“访问日志”，默认展现出 admin 虚拟主机所有的访问日志信息，注意，如果修改虚拟主机访问日志的路径，需要重启 TongWeb 才能显示新路径下的日志。如下图 7.3.6 所示：

时间	访问处理时长 (ms)	访问链接	访问IP	请求方式
2020-09-08 09:41:01	1577	/console/rest/deployers/deploy	192.168.33.193	POST
2020-09-08 09:40:34	511	/console/rest/webconfig/virtualservers/put	192.168.33.193	POST
2020-09-08 09:40:10	60009	/console/dwr/call/plainpoll/ReverseAjax.dwr	192.168.33.193	POST
2020-09-08 09:39:09	60006	/console/dwr/call/plainpoll/ReverseAjax.dwr	192.168.33.193	POST
2020-09-08 09:38:08	60007	/console/dwr/call/plainpoll/ReverseAjax.dwr	192.168.33.193	POST
2020-09-08 09:37:18	547	/console/rest/webconfig/virtualservers/put	192.168.33.193	POST

图 7.3.6 查看访问日志

2) 在访问日志界面中，可以通过搜索日志查看不同虚拟主机的访问日志，如图 7.3.7

时间	访问处理时长 (ms)	访问链接	访问IP	请求方式
2020-09-08 10:04:44	1	/nonXaDsWeb/	192.168.33.193	GET
2020-09-08 10:04:44	1	/nonXaDsWeb/	192.168.33.193	GET
2020-09-08 10:04:44	1	/nonXaDsWeb/	192.168.33.193	GET
2020-09-08 10:04:44	2	/nonXaDsWeb/	192.168.33.193	GET

图 7.3.7 搜索访问日志

3) 在访问日志界面中，可以对访问日志信息进行过滤，分别可以通过时间、访问处理时长、自定义时间来过滤访问日志内容。

4) 在访问日志界面中，可以下载访问日志。点击下载日志按钮，勾选需要下载的日志，如图 7.3.8



图 7.3.8 下载访问日志

7.3.4 快照

快照是全面记录了 TongWeb7 服务器运行时的日志信息。记录的内容包括系统配置信息、监视量、jstack、jmap、访问日志、系统日志、GC 日志和 coredump。

7.3.4.1 生成快照和下载

1) 展开左侧导航栏点击“诊断”节点，双击“快照”，即进入快照管理界面，列表中显示已存在的快照信息；

2) 点击“生成快照”，弹出“生成快照”界面，见图 7.3.9。默认 GC 日志未开启，不能生成快照；如果生成的快照里要包含 GC 日志，需要根据不同的 jdk 设置 GC 日志信息；如 sunjdk, 需要配置 jvm 参数: -Xloggc:../logs/gc.log, 见 [8.2 参数配置](#)。



图 7.3.9 生成快照界面

3) 选择要生成的快照内容，点击创建，即生成快照文件，如图 7.3.10 所示：

时间	系统配置信息	系统日志	监视量	访问日志	jstack	jmap	coredump
2020-07-01 17:02:24	有	—	有	—	有	有	—
2020-07-01 16:53:53	—	有	—	有	—	—	—
2020-07-01 16:53:39	有	有	有	—	有	—	—

图 7.3.10 快照列表界面

4) 生成的快照文件在\${TW7_HOME}/snapshot/下，快照文件以生成快照的时间命名。目录下是快照的内容。具体介绍如下：

快照内容	目录名	快照内容说明
系统配置信息	conf	包含应用服务器 conf 目录下的所有配置文件。
访问日志	access-log	应用服务器各个虚拟机的访问日志以及轮转文件。
系统日志	system-log	应用服务器的系统日志以及轮转文件。
gc 日志	gc-log	记录服务器运行时 gc 信息的日志。
jmap	jmap	使用 jmap -dump:format=b, file=命令生成的服务器相关信息。
coredump	coredump	linux 环境下使用 gcore -o core 文件名 进程号 生成 coredump 文件。
jstack	jstack	使用 jstack -l 命令生成的服务器服务器相关信息。
服务器监视量	monitor	记录当前时间点的监视量值。

7.3.4.2 删除快照

在快照生成界面，选择要删除的快照，点击“删除”，即弹出快照删除确认界面；点击“确认”，则快照记录被永久性删除。

7.3.4.3 设置快照路径

快照默认路径在\${TW7_HOME}/snapshot 目录下，可以根据自己的需求，定制快照路径；在快照生成界面，点击“设置快照路径”，可以设置快照存放路径。

7.3.4.4 快照回放

- 1) 快照回放需要先打开数据持久化开关，见 [7.4.2 监视配置](#)；
- 2) 快照管理页面，存在已生成的快照时，点击快照后面的“回放”链接，即可查看快照回放信息；如下图 7.3.11 所示：



图 7.3.11 快照回放

- 3) 快照回放的内容包括：JVM 资源监视、web 监控通道、JDBC 资源监视。

7.3.4.5 快照下载

在快照生成界面，点击快照记录后面的“下载”，即可以下载快照，文件格式为 zip。

7.3.5 SNMP(简单网络协议)

简单网络管理协议 (SNMP) 是专门设计用于在 IP 网络管理网络节点 (服务器、工作站、路由器、交换机及 HUBS 等) 的一种标准协议，它是一种应用层协议。

TongWeb7 支持 SNMP 协议，开启协议后，可以通过 SNMP 工具查看 TongWeb7 的服务器状态信息。

7.3.5.1 SNMP 代理服务配置

- 1) 展开左侧导航栏点击“诊断”节点，双击“snmp”，即展开 SNMP 代理服务配置界面，如下图 7.3.12 所示：

启用	<input type="checkbox"/>	指定是否启用此 snmp 代理
地址	<input type="text" value="0.0.0.0"/>	此 snmp 服务绑定的 ip 地址，默认为 0.0.0.0
端口	<input type="text" value="161"/>	让 snmp 代理监听来自 snmp 管理器的请求时使用的端口
传输类型	<input type="text" value="udp"/>	
版本	<input type="text" value="v3"/>	snmp 协议版本，支持 v2 和 v3
用户名	<input type="text" value="public"/>	v3 版本的 usm 用户名
验证密钥	<input type="text" value="nmsAuthKey"/>	确保只有授权用户才能请求或接收使用的密钥，长度最长 16 位
隐私密钥	<input type="text" value="myDesPrivateKey"/>	对消息进行加密和解密时使用的密钥，长度为 16 位
引擎 id	<input type="text" value="62:a0:c1:81:11:c3:17:33"/>	标明当前 snmp 代理不同于其他的唯一标识符

图 7.3.12 snmp 代理配置界面

属性说明：

- 启动：是否启动 SNMP 代理服务器，默认为 false,未启动；
- 地址：SNMP 代理服务器监控的 ip 的地址，默认为 0.0.0.0，即监听该服务器上所有 ip 地址；
- 端口：SNMP 代理服务器监控的端口，默认为 161；
- 传输类型：SNMP 代理服务器与网络管理系统之间所使用的传输协议，可选择 udp 和 tcp 协议，默认为 udp 协议；
- 版本：SNMP 协议的版本，目前支持 v2 和 v3 版本。v3 是 v2 的升级版，选择 v3 版本需要填写下方的用户名，验证密钥，隐私密钥和引擎 id；v2 版本不需要填写。默认版本：v3；
- 用户名：必填，v3 版本的 usm 用户名，默认为 public；
- 验证密钥：确保只有授权才能请求和使用的的密钥，加密方式仅支持 MD5，默认值为 nmsAuthKey；
- 隐私密码：对消息进行加密和解密时使用的密钥，加密方式仅支持 DES，加密 key 不能少于 16 个字符，默认值为 myDesPrivateKey；
- 引擎 id：标明当前 SNMP 代理唯一标识符，长度必须大于 5 位，小于 32 位，默认值为: 62a0c18111c31733。

2) OID 值, TongWeb7 目前支持查询的监控类型如下:

OID	Object Name	解释
1.3.6.1.4.1.55566.1.1.1	java.lang:type=OperatingSystem	java 操作系统信息
1.3.6.1.4.1.55566.1.1.2	java.lang:type=Runtime	java 运行时信息
1.3.6.1.4.1.55566.1.1.3	java.lang:type=ClassLoading	java 类加载信息
1.3.6.1.4.1.55566.1.1.4	java.lang:type=Compilation	java 编译信息
1.3.6.1.4.1.55566.1.1.5	java.lang:type=Memory	java 内存信息
1.3.6.1.4.1.55566.1.1.6	java.lang:type=Threading	java 线程信息
1.3.6.1.4.1.55566.1.1.7	java.lang:type=GarbageCollector,name=*	java 垃圾回收信息
1.3.6.1.4.1.55566.1.2.1	config:parent=/Tongweb/Server,name=JDBCConnectionPool*	tongwebjdbc 连接池信息
1.3.6.1.4.1.55566.1.2.2	config:name=WebContainer,parent=/Tongweb/Server	tongwebweb 容器信息
1.3.6.1.4.1.55566.1.3.1	TONGWEB:type=ThreadPool,name=*	tongweb 线程池信息
1.3.6.1.4.1.55566.1.3.2	TONGWEB:j2eeType=WebModule,name=//*/*,*	tongweb 应用信息
1.3.6.1.4.1.55566.1.3.3	TONGWEB:type=Manager,host=*,context=/*	tongweb 管理信息
1.3.6.1.4.1.55566.1.3.4	TONGWEB:type=Server	tongweb 服务信息

3) 部分特殊返回值释义如下:

- Error:通过 Jconsole 查询时，无法获取或报错的值；
- UNSUPPORT: MBean 属于 CompositeData TabularData 这两类的返回值未做支持；
- NULL:当想要值没有返回内容时，例：JDBC 数据源未做配置；

7.3.5.2 SNMP 代理服务示例

- 1) 下载 Snmpwalk, 下载地址: <https://ezfive.com/snmpsoft-tools/>;
- 2) 使用 SNMPWALK 查看 TongWeb7 服务器信息:

- V1 格式: 查看从 OID 为“1.3.6.1.4.1.55566.1.1.1”开始的所有 Object 监控值。

```
Snmwalk.exe -r:127.0.0.1 -os: 1.3.6.1.4.1.55566.1.3.4(OID 值)
```

返回结果如下: 为 TONGWEB:type=Server 的所有属性值。

```
D:\Program Files\SnmpWalk>SnmpWalk.exe -r:168.1.13.71 -os:1.3.6.1.4.1.55566.1.3.4
4
SnmpWalk v1.01 - Copyright (C) 2009 SnmpSoft Company
[ More useful network tools on http://www.snmpsoft.com ]

OID=.1.3.6.1.4.1.55566.1.3.4.1, Type=OctetString, Value=localhost
OID=.1.3.6.1.4.1.55566.1.3.4.2, Type=OctetString, Value=StandardServer[8005]
OID=.1.3.6.1.4.1.55566.1.3.4.3, Type=OctetString, Value=com.tongweb.catalina.com
e.StandardServer
OID=.1.3.6.1.4.1.55566.1.3.4.4, Type=Integer, Value=8005
OID=.1.3.6.1.4.1.55566.1.3.4.5, Type=OctetString, Value=2020-07-01 13:01:27
OID=.1.3.6.1.4.1.55566.1.3.4.6, Type=OctetString, Value=TongWeb
OID=.1.3.6.1.4.1.55566.1.3.4.7, Type=OctetString, Value=7.0.2.5
OID=.1.3.6.1.4.1.55566.1.3.4.8, Type=OctetString, Value=TONGWEB:type=Service
OID=.1.3.6.1.4.1.55566.1.3.4.9, Type=OctetString, Value=TW7-SHUTDOWN
OID=.1.3.6.1.4.1.55566.1.3.4.10, Type=OctetString, Value=STARTED
Total: 10
```

- V2 格式:

```
Snmwalk.exe -r:127.0.0.1 -os:.1.3.6.1.4.1.55566.1.1.1 -v:2c
```

```
Snmwalk.exe -r:127.0.0.1 -os:.1.3.6.1.4.1.55566.1.1.1 -v:2c -
c:public
```

- V3 格式:

```
Snmwalk.exe -r:127.0.0.1 -os:.1.3.6.1.4.1.55566.1.1.1 -sn:public
-v:3 -ap:MD5 -aw:nmsAuthKey -pp:DES -pw:myDesPrivateKey -
ei:62a0c18111c31733
```

- 查看某个 OID

```
Snmwalk.exe -r:127.0.0.1 -os:.1.3.6.1.4.1.55566.1.1.1 -
op:1.3.6.1.4.1.55566.1.1.2 (注: -os 为起始 OID, -op 为终止 OID 位置)
```

- 注意: snmpwalk 仅支持 UDP 协议。

- 3) TongWeb7 提供了支持 TCP 协议的简单调用示例代码 snmpCli.rar, 存放于
\${TW7_HOME}/Samples/snmp 目录中。

7.4 监视

7.4.1 监视概述

监视服务用于显示 TongWeb 服务器在运行时的内存、线程、数据源、协议处理等方面的资源占用情况, 对某些数据进行图表化的展示, 以显示其历史变化轨迹等, 监视服务的意义在于帮助维护人员实时准确地了解系统运行情况, 并能据此对具体的参数进行相应的调节, 以使得整个系统更加稳定和流畅地运行。

7.4.2 监视配置

监视配置可以定制具体监视目标的细节, 包括监视功能的总开关、监视收集到的数据在内存中的存活时间、超出存活时间后的数据是否需要持久化以及存活时间检测周期等, 监视配置中所有的配置项均是保存后即时生效, 无需重启服务器。

1. 展开管理控制台左侧导航树中的“监视”节点;

2. 点击“监视配置”节点，出现如图所示的界面：



图 7.4.1 监视服务配置页面

监视服务配置属性说明如下，

监视功能开关：TongWeb7 监视服务的总开关，只有当该开关打开后监视服务和持久化服务才可生效，勾选后即可打开；

数据存活时间(秒)：该配置表示“监视明细”中可即时追溯到的监视数据保存时长。

存活检测周期(秒)：检测监视数据是否需要进行持久化的时间周期。

当超过“数据存活时间(秒)” + “存活检测周期(秒)”配置时间的监视数据将会被持久化到硬盘文件，持久化的监视数据将不能在“监视明细”中即时查看，需要通过“监视回放”功能进行回放式查看。

监控目标	监视功能开关	数据持久化开关	采集周期(秒)	指示器
JVM 内存	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 内存池	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 垃圾收集器	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 线程	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 编译器信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 类加载信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JVM 运行时信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
操作系统	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
TongWeb 信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
通道信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
XA 数据源信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
数据源信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
事务信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
JCA	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
应用细节信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
应用会话信息	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
应用类加载器	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	
应用资源缓存	<input type="checkbox"/>	<input type="checkbox"/>	十秒钟	

图 7.4.2 监视模块配置页面

监视模块配置，可对具体的功能模块进行定制化的监视配置，在页面上可以看到可配置的监视模块包括：JVM 内存、JVM 内存池、JVM 垃圾收集器、JVM 线程、JVM 编译器信息、JVM 运行时信息、JVM 类加载信息、操作系统、TongWeb 信息、通道信

息、数据源信息、JCA、应用细节信息、应用会话信息、应用类加载器、应用资源缓存、事务信息等。

监视模块配置属性说明如下，

监视功能开关：对应模块的监视开关，选中为开启状态，开启后可在“监视明细”页面查看该模块实时的监视数据，只有当模块的监视功能打开后，模块的数据持久化开关才能生效。

数据持久化开关：对应模块的数据持久化的开关，选中为开启状态，打开后超过“数据存活时间(秒)”配置的时间后的数据将会被持久化到硬盘文件，否则这些数据将会被丢弃。

轮转大小：打开数据持久化开关需要同时配置轮转大小属性，默认为 10MB，日志文件达到所配置大小后进行轮转。

采集周期(秒)：对应模块的监视数据采集周期，周期越短采集的数据精确度越高，同时也意味着消耗更多的系统资源。

指示器：用于直观地显示“采集周期(秒)”的时间长短，便于对比各个模块的“采集周期(秒)”的时间长短。

7.4.3 监视明细

监视明细页面列出了可以监控的具体模块，点击具体模块后，可查看该模块详细的监视数据等，监视配置中各个模块监视功能默认是关闭状态，需要打开对应功能的监视开关，具体参考 7.4.2 章节；

1. 展开管理控制台左侧导航树中的“监视”节点；
2. 点击“监视明细”节点，出现如图所示的界面：

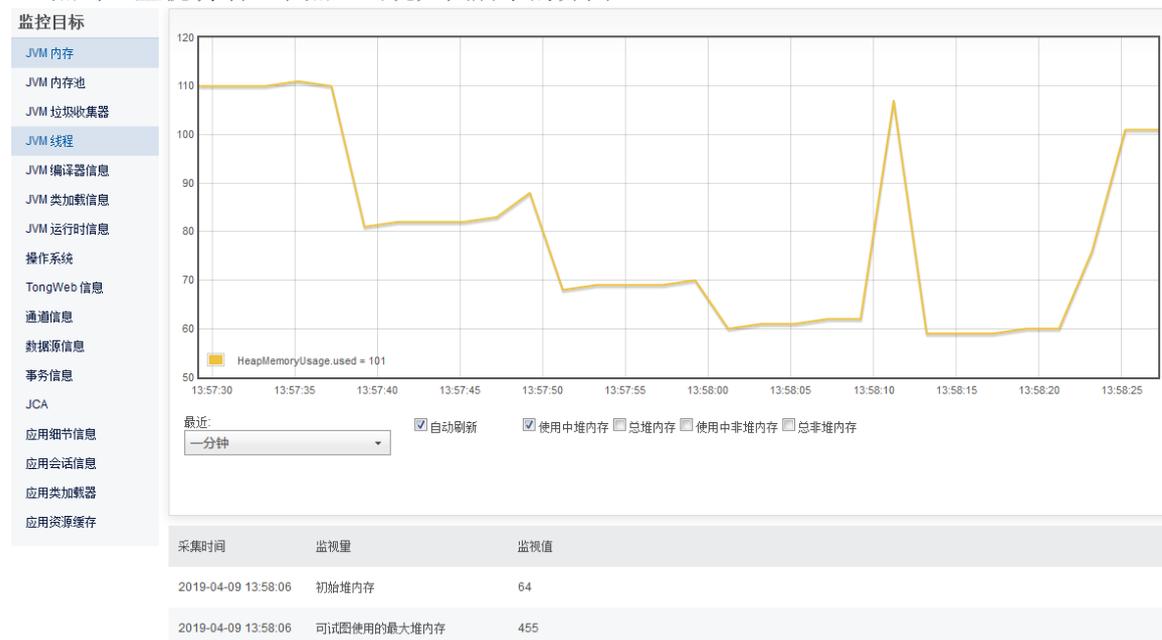


图 7.4.3 监视明细页面

该页面的左侧“监控目标”导航，为各个模块监视数据展示的切换链接，点击具体的监控目标，右侧页面可显示具体模块的监视信息，以“操作系统”监控目标为例说明页面功能，如下图所示，是点击“监控目标”导航中的“操作系统”后显示的操作系统的模块监视数据：



图 7.4.4 操作系统模块监视页面

具体模块监视页面的结构分为上下两部分，其中上半部分显示了可进行图表化的监视数据，可通过下拉列表选择数据显示的范围，如一分钟、两分钟、五分钟等，可通过勾选具体的数据项以在图表上进行绘图展示，页面默认勾选的是第一个数据项，即“CPU 使用率”，同时勾选“内存使用率”数据项后，cpu 和内存的监视数据将会同时绘制到图表上，图表上各个数据项的绘图颜色会自动进行变化以便于区分，模块监视页面的下半部分展示的是该模块的配置或当前状态信息，这部分数据仅显示最后一次收集的数据，不记录历史，同时也不会进行持久化，可用于检测内存对象的实时状态。

7.4.4 模块监视属性

监视明细页面列出了可以监控的具体模块，点击具体模块后，可查看该模块详细的监视数据等，不同的监视模块显示的监视属性不同，下面对所有的监视属性进行说明。

7.4.4.1 JVM 内存

- 总堆内存：当前虚拟机所占用的总堆内存的大小；
- 使用中堆内存：当前虚拟机正在使用的堆内存的大小，虚拟机从操作系统申请到的“总堆内存”，并不是立即使用完毕的，其中一部分是“使用中堆内存”，另一部分是空闲的，当空闲的堆内存过小时，虚拟机会向操作系统进行申请更多的内存，反之，当空闲的堆内存过大时，虚拟机会释放空闲的内存归还给操作系统；
- 总非堆内存：当前虚拟机所占用的总非堆内存的大小；
- 使用中非堆内存：当前虚拟机正在使用的非堆内存的大小，虚拟机从操作系统申请到的“总非堆内存”，并不是立即使用完毕的，其中一部分是“使用中非堆内存”，另一部分是空闲的，当空闲的非堆内存过小时，虚拟机会向操作系统进行申请更多的内存，反之，当空闲的非堆内存过大时，虚拟机会释放空闲的内存归还给操作系统；
- 初始堆内存：初始堆内存大小，可通过参数 `-Xms` 进行配置；
- 可试图使用的最大堆内存：可试图使用的最大堆内存大小，可通过参数 `-Xmx` 进行配置；
- 初始非堆内存：初始非堆内存大小，可通过参数 `-XX:PerSize` 进行配置；
- 可试图使用的最大非堆内存：可试图使用的最大非堆内存大小，可通过参数 `-XX:MaxPermSize` 进行配置；
- 正在回收的对象数量：正在回收的对象数量；
- 是否有内存变动日志信息：当内存由于 GC 发生变化时是否会进行日志输出。

7.4.4.2 JVM 内存池

虚拟机的内存分为堆和非堆两种类型，其中“HEAP”表示类型为堆内存，

“NON_HEAP”表示为非堆内存，堆内存和非堆内存都采用池化的管理方式，堆内存的内存池包括：老年代、幸存区、伊甸园区，非堆内存的内存池包括：代码缓存区、永久代等，虚拟机的每种内存池都可以通过具体的启动参数进行设置，通过该模块监视页面可以查看各个内存池的配置和实时监视信息。

7.4.4.3 JVM 垃圾收集器

该模块监视页面可展示虚拟机中生效的所有的垃圾收集器，通过“切换至”下拉菜单可进行监视数据的切换，每种垃圾收集器可展示的属性包括：垃圾收集器名称、是否有效、收集次数、累积耗时等。

7.4.4.4 JVM 线程

该模块监视页面可展示虚拟机中的线程使用统计信息，包括：“当前所有线程数量”，“当前守护线程数量”，“活跃线程数峰值”，“所有运行过的线程数量”、“死锁的线程”、“当前所有线程消耗的 CPU 时间”、“当前所有线程在用户模式消耗的 CPU 时间”、“当前所有线程状态数”、“最消耗 CPU 时间的线程”，“当前所有线程状态数”可监视处于不同状态（如 RUNNABLE、WAITING、TIMED_WAITING）下的线程的数量，“最消耗 CPU 时间的线程”可监视最消耗 CPU 时间的前 15 个线程。

7.4.4.5 JVM 类加载信息

该模块监视页面可展示当前虚拟机的类加载相关信息，如“当前已加载的类数量”、“所有卸载过的类数量”、“所有加载过的类数量”、“是否有类加载日志信息”。

7.4.4.6 JVM 运行时

该监视模块页面可展示当前虚拟机的运行时信息，包括：“jvm 规范名称”、“jvm 规范版本”、“jvm 规范供应商”、“jvm 实现名称”、“jvm 实现版本”、“jvm 实现供应商”、“jvm 启动时间”、“jvm 运行时长”、“库文件路径”、“引导类路径”、“系统类路径”、“是否支持从引导类路径搜索类文件”、“jvm 启动入参”、“系统属性列表”。

7.4.4.7 操作系统

该监视页面可展示操作系统的相关信息，包括：“CPU 使用率”、“内存使用率”、“系统架构

”、“系统名称”、“系统版本”、“处理器个数”。

7.4.4.8 TongWeb 信息

该模块监视页面展示了当前正在运行的的 TongWeb 信息，包括：“服务器名称”、“服务器版本”、“服务器构建日期”、“停止服务监听地址”、“停止服务监听端口”。

7.4.4.9 通道信息

该模块配置页面显示 TongWeb 中可用的通道的信息，具体监视信息包括：“当前连接数”、“当前线程池线程数”、“正在执行任务的线程数”、“线程池使用率”、“发送的字节数”、“接收的字节数”、“处理时间”、“错误数”、“最大处理时间(毫秒)”、“请求数”，其中“发送的字节数”、“接收的字节数”、“处理时间”、“错误数”、“最大处理时间(毫秒)”、“请求数”都是该通道处理过程的累计的监视值，其中：线程池使用率=正在执行任务的线程数/线程池最大值。

7.4.4.10 数据源信息

“数据源信息”的模块监视页面可显示当前 TongWeb 服务器中可用的数据源信息，包括：“当前正在使用的连接数”、“当前空闲的连接数”、“当前池中连接总数”、“等待连接的线程数”、“连接池名称”、“驱动类名”、“连接串”、“连接池初始连接数”、“连接池最大连接数”、“验证连接使用的 SQL”、“等待连接空闲出来的最大毫秒数”、“最大空闲数”、“释放连接时保留的最小空闲连接数”、“归还连接时提交未决事务”、“归还连接时回滚未决事务”、“清理线程周期”等诸多数据源相关的属性信息。

7.4.4.11 JVM 编译器信息

该模块监视页面可展示当前虚拟机的编译器相关信息，如“是否支持监视编译耗

时”、“编译耗时近似值”以及“JIT(Just-in-time)编译器名称”。

7.4.4.12 事务信息

该模块监视页面可显示，当前服务器中分布式事务的状态信息，包括有：“当前事务数”、“所有提交的事务数”、“所有回滚的事务数”等。

7.4.4.13 JCA

JCA 模块监视页面可以查看 JCA 相关的监视信息，如连接池等信息。

7.4.4.14 应用细节信息

该监视模块显示了应用相关的监视信息，包括有：“应用名称”、“未编码应用路径”、“URL 编码应用路径”、“应用目录”、“应用工作目录（jsp 编译缓存）”、“父优先加载”、“应用请求数”、“应用请求错误数”、“应用请求处理时间”、“应用请求最大处理时间(毫秒)”、“应用请求最小处理时间(毫秒)”、“会话超时时间”、“应用启动时间”、“应用启动耗时”、“基于 Cookie 存取会话 ID”、“忽略注解”。

7.4.4.15 应用会话信息

该监视模块显示了应用会话相关的监视信息，如“当前活跃会话数”、“过期的会话数”、“活跃会话数峰值”、“限制的最大活跃会话数”、“拒绝的会话数”、“会话保存文件”、“创建过的所有会话数”、“最大会话存活时长(秒)”、“失效会话平均存活时长(秒)”。

7.4.4.16 应用类加载器

该模块监视页面显示了应用类加载器相关的配置信息，包括：“类加载器全名称”、是否“父优先加载”、“类加载路径”、是否“可重加载”。

7.4.4.17 应用资源缓存

该模块监视页面显示了应用资源缓存相关信息，包括：“当前缓存数”、“最大缓存数”、“命中缓存”、“查找次数”、“缓存条目存活时长(毫秒)”，缓存可提供静态资源的访问速度。

7.4.5 监视概览

此页集中显示了主要的监视量信息，包括内存、CPU、通道、数据源等，通过该信息可以了解 TongWeb 的整体运行状态，直接点击管理控制台左侧导航树中的“监视概览”节点，或者按如下步骤打开监视概览页面，

1. 展开管理控制台左侧导航树中的“监视”节点；
2. 点击“监视概览”节点，出现如图所示的界面：

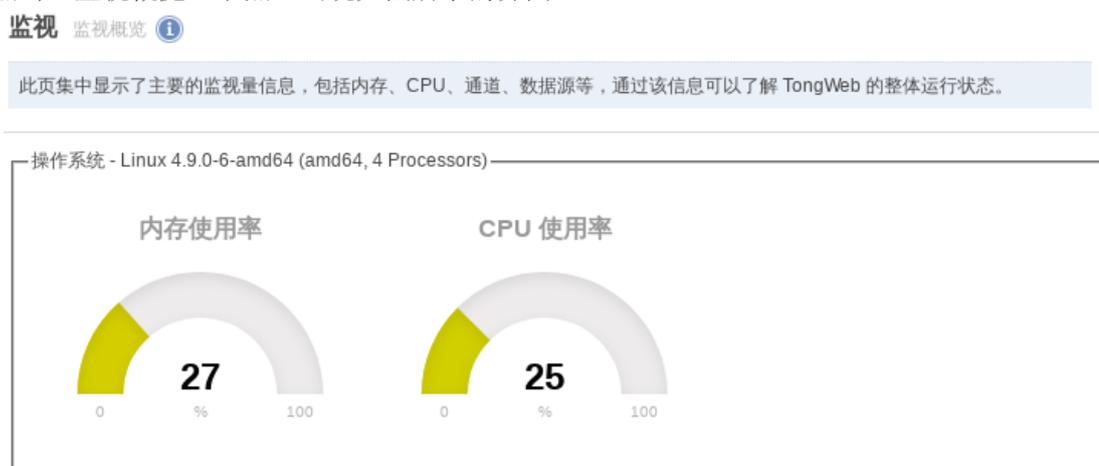


图 7.4.5 监视概览中操作系统信息

监视概览中操作系统信息显示了实时的内存和 cpu 的使用率。

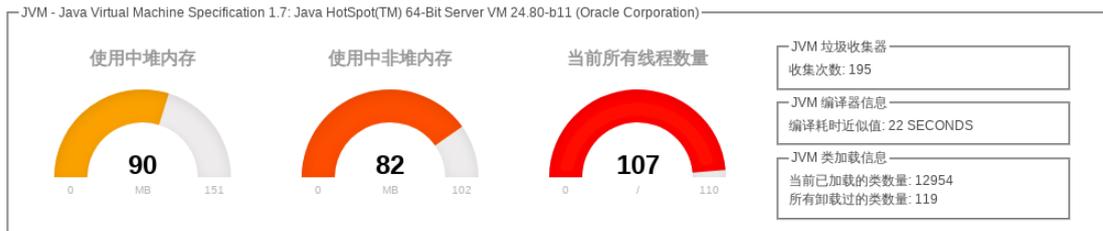


图 7.4.6 监视概览中 JVM 信息

监视概览中 JVM 信息中，“使用中堆内存”显示了当前虚拟机使用中堆内存与总的堆内存的比例关系，数字显示的是以 MB 为单位的内存大小，“使用中非堆内存”显示了当前虚拟机使用中非堆内存与总的非堆内存的比例关系，数字显示的是以 MB 为单位的内存大小，“当前所有线程数量”显示了当前所有线程数量与历史最多的线程数峰值的比例。

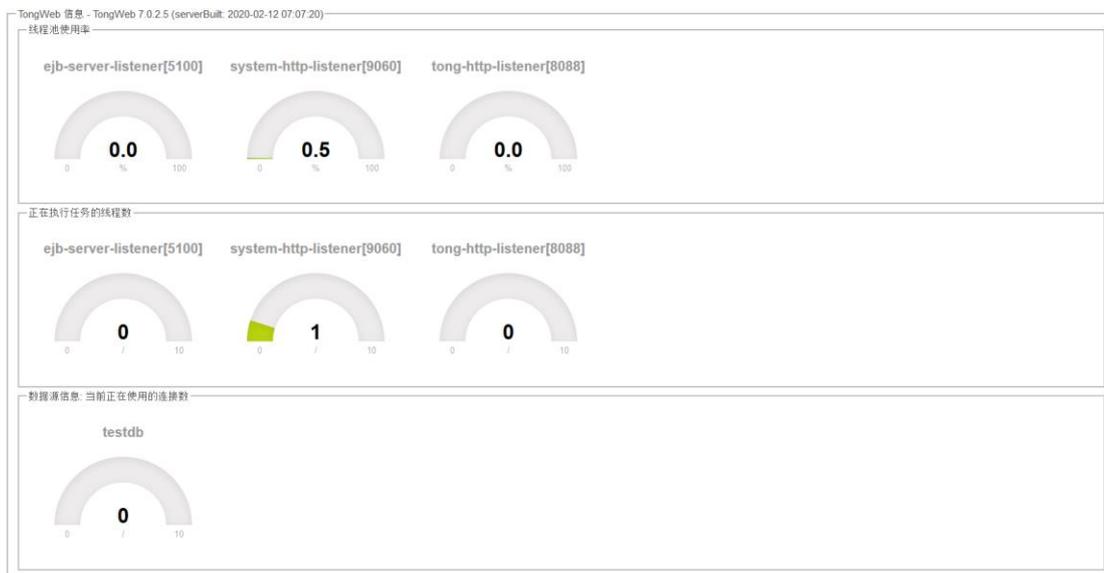


图 7.4.7 监视概览中 TongWeb 通道和数据源信息

监视概览中 TongWeb 通道和数据源信息，其中“正在执行任务的线程数”显示的是正在执行任务的线程数与当前通道关联的线程池中的总线程数的比例，“数据源信息: 当前正在使用的连接数”显示的是当前数据源中正在使用的连接数与总连接数的比例。

7.4.6 监视回放

监视回放页面用于显示过去一段时间的系统 and 应用的监视信息，

1. 展开管理控制台左侧导航树中的“监视”节点；
2. 点击“监视回放”节点，出现如图所示的界面：

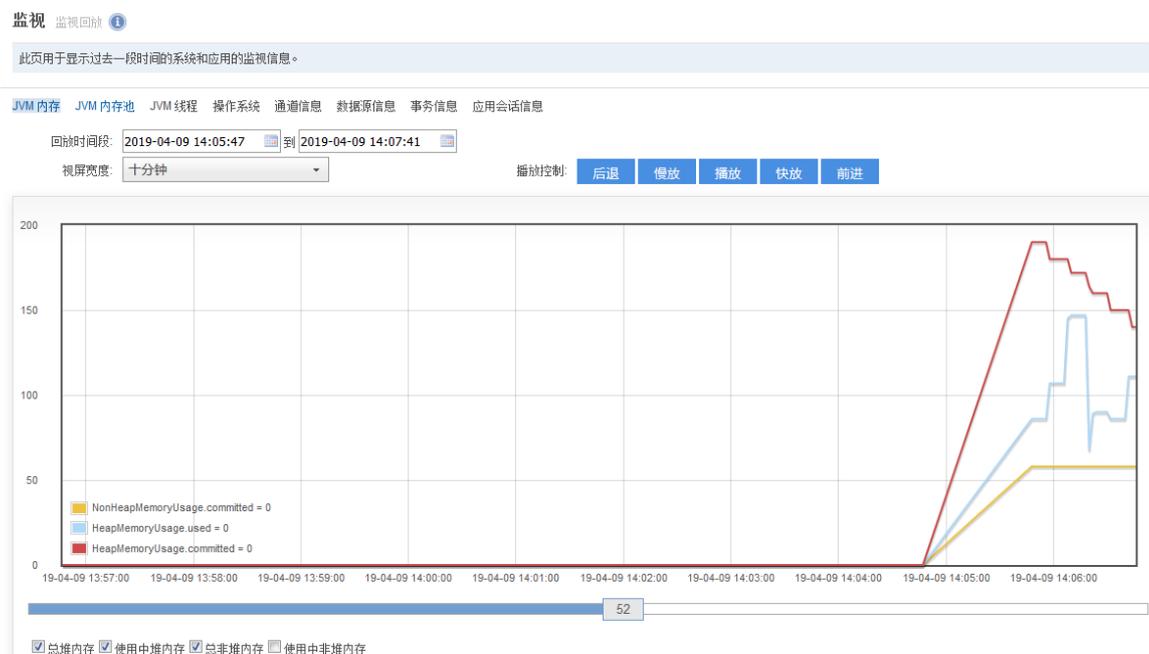


图 7.4.8 监视回放页面

只有支持持久化并且存在持久化文件的模块才能进行持久化回放，所有支持持久化的模块包括：**JVM 内存**、**JVM 内存池**、**JVM 线程**、**操作系统**、**通道信息**、**数据源信息**、**事务信息**、**应用会话信息**，当存在持久化文件时，可点击模块链接进行持久化监视数据回放，回放属性说明，

- **回放时间段**：单次回放的持续时间段；
- **视屏宽度**：页面上最大显示的时间宽度，每次刷新当前宽度的十分之一的数据，视屏宽度越大也就意味着回放的速度越快；
- **播放控制**：可对回放过程进行控制，包括：前进、后退、快放、慢放、播放、暂停。

“监视回放”的显示形式和“监视明细”类似，通过勾选底部不同的数据项，可在图表上进行同时显示不同的监视数据项，以便于对比数据的变化和走势。

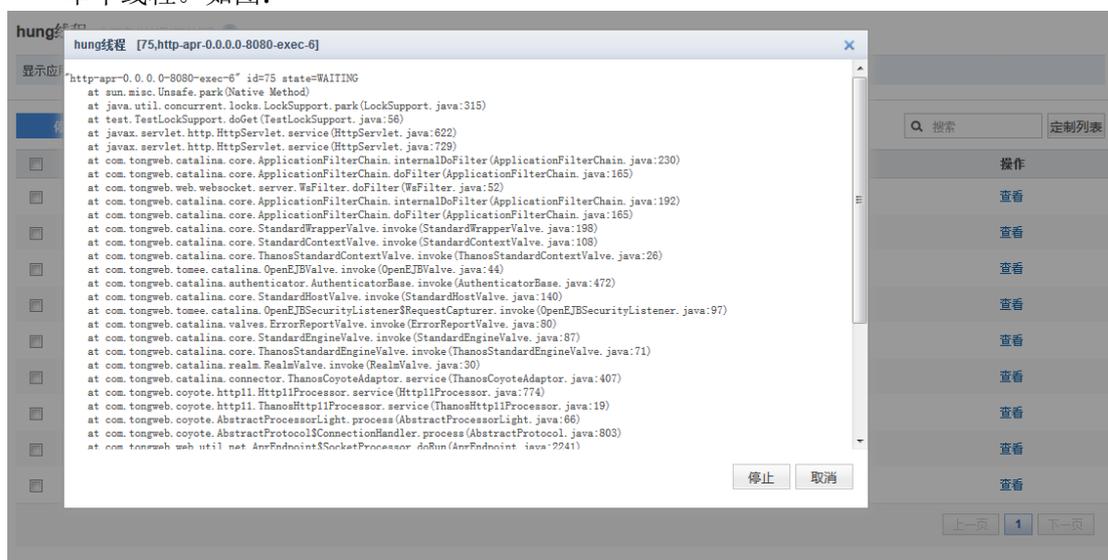
7.4.7 hung 线程

hung 线程页面用于显示处理 web 应用请求超时的线程，超时时间和 [Web 容器配置](#) 中超时线程阈值一致，不配置默认是 60 秒。

1. 展开管理控制台左侧导航树中的“监视”节点；
2. 点击“hung 线程”节点，出现如图所示的界面：



- 勾选所选线程，点击“停止按钮”，弹出风险提示对话框，点击“停止”按钮将停止所选线程。
- 点击每行“查看”按钮，弹出当前线程的调用栈，并有“停止”按钮，停止所选中的单个线程。如图：



● 风险提示

停止线程存在风险，因此停止线程给出风险提示，用户可根据自身情况评估是否存在真正的风险有选择的进行停止。由于停止线程功能仅仅只能依赖于 JDK 提供的实现，因此停止线程存在以下风险：

- 停止线程时线程运行在任何地方即进行停止，无法释放所操作的资源，例如数据库连接等。
- 停止线程时，会释放线程持有的所有锁，因为锁的不恰当释放，会造成多个线程操作的共享数据产生不一致性，这将对数据有事务完整性一致性有要求的业务逻辑产生致命的危险。

由于以上原因，停止线程操作有很大的风险存在，需慎重操作。

7.4.8 阈值配置

系统出现故障或者性能瓶颈时，一些指标会出现异常。预警策略就是配置当服务器

达到某些条件时，自动生成快照，并且可以配置生成的快照包含哪些内容。

在左侧导航栏点击打开“监视-阈值配置”，如下图所示：



- 策略之间的关系：策略条件中达到任意一条就生成快照，还是满足所有条件才生成快照。

- 包含匹配下列：即策略条件，可配置 CPU、内存、GC、http 通道、连接池状态达到某种条件作为一条策略。

- 快照内容：生成的快照包含哪些内容。可选的有系统配置信息、访问日志、系统日志、gc 日志、jmap、jstack、服务器监视量、coredump。

快照的预警策略是在服务器运行中可能出现异常和性能低下的时候触发。具体介绍如下：

- CPU：服务器运行阶段，当 CPU 使用率居高不下时，必然会影响效率，关于 CPU 的预警策略可配置为当 CPU 使用率连续多次监控都超过一定比例时，就自动生成快照。可结合快照中的日志、jstack 信息分析是哪些操作导致 CPU 使用率达到这么高，从而影响性能。

- 内存：服务器运行阶段，当 jvm 的堆内存使用接近最大堆内存时，就很可能出现内存溢出，导致整个 jvm 宕掉。关于内存的预警策略可配置当 jvm 的堆内存使用率连续多次都超过一定比例时，就自动生成快照。对快照中的 gc 日志、jmap 信息进行分析，看哪些对象占用了过多内存，这些对象是否是内存泄露导致。

- GC：服务器运行阶段，当一次 gc 的时间比较长的时候，会影响整个服务器的运行。关于 Gc 的预警策略可配置为当一次 gc 的执行时间达到多少的时候，就自动生成快照。通过快照中的 gc 日志和 jstack 信息分析是否是执行 gc 影响到服务器的运行，是什么原因导致 gc 执行时间过长。

- 通道：当处理 http（或 a.jp）请求的的线程非常多的时候，可能是应用存在瓶颈导致。关于 http/a.jp 的预警策略可配置当某个 http/a.jp 通道的处理线程数超过多少时，将会自动生成快照。根据快照中 jstack 的内容，可以分析应用的性能瓶颈在哪里。

- 连接池：当连接池不可用时，可能会造成性能瓶颈甚至是应用异常。关于连接池的预警策略可以配置当哪几个连接池不可用时，自动生成快照。可结合快照中的日志和配置等信息分析为何出现连接池不可用的情况。

当 TongWeb7 一直满足预警策略中的条件时，为避免不断的生成快照影响服务器性能，规定自动生成快照在一小时内只能生成一定数量的快照，然后即使满足预警策略的条件，也不再生成新的快照。自动生成快照一小时内可生成的快照数，可在启动脚本中通过-Dtongweb.snapshotinhour 配置，默认值是 5。

7.5 日志服务

7.5.1 日志服务概述

TongWeb 的日志服务可以向命令行窗口以及日志文件输出记录相关的系统日志。系统日志记录了 TongWeb 服务器的运行状态，通过分析系统日志的错误信息，可帮助查找系统出错原因。

TongWeb 的日志服务支持细粒度的控制每个模块的日志级别，并提供了多种轮转方式以便用户依据自己的应用场景及需求选择合适的轮转方式。

7.5.2 模块日志级别配置

1. 展开管理控制台左侧导航树中的“日志服务”节点；
2. 点击“模块日志级别配置”节点，出现如图 7.5.1 所示。

Admin:	INFO
Bean Validation:	INFO
CDI:	INFO
Configuration:	INFO
Core:	INFO
Data Source:	INFO
Deployment:	INFO
Ejb Container:	INFO
JPA:	INFO
JCA:	INFO
jms:	INFO
jsf-api:	INFO
jsf-impl:	INFO

图 7.5.1 各模块日志级别配置

在相应模块下选择对应的日志级别，点击保存，则该模块日志级别配置完成。

3. 日志级别说明

TongWeb7 中使用的 Logger 分为两种不同类型。一种是 JDK 自带的 `java.util.logging.Logger`；另外一种是 TongWeb 封装的 `log4j`、`log4j2`、`slf4j` 等日志框架的 `com.tongweb.tongweb.util.Logger`。两个 Logger 类都对日志的输出级别进行了详细的划分，其对比图如下：

log4j、log4j2、slf4j等		
日志级别	数值	描述
OFF	2147483647	关闭—设置为该级别，则关闭所有日志输出，不输出任何内容
FATAL	50000	致命错误—严重的错误事件，将会导致应用程序的退出
ERROR	40000	错误—发生错误事件，但仍然不影响系统的继续运行
WARN	30000	警告—存在隐患，可能导致程序发生错误
INFO	20000	一般信息—提示程序运行状态，当前系统信息等
DEBUG	10000	调试—提示某个参数的当前状态值、程序运行step by step的进展、程序细节的详细内容
TRACE	5000	追踪—非常详尽的记录程序执行信息，当前执行位置，状态参数等
ALL	-2147483648	所有一设置为该级别，则打开所有日志输出，输出所有内容

图 7.5.2 log4j 等日志框架的日志级别

其中，log4j 建议只使用四个级别，分别是 **ERROR**、**WARN**、**INFO**、**DEBUG**。这 4 个级别基本可以覆盖业务中所有的日志输出场景。而 JDK 自带的 `java.util.logging.Logger` 类的日志级别划分如下：

java.util.logging.Logger		
日志级别	数值	描述
OFF	2147483647	关闭—设置为该级别，则关闭所有日志输出，不输出任何内容
SEVERE	1000	严重信息
WARNING	900	警告信息
INFO	800	一般信息
CONFIG	700	配置信息
FINE	500	细微信息
FINER	400	更细微的信息
FINEST	300	最细微的信息
ALL	-2147483648	所有一设置为该级别，则打开所有日志输出，输出所有内容

图 7.5.3 java.util.logging.Logger 日志级别

可以看到，两个 `Logger` 类对日志的级别划分都是通过 `Level` 对应不同的数值来实现的，通过设置日志输出级别的数值，大于该数值的日志才能被输出，达到日志分级过滤的目的。

因此，哪怕两个 `Logger` 对级别划分的细粒度不同，我们也可以结合业务经验，对两者间的关系进行一个对比：

java.util.logging.Logger	log4j、log4j2、slf4j等	使用场景
OFF	OFF	关闭所有日志输出
	FATAL	不建议使用
SEVERE	ERROR	输出严重错误信息—会对系统运行状态产生影响
WARNING	WARN	输出警告信息—一些不严重的异常情况
INFO	INFO	正常输出—正常输出提示信息
CONFIG	DEBUG	调试模式—输出便于调试程序运行的信息
FINE	TRACE	不建议使用
FINER		不建议使用
FINEST		不建议使用
ALL	ALL	打印所有日志输出

图 7.5.4 日志级别

当我们的代码出现问题时，只需要将对应的模块日志级别设置为 WARNING/WARN，就可以快速过滤日志信息，获取异常日志进行问题分析。

7.5.3 系统日志配置

1. 展开管理控制台左侧导航树中的“日志服务”节点；
2. 点击“系统日志配置”节点，出现如图 7.5.5 所示。



图 7.5.5 系统日志配置

可在该页面配置系统日志的相关属性：

- 命令行输出：默认开启，用于配置是否在命令行窗口输出系统日志信息。
- 系统日志轮转：
 - 按大小轮转（默认选项）：该轮转模式下需要同时配置轮转大小属性，默认为 50MB，超过 100MB 以 100MB 为准，日志文件达到配置大小后进行轮转；
 - 按周期轮转：该轮转模式下需要同时配置轮转周期属性，从当前时间起，每隔相应时间（超过 10 天，以 10 天为准）进行系统日志的轮转；
 - 按天轮转：当天（从 0 点到 24 点）的系统日志将被记录在同一个日志文件中。新一天开始（即系统时间 0:00）时，系统日志进行轮转，效果不同于按周期轮转生成的系统日志。
 - 不轮转：不进行日志轮转。
- 日志数量：默认为 20 个，超出该数量后，则会自动删除较早的日志文件。
- 系统日志目录：可以在控制台上设置系统日志 server.log 的存放路径，若输入的是一个相对路径，则会从 TongWeb 安装目录下开始创建。

7.5.4 压缩日志配置

1. 展开管理控制台左侧导航树中的“日志服务”节点；

2. 点击“压缩日志配置”节点，出现如图 7.5.6 所示。

系统日志	<input type="checkbox"/>		
系统日志压缩目录	<input type="text" value="logs"/>	系统日志压缩包存放目录，默认为系统日志的目录	
轮转时间	<input type="text" value="21:00"/>	设置压缩任务的轮转时间	
访问日志	<input type="checkbox"/>		
访问日志压缩目录	<input type="text" value="logs/access"/>	访问日志压缩包存放目录，默认为访问日志的目录	
轮转时间	<input type="text" value="22:00"/>	设置压缩任务的轮转时间	
持久化日志	<input type="checkbox"/>		
持久化日志压缩目录	<input type="text" value="persistence"/>	持久化日志压缩包存放目录，默认为持久化日志的目录	
轮转时间	<input type="text" value="23:00"/>	设置压缩任务的轮转时间	

图 7.5.6 压缩日志配置

可在该页面配置压缩日志的相关属性：

- 压缩功能开关：勾选表示开启对应日志类型的压缩功能。
- 日志压缩目录：压缩日志生成的压缩包存放路径，支持配置绝对路径和相对路径，相对路径从 TongwWeb 的根目录开始，默认与压缩的日志文件存放路径一致。
- 轮转时间：生成压缩文件的时间点。每天定时执行压缩任务。如果当日设置的时间点已经过了，则次日开始执行。TongwWeb 默认为每种类型的日志保留 180 个压缩文件，超过后会删除较早生成的压缩包。

功能说明：日志压缩功能需要配合日志轮转功能进行使用。

以系统日志为例：若开启系统日志的压缩功能，设置压缩包存放目录，并设置压缩执行时间（21 点），程序会在每日 21 点获取系统日志的轮转文件进行压缩，将生成的压缩包存放在指定路径，并删除压缩过的轮转文件。若压缩时不存在系统日志的轮转文件，则当日不生成压缩包。

7.5.5 日志路径配置

1. 展开管理控制台左侧导航树中的“日志服务”节点；
2. 点击“日志路径配置”节点，出现如图 7.5.7 所示。

日志保存路径配置 配置日志的保存路径 

所有的路径配置都支持绝对路径，若是写的相对路径，则从tongweb根目录开始

SQL日志目录	<input type="text" value="logs"/>	设置sql日志的存放目录,不填则为默认路径
审计日志目录	<input type="text" value="logs/audit-log"/>	设置审计日志的存放目录,不填则为默认路径
持久化日志目录	<input type="text" value="persistence"/>	设置持久化日志的存放目录,不填则为默认路径

