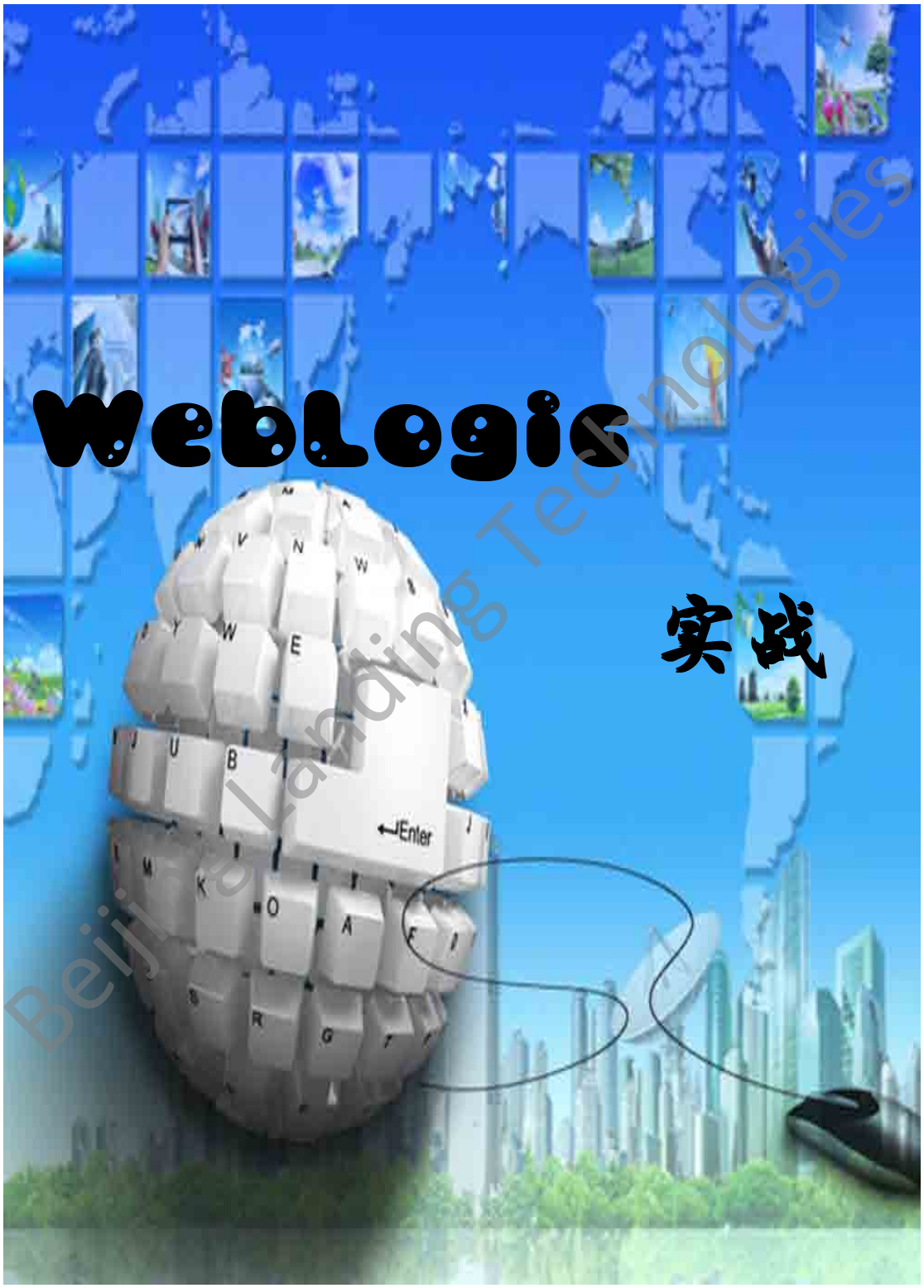
联动北方
LANDING

企业级 IT 运维宝典系列



联动北方 出品

目　录

第 10 章 与开源 SSH 框架的兼容

随着现在开源框架的越来越普及，许多真实的应用项目基于 SSH 进行开发，而在从开发系统往生产系统迁移时，WebLogic 与 SSH 框架的兼容性问题就浮出水面。