图 7.5.7 日志路径配置

可在该页面配置相关日志文件的保存路径，所有的路径配置都支持绝对路径，若是写的相对路径，则从 TongwWeb 根目录开始。

- Sql 日志：默认存放在 TongwWeb 根路径的 logs 文件夹。
- 审计日志：默认存放在 TongwWeb 根路径的 logs/audit-log 文件夹。
- 持久化日志：默认存放在 TongwWeb 根路径的 persistence 文件夹。

7.5.6 审计日志

7.5.6.1 操作页面

展开管理控制台左侧导航树中的“诊断”节点，点击“审计日志”节点，出现审计日志页面，默认审计日志列表展示当前正在使用的日志文件（TW_HOME/logs/audit-log/audit.log）内容。如下图所示：

下载日志 搜索日志

序号	日期	用户	操作结果	事件类型	事件描述	摘要
1	2019.09.17 16:52:30	thanos	成功	启动参数	查看启动参数	无
2	2019.09.17 16:51:40	thanos	成功	日志	查看审计日志	无
3	2019.09.17 16:51:26	thanos	成功	用户	用户登录	无
4	2019.09.17 16:47:27	thanos	成功	日志	查看审计日志	无
5	2019.09.17 16:47:05	thanos	成功	日志	查看审计日志	无
6	2019.09.17 16:46:58	thanos	成功	日志	查看访问日志配置	无
7	2019.09.17 16:46:58	thanos	成功	日志	修改访问日志配置	无
8	2019.09.17 16:46:56	thanos	成功	日志	查看访问日志配置	无
9	2019.09.17 16:46:47	thanos	成功	日志	查看访问日志	无
10	2019.09.17 16:46:34	thanos	成功	监视	查看监视明细	无

(总行数:368,总页数:37) 上一页 首页 1 2 3 4 5 6 7 8 9 10 末页 下一页

图 7.3.6 审计日志页面

审计日志页面默认显示第一页的日志内容，点击“下载日志”按钮，可选择下载指定的日志文件，如下图所示：

下载日志 搜索日志

audit.log 下载

✓ 全选 × 全不选

用户	操作结果	事件类型	事件描述	摘要	
thanos	成功	启动参数	查看启动参数	无	
thanos	成功	日志	查看审计日志	无	
thanos	成功	用户	用户登录	无	
thanos	成功	日志	查看审计日志	无	
5 2019.09.17 16:47:05	thanos	成功	日志	查看审计日志	无

图 7.3.7 审计日志下载页面

点击“搜索日志”按钮，可根据需要查找特定的日志内容，如下图所示：

[下载日志](#) [搜索日志](#)

归档文件: 2019.09.17 12:38:53 - 2019.09.17 16:52:30

开始日期: 2019.09.17 12:38:53 结束日期: 2019.09.17 16:58:16

事件类型: 请选择 关键字:

登录用户: 请选择 操作结果: 不限 成功 失败

每页行数: 10 行 排序方式: 正向 反向

序号	日期	用户	操作结果	事件类型	事件描述	摘要
1	2019.09.17 16:52:30	thanos	成功	启动参数	查看启动参数	无
2	2019.09.17 16:51:40	thanos	成功	日志	查看审计日志	无
3	2019.09.17 16:51:26	thanos	成功	用户	用户登录	无

图 7.3.8 审计日志搜索页面

7.5.6.2 配置

审计日志文件存储目录为 `TW_HOME/logs/audit-log/`，当前使用的日志文件名称为 `audit.log`，轮转的日志文件按轮转时刻的日期和时间命名。

通过 `-D` 参数可以配置日志相关功能，

`audit.log.enabled`: 审计日志开关，`boolean` 类型，`true` 表示开启审计日志记录功能，`false` 表示关闭审计日志记录功能，默认值为 `true`。

`audit.log.file.maxSize`: 审计日志文件按大小轮转阈值，长整数类型，单位“KB”，默认值为 1048576，即 10 MB。

`audit.log.file.retentionDays`: 审计日志文件按日期清理阈值，整数类型，单位为“天”，默认值为 180，即 6 个月。

o

7.6 JavaMail 资源

7.6.1 JavaMail 介绍

JavaMail 应用通过调用 JavaMail API 实现从 Java 应用程序访问网络或 Internet 上支持 Internet 消息访问协议 (Internet Message Access Protocol, IMAP) 和简单邮件传输协议 (Simple Mail Transfer Protocol, SMTP) 的邮件服务器。它本身不提供邮件服务器功能，您必须有权访问邮件服务器，才能使用 JavaMail。

JavaMail API 是一组用于建立邮件系统模型的抽象 API。API 提供了一个与平台无关以及与协议无关的框架来建立邮件应用程序和消息传送应用程序。JavaMail API 提供了用于读取和发送电子邮件的工具。使用 JavaMail API，您可以向应用程序中添加电子邮件功能。

7.6.2 TongWeb7 中的 JavaMail

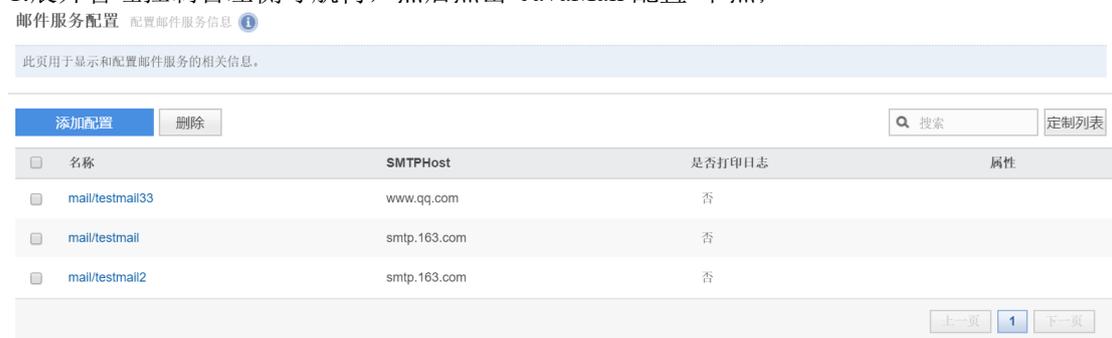
TongWeb7 提供 JavaMail 会话 (`javax.mail.Session`) 的管理。用户在使用过程中需要创建 JavaMail 资源，随后服务器会自动创建 JavaMail 会话，并将 JavaMail 资源中设置的属性设置给 JavaMail 会话。

用户可以在应用程序中通过 `lookup` 方式得到 JavaMail 会话 (`lookup` 应用组件名空间中的 JavaMail 资源获取到 JavaMail 会话)，然后使用 JavaMail 会话进行收发邮件、回复邮件、转发邮件等操作。

7.6.3 JavaMail 资源的使用

7.6.3.1 创建 JavaMail 资源

1. 展开管理控制台左侧导航树，然后点击“JavaMail 配置”节点：



2. 在出现的页面中点击“添加配置”按钮，出现如图所示的创建 JavaMail 资源页面：



属性介绍

- 名称：JavaMail 资源的唯一标识，建议以 mail/开头。
- SMTPHost：邮件服务器地址，如：smtp.163.com
- 是否打印日志：用于调试 javamail 会话，主要是对协议的追踪信息。默认不开启该功能。
- 会话用户名：连接邮件服务器时使用的默认用户名，用于登录验证邮件服务器。
- 会话邮箱：连接邮件服务器后默认的邮件发送者邮箱号，必须在登录验证后的邮箱范围，如果用户名使用的是邮箱号码，则改属性可以设置为相同的内容。发送邮件时则不用再设置邮件的 From 属性。
- 会话密码：连接邮件服务器时使用的默认密码，用于登录验证邮件服务器。
- 属性：其他邮件相关属性，如在实际开发中需要用到其他属性，则以 key=value 的方式配置，多个以分号隔开，如果设置的参数不符合参数规范或者不符合 javamail 参数规范，则不符合规范的参数不会生效。

3. 设置属性完成后，点击“保存”按钮。

7.6.3.2 查看/编辑 JavaMail 资源

1. 展开管理控制台左侧导航树，然后点击“JavaMail 配置”节点：

2. 点击已创建 JavaMail 资源的 JNDI 名，出现如图所示的页面。

邮件服务配置 配置邮件服务信息 ⓘ

此页用于显示和配置邮件服务的相关信息。

[返回](#) | [修改邮件服务配置项](#)

* 名称	mail/testmail	唯一标识, Mail配置名称
* SMTPHost	<input type="text" value="smtp.163.com"/>	邮件服务器地址
是否打印日志	<input type="text" value="否"/>	是否打印详细日志
会话用户名	<input type="text" value="thanos"/>	默认邮件会话用户名
会话邮箱	<input type="text" value="conglinyu1.0@163.com"/>	默认会话邮箱地址
会话密码	<input type="text" value="....."/>	默认邮件会话密码
属性	<input type="text"/>	其他属性, 必须以key=value;key=value,不符合规范的属性不能生效

[保存](#) [取消](#)

说明：属性同“创建 JavaMail 资源”中的属性，名称不可编辑，其他属性均可编辑。

7.6.3.3 删除 JavaMail 资源

1. 展开管理控制台左侧导航树，然后点击“JavaMail 配置”节点；

2. 选中需要删除的 JavaMail 资源。

3. 点击“删除”按钮。

7.6.4 使用 JavaMail 的示例

一、创建 mail 资源，具体创建方式参照 7.7.3.1。

```

try {

    Context ctx = InitialFactoryContext.getContextInstance();
    //lookup查找创建的mail资源, jndi的名称
    Session session = (Session) ctx.lookup("mail/testmail");
    try {
        //创建MIME邮件对象
        MimeMessage mimeMsg =new MimeMessage(session);
        //设置发信人
        //mimeMsg.setFrom(new InternetAddress(MailFrom));
        //设置收信人
        mimeMsg.setRecipients(Message.RecipientType.TO, InternetAddress.parse("xxxx@qq.com"));
        //设置邮件主题
        mimeMsg.setSubject("测试测试1222233333","gb2312");
        //设置邮件内容, 将邮件body部分转化为HTML格式
        mimeMsg.setContent("ceshi chenggong","text/html;charset=gb2312");
        //发送邮件
        Transport.send(mimeMsg);
    } catch (Exception e) {
        e.printStackTrace();
    }
} catch (NamingException e) {
    e.printStackTrace();
}
}

```

8 启动参数配置

8.1 概述

启动参数指的是服务器启动脚本中设置的参数，包括 JVM 参数，服务器参数，环境变量 JAVA_HOME，其中 JVM 参数又包括常见的最大堆内存参数和最小堆内存参数、选择常用的垃圾回收方法（ConcMarkSweepGC、ParallelGC、ParallelOldGC）及远程调试参数，和其他 JVM 参数。

服务器管理控制台提供配置界面，可以按需要对参数进行修改，删除（谨慎使用），和新增参数。配置结果直接保存到相应运行平台对应的启动脚本文件中，重启服务器后生效。

由于 JDK 和服务器各版本参数存在因版本不同而不同的可能性，所以并不能保证所有参数的有效性，因此配置参数时请查阅相关文档，结合当前运行 JDK 和服务器版本，选择正确的参数进行配置。

8.2 参数配置

1. 参数展现。点击管理控制台左侧导航树中的“启动参数配置”节点将出现“启动参数配置”页面。页面按参数类型分别在不同标签页中分组显示从当前运行平台对应的启动脚本中提取出的参数。

启动参数配置 [启动参数配置](#) ⓘ

此页显示了服务器启动参数，包括JVM参数，服务器参数，环境变量JAVA_HOME等，可以执行修改删除和保存操作

jvm参数
其他jvm参数
服务器参数
环境变量

最大堆内存	<input type="text" value="2048m"/>	
最小堆内存	<input type="text" value="2048m"/>	
垃圾回收方法	<input type="text" value="请选择"/>	选择垃圾回收方法
远程调试	<input \$2"="" type="text" value="-Xrunjdpw.transport=dt_socket,server=y,suspend=n,address=\"/>	远程调试项，正确格式：-Xrunjdpw.tran: [%1\$2数字]，中间不含空格

保存
取消

图 8.2.1 启动参数配置

2. 配置参数。在以上参数显示页面对应的文本编辑框中可对已有参数进行修改，在“其他 JVM 参数”和“服务器参数”参数分组类型中可删除参数和新增参数。点击页面下方“保存”按钮进行保存。
3. 启动脚本文件备份。在读取和保存配置参数时，自动对相应运行平台启动脚本文件进行备份，以备手工恢复脚本。备份文件名称为“启动脚本文件名称.template”。
4. 环境变量。环境变量显示 JAVA_HOME 环境变量，当启动脚本中没有配置 JAVA_HOME 时，默认显示当前启动的 TongWeb 使用的 JAVA_HOME 环境变量（启动时最终生效的 JAVA_HOME 环境变量）。当在启动脚本中适当位置配置了 JAVA_HOME 环境变量时，显示配置的环境变量，当保存时，启动脚本调整位置至调用环境变量配置文件 environment.conf 或 environment.bat 行(如果有)之后，保证设置的环境变量覆盖 environment.conf 或 environment.bat 里的 JAVA_HOME 配置。如果清空配置并保存后，将清除启动脚本中 JAVA_HOME 配置，同时管理控制台显示当前启动的 TongWeb 使用的 JAVA_HOME 环境变量。

8.3 参数格式

参数填写需遵循一定的格式，比如：

1. 常见参数格式：

- D<name>=<value> JVM 参数和 服务器参数适用
 - XX:+<option> JVM 参数（非稳定参数）
 - XX:-<option> JVM 参数（非稳定参数）
 - XX:<option>=<string> JVM 参数（非稳定参数）
 - XX:<option>=<number> number 后面可跟单位 M ,m ,K ,k ,g ,G JVM 参数（非稳定参数）
 - X<option> JVM 参数（扩展参数）
- 对于 -Xmx -Xms 须符合 -X<option><size> 格式，size 为 正整数，后面可跟单位 M ,m ,K ,k ,g ,G 。 -X<option> 指的是 -Xmx -Xms
- <option> 非 -X 和 -XX 例如 -server JVM 参数（标准参数）

以上如未特别说明，<> 中内容为字符串或数字

2. 远程调试参数格式：-

Xrunjdpw:transport=dt_socket,server=[y|n],suspend=[y|n],address="[%1|\$2|具体数字]"

说明：

[y|n] 表示值为 y 或者 n

%1 和 %2 分别表示命令行运行 windows 和 linux/unix 启动脚本 debug 选项时传入参数对应的变量名称，不可换做其他值，除了一个具体数字，该数字代表远程调试监听端口号，这时命令行传入的 debug 端口号无效。

3. 其他说明：

最大堆内存和最小堆内存参数可以不填，由 JVM 来控制。JAVA_HOME 环境变量可不设置，以上参数或环境变量如果设置后，将写入启动脚本文件中。

8.4 使用限制

为了保证配置启动参数后，能正确启动服务器，除了上述配置参数的说明，使用上还有些限制，需要特别说明的是，这些使用限制的产生原因大多数并不是启动参数配置功能本身的支持程度问题，而是操作系统程序对启动脚本的解释执行支持程度的问题，例如：

1. 中文支持限制。

启动参数包括 JAVA_HOME 环境变量不支持中文，因为操作系统脚本执行程度对写入脚本的中文解析时会出错，导致脚本程序无法正常执行，JVM 对自有参数含中文支持程度不够，例如 -Xloggc:/path/gc.log, path 中如果包含中文，并不产生 gc 日志，一个检验的标准就是不通过启动参数配置功能进行配置，手工配置同样参数产生同样的效果，但这个效果并不是管理控制台进行参数配置带来的，启动参数配置功能仅仅是个替代手工配置的工具，另外需要说明的是，如果环境变量含中文，例如 JDK 配置路径包含中

文，但是没有写入启动脚本中，JVM 却能识别并支持该中文路径，但这属于 JVM 对中文的支持程度问题，与管理控制台启动参数配置功能无关。

2. JAVA_HOME 环境变量路径中不支持空格。

这个限制仍然和操作系统脚本执行程序有关，脚本解释执行程序自动将空格分隔为参数，如果 JAVA_HOME 环境变量路径中包含空格，将被分解为多个参数，造成脚本执行错误。如果参数中包含空格可以用引号包含。

3. 禁止手工改变启动脚本程序结构，否则启动参数配置管理功能将可能无法正常运行。这个限制产生原因和启动参数配置管理功能设计和实现有关，启动参数配置能设计为通过 web 界面配置替代手工直接修改启动脚本文件的工具，直接作用于启动脚本，启动脚本是纯文本无标准格式的文件，所以它的解析只能依赖于固定的结构，一旦手工修改了启动脚本结构，将造成启动脚本无法正确解析并执行。因此有以下限制，

a) 手工修改启动脚本文件仅限修改 JVM_OPTS , TW_OPTS 等文字区域的参数(包括增加，修改，删除)，不建议随意在其他地方加条件判断，函数等其他脚本程序逻辑单元。

b) 对于现有条件判断语句也实现了支持，例如：

```
If %JAVA_VERSION% GEQ 8 (  
set JAVA_OPTS= -XX:MaxMetaspaceSize=192m  
) else (  
set JAVA_OPTS=-XX:MaxPermSize=192m  
)
```

这段条件逻辑语句，已经可以支持，但位置只能位于 TW_OPTS 之前，也就是现有位置。这段条件逻辑语句里的参数无法通过 web 界面被删除（可以手工删除），否则可能破坏掉该条件语句的逻辑合理和完整性，但是可以修改参数的值。

另外，根据条件逻辑语句中 JVM 识别生效的参数，将可在界面显示编辑。修改后 JVM 堆参数会移至 JVM_OPTS 区域。

8.5 外部 JVM 参数配置

在某些 linux/unix 系统下，由于操作系统安全策略限制等原因，不允许对可执行脚本进行修改，即不能修改 startserver.sh 和 environment.conf 脚本，此时可以通过修改 external.vmoptions 文件来配置启动参数，参数需是标准的 Java 启动参数，多个参数以空格分隔，例如：-Xms256m -Xmx512m -XX:MaxPermSize=192m。

9 JCA

9.1 概述

JCA(Java Connector Architecture) 提供了一个应用服务器和企业信息系统(EIS)连接的标准 Java 解决方案，以及把这些系统整合起来的方法。JCA 简化了异构系统的集成，用户只要构造一个基于 JCA 规范连接器应用，并将该连接器应用部署到 J2EE 服务器上，这样不用编写任何代码就可以实现 EIS 与 J2EE 应用服务器的集成。

9.2 TongWeb7 中的 Connector

TongWeb7 中的 JCA 架构实现了 JCA1.7 规范。

Connector 应用也称为资源适配器，它是允许应用程序与企业信息系统(EIS)进行交互式操作的 Java EE 组件。类似其他 Java EE 模块，安装连接器应用即部署该连接器应用。

TongWeb7 同时提供连接器连接池、托管对象资源，连接器连接池是一组用于特定 EIS 的可重复使用的连接。要创建连接器连接池，请指定与池关联的连接器应用（资源适配器）。托管对象资源用于封装在连接器应用中的托管对象(ra.xml 中定义的 adminobject)，如 JMS 资源中通过托管对象资源提供目的地对象。

9.3 线程池

9.3.1 连接池概述

线程池的作用就是限制系统中执行线程的数量。根据系统的环境情况，可以设置线程数量，达到运行的最佳效果。

- 1) 多个任务重用线程，线程创建的开销被分摊到了多个任务上。这样在请求到达时线程已经存在，所以也消除了线程创建所带来的延迟。
- 2) 通过适当地调整线程池中的线程数目，也就是当请求的数目超过某个阈值时，就强制其它任何新到的请求一直等待，直到获得一个线程来处理为止，从而可以防止资源不足。

9.3.2 TongWeb7 中的线程池

线程用于处理用户程序组件的用户请求。服务器接收到请求时，它会将请求指定给线程池中的空闲线程。该线程执行客户机的请求并返回结果。如果请求需要使用的系统资源当前正处于忙碌状态，则线程会在允许请求使用该资源前，等待资源回到空闲状态。

线程池可以指定最大线程数和最小线程数，线程池在这两个值之间动态调整。指定的最小线程池大小将通知服务器为应用程序请求至少分配该大小的预留线程数。可以将线程数增加到所指定的最大线程池大小。如果增加可供进程使用的线程数，则该进程可以同时为更多的应用程序进行响应。因此用户可以调整最大线程数和最小线程数来提高性能。

9.3.3 创建线程池

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 线程池”节点；列表显示所有已创建的线程池。其中“default-thread-pool”为 TongWeb7 默认提供的线程池如图 9.3.1 所示：

JCA线程池管理 管理JCA线程池 ⓘ

本页面显示了已创建的JCA线程池，可以更新、删除所创建的线程池或者创建新的线程池。

<input type="checkbox"/>	名称	最小线程数	最大线程数	等待队列	线程空闲时间
<input type="checkbox"/>	default_thread_pool	10	200	100	3600
<input type="checkbox"/>	testpool	10	200	100	3600

图 9.3.1 线程池列表页面

2. 点击“创建线程池”按钮，出现如图 9.3.2 所示的“创建线程池”页面：

JCA线程池管理 管理JCA线程池 ⓘ

本页面显示了已创建的JCA线程池，可以更新、删除所创建的线程池或者创建新的线程池。

[返回](#) | [创建JCA线程池](#)

* 线程池名称	<input type="text"/>	唯一标识符，JCA线程池名称
* 最小线程数	<input type="text" value="10"/>	线程池内的核心线程数
* 最大线程数	<input type="text" value="200"/>	线程池内的最大线程数
* 等待队列	<input type="text" value="100"/>	当核心线程全部被占用时，新的任务将被保存在等待队列中
* 线程空闲超时时间	<input type="text" value="3600"/>	超出核心线程数的线程如果在该空闲时间内没有收到新的任务，则销毁该线程，单位为秒，设置为0表示关闭线程空闲超时

[保存](#) [取消](#)

图 9.3.2 线程池创建页面

- 线程池名称：线程池的唯一标识 ID。
 - 最小线程数：线程池中线程个数的下限，当线程池初始化时创建该数量的线程。
 - 最大线程数：线程池中线程个数的上限。
 - 等待队列：当核心线程全部被占用时，新的任务将被保存在等待队列中。等待队列在初始化时会占用部分内存，如果队列配置过大可能会导致线程池创建失败。
 - 线程空闲超时时间：超出核心线程数的线程如果在该空闲时间内没有收到新的任务，则销毁该线程，单位为秒，设置为 0 表示关闭线程空闲超时。
3. 设置属性完成后，点击“保存”按钮。

9.3.4 查看/编辑线程池

1. 展开管理控制台左侧导航树中的“JCA”节点，并点击“JCA 线程池”节点；
2. 点击要查看/编辑的线程池名称，出现如图所示页面：

JCA线程池管理 管理JCA线程池 ⓘ

本页面显示了已创建的JCA线程池，可以更新、删除所创建的线程池或者创建新的线程池。

[返回](#) | [编辑JCA线程池](#)

线程池名称	testpool	唯一标识符，JCA线程池名称
* 最小线程数	<input type="text" value="10"/>	线程池内的核心线程数
* 最大线程数	<input type="text" value="200"/>	线程池内的最大线程数
* 等待队列	<input type="text" value="100"/>	当核心线程全部被占用时，新的任务将被保存在等待队列中
* 线程空闲超时时间	<input type="text" value="3600"/>	超出核心线程数的线程如果在该空闲时间内没有收到新的任务，则销毁该线程，

[保存](#)

图 9.3.3 线程池编辑页面

具体属性说明同“创建线程池”中的属性说明。且“线程池名称”不能编辑，其他属性均可编辑。

3. 编辑属性完成后，点击“保存”按钮。

9.3.5 删除线程池

1. 展开管理控制台左侧导航树中的“JCA”节点，并点击“JCA 线程池”节点；
2. 选中需要删除的线程池；
3. 点击“删除”按钮。

9.3.6 Connector 应用中指定线程池

用户需要在 Connector 应用部署过程中指定自定义的线程池，若不指定使用默认提供的线程池 default-thread-pool。

9.4 JCA 连接池

9.4.1 创建连接池

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 连接池”节点，如图 9.4.1；



图 9.4.1 连接池列表页面

2. 在出现的“连接池”页面中点击“创建连接池”按钮，页面显示信息如下图 12.4.2：



图 9.4.2 创建连接池

- 基本属性
- 名称：连接池的名称，连接池的唯一标识。

- 资源适配器：用来创建该连接池的连接器应用的应用名。
- 连接定义：连接器应用中 ra.xml 中定义的 connection-definition-interface。
- 池设置
 - 最小连接数：池中连接的最小数目。该值还确定了首次创建池或应用服务器启动时，置于池中的连接的数目。默认值为 10。
 - 最大连接数：池中连接的最大数目。默认值为 100。
 - 空闲超时：连接在池中保持空闲的最长时间（以分钟为单位）。一旦超过此时间，即从池中删除该连接。默认值为 10 分钟。
 - 等待超时：在达到超时之前，请求连接的应用程序所等待的时间。默认值为 60 秒。
 - 连接匹配：获取连接时是否由资源适配器进行匹配，默认为否。
 - 事务支持：设置连接池的事务支持类型。可选值为 NoTransaction、LocalTransaction 和 XATransaction。
 - NoTransaction：表示资源适配器不支持资源管理器本地事务或 JTA 事务，也不实现 XAResource 或 LocalTransaction 接口。对于 JAXR 资源适配器，需要从“事务支持”菜单中选择“无”。JAXR 资源适配器不支持本地或 JTA 事务。
 - LocalTransaction：表示资源适配器将通过实现 LocalTransaction 接口来支持本地事务。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。
 - XATransaction：表示资源适配器将通过实现 LocalTransaction 和 XAResource 接口来支持资源管理器本地事务和 JTA 事务。XA 事务由事务管理器在资源管理器外部进行控制和调整。本地事务的管理在资源管理器内部进行，不涉及任何外部事务管理器。
- 其他属性
 - 配置连接器应用中 ra.xml 中 connection-definition 元素下定义的 config-property 属性

9.4.2 查看/编辑连接池

1. 展开管理控制台左侧导航树中的“JCA”节点，并点击“JCA 线程池”节点；
2. 点击要查看/编辑的线程池名称，出现如图所示 9.4.3 页面；

返回
编辑JCA连接池

连接池名称	mytest	唯一标识符，JCA连接池名称
* 资源适配器	genericra	用于创建该连接池的连接器应用
* 连接定义	javax.jms.QueueConnectio...	连接器应用中ra.xml中定义的connection-definition-interface
* 最小连接数	10	连接池的初始化连接数
* 最大连接数	100	连接池中的最大连接数
* 等待超时	60	从连接池中获取连接的最长等待时间，单位为秒
* 空闲超时	600	连接的空闲超时时间，单位为分钟
连接匹配	<input type="checkbox"/> 开启	获取连接时由资源适配器进行匹配
事务支持	NoTransaction	连接池的事务支持类型

属性配置

新增

属性名称	值	操作
ConnectionValidationEnab	<input type="text"/>	删除
ConnectionFactoryJndiNa	<input type="text"/>	删除
ClientId	<input type="text"/>	删除

保存

图 9. 4. 3 连接池编辑页面
属性同创建连接池资源的属性配置。

9.4.3 删除连接池资源

1. 展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 连接池”节点；
2. 选中要删除的 JCA 连接池；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

9.4.4 创建安全映射

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 连接池”节点；
2. 右侧页面中点击某连接池的“安全映射”链接，进入“安全映射”页面；
3. 点击页面左侧的“创建安全映射”按钮，具体配置如下图：

创建安全映射页面。

|

连接池名称 **mytest**

* 安全映射名称 唯一标示符，安全映射名称

请选择 用户组 主体 调用方的身份，即在访问带安

* 用户组 用户组名称

* 用户名 访问EIS所需的用户名

* 密码 访问EIS所需的密码

图 9.4.4 安全映射创建页面

- 连接池名称：所选连接池的名称，不可以改变。
- 安全映射名称：安全映射的名称，安全映射的唯一标识。
- 选择用户组/主体：调用方的身份，即在访问带安全的应用时，成功登陆的用户的身份，可以用已定义的主体或用户组代表。
- 用户组/主体：用户组/主体名称。
- 用户名：访问 EIS 所需的用户名, 长度为 1-256 位。
- 密码：访问 EIS 所需的密码，长度为 4-32 位。

9.4.5 查看/编辑安全映射

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 连接池”节点；
2. 右侧页面中点击某连接池的“安全映射”按钮，进入“安全映射”页面。

本页面显示了已创建的安全映射，可以更新、删除所创建的安全映射。

<input type="checkbox"/>	名称
<input type="checkbox"/>	test

图 9.4.5 安全映射列表页面

3. 点击要查看/编辑的安全映射名称，便可查看/编辑安全映射，属性配置见安全映射创建页面。

9.4.6 删除安全映射

1. 展开管理控制台左侧导航树中的“JCA”节点，然后点击“JCA 连接池”节点；
2. 点击“安全映射”链接，选择想要删除的安全映射；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

9.5 托管对象资源

9.5.1 创建托管资源对象

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“托管对象资源”节点，如图 12.5.1：



图 9.5.1 托管对象资源列表页面

2. 在出现的“托管对象资源”页面中点击“创建”按钮，页面显示的信息如下图 9.5.2：



本页面显示了已创建的托管对象资源，可以更新、删除所创建的托管对象资源或者创建新的托管对象资源。

|

* JNDI名称

* 资源适配器 资源适配器

* 资源类型 资源类型

属性配置

属性名称	值
<input type="text" value="DestinationJndiName"/>	<input type="text"/>
<input type="text" value="DestinationProperties"/>	<input type="text"/>

图 9. 5. 2 托管对象资源创建页面

- 基本设置
 - JNDI 名称：托管对象资源的 JNDI 名称。
 - 资源类型：连接器应用中 ra.xml 中定义的 adminobject-interface。
 - 资源适配器名称：与托管对象资源关联的已部署的连接器应用的应用名。
 - 属性设置
 - 连接器应用中 ra.xml 中 adminobject 元素下定义的 config-property 属性。
3. 点击“保存”，返回资源适配器列表页面。

9.5.2 查看/编辑托管资源对象

1. 依次展开管理控制台左侧导航树中的“JCA”节点，然后点击“托管对象资源”节点；
2. 点击要查看/编辑的托管对象资源的 JNDI 名称，编辑托管对象资源 adminobject 的信息显示如下图 9. 5. 3：

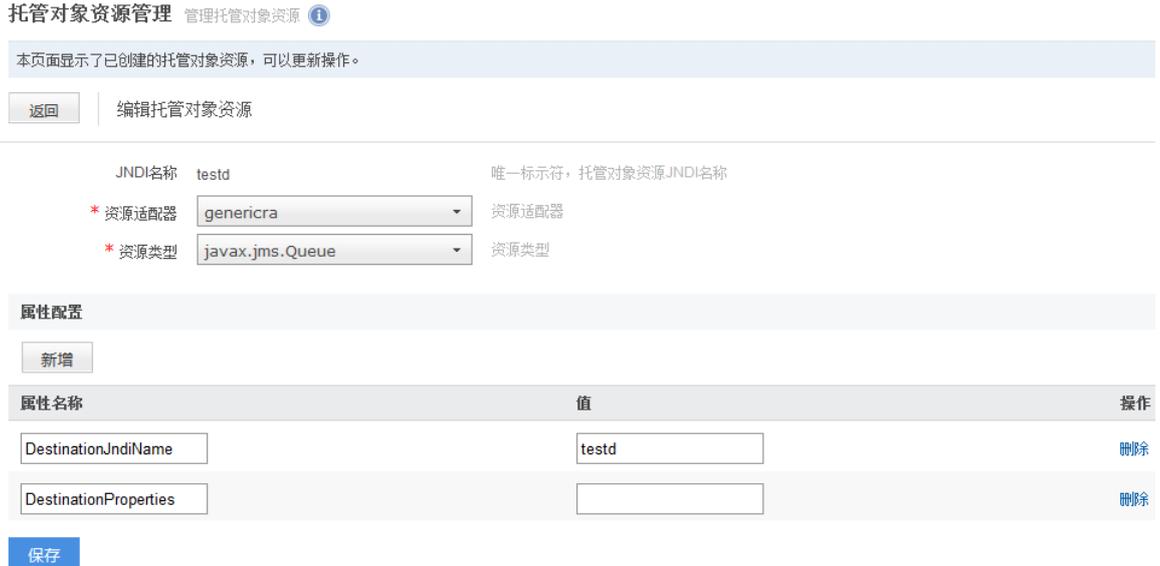


图 9.5.3 托管对象资源编辑页面

9.5.3 删除托管资源对象

1. 展开管理控制台左侧导航树中的“JCA”节点，然后点击“托管资源对象”节点；
2. 选中要删除的 JCA 托管资源对象；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

10 工作管理器

10.1 概述

工作管理器(commonj.work.WorkManager 接口)是 [JSR 237](#) 定义的用于在 JEE 应用中并发编程的 API。通过此 API，可以并发编写在 Java EE 应用程序中的 EJB 和 Servlet 程序，应用可通过工作管理器使用线程池中线程运行应用程序创建的工作任务，而这些线程是被服务器所管理的，而不是应用程序创建的需要由应用程序自身维护的，此 API 通常被称为 CommonJ, TongWeb 提供对此 API 的支持。

10.2 创建工作管理器

1. 点击管理控制台左侧导航树“工作管理器”，出现已创建的工作管理器列表页面。
2. 在工作管理器列表页面点击按钮“创建工作管理器”，在如下图 10.2.1 页面配置工作管理器参数，点击“保存”创建工作管理器资源。

工作管理器管理 管理工作管理器资源 ⓘ

本页面显示可创建新的工作管理器资源。

[返回](#) | [创建工作管理器](#)

* 工作管理器名称	<input type="text"/>	唯一标识符，工作管理器JNDI名称
* 最小线程数	<input type="text" value="2"/>	工作管理器线程池内的核心线程数
* 最大线程数	<input type="text" value="5"/>	工作管理器线程池内的最大线程数
* 等待队列	<input type="text" value="20"/>	当工作管理器核心线程全部被占用时，新的工作任务将被保存在等待队列中
* 允许最大后台任务数	<input type="text" value="10"/>	允许最大后台任务数

[保存](#) [取消](#)

图 10.2.1 工作管理器资源创建页面

配置项说明：

- 工作管理器名称：标识 work Manager 实例，同时也是绑定 jndi 服务上资源的 jndi 名称
- 最小线程数：线程池最小线程数，默认值 2
- 最大线程数：线程池最大线程数，默认值 5
- 允许最大后台任务数：允许作为后台任务允许的最大任务数，默认值 10
- 等待队列：等待队列小大小，默认值 20

注意：工作管理器配置项保存在 TongWeb 安装根目录的子目录 conf 下的 workmanagers.xml 文件中。

10.3 查看/编辑工作管理器

1. 点击管理控制台左侧导航树“工作管理器”，出现已创建的工作管理器列表页面。
2. 在工作管理器列表页面 JNDI 名称列点击需要编辑的工作管理器链接，在如下图 10.3.1 页面编辑工作管理器参数，点击“保存”按钮，保存工作管理器资源。

本页面显示了已创建的工作管理器资源，可以更新工作管理器资源。

返回	修改工作管理器	
工作管理器名称	wm/FooWorkManager	唯一标示符，工作管理器JNDI名称
* 最小线程数	<input type="text" value="2"/>	工作管理器线程池内的核心线程数
* 最大线程数	<input type="text" value="5"/>	工作管理器线程池内的最大线程数
* 等待队列	<input type="text" value="20"/>	当核心线程全部被占用时，新的任务将被保存在等待队列中
* 最大后台任务数	<input type="text" value="10"/>	允许最大后台任务数
<input type="button" value="保存"/>		

图 10.3.1 工作管理器资源编辑页面

10.4 删除工作管理器资源

1. 点击管理控制台左侧导航树“工作管理器”，出现已创建的工作管理器列表页面。
2. 在工作管理器列表页面勾线需要删除的工作管理器资源，点击“删除按钮”，确认后删除工作管理器资源。删除后，在 JNDI 树上不再显示相应的工作管理器资源。

10.5 使用工作管理器

应用通过资源引用使用工作管理器实例:例如 web 应用:

在 web.xml 中配置:

```
<resource-ref>
    <res-ref-name>wm/FooWorkManager</res-ref-name>
    <res-type>commonj.work.WorkManager</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
</resource-ref>
```

在代码中通过 jndi lookup :

```
Context ctx = new InitialContext();
wm = (WorkManager) ctx.lookup("java:/comp/env/wm/FooWorkManager");
然后运用 commonj API 调度任务执行。
```

11 类加载分析工具

11.1 概述

在应用移植、部署等业务操作中，如果某些类文件存在多份，或者同一个类被不同

的类加载器加载，就可能遇到运行期类型不兼容等类加载冲突问题，一般此类问题的解决办法都需要首先定位到冲突的类资源，然而冲突类资源的定位有时候会是非常困难或复杂的，此类加载分析工具可以在一定程度上帮助开发或维护等人员快速定位此类问题的发生原因，并尝试给出一些简单的建议性的解决方案。

11.2 名词定义

11.2.1 冗余

在一个类加载器的加载路径范围内，同一个类（类全限定名相同）在不同的 jar 或文件夹内，存在多份则称为冗余类，多份的冗余类能够被真正加载到内存里的只有一份，具体要加载哪个由 java 虚拟机决定，冗余类可能会导致我们加载到了错误的类（如版本错误）。

11.2.2 潜在冲突

被不同的类加载器加载的同一个类（类全限定名相同），在相互作用（如类型转换）时，会发生类型不匹配的错误，这种类在运行时具有潜在的风险，称为潜在冲突类。

11.3 类加载器树

11.3.1 类加载器树概述

类加载器树用于查看系统级和应用级类加载器的信息，类加载器树状结构可以清晰地展示出类加载器之间的层次关系，也可以查看具体的类加载器的详细信息。

11.3.2 类加载器树使用示例

1. 展开管理控制台左侧导航树中的“类加载分析工具”节点，点击“类加载器树”节点，出现如图 11.1 所示的类加载器信息页面；
2. 在类加载器树状结构中使用鼠标定位可以查看的系统级或应用级的类加载器，鼠标定位处便显示该类加载器的描述信息（如系统类加载器）；
3. 在类加载器树状结构中，点击需要查看的具体类加载器，具体信息如图 9.3.1:

类加载器树 展示类加载器的相关信息 ⓘ

用于展示类加载器的层次结构与详细信息。

类加载器树	类加载器信息
<ul style="list-style-type: none">BootstrapClassLoader<ul style="list-style-type: none">sun.misc.Launcher\$ExtClassLoader<ul style="list-style-type: none">sun.misc.Launcher\$AppClassLoader<ul style="list-style-type: none">com.tongweb.web.thor.loader.StandardClassLoader<ul style="list-style-type: none">MyApp<ul style="list-style-type: none">MyApp:MyWebTC_examplesejb2<ul style="list-style-type: none">ejb2:ejb2_war	<p>描述: 系统类加载器</p> <p>类型: sun.misc.Launcher\$AppClassLoader</p> <p>已加载类: 34 个</p> <p>加载路径:</p> <ul style="list-style-type: none">• /home/wangpl/mine/soft/jdk1.8.0_25/lib/tools.jar• /home/wangpl/Desktop/tw6/lib/launcher.jar• /home/wangpl/Desktop/tw6/lib/tongejb-javaagent.jar

图 11.1 类加载器信息

点击后页面右侧展示出该类加载器的详细信息，详细信息包括：类加载器的类型（Class 类型）、加载路径、已加载的类个数（不支持 AIX 系统）。

11.4 类资源分析

11.4.1 类资源分析概述

类资源分析用于分析应用内部，某个类是否存在潜在冲突和冗余现象。应用可以是已部署的应用，也可以是一个以 jar、war、ear 为后缀的应用文件。

11.4.2 类资源分析使用示例

1. 展开管理控制台左侧导航树中的“类加载分析工具”节点；
2. 点击“类资源分析”节点，出现“类资源分析”页面；

3. 在“类资源分析”页面，“选择应用”下拉框默认显示“上传应用”，点击可切换到其它已部署的应用，在“类全名称”输入框输入要分析的类的全限定名（如 Object 类则输入 java.lang.Object）；
4. 点击“分析”按钮，在“分析”按钮下方可以看到分析的结果，如图 11.2 所示：

类资源分析 分析具体类的相关信息

用于分析具体类的加载与分布情况。

* 选择应用 指定或上传一个应用

* 类全名称 类的完全限定名，如String类的类全名称为java.lang.String

分析

— 类 java.lang.Object 在 ejb2 中的分析报告 —

加载明细:	类加载器	加载位置
	JVM31 导类加载器	/home/wangpl/mine/soft/jdk1.8.0_25/jre/lib/rt.jar
	该类的加载状态正常。	

分布明细: 未找到。

— 类 java.lang.Object 在 ejb2:ejb2_war 中的分析报告 —

加载明细:	类加载器	加载位置
	JVM31 导类加载器	/home/wangpl/mine/soft/jdk1.8.0_25/jre/lib/rt.jar
	该类的加载状态正常。	

分布明细: 未找到。

图 11.2 类资源分析结果

11.4.3 类资源分析结果

分析报告中包含“加载明细”和“分布明细”两部分。

“加载明显”展示的是该类可能被加载的情形，包括加载器的名称（鼠标移至加载器的名称前面的小图标可显示出加载器的类型）和加载的位置，如果只有一个类加载器可以加载到该类，则会有提示信息“该类的加载状态正常”；如果有多个类加载器可以加载到该类，则提示“检测到该类有潜在冲突”，并可以看到简单的“建议方案”信息，“建议方案”信息为参考性的解决方案信息，具体的问题解决方案还要视具体的应用问题而定。

“分布明细”展示的是该类在指定的应用内的分布情况，如果该类只有一个分布位置，则会有提示信息“该类的分布状态正常”；如果有多个分布信息，则提示“检测到该类存在冗余”，并可以看到简单的“建议方案”信息，“建议方案”信息为参考性的解决方案信息，具体的问题解决方案还要视具体的应用问题而定。

11.5 类冲突检测

11.5.1 类冲突检测概述

类冲突检测用于分析整个应用内存在的潜在冲突类和冗余类，相比“类资源分析”一次只能分析一个类，“类冲突检测”则可以一次分析应用内所有的类。

11.5.2 类冲突检测使用示例

1. 展开管理控制台左侧导航树中的“类加载分析工具”节点；
2. 点击“类冲突检测”节点，出现“类冲突检测”页面；
3. 在“类冲突检测”页面，“选择应用”下拉框，默认显示“上传应用”，点击可切换到其它已部署的应用，“生成报告”复选框默认没有勾选，如果勾选则在类冲突检测完毕后自动生成检测结果存档文件，见 [11.5.4 类冲突检测报告](#)；
4. 点击“检测”按钮，在“检测”按钮下方可以看到检测的结果，如图 11.3 所示：

类冲突检测 检测整个应用的冲突情况 ⓘ

用于检测整个应用的类加载冲突情况。

* 选择应用 指定或上传一个应用

生成报告 开启 默认为关闭，开启后在服务器logs/cia目录下生成本次检测的报告文件

— **ejb2** 的检测报告 —

未检测到有潜在冲突类且不存在冗余类。

— **ejb2:ejb2_war** 的检测报告 —

未检测到有潜在冲突类且不存在冗余类。

图 11.3 类冲突检测结果

11.5.3 类冲突检测结果

类冲突检测结果和类资源分析结果一样分为“加载明细”和“分布明细”两部分，“加载明细”部分可以看到有潜在冲突类的总个数和各自的详细加载信息，“分布明细”部分可以看到存在冗余类的总个数和各自的分布信息，类冲突检测结果以具体的类为单元进行信息展示，对于单个类的展示信息同“类资源分析”功能中的“类资源分析结果”相似，见 [11.4.3 类资源分析结果](#)。

对于含有多模块的企业应用，在展示类冲突检测结果时是以模块为单位进行展示的，如图 9.5.2 中分别展示了 ejb2 模块和 ejb2:ejb2_war 模块的类冲突检测结果，其中 ejb2 模块是企业应用 ejb2 中的 ejb 模块，ejb2:ejb2_war 模块是企业应用 ejb2 中的 web 模块。

11.5.4 类冲突检测报告

在使用“类冲突检测”功能时，如果勾选了“生成报告”复选框则会生成类冲突检测结果存档文件，位于 %TongWeb7_HOME%/logs/cia 目录下，命名格式为“应用名称-时间戳.txt”，其内容和页面检测结果一致，如图 11.4 所示：

```
=====ejb2的检测报告=====
未检测到有潜在冲突类且不存在冗余类。
-----

=====ejb2:ejb2_war的检测报告=====
未检测到有潜在冲突类且不存在冗余类。
```

图 11.4 类冲突检测报告

12 JMS 服务

12.1 概述

TongWeb7 的 JMS 服务符合 JMS1.1 规范，提供外置 JMS Server 以提供 JMS 服务。

12.2 JMS 主要功能

JMS 用于在两个应用程序之间，或分布式系统中发送消息，进行异步通信。

JMS 应用程序有四个组成部分：JMS 服务提供者、消息管理对象、消息的生产者、消费者和消息本身。

- JMS 服务提供者：JMS 接口的一个实现。提供者可以是 Java 平台的 JMS 实现，也可以是非 Java 平台的面向消息中间件的适配器。
- 消息管理对象提供对消息进行操作的 API，JMS API 中有两个消息管理对象：ConnectionFactory 和 Destination，根据消息的消费方式的不同

ConnectionFactory 可以分为 QueueConnectionFactory 和 TopicConnectionFactory, Destination 可以分为 Queue 和 Topic, 用这两个管理对象可以建立到消息服务的会话。

- 消息的生产者和消费者。消息的生产者即创建并发送消息的 JMS 客户。消息的消费者即接收消息的 JMS 客户。根据消息模式的不同, 消息的消费者分为两类: Subscriber 和 Receiver, 同样消息发送者也为两类: Publisher 和 Sender。
- 消息: 在 JMS 应用程序之间传递的数据的对象。JMS API 中提供五种消息: 映射消息 (MapMessage)、文本消息 (TextMessage)、字节消息 (ByteMessage)、流消息 (StreamMessage) 和对象消息 (ObjectMessage)。

JMS 提供两种消息模式:

1. 点对点模式 (PTP)

PTP (Point-to-Point) 基于队列, 一个生产者向一个特定的队列发布消息, 一个消费者从该队列中读取消息。生产者知道消费者的队列, 并将直接将消息发送到消费者的队列。JMS 队列用于存放那些被发送且等待阅读的消息。一旦一个消息被阅读, 该消息将被从队列中移走。

- 点对点模式的特点:
 - 只有一个消费者将获得消息。
 - 生产者不需要在接收者消费该消息期间处于运行状态, 接收者也同样不需要在消息发送时处于运行状态。

2. 发布/订阅模式 (Pub/Sub)

Pub/Sub 模式支持向一个特定的消息主题发布消息。0 或多个订阅者可能对来自特定消息主题的消息感兴趣。在这种模型下, 发布者和订阅者彼此不知道对方。

- 发布/订阅模式的特点:
 - 多个消费者可以获得消息。
 - 在发布者和订阅者之间存在时间依赖性。

发布者需要建立一个订阅 (subscription), 以便客户能够购订阅。订阅者必须保持持续的活动状态以接收消息, 除非订阅者建立了持久的订阅。在订阅者建立了持久订阅的情况下, 订阅者未连接时, 发布的消息将在订阅者重新连接时重新发布。

12.3 JMS 提供的主要接口

1. ConnectionFactory 接口 (连接工厂)

用户用来创建到 JMS 提供者的连接。JMS 客户通过该接口访问连接, 这样当 JMS 提供者改变时, 代码不需要进行修改。用户在 JNDI 名字空间中配置连接工厂, 这样, JMS 客户才能够查找到它们。根据消费方式的不同, 用户可以使用队列连接工厂 (QueueConnectionFactory) 或者主题连接工厂 (TopicConnectionFactory)。

2. Connection 接口 (连接)

连接代表了应用程序和消息服务器之间的通信链路。在获得了连接工厂后, 就可以创建一个与 JMS 提供者的连接。根据不同的连接类型, 连接允许用户创建会话, 以发送或接收消息。

3. Destination 接口 (目的地)

消息目的地指消息发布和接收的地点, 或者是队列 (Queue), 或者是主题 (Topic)。用户创建这些对象并通过 JNDI 获取。和连接工厂一样, 用户可以使用 PTP 的队列或者 Pub/Sub 的主题。

4. Session 接口 (会话)

表示一个单线程的上下文, 用于发送和接收消息。由于会话是单线程的, 所以消息是按照发送的顺序一个一个接收的。会话的好处是它支持事务。如果用户选择了事务支持, 会话上下文将保存一组消息, 直到事务被提交才发送这些消息。在提交事务之前, 用户可以使用回滚操作取消这些消息。一个会话允许用户创建消息生产者来发送消息, 创建消息消费者来接收消息。

5. MessageConsumer 接口 (消息消费者)

由会话创建的对象，用于接收发送到目的地的消息。消费者可以同步地（阻塞模式），或异步（非阻塞）接收队列和主题类型的消息。

6. MessageProducer 接口（消息生产者）

由会话创建的对象，用于发送消息到目的地。用户可以创建某个目的地的发送者，也可以创建一个通用的发送者，在发送消息时指定目的地。

7. Message 接口（消息）

消息是在生产者和消费者之间传送的对象。

12.4 TongWeb7 中的 JMS

TongWeb7 对 JMS 的支持是集成外置 JMS Server。用户通过连接工厂资源和目的地资源使用 JMS Server 上的消息管理对象(ConnectionFactory 和 Destination)。TongWeb7 提供管理控制台对连接工厂和目的地资源进行管理。管理控制台的使用详见 12.5 小节。