10.1 MVC 模型

在讨论 SSH 框架相关内容之前，有必要对于现在开源框架的核心底层模型做一个适当的了解和回顾。

10.1.1 MVC 简介

MVC (Modal View Controler)本来是存在于 Desktop 程序中的，M 是指数据模型，V 是指用户界面，C 则是控制器。使用 MVC 的目的是将 M 和 V 的实现代码分离，从而使同一个程序可以使用不同的表现形式。比如一批统计数据你可以分别用柱状图、饼图来表示。C 存在的目的则是确保 M 和 V 的同步，一旦 M 改变，V 应该同步更新。

模型-视图-控制器（MVC）是 Xerox PARC 在八十年代为编程语言 Smalltalk－80 发明的一种软件设计模式，至今已被广泛使用。最近几年被推荐为 Sun 公司 J2EE 平台的设计模式，并且受到越来越多的使用 ColdFusion 和 PHP 的开发者的欢迎。模型-视图-控制器模式是一个有用的工具箱，它有很多好处，但也有一些缺点。

10.1.2 MVC 如何工作

MVC 是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。使用 MVC 应用程序被分成三个核心部件：模型、视图、控制器，它们各自处理自己的任务。

1. 视图

视图是用户看到并与之交互的界面。对老式的 Web 应用程序来说，视图就是由 HTML 元素组成的界面，在新式的 Web 应用程序中，HTML 依旧在视图中扮演 着重要的角色，但一些新的技术已层出不穷，它们包括 Macromedia Flash 和象 XHTML，XML/XSL，WML 等一些标识语言和 Web services。如何处理应用程序的界面变得越来越有挑战性。MVC 一个大的好处是它能为你的应用程序处理很多不同的视图。在视图中其实没有真正的处理发生，不管这些数据是联机存储的还是一个雇员列表，作为视图来讲，它只是作为一种输出数据并允许用户操纵的方式。

2. 模型

模型表示企业数据和业务规则。在 MVC 的三个部件中，模型拥有最多的处理任务。例如它可能用像 EJB 和 ColdFusion Component 这样的构件对象来处理数据库。被模型返回的数据是中立的，就是说模型与数据格式无关，这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用，所以减少了代码的重复性。

3. 控制器

控制器接受用户的输入并调用模型和视图去完成用户的需求。所以当单击 Web 页面中的超链接和发送 HTML 表单时，控制器本身不输出任何东西和做任何处理。它只是接收请求并决定调用哪个模型构件去处理请求，然后用确定用哪个视图来显示模型处理返回的数据。

现在我们总结 MVC 的处理过程，首先控制器接收用户的请求，并决定应该调用哪个模型来进行处理，然后模型用业务逻辑来处理用户的请求并返回数据，最后控制器用相应的图格式化模型返回的数据，并通过表示层呈现给用户。

10.1.3 为什么要使用 MVC

大部分 Web 应用程序都是用像 ASP，PHP，JSP，或者 CFML 这样的过程化语言来创建的。它们将像数据库查询语句这样的数据层代码和像 HTML 这样的表示层 代码混在一起。经验比较丰富的开发者会将数据从表示层分离开来，但这通常不是很容易做到的，它需要精心的计划和不断的尝试。MVC 从根本上强制性的将它们分开。