TongWeb7 通过 JCA 提供对外置 JMS Server 的集成。通过资源适配器与第三方 JMS Server 连接。目前 TongWeb7 提供两种集成方式，第一种方式是 JNDI，第二种方式是 JavaBean。前者基于 JNDI 服务并需要将 JMS 对象绑定到第三方 JMS Server 名字服务上，后者基于 JavaBean 的映射而无需将 JMS 对象绑定到名字服务上。

本文提供了 TongWeb7 与 ActiveMQ 和 TongLINK/Q 的集成参考。TongWeb7 默认提供与 ActiveMQ 的集成。提供内容说明如下：

- 已部署通用连接器应用
genericra.rar (TongWeb7_HOME/applications/genericra)，且默认应用名为 genericra。
- ActiveMQ 版本为 ActiveMQ5.14.1 (ActiveMQ 位置为 TongWeb7_HOME/apache-activemq)。注：如果产品名为 apache-activemq-win 或者 apache-activemq-linux，请根据对应平台将 apache-activemq-win 或者 apache-activemq-linux 名字手动修改为 apache-activemq
- JNDI 名字服务使用 Sun 公司提供的文件系统储存 JNDI 对象的 JNDI 服务 fscontext.jar。

查看/编辑默认集成属性的步骤如下：

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 点击应用名称 genericra；
3. 页面会显示已设置的属性，具体属性见下图：

基本属性

应用名称	genericra	应用名称
应用位置	D:/Work/TW_2019-10-12/bin/./applications	应用位置
	/genericra	
部署顺序	<input type="text" value="1"/>	默认的部署顺序是100。如果需要调整部署顺序的话，可以指定
线程池	<input type="text" value="default-thread-pool"/>	如果不选择，默认为default-thread-pool
描述	<input type="text"/>	该应用的描述信息

属性设置

属性名称	值
<input type="text" value="LogLevel"/>	<input type="text" value="INFO"/>
<input type="text" value="JndiProperties"/>	<input type="text" value="java.naming.factory.initial=com.sun.jndi.fscontext."/>
<input type="text" value="RMPolicy"/>	<input type="text" value="OnePerPhysicalConnection"/>
<input type="text" value="SupportsXA"/>	<input type="text" value="true"/>
<input type="text" value="ProviderIntegrationMode"/>	<input type="text" value="jndi"/>

4. 默认使用的是 JNDI 集成方式，如果要调整为 JavaBean 方式，点击全部属性，并按照 JavaBean 方式对相关属性进行编辑并保存。（具体编辑内容同下一节的使用 JavaBean 集成）。

附：默认设置的属性：

```
SupportsXA:false
RMPolicy:OnePerPhysicalConnection
LogLevel:INFO
ProviderIntegrationMode:jndi
JndiProperties:java.naming.factory.initial=com.sun.jndi.fscontext.ReffSContextFactory, java.naming.provider.url=file:${tongweb.root}/ apache-activemq/conf
```

附：全部属性：

LogLevel	INFO
CommonSetterMethodName	
DeliveryConcurrencyMode	
QueueConnectionFactoryClassName	
TopicClassName	
UseFirstXAForRedelivery	
Password	
UserName	
ConnectionFactoryClassName	
ConnectionFactoryProperties	
JndiProperties	java.naming.factory.initial=
RMPolicy	OnePerPhysicalConnectio
XAConnectionFactoryClassName	
SupportsXA	false
UnifiedDestinationClassName	
MDBDeploymentRetryInterval	
QueueClassName	
XAQueueConnectionFactoryClassName	
DeliveryType	
XATopicConnectionFactoryClassName	
MDBDeploymentRetryAttempt	
ProviderIntegrationMode	jndi
Monitoring	
TopicConnectionFactoryClassName	

12.4.1 与 ActiveMQ 集成

ActiveMQ 是目前流行的开源的 JMS Server。以下集成参考使用 TongWeb7 提供的 ActiveMQ5.14.1 (ActiveMQ 位置为 TongWeb7_HOME/ apache-activemq)。

1. 使用 JNDI 集成方式

采用 JNDI 集成方式时需要借助于 JNDI 服务，以下集成参考中使用的 JNDI 服务为 ActiveMQ 自带的 JNDI 服务。

- 1) 启动 TongWeb7 服务器；
- 2) 打开 TongWeb7 的管理控制台，依次展开管理控制台左侧导航树中的“应用管理”节点；

- 3) 点击“部署”按钮，选择自己要使用的连接器应用（.rar 类型），按照应用部署说明，开始部署连接器应用（Connector 应用）。部署完成后，展开管理控制台左侧导航树中的“应用管理”节点，然后点击刚刚部署成功的 Connector 应用，开始编辑属性；
- 4) 点击“全部属性”，配置使用 JNDI 方式集成所需的属性：
 - SupportsXA: 指定 JMS 客户机是否支持 XA，合法值为 true 和 false，默认为 false。
 - RMPolicy: 事务管理器使用 XAResource 的 isSameRM 方法来确定两个 XAResource 所表示的资源管理器实例是否相同。合法值为 ProviderManaged 和 OnePerPhysicalConnection（默认值）。将 RMPolicy 设置为 ProviderManaged 时，JMS 提供者将负责确定通用资源适配器中的 RMPolicy 和 XAResource 包装器仅将 isSameRM 调用委托给消息队列提供者的 XA 资源实现。这应该适用于大多数消息队列产品。有些 XAResource 实现（如 IBMMQ 系列）依靠每个物理连接的资源管理器，当在单个事务中针对同一队列管理器存在入站和出站通信时（例如 MDB 将响应发送到目的地），这将导致问题。将 RMPolicy 设置为 OnePerPhysicalConnection 时，通用资源适配器中 XAResource 包装器实现的 isSameRM 将检查两个 XAResource 是否使用同一物理连接，然后再委托给被包装的对象。
 - LogLevel: 资源适配器的日志级别，合法值为 SEVERE、WARNING、INFO、CONFIG、FINE、FINER、FINEST 和 OFF，默认值为 INFO。
 - ProviderIntegrationMode: 确定资源适配器与 JMS 客户机之间的集成模式，合法值为 javabean 和 jndi，默认为 jndi。
 - JndiProperties: 指定连接到 JMS 提供者的 JNDI 时使用的 JNDI 提供者属性。仅当 ProviderIntegrationMode 为 jndi 时才使用。
- 5) 配置属性完成后，点击“保存”按钮
- 6) 启动 ActiveMQ, TongWeb7 与 ActiveMQ 的集成工作便完成。

2. 使用 JavaBean 集成方式

- 1) 启动 TongWeb7 服务器；
- 2) 打开 TongWeb7 的管理控制台，依次展开管理控制台左侧导航树中的“应用管理”节点；
- 3) 点击“部署”按钮，选择自己要使用的连接器应用（.rar 类型），按照应用部署说明，开始部署连接器应用（Connector 应用）。部署完成后，展开管理控制台左侧导航树中的“应用管理”节点，然后点击刚刚部署成功的 Connector 应用，开始编辑属性；
- 4) 点击“全部属性”，配置使用 JavaBean 方式集成所需的属性：
 - SupportsXA: 指定 JMS 客户机是否支持 XA，合法值为 true 和 false。
 - RMPolicy: 事务管理器使用 XAResource 的 isSameRM 方法来确定两个 XAResource 所表示的资源管理器实例是否相同。合法值为 ProviderManaged 和 OnePerPhysicalConnection。将 RMPolicy 设置为 ProviderManaged（默认值）时，JMS 提供者将负责确定通用资源适配器中的 RMPolicy 和 XAResource 包装器仅将 isSameRM 调用委托给消息队列提供者的 XA 资源实现。这应该适用于大多数消息队列产品。有些 XAResource 实现（如 IBMMQ 系列）依靠每个物理连接的资源管理器，当在单个事务中针对同一队列管理器存在入站和出站通信时（例如 MDB 将响应发送到目的地），这将导致问题。将 RMPolicy 设置为 OnePerPhysicalConnection 时，

通用资源适配器中 XAResource 包装器实现的 isSameRM 将检查两个 XAResource 是否使用同一物理连接，然后再委托给被包装的对象。

- **LogLevel:** 资源适配器的日志级别，合法值为 SEVERE、WARNING、INFO、CONFIG、FINE、FINER、FINEST 和 OFF。
- **ProviderIntegrationMode:** 确定资源适配器与 JMS 客户机之间的集成模式，合法值为 javabean 和 jndi，默认为 javabean。
- **ConnectionFactoryClassName:** JMS 客户机的 `javax.jms.ConnectionFactory` 实现的类名。在 `ProviderIntegrationMode` 为 javabean 时使用。
- **QueueConnectionFactoryClassName:** JMS 客户机的 `javax.jms.QueueConnectionFactory` 实现的类名。在 `ProviderIntegrationMode` 为 javabean 时使用。
- **TopicConnectionFactoryClassName:** JMS 客户机的 `javax.jms.TopicConnectionFactory` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **XAConnectionFactoryClassName:** JMS 客户机的 `javax.jms.ConnectionFactory` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **XAQueueConnectionFactoryClassName:** JMS 客户机的 `javax.jms.XAQueueConnectionFactory` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **XATopicConnectionFactoryClassName:** JMS 客户机的 `javax.jms.XATopicConnectionFactory` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **UnifiedDestinationClassName:** JMS 客户机的 `javax.jms.Destination` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **QueueClassName:** JMS 客户机的 `javax.jms.Queue` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **TopicClassName:** JMS 客户机的 `javax.jms.Topic` 实现的类名称。在将 `ProviderIntegrationMode` 指定为 javabean 时使用。
- **ConnectionFactoryProperties:** 指定 JMS 客户机的 javabean 属性名称以及属性值。仅当 `ProviderIntegrationMode` 为 javabean 时才需要此属性。

5) 配置属性完成后，点击“保存”按钮；

启动 ActiveMQ, TongWeb7 与 ActiveMQ 的集成工作便完成。

12.4.2 与 TongLINKQ8.1 集成

配置前准备

已经正确安装配置了 TLQ8.1，以下示例中使用的 TLQ8.1 的主要配置如下：TLQ8.1 的 `jms broker` & `jndi broker` 的监听端口为 10024，并建立了一个接收队列 `lq1`，其 `jndi` 别名 `JndiQueueName` 为 `MyQueue`，假设 TLQ8 服务器 IP 为 168.1.15.135，TLQ 的详细配置过程参见 TongLINKQ8.1 用户手册。

本文下面提到的 `TongJMS_ra.rar` 位于 TLQ 安装目录 `...\\TLQ8\\java\\lib` 目录下。

集成 TLQ:

- 1) 启动 TongWeb7 服务器；
- 2) 打开 TongWeb7 的管理控制台，依次展开管理控制台左侧导航树中的“应用管理”节点；
- 3) 点击“部署”按钮，选择自己要使用的 TLQ 连接器应用（TongJMS_ra.rar），按照应用部署说明，开始部署连接器应用（Connector 应用）。部署完成后，展开管理控制台左侧导航树中的“应用管理”节点，然后点击刚刚部署成功的 Connector 应用，开始编辑属性；
- 4) 设置 ConnectionURL 属性值为：tlq:// 168.1.15.135:10024，点击“保存”按钮；

启动 TLQ8.1, TongWeb7 与 TLQ8.1 的集成工作便完成。

附：全部属性

ClientIDPrefix	<input type="text"/>
TransformerMBeanName	<input type="text"/>
Password	<input type="text"/>
ConfigurationTemplate	<input type="text"/>
MBeanObjectName	<input type="text"/>
UserName	<input type="text"/>
ConnectionURL	<input type="text"/>
ProjectInfo	<input type="text"/>
Options	<input type="text"/>
ConcurrencyMode	<input type="text"/>
ClearTextPassword	<input type="text"/>
EndpointPoolMaxSize	<input type="text"/>
ConfigurationJNDIName	<input type="text"/>
MBeanServerDomain	<input type="text"/>

12.4.3 查看/编辑集成属性

1. 依次展开管理控制台左侧导航树中的“应用管理”节点；
2. 点击已部署的 Connector 应用名称；
3. 点击“已设属性”查看/编辑已设置的集成属性，点击“全部属性”查看/编辑全部的集成属性；
4. 编辑属性完成后，点击“保存”按钮。

说明：用户可以在默认提供的集成配置中修改。也可以重新部署 genericra.rar 应用增加集成配置。

12.5 JMS 资源的使用

用户可通过连接工厂资源进行管理和使用 ConnectionFactory, 通过目的地资源进行

管理和使用 Destination。

12.5.1 创建连接工厂资源

1. 依次展开管理控制台左侧导航树中的“JMS 服务”节点，然后点击“连接工厂”节点；列表显示所有已创建的连接工厂。如图 12.5.1 所示：



图 12.5.1 连接工厂列表

2. 在出现的 JMS 连接工厂列表页中点击“创建连接工厂”按钮，出现如图 10.5.2 所示的创建 JMS 连接工厂页面：

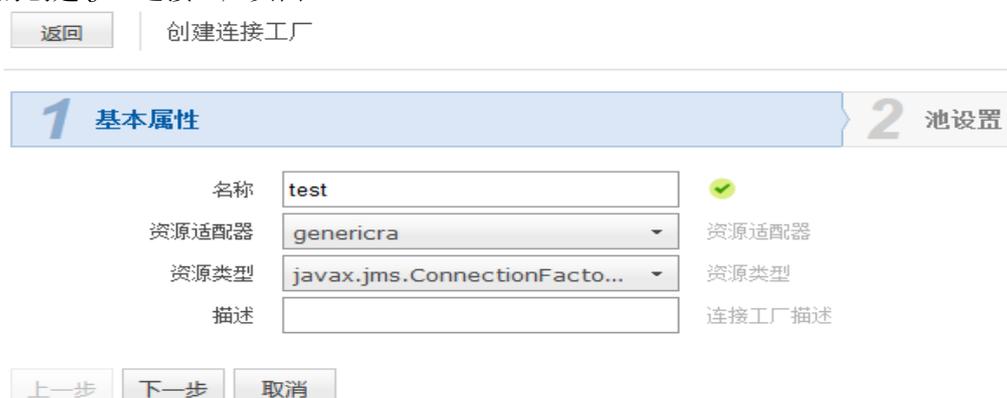


图 12.5.2 创建 JMS 连接工厂

点击“下一步”，进行“池设置”，如下图所示 12.5.3：



图 12.5.3 池设置页面

点击“下一步”，进行“其他属性”设置，如图：12.5.4 所示，点击“完成”，连接工厂创建成功。

此页显示了已创建的jms连接工厂，可以更新、删除所创建的连接工厂或者创建新的连接工厂

[返回](#) | [创建连接工厂](#)

1 基本属性		2 池设置		3 其他属性	
其他Property属性					
添加					
属性	值	操作			
ConnectionFactoryJndiName	test	删除			
ClientId		删除			
ConnectionValidationEnabled		删除			
上一步 完成 取消					

图 12.5.4 其他属性设置

- 基本设置

- 名称：连接工厂的 JNDI 名，连接工厂的唯一标识。

如上面例子的 jndi 名为 test

- 资源适配器：已部署的连接器应用的应用名称。默认为系统自带的通用适配器 genericra。可以通过下拉列表来更换所需要的连接器应用。
- 资源类型：连接工厂资源的类型，可选值为 `javax.jms.ConnectionFactory`、`javax.jms.QueueConnectionFactory`、`javax.jms.TopicConnectionFactory`。
- 描述：连接工厂资源的描述信息。

- 池设置

- 最小连接数：连接池启动后的初始连接数和最小连接数，默认值为 10。
- 最大连接数：连接池中连接数的最大值，默认值为 100。
- 等待超时时间：从连接池中获取连接时的最长等待时间，默认值为 60 秒。
- 空闲超时时间：连接在池中保持空闲的最长时间，一旦超过此时间，连接将会被从池中删除，默认值 10 分钟。
- 连接匹配：获取连接时由资源适配器进行匹配。默认不开启。
- 事务支持：连接池中连接支持的事务类型，可选值为 `NoTransaction`、`LocalTransaction` 和 `XATransaction`。默认值为 `NoTransaction`。

- 属性设置（与所选资源适配器有关）

genericra:

- ClientID：用于 JMS Server 识别消息订阅者身份的 ID。持久订阅时，客户端向 JMS 注册这个 ID，当这个客户端处于离线时，JMS Provider 会为此 ID 保存所有发送到主题的消息，当客户再次连接到 JMS Provider 时，会根据自己的 ID 得到所有当自己处于离线时发送到主题的消息。

- `ConnectionValidationEnabled`: 如果设置为 `true`, 资源适配器将使用异常侦听器捕捉任何连接异常, 并向应用服务器发送 `CONNECTION_ERROR_OCCURED` 事件。
- `ConnectionFactoryJndiName`: JMS 提供者的连接工厂对象绑定的 JNDI 名称。仅当集成方式为 JNDI 时才使用此属性。

tlq (TongJMS_ra) :

- `ConnectionURL`: 连接 URL, 指连接到 TLQ jms broker 的 URL。如:
tlq://168.1.15.135:10024
- `UserName`: 用户名。
- `Password`: 密码。
- `Options`: 选项。

说明:

如果 `genericra` 配置成支持 XA 事务, 那么连接工厂必须是 XA 的连接工厂;

如果 `genericra` 配置成不支持 XA 事务, 那么连接工厂必须是非 XA 的连接工厂;

当使用 XA 的连接工厂时, 那么客户端应用程序必须启用一个事务, 在事务中获取连接, 执行操作。

3. 设置好上述相应属性后, 点击“完成”按钮, 创建成功之后返回连接工厂列表, 在页面顶端提示创建成功信息。如果创建失败, 提示创建失败信息, 点击详情可以查看失败原因。如图 12.5.5 所示:



图 12.5.5 JMS 连接工厂创建返回列表

12.5.2 查看/编辑连接工厂资源

1. 依次展开管理控制台左侧导航树中的“JMS 服务”节点, 然后点击“连接工厂”节点;
2. 点击需要查看/编辑的连接工厂名, 页面显示如下图 12.5.6 所示:

返回 | 编辑连接工厂

名称 testc 唯一标识符，连接工厂JNDI名称

资源适配器 genericra 资源适配器

资源类型 javax.jms.ConnectionFacto... 资源类型

描述 连接工厂描述

* 最小连接数 10 连接池的初始化连接数

* 最大连接数 100 连接池中的最大连接数

* 等待超时 60 从连接池中获取连接的最长等待时间，单位为秒

* 空闲超时 10 连接的空闲超时时间，单位为分钟

连接匹配 开启 获取连接时由资源适配器进行匹配

事务支持 NoTransaction 连接池的事务支持类型

其他Property属性

添加

属性	值
ConnectionFactoryJndiNa	testc
ClientId	
ConnectionValidationEnab	

保存 取消

图 12. 5. 6 连接工厂编辑页面
属性同创建连接工厂资源的属性。

12. 5. 3 删除连接工厂资源

1. 展开管理控制台左侧导航树中的“JMS 服务”节点，然后点击“连接工厂”节点；
2. 选中要删除的 JMS 连接工厂；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

12. 5. 4 创建目的地资源

1. 依次展开管理控制台左侧导航树中的“JMS 服务”节点，然后点击“目的地”节点；列表显示所有已创建的目的地资源。如图 12. 5. 7 所示：

JMS目的地 管理目的地

此页显示了已创建的Jms目的地，可以更新、删除所创建的目的地或者创建新的目的地

创建目的地 删除 定制列表

名称	资源类型
testd	javax.jms.Queue

上一页 1 下一页

图 12. 5. 7 目的地列表
2. 在出现的 JMS 目的地列表页中点击“创建目的地”按钮，出现如图 12. 5. 8 所示的创建 JMS 目的地页面：

返回
创建目的地

名称	<input type="text" value="testdd"/>	✔	
资源适配器	<input type="text" value="genericra"/>		资源适配器
资源类型	<input type="text" value="javax.jms.Queue"/>		资源类型
描述	<input type="text"/>		目的地描述

其他Property属性

属性	值
<input type="text" value="DestinationJndiName"/>	<input type="text"/>
<input type="text" value="DestinationProperties"/>	<input type="text"/>

图 12.5.8 创建 JMS 目的地

- 属性设置
 - 名称：目的地的 JNDI 名，目的地的唯一标识。。
如上面例子的 jndi 名为 testdd
 - 资源适配器：已部署的连接器应用的应用名称。默认为系统自带的通用适配器 genericra。可以通过下拉列表来更换所需要的连接器应用。
附：内置 JMS Server 的连接器应用名为 JmsResourceAdapter。
 - 资源类型：目的地资源的类型，可选值为 javax. jms. Queue 和 javax. jms. Topic。
 - 描述：目的地资源的描述信息。
- 属性设置（与所选资源适配器有关）
 - genericra:
 - DestinationJndiName：JMS 提供者的目的地对象绑定的 JNDI 名称。仅当集成方式为 JNDI 时才使用此属性。
 - DestinationProperties：指定 JMS 客户机的 javabean 属性名称以及目的地的值，是以逗号分隔的名值对，如 PhysicalName=Send。仅当集成方式为 JavaBean 时才使用此属性。
 - tlq (TongJMS_ra) :
 - Name：JMS 提供者的目的地对象绑定的 JNDI 名称。仅当集成方式为 JNDI 时才使用此属性。目前 TLQ 默认自带为 MyQueue/ MyTopic。根据具体 tlq 环境配置而定。
 - Description：描述。

3. 设置好上述相应属性后，点击“保存”按钮，创建成功之后返回目的地列表，在页面顶端提示创建成功信息。如果创建失败，提示创建失败信息，点击详情可以查看失败原因。如图 12.5.9 所示：



图 12.5.9 JMS 目的地列表

12.5.5 查看/编辑目的地资源

1. 依次展开管理控制台左侧导航树中的“JMS 服务”节点，然后点击“目的地”节点；
2. 点击需要查看/编辑的目的地名，页面显示如下图 12.5.10 所示：

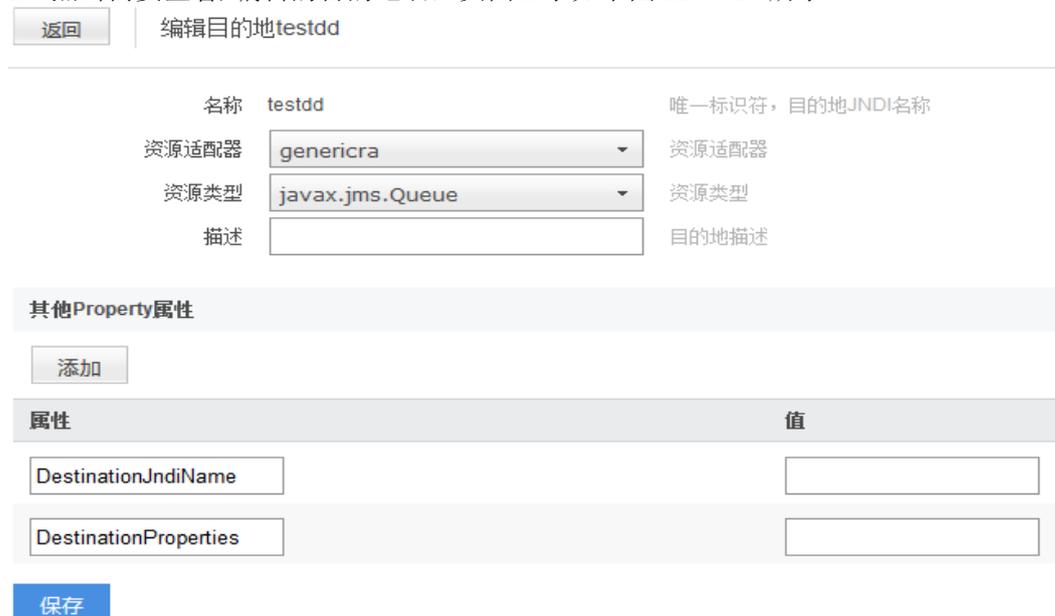


图 12.5.10 目的地编辑页面

属性同创建目的地资源的属性。

12.5.6 删除目的地资源

1. 展开管理控制台左侧导航树中的“JMS 服务”节点，然后点击“目的地”节点；
2. 选中要删除的 JMS 目的地；
3. 点击页面上方的“删除”按钮，在页面上方显示删除结果。

12.6 JMS 使用示例

示例一：使用 ActiveMQ——JNDI 集成方式（使用默认提供的名字服务）

示例目标：使用默认提供的 JMS Server ActiveMQ(JNDI 集成方式)，使用默认提供的名字服务(Sun 提供的 fscontext.jar)，使用 Web 应用向 ActiveMQ 的队列中发送消息，并通过 Web 应用接收消息。【用例详细可参见 `${tongweb.root}/samples/jms/JmsDemoRA.war`】

示例步骤如下：

1. 查看默认的集成配置。具体步骤见“12.5.2 查看/编辑集成属性”。详细信息如下图所示：

基本属性

应用名称	genericra	应用名称	
应用位置	D:\Work\tongweb-webprofile-1.5.0\applications/genericra	应用位置	
部署顺序	<input type="text" value="100"/>	默认的部署顺序是100，如果需要调整部署顺序的话，可以指定	
描述	<input type="text"/>	应用的描述信息	
线程池	<input type="text" value="default_thread_pool"/>	如果不选择，默认为default_thread_pool	

属性设置

属性名称	值
<input type="text" value="LogLevel"/>	<input type="text" value="INFO"/>
<input type="text" value="JndiProperties"/>	<input type="text" value="java.naming.factory.initial"/>
<input type="text" value="RMPolicy"/>	<input type="text" value="OnePerPhysicalConnect"/>
<input type="text" value="SupportsXA"/>	<input type="text" value="false"/>
<input type="text" value="ProviderIntegrationMode"/>	<input type="text" value="jndi"/>

需要配置的属性如下：

SupportsXA:false

RMPolicy:OnePerPhysicalConnection

LogLevel:INFO

ProviderIntegrationMode:jndi

JndiProperties:java.naming.factory.initial=com.sun.jndi.fscontext.ReffSContextFactory, java.naming.provider.url=\${tongweb.root}/apache-activemq/conf”

2. 启动 ActiveMQ，使用 JMS 命令行工具创建 ActiveMQ 对象，并将对象绑定到默认的名字服务。在 TongWeb7_HOME/bin/执行命令如下：

```
jmstool CF testc
```

jmstool Q testd

运行命令后，在 TongWeb7_HOME/apache-activemq/conf 下生成.bindings 文件，该文件中存放绑定对象的信息。注：如果使用 XA 事务，请使用 jmstool XACF 创建连接工厂。

注意：在 UNIX 或 linux 上使用 ActiveMQ 时，需保证机器的主机名是由大写字母（A-Z）、小写字母（a-z）、数字（0-9）、连接符（-）组成，不含其它特殊字母，否则可能引起 ActiveMQ 启动异常。

3. 创建连接工厂资源。具体步骤见“[12.5.1 创建连接工厂资源](#)”相关章节。

创建名称为 testc 的连接工厂资源，其中属性 ConnectionFactoryJndiName 的值为 testc(步骤 2 中通过 jmstool 创建的 ActiveMQ 连接工厂对象绑定到 JNDI 服务上的名字，此处为方便将名称与 ActiveMQ 连接工厂对象 jndi 名设成一样的)，具体信息如下图：

返回编辑连接工厂

名称	testc	唯一标识符，连接工厂 JNDI 名称
资源适配器	genericra	资源适配器
资源类型	javax.jms.ConnectionFacto...	资源类型
描述		连接工厂描述
* 最小连接数	10	连接池的初始化连接数
* 最大连接数	100	连接池中的最大连接数
* 等待超时	60	从连接池中获取连接的最长等待时间，单位为秒
* 空闲超时	600	连接的空闲超时时间，单位为秒
连接匹配	<input checked="" type="checkbox"/> 开启	获取连接时由资源适配器进行匹配
事务支持	NoTransaction	连接池的事务支持类型

其他Property属性

属性	值
ConnectionFactoryJndiName	testc
ClientId	
ConnectionValidationEnab	

4. 创建目的地资源。具体步骤见“[12.5.4 创建目的地资源](#)”相关章节。

创建名称为 testd 的目的地资源，其中属性 DestinationJndiName 的值为 testd(步骤 2 中通过 jmstool 创建的 ActiveMQ Queue 对象绑定到 JNDI 服务上的名字，此处为方便将名称与 ActiveMQ Queue 对象 jndi 名设成一样的)。具体信息如下图：

JMS目的地 管理目的地

此页显示了已创建的jms目的地，可以更新、删除所创建的目的地或者创建新的目的地

编辑目的地testd

名称	testd	唯一标识符，目的地JNDI名称
资源适配器	<input type="text" value="genericra"/>	资源适配器
资源类型	<input type="text" value="javax.jms.Queue"/>	资源类型
描述	<input type="text"/>	目的地描述

其他Property属性

属性	值	操作
<input type="text" value="DestinationJndiName"/>	<input type="text" value="testd"/>	<input type="button" value="删除"/>
<input type="text" value="DestinationProperties"/>	<input type="text"/>	<input type="button" value="删除"/>

5. 使用 web 应用收发消息。

发送消息端代码：

```
Context ctx = new InitialContext();
QueueConnectionFactory factory = (QueueConnectionFactory) ctx.lookup("testc");
Queue queue = (Queue) ctx.lookup("testd");
QueueConnection connection = factory.createQueueConnection();
QueueSession session = connection.createQueueSession(false, 1);
QueueSender sender = session.createSender(queue);
TextMessage msg = session.createTextMessage("This is the test message!");
sender.send(msg);
```

接受消息端代码：

```
Context ctx = new InitialContext();
QueueConnectionFactory factory = (QueueConnectionFactory) ctx.lookup("testc");
Queue queue = (Queue) ctx.lookup("testd");
QueueConnection connection = factory.createQueueConnection();
connection.start();
QueueSession session = connection.createQueueSession(false, 1);
QueueReceiver receiver = session.createReceiver(queue);
TextMessage msg = receiver.receive(1000L);
if ((msg instanceof TextMessage)) {
    messageText = ((TextMessage)msg).getText();
    System.out.println("message content:" + messageText);
}
```

示例二：使用 ActiveMQ——JavaBean 集成方式

示例目标：使用默认提供的 JMS Server ActiveMQ(JavaBean 集成方式)，使用 Web 应用向 ActiveMQ 的队列中收发消息。

示例步骤如下：

1. 查看默认的配置集成。具体步骤见“12.5.2 查看/编辑集成属性”。点击查看全部属性。详细信息如下图：

应用名称	genericra	应用名称
应用位置	D:\Work\ongweb-webprofile-1.5.0\applications/genericra	应用位置
部署顺序	<input type="text" value="100"/>	默认的部署顺序是100, 如果需要调整部署顺序的话, 可以指定
描述	<input type="text"/>	应用的描述信息
线程池	<input type="text" value="default_thread_pool"/>	如果不选择, 默认为default_thread_pool

属性设置

已设属性

属性名称	值
LogLevel	INFO
CommonSetterMethodName	
DeliveryConcurrencyMode	
QueueConnectionFactoryClassName	
TopicClassName	
UseFirstXAForRedelivery	
Password	
UserName	
ConnectionFactoryClassName	
ConnectionFactoryProperties	
JndiProperties	java.naming.factory.initial
RMPolicy	OnePerPhysicalConnection
XAConnectionFactoryClassName	
SupportsXA	false
UnifiedDestinationClassName	
MDBDeploymentRetryInterval	
QueueClassName	
XAQueueConnectionFactoryClassName	
DeliveryType	
XATopicConnectionFactoryClassName	
MDBDeploymentRetryAttempt	
ProviderIntegrationMode	jndi
Monitoring	
TopicConnectionFactoryClassName	

保存

需要配置的属性如下:

SupportsXA: false

RMPolicy: OnePerPhysicalConnection

LogLevel: INFO

```
ProviderIntegrationMode: javabean
ConnectionFactoryClassName: com.tongweb.activemq.ActiveMQConnectionFactory
QueueConnectionFactoryClassName:
com.tongweb.activemq.ActiveMQConnectionFactory
TopicConnectionFactoryClassName:
com.tongweb.activemq.ActiveMQConnectionFactory
XAConnectionFactoryClassName:
com.tongweb.activemq.ActiveMQXAConnectionFactory
XAQueueConnectionFactoryClassName:
com.tongweb.activemq.ActiveMQXAConnectionFactory
XATopicConnectionFactoryClassName:
com.tongweb.activemq.ActiveMQXAConnectionFactory
UnifiedDestinationClassName: com.tongweb.activemq.command.ActiveMQDestination
QueueClassName : com.tongweb.activemq.command.ActiveMQQueue
TopicClassName: com.tongweb.activemq.command.ActiveMQTopic
ConnectionFactoryProperties: BrokerURL=tcp://127.0.0.1:61616
```

2. 启动 ActiveMQ，使用 JMS 命令行工具创建 ActiveMQ 对象，并将对象绑定到默认的名字服务。在 TongWeb7_HOME/bin/执行命令如下：

```
jmstool Q testd
```

运行命令后，在 TongWeb7_HOME/apache-activemq/conf 下生成.bindings 文件，该文件中存放绑定对象的信息。

注意：在 UNIX 或 linux 上使用 ActiveMQ 时，需保证机器的主机名是由大写字母（A-Z）、小写字母（a-z）、数字（0-9）、连接符（-）组成，不含其它特殊字母，否则可能引起 ActiveMQ 启动异常。

3. 创建连接工厂资源。具体步骤见 “[12.5.1 创建连接工厂资源](#)” 相关章节。创建名称为 testc 的连接工厂资源，其中属性部分不用设置，具体信息如下图：

名称	testc	唯一标识符，连接工厂JNDI名称
资源适配器	genericra	资源适配器
资源类型	javax.jms.ConnectionFacto...	资源类型
描述		连接工厂描述
* 最小连接数	10	连接池的初始化连接数
* 最大连接数	100	连接池中的最大连接数
* 等待超时	60	从连接池中获取连接的最长等待时间，单位为秒
* 空闲超时	10	连接的空闲超时时间，单位为分钟
连接匹配	<input type="checkbox"/> 开启	获取连接时由资源适配器进行匹配
事务支持	NoTransaction	连接池的事务支持类型

其他Property属性

添加

属性	值
ConnectionFactoryJndiName	
ClientId	
ConnectionValidationEnab	

保存 取消

4. 创建目的地资源。具体步骤见“12.5.4 创建目的地资源”相关章节。

创建名称为 testd 的目的地资源，其中属性 DestinationProperties 的值为 PhysicalName=testd（步骤 2 中通过 jmstool 创建的 ActiveMQ Queue 对象绑定到 JNDI 服务上的名字，此处为方便将名称与 ActiveMQ Queue 对象 jndi 名设成一样的）。具体信息如下图：

JMS目的地 管理目的地 ⓘ

此页显示了已创建的jms目的地，可以更新、删除所创建的目的地或者创建新的目的地

返回 | 编辑目的地testd

名称	testd	唯一标识符，目的地JNDI名称
资源适配器	genericra	资源适配器
资源类型	javax.jms.Queue	资源类型
描述		目的地描述

其他Property属性

添加

属性	值	操作
DestinationJndiName		删除
DestinationProperties	PhysicalName=testd	删除

保存

5. 使用 web 应用收发消息。同 10.6.1 中 5 使用 web 应用收发消息。

示例三：使用 TongLINK/Q8.1

示例目标：使用 TongLINK/Q8.1 (JNDI 集成方式)，使用 TongLINK/Q 自带的名字服务，使用 Web 应用向 TongLINK/Q 的队列中收发消息。

示例步骤如下：

1. 远程部署，选中 TLQ 的 TongJMS_ra.rar 资源适配器，TongJMS_ra.rar 保存在 ...\TLQ8\java\lib 目录下，并设置 ConnectionURL 属性值为：tlq://168.1.15.135:10024。详细信息如下图所示：

2. 创建连接工厂资源。具体步骤见“[12.5.1 创建连接工厂资源](#)”相关章节。

创建连接工厂，如果需要用到的远程连接工厂，名称为 RemoteQueueConnectionFactory
如果测试本地方式，连接工厂名称为：LocalQueueConnectionFactory，并设置属性：
ConnectionURL 为 tlq://168.1.15.135:10024。详细信息如下图所示：

3. 创建目的地资源。具体步骤见“[12.5.4 创建目的地资源](#)”相关章节。

创建名称为 lq1 的目的地资源，其中属性 Name 的值为 MyQueue（与 tlq 中 tlqjndi.conf 的配置保持一致）。具体信息如下图：

此页显示了已创建的jms目的地，可以更新、删除所创建的目的地或者创建新的目的地

[返回](#) | [编辑目的地lq1](#)

名称	lq1	唯一标识符，目的地JNDI名称
资源适配器	TongJMS_ra	资源适配器
资源类型	javax.jms.Queue	资源类型
描述		目的地描述

其他Property属性

[添加](#)

属性	值	操作
Name	MyQueue	删除

[保存](#)

4. 使用 web 应用收发消息。

13 TongWeb 域

13.1 概述

TongWeb 域功能，其定义为逻辑服务器管理，通过安装介质首次安装的服务器为物理服务器，通过物理服务器的域功能可创建出多个逻辑服务器，这些逻辑服务器各自的配置信息、日志文件等私有属性保存在与其对应的各个域中，目前定义为一个域只管理一个逻辑服务器，逻辑服务器依赖的公有属性（如 lib 文件、license 文件、系统应用、Agent、apache-activemq、Hazelcas、samplest 等）都引自物理服务器（域中不含这些物理文件），基于域功能，只需要安装一份物理 TongWeb 就可以创建多个 TognWeb 实例。

域管理的逻辑服务器在功能上和物理服务器相比裁剪掉了集中管理工具（heimdall），其它的功能完全一致。

13.2 创建 TongWeb 域

物理 TongWeb 提供创建域脚脚本，创建时需要指定一个名字，创建后在物理 TW_HOME/domains 目录下会生成一个以该名字命名的目录，该目录就是一个逻辑 TongWeb 服务器，这种域成为“相对域”，创建时也可以指定一个绝对路径用以保存域文件，这种域称为“绝对域”，“相对域”在物理 TongWeb 路径变化后不用任何修改仍可使用，“绝对域”在物理 TongWeb 路径变化后需手动修改其相关脚本更新为新的物理 TongWeb 路径。

以 Linux 平台为例，创建名称为“tw_domain_1”的相对域，在 TW_HOME/bin 目录下运行 ./domain.sh create tw_domain_1，若要创建绝对路径为 /opt/tw_domain_1 的绝对域，则运行 ./domain.sh create /opt/tw_domain_1。

13.3 删除 TongWeb 域

物理 TongWeb 提供删除域脚脚本，删除“相对域”时候需要指定域的名字（即物理 TW_HOME/domains 下的文件夹名称）即可，删除“绝对域”则需要制定其绝对路径。

以 Linux 平台为例，删除名称为“tw_domain_1”的相对域，在 TW_HOME/bin 目录下运行 ./domain.sh delete tw_domain_1，若要删除绝对路径为 /opt/tw_domain_1 的绝对域，则运行 ./domain.sh delete /opt/tw_domain_1。

13.4 启动 TongWeb 域

物理 TongWeb 提供启动域脚脚本，启动“相对域”时候需要指定域的名字，启动“绝对域”则需要制定其绝对路径，同时域本身的 bin 目录下也提供了其启动脚本，可

直接使用，使用时不需要指定任何名字或路径。

以 Linux 平台为例，要启动名称为“tw_domain_1”的相对域，在 TW_HOME/bin 目录下运行 ./startdomain.sh tw_domain_1，若要启动绝对路径为 /opt/tw_domain_1 的绝对域，则运行 ./startdomain.sh /opt/tw_domain_1。当然，也可以在逻辑 TongWeb 的 bin 目录下运行 ./startserver.sh 启动该服务器。

13.5 停止 TongWeb 域

物理 TongWeb 提供停止域脚脚本，停止“相对域”时候需要指定域的名字，停止“绝对域”则需要制定其绝对路径，同时域本身的 bin 目录下也提供了停止动脚本，可直接使用，使用时不需要指定任何名字或路径。

以 Linux 平台为例，要停止名称为“tw_domain_1”的相对域，在 TW_HOME/bin 目录下运行 ./stopdomain.sh tw_domain_1，若要停止绝对路径为 /opt/tw_domain_1 的绝对域，则运行 ./stopdomain.sh /opt/tw_domain_1。当然，也可以在逻辑 TongWeb 的 bin 目录下运行 ./stopserver.sh 停止该服务器。

13.6 注意事项

由于新创建出来的域运行 TongWeb 实例依赖物理 TongWeb，所以域的位置必需和物理 TongWeb 必需处于同一文件系统中，且对于绝对路径创建的域若后期物理 TongWeb 路径发生了变化，必需手动修改域的启动脚本指向新的物理 TongWeb。

14 安全防护

在 [7.2 安全服务](#) 中，TongWeb7 支持控制台登录失败约束、支持三员分立、安全审计和审计日志、支持传输层安全保护、支持 JAAS 和 JACC 外，在网络信息安全服务方式，还支持以下各种方式：

14.1 HttpOnly 支持

如果 cookie 中设置了 HttpOnly 属性，那么通过 js 脚本将无法读取到 cookie 信息，这样能有效的防止 XSS 攻击，窃取 cookie 内容，这样就增加了 cookie 的安全性。

TongWeb 支持在应用的 web.xml 配置 httpOnly 属性，防止 cookie 被劫持。配置参数如下：

```
<session-config>
  <cookie-config>
    <http-only>true</http-only>
  </cookie-config>
</session-config>
```

http-only 值设置 true，表示不能通过客户端脚本访问 cookie；设置为 false，表示可以被访问到。

14.2 HTTP trace 支持

TRACE 方法是 HTTP（超文本传输）协议定义的一种协议调试方法，该方法使得服务器原样返回任何客户端请求的内容。

14.2.1 启用 trace 功能

新建 http 通道，“禁用 HTTP 请求方法”勾选 TRACE。通道创建具体参考 [4.4.2.1](#)

其他设置

禁用HTTP请求方法	TRACE	要禁用的HTTP请求方法
上传超时时间	请选择	超时时间，以毫秒为单位
URL编码格式	<input checked="" type="checkbox"/> TRACE	用于解码URI字符的编码格式，默认GBK
parse-body-methods	<input type="checkbox"/> OPTIONS	用于rest，默认支持POST
POST请求最大字节数	<input type="checkbox"/> HEAD	POST请求允许的最大字节数，其中-1代表没有限制
uri处理	<input type="checkbox"/> CONNECT	如果ContentType中指定了编码规范，则可以不使用URL编码格式
虚拟主机	<input type="checkbox"/> DELETE	基于IP的虚拟主机
DNS反向查找	<input type="checkbox"/> PUT	DNS反向查找
Referer头验证	<input type="checkbox"/> 开启	开启验证HTTP Referer请求头，不被允许的Referer请求将被禁止

14.2.2 禁用 trace 功能

点击需要编辑的通道，去掉勾选的 TRACE 方法。

其他设置

禁用HTTP请求方法	请选择	要禁用的HTTP请求方法
上传超时时间	请选择	超时时间，以毫秒为单位
URL编码格式	<input type="checkbox"/> TRACE	用于解码URI字符的编码格式，默认GBK
parse-body-methods	<input type="checkbox"/> OPTIONS	用于rest，默认支持POST
POST请求最大字节数	<input type="checkbox"/> HEAD	POST请求允许的最大字节数，其中-1代表没有限制
uri处理	<input type="checkbox"/> CONNECT	如果ContentType中指定了编码规范，则可以不使用URL编码格式
虚拟主机	<input type="checkbox"/> DELETE	基于IP的虚拟主机
DNS反向查找	<input type="checkbox"/> PUT	DNS反向查找
Referer头验证	<input type="checkbox"/> 开启	开启验证HTTP Referer请求头，不被允许的Referer请求将被禁止

14.3 X-Frame-Options

防止 WEB 应用页面被其他应用 frame 嵌套，可以通过在 web.xml 中配置 filter 的方式来支持此功能。

在 WEB 应用的 web.xml 中加入 filter 配置

```

<filter>
  <filter-name>securityFilter</filter-name>
  <filter-class>com.tongweb.catalina.filters.HttpHeaderSecurityFilter</filter-
class>
  <init-param>
    <param-name>antiClickJackingEnabled</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>antiClickJackingOption</param-name>
    <param-value>DENY</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>securityFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

```

其中 antiClickJackingOption 参数有三种：

- a) DENY: 浏览器拒绝当前页面加载任何 Frame 页面。
- b) SAMEORIGIN: 页面只能加载入同源域名下的页面。
- c) ALLOW-FROM: 只能被嵌入到指定域名的框架中。(部分浏览器不支持)

详细解释:

过滤器类别名称

HTTP 标头安全过滤器的过滤器类名称为:

com.tongweb.catalina.filters.HttpHeaderSecurityFilter

初始化参数

HTTP 标头安全过滤器支持以下初始化参数:

属性	描述
antiClickJackingEnabled	应使用 X-Frame-Options 在响应中设置防点击劫持标头。任何已经存在的防点击劫持标头都将被替换。如果未指定, 默认 true。
antiClickJackingOption	取值范围是 DENY, SAMEORIGIN, ALLOW-FROM (不区分大小写)。如果未指定, 默认 DENY。
antiClickJackingUri	如果 ALLOW-FROM 用于 antiClickJackingOption, 则表示应允许使用哪个 URI。如果未指定, 将使用空字符串的默认值。

目前【SAMEORIGIN】使用场景只支持火狐浏览器。

14.4 防 SDOS 攻击 (慢攻击)

SDOS 攻击 (慢攻击) 在众多网络攻击技术中是一种简单有效并且具有很大危害性的攻击方法。它通过各种手段消耗网络带宽和系统资源, 或者攻击系统缺陷, 使正常系统的正常服务陷于瘫痪状态, 不能对正常用户进行服务, 从而实现拒绝正常的用户访问服务。TongWeb7 通过支持慢攻击检测、慢攻击容忍次数、黑名单移除时间等来防止 SDOS 攻击。

页面配置参照 [4.1.1 Web 容器配置](#) http 慢攻击检测

同时也可以 在 tongweb.xml 的 tongweb/server/web-container/property 节点中配置如下参数:

属性名称	对应 xml 配置属性	属性说明	是否实时生效
完整请求时间	complete.message.timeout.seconds	完整请求时间, 通过配置完整请求时间来判断是否为慢攻击, 时间单位为秒, 如果配置值大于 0, 则认为开启慢攻击检测, 默认值为 0, 关闭慢攻击检测	否, 重启服务器生效
慢攻击容忍次数	max.attack.times	慢攻击容忍次数, 为防止处理掉正常请求, 支持配置容忍次数, 同一个 IP 地址超过配置数, 则加入黑名单, 默认配置为 3	否, 重启服务器生效
黑名单移除时间	blacklist.expire.d.hours	黑名单移除时间, 通过配置时间来清除已经加入到黑名单中的 IP 地址, 时间单位为小时, 默认配置为 12 小时	否, 重启服务器生效
中断当前连接	interrupt.current.connect	中断当前连接, 布尔类型, 配置为 true 表示如果当前发送消息的连接处于黑名单中, 则中断, 默认配置为 true	否, 重启服务器生效

14.5 支持国密算法的负载均衡软件 THS

TongWeb7 应用服务器内置提供了软件负载均衡器 TongHttpServer（简称 THS），这是一款功能强大、稳定高效、高性价比、易于使用、便于维护的负载均衡软件产品。THS 不仅可以满足用户对负载均衡服务的需求，提升系统可靠性、高效性、可扩展性及资源利用率，还具有很高的性价比，可以有效降低系统的建设成本、维护成本，并且使用简单、维护便捷。同时 THS 的“智能化”特点能够极大地提升系统的运维效率、减少运维工作量、降低因误操作引发事故的几率。

该 THS 软件包含如下核心功能：

- 支持 OSI 七层负载均衡功能；支持 HTTP、HTTPS、AJP 等协议；
- 支持多种 OSI 负载均衡常用功能，如会话保持、访问控制等；
- 支持多种负载均衡策略；支持 HA 功能、集群功能，支持浮动 IP；
- 支持 TCP、GET、HEAD 三种服务器健康检查方式；
- 提供安全防护功能，包括访问控制、防御慢查询攻击；
- 支持国密通信 GMTLS1.1 通信协议及 ECC-SM3-SM4 密码套件；
- 提供管理控制台，以及管理 API 支持第三方工具对程序进行管理维护。

具体用法参考《TongHttpServer 用户手册》

15 工具使用指南

15.1 Eclipse 中 TongWeb7 插件

为了满足用户能在 Eclipse 中快速开发应用程序，并部署应用到 TongWeb7 上,我们开发了 Eclipse 插件工具。

15.1.1 功能概述

Eclipse 插件的功能主要包括在 Eclipse 中对 TongWeb7 应用服务器添加、配置修改、启动和停止、应用部署、调试等功能。

15.1.2 安装 Eclipse

本手册基于 eclipse-jee-2019-09-R-win32-x86_64 版本编写，其他版本相应描述位置可能会略有不同。下载地址为 <https://www.eclipse.org/downloads/packages/>。

Eclipse 的官网上下载 eclipse-jee-2019-09-R-win32-x86_64 版本或 eclipse-java-juno-win32-x86_64 均可，下载完成后，解压即可。

15.1.3 安装 Eclipse 插件

将\${TW7_HOME}/tools/eclipse-plugins 下的两个 jar 文件，拷贝至 eclipse_home/plugins 目录下，重启 Eclipse 开发工具即可。

- com.tongweb.jst.server.tongweb.core-1.0.jar
- com.tongweb.jst.server.tongweb.ui-1.0.jar

15.1.4 Eclipse 插件的使用

安装完成后，重启 Eclipse 就可以使用插件了，下面将介绍 TongWeb7 Eclipse 插件的具体功能使用。

15.1.4.1 添加 TongWeb7 服务器

- 通过 Preferences 中添加

1) 在 Eclipse 中，打开“Window”->“Preferences”菜单，弹出“Preferences”窗口，点击“Server”->“Runtime Environment”，如下图 15.1.1 所示：

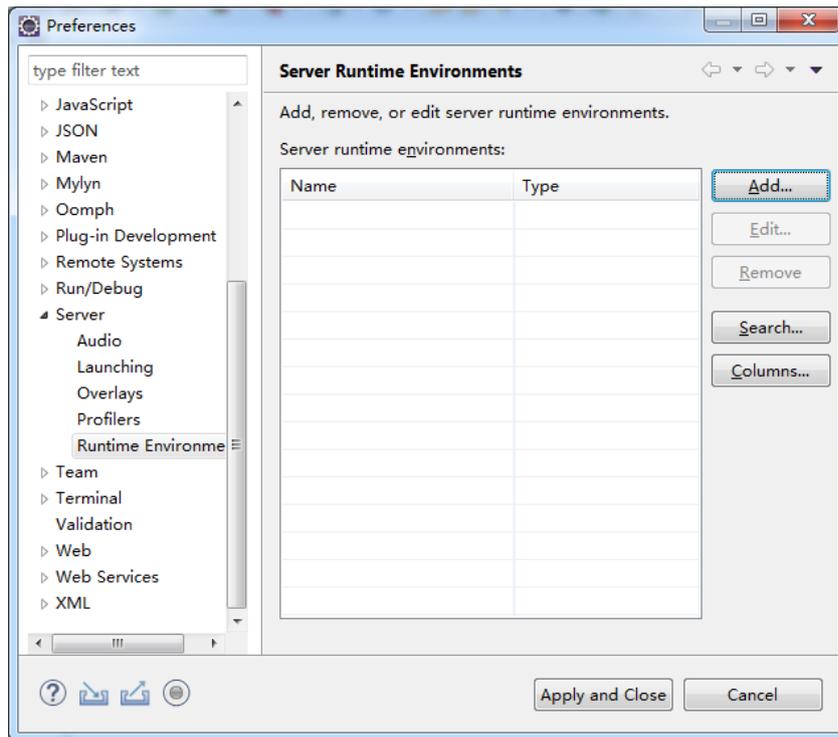


图 15.1.1 添加 Server 界面

- 2) 点击界面中的“Add”按钮，在弹出的新建对话框中，选择 TongWeb 服务器，可以增加一个服务器配置，如下图 15.1.2 所示：

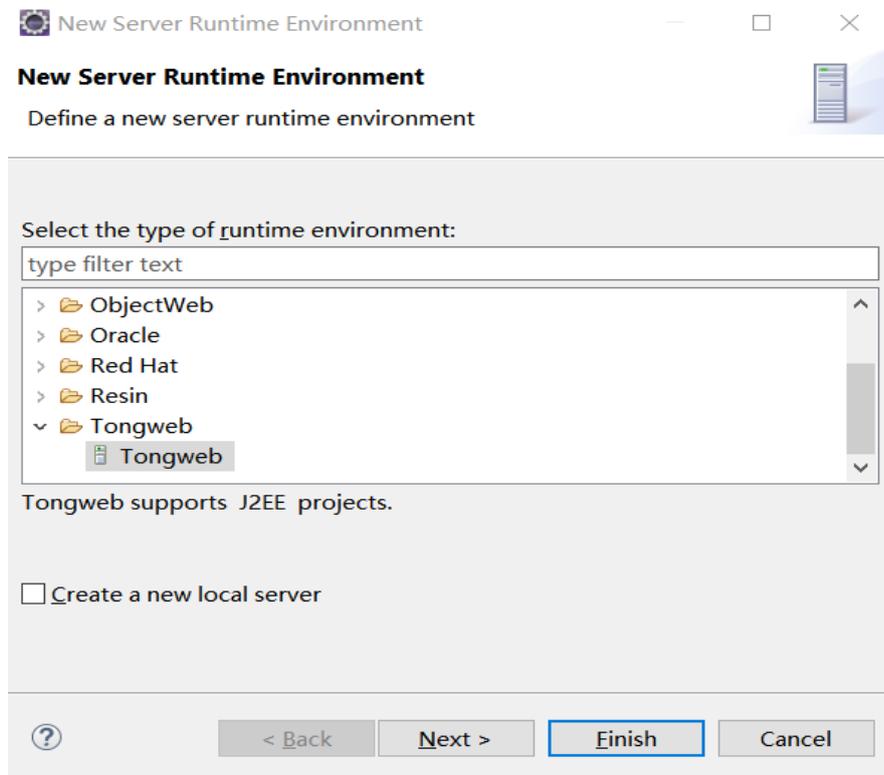


图 15.1.2 选择 TongWeb 服务器

- 3) 若选中界面中的 Create a new local server 选项，则在服务器配置完成后，将同时生成一个服务器运行时实例，可以在 Server 视图中查看到。在界面中选择“Next”，配置 TongWeb7 服务器基本信息，即输入名称和 TongWeb7 服务器的路径，如下图

15.1.3 所示:

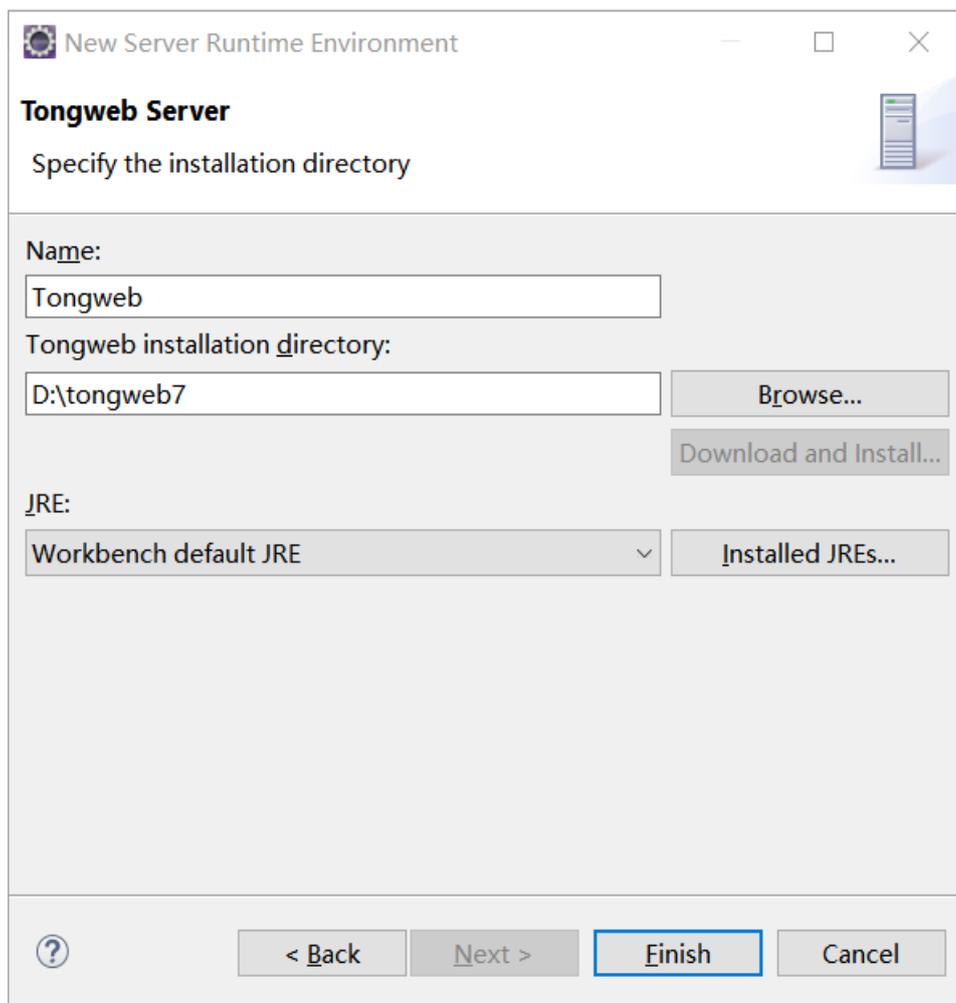


图 15.1.3 添加 TongWeb7 服务器路径

- 4) 点击“Finish”按钮，即 TongWeb7 应用服务器在 Eclipse 中添加成功；
- 5) 创建完成后，选择新建的服务器“TongWeb”，点击“Edit”，可以对该服务器进行编辑；
- 6) 选择新建的服务器“TongWeb”，点击“Remove”，可以删除已添加的服务器，如下图 15.1.3 所示：

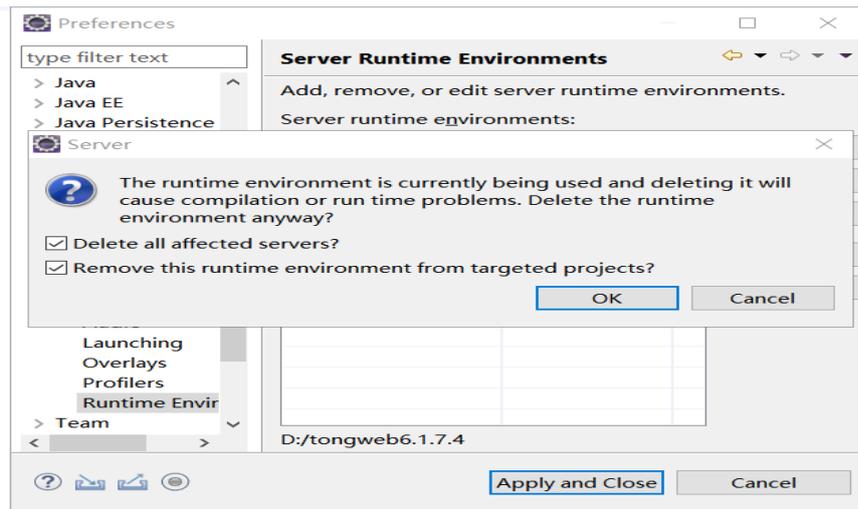


图 15.1.4 删除 TongWeb 服务器

当存在已有应用部署到 TongWeb7 服务器上时，点击“Remove”，如出现提示信息。选中“Delete all effected servers”复选框，则连同已引用该配置的服务实例一同删除，否则只删除服务器配置信息

● 在 Server 中添加

- 1) 通过菜单 “Windows” -> “Show View” -> “Server” 打开 Server 视图。右键单击 Server 图空白处，选择 New->Server，进行添加，如下图 15.1.5 所示：

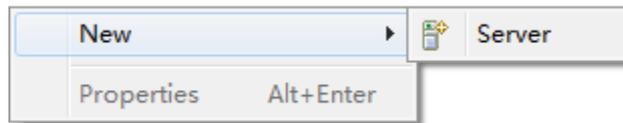


图 15.1.5 添加 Server

- 2) 如果 “Preferences” 中已添加过服务器配置，用户可在界面中选择一个已有的服务器配置。如下图 15.1.6 所示，点击 “Finish” 完成。

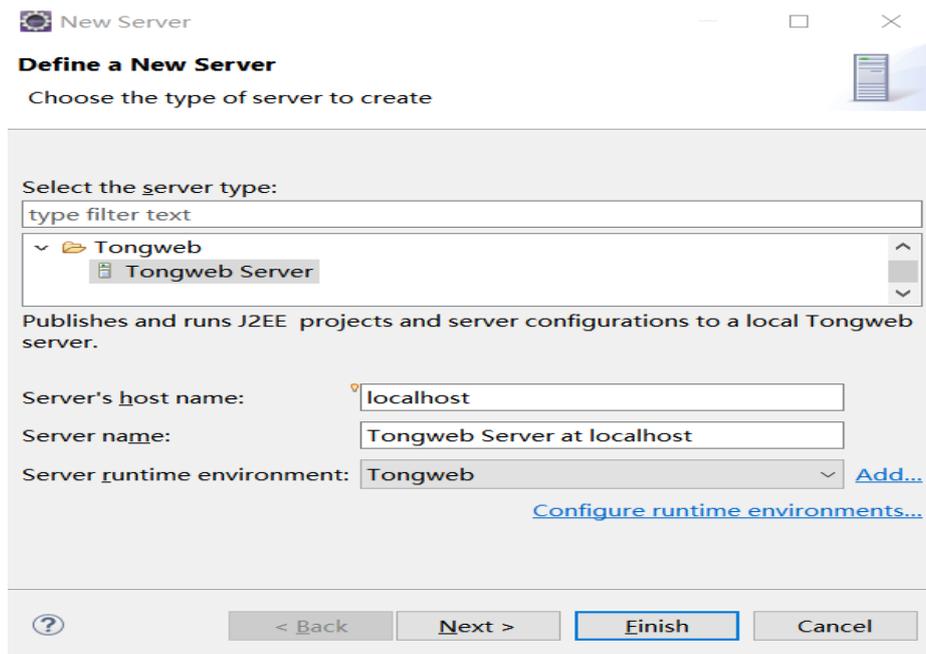


图 15.1.6 添加已有的 Server

- 3) 如果没有在“Preferences”中添加过服务器配置，此时用户可在后续菜单中指定 Server runtime environment 的配置信息和 TongWeb7 的安装目录。

15.1.4.2 修改服务器实例配置

在 Server 视图中，双击已添加的 TongWeb7 服务器实例配置，可打开 Over View 配置界面，对 TongWeb7 服务器实例的运行配置进行编辑。如 15.1.7 所示：

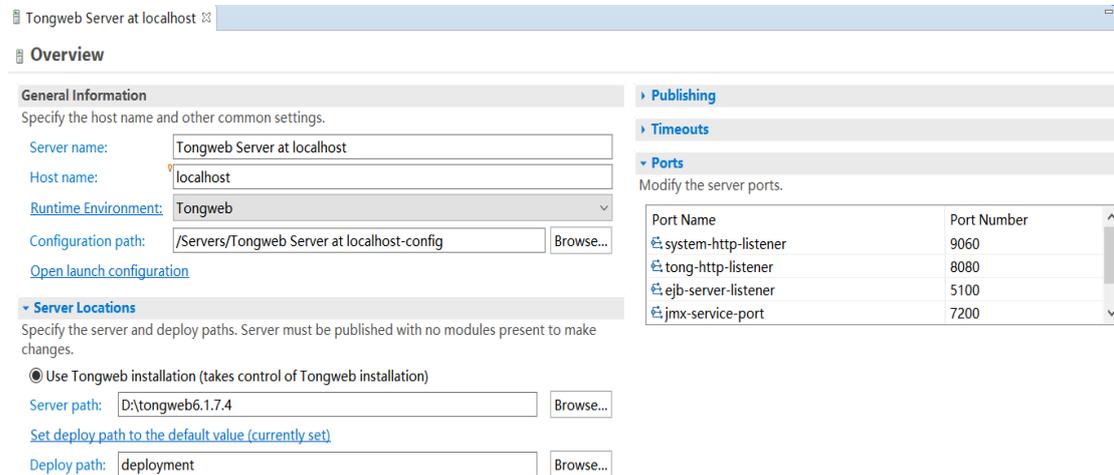


图 15.1.7 TongWeb7 服务器编辑界面

配置说明：

General Information:

- Server Name: 服务器名称;
- Host Name: 主机名，默认为 localhost;
- Runtime Environment: 运行时配置名;
- Configuration Path: 主要配置文件保存路径;

Server Locations

- Use Tongweb installation(使用 TongWeb7 的安装目录);
- Server path: 为当前选择 Runtime Environment 配置对应的安装路径。当没有应用部署时，可以修改为其他已安装 TongWeb7 的目录，否则不可修改;
- Deploy path: 默认的部署目录;

Ports

- Ports: 通过 Ports，用户可修改的 TongWeb7 服务器端口;

15.1.4.3 添加和移除工程

- 1) 在“Server”视图中，选择要添加工程的 TongWeb7 服务器实例，点击右键菜单，在弹出的快捷菜单中选择“Add and Remove”菜单，如 15.1.8 所示，弹出“Add and Remove”界面，可以添加、删除工程到服务器实例中。

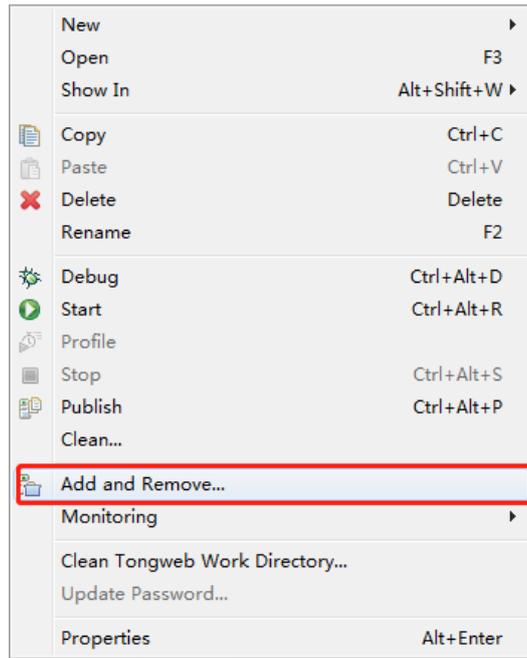


图 15.1.8 添加删除工程实例

- 2) 工程添加至服务器后，需要 Publish 到服务器实例，工程才会被正式部署至服务器的运行环境中。在 Server 视图，选择要执行 publish 操作的 TongWeb7 服务器，点击右键菜单，进行 publish 操作，右键菜单如下图所示：

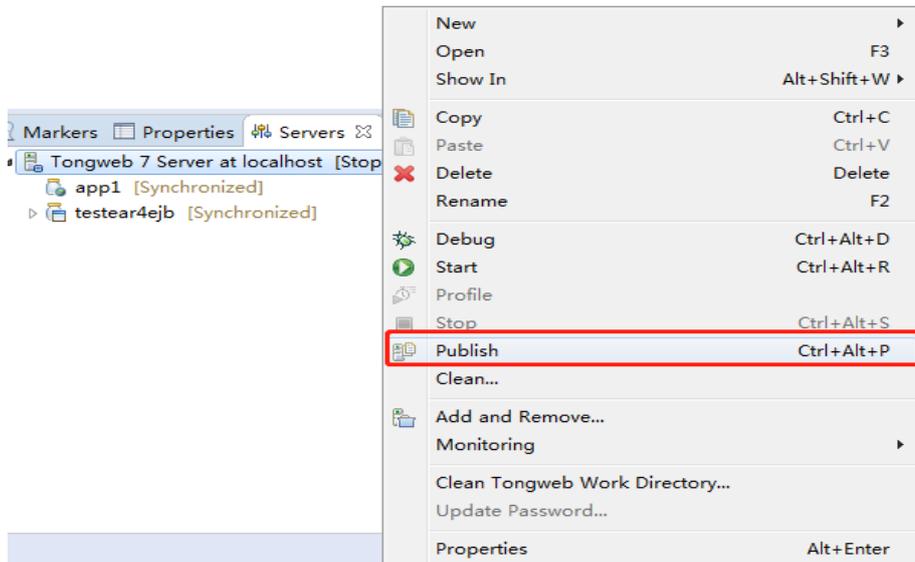


图 15.1.9 发布工程

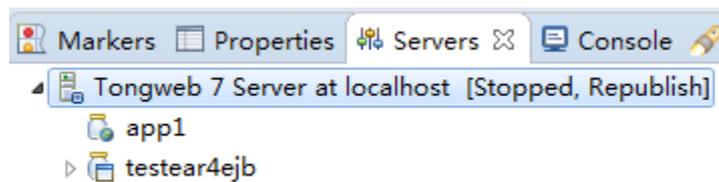


图 15.1.10 发布前应用状态

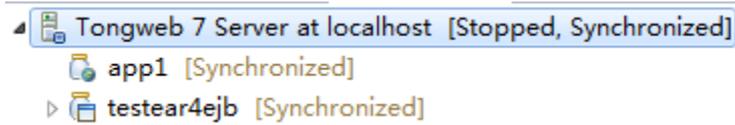


图 15.1.11 发布后应用状态

15.1.4.4 启动和停止服务器

Server 视图，选择要启动的 TongWeb7 服务器通过右键菜，进行启动和停止服务器操作。

- 启动服务器

通过点击右键菜单 Start 按钮，可启动服务器，同一时间只支持启动一个 TongWeb7 实例。如下图 15.1.12 所示：

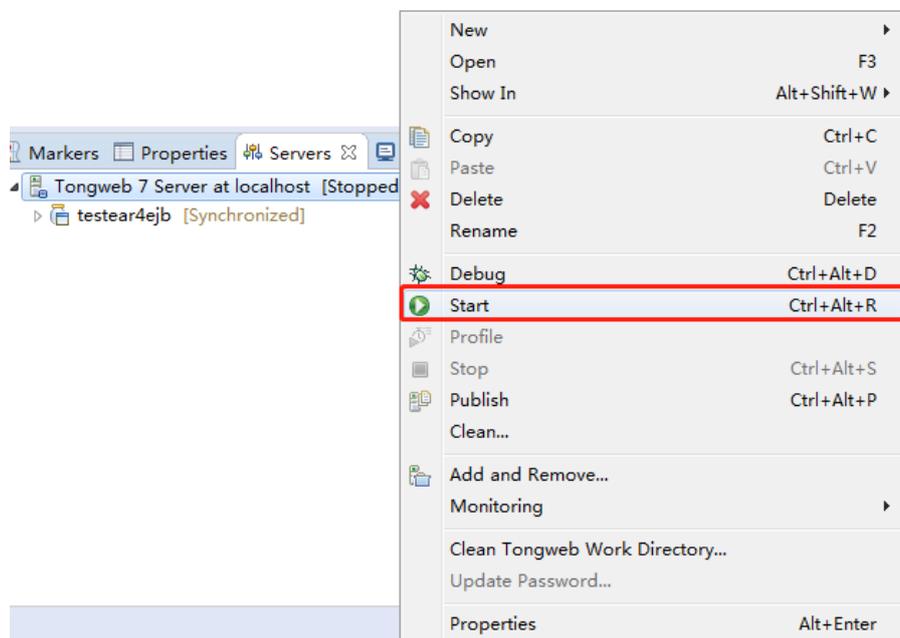


图 15.1.12 启动服务器

- 停止服务器

Server 视图中，通过点击右键 Stop 按钮，可停止一个已运行的 TongWeb7 服务器

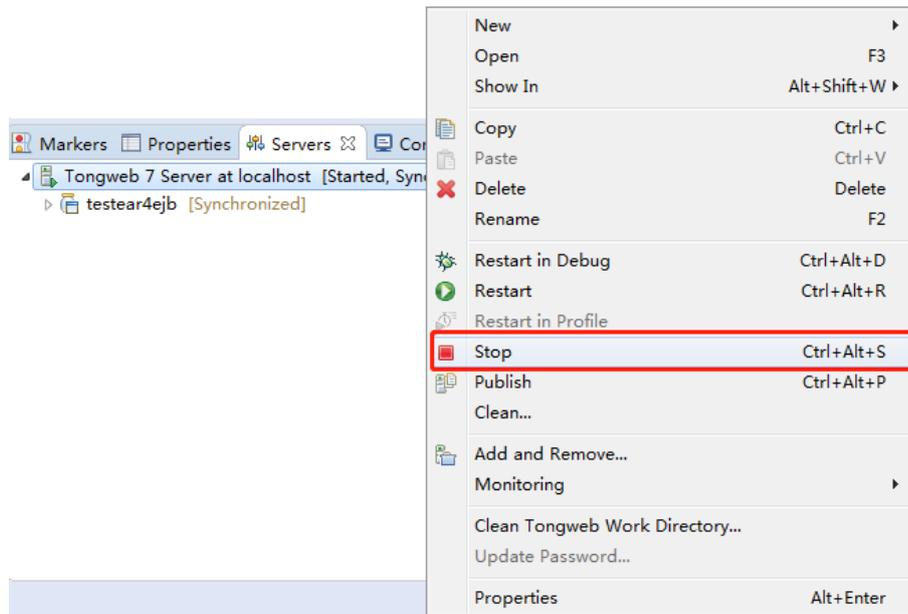


图 15.1.13 停止服务器

15.1.4.5 使用 Debug 功能

- 1) 用户可以在 Web 工程中的 Java 中设置断点，以 debug 方式启动 TongWeb7 实例。在 Server 视图中，右键选择“Debug”按钮启动 TongWeb7 服务器。
- 2) 如果需要配置远程 Debug 端口，需要在“Run”->“Debug configuration”->“Remote Java Applications”里，右键“New configuration”，在打开的界面中配置远程监听端口。

15.1.5 Eclipse 插件的卸载

- 1) 删除 TongWeb7 运行实例，在“Show”->“View”->“Server”中，选择要删除的 TongWeb7 实例，将其删除；
- 2) 删除 TongWeb7 服务器配置信息，通过“Windows”->“Preferences”菜单，打开“Server”->“Runtime Environment”界面，在右侧“Server Runtime Environments”中依次选中所有 TongWeb 服务器,选择“Remove”菜单删除；
- 3) 删除 Eclipse/plugins 下 TongWeb7 插件；
- 4) 重启 Eclipse。

15.2 Eclipse 中混淆器使用

Eclipse 混淆器可以对用户工程中的 Java 应用进行知识产权保护，打乱 class 文件中的符号信息，从而使对工程的反向编译变得非常困难。

- 1) 将\${TW7_HOME}/tools/eclipse-plugins 下的 com.tongtech.guardvplugin-1.0.0.jar 文件，拷贝至 eclipse_home/plugins 目录下，重启 Eclipse 开发工具即可。
- 2) 在 Eclipse 中新建一个 Java 工程，或在已存在 Java 工程上，在工程根目录右键选择“GuardV”->“settings”，进入此工程的混淆器配置界面，如图 15.2.1 所示：

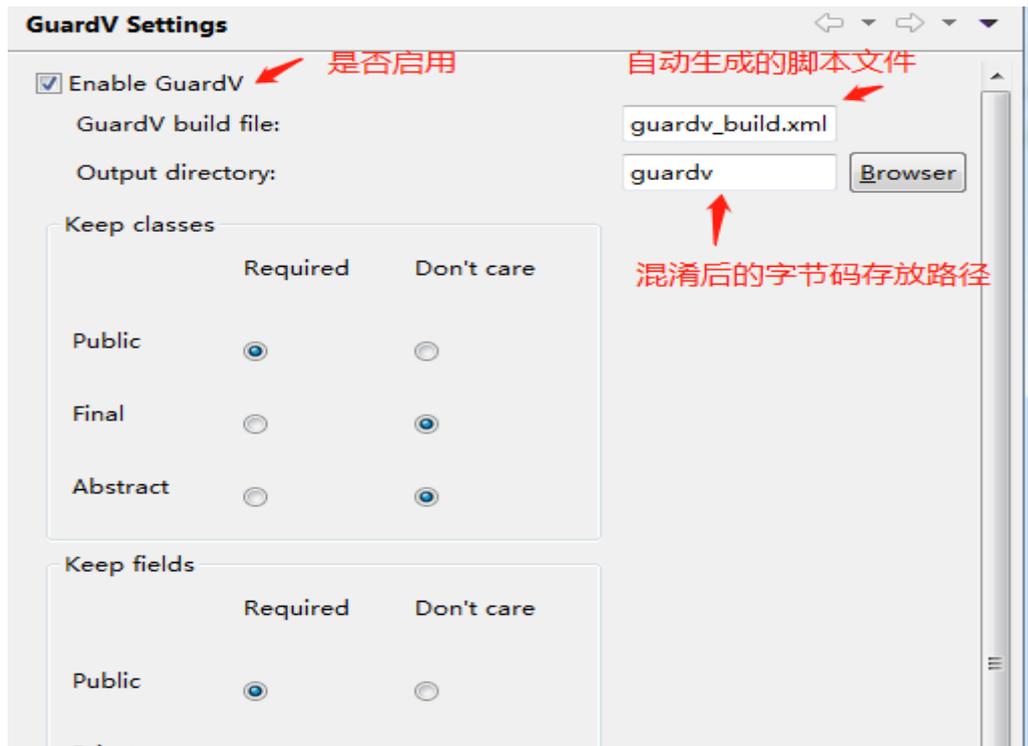


图 15. 2. 1 混淆器配置界面

配置说明:

- Keep classes: 可根据类的作用域来选择是否被混淆/保留;
 - Keep fields: 混淆时, 可判断成员变量的作用域来选择是否被混淆/保留;
 - Keep methods: 混淆时, 可判断成员方法的作用域来选择是否不被混淆/保留;
 - Required: 保留不混淆;
 - Don't care: 不关心, 工具自动判断是否保留或混淆;
- 3) 完成配置后, 点击“Apply and Close”, 工具会自动在工程目录下生成一个 xml 文件, 默认文件名 guardv_build.xml;
 - 4) 编译混淆: 在工程根目录下右键执行 “GuardV”->“build”, 根据 guardv_build.xml 文件, 混淆开始执行, 可以查看 Console 日志输出, 最终混淆后的字节码文件存放在工程的 guardv 目录下。

15.3 IDEA 插件

为了满足用户能在 IDEA 中快速开发应用程序, 并部署应用到 TongWeb7 上, 我们开发了 IDEA 插件工具。

15.3.1 IDEA 插件功能概述

IDEA 插件的基本功能主要包括 TongWeb7 应用服务器的添加、配置修改以及启动和停止、应用部署、调试等功能。

使用 IDEA 插件可以在开发工具中添加 TongWeb7 服务器, 并不影响源文件的前提下修改端口信息。

使用 IDEA 插件, 可以在 IDEA 开发工具中配置多个 TongWeb7 实例, 通过定义不同的端口来实现多实例运行。

15.3.2 IDEA 插件的安装与卸载

本手册基于 IDEA 2018.3.2 版本编写，其他版本相应描述位置可能会有不同。

15.3.2.1 安装 IDEA

直接到 IDEA 的官网上下载 IntelliJ IDEA Ultimate 版本，下载完成后，安装即可。
地址为 <https://www.jetbrains.com/idea/download/>。

15.3.2.2 安装 IDEA 插件

1) 打开 IDEA 开发工具，依次打开“File”->“Settings”->“Plugins” 在右侧面板顶部菜单中点击“齿轮”图标，选择“Install Plugin from Disk”；

2) 选择插件安装包，插件安装包路径为\${TW7_HOME}/tools/idea-plugins，找到 TongWeb7-IDEA-\${IDEA 版本}.zip；

3) 重启 IDEA 开发工具。

注：IDEA 部分版本的插件可能兼容，但是不能确定是否一定兼容，如果找不到对应版本的插件时，可以尝试使用其他版本。

15.3.2.3 卸载 IDEA 插件

1) 删除 TongWeb7 的运行配置，在主面板顶部菜单中选择“Run”->“Edit Configurations”，在左边树菜单中找到 TongWeb 的相关运行配置并移除；

2) 移除配置的 TongWeb7 服务器信息，在“File”->“Settings”->“Build,Execution,Deployment”->“Application Servers”，移除右边面板的所有 TongWeb 服务器；

3) 进入“File”->“Settings”->“Plugins”找到 TongWeb 插件，进入详情，在右上角下拉选择中选择卸载重启工具即可完成卸载。

15.3.3 IDEA 中 TongWeb7 插件的使用

安装完成后，重启 IDEA 就可以使用插件了，下面将介绍 TongWeb7 IDEA 插件的具体功能使用。

15.3.3.1 在 IDEA 中添加 TongWeb7

● 通过 Application Servers 中添加

1) 打开“Settings”视图，即在 IDEA 的选项中依次选择

“File”->“Settings”“Build,Execution,Deployment”>“Application Servers”，点击“+”号，选择“TongWeb Server”，如下图 15.3.1 所示：

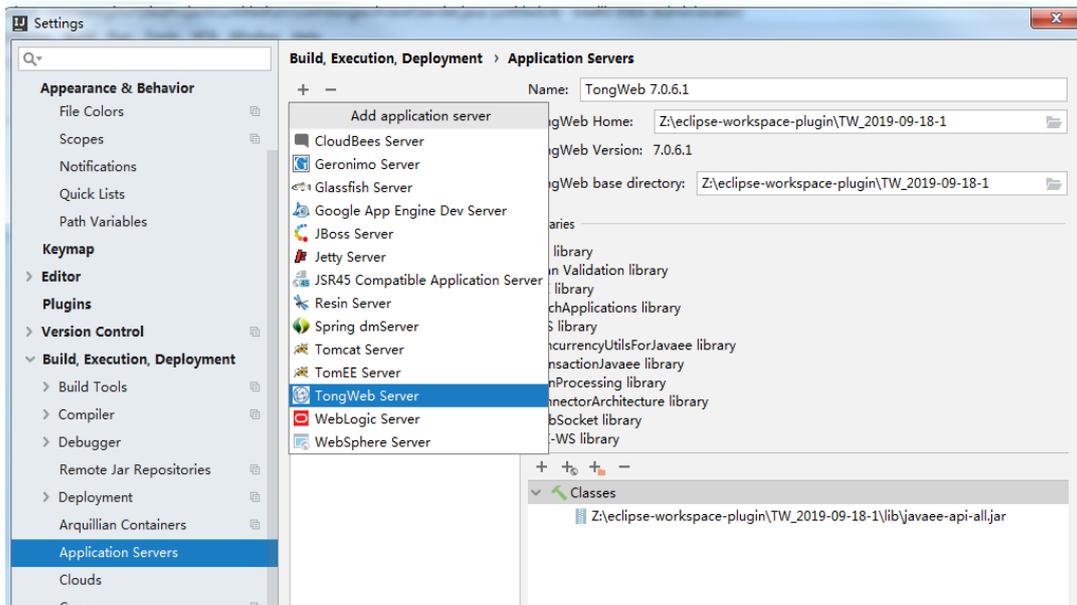


图 15.3.1 选择 TongWeb Server

- 2) 进入添加 TongWeb7 服务器界面，选择 TongWeb7 服务器路径，如图 15.3.2 所示：

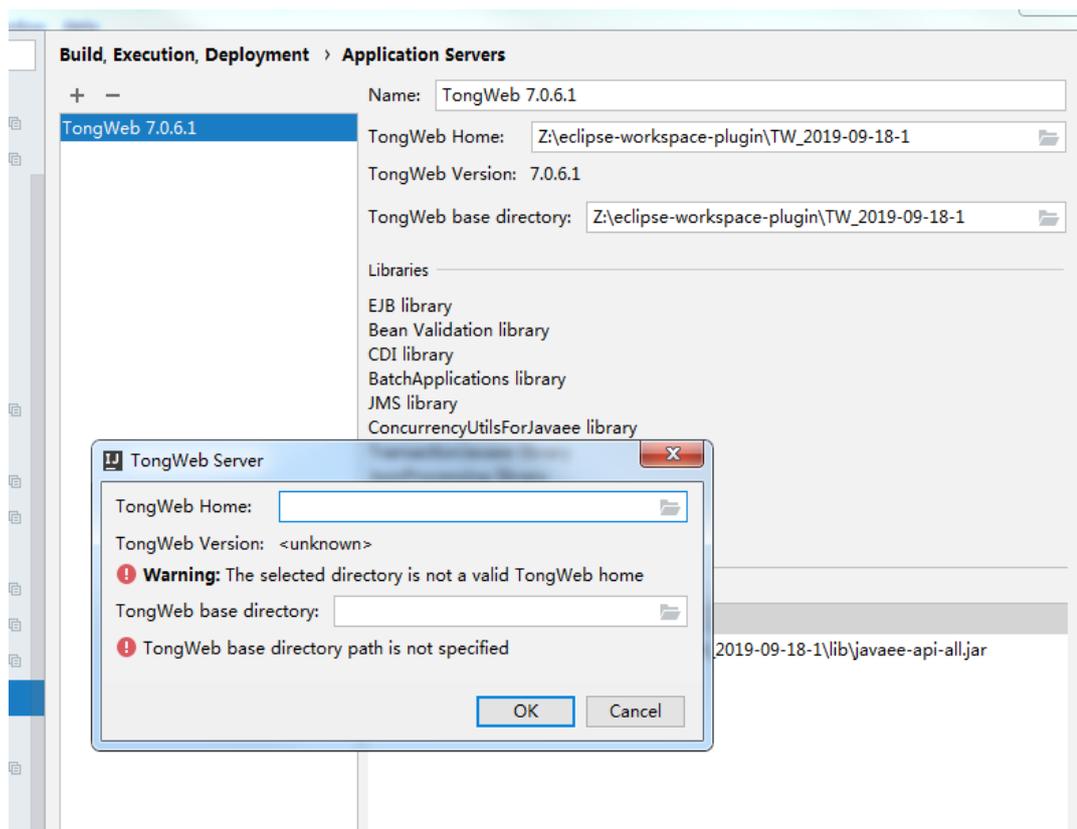


图 15.3.2 添加 TongWeb Server 界面

● 通过 Run/Debug Configurations 中添加

- 1) 打开“Run/Debug Configurations”视图，即在 IDEA 的选项中依次选择“Run”->“Edit Configurations”，点击“+”新建“TongWeb Server”->Local 配置，如下图 15.3.3 所示：

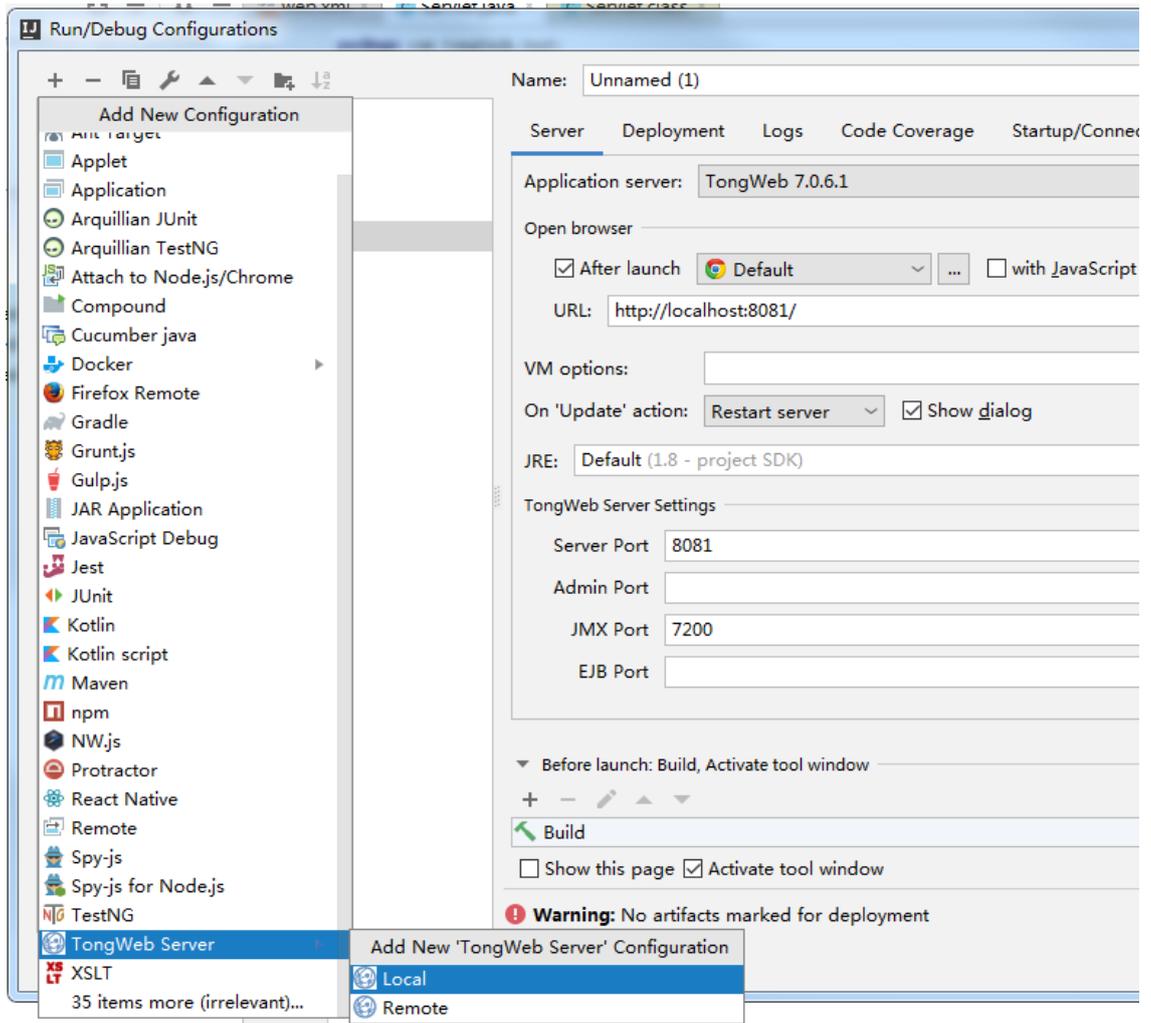


图 15.3.3 添加 TongWeb Server

- 2) 通过选择“Configure...”配置 TongWeb7 服务器；通过“...”配置，选择 TongWeb7 目录，确定即可，如下图 15.3.4 所示：

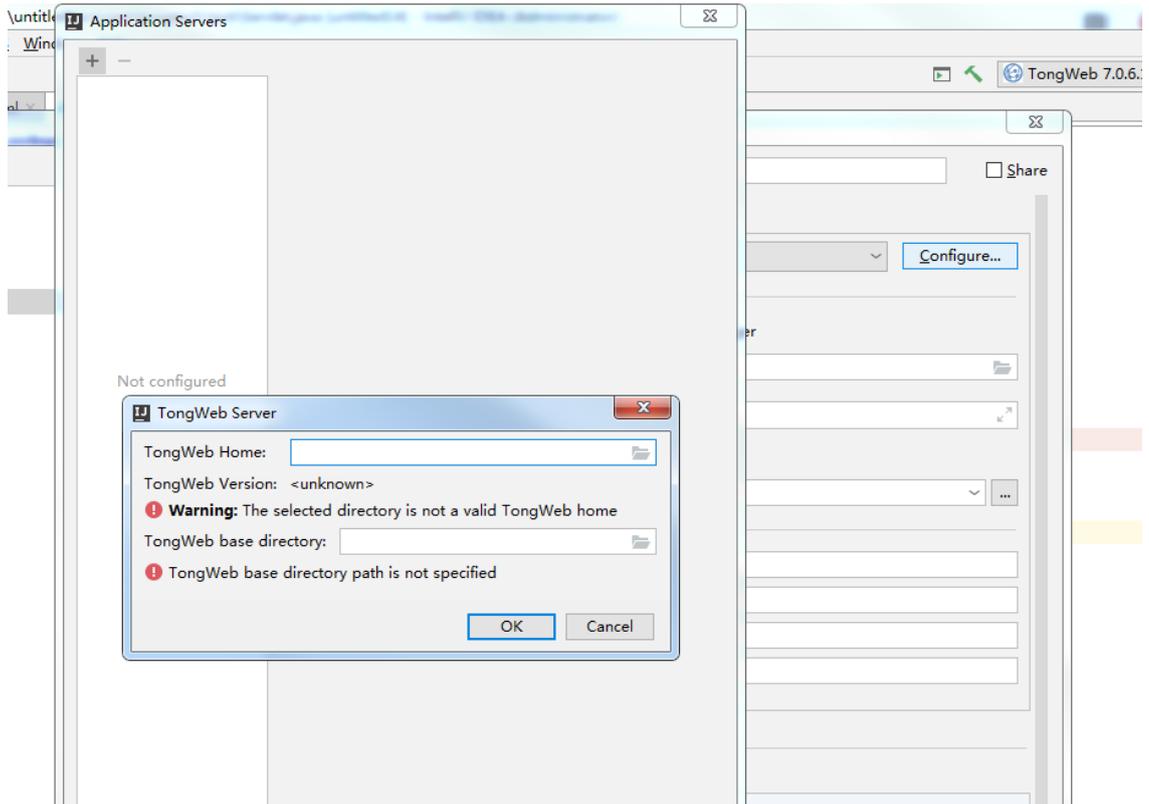


图 15.3.4 添加 TongWeb Server

15.3.3.2 部署工程

1) 在“Run->Debug->Edit Configurations”视图中新建 TongWeb7 运行配置，首先选择或配置 TongWeb 服务器，在 Deployment 面板中选择要部署的项目工程或者外部工程，如需使用不同的端口启动服务器，则需要在“Server”面板修改成未被占用的端口，其他参数视情况修改。如图 15.3.5 所示：