尽管构 MV 应用程序需要一些额外的工作，但是它给我们带来的好处是无庸质疑的。首先，最重要的一点是多个视图能共享一个模型，正如我所提及的，现在需要用越来越多的方式来访问你的应用程序。对此，其中一个解决之道是使用 MVC，无论你的用户想要 Flash 界面或是 WAP 界面；用一个模型就能处理它们。由于你已经将数据和业务规则从表示层分开，所以你可以最大化的重用你的代码了。由于模型返回的数据没有进行格式化，所以同样的构件能被不同界面使用。例如，很多数据可能用 HTML 来表示，但是它们也有可能要用 Macromedia Flash 和 WAP 来表示。模型也有状态管理和数据持久性处理的功能，例如，基于会话的购物车和电子商务过程也能被 Flash 网站或者无线联网的应用程序所重用。因为模型是自包含的，并且与控制器和视图相分离，所以很容易改变你的应用程序的数据层和业务规则。如果你想把你的数据库从 MySQL 移植到 Oracle，或者改变你的基于 RDBMS 数据源到 LDAP，只需改变你的模型即可。一旦你正确的实现了模型，不管你的数据来自数据库或是 LDAP 服务器，视图将会正确的显示它们。

由于运用 MVC 的应用程序的三个部件是相互对立，改变其中一个不会影响其它两个，所以依据这种设计思想你能构造良好的松偶合的构件。控制器的也提供了一个好处，就是可以使用控制器来联接不同的模型和视图去完成用户的需求，这样控制器可以为构造应用程序提供强有力的手段。给定一些可重用的模型和视图，控制器可以根据用户的需求选择模型进行处理，然后选择视图将处理结果显示给用户。

10.1.4 MVC 的缺点

MVC 的缺点是由于它没有明确的定义，所以完全理解 MVC 并不是很容易。使用 MVC 需要精心的计划，由于它的内部原理比较复杂，所以需要花费一些时间去思考。你将不得不花费相当可观的时间去考虑如何将 MVC 运用到你的应用程序，同时由于模型和视图要严格的分离，这样也给调试应用程序到来了一定的困难。每个构件在使用之前都需要经过彻底的测试。一旦你的构件经过了测试，你就可以毫无顾忌的重用它们了。

根据经验，由于我们将一个应用程序分成了三个部件，所以使用 MVC 同时也意味着你将要管理比以前更多的文件，这一点是显而易见的。这样好像我们的工作量增加了，但是请记住这比起它所能带给我们的好处是不值一提。 MVC 并不适合小型甚至中等规模的应用程序，花费大量时间将 MVC 应用到规模并不是很大的应用程序通常会得不偿失。

MVC 设计模式是一个很好创建软件的途径，它所提倡的一些原则，像内容和显示互相分离可能比较好理解。但是如果你要隔离模型、视图和控制器的构件，你可能需要重新思考你的应用程序，尤其是应用程序的构架方面。如果你肯接受 MVC，并且有能力应付它所带来的额外的工作和复杂性，MVC 将会使你的软件在健壮性，代码重用和结构方面上一个新的台阶。

10.2 开源框架综述

10.2.1 struts 简介

Struts 是 Apache 基金会 Jakarta 项目组的一个 Open Source 项目，它采用 MVC 模式，能够很好地帮助 java 开发者利用 J2EE 开发 Web 应用。和其他的 java 架构一样，Struts 也是面向对象设计，将 MVC 模式"分离显示逻辑和业务逻辑"的能力发挥得淋漓尽致。Structs 框架的核心是一个弹性的控制层，基于如 Java Servlet，JavaBean，ResourceBundle 与 XML 等标准技术，以及 Jakarta Commons 的一些类库。Struts 由一组相互协作的类（组件）、Servlet 以及 jsp tag lib 组成。基于 struts 构架的 web 应用程序基本上符合 JSP Model2 的设计标准，可以说是一个传统 MVC 设计模式的一种变化类型。

1. Model 部分

由 JavaBean 组成，ActionForm 用于封装用户的请求参数，封装成 ActionForm 对象，该对象被 ActionServlet 转发给 Action，Action 根据 ActionFrom 里面的请求参数处理用户的请求。