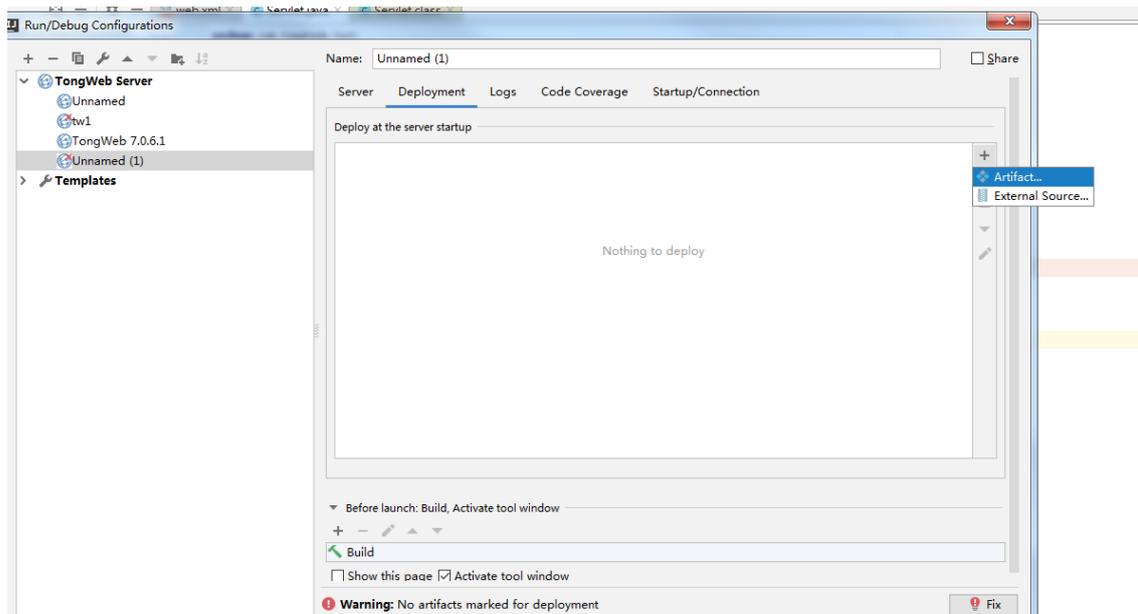


图 15.3.5 添加工程到服务器上

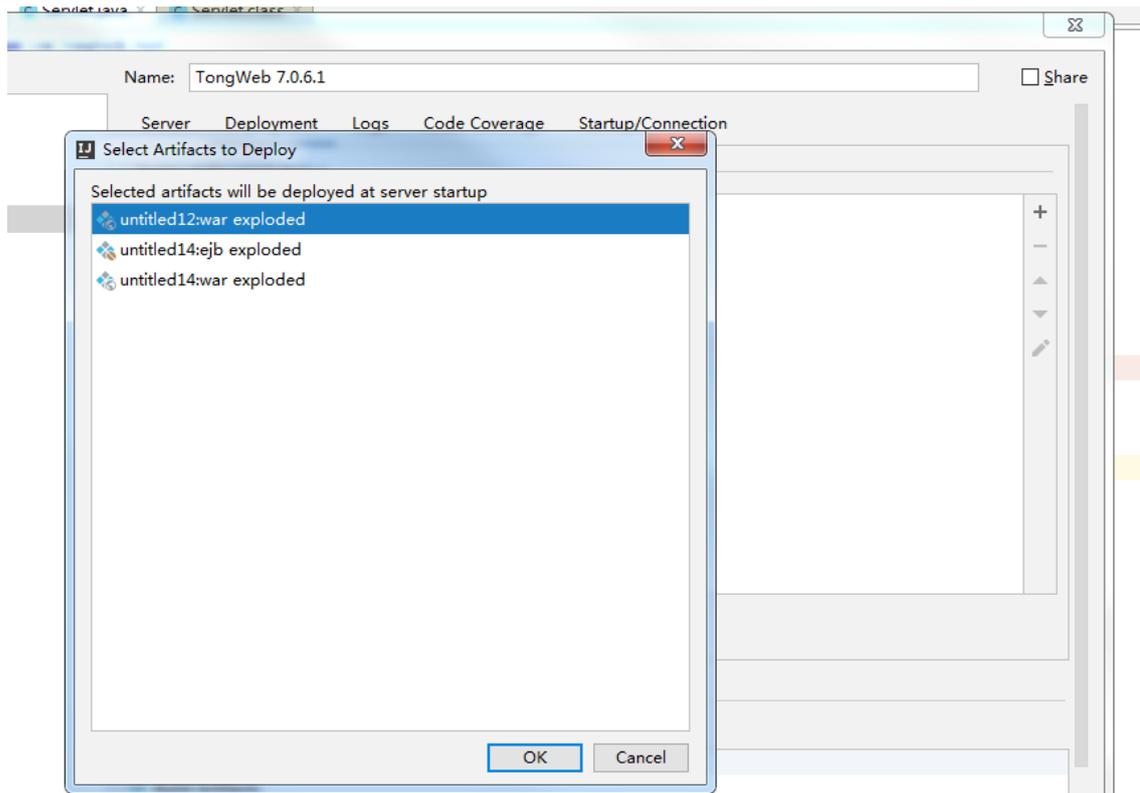


图 15.3.6 添加要部署的项目工程

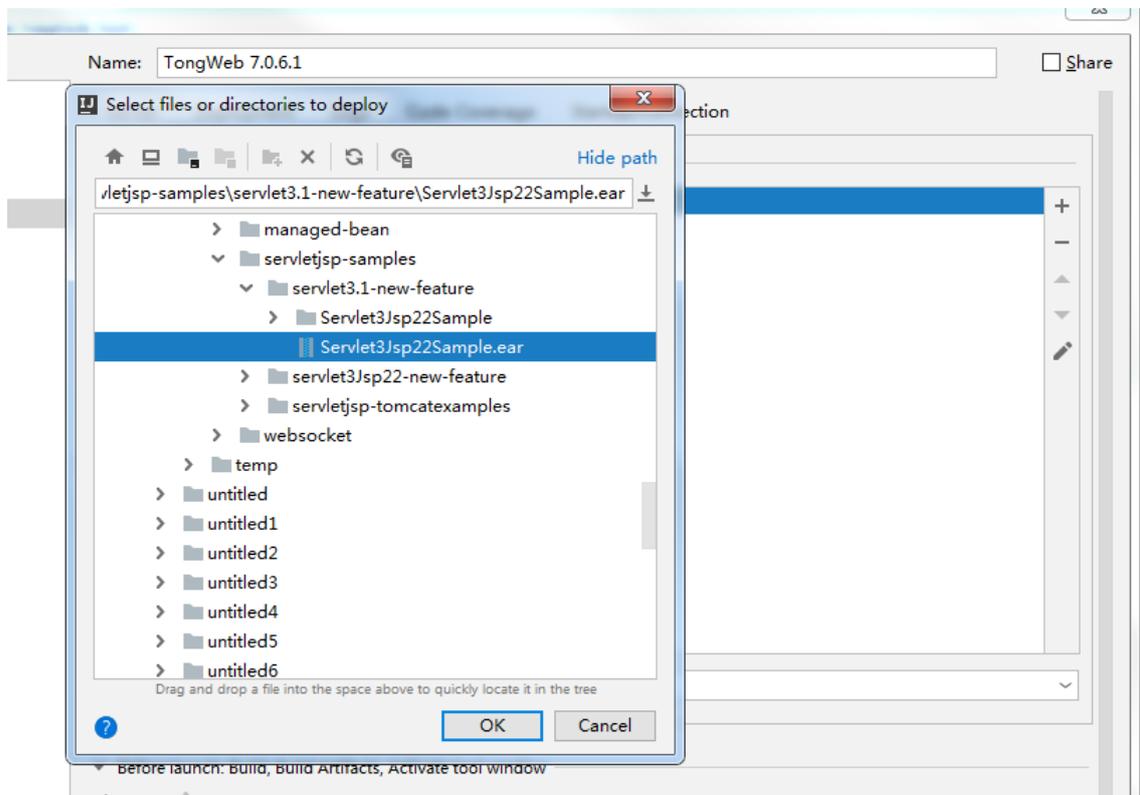


图 15.3.7 添加一个外部工程

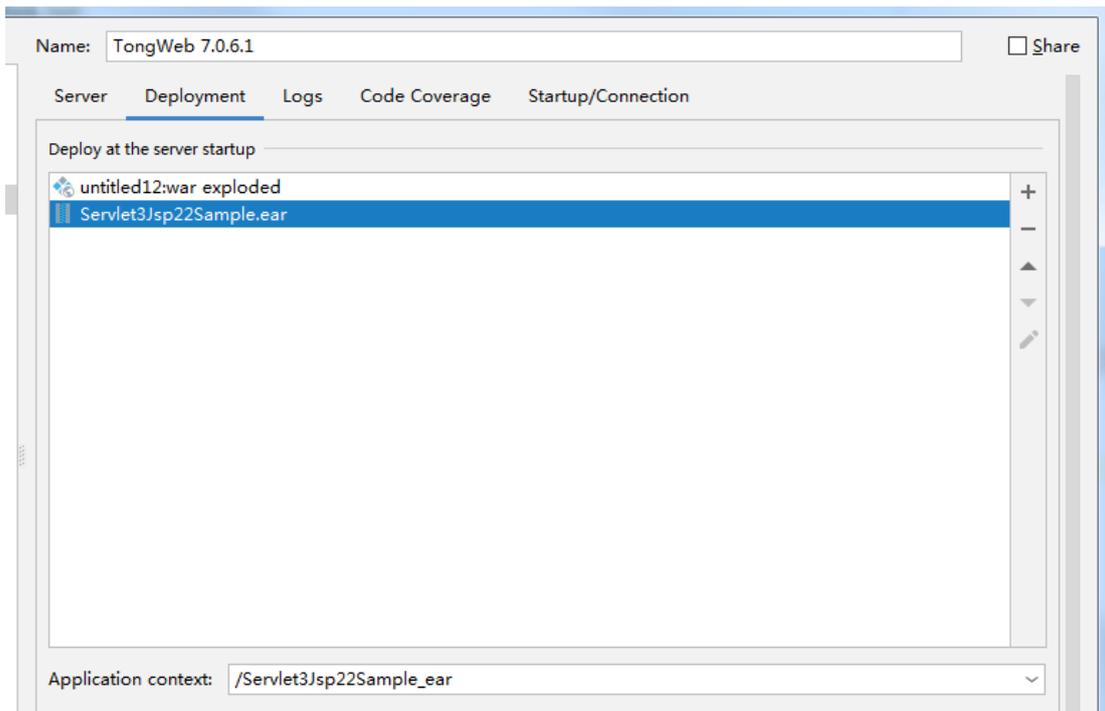


图 15.3.7 工程添加完成

2) 添加完成以后，便可启动(绿色三角形图标)或者调试（绿色小虫子图标）程序；如图 15.3.8 所示：

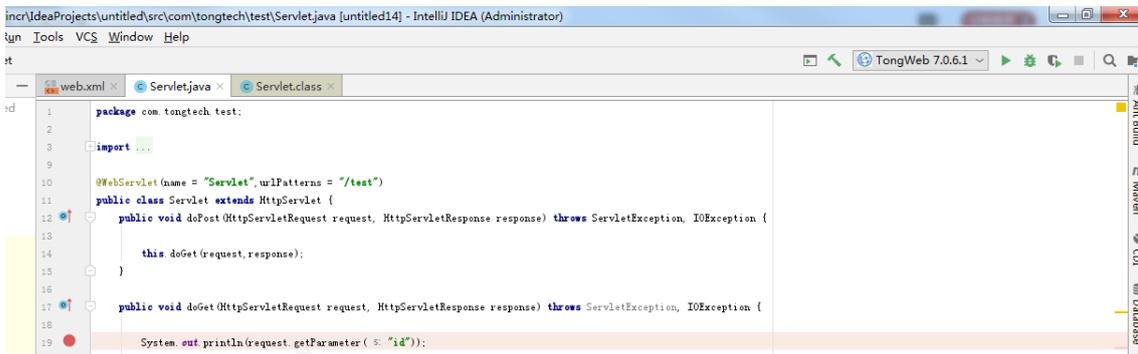


图 15.3.8 执行启动

3) 可以观察“Output”中日志信息，启动完成之后，可以通过浏览器访问项目，如图 15.3.9 所示：

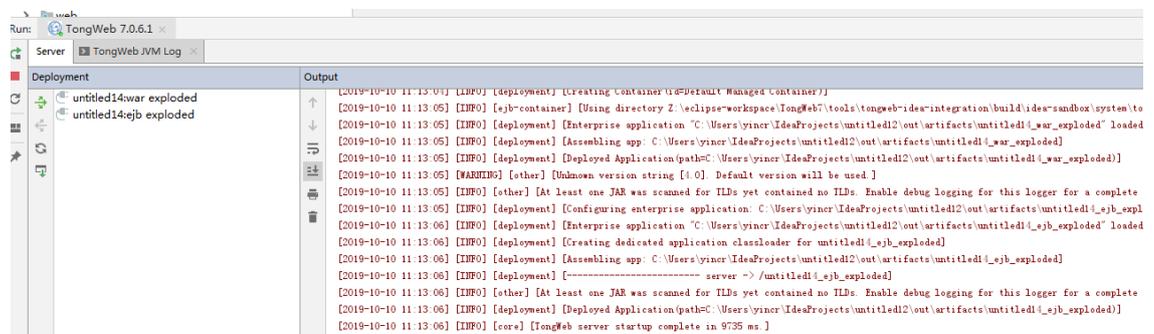


图 15.3.9 服务启动日志

注：如果部署的应用有前置依赖，比如 JPA，可在 IDEA 中启动容器，然后访问控制台，创建并配置需要的数据源，最后在 IDEA 中部署 JPA 应用，重启容器即可正常访

问。

15.3.3.3 使用 Debug 功能

在 Web 工程中的 Java 类设置断点，以 debug 启动即可。

15.3.3.4 解部署工程

在 “Run->Debug->Edit Configurations” 的 “TongWeb Server” 视图中，进入 “Deployment” 面板，选择要删除的应用工程，点击 “-” 即可解部署工程。

15.4 Patch 补丁工具使用说明

15.4.1 工具简介

Patch 补丁工具用于制作补丁和安装补丁，将以前的 TongWeb7 版本升级至指定版本。

注：升级前请先备份要升级的 TW

15.4.2 环境需求

JDK1.8 及以上版本，Windows 和 Linux 都可以制作和升级，下面以 Linux 环境为例。

15.4.3 配置说明

TongWeb7 补丁制作和升级过程中，需要涉及到以下几个版本，说明如下：

旧版本	待升级的版本，制作补丁时，需要安装干净版本
新版本	旧版本需要升级到的目标版本
用户版本	用户正在使用的版本，版本与旧版本一致

规则：

注：以下操作，全部可以通过定义特殊处理规则，进行过滤处理。

- 新版本中存在，旧版本中不存在的文件或属性，全部视为新增功能，需要迁移；
- 新版本中不存在，旧版本中存在的文件或属性，全部视为删除；
- 未定义特殊处理的文件，通过对比新旧版本 MD5，如果不一致统一视为覆盖更新；
- 文件属性根据定义配置规则，进行唯一标记，通过在 xmlProperties.xml 中定义；
- 文件属性在用户版本中有修改，保留用户修改值。
- 用户文件属性值和旧版本属性值一致，且新版属性值不同，则进行属性值更新。

1) 编写配置文件：xmlProperties.yam

xmlProperties.yaml 在 patch-7.0.jar 文件里，编辑时，需要解压 patch-7.0.jar，对里面的配置文件进行编辑，大部分情况下，不需要对该文件进行修改。说明如下：

定义	取值	描述
文件名称	无	定义需要处理的文件
ns	当前文档的 namespace	如果文档有 NameSpace 属性需要定义

uniqueDefine	唯一标识：属性用属性名标识，内部元素用<元素名称>标识	对元素进行唯一标识，便于识别通过相同标识进行文件内部属性对比
--------------	-----------------------------	--------------------------------

配置文件对应为

```

1  tongweb.xml:
2      ns:
3      uniqueDefine:
4          /tongweb/apps/web-app: name
5          /tongweb/apps/ejb-app: name
6          /tongweb/server/virtual-host: name
7          /tongweb/server/http-listener: name
8          /tongweb/servlet: <servlet-name>
9          /tongweb/servlet/init-param: <param-name>

```

2) 定义特殊处理文件 customize.yaml

特殊处理文件可定义文件名称修改，文件夹升级过滤，文件全部替换，避免或添加特殊处理操作。定义需处理特定操作，标准 yam 格式。编辑格式如下：

操作	包含定义	描述
filter:		定义特出处理的内容
	/文件相对路径: skip	需跳过文件的文件列表，需要文件或者文件夹相对路径
rename:		定义需要重命名的文件或者文件夹
	- /原文件: 新文件	<p>重命名的文件列表，文件夹和内部文件都需要重命名时，有先后顺序。</p> <p>1、先定义文件夹重命名，则内部文件的路径中，都需要按重命名后的路径填写。</p> <p>2、先定义内部文件重命名，则按原路径定义，最后在修改文件夹名。</p>

例如需求如下：

- 在制作补丁时，跳过 apache-activemq-linux 的升级对比；
- 在制作补丁，跳过 apache-activemq-win 的升级对比；
- 升级补丁时，对/Agent/bin 文件夹进行重命名为 bin1；
- 升级补丁时，对/Agent/bin1 下的 start.sh 重命名为 startserver.sh；

配置如下图：

```

filter:
  /apache-activemq-linux: skip
  /apache-activemq-win: skip
rename:
  - /Agent/bin: /Agent/bin1
  - /Agent/bin1/start.sh: /Agent/bina/startserver.sh

```

15.4.4 制作补丁

TongWeb7 补丁制作工具，是通过要升级的版本与最新版本的比较，然后从最新版本把差异文件保留目录格式摘出来存放到一个名为 patchxxxx 的文件夹下，制作过程如下：

- 1) 在\${TW7_HOME}/tools/patch 目录下，解压 patch-7.0.zip，解压后，patch-7.0 目录下包含以下目录结构如下图：

名称	修改日期	类型	大小
lib	2019/10/22 9:22	文件夹	
patch-7.0.jar	2019/10/22 9:22	Executable Jar File	50 KB

- 2) 分别安装需要升级版本的 TongWeb7、要升级到指定版本的 TongWeb7，安装完成后，在 patch-7.0 目录下执行以下命令：

格式：

```
java -jar patch-7.0.jar patch 补丁包路径 待升级版本路径 升级版本路径
```

如：

```
java -jar patch-7.0.jar patch /home/tongweb/patch/TW7025_T0_TW7040_patch
/home/tongweb/TW7.0.2.5 /home/tongweb/TW7.0.4.0
```

提示 patch Success； 则补丁生成成功；

说明：

/home/tongweb/patch/TW7025_T0_TW7040_patch:生成的补丁路径；

/home/tongweb/TW7.0.2.5: 待升级版本路径；

/home/tongweb/TW7.0.4.0: 需要升级到指定版本路径；

15.4.5 升级补丁

将制作好的补丁包**patch.jar 和补丁迁移工具 patch-7.0.zip 完全复制到待升级服务器上。分别解压**patch.jar 和 patch-7.0.zip，解压后，在 patch-7.0 目录执行以下命令：

格式：

```
java -jar patch-7.0.jar patch 补丁包路径 待升级版本路径 升级版本路径
```

如：

```
java -jar patch-7.0.jar migration /home/tongweb/TW7025_T0_TW7040_patch
/home/tongweb/TW7025 /home/tongweb/TW7025_backup
```

提示 Migration Success: 则补丁升级成功

说明:

/home/tongweb/TW7025_T0_TW7040_patch: 补丁包路径;

/home/tongweb/TW7025: 待升级的版本路径;

/home/tongweb/TW7025_backup: 为此次升级被删除文件的备份地址, 无需备份可不带次参数, 将不进行备份;

15.5 TongAPM 工具使用说明

APM 是 Application Performance Management (应用性能监控) 的缩写, 通过使用 APM 工具, 能够获取精准的大量监控信息, 迅速定位应用的性能瓶颈, 协助开发/运维人员优化应用性能。

15.5.1 安装 TongAPM 工具

TongAPM 由客户端 tongapm.jar 和管理控制台 tongapm.war 两部分组成, 该文件存在于 $\${TW7_HOME}/lib/apm$ 目录下, tongapm.war 不能单独部署使用, 必须结合 tongapm.jar 一起使用。

- 1) 使用 APM 需要配置 JVM 的启动参数, 通常情况下可以通过 $\${TW7_HOME}/bin/external.vmoptions$ 文件, 指定 APM 的 jar 包路径来开启 APM 功能, 使用方式如下:

JDK9 及以下版本:

需要在 $\${TW7_HOME}/bin/external.vmoptions$ 添加以下参数:

```
-javaagent: $\${TongWeb\_Home}/lib/apm/tongapm.jar$ 
```

```
-Xbootclasspath/a: $\${TongWeb\_Home}/lib/apm/tongapm.jar$ 
```

JDK9 以上版本:

1. 将 $\${TW7_HOME}/lib/apm/tongapm.jar$ 复制到 $\${TW7_HOME}/lib/endorsed$ 目录下;

2. 在 $\${TW7_HOME}/bin/external.vmoptions$ 添加 `-javaagent: $\${TongWeb_Home}/lib/apm/tongapm.jar$` 配置;

- 2) 配置完参数后, 重启应用服务器。
- 3) 登录 console 控制台, 部署 tongapm.war 到应用服务器上、部署完成如图 15.5.1 所示



名称	前缀	应用类型	部署源类型	部署方式	虚拟主机	状态	操作
tongapm	/tongapm	war	文件部署	控制台部署	server	已启动	重部署 http访问
genericra		rar	目录部署	控制台部署	server	已启动	重部署

图 15.5.1 TongWeb 应用管理

- 4) 在图 15.5.1 中点击 tongapm.war 后面的 http 连接, 进入 TongAPM 控制台, 首页如图 15.5.2 所示:



图 15.5.2 TongAPM 管理控制台

15.5.2 APM 配置

在 TongAPM 应用性能管理控制台中，点击“APM 配置”，进入 APM 配置界面，如下图 15.5.3 所示：



图 15.5.3 APM 配置界面

15.5.3 慢请求分析首页

在 APM 管理平台左侧导航树中，点击“慢请求分析”节点，进入慢请求分析首页，如图 15.5.4 所示：



图 15.5.4 慢请求分析首页

慢请求页面功能主要包含以下信息：

- “过滤请求” 的值可控制只显示包含了输入值的结果；
- “平均响应时间” 默认显示前 20 条记录、可根据“显示行数” 按钮改变显示条数；
- “停止分析” 按钮表示停止请求分析功能并清除历史数据；
- “刷新”， 按钮，可展示当前页面选择条件最新分析结果；

15.5.4 慢请求追踪

在“慢请求分析”界面中，点击列表后面“追踪”按钮，进入慢请求追踪列表界面，如下图 15.5.5 所示：



图 15.5.5 慢请求追踪列表页

在“慢请求追踪”列表中，点击具体的请求记录，进入调用过程分解页面，该页面详细展示此次请求的关键模块调用，如下图所示 15.5.6：

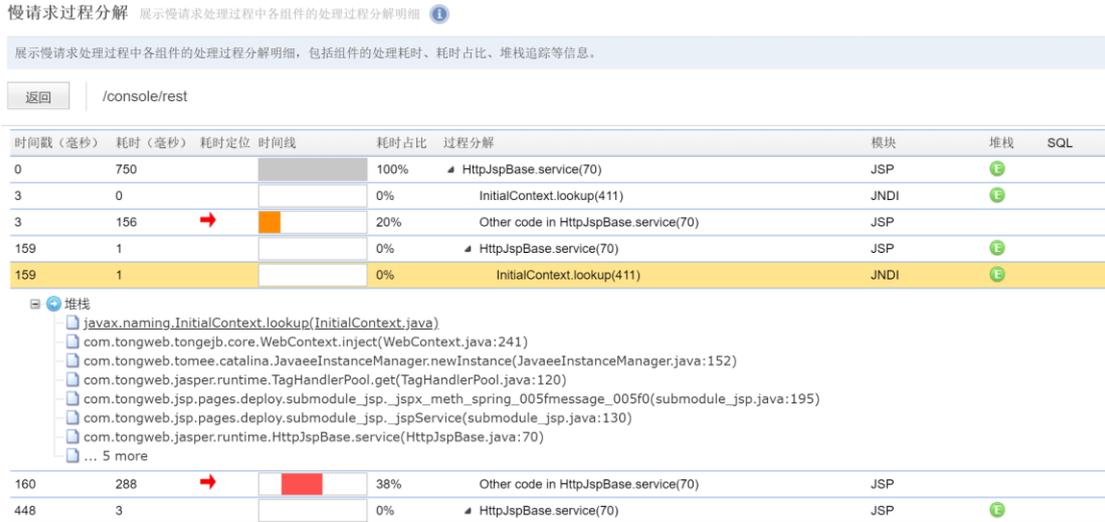


图 15.5.6 慢请求过程分解页面

15.5.5 类方法分析

在 APM 管理台左侧导航树中，点击“类方法分析”节点，进入慢请求分析首页，如图 15.5.7 所示：

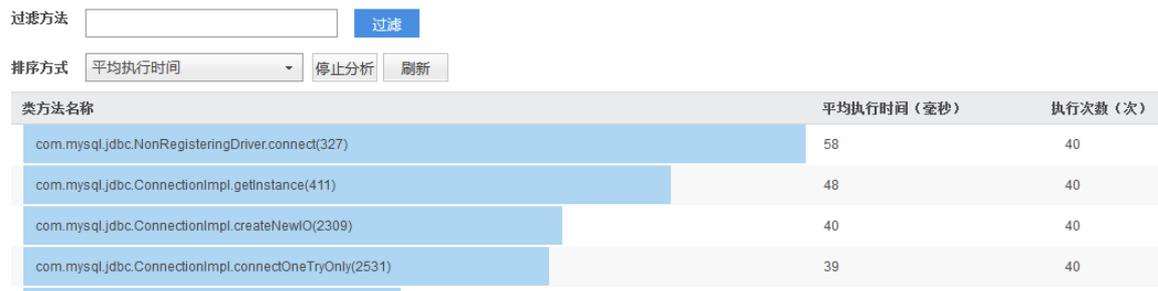


图 15.5.7 类方法分析页面

15.5.6 线程剖析

点击左侧导航树中的“线程剖析”节点，出现线程剖析功能主界面，如图 15.5.8 所示



图 15.5.8 线程剖析功能主界面

在图 15.5.8 “线程剖析列表”中，点击某次的线程剖析结果时间，进入该次线程剖析的结果展示页面，如图 15.5.7 所示：

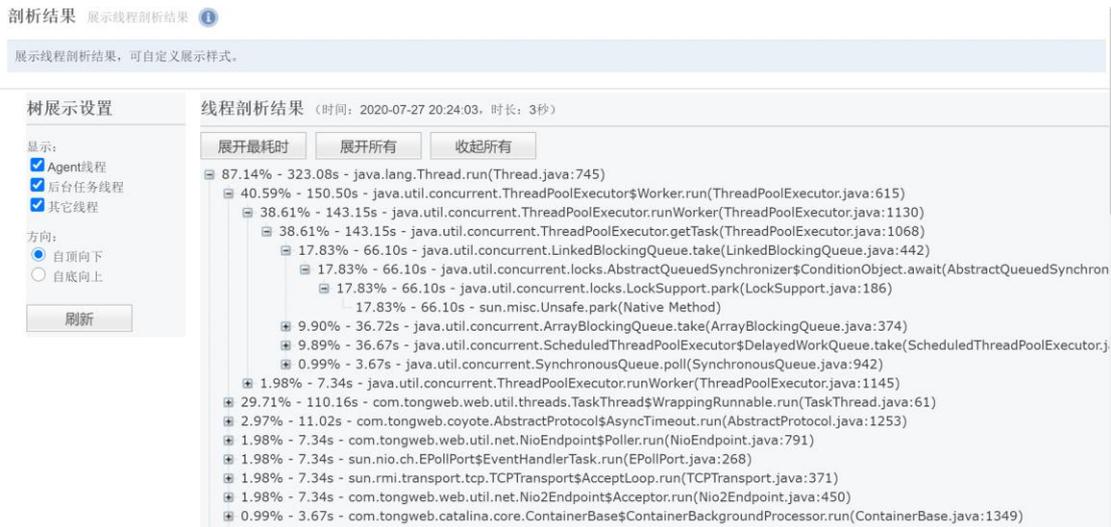


图 15.5.8 线程剖析的结果展示页面

15.5.7 JDBC-TOP SQL

展开管理控制台左侧导航树中的“JDBC”节点，点击“TOP SQL”节点，进入 TOP SQL 页面，如图 15.5.9 所示：

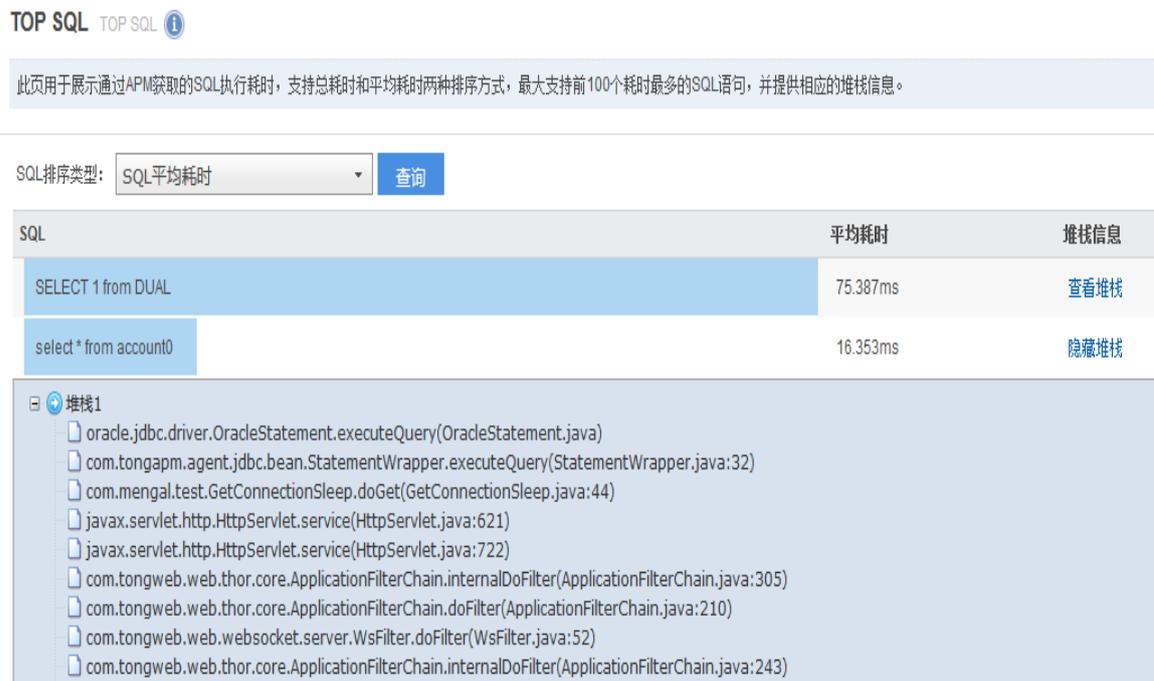


图 15.5.9 TOP SQL 界面

15.5.8 JDBC-JDBC 资源泄漏

展开管理控制台左侧导航树中的“JDBC”节点，点击“JDBC 资源泄漏”节点，进入 JDBC 资源泄露页面，如图 15.5.10 所示：

JDBC资源泄漏

此页用于展示通过APM监测到的JDBC资源泄漏，包括连接池器和Statement池器，并提供相应堆栈以对泄露代码进行定位。

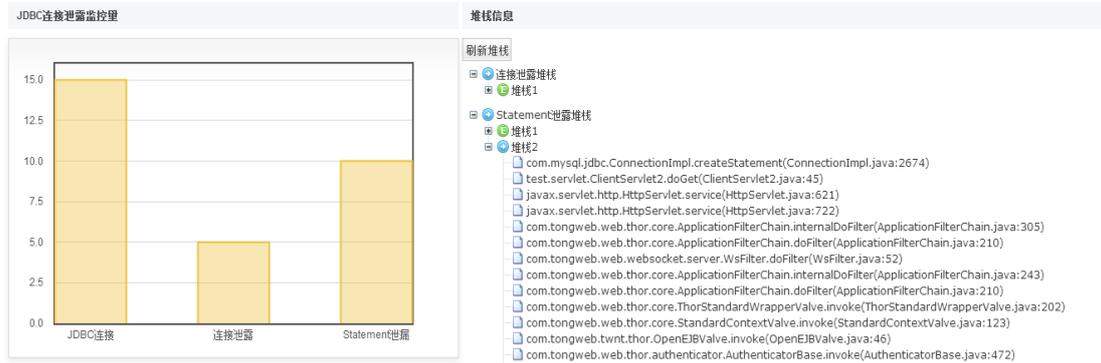


图 15.5.10 JDBC 资源泄漏

注：以下是 JDBC 连接所支持的数据库

- Oracle
- Mysql
- MicroSoft SQL Server2000
- DB2
- Sybase

15.5.9 内存分析-潜在内存泄漏

展开 APM 管理控制台左侧导航树中的“内存分析”，点击”潜在内存泄露“节点，进入潜在内存泄漏分析界面，如图 15.5.11 所示：



图 15.5.11 潜在内存泄漏

每条记录后面，有“查看堆栈”连接，进去后可以查看堆栈的详细信息。

15.5.10 大对象分析

展开 APM 管理控制台左侧导航树中的”内存分析“，点击”大对象分析“节点，进入大对象分析详情界面，如图 15.5.12 所示：

此页用于展示当前占用内存存在1MB以上并且正在持续增长的类实例的相关信息，包括对象的类名、实例数、占用内存、内存占比。

类名	实例数	占用内存	内存占比
java.lang.String	176042	4.02MB	0.19%
java.util.LinkedHashMap\$Entry	71993	2.74MB	0.13%
[Ljava.util.HashMap\$Entry;	22409	1.84MB	0.08%
java.lang.Class	17413	1.58MB	0.07%
java.util.concurrent.ConcurrentHashMap\$HashEntry	39106	1.19MB	0.05%

图 15.5.12 潜在内存泄漏

15.6 appclient 工具

在 TongWeb7 应用服务器\${TW7_HOME}/tools/appclient 目录下，提供了 appclient 实用命令行工具来运行客户端程序，即支持客户端应用调用远程 EJB 对象。下面新建一个具体 EJB 用例介绍 appclient 命令行工具的使用。

15.6.1 安装 Eclipse

Eclipse 的官网下载地址为 <https://www.eclipse.org/downloads/packages/>，根据电脑位数下载对应的 Eclipse 版本，下载完成后，解压即可。

15.6.2 在 Eclipse 中创建客户端应用

1) 在 Eclipse 中点击“Create a Java EE application client project”，创建一个 Java EE 项目，如图 15.6.1 所示：

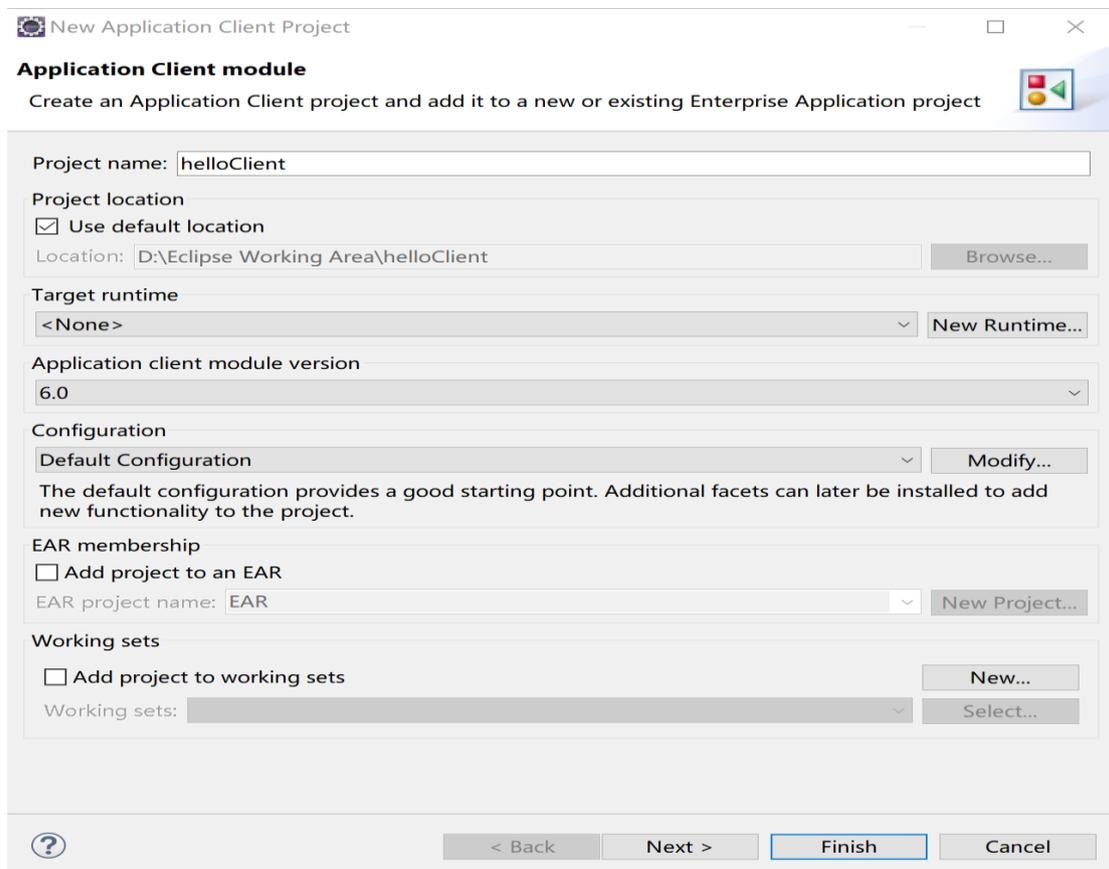


图 15.6.1 创建 JavaEE 项目

2) 点击”Next>“下一步，继续“Next>”进入以下界面，如图 15.6.2 所示：

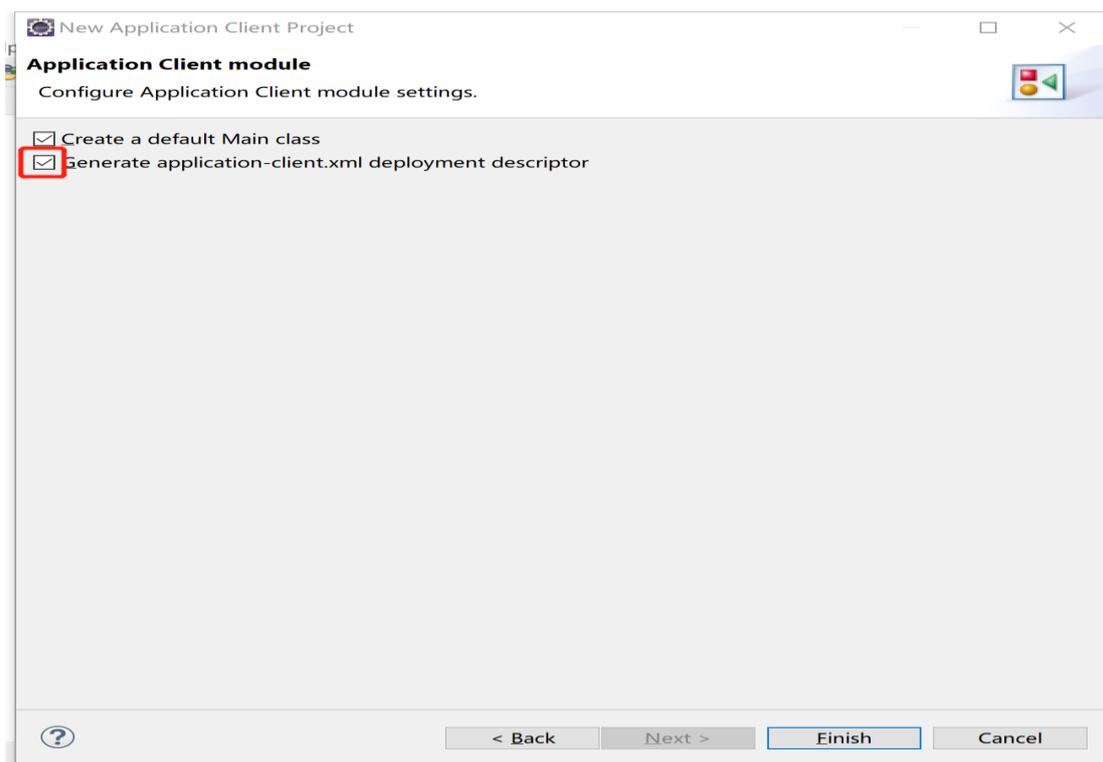


图 15.6.2 构建部署文件

3) 点击“Finish”，项目创建完后进 eclipse 主界面，如图 15.6.3 所示：

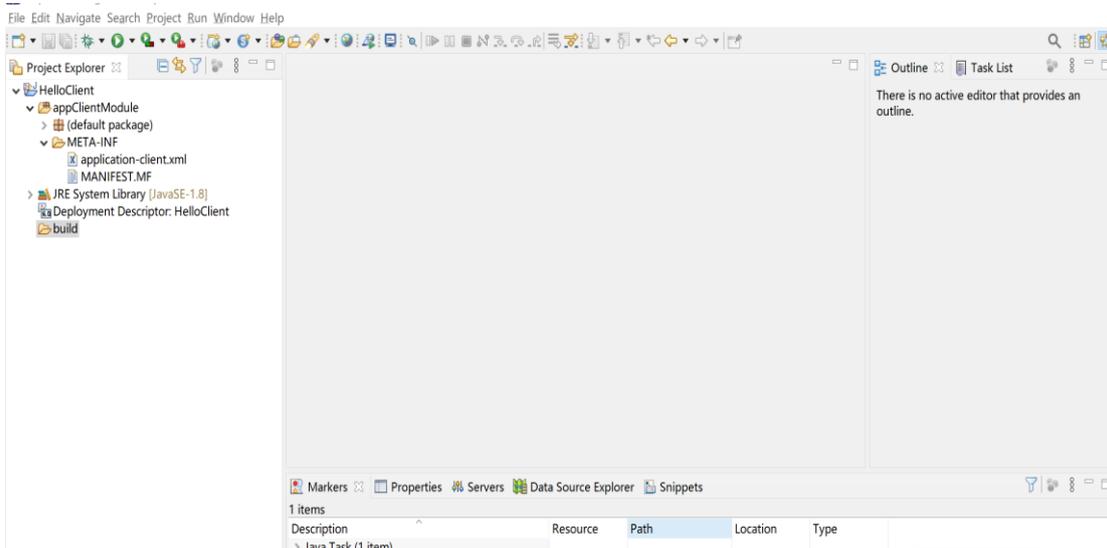


图 15.6.3 创建完 JavaEE 工程

4) 右键点击该项目>光标移动到 Build Path>Configure Build Path ...进入配置环境变量界面，选择“Libraries”如图 15.6.4 所示：

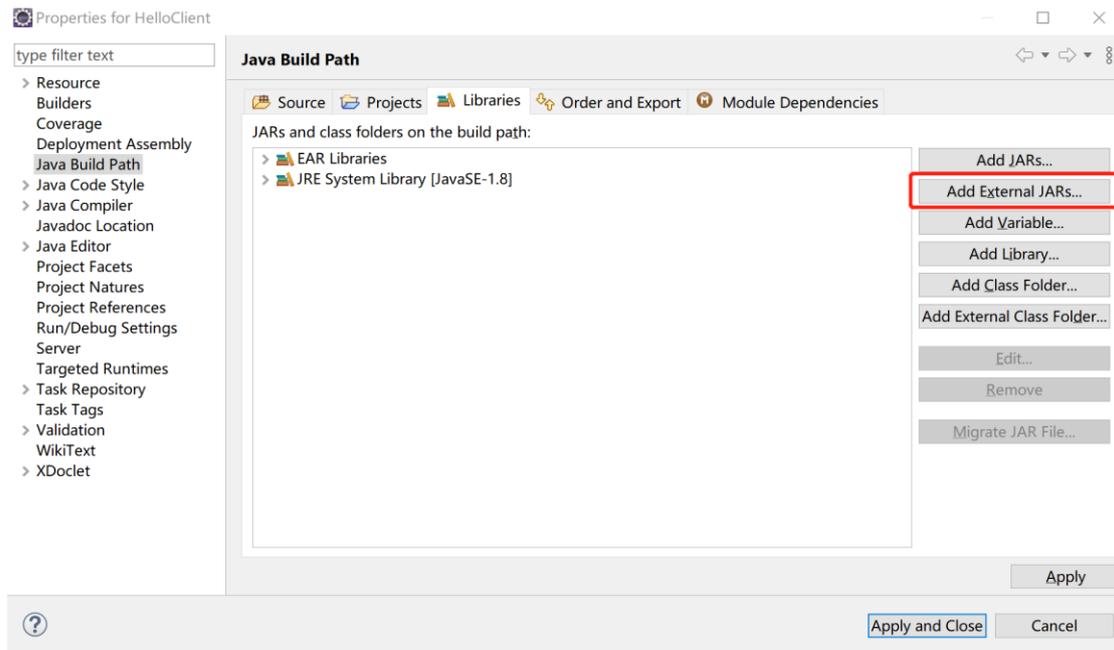


图 15.6.4 配置环境变量

5) 在上图 15.6.4 中，点击红色标记框中的“Add External JARs...”添加外部 jar 包，选择好需要引用的外部 jar 包，如图 15.6.5 所示。加载后，现金“Apply and close”应用并关闭。

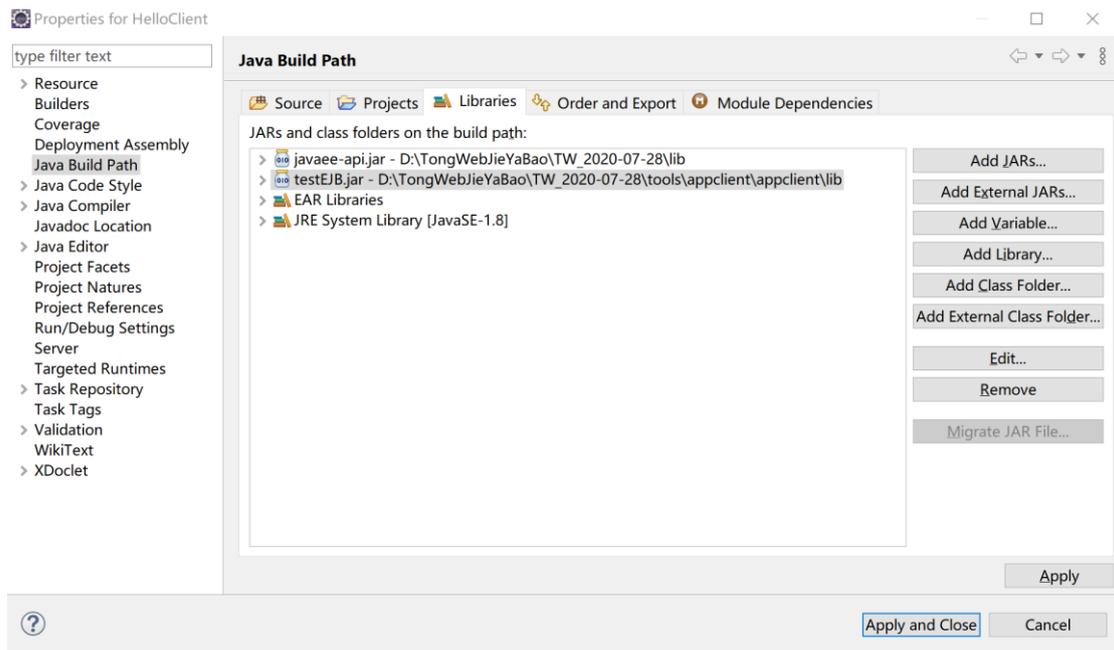


图 15.6.5 配置外部 jar 包

6) 创建 main-class: 右键项目里面的包目录>光标移动到“new”>选择“Class”键入创建 Class 类界面，如图 15.6.6 所示：

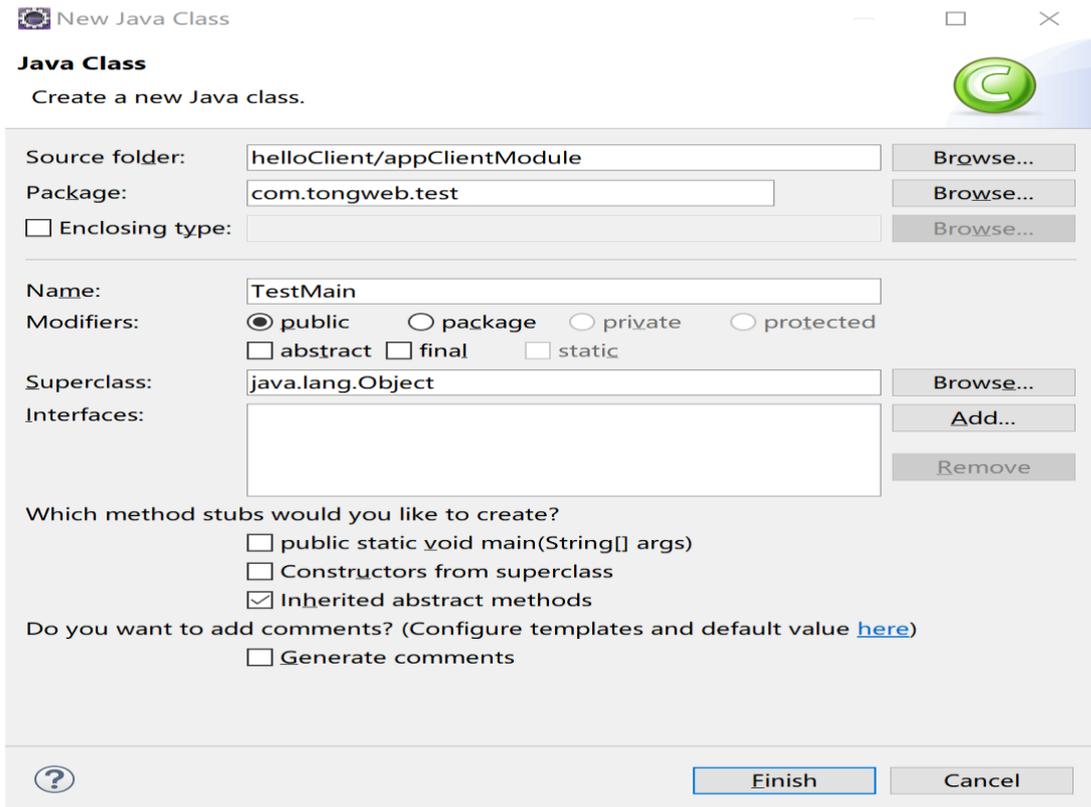


图 15.6.6 创建 Class 类

7) 在新建的“TestMain”类，编写客户端代码，如下图 15.6.7 所示：

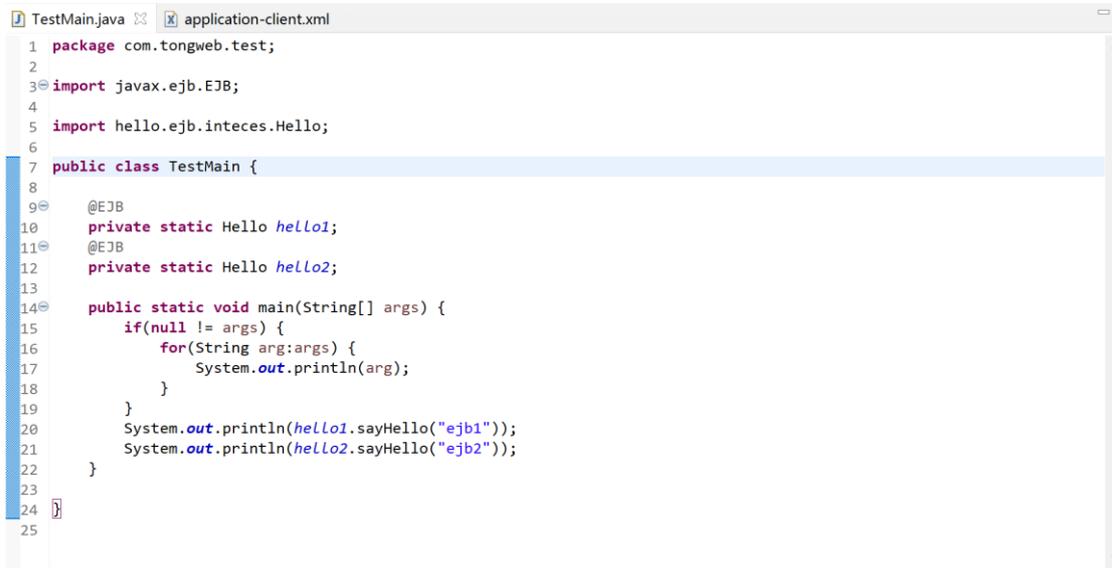


图 15.6.7 客户端代码

8) 在“application-client.xml”文件中，点击文件下方“Source”进入源代码编辑，如图 15.6.8 所示：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <application-client version="6" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
3 <display-name>HelloClient</display-name>
4 <ejb-ref>
5 <!-- 表示Main-Class中使用EJB组件的属性，格式为类全名/属性名 -->
6 <ejb-ref-name>com.tongweb.test.TestMain/hello1</ejb-ref-name>
7 <!-- EJB在TW中的JNDI名称，可在控制台JSDI远程EJB域中查找 -->
8 <lookup-name>global/HelloEJB/HelloBean!hello.ejb.interface.Hello</lookup-name>
9 </ejb-ref>
10
11 <ejb-ref>
12 <ejb-ref-name>com.tongweb.test.TestMain/hello2</ejb-ref-name>
13 <lookup-name>global/HelloEJB/HelloBean!Hello.ejb.interface.Hello</lookup-name>
14 </ejb-ref>
15
16 </application-client>
```

图 15.6.8 描述文件

9) 指定 main-class 的同时也需配置 class-path 值，一般就是配置依赖的 EJB 模块，如下图 15.6.9 所示：

```
1 Manifest-Version: 1.0
2 Class-Path: helloEJB.jar
3 Main-Class: com.tongweb.test.TestMain
4
5
```

图 15.6.9 描述文件

10) 右键选择指定项目>光标移动到“Export”>选择“App Client JAR file”进入打包界面，如图 15.6.10 所示：

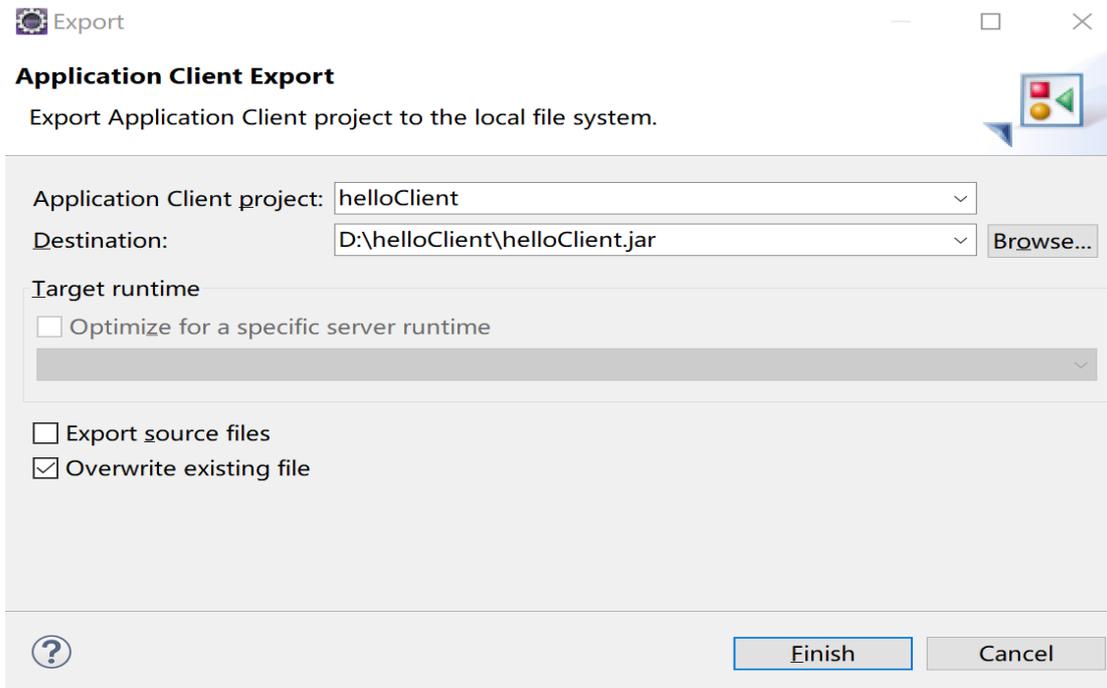


图 15.6.10 生成 jar 文件

15.6.3 Applient 工具调用 EJB

在 `${TW7_HOME}/tools/appclient/` 目录下，解压 `appclient.zip`。使用时需确保 `helloClient.jar` 和 `class-path` 里面的依赖在同一个位置，否则会提示找不到类，如果依赖有数据库驱动包，那么需要将驱动包放置在 `appclient/lib` 目录下。

- 用法

```
applient jarfile [url<url>] [arg...]
```

jarfile: 指定的 jar 文件;

url: 指定服务器 url, 服务器格式为 IP: port;

- 示例

```
sh appclient/bin/appclient.sh helloClient.jar
```

```
localhost:appclient yincr$ sh appclient/bin/appclient.sh helloClient.jar 127.0.0.1:5100 arg1 arg2
TongWeb EJB SERVER URL:http://127.0.0.1:5100/ejbserver/ejb
三月 10, 2020 11:32:57 下午 com.tongweb.tongejb.util.LogStreamAsync run
信息: AntiJarLocking enabled. Using URL cache dir /var/folders/sh/1x_c0q4s1mz90vvsft2q9pbm0000gn/T/temp46391
arg1
arg2
Hello: ejb1
Hello: ejb2
```

1. 附录 1 应用配置说明

1.1 .tongweb-web.xml

元素/属性名	说明
property	web 应用其它可配的属性
以下为 session 高可用相关属性配置。	

hazelcast-group-name	集群组名称，默认为 dev
hazelcast-group-password	集群组密码，默认值为 dev-pass
hazelcast-cluster-members	要连接到的集群节点 ip。可以指定 1 个或多个，用逗号分隔；可以加端口号也可以不加。客户端按此顺序尝试连接服务端，直到连接上。 127.0.0.1 IP 地址 IP 地址列表。
hazelcast-copysafe	是否进行副本序列化，默认为 false
hazelcast-asyncwrite	是否异步写入 false（默认同步）
hazelcast-uriignore	不保存在缓存集群里的资源类型，默认值为 <code>.*\.(png gif jpg css js)\$</code>
hazelcast-timeout	对缓存集群的操作超时，单位：毫秒 默认值 100
hazelcast-stick	Web 集群亲和设置开关，默认为 true（亲和）。设置为 false 表示非亲和。该参数与 <code>-Dwebcluster.session.sticky</code> 参数至少有一个设置了非亲和，则 tw 集群将运行在非亲和模式，只有在两个参数都是亲和配置（默认值）下，tw 集群才会运行在亲和模式下。
hazelcast-enabled	是否启用集群功能，默认值为 false
元素/属性名	说明
context-root	Web 应用访问前缀，该值可能被部署 Web 应用时的“应用前缀”属性覆盖。
元素/属性名	说明
security-role-mapping	把角色映射到当前安全域中的用户或组。最多配置一个。
role-name	被映射的角色名称，根据用户名或者用户所在组进行映射。 角色名(唯一)，对应 JavaEE 标准部署描述符文件中的 security-role 元素
principal-name	拥有该角色的用户名，如果有 group-name，用户名可以写 0 个或者多个，如果没有 group-name，用户名至少要有 1 个 注意：在没有 group-name 时，只有 principal-name 设置的用户名可以通过安全认证，需要配置多个用户名时，用冒号分隔。
group-name	拥有该角色的组名，如果有 principal-name，组名可以写 0 个或者多个，如果没有 principal-name，组名至少要有 1 个。 注意：当应用绑定的安全域的分配组和应用中的 group-name 设置的组相同，则应用绑定的安全域中的所有用户都可以通过安全认证。需要配置多个组名时用冒号分隔。
元素/属性名	说明
resource-links	用于指定对外部资源引用的声明，最多配置一个，包含 0

	个或多个 resource-link 元素
name	被创建的资源连接的名称，相对于 java:comp/env 上下文。
global	在全局的 jndi 上下文中，连接的全局资源的名称。
type	资源类型，全类名。如：java.lang.Integer
元素/属性名	说明
loader	配置 Web 应用用来加载资源的类加载方式，最多配置一个。
delegate	当为 true 时，加载类时采用的是双亲委托机制，即先从父类加载器加载类，依次递归，如果父类加载器可以完成类加载任务，就成功返回；只有父类加载器无法完成此加载任务时，才由加载 web 应用的类加载器从 web 应用中查找和加载类。为 false 时，反之。默认 false
search-external-first	当 true 时，首先从 external-classpath 配置的路径寻找资源。false 时，从 classload 加载。默认 false。
external-classpath	该属性可以配置目录或者 jar 的路径。目录时，首先在该目录下寻找 class 进行加载；为 jar 时，将首先在该 jar 中寻找 class 进行加载，同时支持*.jar 格式，即首先在上级目录下的所有 jar 中寻找；目录或者 jar 不存在时自动跳过，多个路径使用分号分隔。
元素/属性名	说明
watched-resource	监视 web 应用的静态资源，如果被监视的资源有更新，则重部署应用。 可以包含一个或多个 property 元素。以下的 name 和 value 为 property 的属性，例如： <code><property name="web" value="WEB-INF/web.xml"></code>
name	任意有意义的名称。
Value	要监视的资源的物理路径。
元素/属性名	说明
listeners	该 listener 用于监听 Context 组件(即当前应用)的事件。通过这个 listener，可以实现在监听到 Context 的事件(例如应用启动前后，应用卸载等)发生时执行用户自定义的逻辑。可以包含 0 个或多个 listener 元素。 以下的 name 和 class-name 为 listener 的属性，例如： <code><listener name="test" class-name="com.tongweb.web.monitor.WebAppMonLifecycleListener"></code> 。
name	名称任意有意义的名称
Value	为一个实现了 com.tongweb.web.thor.LifecycleListener 接口的类。
元素/属性名	说明
jar-scanner	它用于扫描 web 应用程序的 jar 文件。
scan-all-directories	默认 false。如果为 true，到类加载路径下的所有目录

	去判断其是否是展开的 jar 文件。
scan-all-files	默认 false。如果是 true，检查所有的在类加载路径下的文件是不是 jar 文件，而不仅仅是. jar 结尾的文件。
scan-class-path	默认 false。如果为 true，除了 web 应用，所有的其他的类加载路径都会被扫描，以加载 jar 文件。
元素/属性名	说明
manager	设置 session 管理器的实现类名，session 管理器是 web 容器用来创建、维持 Web 应用的 HTTP Session 的组件，最多可以设置一个。
distributable	当为 true 时，所有的 session 属性必须实现序列化。默认为 false。
max-active-sessions	最大活跃 session 数，-1（默认）表示无限制。
max-inactive-interval	会话超时时间，默认为 30 分钟
session-id-length	session id 的长度，默认为 16。
元素/属性名	说明
session	应用中 Session 相关的属性设置
cookie-properties	设置 Session Cookie 相关属性，包括 Session Cookie 的名字，如果不设置这个属性，将使用默认值：JSESSIONID。
以下为共享库依赖配置。	
元素/属性名	说明
sharedlib-config	共享库配置的顶级元素，最多可以设置一个。例如： <pre><sharedlib-config> <shared-lib-relation/> <app-exclude/> </sharedlib-config></pre>
shared-lib-relation	关联共享库的 Group，最多可以设置一个。 例如： <pre><shared-lib-relation> <shared-lib-ref name="springlibs1"/> <shared-lib-ref name="springlibs2"/> </shared-lib-relation></pre>
shared-lib-ref	使用 name 指定已配置好的共享库名称，需要先在控制台配置。此标签可多个。如果当前 name 指定的共享库不存在，则忽略此条配置。例如： <pre><shared-lib-ref name="springlibs1"> <exclude-jar> <path>/home/userlib/service.jar</path> </exclude-jar> </shared-lib-ref></pre> 注： path 为绝对路径，可以查看%TW7_HOME%/conf/assets.xml 里面路径。
exclude-jar	配置当前关联的共享库中要排除的 jar 包。子标签 path 指定包的绝对位置。可为空。

app-exclude	配置排除当前应用自带的 jar 包，最多可以设置一个。即排除 lib/xx.jar。子标签 path 指定包的全名称。可为空。 例如： <pre><app-exclude> <path>aopalliance-1.0.jar</path> <path>commons-digester-2.1.jar</path> </app-exclude></pre>
-------------	--

注：控制台部署的共享库的配置优先 tongweb-web.xml。

1.2 . tongweb-ejb-jar.xml

在这个 ejb 的自定义配置文件中，可以定义多个 ejb-deployment 节点，而这里每一个 ejb-deployment 节点的内容，都代表着一个 ejb，可以在这个自定义部署描述文件"定制"每一个 ejb 的内容，例子如下：

```
<tongweb-ejb-jar >
  <ejb-deployment ejb-name ="" > //ejb 的名称
    <resource-link res-id="" res-ref-name=""></resource-link>
  //资源映射
    <pool></pool> //ejb 容器定制映射
    <pass-by-reference></pass-by-reference> //性能参数
  </ejb-deployment>
</tongweb-ejb-jar>
```

具体属性描述如下：

ejb-deployment 节点

元素/属性名	说明
ejb-name	ejb 名称, 对应标准部署描述符文件 (ejb-jar.xml) 中的 ejb-name 元素
resource-link	定义资源引用
pool	配置该无状态会话 bean 或消息驱动 bean 的实例池属性
pass-by-reference	Ejb 的本地调用时, 传递的复杂对象参数, 是否传递引用。设置为 true 为传递引用, 设置为 false 则为传递值

Jndi

元素/属性名	说明
name	定义 ejb 的接口的全局 jndi 名
interface	ejb2 中设置 home 接口名, ejb3 中设置 local 或者 remote 接口名。无接口的 ejb 直接使用 ejb 类名

resource-link

元素/属性名	说明
res-id	引用的自定义 jndi 名
res-ref-name	引用的全局 jndi 名

pool

元素/属性名	说明
max	池里面最大实例数
strict	是否禁止池溢出, 设置为 true 超过最大实例数, 不再分配
min	池里面最小实例数

garbage-collection	是否支持垃圾回收软引用，设置为 true，池里面的实例都是软引用
max-age	实例存在最大生存时间，单位为小时，默认为 0，永不超时
replace-aged	超过实例最大生存时间，再进行实例替换
replace-flushed	当进行池刷新时，进行实例替换
max-age-offset	延迟参数
idle-timeout	空闲超时时间，单位为分钟，默认为 0，永不超时
interval	池周期扫描时间，默认为 5 分钟

pass-by-reference

元素/属性名	说明
enabled	是否使用引用传递

1.3 .default-web.xml

com.tongweb.web.jasper.servlet.ThanosJspServlet 初始化参数说明

参数名	参数类型	默认值	说明
development	Boolean	true	Jasper 是否在开发模式中使用
fork	Boolean	true	Ant 是否应该将其 java 编译的 JSP 页面分开
keepGenerated	Boolean	true	是否要保留生成的 Java 文件
trimSpaces	Boolean	false	指令或操作之间的空格是否应该修剪
isPoolingEnabled	Boolean	true	确定是否启用标记处理程序池
mappedFile	Boolean	true	是否要支持“映射”文件？这将产生在 JSP 文件的每一行有一个 print 语句的 servlet。这似乎是一个非常好的调试功能。
classDebugEnabled	Boolean	true	是否要在类文件中包含调试信息
checkInterval	Int	0	后台编译线程检查间隔（秒）
isSmapSuppressed	Boolean	false	JSR45 调试的 SMAP 信息生成是否被禁止
isSmapDumped	Boolean	false	JSR45 调试的 SMAP 信息应该转储到文件中吗
genStringAsCharArray	Boolean	false	文本字符串是作为字符数组生成的吗
scratchDir	File		生成的 servlet 文件目录
ieClassId	String		需要像 IE 的版本 4 和 5 一样有这个。可以从因此如果它在未来发生变化，所需的只是要具有 ieClassId=" <code><value></code> " 类型的 jsp initParam 默认值：

			clsid:8AD9C840-044E-11D1-B3E9-00805F499D93
classpath	String	null	编译生成的 servlet 时应该使用什么类路径
compiler	String	null	要使用的编译器
compilerTargetVM	String	1.7	编译器目标 VM
compilerSourceVM	String	1.7	编译器源 VM
compilerClassName	String	null	编译器类名
TldCache	TldCache	null	TLD uri、资源路径和已解析文件的缓存
jspConfig	JspConfig	null	Jsp 配置信息
javaEncoding	String	UTF 8	生成 JSP 的 Java 平台编码
modificationTestInterval	int	4	修改测试间隔
recompileOnFail	Boolean	false	失败后是否立即尝试重新编译
xpoweredBy	Boolean		X-Powered-By 响应头的生成是否已启用/禁用
displaySourceFragment	Boolean	true	我们是否应该在异常消息中包含一个源片段
maxLoadedJsps	int	-1	每个 web 应用程序加载的 jsp 的最大数目。如果还有更多加载了 jsp，它们将被卸载。
jspIdleTimeout	int	-1	卸载 JSP 之后的空闲时间（秒）如果未设置或小于或等于 0，则不卸载任何 jsp。
strictQuoteEscaping	Boolean	true	JSP.1.6 应该严格应用于使用 scriptlet 定义的属性吗
quoteAttributeEL	Boolean	true	当 EL 用于 JSP 属性值时，是否引用 JSP.1.6 中描述的属性是否应用于表达式

1.4 .ejb-jar.xml

<ejb-jar>

EJB 部署描述符的根元素。它包含 ejb-jar 文件的可选描述、可选显示名称、可选小图标文件名、可选大图标文件名、所有包含的企业 bean 的必需结构信息、可选应用程序集描述符以及 ejb jar 的 ejb 客户端 jar 文件的可选名称。

<description>

ejb-jar 文件的描述性说明。

<display-name>

ejb-jar 文件显示的名称。

<small-icon>

元素包含小图标(16 x 16)的名字,这个名称是在 ejb-jar 文件的相对路径,文件必须是 jpg 或者是 gif 格式,文件名必须以.jpg.或者 gif 为后缀,图标可以被工具使用。

<large-icon>

元素包含小图标(32 x 32)的名字,这个名称是在 ejb-jar 文件的相对路径,文件必须是 jpg 或者是 gif 格式,文件名必须以.jpg.或者 gif 为后缀,图标可以被工具使用。

<enterprise-beans>

该元素包含一个或多个企业 bean 的声明。

<session>

该元素声明一个会话 bean 。该声明包括：

1. 可选的描述；
2. 可选的显示名称；
3. 可选的小图标文件名；
4. 可选的大图标文件名；
5. 分配给在部署描述中的企业 bean 的名称；
6. 会话 bean 的 home 和 remote 接口的名称；
7. 会话 bean 的实现类；
8. 会话 Bean 的状态管理类型；
9. 会话 Bean 的事务管理类型；
10. bean 的环境入口的可选声明；
11. Bean 的 EJB 引用的可选声明；
12. 安全角色引用的一个可选的声明；
13. 以及一个可选的声明 bean 的资源工厂引用。

可选的元素当它们描述的内容为空时,可以将它们省略掉。

<description>

一个简短的描述。

<display-name>

该元素中包含的目的在于通过工具将显示一个简短的名称。

<small-icon>

该元素包含含有一个小的（ 16×16）的图标图像的文件的名称。该文件的名称是在 ejb -jar 文件中的相对路径。图像必须是在 JPEG 或 GIF 格式，文件名必须以后缀结尾分别“ .JPG ”或“ .GIF ”。该图标可以使用工具。

<large-icon>

该元素包含含有一个大的（ 32 ×32）的图标图像的文件的名称。该文件的名称是在 ejb -jar 文件中的相对路径。图像必须是在 JPEG 或 GIF 格式，文件名必须以后缀结尾分别“ .JPG ”或“ .GIF ”。该图标可以使用工具。

<ejb-name>

ejb-name 元素指明了企业 bean 的名称,它是由 ejb-jar 文件的编写者命名的,这个名称在同一个 ejb-jar 文件中必须是唯一的,企业 bean 的代码不依赖于它,与它隔离的,所以企业 bean 的名称可以在应用装配的过程中改变而不会影响企业 bean 的功能.在发布描述器中的 ejb-name 与应用发布者即将分配给企业 bean 的 home 接口的 JNDI 的名称没有结构性的关系.名称必须定义的有意义。

<home>

元素包含企业 bean 的具有完全限定名(具有完整的包名)的 home 接口。

<remote>

元素包含企业 bean 的具有完全限定名(具有完整的包名)的 remote 接口。

<ejb-class>

元素包含企业 bean 实现类的完全限定名(具有完整的包名)。

<session-type>

该元素描述会话 bean 是否有状态的会话，或者无状态会话。会话型元素必须是下面两个之一：Stateful/Stateless。

<transaction-type>

该元素必须是以下两者之一：Bean/Container。

<transaction-scope>

transaction scope 元素指定企业 bean 是否要求在其方法中必须使用分布式事务，以确定是否可以使用本地事务优化。transaction scope 元素必须是以下两个元素之一：Local Distributed transaction scope 元素是可选的。如果未指定，则容器必须假定必须使用分布式事务。

<env-entry>

env-entry 元素包含企业 bean 的环境条目的声明。

该声明由一个可选的描述，环境条目的名称，以及一个可选值。

<description>

一个简短的描述。

<env-entry-name>

一个企业 bean 的环境条目名称。

<env-entry-type>

元素包含了具有完全限制的企业 bean 代码期望得到的 Java 类型值。以下是的 env-entry -型的合法值： java.lang.Boolean ， java.lang.String ， java.lang.Integer ， java.lang.Double ， java.lang.Byte ， java.lang.Short ， java.lang.Long 、 java.lang.Float。

<env-entry-value>

元素描述了一个企业 bean 的环境条目的值。

<ejb-ref>

ejb -ref 元素用于一个引用到另一个企业 bean 的 home 声明。该声明是一个可选描述;在引用的企业 bean 代码中使用 EJB 引用名称;被引用企业 bean 的预期类型， 被引用企业 bean 的预期 home 和 remote 接口， 以及一个可选的 ejb-link 信息。可选的 ejb-link 元素用来指定引用的企业 bean 。它通常用在包含组装应用程序的 ejb - jar 文件。

<description>

一个简短的描述。

<ejb-ref-name>

在 ejb - ref-name 元素包含一个 EJB 引用的名称。EJB 参照是企业 bean 环境中的一条条目。建议名称的前缀是“EJB /”。

<ejb-ref-type>

ejb-ref -type 元素包含被引用的企业 bean 的类型。ejb-ref -type 元素必须是下列之一：Entity / Session。

<home>

该元素包含企业 bean 的 home 接口的完全限定名称。

<remote>

该元素包含企业 bean 的远程接口的完全限定名称。

<ejb-link>

在 ejb -link 元素是用在 ejb-ref 元素指定 EJB 引用链接到在 ejb- jar 文件中的另一个企业 Bean 。在 ejb-link 元素的值必须是一个 enterprise bean 在同一个 ejb-jar 文件，或者在同一个 J2EE 应用程序单元的另一个 ejb-jar 文件中的 ejb-name 。

<security-role-ref>

该元素包含在企业 bean 的代码中安全角色引用的声明。该声明包含一个可选的描述，在代码中使用的安全角色名称，并定义安全角色的可选链接。在该元素的值必须是作为参数传递给 EJBContext.isCallerInRole （字符串角色名）方法的字符串。role-link 元素的值必须是在该元素中定义的安全角色之一的名称。

<description>

一个简短的描述。

<role-name>

role-name 元素包含安全角色的名称。

<role-link>

role-link 元素用来将安全角色引用链接到已定义的安全角色。role-link 元素必须包含在 security-role 元素中定义过。

<security-identity>

该元素指定调用者的安全标识是否用于的输入调用 bean 的方法执行，或者是否一个特定的 run-as identity 将被使用。

<discription>

一个简短的描述。

<use-caller-identity>

该元素指定调用者的安全标识可以用作对企业 bean 的方法执行的安全标识。

<run-as-specified-identity>

该元素指定 run-as-identity 将用于企业 bean 的方法的执行。

<description>

一个简短的描述。

<role-name>

该 `role-name` 元素包含安全角色的名称。

<resource-ref>

包含企业 bean 对外部资源的引用的声明。它由一个可选的描述、资源工厂引用名称、企业 bean 代码所期望的资源工厂类型的指示和认证类型(bean 或者 container)组成。

<description>

简短的描述

<res-ref-name>

指定资源工厂引用的名称

<res-type>

指定数据源的类型。类型由数据源期望实现的 Java 接口(或类)指定。

<res-auth>

定义是由企业 bean 的代码作为资源管理器,还是由容器作为资源管理器。发布者可以对容器提供支持,这个元素的值必须是 Application 和 Container 中的一个。

<resource-env-ref>

包含企业 bean 对与企业 bean 环境中的资源相关联的受管理对象的引用的声明。它由可选的描述、资源环境引用名称和企业 bean 代码期望的资源环境引用类型的指示组成。用于:实体、消息驱动和会话。示例:jms/StockQueue javax.jms.Queue

<description>

一个简短的描述。

<resource-env-ref-name>

指定资源环境引用的名称;它的值是企业 bean 代码中使用的环境条目名称。

<resource-env-ref-type>

指定资源环境引用的类型。

<entity>

描述 entity bean 的,这些描述性的声明有以下各项组成:

1. 可选的描述符;
2. 可选的显示名称;
3. 可选的小图标文件名;
4. 可选的大图标文件名;
5. 在发布描述中指定企业 bean 的名称;
6. entity bean 的 home 和 remote 接口的名称;
7. entity bean 的实现类;
8. entity bean 持久性管理的类型;
9. 主键类的名称;
10. Entity Bean's reentrancy 的声明;
11. 可选的容器管理的字段的列表;

12. 可选的指定的 **primary key** 的字段名;

13. **bean** 环境条目的声明;

14. **bean** 参照的声明;

15. 安全角色参照的声明;

16. 资源工厂的参照声明(可选),等等;

如果 **entity** 持久管理类型是 **container** 的话,在描述中就可以将 **primarykey-field** 元素表示出来.至少有一个 **cmpfield** 元素,而 **bean** 管理的持久性,可不包含这些属性。

<description>

一个简短的描述

<display-name>

要显示的名称,这个名字可以被别的工具引用到。

<small-icon>

包含小图标(16 x 16)的名字,这个名称是在 **ejb-jar** 文件的相对路径,文件必须是 **jpg** 或者是 **gif** 格式,文件名必须以 **.jpg** 或者 **gif** 为后缀,图标可以被工具使用。

<large-icon>

包含小图标(32 x 32)的名字,这个名称是在 **ejb-jar** 文件的相对路径,文件必须是 **jpg** 或者是 **gif** 格式,文件名必须以 **.jpg** 或者 **gif** 为后缀,图标可以被工具使用。

<ejb-name>

指明了企业 **bean** 的名称,它是由 **ejb-jar** 文件的编写者命名的,这个名称在同一个 **ejb-jar** 文件中必须是唯一的,企业 **bean** 的代码不依赖于它,与它隔离的,所以企业 **bean** 的名称可以在应用装配的过程中改变而不会影响企业 **bean** 的功能.在发布描述器中的 **ejb-name** 与应用发布者即将分配给企业 **bean** 的 **home** 接口的 **JNDI** 的名称没有结构性的关系.名称必须定义的有意义。

<home>

包含企业 **bean** 的具有完全限定名(具有完整的包名)的 **home** 接口。

<remote>

包含企业 **bean** 的具有完全限定名(具有完整的包名)的 **remote** 接口。

<ejb-class>

包含企业 **bean** 的具有完全限定名(具有完整的包名)的实现类。

<persistence-type>

指定 **entity bean** 的持久性管理的类型,它必须是下面两个之一:**Bean / Container**。

<prim-key-class>

指定了具有完全限定的 **entity bean** 的主键类的名称,如果主键类的定义依赖发布的时间,主键类元素的值应该是 **java.lang.Object**。

<reentrant>

指定一个实体 **bean** 是否可重入与否。该元素必须是两者之一: **True** 或 **False**。

<cmp-version>

该元素指定容器管理持久性的实体 bean 的版本。以下是 CMP - version 元素的合法值： 1.x 和 2.x。CMP-version 元素的默认值是 2.x。

<abstract-schema-name>

该元素指定与 cmp-version 2.x 实体 bean 对应的抽象模式名（表名），用在 EJB QL 查询。例如，实体 bean 类是 com.acme.commerce.OrderBean 命名为 OrderBeanTable。

<cmp-field>

描述容器管理的字段.包括一个可选的字段描述,字段的名称.

<description>

一个简短的描述。

<field-name>

元素指定容器管理的字段的名称,名称必须企业 bean 的公共字段或者是它的超类中的公共字段。

<primkey-field>

指定容器管理的 Bean 的主键字段.它必须是在 cmp-field 元素中声明过的,类型必须和主键类的一样,如果主键映射为多个容器管理的字段(也就是说是一个复合主键),则没必要条件使用这个元素.在这种情况下,主键类的字段必须是公共的,它们的名称必须与包含主键的 entity bean 的字段名相符合.

<transaction-scope>

元素指定企业 bean 是否要求必须使用分布式事务来实现其是否可以使用本地事务优化的方法。事务范围元素必须是以下两个元素之一:本地分布式事务范围元素是可选的。如果没有指定，容器必须假定必须使用分布式事务。

<env-entry>

元素包含企业 bean 的环境条目的声明。声明由一个可选的描述、环境条目的名称和一个可选的值组成。

<description>

简短的描述

<env-entry-name>

一个企业 bean 的 environment entry 的名称。

<env-entry-type>

元素包含了具有完全限制的企业 bean 代码期望得到的 Java 类型值,别的合法的类型值 :java.lang.Boolean, java.lang.String, java.lang.Integer,java.lang.Double, java.lang.Byte, java.lang.Short,java.lang.Long, java.lang.Float。

<env-entry-value>

描述了一个企业 bean 的环境入口的值

<ejb-ref>

对别的企业的 bean 的 home 接口的引用.描述由以下部分组成:

1. 可选的描述符。
2. 使用的引用(参照)的 EJB 的名称。
3. 引用的企业 bean 的期望的类型及 home 和 remote 接口。
4. 可选的 ejb-link 信息-指定与引用的企业 bean 有关系的 bean,它通常用在包含组装应用程序的 ejb - jar 文件。

<description>

简短的描述。

<ejb-ref-name>

引用的企业 bean 的名称,EJB 参照是企业 bean 环境中的一条条目,建议在名称前面加上“ejb/”。

<ejb-ref-type>

引用的企业 bean 的类型,它的值必须是 Entity 和 Session 两者之一。

<home>

包含被引用的企业 bean 的完全限定名(具有完整的包名)的 home 接口<remote>
包含被引用的企业 bean 的完全限定名(具有完整的包名)的 remote 接口

<ejb-link>

链接到被引用的企业 bean,它的值(有联系企业 bean)必须在同一个 ejb-jar 文件中,或者是在同一 J2EE Application 中的别的 ejb-jar 中(就是 ejb-name 元素定义的被引用的企业 bean 名称)

<security-role-ref>

元素包含企业 bean 代码中安全角色引用的声明。声明由一个可选的描述、代码中使用的安全角色名称和一个到定义的安全角色的可选链接组成。

role-name 元素的值必须是用作 EJBContext 参数的字符串,传递给 isCallerInRole (String roleName)方法。

role-link 元素的值必须是安全角色元素中定义的安全角色之一的名称。

<description>

简短的描述

<role-name>

安全角色的名称

<role-link>

一个已经定义的安全角色的连接,role-link 元素的值必须是在 security-role 元素中定义过。

<security-identity>

指定调用者的安全标识是用于执行企业 bean 的方法,还是使用特定的 run-as 标识。

<description>

简短的描述。

<use-caller-identity>

指定调用者的安全标识可以用作对企业 bean 的方法执行的安全标识。

<run-as-specified-identity>

将 run-as 标识指定为企业 bean 的方法执行所使用的标识。

<description>

简短的描述

<role-name>

安全角色的名称。

<resource-ref>

和连接相同。

<resource-env-ref>

和连接相同。

<query>

用于指定查找器或选择查询。它包含查询的可选描述、它所使用的 finder 或 select 方法的规范，以及定义查询的 EJB QL 查询字符串或查询规范。

<description>

简短的描述。

<query-method>

用于为 finder 或 select 查询指定方法。

<method-name>

包含企业 bean 的方法，或星号（ * ）字符的名称。星号时使用的元素表示企业 bean 的 remote 和 home 接口中的所有方法。

<method-params>

包含方法参数的完全限定的 Java 类名的列表。

<method-param>

方法参数的完全限定的 Java 类名。

<ejb-ql>

包含定义 Finder 或 select 的 EJB QL 查找器或选择查询字符串。此元素在查询元素的范围内定义，其内容指定使用查询的查找程序或 select 方法。内容必须是指定查询的实体 bean 的有效 EJB QL 查询字符串。如果没有指定 ejb-ql 元素，则必须使用 query-spec 元素来指定查询的语义

<query-spec>

用于指定 Finder 或 select 查询。它包含查询语义的精确描述。query-spec 元素应该只用于指定那些在 EJB QL 中无法捕获语义的查询。

<message-driven>

消息驱动元素声明了一个消息驱动 bean。声明包括：可选描述；可选显示名称；可选小图标文件名；可选大图标文件名；在部署描述符中分配给企业 bean 的名称；message 驱动 bean 的实现类；消息驱动 bean 的事务管理类型；关于 bean 的 onMessage 方法应使用分布式事务还是本地事务的可选声明；消息驱动 bean 的消息选择器的可选声明；如果使用 bean 管理的事务划分，则消息驱动 bean 的确认模式的可选声明；消息驱动 bean 的预期目标类型的可选声明；bean 的环境项的可选声明；bean 的 EJB 引用的可选声明；用于执行 bean 方法的安全标识的可选声明；bean 的资源工厂的可选声明引用；以及 bean 的资源环境引用的可选声明。

<description>

一个简短的描述。

<display-name>

该元素中包含的目的在于通过工具将显示一个简短的名称。

<small-icon>

该元素包含含有一个小的（16×16）的图标图像的文件的名称。该文件的名称是在 ejb-jar 文件中的相对路径。图像必须是在 JPEG 或 GIF 格式，文件名必须以后缀结尾分别“.JPG”或“.GIF”。该图标可以使用工具。

<large-icon>

该元素包含含有一个大的（32×32）的图标图像的文件的名称。该文件的名称是在 ejb-jar 文件中的相对路径。图像必须是在 JPEG 或 GIF 格式，文件名必须以后缀结尾分别“.JPG”或“.GIF”。该图标可以使用工具。

<ejb-name>

ejb-name 元素指明了企业 bean 的名称,它是由 ejb-jar 文件的编写者命名的,这个名称在同一个 ejb-jar 文件中必须是唯一的,企业 bean 的代码不依赖于它,与它隔离的,所以企业 bean 的名称可以在应用装配的过程中改变而不会影响企业 bean 的功能.在发布描述器中的 ejb-name 与应用发布者即将分配给企业 bean 的 home 接口的 JNDI 的名称没有结构性的关系.名称必须定义的有意义。

<ejb-class>

元素包含企业 bean 实现类的完全限定名(具有完整的包名)。

<transaction-type>

该元素必须是以下两者之一：Bean/Container。

<transaction-scope>

transaction scope 元素指定企业 bean 是否要求在其方法中必须使用分布式事务，以确定是否可以使用本地事务优化。transaction scope 元素必须是以下两个元素之一：Local Distributed transaction scope 元素是可选的。如果未指定，则容器必须假定必须使用分布式事务。

<jms-message-selector>

用于指定 jms 消息选择器，用于确定消息驱动 bean 将接收哪些消息。

<jms-acknowledge-mode>

指定 jms AUTO_ACKNOWLEDGE 或 DUPS_OK_ACKNOWLEDGE 消息确认语义,应用于使用 bean 管理的事务划分的消息驱动 bean 的 onMessage 消息。jms 确认模式元素必须是以下两个元素之一: auto-acknowledge dups-ok-acknowledge

<message-driven-destination>

消息驱动的目的地元素向部署人员提供关于消息驱动 bean 是用于队列还是主题的建议。声明包括: message 驱动 bean 的预期目的地的类型, 以及如果消息驱动 bean 被分配给主题, 则应使用持久订阅还是非持久订阅的可选声明。

<jms-destination-type>

<res-ref-name>

1.5 . persistence.xml

<persistence>

持久化配置的根元素。

<persistence-unit>

数据持久化单元, 包含 name 属性用于定义持久化单元的名字 (name 必选,空值也合法); transaction-type 指定事务类型(可选)

<description>

描述信息(可选)。

<provider>

javax.persistence.PersistenceProvider 接口的一个实现类(可选)。

<jta-data-source>

持久化提供商使用的 JTA 数据源的全局 JNDI 名称(可选)

<non-jta-data-source>

持久化提供商使用的 non-JTA 数据源的全局 JNDI 名称(可选)

<mapping-file>

声明 orm.xml 所在位置.(可选)

<jar-file>

以包含 persistence.xml 的 jar 文件为基准的相对路径,添加额外的 jar 文件.(可选)

<class>

显式列出实体类,在 Java SE 环境中应该显式列出.(可选)

<exclude-unlisted-classes/>

声明是否扫描 jar 文件中标注了@Entity 类加入到上下文.若不扫描,则如下:(可选)

<properties>

厂商专有属性列表(可选)

<property>

厂商专有属性(可选)。包含 name 和 value 属性。

2. 附录 2 TongWeb7 配置说明

2.1. EJB 配置说明

2.1.1. 无状态会话 bean 配置

配置项	中文名	默认值	参数说明	是否实时生效
access-timeout	等待超时	30 秒	从池中获取实例等待的超时时间	实时生效
max-size	最大实例数	10	池中 Bean 实例数量的最大值	实时生效
min-size	最小实例数	0	池中 Bean 实例数量的初始值和最小值	实时生效
strict-pooling	池溢出策略	true	当池中正在使用中的实例数量达到最大实例数，又有请求申请池分配实例时的实例分配策略。为 false 时，会创建一个新的实例给新来的请求，请求使用实例处理完后直接丢弃，而不在返回池中。	实时生效
max-age	实例超时时间	0 小时	实例在池中允许存活的最大的时间	实时生效
replace-aged	实例超时替换	true	当池中的实例的存活时间超过“实例超时时间”后的策略，是否被替换	实时生效
replace-flushed	刷新	false	当调用池的刷新操作时，是否更新池中的实例	实时生效
max-age-offset	创建实例延迟参数	0%	当多个实例同时创建时，避免实例同时退休，每个实例按一定的时间比例延迟退休	实时生效
idle-timeout	实例空闲超时时间	0 分钟	实例的空闲时间超过该时间，则从实例池中删除该实例	实时生效
garbage-collection	实例垃圾回收	false	开启实例垃圾回收，那么池中的实例在 jvm 中以软引用方式保存。当 jvm 内存紧张时就会回收池中的实例	实时生效
sweep-interval	扫描频	5 分	例池周期扫描池中实例，清理或	实时生

	率	钟	者替换超时、空闲超时、刷新实例。这个属性配置了实例池多久进行一次扫描	效
callback-threads	执行替换线程数	5	替换池中的实例时，会用线程池来进行替换操作。这个属性配置了线程池的线程数	实时生效
close-timeout	实例关闭超时时间	5分钟	关闭池操作的超时时间	实时生效

2.1.2. 有状态会话 bean 配置

配置项	中文名	默认值	参数说明	是否实时生效
access-timeout	等待超时	30秒	每次调用等待一个可用 bean 实例的最大时间，超过这个时间将报错	实时生效
timeout	会话空闲超时	20分钟	一个 bean 在两次调用之间能够等待的最大时间	实时生效
frequency	扫描频率	60秒	实例缓存周期扫描的时间频率	实时生效
capacity	缓存最大容量	1000	指定了 bean 容器中缓存的最大容量	实时生效
bulk-passivate	钝化实例数	100	定义每次进行钝化处理的实例数量	实时生效

2.1.3. 单例会话 bean 配置

配置项	中文名	默认值	参数说明	是否实时生效
access-timeout	等待超时	30秒	每次调用等待一个可用的 bean 实例的最大时间，超过这个时间将报错	实时生效

2.1.4. 消息驱动 bean 配置

配置项	中文名	默认值	参数说明	是否实时生效
access-timeout	等待超时	30秒	从池中获取实例等待的超时时间	实时生效
max-size	最大实例数	10	池中 Bean 实例数量的最大值	实时生效
min-size	最小实例数	0	池中 Bean 实例数量的初始值和最	实时生效

	例数		小值	效
strict-pooling	池溢出策略	true	当池中正在使用中的实例数量达到最大实例数，又有请求申请池分配实例时的实例分配策略。为 false 时，会创建一个新的实例给新来的请求，请求使用实例处理完后直接丢弃，而不在返回池中。	实时生效
max-age	实例超时时间	0 小时	实例在池中允许存活的最大的时间	实时生效
replace-aged	实例超时替换	true	当池中的实例的存活时间超过“实例超时时间”后的策略，是否被替换	实时生效
replace-flushed	刷新	false	当调用池的刷新操作时，是否更新池中的实例	实时生效
max-age-offset	创建实例延迟参数	0%	当多个实例同时创建时，避免实例同时退休，每个实例按一定的时间比例延迟退休	实时生效
idle-timeout	实例空闲超时时间	0 分钟	实例的空闲时间超过该时间，则从实例池中删除该实例	实时生效
garbage-collection	实例垃圾回收	false	开启实例垃圾回收，那么池中的实例在 jvm 中以软引用方式保存。当 jvm 内存紧张时就会回收池中的实例	实时生效
sweep-interval	扫描频率	5 分钟	例池周期扫描池中实例，清理或者替换超时、空闲超时、刷新实例。这个属性配置了实例池多久进行一次扫描	实时生效
callback-threads	执行替换线程数	5	替换池中的实例时，会用线程池来进行替换操作。这个属性配置了线程池的线程数	实时生效
close-timeout	实例关闭超时时间	5 分钟	关闭池操作的超时时间	实时生效

2.2. JDBC 配置说明

2.2.1. 连接池基本配置

属性名	中文名	默认值	属性说明	是否实时生效
name	名称	null	JDBC 资源的 jndi 名称。	实时

	(JDBC 资源的 jndi 名)			生效
jdbc-driver	数据库驱动类名	null	实现 DataSource /XADataSource API 的特定于供应商的类名。该类位于 JDBC 驱动程序中。	实时生效
jdbc-url	连接 url	null	连接数据库时所用的 url，例如： jdbc:oracle:thin:@host[:port]/service	实时生效
user-name	用户名	null	数据库的用户名	实时生效
password	密码	null	数据库的密码	实时生效
connection-properties	连接属性	null	当数据库类型为 Driver 类型时，driver 所需要的连接参数，（不是必填项，根据用户需要填写，填写的内容就是驱动的连接参数）格式必须是 propertyName=property，多个参数用分号分隔，参数 user/password 将被明确传递，所以不需要包括在这里	实时生效
class-path	驱动路径	null	数据库驱动包所在的路径（路径中包括驱动包文件）	实时生效

2.2.2. 连接池池设置

属性名	中文名	默认值	属性说明	是否实时生效
initial-size	初	10	池中连接的最小数目。该值还确定了	实

	始化连接数（最小连接数）		首次创建池置于池中的连接的数目	时生效
max-active	最大连接数	100	池中连接的最大数目。默认值为100，当池中连接数大于等于连接的最大数时，不再为请求创建连接，而是等待空闲连接。	实时生效
max-wait-time	等待超时间	30000 ms	当没有可用连接时，连接池等待连接被归还的最大时间（以毫秒计数），超过时间则抛出异常	实时生效

2.2.3. 连接池验证连接属性配置

属性名	中文名	默认值	属性说明	是否实时生效
validation-query	验证连接的sql语句	null	用于测试数据库连接的SQL语句用。	实时生效
test-on-borrow	是否获取连接时验证	false	指明是否在从池中取出连接前进行检验，如果检验失败，则将连接销毁。如果设置为 true，会根据验证间隔时间来判断是否验证，如果没有达到验证间隔时间不会对连接进行验证，validation-query 参数必须设置为非空字符串（参考 validation-interval）	实时生效
test-on-connet	是否创建连接时验证	false	设为 true 时，当创建新连接的时候进行验证，不管是否到达验证间隔时间都会进行。此时，validation-query 参数必须设置为非空字符串，当连接	实时生效

	证		池初始化时，如果验证失败将导致连接池连接清空，连接池关闭。其他时刻创建新连接验证失败时，会将连接销毁。	
test-on-return	是否返回连接时验证	false	指明是否在连接归还池之前进行检验。设置为 true，会根据验证间隔时间来判断是否验证，如果没有达到验证间隔时间不会对连接进行验证，validation-query 参数必须设置为非空字符串。验证失败时，将连接销毁。	实时生效
test-while-idle	连接有效性检查	false	是否对空闲连接进行检验。如果设置为 true，那么会根据验证间隔时间来判断是否验证，如果没有达到验证间隔时间不会对连接进行验证，如果检测失败，则将连接销毁。 validation-query 参数必须设置为非空字符串	实时生效
validator-class-name	验证类名	null	实现 com.tongweb.web.webutil.jdbc.pool.Validator 接口的类名，必须存在默认或明确的无参构造方法。将建立一个指定类的实例作为验证器，用来代替执行查询的连接验证。例如：com.mycompany.project.SimpleValidator。实际上就是自己定义一个类实现上述接口并达到验证的效果。	实时生效
log-validation-errors	打印验证失败信息	false	如果设置为 true，将在验证失败时打印失败信息	实时生效

2.2.4. 连接池的高级属性配置

属性名	中文名	默认值	属性说明	是否实时生效
jta-managed	是否开启事务连接	true	使用 JTA 事务管理的开关，如果想使用事务，必须开启事务连接	实时生效
default-auto-commit	连接池创	true	连接池创建的连接的默认的	实

	建的连接的默认的 auto-commit 状态		auto-commit 状态	时生效
time-between- eviction-runs	检查连接的周期	5000ms	连接检查的周期，泄露超时，连接有效性的检查，以及空闲超时等功能都依赖于该属性，它将决定间隔多长时间检查一次连接，判断池中连接是否泄露，空闲超时，是否有效。	实时生效
min-evictable-idle-time	空闲超时时间	60000ms	连接在池中保持空闲而不被回收的最小时间值。如果连接在池中空闲的时间超过该值，连接将被回收（销毁）。	实时生效
remove-abandoned	泄露回收	false	泄露回收功能，设为 true 时，如果连接发生泄露超时（连接占用的时间超过泄露超时时间），将连接丢弃。这样可以为写法糟糕的没有关闭连接的程序修复数据库连接。	
remove-abandoned-timeout	泄露超时时间	0s	泄露的连接可以被删除的超时值，当泄露回收功能开启时，如果连接被占用的时间超过该值，连接将被回收。这个值应该设为应用中查询执行最长的时间。	实时生效
log-abandoned	丢弃连接时打印日志	false	如果设为 true, 在丢弃泄露连接的时候打印日志。（配置该属性影响性能）	实时生效
validation-interval	连接验证时间间隔	30000ms	（以毫秒为单位）避免过度验证，保证验证不超过这个频率。如果一个连接应该被验证，但上次验证未达到指定间隔，将不再次验证。	
max-age	连接最大寿命	0	保持连接的最大毫秒数。当一个连接被归还时，如果自连接被创建创建到当前时间的时差大于该值，连接将被关闭而不是回到池中。如果设为 0，表示始终保持连接。	实时生效
jdbc-interceptors	拦截器配	null	连接池的语句跟踪，语句缓	实

	置		存, sql 日志功能都是通过该属性配置。	时生效
--	---	--	-----------------------	-----

2.3. Web 容器配置说明

2.3.1. 容器配置

属性名称	属性说明	是否实时生效
jvm-route	在使用 session 亲和的负载均衡场景中需要设置的 TongWeb 标识符。该标识符在集群中必须是唯一的。其会附加到生成的 sessionID 中。	实时生效
parameter-encoding	请求参数解码的字符集, 缺省值为 GBK。	实时生效
response-encoding	应答编码的字符集, 缺省值为 GBK。	实时生效
jsp-development	是否启用 JSP 开发模式, 默认为 true。	实时生效

2.3.2. access-log

属性名称	属性说明	是否实时生效
log-dir	日志文件位置。可以是绝对路径或相对路径（相对于 TongWeb 的安装路径），默认值为 logs。	实时生效
prefix	日志文件名的前缀。默认“access_log.”。	实时生效
suffix	日志文件名的后缀。默认“txt”。	实时生效
log-extend	选用传统日志格式，或者使用扩展的访问日志，默认为传统格式，即 false。它决定日志格式 pattern。	实时生效
pattern	定义日志格式。当 log-extend 为 false 时采用传统的方式，为 true 时采用扩展的方式。默认为“%{yyyyMMddHHmmssSSS}t %U %m %a %D”，详细格式说明见 访问日志格式配置说明 。	实时生效
file-date-format	文件日期格式，影响到文件名。默认为“yy.MM.dd.HH”表示按小时轮询，若设为 yyyy-MM-dd，表示按天轮询。	实时生效
rotatable	日志轮转开关。默认 true。	实时生效
encoding	定义写日志文件的字符集。不设则使用系统默认的字符集。	实时生效
buffered	如果为 false，每一次请求写一次日志，为 true 时则直到缓冲满才写，缓冲大小为 128000。默认为 true。	实时生效
checkexists	为 true 时每次记录日志都会判断访问日志文件是否存在，为 false 时则不检查，默认 false。	实时生效

访问日志格式配置说明

传统模式：

访问日志格式	访问日志格式说明
%a	远程主机的 IP 地址。
%A	本地 IP 地址
%b	发送字节数，包含 HTTP 请求头，如果为 0 时其值为 '-'
%B	发送字节数，不包含 HTTP 请求头
%h	远程主机名（当该通道的 enable-lookups 属性为 false 时，返回 IP 地址）
%H	请求协议
%l	已认证的远程逻辑用户名，一般返回 '-'
%m	请求方法（GET, POST, 等等）
%p	接受该请求的本地端口
%q	请求参数，（以 '?' 开头）
%r	请求的第一行（请求方法以及 request URI）
%s	HTTP 响应状态码
%S	用户的 session ID
%t	日期和时间，使用 Common Log 格式
%u	认证的远程用户，没有则返回 '-'
%U	请求的 URL 路径
%v	本地 server 名称
%D	处理请求所有时间，单位为毫秒
%T	处理请求所有时间，单位为秒（seconds）
%I	当前请求线程名

此外，它还支持写入请求/响应头，cookies，session，请求属性与时间戳格式的信息

访问日志格式	访问日志格式说明
% {xxx} i	请求头信息，xxx 为请求头属性。
% {xxx} o	响应头信息。
% {xxx} c	cookies 信息。
% {xxx} r	xxx 为 ServletRequest 的属性。
% {xxx} s	xxx 为 HttpSession 的属性。
% {xxx} t	xxx 是一个加强的 SimpleDateFormat 模式，支持所有的 SimpleDateFormat 格式，并且支持额外的属性。
% {xxx} t	额外支持的属性。

两个简写配置：

访问日志格式	访问日志格式说明
common	%h %l %u %t \"%r\" %s %b

combined	%h %l %u %t \"%r\" %s %b \"% {Referer}i\" \"%{User-Agent}i\"
----------	--

扩展模式:

扩展模式的访问日志其实并不是对原来访问日志的扩展，而是以不同的 pattern 对请求/响应进行记录。从属性来看，它只少了个“locale”属性。以及 pattern 采用不同的值。这里的 pattern 是由一些格式符号组成，有一些符号须要一些额外的前缀。一般的前缀有：“c”表示客户端即远程。“s”表示服务器端即本地，“cs”表示客户端到服务器端。“sc”表示服务器端到客户端。“x”表示“application specific”

访问日志格式	访问日志格式说明
bytes	发送字节，不包括 HTTP headers，如果为零显示为‘-’
c-dns	远程主机名（或者 IP 地址，如果连接器的 enableLookups 为 false）
c-ip	远程（客户端）IP 地址
cs-method	请求方法（GET, POST 等）
cs-uri	请求的 URL
cs-uri-query	请求参数（如果存在，以“?”开头）
cs-uri-stem	请求的 URL 路径
date	yyyy-mm-dd format 格式的 GMT 日期
s-dns	本地主机名
s-ip	本机 ip 地址
sc-status	响应状态
time	该请求的时间
time-taken	从开始到结束，服务器在该请求上所花的时间（以秒为单位）
x-threadname	当前请求线程名

可以用 x-H(XXX) 表示 HttpServletRequest 的方法。

x-H(authType)
x-H(characterEncoding)
x-H(contentLength)
x-H(locale)
x-H(protocol)
x-H(remoteUser)
x-H(requestedSessionId)
x-H(requestedSessionIdFromCookie)

x-H(requestedSessionIdValid)
x-H(scheme)
x-H(secure)

也支持从 headers cookies, context, request 或 session 属性和 request 参数中获取信息:

访问日志格式	访问日志格式说明
cs(XXX)	请求头
sc(XXX)	响应头
x-A(XXX)	servlet 上下文属性
x-C(XXX)	具有专用名的首个 cookie 值。例如, 要记录一个叫做 JSESSIONID, 则指定该属性为 x-C(JSESSIONID)。
x-O(XXX)	将所有名称为 XXX 的响应头串接
x-P(XXX)	请求参数
x-R(XXX)	request 属性
x-S(XXX)	session 属性
cs(XXX)	请求头
sc(XXX)	响应头
x-A(XXX)	servlet 上下文属性
x-C(XXX)	具有专用名的首个 cookie 值。例如, 要记录一个叫做 JSESSIONID, 则指定该属性为 x-C(JSESSIONID)
x-O(XXX)	将所有名称为 XXX 的响应头串接

2.3.3. virtual-host

属性名称	属性说明	是否实时生效
name	虚拟主机的唯一标识。	实时生效
alias	虚拟主机的别名, 可以有多个, 用逗号分隔。	实时生效
status	虚拟主机的运行状态, 启动或停止。默认值为 true。	实时生效
accesslog-enabled	访问日志开关。默认 true。	实时生效
listeners	与该虚拟主机关联的通道名称的列表。	实时生效
sso-enabled	是否开启单点登录。默认为 false。	实时生效
remote-filter-enabled	远程过滤开关。默认 false。	实时生效
accesslog-dir	访问日志文件存放路径。	实时生效
addValve	用户自定义 Valve。	实时生效

2.3.3.1. Remote-filter

属性名称	属性说明	是否实时生效
------	------	--------

allow-addr	允许访问的远程 IP（正则表达式或者具体的 IP 地址）	实时生效
deny-addr	禁止访问的远程 IP（正则表达式或者具体的 IP 地址）	实时生效
allow-host	允许访问的远程主机（正则表达式或者具体的主机名称）	实时生效
deny-host	禁止访问的远程主机（正则表达式或者具体的主机名称）	实时生效
filter-host-type	远程主机的过滤类型（允许/禁止）	实时生效
filter-addr-type	远程 IP 的过滤类型（允许/禁止）	实时生效

2.3.4. http-listener

属性名称	属性说明	是否实时生效
name	HTTP 通道名称，唯一标识。	实时生效
status	表示 HTTP 通道的运行状态，启动或停止，默认为启动（started）。	实时生效
port	HTTP 通道的监听端口。	实时生效
address	HTTP 通道的监听地址，可以为 0.0.0.0 或非 0.0.0.0，不指定时表示监听本机所有地址（0.0.0.0）。	实时生效
ssl-enabled	表示是否使用 SSL 协议。为 true 时表示使用，默认为 false。	实时生效
redirect-port	非 ssl 到 ssl 的重定向端口，默认为 443。	实时生效
uri-encoding	默认 GBK，用于解码 URI 字节的编码格式。	实时生效
use-body-encoding-for-uri	如果 contentType 中指定了编码规范，则可以不使用 URLEncoder。默认为 false。	实时生效
max-parameter-count	被容器自动解析的参数/值对（GET&POST）的最大值。参数/值对超出这个范围，超出的部分将被忽略。默认 10000。	实时生效
max-post-size	Form 表单使用 Post 方法时可提交的最大字节数，0 表示禁用此属性，默认为 2097152（2M）。	实时生效
parse-body-methods	逗号分隔的 http 方法列表，用于 rest 风格。不可以是 TRACE。默认支持 GET 和 POST。	实时生效
default-virtual-host	此通道对应的默认虚拟主机。	实时生效
proxy-url	为服务器配置代理，包括	实时生效

	proxyName&proxyPort 。 形 式 为 “ip:port” 。	
io-mode	io 模式	实时生效
http2-enabled	开启 http2 协议功能	实时生效

2.3.4.1. ssl

属性名称	属性说明	是否实时生效
alias	服务器端证书别名。默认 twnt。	实时生效
keystore-file	证书存放路径。默认位置为 conf/server.keystore。	实时生效
keystore-pass	证书密码。	实时生效
keystore-type	证书类型。	实时生效
client-auth	客户端认证。如果为 true，需要一个来自 client 证书的认证。默认 false。	实时生效
ssl-protocol	设置使用的 SSL 协议，如果不设置，默认值是 TLS。可以从 JVM 的文档中查看创建 SSLContext 对象时，可以使用的算法的值。	实时生效
truststore-file	信任证书，用来认证客户端证书。	实时生效
truststore-pass	信任证书密码。	实时生效
truststore-type	信任证书类型。	实时生效
openssl-enabled	开启 openssl 功能	实时生效

2.3.4.2. protocol

属性名称	属性说明	是否实时生效
not-allow-HTTP-methods	要禁用的 HTTP 请求方法。	实时生效
max-threads	通道可创建的最大线程数，一个线程处理一个请求。默认为 200。	实时生效
min-spare-threads	最小备用线程。即启动时初始化的线程数。默认 10。	实时生效
connection-timeout	网络 (socket) 连接超时，默认 60000ms。设置为 -1 表示永不超时。	实时生效
keep-alive-timeout	通道等待其它 AJP/HTTP 请求的时间，超时则关闭连接。如果不设置则使用 connection-timeout 的值，-1 表示永不超时。	实时生效
processor-cache	该设置用于设置缓存对象的个数。默认为 200，-1 表示无限制。	实时生效

tcp-no-delay	设置 ServerSocket 的 TCP_NO_DELAY 属性，在多数场景下可以提高性能。默认为 true。	实时生效
xpowered-by	是否在 reponse 的 http 响应头里生成 X-Powered-By 信息。默认 false。	实时生效
use-ipv-hosts	基于 ip 的虚拟主机，默认为 false。	实时生效
enable-lookups	设为 true 时，可以调用 request.getRemoteHost 去执行 DNS lookups 以返回远程客户端的真实 host name。默认 false。	实时生效
async-timeout	异步请求超时，默认为 10000ms。	实时生效
backlog	指定当所有可以使用的处理请求的线程数都被使用时，可以放到处理队列中的请求数，就是被排队的请求数，超过这个数的请求将拒绝连接。默认为 100。	实时生效
max-header-count	容器允许的最大请求头个数。当请求中包含头字段个数大于该值时，请求会被拒绝。值小于 0 表示没有限制。默认值为 100。	实时生效
accept-thread-count	用于接受连接的线程数。该值默认为 1。当有大量短连接或使用多 CPU 的机器时，都需要增大该值，一般情况下，真正使用的不会超过 2 个。配置最大值不要超过 4。	实时生效
allow-host-referer-header		实时生效
allow-ip-referer-header		实时生效

2.3.4.3. http-options

属性名称	属性说明	是否实时生效
compression	是否传输压缩，可选值为 on, off, force。默认是 on。	实时生效
compressable-mime-type	用于 http 压缩的 MIME 类型列表（各类型间用逗号分隔）。默认 text/html, text/xml, text/plain。	实时生效
compression-min-size	启用传输压缩的内容最小值，缺省值是 2048 字节 (2K)。	实时生效
disable-upload-timeout	是否为数据上传指定更长的连接超时时间。默认 true。	实时生效
max-http-header-size	请求与请求头的最大值，bytes 表示，默认是 8192 (8k)。	实时生效
max-keep-alive-requests	默认为 100, http 协议就是在 timeout 时间内又有新的连接过来，同时 max 会自动减 1，直到为 0，默认为 100。	实时生效
restricted-user-agents	值为正则表达式，用于匹配 user-agent Header 指定哪些	实时生效

	HTTP clients 不使用 keep alive。默认为空字符串，表示不限制。	
connection-upload-timeout	数据上传时，专门指定的连接超时时间。如果 disable-upload-timeout 为 false 时生效。	实时生效
no-compression-user-agents	此值为一个正则表达式，用于匹配 user-agent Header 指定哪些 HTTP clients 不使用 compression。默认为空字符串，表示没有限制。	实时生效

2.3.4.4. advance

属性名称	属性说明	是否实时生效
disable-keep-alive-percentage	为提高可扩展性，在长连接失效之前被使用的处理线程的百分比。小于 0 的值将被设为 0，大于 100 的值将被设为 100。默认值是 75。	实时生效
selector-timeout	选择轮询器 selector 的超时时间（以毫秒为单位）。这个值非常重要，因为连接清理工作也是在同一个线程里的，所以不要将此值设置的太高。默认值是 1000 毫秒。	实时生效
usecomet	是否允许使用 Comet servlet，默认为 true。	实时生效
use-sendfile	是否使用 sendfile 属性，默认为 true，如果通道支持 sendfile 特性，在静态资源大于 48K 时，sendfile 特性要优先于 compression 生效，此时资源不会被压缩，只有资源小于 48K 才会采用通道的 GZIP 压缩。使用压缩功能（节省带宽）还是使用 sendfile（节省 CPU 周期）特性需要做出权衡。该特性默认开启，如果要关闭，可以在 TW_HOME/conf/web.xml 或者在 WEB 应用的 web.xml 中设置该属性中设置 sendfileSize 属性为负数。	实时生效
oom-parachute	如果该参数大于 0，通道启动时会创建相应大小的 bytes 数组，做为数据块保留。默认大小为 1M。当 OOM(OutOfMemoryError) 发生	实时生效

	<p>时，数据块会被释放，此时 JVM 有更多的内存空间去执行清理操作，尽可能使应用服务器完成请求，但不保证能成功。所以，该参数主要是为了先占用一部分内存，在 OOM 产生时，释放内存，并清理通道内保存的一些组件内容，使应用服务器能继续工作而不是直接宕掉。注意，它只关注 java heap space 的 OOM，不保证能够恢复。它对 java heap 以外的 OOM 不起作用。</p>	
--	--	--

2.3.5. ajp-listner

属性名称	属性说明	是否实时生效
name	AJP 通道名称，唯一标识。	实时生效
status	表示 AJP 通道的运行状态，启动或停止，默认为启动 (started)	实时生效
port	AJP 通道的监听端口。	实时生效
address	AJP 通道的监听地址，可以为 0.0.0.0 或非 0.0.0.0，不指定时表示监听本机所有地址 (0.0.0.0)。	实时生效
block-enabled	通道是否使用阻塞 socket 处理请求，默认为 true。	实时生效
redirect-port	非 ssl 到 ssl 的重定向端口。	实时生效
uri-encoding	默认 GBK，用于解码 URI 字节的编码格式。	实时生效
use-body-encoding-for-uri	如果 contentType 中指定了编码规范，则可以不使用 URLEncoder。默认为 false。	实时生效
max-parameter-count	被容器自动解析的参数/值对 (GET&POST) 的最大值。参数/值对超出这个范围，超出的部分将被忽略。默认 10000。	实时生效
max-post-size	Form 表单使用 Post 方法时可提交的最大字节数，0 表示取消限制，默认为 2097152 (2M)。	实时生效
parse-body-methods	逗号分隔的 http 方法列表，用于 rest 风格。不可以是 TRACE。默认支持 GET 和 POST。	实时生效
proxy-url	为服务器配置代理，包括 proxyName&proxyPort。形式为“ip:port”。	实时生效
default-virtual-host	默认虚拟主机	实时生效
io-mode	AJP 通道的 io 模式	实时生效

create-time	AJP 通道的创建时间	实时生效
protocol	protocol	实时生效

2.3.5.1. ajp-options

属性名称	属性说明	是否实时生效
require-secret	只有负载均衡器发送的请求中包含了指定的 secret keyword (加密关键字)，请求才会被应用服务器接收，否则返回 403 (访问拒绝)。要使用此功能，需要在负载均衡器中配置与此相关的属性，且其值与此处的 require-secret 的值相同。 例如负载均衡器为 apache，可以在其 worker.properties 文件中设置“secret”属性 (这个属性的值就是所谓的 secret keyword)，设置其值与此处的配置相同即可。	实时生效
packet-size	ajp packet 的最大值，byte 类型。应该与 mod_jk 的 max_packet_size 的值相同，默认 8192，如果设的值小于 8192，则采用默认值。	实时生效

2.4. 日志服务配置

2.4.1. 日志服务配置属性

配置项	中文名	默认值	参数说明	是否实时生效
rotation-limit	轮转文件大小	50 MB	如果开启了日志轮转并且这个值设置大于 0，当日志文件大小达到这个值后，轮转日志文件。轮转日志格式为“日志名称_date_count.后缀”，其中 date 为日志轮转的那天，count 是同一 date 内多次轮转时第几次轮转的序号。	实时生效
rotation-timelimit	轮转文件时间	0	如果开启了日志轮转并且这个值设置大于 0，每隔这个属性配置的时间，就进行一次日志文件的轮转。单位为分钟。轮转日志格式为“日志名称_date_count.后缀”，其中 date 为日志轮转的那天，count 是同一 date 内多次轮转时第几次轮转的序号。	实时生效
rotation-file-count	轮转文件个数	20	日志文件轮转时，日志轮转文件的最大个数。	实时生效
rotation	日志轮转	false	是否开启日志轮转	实时生效
verbose	控制台日志	true	日志是否向控制台输出	实时生效

rotation-by-day	当天 轮转	false	如果开启该功能，则生成的日志文件内不会包含不同天的日志。例如，如果没有开启周期轮转且每天生成的日志文件大小没有超过配置，则每个轮转文件内的日志为当天 0 点到 23:59 一整天的日志。	实时生效
file	系统日志 文件路径	\${tongweb.root}/logs/server.log	系统日志文件的路径。	实时生效
log-format	日志记录的格式	[%d{yyyy-MM-dd HH:mm:ss}] [%p] [%c] [%m] %n	系统日志的记录格式。	实时生效

2.5. 事务配置

2.5.1. 事务配置属性

配置项	中文名	默认值	参数说明	是否实时生效
transaction-timeout-in-seconds	事务超时时间	0	全局事务的超时时间，以秒为单位，0 表示永不超时	否，只提供在 tongweb.xml 中配置，重启服务器生效

2.6. JSF 配置

2.6.1. JSF 配置属性

配置项	中文名	默认值	参数说明	是否实时生效
jsf	是否关闭服务	false	server 节点中配置该属性。是否关闭应用服务器的	否，只提供在 tongweb.xml

	器 jsf 实现		faes 实现，默认为 false。 配置为 true 时，使用应用自身的 jsf 实现。为 false 时使用应用服务器的 jsf 实现。	l 中配置， 重启服务器 生效。
--	----------------	--	---	------------------------

3. 附录 3 常见问题说明

3.1. 跨域访问技术 CORS 配置说明

CORS 是一个 W3C 标准，全称是“跨域资源共享”（Cross-origin resource sharing）。

它允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而克服了 AJAX 只能同源使用的限制。

TongWeb 内部提供了 CORS Filter 过滤器是 W3C 的 CORS（跨源资源共享）规范的一种实现，它是跨源请求的机制。该过滤器的工作原理是将所需的 Access-Control-* 标头添加到 HttpServletResponse 对象。

CORS 过滤器的过滤器类名是 com.tongweb.catalina.filters.CorsFilter。使用此过滤器所需的最少配置是：