JavaBean 则封装了底层的业务逻辑，包括数据库访问等。

2. View 部分

该部分采用 JSP 实现。

Struts 提供了丰富的标签库，通过标签库可以减少脚本的使用，自定义的标签库可以实现与 Model 的有效交互，并增加了现实功能。

3. Controller 组件

Controller 组件有两个部分组成——系统核心控制器，业务逻辑控制器。

系统核心控制器，对应 Structs 中的 ActionServlet。该控制器由 Struts 框架提供，继承 HttpServlet 类，因此可以配置成标注的 Servlet。该控制器负责拦截所有的 HTTP 请求，然后根据用户请求决定是否要转给业务逻辑控制器。

业务逻辑控制器，负责处理用户请求，本身不具备处理能力，而是调用 Model 来完成处理，对应 Action 部分。

10.2.2 Spring 简介

Spring 是一个开源框架，它由 Rod Johnson 创建。它是为了解决企业应用开发的复杂性而创建的。Spring 使用基本的 JavaBean 来完成以前只可能由 EJB 完成的事情，然而，Spring 的用途不仅限于服务器端的开发。从简单性、可测试性和松耦合的角度而言，任何 Java 应用都可以从 Spring 中受益。

- 目的：解决企业应用开发的复杂性
- 功能：使用基本的 JavaBean 代替 EJB，并提供了更多的企业应用功能
- 范围：任何 Java 应用

简单来说，Spring 是一个轻量级的控制反转(IoC)和面向切面(AOP)的容器框架。

1. 轻量

从大小与开销两方面而言 Spring 都是轻量的。完整的 Spring 框架可以在一个大小只有 1MB 多的 JAR 文件里发布。并且 Spring 所需的处理开销也是微不足道的。此外，Spring 是非侵入式的：典型地，Spring 应用中的对象不依赖于 Spring 的特定类。

2. 控制反转

Spring 通过一种称作控制反转（IoC）的技术促进了松耦合。当应用了 IoC，一个对象依赖的其它对象会通过被动的方式传递进来，而不是这个对象自己创建或者查找依赖对象。你可以认为 IoC 与 JNDI 相反--不是对象从容器中查找依赖，而是容器在对象初始化时不等对象请求就主动将依赖传递给它。

3. 面向切面

Spring 提供了面向切面编程的丰富支持，允许通过分离应用 的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的--完成业务逻辑--仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

4. 容器

Spring 包含并管理应用对象的配置和生命周期，在这个意义上它是一种容器，你可以配置你的每个 bean 如何被创建——基于一个可配置原型（prototype），你的 bean 可以创建一个单独的实例或者每次需要时都生成一个新的实例——以及它们是如何相互关联的。然而，Spring 不应该被混同于传统的重量级的 EJB 容器，它们经常是庞大与笨重的，难以使用。

5. 框架

Spring 可以将简单的组件配置、组合成为复杂的应用。在 Spring 中，应用对象被声明式地组合，典型地是在一个 XML 文件里。Spring 也提供了很多基础功能（事务管理、持久化框架集成等等），将应用逻辑的开发留给了使用者。

所有 Spring 的这些特征使你能够编写更干净、更可管理、并且更易于测试的代码。它们也为 Spring 中的各种模块提供了基础支持。Spring 是潜在地一站式解决方案，定位于与典型应用相关的大部分基础结构。它也涉及到其他 framework 没有考虑到的内容，并有如下具体特点

- 方便解耦，简化开发

通过 Spring 提供的 IoC 容器，我们可以将对象之间的依赖关系交由 Spring 进行控制，避免硬编码所造成的过度程序耦合。有了 Spring，用户不必再为单实例模式类、属性文件解析等这些很底层的需求编写代码，可以更专注于上层的应用。

- AOP 编程的支持

通过 Spring 提供的 AOP 功能，方便进行面向切面的编程，许多不容易用传统 OOP 实现的功能可以通过 AOP 轻松应付。

- 声明式事务的支持

在 Spring 中，我们可以从单调烦闷的事务管理代码中解脱出来，通过声明式方式灵活地进行事务的管理，提高开发效率和质量。

- 方便程序的测试

可以用非容器依赖的编程方式进行几乎所有的测试工作，在 Spring 里，测试不再是昂贵的操作，而是随手可做的事情。

- 方便集成各种优秀框架

Spring 不排斥各种优秀的开源框架，相反，Spring 可以降低各种框架的使用难度，Spring 提供了对各种优秀框架（如 Struts,Hibernate、Hession、Quartz）等的直接支持。