```
<filter>
  <filter-name>CorsFilter</filter-name>
  <filter-class> com.tongweb.catalina.filters.CorsFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>CorsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

此外，Filter 还支持一些参数配置如下：

属性	描述
cors.allowed.origins	允许访问资源的源列表。可以指定*表示接受任意域名的请求。 否则，可以提供逗号分隔的起源白名单。 例如： http://www.w3.org,https://www.apache.org。
cors.allowed.methods	以逗号分隔的 HTTP 方法列表，可使用跨源请求来访问资源。默认值：GET，POST，HEAD，OPTIONS
cors.allowed.headers	跨域允许包含的头，默认值：Origin，Accept，X-Requested-With，Content-Type，Access-Control-Request-Method，Access-Control-Request-Headers
cors.exposed.headers	CORS 请求时，除了 6 个简单的响应标头： <ul style="list-style-type: none"> • Cache-Control • Content-Language • Content-Type • Expires • Last-Modified • Pragma 如果想拿到其他字段，就必须在该字段里面指定。例如 Content-Length，如果是多

	个用逗号分割。
<code>cors.preflight.maxage</code>	这个响应首部表示 preflight request（预检请求）的返回结果（即 Access-Control-Allow-Methods 和 Access-Control-Allow-Headers 提供的信息）可以被缓存多久。 返回结果可以用于缓存的最长时间，单位是秒。如果值为 -1，则表示禁用缓存，每一次请求都需要提供预检请求，即用 OPTIONS 请求进行检测。
<code>cors.support.credentials</code>	请求的认证信息通常存在 cookie 中，这个字段的值是一个布尔值，表示是否允许发送 Cookie。默认情况下，Cookie 不包括在 CORS 请求之中。设为 true，即表示服务器明确许可，Cookie 可以包含在请求中，一起发给服务器。默认值：false。
<code>cors.request.decorate</code>	用于控制是否应将 CORS 特定属性添加到 HttpServletRequest 对象的标志。默认值：true

下面是覆盖默认值的一个比较全的配置：

```

<filter>
  <filter-name>CorsFilter</filter-name>
  <filter-class>com.tongweb.catalina.filters.CorsFilter</filter-class>
  <init-param>
    <param-name>cors.allowed.origins</param-name>
    <param-value>*</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.methods</param-name>
    <param-value>GET,POST,HEAD,OPTIONS,PUT</param-value>
  </init-param>
  <init-param>
    <param-name>cors.allowed.headers</param-name>
    <param-value>Content-Type,X-Requested-With,accept,Origin,Access-Control-Request-Method,Access-Control-Request-Headers</param-value>
  </init-param>
  <init-param>
    <param-name>cors.exposed.headers</param-name>
    <param-value>Access-Control-Allow-Origin,Access-Control-Allow-Credentials</param-value>
  </init-param>
  <init-param>
    <param-name>cors.support.credentials</param-name>
    <param-value>>true</param-value>
  </init-param>
  <init-param>
    <param-name>cors.preflight.maxage</param-name>
    <param-value>10</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>CorsFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>

```

3.2. TongWeb 及其集中管理工具对 IPv6 支持

一、如果需要服务器绑定 IPv6 类型的 jmx 地址则需要服务器启动脚本中指定实际的 IPv6 地址。需要在 TW_HOME/bin/ external.vmoptions 中添加参数的方式来支持，配置如下（注意地址需要用中括号括起来）：

```
-Djava.rmi.server.hostname=[2001:da8:2004:1000:202:116:160:41]
```

二、另外集中管理工具如果 master 的绑定地址为 IPv6，指向需要将 agent 的配置文件中（agent.xml）进行相应的修改来指定 master 地址，如：

```
<masterIp>[2001:0:53aa:64c:c73:7f7:c26b:35dd]</masterIp>
```

如果集中管理工具在管理 agent 时希望以 IPv6 地址进行注册时，则需要在 agent.xml 中指定 agentIp 如：

```
<agentIp>[2001:0:53aa:64c:c73:7f7:c26b:35dd]</agentIp>
```

三、如果希望配置通道地址或虚拟机允许、禁止地址为 IPv6 类型，则可以直接填写实际的 IPv6 地址（不需要加中括号）。如：2001:da8:2004:1000:202:116:160:41。

四、JDBC 创建及编辑页面提供 IPv4 及 IPv6 模板。

3.3. TongWeb7 支持 Invoker Servlet

问题描述：TongWeb7 如何配置 Invoker Servlet

解决办法：支持以下两种配置方式。

- 1、在 TongWeb7 安装目录下的 conf/default-web.xml 中配置（推荐）
- 2、在应用的 web.xml 中配置

配置内容如下：

```
<servlet>
  <servlet-name>invoker</servlet-name>
  <servlet-class>
    com.tongweb.catalina.servlets.InvokerServlet
  </servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>0</param-value>
  </init-param>
  <load-on-startup>2</load-on-startup>
</servlet>
<servlet-mapping>
  <servlet-name>invoker</servlet-name>
  <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

其中 debug 参数大于 0 则在访问 Invoker Servlet 时打印详细的日志信息。

3.4. TongWeb7 对 JDK9 以上版本支持说明

TongWeb7 目前支持 JDK1.7 到 JDK11，其中 JDK 的版本大于 1.9 则需要注意以下配置：

- 1.如果需要在 external.vmoptions 或管理控制台增加 JDK9 及以上的 JVM 参数，键值必须是以“=”进行连接，如：--add-opens=java.base/java.lang=ALL-UNNAMED，参数中间不支持空格的写法。
- 2.APM 工具在启动脚本的配置在 JDK9 以上有配置变化，详情请见 APM 使用手册配置说明。

4. 附录 4 开机自启动服务安装工具说明

通过运行该工具可以将 TongWeb 配成 windows 服务，在 Linux 下自动配成开机自启动。

使用方法:

进入通过安装程序安装好的 TongWeb7 的 bin 目录下, 通过命令行执行下列命令:

Windows: installservice.bat

Linux: sh installservice.sh

注:

redhat 系统中, 自启动脚本添加到/etc/init.d/StartTongWeb;

suse 系统中, 自启动脚本添加到/etc/profile.d/ StartTongWeb.sh;

solaris 系统中, 自启动脚本添加到/etc/rc3.d/StartTongWeb;

aix 系统中, 自启动脚本添加到/etc/inittab;

hp 系统中, 自启动脚本添加到/sbin/rc3.d/StartTongWeb;

注意: 如果自启动不了, 1、确认当前操作系统版本对于如上目录是否是自启动。(可以手动执行如上目录确认)。2、确认要启动的 TongWeb 能否正常启动, 可以手动启动确认或者查看日志文件看启动日志。

关于 windows 服务:

通过开机自启动服务安装工具只是将 TongWeb 配成了 windows 服务, 服务后续的启动、停止和删除都需要通过命令行或者 windows 服务。

TongWeb 在 windows 平台提供以服务方式启动。支持 64 位和 32 位系统。

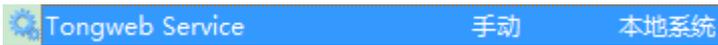
在安装服务之前, 首先需要运行 TongWeb7_HOME/service 目录下的 setEnv.bat。

然后操作方法如下:

- 1) 在 TongWeb7_HOME/service/bin 目录下运行 TongwebService.exe -install, 注册 TongWeb 服务。注册后用户可查看 window 服务是否已经增加了名为“Tong ”的服务(服务名称可配置, 默认为“Tong”)。【也可以通过安装工具完成】
- 2) 在 TongWeb7_HOME/service/bin 目录下运行 TongwebService -start, 启动 TongWeb 服务;
- 3) 在 TongWeb7_HOME/service/bin 目录下运行 TongwebService -stop, 停止 TongWeb 服务;
- 4) 在 TongWeb7_HOME/service/bin 目录下运行 TongwebService -uninstall, 卸载 TongWeb 服务;

相关参数说明:

TongWeb7_HOME/service/conf/tw.xml 中的参数说明如下所示:

Service 子 标签名称	说明
id	服务的名称
Name	服务的显示名称, 也就是通过 Windows 的服务管理查看到的服务名称。如果配置成: <pre><id>TongwebService</id> <name>Tongweb Service</name></pre> 那么在服务管理中将会如下图所示:  查看服务属性如下图所示: 服务名称: TongwebService 显示名称: Tongweb Service
description	对服务的描述
logpath	日志路径。服务的安装启动日志路径配置如下:

	<logpath>TongWeb7_HOME\log</logpath> 可以修改为指定路径。
timeout	设置的服务器启动超时时间，超过时间时服务器还未启动成功，则服务器停止启动。单位是毫秒。
startargument	启动 TongWeb 的各项参数
stopargument	停止 TongWeb 的各项参数

作用范围说明：如果不存在域，就启动/停止该 TongWeb，如果存在域，会启动/启动域中所有 TongWeb。

5. 附录 5 TongWeb7 使用 jmxmp 协议

TongWeb 的 JMX 支持两种协议，分别是 rmi 协议和 mp 协议。默认是 rmi 协议。其切换的开关在 TongWeb 的配置文件 tongweb.xml 中。其中关于 jmx 的描述中：

```
<jmx-service port="7200" address="127.0.0.1" protocol="rmi"/>
```

这里 protocol 指定了 TongWeb 启动时所使用的协议。如果需要使用 jmxmp 协议。可以在这里配置成：

```
<jmx-service port="7200" address="127.0.0.1" protocol="mp"/>即可。
```

6. 附录 6 使用命令行安装 TongWeb

此命令行工具的用途是在同一台机器上一次安装多个 TongWeb，并保证一次安装的 TongWeb 之间端口互不冲突。

附：多次安装间的端口冲突不在验证范围内。

Linux/Unix 环境下不支持 TongWeb 安装路径中带空格。会引起启动失败的问题。

使用方法：

进入通过安装程序安装好的 TongWeb 的 bin 目录下，通过命令行执行下列命令：

Windows：

```
installTongWebs.bat num
installTongWebs.bat num pathPrefix
installTongWebs.bat configFilePaht
```

Linux/Unix：

```
sh installTongWebs.sh num
sh installTongWebs.sh num pathPrefix
sh installTongWebs.sh configFilePaht
```

如下：以 Linux/Unix 为例说明各个命令：

installTongWebs.sh num

功能说明	使用默认路径和默认名称安装 TongWeb，默认路径为命令行所在 TongWeb 的上一层目录，默认名称为 TongWeb-N，N 为当前安装的第 N 个 TongWeb。
参数说明	num: 要安装的 TongWeb 的个数，范围为 1-100 的正整数。
实例	sh installTongWebs.sh 9 命令行所在 TongWeb 的路径为/home/tong/TongWeb
安装结果	在/home/tong/目录下，有 9 个名为 TongWeb-N 的安装了 TongWeb 的文件夹，N 为 1-9。 TongWeb-N 包含命令行所在 TongWeb 的全部目录，并重新生成配置文件 tongweb.xml 的几个必需的初始化端口，比如控制台端口、默认 HTTP 通道端口、HTTPS 通道端口、AJP 通道端口、EJB 通道端口、JMX 端口、服务器停止端口、JMS 端口，保证一次安装过程中的所有

	TongWeb 之间端口互不冲突，且与机器中正在运行的程序所用端口互不冲突。
输出文件	<p>在对应安装路径下可以看到安装成功的各个 TongWeb，并可启动成功。在命令行所在 TongWeb 的 logs 目录下会看到本次安装的日志文件 domain.log，其中记录了各个 TongWeb 的安装信息。</p> <p>文件格式：</p> <pre>/home/TW- 1, consolePort:9061, httpPort:8081, httpsPort:8444, ajpPort:8010, ejbPort:5101, jmxPort:7201, shutdownPort:8006, jmsPort:61667 /home/TW- 2, consolePort:9062, httpPort:8082, httpsPort:8445, ajpPort:8011, ejbPort:5102, jmxPort:7202, shutdownPort:8007, jmsPort:61668</pre>
约束条件	<p>使用该命令行工作有如下几点需要注意：</p> <p>使用命令行前提是同一台机器上已有至少一个安装成功的 TongWeb 且该 TongWeb 可以正常启动运行，然后可使用该 TongWeb 的命令行进行安装。</p> <p>对端口的自动配置保证不冲突的控制范围在以命令行所在 TongWeb 的一次安装中，并不对多次安装间进行控制。</p>

installTongWebs.sh num pathPrefix

功能说明	使用 pathPrefix 安装 TongWeb，pathPrefix 包含路径和名称前缀安装。其中，路径为 TongWeb 的安装目录，名称前缀-N 为 TongWeb 的名称。
参数说明	<p>num: 指要安装的 TongWeb 的个数，范围为 1-100 的正整数</p> <p>pathPrefix: 为一个合法的绝对路径，且路径中不能包含空格。</p>
实例	<p>sh installTongWebs.sh 9 /home/tongweb/TW</p> <p>命令行所在 TongWeb 的路径为/home/tong/TongWeb7</p>
安装结果	<p>在/home/tongweb 目录下，有 9 个名称为 TW-N 的安装了 TongWeb 的文件夹，N 为 1-9。</p> <p>文件夹内容同 5.1 安装结果中的文件夹内容。</p>
输出文件	同 5.1 的输出文件
约束条件	同 5.1 的约束条件

installTongWebs.sh configFilePaht

功能说明	使用 installTongWebs.xml 中配置的 TongWeb 安装信息进行安装
参数说明	<p>configFilePath 包含安装使用的配置文件 installTongWebs.xml 的全路径。</p> <p>installTongWebs.xml 中安装信息包括目标安装路径、控制台端口、默认 HTTP 端口、HTTPS 端口、AJP 端口、EJB 端口、JMX 端口、服务器关闭端口、JMS 端口。</p> <p>其中，目标安装路径是必须的、不可置空的，由字母数字，中横线“-”，下划线“_”，小数点“.”和空格组成的系统不存在的合法路径，且相互之间不重复，且路径中不能包含空格。</p> <p>端口非必须的、可置空的，是 1-65535 范围内的不与正在运行的程序冲突的端口值，且相互之间不重复。</p>
实例	<p>sh installTongWebs.sh /home/tongweb/installTongWebs.xml</p> <p>installTongWebs.xml 内容：</p> <pre><?xml version="1.0" encoding="UTF-8"?> <install-tongwebs> <tongweb destinate-path="/home/TW1" console-port="9061" http-port="" https-port="" ajp-port="" ejb-port="" jmx-</pre>

	<pre>port="" shutdown-port="" jms-port="" /> <tongweb destinate-path="/home/TW2" console-port="" http- port="8082" https-port="" ajp-port="" ejb-port="" jmx-port="" shutdown-port="" jms-port="" /> </install-tongwebs> 命令行所在 TongWeb 的路径为/home/tong/TongWeb7</pre>
安装结果	<p>在/home 目录下，有 2 个名称分别为 TW1 和 TW2 的已经安装了 TongWeb 的文件夹。</p> <p>文件夹内容同 5.1 安装结果中的文件夹内容。</p>
输出文件	同 5.1 的输出文件
约束条件	同 5.1 的约束条件

7. 附录 7 jmstool 命令使用说明

功能说明	将 ActiveMQ 对象绑定到默认的名字服务。
相关参数说明	<p>以 windows 为例：</p> <pre>jmstool.bat type name port user password</pre> <p>type:物理资源类型，必选。包括如下几种：</p> <p>Q: 创建队列。</p> <p>T: 创建主题。</p> <p>CF: 创建连接工厂。</p> <p>XACF: 创建事务连接工厂。</p> <p>name:物理资源名称，唯一标识，不可重复。必选。</p> <p>port: 要使用的本地 activemq 的 port，可选。默认为 61616。</p> <p>user:用户名，可选。</p> <p>password: 密码，可选。</p> <p>如果是通过 ssl 连接，在上述使用上加 ssl 即可。即</p> <pre>jmstool.bat ssl type name port user password</pre> <p>其中 ssl 连接方式中 port 端口默认为 61617</p>
实例	<pre>jmstool Q testd jmstool CF testc</pre>
验证方式	<p>以上面的实例为例：</p> <p>Windows 系统：</p> <p>在%TongWeb7_HOME%\apache-activemq\conf 目录中找到.bindings 文件，查看文件中有名称为 testd 和 testc 的绑定信息。</p> <p>Linux 系统：</p> <p>在\${TongWeb7_HOME}/apache-activemq/conf 目录中找到.bindings</p>

	文件，查看文件中有名称为 testd 和 testc 的绑定信息。
注意事项	在 Linux 系统中不允许使用带有字符 “\” 的名称，如：jmstool CF jms\testc 或 jmstool CF jms\\testc 等。

8. 附录 8 参数说明

服务器启动脚本包括 TongWeb7_HOME/bin/startserver.bat、startserver.sh、stopserver.bat 和 stopserver.sh，脚本中支持的主要参数如下所示：

TongWeb7 参数说明如下：

参数名称	参数说明
-Dtongweb.java="%JAVA_HOME%"	JDK 安装的根目录。当用户需要更换 TongWeb7 使用的 jdk 路径的时候，可以设置这个参数。
-Dtongweb.upload	临时目录下的存放所上传的应用的目录。 通常为 %TongWeb7_HOME%\temp\upload。当用户需要更换其他路径作为上传应用目录的时候，可以对此参数进行设置。
-Dtongweb.app	已部署应用的目录，在脚本中必须设置为 %TongWeb7_HOME%\deployment。
-Dtongweb.sysapp	应用服务器的系统应用的目录，例如管理控制台 console 就在这个目录下。 在脚本中必须设置为 %TongWeb7_HOME%\applications。
-Dtongweb.home	TongWeb7 的安装路径 (TongWeb7_HOME)，所设参数值为一个文件夹，该文件夹下包含 TongWeb7 的 bin、conf、lib 等文件夹。
-Djava.io.tmpdir	服务器产生的临时文件以及应用预编译文件所在的目录。
-Duser.dir	应用服务器 TongWeb7 的启动脚本和停止脚本所在的目录。
-DMonitorDay	当触发快照时，监视量收集的天数是可以设置的，默认是收集最近五天的监视量，当用户希望设置其他天数的监视量，可以通过这个参数来进行设置。
-Djava.awt.headless	如果使用 headless 模式处理 awt 程序，设置为 true；否则以字符界面启动 TongWeb7 监控图形不出来。
-Dtongweb.jndi.lookup.relaxVersion=false	应用更新远程调用 ejb 开关设置，如果为 true，每次调用的都是未退休的

	<p>ejb, 为 false, 同一个 session 的请求调用的是同一个 ejb。</p>
<p>-Dcom.tongweb.commons.logging.Log=com.tongweb.commons.logging.impl.Jdk14Logger</p>	<p>指定 TongWeb7 的日志服务实现类, 当用户希望使用其他的日志服务实现类来代替 TongWeb7 的日志服务可以通过此参数进行设置。</p>
<p>-Djava.security.policy</p>	<p>服务器使用 Java 安全管理器对服务器资源的访问作进一步控制。JVM 的安全机制需要一个安全策略文件 (.policy) 来定义访问权限, 这些权限定义了运行在一个 JVM 实例中的类是否可以执行某项运行时操作。 在启动服务器时可以用下面的 -Djava.security.policy 属性指定安全策略的全路径名。</p>
<p>-Djava.security.manager</p>	<p>如果需要 -Djava.security.policy 指定的安全策略文件生效, 需要在 java 启动参数中增加 -Djava.security.manager。</p>
<p>-Dtongweb.jconsole.cbport</p>	<p>外部使用的 jmx 连接端口。由于 JMX 连接使用 rmi 协议, 在不指定端口的情况下, rmi 产生的 stub 将使用的随机产生的端口。当开启防火墙时, 访问随机端口的请求将被防火墙拦截, 导致连接被拒绝。解决方案是, 通过参数指定 stub 所用的端口, 然后在防火墙上开放此端口。涉及到使用 JMX 连接的地方为应用服务器的 JMX 连接, 如果 TongWeb7 所在机器开启了防火墙, 需要在 TongWeb7 的启动脚本中。加入增加下列参数: -Dtongweb.jconsole.cbport = 端口号 -Dtongweb.rmi.jmx.cbport= 端口号</p>
<p>-Dtongweb.rmi.jmx.cbport</p>	<p>内部使用的 jmx 连接端口。</p>
<p>-Dopenejb.crosscontext=true</p>	<p>对于跨上下文的访问, 在 TongWeb7 中需要在启动脚本中配置参数 -Dopenejb.crosscontext=true 即可。</p>
<p>-Dibm_heapdumpdir</p>	<p>指定 IBM JDK 生成的堆内存镜像文件所在的文件夹。参数值必需和环境变量 export IBM_HEAPDUMPDIR 值相同。</p>
<p>-Dibm_javacoredir</p>	<p>指定 IBM JDK 生成的栈内存镜像文件所在的文件夹。参数值必需和环境变量 export IBM_JAVACOREDIRE 值相同。</p>
<p>-Dtongweb.X_Frame_Options</p>	<p>配置 Http 和 Ajp 的 X_Frame_Options</p>

	<p>响应头。可选配置参数如下：DENY，浏览器拒绝当前页面加载任何 Frame 页面；SAMEORIGIN，frame 页面的地址只能为同源域名下的页面；ALLOW-FROM，允许 frame 加载的页面地址；指定为 NONE 或者不配置该参数则在响应头中不存在该内容。</p>
-Dremote.http	<p>在 EJB 远程调用的服务端设置，配置 EJB 远程调用的协议格式。如果设置为 true 那么将接受 http 协议的 EJB 远程调用，如果设置为 false，将接收 remote 协议的 EJB 远程调用。默认为 false。</p>
-Dremote.url.translate	<p>在 EJB 远程调用的客户端设置。进行远程调用地址的转换。如配置成 remotetohttp，将会把 EJB 远程调用的 remote://ip:port 格式的地址转换为 http://ip:port/ejbserver/ejb。如果设置为 httpstoremote，那么将 http://ip:port/ejbserver/ejb 格式的地址转换为 remote://ip:port。</p>
-DSharedSessionEnabled	<p>应用共享 Session 开关, 开启后各个应用使用共享的 session，设置 true 或者 false，默认 false 表示关闭。</p>
-DSharedSessionContext	<p>创建 Session 时，是否使用 http 请求里基于 Cookie 附带的 Session ID，默认 false 表示关闭，关闭后创建 Session 会自动生成随机 ID，开启后则复用 Cookie 里的 ID。该功能一般在跨应用或跨进程实现 SSO 时使用。</p>
-DenableJPA	<p>默认为 true，设置为 false 是代表采用应用自带的 jpa 实现。</p>
-DjpaAutoCreate	<p>默认为 true，如果发现 persistence.xml 配置的数据源不存在则自动找到一个已有数据源创建一个可用数据源。考虑到采用的数据源不一定是用户希望用的库所以提供了关闭功能，即设置为 false。</p>
-DcontentLength.limit	<p>设置 POST 请求中的 content length 限制，避免一直占住这个连接不断开，一般针对 POST 请求的攻击，默认值为 Integer 的最大值，例如 -DcontentLength.limit=10000。</p>
-Dread.http.header.timeout	<p>读一个字节耗费的毫秒数(单位：毫秒/字节)，超过或等于这个速率算超</p>

	时，默认值为 Long 的最大值，例如： -Dread.http.header.timeout=500。 与-DcontentTypeLength.limit 参数配合使用。
-Dread.http.header.timeout.limit	请求头读取超时次数，超过次数将其列入黑名单，开始拒绝请求(连接)，默认值为 Integer 的最大值，例如： - Dread.http.header.timeout.limit=3。与 -Dread.http.header.timeout 配合使用。
-Dblacklist.expired=xxx	对清理黑名单解锁时间进行设置，默认值为 12（单位：小时），例如：-Dblacklist.expired=1。与 -Dread.http.header.timeout.limit 配合使用。
-DenableSigar	默认为 true，设置为 false 是关闭采用 sigar 获取 CPU 监视量，因为某些平台会出现不兼容情况，这样可以继续观察其他监视量，只忽略掉 CPU 监视量。
-DCreateOne	连接池中如果对每个连接都进行有效性验证是非常耗时的，影响用户体验。-DCreateOne=true 的作用是，在检查到第一个失效连接后，强制让连接池中所有 idle 连接都失效，而不再去进行有效性检查，默认值为 false。
-DstatelessUseGlobal	场景如在 tongweb-ejb-jar.xml 中指定实例池的大小等参数，在控制台保存无状态会话 bean 默认是会覆盖的，可以配置此参数优先使用应用下面配置的池参数。-DstatelessUseGlobal 默认为 true，即全局配置优先。如果有此需求可以配置为 false。
-DConnectionPoolDebug	默认为 false，不开启。设置为 true 则开启记录 EJB 调用连接池的分析日志。
-DskipValidationXML	默认为 false，不开启。设置为 true 时则在应用部署时不会读取并解析应用中的验证框架配置文件 validation.xml。
-Daudit.log.enabled	审计日志开关，boolean 类型，true 表示开启审计日志记录功能，

	false 表示关闭审计日志记录功能，默认值为 true。
-Daudit.log.file.maxSize	审计日志文件按大小轮转阈值，长整数类型，单位“KB”，默认值为 1048576，即 10 MB。
-Daudit.log.file.retentionDays	审计日志文件按日期清理阈值，整数类型，单位为“天”，默认值为 180，即 6 个月。
-DWebModuleOnly	服务器处理 JPA、Validation、CDI、JSF 的开关，boolean 类型，true 表示关闭 TongWeb 服务器的处理，false 表示开启，默认值为 false。
-DShutdownSocketDisabled	以往的方式是在服务器启动后会开启一个 TCP 监听服务，默认端口为 8005，主要用于接收停止脚本发送的停止命令，配置该参数可以选择是否开启该服务。该参数的值为 true 或 false，true 表示禁用该服务，false 表示保留该服务。若该服务被禁用，则通过文件监控的方式来实现该功能。
-DFullURLMode	补充 requestURI 地址，boolean 类型，true 表示将应用配置的重定向地址补充到 requestURI 中，false 表示忽略，默认值为 false。
GT_ENABLED	表示是否开启全局事务支持功能，类型为 Boolean，默认值为 false，即不开启。
TX_RECOVERY	表示是否开启事务日志和恢复功能，类型为 Boolean，默认值为 false，即不开启。
-DenableParallelDeploy	为了满足用户部署应用多导致服务器启动慢的问题，设置为 true 为开启后服务器启动时并行部署非系统应用，默认为 false。
-DcustomLogMoudle	<p>应用中存在如：</p> <pre>Logger.getLogger("org.postgresql.Driver")</pre> <p>获取到的 Logger 对象，如果业务逻辑中对其进行删除等操作可能会导致与 TongWeb 的日志系统产生冲突。</p> <p>为了解决此类问题可以设置该参数，</p>

	<p>要求配置以逗号分隔的包名前缀，匹配相应包名不被 TongWeb 自带的日志系统接管。</p> <p>如 DcustomLogMoudle=org.postgresql</p>
--	--

Java 参数说明如下：

参数名称	参数说明
-Xmx512m	指定 JVM 堆的最大值，默认为物理内存的 1/4 或者 1G，最小为 2M 。
-XX:+LogVMOutput	记录 Java 虚拟机的输出，+表示打开 。
-XX:MaxPermSize	设置 JVM 堆中“永生代”的最大容量 。
- XX:+UnlockDiagnosticVMOptions	当这个参数打开时，可以使用其它的 JVM 诊断参数 。
-XX:LogFile	Java 虚拟机的输出被记录到这个参数指定的文件中，如%TongWeb7_HOME%\logs\jvm.log 。
-Xloggc:gc.log	指定垃圾收集日志文件。
-Xdump:heap	开启 IBM JDK 生成堆栈内存镜像。IBM JDK 不支持 jmap 和 jstack 命令，添加此参数后配合 kill -3 pid 系统命令以生成堆栈镜像。这个参数要配合 export IBM_HEAPDUMPDIR 和 export IBM_JAVACOREDIRE 两个环境变量来指定镜像生成的位置。
- Djava.util.logging.manager= com.tongweb.log.TongwebLogManager	指定日志管理类，必须是 LogManager 的子类。此参数为必须参数。
-Djava.endorsed.dirs	在这个目录下的某种标准的实现将重载 JDK 内对该标准的实现，如果不设这个参数，默认指向 JAVA_HOME 下的 lib\endorsed 目录。如 %TongWeb7_HOME%\lib\endorsed。
- Djava.security.auth.login.config= "%TongWeb7_HOME%\conf\security\login.config"	指定 JAAS 配置文件（包含对所有登录模块 LoginModule 的引用）的路径 。
- Djava.net.preferIPv4Stack=true	当需要 TongWeb7 运行于 AIX 上时，需要在启动脚本中有这此选项配置。设置为 true 时用于限制优先使用 IP4 地址。
-Dibm.stream.nio=true	当需要 TongWeb7 运行于 AIX 上时，需要在启动脚本中有这此选项配置，它用于处理 IO 和 NIO 转换器排序问题。如果此选项设置为 true，那么将使用 NIO 转换器而不是 IO 转换器。缺省情况下，使用 IO 转换

	器。
-javaagent: "%TongWeb7_HOME%\lib\tongejb-javaagent.jar"	应用服务器需要通过 agent 加载的系统包，这个参数指定它所在的路径。
-Djava.security.egd	随机数获取源，解决 JVM 在 Linux 上启动慢的问题。
-DsourceAbandonTime	连接泄漏后清理线程关闭连接时等待 connection 关闭线程的毫秒数，如果不配置清理线程默认等待关闭线程 3 秒后执行。（单位毫秒）
-Djavax.persistence.provider	指定服务器 JPA 默认的 provider 实现。比如当用户希望当前服务器下面的所有应用都采用 hibernate 的 provider 实现时，可以在启动脚本中加入 -Djavax.persistence.provider=org.hibernate.ejb.HibernatePersistence。
-Drequire-secret	指定 ajp 协议的默认密码，该参数的优先级低于 tongweb.xml。注：设置该参数后，需要同步修改 apache 或其它支持 ajp 协议的负载均衡器的配置文件，以 apache 为例，在其 workers 配置文件中添加 'secret' 密码属性，形如： worker.tongweb1.secret=ajp 协议密码。
-DdisableVerCode	是否禁用控制台登录的验证码机制，true 表示禁止，其它值或不存在该参数表示不禁止，默认为 true。
-Dxss_defense	xss 攻击防护开关，true 表示开启，其它值或不存在该参数表示不开启。该功能会检查请求中的所有参数名和参数值，若包含“尖括号”、“eval()”、“javascript”等特殊字符则强制返回 403。
-Dxss_apps	xss 攻击防护生效的应用名，多个应用名以逗号分隔。
-DjvProfile	是否支持 Validation 和 jpa 的开关，boolean 类型，true 表示关闭，false 表示开启，默认是 true。
-DLoadClassCache.Disabled=true	是否启用禁用重复类加载快速完成缓存功能，开启后相同类型的类只加载一次，其加载成功或失败的结果将被缓存，用于提供重复类加载时的速度，boolean 类型，true 表示禁用，false 表示开启。注意：对于 jpa 应用，必需禁用该功能，原因是 jpa 应用会尝试在加载失败后动态构建类并再次尝试加载。
-DuseBeanManagerInCompJNDI	是否往全局 jndi 树上注册“BeanManager”，boolean 类型，true 表示注册，false 表示不注册。该配置用于解决和应用自带 BeanManager 冲突问题，默认 false。
-Dwebcluster.session.sticky	Web 集群亲和设置开关，默认为 true（亲和）。设置为 false 表示非亲和。
-DRedirectWhenAbsolute	重定向时是否将绝对路径添加到响应头的 Location 字段，boolean 类型，true 表示使用（包含协议名称、主机名称、端口号、应用访问名称以及资源名称），false 表示不使用（只包含资源名称），默认为 false。

-DdisableVerCode	是否禁用控制台的验证码，boolean 类型，true 表示禁用，false 表示不禁用，默认为 true。
------------------	--

9. 附录 9 命令行使用说明

命令行的启动脚本位于\${tongweb.home}/bin 目录，以 commandstool 命名，有两个版本 bat(windows)和 sh(*nix)。

9.1 基本使用说明

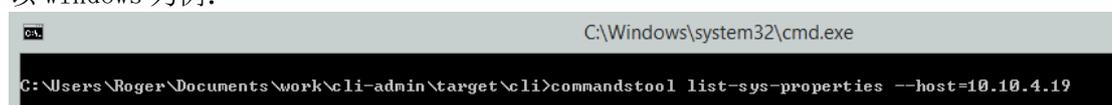
9.1.1 执行方式

本命令行工具存在两种执行命令的方式：直接执行和多级交互式执行。

直接执行

即直接在命令提示符下输入脚本文件，相应命令及参数。

以 windows 为例：



```
C:\Windows\system32\cmd.exe
C:\Users\Roger\Documents\work\cli-admin\target\cli>commandstool list-sys-properties --host=10.10.4.19
```

多级交互式执行

使用时启动 commandstool 脚本，进入次级命令提示符，只需输入相应命令，不用多次执行 commandstool 脚本。

9.1.2 安全认证

TongWeb 中，每一次执行命令都需要进行安全认证，此安全认证是系统应用受保护资源所需的认证。执行命令时需要提供用户名密码才能通过认证。

认证方式分为两种：

- 交互方式，即可以通过用户名密码的提示来进行认证。
- 直接方式，直接在输入命令时，将须认证的用户名与密码提供出来。
- 文件存储密码方式，可以在操作系统的用户目录下，保存“.asadminprefs”文件。文件中可以指定命令的用户名和密码，格式为：

```
AS_ADMIN_user=cli
```

```
AS_ADMIN_password=cli123.com
```

对于交互方式及直接方式的用户名密码的设置，可以通过控制台的文件类型安全域中管理用户的方式来指定。同时需要注意需要指定命令行用户的组“cli”才可以进行命令行操作。

9.1.3 参数格式

所有的参数都可以--param=value 的格式进行输入。如果参数为 boolean 值，则可以省略“=value”，值为描述属性的默认值。另外参数的值当中如果包含【空格】如一些文件和地址等作为属性值，或者特殊字符如“&”等符号，需要将该属性用“引号”包括起来，命令行才会识别这个值。举个例子如：
jdbcurl="jdbc:mysql://localhost/tw6?useUnicode=true&characterEncoding=GBK"。

9.1.4 错误提示

当输入错误命令的时候，命令行工具会提示输入错误；如果输入的内容不包含必选参数，命令行工具还会提示哪个参数是必需的；诸如 create-**一类的命令，都需要一个操作数，如 create-http-listener 要创建一个名为 test-listener 的通道，则“test-listener”就是这个操作数。如果输入的命令不包含这个操作数，命令行工具也会给出错误提示。

例如当您输入一个错误的命令，错误提示如下：

```

commandstool> create-hhttp-listener
CLI001 命令 create-hhttp-listener 无效。要获得有效命令的列表, 请使用 "commandstool help"。
Closest matching command(s):
    create-http-listener
Use "help" command for a list of valid commands.
commandstool>

```

9.1.5 参数默认值

命令行工具提供了默认配置文件, 在 tongweb 安装路径/lib/下的 tongweb.jar 包中的 CLIDescriptor.xml 文件, 根据参数名找到对应参数, 设置/修改其 default 值即可

9.2 基本参数

命令行包含一些基本参数, 这些参数适用于任何命令, 比如当您需要通过命令行管理一个远程 TongWeb 实例, 就可以通过如 --host、--port 参数来实现。例如要通过命令行对 ip 为 192.168.110.90, 端口为 9060 的服务器实例, 进行启动 test-listener 通道的操作:

```
commandstool> start-http-listener --host=192.168.110.90 --port=9060 test-listener
```

下面列出所有的基本参数:

--host: 用于远程连接, 如果需要连接远程服务器, host 代表需要连接的远程 ip 地址。默认 localhost。

--port: 要连接的远程端口。默认 9060。

--secure: boolean 类型, 选则 http/https 访问服务器。默认为 false, 即 http。

--user: 远程连接所须的认证用户。如不填写, 则专门进行交互输入。默认为 cli。

--password: 认证用户所须的密码。如不填写, 则专门进行交互输入。默认为 cli123.com。

--passwordfile: 是保存用户密码的文件 (文件中的属性名必须是 AS_ADMIN_password)

--echo: 将一条完整的命令反馈出来。

--terse: 修改日志级别, true 为 INFO, false 为 FINE。

--interactive: 是否使用交互式输入。默认是 true

--help: 帮助

9.3 Commandstool 使用命令

下面是所有 commandstool 的命令列表。

9.3.1 change-admin-password

功能说明	用于修改管理密码。此命令为交互式命令, 提示用户提供旧的和新的管理密码并进行确认。该命令只能修改位于 "cli" 组的管理用户。
相关参数	-
实例	<pre>commandstool> change-admin-password --user cli Please enter the old admin password>cli123.com Please enter the new admin password>cli123.com Please enter the new admin password again>cli123.com</pre>
验证方式	执行命令 list-http-listeners 时, 输入用户名 cli, 密码 cli123.com 可以正确查看所有的通道。

9.3.2 create-auth-realm

功能说明	创建安全域。若用相同名称创建即是更新。
相关参	--realmtype 安全域类型, file sql ldap jaas script spi

数	<p>, 默认为 file</p> <p>--allrolesmodel 对角色认证的限制级别, strict authOnly strictAuthOnly, 默认 strict</p> <p>--usecontextclassloader 选择类加载方式, context container, 默认 context。</p> <p>--x509UsernameRetrieverClassName X509 证书解析器实现类 默认 com.tongweb.catalina.realm.X509SubjectDnRetriever</p> <p>--lockEnabled 是否使用锁定机制 默认锁定</p> <p>--failureCount 允许错误次数 默认 5 次</p> <p>--lockOutTime 锁定超时 默认 300s</p> <p>--cacheSize 缓存大小 1000 个</p> <p>--cacheRemovalWarningTime 最小缓存时间被锁定的用户在缓存中的时间少于此值, 则记录警告日志, 默认为 3600 秒</p> <p>--realmdigest 选择加密方式, SM3 MD5 SHA-1 SHA-256 SHA-512 DIGEST, 默认 MD5。</p> <p>--groupfilename 指定组文件路径, 默认为系统默认的组文件。</p> <p>--userfilename 指定用户文件路径, 默认为系统默认的用户文件。</p> <p>--usedatasource 使用数据源。</p> <p>--datasourcename 数据源名称。</p> <p>--jdbcurl JDBC 的 url 地址。</p> <p>--dbuser JDBC 的用户名。</p> <p>--jdbcdriver JDBC 驱动。</p> <p>--dbpassword JDBC 密码。</p> <p>--userselect 选择用户 sql。</p> <p>--groupselect 组 sql。</p> <p>目标参数: 安全域名称。名称由数字、字母、下划线和'-'组成, 首字符为英文</p>
实例	<p>file 类型: create-auth-realm --realmtype=file --realmdigest=MD5 --groupfilename=groups.properties --userfilename=users.properties test-cli-realm</p> <p>jdbc 类型: create-auth-realm --realmtype=sql --realmdigest=MD5 --usedatasource=true --datasourcename=jdbctest test-cli-realm</p>
验证方式	<p>依次点击管理控制台左侧导航树中的安全服务->安全域管理, 查看 test-cli-realm 已经创建成功。</p>

9.3.3 delete-auth-realm

功能说明	删除认证域。
相关参数	目标参数: 安全域名称。
实例	delete-auth-realm test-realm
验证方式	依次点击管理控制台左侧导航树中的安全服务->安全域管理, 查看 test-cli-realm 已经被删除。

9.3.4 create-file-user

功能说明	在密钥文件中创建具有指定用户名、密码和组的条目。可以创建多个以冒号分隔的组。如果存在相同的用户名则为更新
相关参数	<p>--groupname 组名, 即用户所在分组, 多个可用逗号分隔。</p> <p>--filepassword 密码。</p> <p>--realmname 安全域名称</p> <p>--realmdigest 加密类型, 包括 SM3 MD5 SHA-1 SHA-256 SHA-512 DIGEST</p>

	目标参数：用户名称。名称由数字、字母、下划线和'-'组成，首字符为英文
实例	create-file-user --groupname=cli --filepassword=1234 --realmname=test-cli-realm samantha 注：其中--realmname=test-cli-realm 这里指定的 realmname，必须是在安全域中是已经存在的。
验证方式	依次点击管理控制台左侧导航树中的安全服务->安全域管理，找到安全域名称 test-cli-realm，点击管理用户，查看 samantha 已经在 cli 组中

9.3.5 delete-file-user

功能说明	删除指定文件安全域用户。
相关参数	-- realmname: 安全域名称 目标参数：用户名称。
实例	delete-file-user --realmname=test-cli-realm samantha
验证方式	依次点击管理控制台左侧导航树中的安全服务->安全域管理，找到安全域名称 test-cli-realm，点击管理用户，查看组和用户。

9.3.6 create-http-listener

功能说明	添加新的 HTTP 侦听器。
相关参数	--securitabled Boolean 型，false 建立 http 连接器，true 则建立 https 连接器。默认 false。 --listeneraddress 指定连接器的监听地址。默认监听本机所有地址 --listenerport 指定连接器的监听端口。必填。 --xpowered 默认 false --defaulttvs 指定默认的虚拟主机。必填 --redirectport 指定连接器的重定向端口。可选。 --blockabled 选择是否阻塞，true 为 bio，false 为 nio，默认为 false。 --backlog 默认 100。 --maxthreads 默认 200。 --minsparethreads 默认 10。 --sslalias 证书别名。必填。 --keystorefile 证书路径，必填。 --keystorepass 证书密码，必填。 --keystoretype 证书类型，默认 JKS。 --ssltype 协议类型，默认 TLS。 --clientauth 是否使用客户端认证。默认 false。 --truststorefile 信任证书路径。 -- truststorepass 信任证书密码。 --truststoretype 信任证书类型。 目标参数：连接器名称。
实例	create-http-listener --listeneraddress=0.0.0.0 --securitabled=false --listenerport=7272 --defaulttvs=server test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->HTTP 通道管理，test-listener 已经创建成功。

9.3.7 update-http-listener

功能说	更新 HTTP 侦听器。
-----	--------------

明	
相关参数	<p>--securitabled Boolean 型, false 建立 http 连接器, true 则建立 https 连接器。默认 false。</p> <p>--listeneraddress 指定连接器的监听地址。默认监听本机所有地址</p> <p>--listenerport 指定连接器的监听端口。必填。</p> <p>--xpowered 默认 false</p> <p>--defaultvs 指定默认的虚拟主机。必填</p> <p>--redirectport 指定连接器的重定向端口。可选。</p> <p>--blockabled 选择是否阻塞, true 为 bio, false 为 nio, 默认为 false。</p> <p>--backlog 默认 100。</p> <p>--maxthreads 默认 200。</p> <p>--minsparethreads 默认 10。</p> <p>--sslalias 证书别名。必填。</p> <p>--keystorefile 证书路径, 必填。</p> <p>--keystorepass 证书密码, 必填。</p> <p>--keystoretype 证书类型, 默认 JKS。</p> <p>--ssltype 协议类型, 默认 TLS。</p> <p>--clientauth 是否使用客户端认证。默认 false。</p> <p>--truststorefile 信任证书路径。</p> <p>-- truststorepass 信任证书密码。</p> <p>--truststoretype 信任证书类型。</p> <p>目标参数: 连接器名称。</p>
实例	<pre>update-http-listener --listeneraddress=0.0.0.0 --securitabled=false --listenerport=7272 --defaultvs=server test-listener</pre>
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->HTTP 通道管理查看 test-listener 属性修改成功。

9.3.8 delete-http-listener

功能说明	删除指定的 HTTP 侦听器。
相关参数	目标参数: 连接器名称。
实例	delete-http-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->HTTP 通道管理, test-listener 已经被删除。

9.3.9 start-http-listener

功能说明	启动指定的 HTTP 侦听器。
相关参数	目标参数: 连接器名称。
实例	start-http-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->HTTP 通道管理, test-listener 已经被启动。

9.3.10 stop-http-listener

功能说明	停止指定的 HTTP 侦听器。
相关参数	目标参数: 连接器名称。

实例	stop-http-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->HTTP 通道管理, test-listener 已经被停止。

9.3.11 create-ajp-listener

功能说明	添加新的 AJP 侦听器。
相关参数	<p>--listeneraddress 指定连接器的监听地址。默认监听本机所有地址</p> <p>--listenerport 指定连接器的监听端口。必填。</p> <p>--requireSecret ajp 协议密码。必填。</p> <p>--blockabled 选择是否阻塞, true 为 bio, false 为 nio, 默认为 false。</p> <p>--backlog 默认 100。</p> <p>--maxthreads 默认 200。</p> <p>--minsparethreads 默认 10。</p> <p>目标参数: 连接器名称。</p>
实例	create-ajp-listener --listeneraddress=0.0.0.0 --listenerport=8383 --requireSecret=ajp test-ajp-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->AJP 通道管理, test-listener 已经创建成功。

9.3.12 delete-ajp-listener

功能说明	删除指定的 AJP 侦听器。
相关参数	目标参数: 连接器名称。
实例	delete-ajp-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->AJP 通道管理, test-listener 已经被删除。

9.3.13 start-ajp-listener

功能说明	启动指定的 AJP 侦听器。
相关参数	目标参数: 连接器名称。
实例	start-ajp-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->AJP 通道管理, test-listener 已经被启动。

9.3.14 stop-ajp-listener

功能说明	启动指定的 AJP 侦听器。
相关参数	目标参数: 连接器名称。
实例	stop-ajp-listener test-listener
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->AJP 通道管理, test-listener 已经被停止。

9.3.15 create-virtual-server

功能说明	添加新的虚拟主机。
相关参	--hostalias 虚拟主机别名, 可选。虚拟主机别名为 ip 或者域名或者由数

数	字、字母、下划线组成，首字母为英文，多个名称可用逗号分割 --listeners 关联的监听器列表，用逗号隔开。 --accesslog 默认 true 为开启访问日志 目标参数：虚拟机名称。允许的主机名称由字母、数字、下划线、"."组成
实例	create-virtual-server --hostalias=jmcom --listeners=test-listener test-virtual-host
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->虚拟主机管理，test-virtual-host 已经创建成功。

9.3.16 delete-virtual-server

功能说明	删除指定的虚拟主机。
相关参数	目标参数：虚拟机名称。
实例	delete-virtual-server test-virtual-host
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->虚拟主机管理，test-virtual-host 已经被删除。

9.3.17 start-virtual-server

功能说明	启动指定的虚拟主机。
相关参数	目标参数：虚拟机名称。
实例	start-virtual-server test-virtual-host
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->虚拟主机管理，test-virtual-host 已经被启动。

9.3.18 stop-virtual-server

功能说明	停止指定的虚拟主机。
相关参数	目标参数：虚拟机名称。
实例	stop-virtual-server test-virtual-host
验证方式	依次点击管理控制台左侧导航树中的 Web 容器配置->虚拟主机管理，test-virtual-host 已经被停止。

9.3.19 create-jdbc-connection-pool

功能说明	注册具有指定 JDBC 连接池名称的 JDBC 连接池。
相关参数	--jdbcdriver 选择数据库驱动类名，为必要参数。 --jdbcurl 连接数据库所须的 url，必选。 --dbuser 连接数据库所须的用户名，必选。 --dbpassword 用户名密码。 --jdbcclasspathjdbc 启动路径。 --testquery 测试连接 --jtamanaged 使用 jta 管理。 --jdbcinitialsize jdbc 初始化连接数。 --jdbcmaxactive 最大数据库连接数。

	<p>--jdbcmaxwaittime 最大等待时间。 --removeabandonedtimeout 泄露超时时间。 --testonborrow 获取连接时验证 --testonconnect 创建连接时验证 --testonreturn 归还连接时验证 --testWhileIdle 连接有效性检查 --testonborrow 获取连接时验证。 --testonreturn 归还连接时验证。 --queryTimeout 语句超时。 --sqllog sql 日志。 --threshold SQL 执行时间过滤。 --stmtcache 语句缓存。 --maxCache 语句最大缓存数。 --prepared PreparedStatement 语句缓存。 --callable CallableStatement 语句缓存。 --stmtfn 语句跟踪 --leakCheck 即时泄露回收 目标参数：数据源名称。名称由数字、字母、下划线、'和/'组成，首字符为英文，尾字符不能为/</p>
实例	<pre>create-jdbc-connection-pool --jdbcdriver=com.mysql.jdbc.Driver jdbcurl=jdbc:mysql://10.10.4.55:3306/test1 --dbuser=cli --dbpassword=cli123.com --jdbcclasspath=/path/to/mysql-connector- java-5.1.18.jar test-cli-jdbc</pre>
验证方式	<p>依次点击管理控制台左侧导航树中的 JDBC 配置->连接池管理，test_jdbcpool 已经成功创建。</p>

9.3.20 update-jdbc-connection-pool

功能说明	<p>修改指定名称的 JDBC 连接池属性。</p>
相关参数	<p>--jdbcdriver 选择数据库驱动类名。 --jdbcurl 连接数据库所需的 url。 --dbuser 连接数据库所需的用户名。 --dbpassword 用户名密码。 --jdbcclasspathjdbc 启动路径。 --testquery 测试连接 --jtamanaged 使用 jta 管理。 --jdbcinitialsize jdbc 初始化连接数。 --jdbcmaxactive 最大数据库连接数。 --jdbcmaxwaittime 最大等待时间。 --removeabandonedtimeout 泄露超时时间。 --testonborrow 获取连接时验证 --testonconnect 创建连接时验证 --testonreturn 归还连接时验证 --testWhileIdle 连接有效性检查 --testonborrow 获取连接时验证。</p>

	--testonreturn 归还连接时验证。 --queryTimeout 语句超时。 --sqllog sql 日志。 --threshold SQL 执行时间过滤。 --stmtcache 语句缓存。 --maxCache 语句最大缓存数。 --prepared PreparedStatement 语句缓存。 --callable CallableStatement 语句缓存。 --stmtfn 语句跟踪 --leakCheck 即时泄露回收 目标参数：数据源名称。名称由数字、字母、下划线、'-'和'/'组成，首字符为英文，尾字符不能为'/'
实例	update-jdbc-connection-pool --jdbcdriver=com.mysql.jdbc.Driver jdbcurl=jdbc:mysql://10.10.4.55:3306/test1 --dbuser=cli --dbpassword=cli123.com --jdbcclasspath=/path/to/mysql-connector- java-5.1.18.jar test-cli-jdbc
验证方式	依次点击管理控制台左侧导航树中的 JDBC 配置->连接池管理，查看 test_jdbcpool 连接池的属性已修改成功。

9.3.21 delete-jdbc-connection-pool

功能说明	删除具有指定 JDBC 连接池名称的 JDBC 连接池。
相关参数	目标参数：数据源名称。
实例	delete-jdbc-connection-pool test_jdbcpool
验证方式	依次点击管理控制台左侧导航树中的 JDBC 配置->连接池管理，test_jdbcpool 已经删除成功。

9.3.22 deploy

功能说明	部署企业应用程序、Web 应用程序、EJB 模块。commandstool 工具还支持远程部署功能。
相关参数	-- application: 客户端应用文件的路径，必选。 -- defaultvs: 虚拟服务器。 -- contextroot: 应用前缀。只有在 web 应用程序部署时可用。 -- precompilejsp: jsp 是否预编译。 -- deployorder: 设置部署顺序。 -- appdescription: 应用描述。 -- delegate: 类加载策略，默认是子优先 false，如果想配置父优先则设置为 true。 目标参数：应用名称。
实例	Deploy --contextroot=/dbpooltest --precompilejsp=true -- application=D:\\mydesk\\tomcat\\command\\ dbpooltest.war testapp 远程部署实例： deploy --host=192.168.110.90 --port=9060 --user=cli -- passwordfile=C:\\passwordfile --precompilejsp=true --

	defaultvs=server --contextroot=dbpooltest -- applocation=C:/test/dbpooltest.war testapp (host、port、user、passwordfile 参见 4.2 基本参数)
注意事 项	1. 使用 deploy 命令进行目录部署时，仅支持本地主机。如果向远程主机部署应用，只支持打包的应用，不支持目录。
验证方 式	依次点击管理控制台左侧导航树中的应用管理，testapp 已经部署成功。

9.3.23 undeploy

功能说 明	解除部署在指定目标上的组件。支持远程解部署功能
相关参 数	目标参数：应用名称。
实例	undeploy dbpooltest 远程解部署实例： commandstool undeploy --host 192.168.110.21 --port 9060 -- user cli --passwordfile F:/My_Test/TWNSSSP-383/passwordfile dbpooltest (host、port、user、passwordfile 参见 4.2 基本参数) 注意：1. 解部署应用时，需要正确的指定应用名称。 应用名称的默认值是去掉文件后缀之后的应用名称，例如：应用为 c:/test2.ear，默认的应用名称为 test2。如果应用名称不正确，将 导致解部署失败。为了确认应用名称，可以通过 list-apps 查看是否 有当前应用名称存在。
验证方 式	依次点击管理控制台左侧导航树中的应用管理，dbpooltest 已经解除部署。

9.3.24 redeploy

功能说 明	重新部署在指定目标上的组件。支持远程重新部署功能
相关参 数	以下参数是远程重部署必须设置的参数，也就是基本参数： -- applocation：客户端应用文件的路径，可选。 目标参数：应用名称。
实例	如果需要重部署一个新的应用文件的话： redeploy -- applocation =E:/test.war dbpooltest 如果重部署当前应用，而不替换文件的话： redeploy dbpooltest 远程解部署实例： commandstool redeploy --host 192.168.110.21 --port 9060 -- user cli -passwordfile=F:/My_Test/TWNSSSP-383/passwordfile --applocation=c:/test2.ear dbpooltest (host、port、user、passwordfile 参见 4.2 基本参数) 注意：1. dbpooltest 表示应用名称。 2. 重部署应用时，需要正确的指定应用名称。 应用名称的默认值是去掉文件后缀之后的应用名称，例如：应用为 c:/test2.ear，默认的应用名称为 test2。如果应用名称不正确，将 导致重部署失败。为了确认应用名称，可以查看 TongWeb 的管理控制 台上已部署应用的应用名称这一项。
注意事 项	1. 使用 redeploy 命令进行目录部署时，仅支持本地主机。如果向远程主机部署应用，只支持打包的应用，不支持目录。
验证方	依次点击管理控制台左侧导航树中的应用管理查看 dbpooltest，点击

式	Http 访问，查看修改是否生效。
---	-------------------

9.3.25 ping-jdbc-connection-pool

功能说明	测试连接池是否可用。
相关参数	-
实例	ping-jdbc-connection-pool __TimerPool

9.3.26 help

功能说明	显示所有命令。
相关参数	-
实例	help

9.3.27 list-apps

功能说明	列出所有部署的应用。
相关参数	-
实例	list-apps

9.3.28 list-file-users

功能说明	指定文件类型安全域的文件用户、组的列表。
相关参数	目标参数：安全域名称。
实例	list-file-users twns-realm

9.3.29 list-http-listeners

功能说明	列出现有 HTTP 侦听器。
相关参数	-
实例	list-http-listeners

9.3.30 list-ajp-listeners

功能说明	列出现有 AJP 侦听器。
相关参数	-
实例	list-ajp-listeners

9.3.31 list-jdbc-connection-pools

功能说明	获取已创建的 JDBC 连接池。
相关参数	-
实例	list-jdbc-connection-pools

9.3.32 list-virtual-servers

功能说明	列出存在的虚拟服务器。
相关参数	-
实例	list-virtual-servers

9.3.33 list-auth-realms

功能说明	列出身份验证领域。
相关参数	-
实例	list-auth-realms

9.3.34 list-sys-properties

功能说明	列出服务器系统属性。
相关参数	-
实例	list-sys-properties

9.3.35 list-jms-connection-factory

功能说明	获取已创建的 JMS 连接工厂资源
相关参数	- factory
实例	list-jms-connection-factory

9.3.36 list-jms-destination

功能说明	获取已创建的 JMS 目的地资源
相关参数	-
实例	list-jms-destination
功能说明	获取已创建的 JMS 目的地资源

9.3.37 create-jms-connection-factory

功能说明	创建具有指定 JMS 连接工厂名称的 JMS 连接工厂。
相关参数	--restype 连接工厂类型，可选，默认值为 javax.jms.ConnectionFactory 用此参数指定时，其取值是 javax.jms.ConnectionFactory, javax.jms.QueueConnectionFactory, javax.jms.TopicConnectionFactory 中的任意一个。 --adapter 连接工厂所使用的资源适配器，默认为 genericra。也可能是其他的已部署好的资源适配器。可选。 --description 描述，可选。 --minSize 连接池的初始化连接数，默认为 10。可选。 --maxSize 连接池中的最大连接数，默认为 100。可选。 --blockingTimeoutSeconds 从连接池中获取连接的最长等待时间，单

	<p>位为秒，默认为 60 秒。可选。</p> <p>--idleTimeoutMinutes 连接的空闲超时时间，单位为分。默认为 10 分钟。可选。</p> <p>--matchConnections 获取连接时由资源适配器进行匹配，默认为 false。可选。</p> <p>--transactionSupport 连接池的事务支持类型，默认值为 NoTransaction</p> <p>用此参数指定时，其取值是 NoTransaction、LocalTransaction 和 XATransaction 中的任意一个。</p> <p>--properties 属性。取决于所使用的适配器。格式为 key:value 如果有多项则用;隔开。</p> <p>目标参数：连接工厂名称。</p>
实例	<pre>create-jms-connection-factory --restype javax.jms.ConnectionFactory --adapter genericra test 或者 create-jms-connection-factory -- restype=javax.jms.ConnectionFactor y --adapter=genericra --description=test --minSize=11 -- maxSize=101 --blocking TimeoutSeconds=60 --idleTimeoutMinutes=10 -- matchConnections=false --transaction Support=NoTransaction -- properties=ConnectionFactoryJndiName:test test</pre>

9.3.38 create-jms-destination

功能说明	创建具有指定 JMS 目的地名称的 JMS 目的地资源。
相关参数	<p>--restype 目的地类型，可选，默认值为 javax.jms.Queue</p> <p>用该参数指定时取值为 javax.jms.Queue, javax.jms.Topic 中的任意一个。</p> <p>--adapter 连接工厂所使用的资源适配器，默认为 genericra。也可能是其他的已部署好的资源适配器。可选。</p> <p>--description 描述，可选。</p> <p>--properties 属性。取决于所使用的适配器。格式为 key:value 如果有多项则用;隔开。</p> <p>目标参数：目的地名称。</p>
实例	<pre>create-jms-destination --adapter genericra --restype javax.jms.Topic test 或者 create-jms-destination --restype=javax.jms.Queue -- adapter=genericra --description=test -- properties=DestinationJndiName:test test</pre>
验证方式	依次点击管理控制台左侧导航树中的 JMS 服务->目的地，test 已经成功创建。

9.3.39 delete-jms-connection-factory

功能说明	删除具有指定 JMS 连接工厂名称的 JMS 连接工厂资源。
相关参数	目标参数：连接工厂名称。

实例	delete-jms-connection-factory test

9.3.40 delete-jms-destination

功能说明	删除具有指定 JMS 目的地名称的 JMS 目的地资源。
相关参数	目标参数：目的地名称。
实例	delete-jms-destination test

9.3.41 list-adapter-adminobject

功能说明	获取已创建的托管资源对象
相关参数	-
实例	list-adapter-adminobject
功能说明	获取已创建的托管资源对象

9.3.42 list-connector-connection-pools

功能说明	获取已创建的 JCA 连接池资源
相关参数	-
实例	list-connector-connection-pools

9.3.43 create-connector-connection-pool

功能说明	创建具有指定 JCA 连接池名称的 JCA 连接池。
相关参数	<p>--adapter 连接工厂所使用的资源适配器。必须是已部署好的资源适配器。必选。</p> <p>--definition 连接器应用中 ra.xml 中定义的连接池定义接口，必选。</p> <p>--minSize 连接池的初始化连接数，默认为 10。可选。</p> <p>--maxSize 连接池中的最大连接数，默认为 100。可选。</p> <p>--takeTimeout 从连接池中获取连接的最长等待时间，单位为秒，默认为 60 秒。可选。</p> <p>--idleTimeoutMinutes 连接的空闲超时时间，单位为分。默认为 10 分钟。可选。</p> <p>--matchConnections 获取连接时由资源适配器进行匹配，默认为 false。可选。</p> <p>--transactionSupport 连接池的事务支持类型，默认值为 NoTransaction</p> <p>用此参数指定时，其取值是 NoTransaction、LocalTransaction 和 XATransaction 中的任意一个。</p> <p>--properties 属性。取决于所使用的适配器。格式为 key:value 如果有多项则用 ; 隔开。</p> <p>目标参数：连接池名称。名称由数字、字母、下划线、'-'和'/'组成，首字符为英文，尾字符不能为 '/'</p>
实例	create-connector-connection-pool --adapter genericra --

	<pre>definition javax.jms.ConnectionFactory test 或者 create-connector-connection-pool --adapter genericra -- definition javax.jms.ConnectionFactory --minSize=11 -- maxSize=101 --takeTimeout=60 --idleTimeoutMinutes=10 -- matchConnections=false --transactionSupport=NoTransaction -- properties=ConnectionFactoryJndiName:test test</pre>

9.3.44 delete-connector-connection-pool

功能说明	删除具有指定 JCA 连接池名称的 JCA 连接池资源。
相关参数	目标参数：连接池名称。
实例	delete-connector-connection-pool test
验证方式	依次点击管理控制台左侧导航树中的 JCA->JCA 连接池，test 已经删除成功。

9.3.45 list-threadpools

功能说明	获取已创建的 JCA 线程池
相关参数	-
实例	list-threadpools
功能说明	获取已创建的 JCA 线程池

9.3.46 create-threadpool

功能说明	创建具有指定名称的 JCA 线程池。
相关参数	<pre>--minjcthreads: 线程池内的核心线程数。 --maxjcthreads: 线程池内的最大线程数。 --waitqueue: 等待队列，当核心线程全部被占用时，新的任务将被保存在等待队列中。 --threadtimeout: 超出核心线程数的线程如果在该空闲时间内没有收到新的任务，则销毁该线程，单位为秒，配置为 0 表示关闭线程空闲超时。</pre>
实例	<pre>create-threadpool test 或者 create-threadpool --minjcthreads 10 --maxjcthreads 200 -- waitqueue 10 --threadtimeout 3600 test</pre>
验证方式	依次点击管理控制台左侧导航树中的 JCA->JCA 线程池，test 已经成功创建。

9.3.47 delete-threadpool

功能说明	删除具有指定名称的 JCA 线程池。
相关参数	目标参数：线程池名称。
实例	delete-threadpool test

验证方式	依次点击管理控制台左侧导航树中的 JCA->JCA 线程池，test 已经删除成功。
------	--

9.3.48 create-adapter-adminobject

功能说明	创建托管资源对象。
相关参数	<p>--restype 目的地类型，可选，默认值为 javax.jms.Queue 用该参数指定时取值为 javax.jms.Queue, javax.jms.Topic 中的任意一个。</p> <p>--adapter 连接工厂所使用的资源适配器，默认为 genericra。也可以是其他的已部署好的资源适配器。可选。</p> <p>--properties 属性。取决于所使用的适配器。格式为 key:value 如果有多项则用;隔开。</p> <p>目标参数：目的地名称。</p>
实例	<pre>create-adapter-adminobject --adapter=genericra --restype=javax.jms.Topic test</pre> <p>或者</p> <pre>create-adapter-adminobject --restype=javax.jms.Queue --adapter=genericra --properties=DestinationJndiName:test test</pre>
验证方式	依次点击管理控制台左侧导航树中的 JCA->资源托管对象，test 已经成功创建。

9.3.49 delete-adapter-adminobject

功能说明	删除托管资源对象。
相关参数	目标参数：托管资源 jndi 名称。
实例	<pre>delete-adapter-adminobject test</pre>
验证方式	依次点击管理控制台左侧导航树中的 JCA->托管资源对象，test 已经删除成功。

9.3.50 list-connector-security-maps

功能说明	列出属于指定连接器连接池的安全性映射。
相关参数	目标参数：连接池名称。
实例	<pre>list-connector-security-maps testMap</pre>
功能说明	列出属于指定连接器连接池的安全性映射。

9.3.51 create-connector-security-map

功能说明	为指定连接器连接池创建安全性映射。如果不存在安全性映射，则会创建新的安全性映射。
相关参数	<p>--poolname: 连接器连接池名。为必要参数。</p> <p>--principals: 主体。该项与 usergroups 只能选择其一。为必要参数。</p> <p>--usergroups: 用户组。</p> <p>--mappedusername: 访问 EIS 所需的用户名。为必要参数。</p>

	<p>--mappedpw: 访问 EIS 所需的密码。为必要参数。</p> <p>目标参数:安全映射名称名称由数字、字母、下划线、'-'和'/'组成,首字符为英文,尾字符不能为'/'</p>
实例	<pre>create-connector-security-map --poolname=testpool -- usergroups=eis_groups --mappedusername=hewj --mappedpw=test testpoolmap</pre>
验证方式	<p>依次点击管理控制台左侧导航树中的资源管理->JCA->JCA 连接池,找到 testpool 并点击安全映射, testpoolmap 已经创建成功。</p>

9.3.52 delete-connector-security-map

功能说明	<p>删除指定连接器连接池的安全性映射。</p>
相关参数	<p>--poolname: 连接器连接池名。为必要参数。</p>
实例	<pre>delete-connector-security-map --poolname=testpool testpoolmap</pre>
验证方式	<p>依次点击管理控制台左侧导航树中的资源管理->JCA->JCA 连接池,找到 testpool 并点击安全映射, testpoolmap 已经删除。</p>

9.3.53 version

功能说明	<p>查看 TongWeb 的版本号。</p>
相关参数	<p>-</p>
实例	<pre>version</pre>
功能说明	<p>查看 TongWeb 的版本号。</p>

9.3.54 web-container-config

功能说明	<p>查看 web 容器参数配置。</p>
相关参数	<p>-</p>
实例	<pre>web-container-config</pre>
功能说明	<p>查看 web 容器参数配置。</p>

9.3.55 update-web-container-config

功能说	<p>更新 web 容器参数配置</p>
-----	--------------------------------------

明	
相关参数	<p>--jvmroute: 在负载均衡场景中所必须的唯一标识, 以支持 session 亲和。可选。</p> <p>--jspdevelopment: 启用 JSP 开发模式。可选。</p> <p>--parameterencoding: 默认请求参数解码字符集, 此属性修改后需要重启服务器才能生效。可选。</p> <p>--responseencoding: 默认应答编码字符集, 此属性修改后需要重启服务器才能生效。可选。</p> <p>--sessiontimeout: session 超时时间, 以分钟为单位。可选。</p> <p>--htt: hungThreadThreshold, 超时线程阈值, 以秒为单位。取值范围[1-2147483647], 可选。</p> <p>--hv : 主机名验证器, 取值范围[TWHostnameVerifier NullHostnameVerifier CustomerHostnameVerifier:FullClassName]。可选。</p> <p>--sl: session 复制日志开关, 取值范围[true false]。可选。</p> <p>--cmts: 完整请求时间, 单位为秒, 取值范围[1-2147483647], 可选。</p> <p>--mat: 慢攻击容忍次数, 取值范围[1-2147483647], 可选。</p> <p>--beh: 黑名单移除时间, 单位为小时, 取值范围[1-2147483647], 可选。</p> <p>--icc: 中断当前连接, 取值范围[true false], 可选。</p> <p>--hgg: 防 host 头攻击, 取值范围[true false], 可选。</p> <p>--hggw: 主机名白名单, 可选。</p>
实例	<pre>update-web-container-config --jvmroute=a1 -- jspdevelopment=false --parameterencoding=GBK -- responseencoding=GBK --sessiontimeout=60 --sl=false --htt=180 --hv=TWHostnameVerifier --cmts=60 --mat=10 --beh=1 --icc=true --hgg=false --hggw=127.0.0.1</pre>
功能说明	更新 web 容器参数配置。

9.3.56 server-log-config

功能说明	查看系统日志参数配置。
相关参数	-
实例	server-log-config
功能说明	查看系统日志参数配置。

9.3.57 update-server-log-config

功能说	更新系统日志参数配置。
-----	-------------

明	
相关参数	<p><code>--verbose</code>: 是否在命令行输出系统日志信息。可选。</p> <p><code>--rotation</code>: 是否开启系统日志的轮转。可选。</p> <p><code>--rotationtype</code>: 系统日志轮转类型, 可选值为 <code>bysize</code>、<code>bytime</code>、<code>byday</code>。可选。</p> <p><code>--rotationvalue</code>: 系统日志轮转数值, 该数值由数字和单位组成。系统日志轮转类型为 <code>bysize</code> 时, 单位可选 KB 或 MB, 当配置超过 100MB 时, 以 100MB 为准; 系统日志轮转类型为 <code>bytime</code> 时, 单位可选 H 或 D, 当配置超过 10D (10 天) 时, 以 10D 为准; 系统日志轮转类型为 <code>byday</code> 时, 不需要填写该数值。</p> <p><code>--filecount</code>: 系统日志超过该数量后则会自动删除较早的日志文件。可选。</p>
实例	<pre>update-server-log-config --verbose=true --rotation=true --rotationtype=bysize --rotationvalue=100MB --filecount=50</pre>
功能说明	更新系统日志参数配置。

9.3.58 set-jvm-arg

功能说明	配置启动脚本 JVM 参数。
相关参数	<code>--arg</code> : jvm 参数。必选, 每次只能加一个参数。
实例	<pre>set-jvm-arg --arg=-Xmx1024m</pre>
功能说明	配置启动脚本 JVM 参数。

9.3.59 delete-jvm-arg

功能说明	删除启动脚本 JVM 参数。
相关参数	<code>--arg</code> : jvm 参数。必选。
实例	<pre>delete-jvm-arg --arg=-Xmx1024m</pre>
功能说明	删除启动脚本 JVM 参数。

9.3.60 set-server-arg

功能说明	设置启动脚本服务器参数。
------	--------------

相关参数	-Dxxx=yyy 服务器参数名和值。必选。 xxx 为参数名, yyy 为参数值
实例	set-server-arg -DWebModuleOnly=true
功能说明	设置启动脚本服务器参数。

9.3.61 delete-server-arg

功能说明	设置启动脚本服务器参数。
相关参数	xxx : 服务器参数名。必选。
实例	delete-server-arg WebModuleOnly
功能说明	设置启动脚本服务器参数。

9.3.62 update-auto-deploy-config

功能说明	修改自动部署开关
相关参数	--autoDeployEnabled : 自动部署开关。Boolean 型, 必选参数, 默认 false。 --autoDeployDir: 自动部署路径。必选参数。默认空字符串。 --autoDeployCheckInterval: 自动部署超时时间, 必选参数, 单位毫秒, 默认 3000 --hotDeployEnabled: 热部署开关。Boolean 型, 必选参数, 默认 false
实例	update-auto-deploy-config --autoDeployEnabled=true --autoDeployDir="d:/temp/autodeploy" --autoDeployCheckInterval=4444 --hotDeployEnabled=false auto
验证方式	点击管理控制台左侧导航树中的“服务”, 查看对应属性已修改为设置的值。

10. 附录 10 提供 NativeJdbcExtractor 工具类

提供 NativeJdbcExtractor 工具类用于通过 TongWeb 代理连接获取真实数据库连接对象, 用法如下:

```
try {
    DataSource dataSource = (DataSource) new
InitialContext().lookup("jdbc/test");
    Connection con = dataSource.getConnection();
    System.out.println("逻辑连接: " + con);
}
```

```

    Connection physicalconnection =
NativeJdbcExtractor.extractNativeConnection(con);
    System.out.println("物理连接: " + physicalconnection);
    con.close();
} catch (Exception e) {
    e.printStackTrace();
}
}

```

11. 附录 11 数据库密码加密脚本

TongWeb 提供了加密数据库密码的工具类脚本，password.sh 和 password.bat，可通过命令行将密码明文转为密文。以 linux 平台为例，加密脚本使用示例：

sh password.sh abc123.com，加密结果输出：

```
encrypt 'abc123.com' : mB3GuIpB/InpjY9+EogTKw==
```

如上，“mB3GuIpB/InpjY9+EogTKw==”即为明文“abc123.com”加密后的字符串。

12. 附录 12 通过 JCA 集成 Tuxedo 配置说明

Tuxedo 是 Oracle 提供的一个客户机/服务器的“中间件”产品，它在客户机和服务器之间进行调节，以保证正确地处理事务。它用 C 语言技术开发的并且有很高性能。它具备分布式事务处理和应用通信功能，并提供完善的各种服务来建立、运行和管理关键任务应用系统。开发人员能够用它建立跨多个硬件平台、数据库和操作系统的可互操作的应用系统。而 TongWeb 提供了通过 JCA 的方式来集成 Tuxedo。

下面介绍一个基本的用例场景：Tongweb 端部署 Tuxedo JCA Adapter，创建该资源适配器对应的连接资源，部署到 Tongweb 上的应用中通过 JNDI 名 lookup 到之前创建的连接资源，然后通过连接资源中相应的方法来调用 Tuxedo 端暴露的远程服务。这是一个通过 CCI 调用 JATMI 的典型流程。

12.1 Tuxedo 远端服务配置简介

这里通过 Tuxedo_home\samples\atmi\javaapp\jsimpapp 中的用例来了解一下发布远端服务的配置及步骤：

- 环境变量设置：

```

TUXDIR=/home/ORACLE_HOME/tuxedo12.2.2.0.0/; export TUXDIR
APPDIR=$TUXDIR/samples/atmi/javaapp/jsimpapp;export APPDIR
JAVA_HOME=/usr/java/jdk1.8.0_144/jre; export JAVA_HOME
JVMLIBS=$JAVA_HOME/lib/amd64/server:$JAVA_HOME/bin
PATH=$TUXDIR/bin:$JAVA_HOME/bin:$PATH; export PATH
COBCPY=$TUXDIR/cobinclude; export COBCPY
COBOPT="-C ANS85 -C ALIGN=8 -C NOIBMCOMP -C TRUNC=ANSI -C OSEXT=cbl"; export
COBOPT
SHLIB_PATH=$TUXDIR/lib:$JVMLIBS:$SHLIB_PATH; export SHLIB_PATH
LIBPATH=$TUXDIR/lib:$JVMLIBS:$LIBPATH; export LIBPATH
LD_LIBRARY_PATH=$TUXDIR/lib:$JVMLIBS:$ORACLE_HOME/lib$LD_LIBRARY_PATH; export
LD_LIBRARY_PATH
TUXCONFIG=$APPDIR/tuxconfig; export TUXCONFIG
BDMCONFIG=$APPDIR/bdmconfig; export BDMCONFIG

```

- ubbsimple 及 domconfig 文件配置：编写这两个文件，目的是把编译好的服务 class 文件配置成远程服务。
- 生成 tuxconfig 及 domconfig 文件：在 %APPDIR% 目录下，用 tmloadcf -y ubbsimple 命令生成二进制的 tuxconfig 配置文件，再使用 dmlloadcf -y domconfig 命令生成二进制的 bdmconfig 配置文件

- 启动服务端程序：tmboot -y
- 查看已启动的服务程序：在%APPDIR%目录下输入：tadmin

12.2 com.oracle.tuxedo.TuxedoAdapter.rar 配置

1. 配置 com.oracle.tuxedo.TuxedoAdapter.rar 的 META-INF 下的 ra.xml。

注意事项：

- 使用 tuxedo 自定义的 dmconfig 配置文件，<resourceadapter-class>的值需要写成:com.oracle.tuxedo.adapter.TuxedoResourceAdapter，如：

```
<resourceadapter-class>com.oracle.tuxedo.adapter.TuxedoResourceAdapter</resourceadapter-class>
```

- 配置 dmconfig 文件的位置，即添加<config-property>，如：

```
<config-property>
<config-property-name>dmconfig</config-property-name>
<config-property-type>java.lang.String</config-property-type>
<config-property-
value>/home/wanglc/ORACLE_HOME/tuxedo12.2.2.0.0/samples/atmi/javaapp/jsimpapp/com.
oracle.tuxedo.TuxedoAdapter/dmconfig.xml</config-property-value>
</config-property>
```

2. 配置 com.oracle.tuxedo.TuxedoAdapter.rar 中的 dmconfig.xml 文件，需要注意的是 dmconfig.xml 中的<LocalAccessPoint>和<RemoteAccessPoint>的内容需要和 Tuxedo 服务端的 domconfig 中配置统一；<Import>内配置客户端需要引用的远程服务。
3. 将 com.oracle.tuxedo.TuxedoAdapter.rar 的 META-INF 下的其他 XXX-ra.xml 删除。

12.3 TongWeb 相关部署及配置

1. 部署 com.oracle.tuxedo.TuxedoAdapter.rar。

注意事项：

- 控制台不支持名称带点，所以需要修改名称，如 TuxedoAdapter；
- rar 在重新启动服务器时需要优先于其他类型应用部署，所以在部署此 rar 应用时候，部署顺序属性需要设置比较小的数值，如：1。

2. 配置 JCA 连接池。

通过 tongweb 的控制台“JCA -> JCA 连接池”创建一个连接池，名称为如：tuxedoPool，资源适配器选择如 TuxedoAdapter。

3. 部署客户端应用。

13. 附录 13 Apache 作为 Proxy Server 配置说明

1. 将 Apache2.2 作为 Proxy Server 的使用步骤

1. 启动 TongWeb7 服务器；
2. 通过管理控制台创建 HTTP 或 HTTPS 通道，配置“代理服务器的 URL”为 http(s)://proxyName:proxyPort (proxyName 为 Apache 的地址，proxyPort 为 Apache 的监听端口)，创建 HTTP/HTTPS 通道的步骤见“创建通道”；
3. 将步骤 2 中创建的通道绑定到缺省虚拟主机 (server)；
4. 将应用部署到缺省虚拟主机 server 上。
5. 在 Apache 中配置 Proxy 转发的信息，配置方法如下：

2. 配置基于 HTTP 的 proxy 转发

1. 将<Apache_Install_Dir>/conf/httpd.conf 中 LoadModule 信息去掉注释：

```
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_module modules/mod_proxy.so
```

2. 在<Apache_Install_Dir>/conf/httpd.conf 的尾部添加如下内容，其中 168.1.103.13 是 TongWeb7 中 HTTP 通道的监听地址，9001 为 HTTP 通道的监听端口。test1 为要代理的应用。

```
ProxyPass /test1 http://168.1.103.13:9001/test1
ProxyPassReverse /test1 http://168.1.103.13:9001/test1
```

3. 重启 Apache;
4. 通过 “http://ApacheAddress:Port/test1 (ApacheAddress 是 Apache 的 IP 地址, Port 是 Apache 的 HTTP 监听端口)” 成功访问 test1 应用。

3. 配置基于 HTTPS 的 Proxy 转发

前提: 基于 HTTPS 的 proxy 转发要求 Apache 支持 SSL。

1. 将<Apache_Install_Dir>/conf/httpd.conf 中 LoadModule 信息去掉注释:

```
LoadModule proxy_connect_module modules/mod_proxy_connect.so
LoadModule proxy_http_module modules/mod_proxy_http.so
LoadModule proxy_module modules/mod_proxy.so
```

2. 在<Apache_Install_Dir>/conf/extra/httpd-ssl.conf 中添加如下内容

```
<VirtualHost _default_:443>
.....
ProxyPass /test1 https://168.1.103.13:9003/test1
//其中: 168.1.103.13 是 TongWeb7 HTTPS 通道的监听地址, 9003 为 HTTPS 通道的监听
端口。
.....
</VirtualHost>
```

3. 重启 Apache;
4. 通过 “https://apacheServerName:443/test1 (apacheServerName 是生成证书时设置的域名, 443 是 Apache 的 HTTPS 端口)” 成功访问应用。

4. 在 Win32 平台上安装配置 Apache 2.x with SSL 支持

1. 下载 apache 安装文件
2. 下载 openssl, 并复制到 apache 的 bin 目录下
3. 通过命令行生成 SSL Cert & Key

```
(1)<apache-install-dir>\bin>set OPENSSL_CONF=openssl.cnf
(2)生成证书请求文件 (.csr) 和私密文件 (privkey.pem)
<apache-install-dir>\bin>openssl req -config openssl.cnf -new -out server.csr
命令执行后信息如下:
```

```
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.+++++
....+++++
```

```
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
```

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
```

```
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
-----
Country Name (2 letter code) []:CN
State or Province Name (full name) []:Beijing
Locality Name (eg, city) []:Beijing
Organization Name (eg, company) []:Tongtech
Organizational Unit Name (eg, section) []:TongWeb
Common Name (eg, your websites domain name) []:apacheServerName
Email Address []:aa@apacheServerName
```

Please enter the following 'extra' attributes
to be sent with your certificate request

A challenge password []:changeit

(3)生成 server.key

```
<apache-install-dir>\bin> openssl rsa -in privkey.pem -out server.key //需提供生成 privkey.pem 的密码 (这里为 changeit)
```

(4)生成 server.crt

```
<apache-install-dir>\bin>openssl x509 -in server.csr -out server.crt -req -signkey server.key -days 365
```

4. 将生成的 server.key 和 server.crt (生成的 server.key 和 server.crt 存放在 <apache-install-dir>/bin/) 复制到<apache-install-dir>/conf/

5. 在<apache-install-dir>/conf/httpd.conf 中, 取消下面两行内容的注释:

```
LoadModule ssl_module modules/mod_ssl.so
```

```
Include conf/extra/httpd-ssl.conf
```

6. 测试 SSL 是否成功:<apache-install-dir>\bin>httpd.exe -D SSL -t, 如果显示 Syntax OK 表示安装成功。

7. 重启 Apache;

8. 访问 https://apacheServerName:443/, 页面返回 it works, 则 Apache 的 SSL 支持安装成功。

说明:

Apache 缺省的 HTTPS 端口为 443。如果 HTTPS 端口冲突, 需将 httpd-ssl.conf 中的 443 改成新的端口。修改项包括

```
Listen 443
```

```
<VirtualHost _default_:443>
```

```
ServerName localhost:443
```

14. 附录 14 TongWeb 提供等同 Apache 的 rewrite 重写机制功能配置说明

TongWeb 提供了重写 Valve (Rewrite Valve) 来实现 URL 重写功能, 方式非常类似于 Apache HTTP Server 的 mod_rewrite 模块。

配置:

重写 Valve 使用包含重写指令的 rewrite.config 文件, 且必须放在 Web 应用的 WEB-INF 文件夹中。

TongWeb 通过识别此文件, 自动添加 com.tongweb.catalina.valves.rewrite.RewriteValve 到相应的应用中, 此应用部署后的访问都会受到指令文件规则的处理。

文件内容举例:

```
RewriteCond %{REQUEST_PATH} !-d
```

```
RewriteCond %{REQUEST_PATH} !-f
```

```
RewriteRule ^(.*)$ index.php/$1 [L]
```

表达式的作用是除了一些静态文件、路径之外的请求都会在请求 URL 加上 “index.php” 的前缀来进行访问。如应用前缀配置成 “/” 后, 访问的变化为:
http://127.0.0.1:8088/home/Demo/demo -->

```
http://127.0.0.1:8088/index.php/home/Demo/demo
```

指令

rewrite.config 文件包含一系列指令，这些指令和 Apache 提供的 mod_rewrite 所用的指令很像，尤其是核心的 RewriteRule 与 RewriteCond 指令。

14.1 RewriteCond

格式: RewriteCond TestString CondPattern

RewriteCond 指令定义了一个规则条件。一个或多个 RewriteCond 指令可以优先于 RewriteRule 指令执行。如果 URI 当前状态匹配它的模式，并且满足了这些条件，才会使用下列规则。

下面列出了重写 Valve 专有的变量：

- **REQUEST_PATH**
对应用于映射的完整路径。
- **CONTEXT_PATH**
对应映射的上下文的路径。
- **SERVLET_PATH**
对应 Servlet 路径。
- **THE_REQUEST**
由浏览器发送给服务器的完整 HTTP 请求代码行（比如，GET /index.html HTTP/1.1）。这并不包括任何由浏览器发送的额外报头。
- **REQUEST_URI**
HTTP 请求代码行中所请求的资源（如/index.html）。
- **REQUEST_FILENAME**
与请求相匹配的文件或脚本的完整本地文件系统路径。
- **HTTPS**
当连接使用 SSL/TLS 时，含有文本 "on"，否则含有 "off"。

- **范例：**

- 假如想根据请求头的 User-Agent: 对网站主页进行重写，可以使用下列代码：

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla.*
RewriteRule ^/$ /homepage.max.html [L]
RewriteCond %{HTTP_USER_AGENT} ^Lynx.*
RewriteRule ^/$ /homepage.min.html [L]
RewriteRule ^/$ /homepage.std.html [L]
```

- 说明：如果使用的浏览器将它自身标识为 'Mozilla'（包括 Netscape Navigator、Mozilla，等等），那么这就是内容最大化主页（max homepage。它可以包含框架或其他特性）；如果使用的是 Lynx 浏览器（基于终端的），那么获得的就是内容最小化的主页（min homepage）——专为便于浏览文本而设计的版本；如果这些条件都不适用（你使用的是其他浏览器，或者你的浏览器将其自身标识为非标准内容），那么得到的就是标准主页（std homepage）。

14.2 RewriteRule

格式: RewriteRule Pattern Substitution

RewriteRule 指令是重写机制的核心。此指令可以多次使用，每个实例都定义一个单独的重写规则。这些规则的定义顺序尤为重要，因为这是在运行时应用它们的顺序。

模式是一个作用于当前 URL 的兼容 perl 的正则表达式。如：

```
RewriteRule ^(.*)$ index.php/$1 [L]
```