- 降低 J2EE API 的使用难度

Spring 对很多难用的 J2EE API（如 JDBC，JavaMail，远程调用等）提供了一个薄薄的封装层，通过 Spring 的简易封装，这些 J2EE API 的使用难度大为降低。

10.2.3 Hibernate 简介

Hibernate 是一个开放源代码的对象关系映射框架，它对 JDBC 进行了非常轻量级的对象封装，使得 Java 程序员可以随心所欲的使用对象编程思维来操作数据库。 Hibernate 可以应用在任何使用 JDBC 的场合，既可以在 Java 的客户端程序使用，也可以在 Servlet/JSP 的 Web 应用中使用，最具革命 意义的是，Hibernate 可以在应用 EJB 的 J2EE 架构中取代 CMP，完成数据持久化的重任。

Hibernate 的核心接口一共有 5 个，分别为:Session、SessionFactory、Transaction、Query 和 Configuration。这 5 个核心接口在任何开发中都会用到。通过这些接口，不仅可以对持久化对象进行存取，还能够进行事务控制。下面对这五个核心 接口分别加以介绍。

1. Session 接口

Session 接口负责执行被持久化对象的 CRUD 操作(CRUD 的任务是完成与数据库的交流，包含了很多常见的 SQL 语句。)。但需要注意的是 Session 对象是非线程安全的。同时，Hibernate 的 session 不同于 JSP 应用中的 HttpSession。这里当使用 session 这 个术语时，其实指的是 Hibernate 中的 session，而以后会将 HttpSession 对象称为用户 session。

2. SessionFactory 接口

SessionFactory 接口负责初始化 Hibernate。它充当数据存储源的代理，并负责创建 Session 对象。这里用到了工厂模式。需要注 意的是 SessionFactory 并不是轻量级的，因为一般情况下，一个项目通常只需要一个 SessionFactory 就够，当需要操作多个数据库时，可以为每个数据库指定一个 SessionFactory。

3. Configuration 接口

Configuration 接口负责配置并启动 Hibernate，创建 SessionFactory 对象。在 Hibernate 的启动的过程中，Configuration 类的实例首先定位映射文档位置、读取配置，然后创建 SessionFactory 对象。

4. Transaction 接口

Transaction 接口负责事务相关的操作。它是可选的，开发人员也可以设计编写自己的底层事务处理代码。

5. Query 和 Criteria 接口

Query 和 Criteria 接口负责执行各种数据库查询。它可以使用 HQL 语言或 SQL 语句两种表达方式。

10.3 WebLogic 与 Spring 的兼容性

企业级 Spring Framework 的非侵入性 IoC 研发模型不仅依赖于对 J2EE 应用服务器可用的特性集，而且旨在补充该特性集。事实上，在苛刻的生产环境中，底层应用服务器基础架构所提供的服务质量对于 Spring 应用程式的可靠性、可用性和性能非常重要。WebLogic Server 所提供的企业级特性能增强 Spring 应用程式的所有方面。

10.3.1 集群管理和部署

一个 WebLogic Server 集群包括多个 WebLogic Server 服务器实例，这些服务器实例同时运行并一起工作，从而提高了可伸缩性和可靠性。对客户端来说是透明的，集群对外就像单个的 WebLogic Server 实例一样。构成集群的服务器实例既能运行在同一台机器上，也能位于不同的机器上。能通过在现有的机器上向集群添加另外的服务器实例，或向集群添加机器以驻留增加的服务器实例，来提高集群的容量。

WebLogic Server 集群为 Spring 应用程式提供了一个企业级的部署平台，虽然其他的技术产品也支持类似的特性，不过他们不具有 WebLogic Server 所提供的丰富性和易用性。Spring 应用程式通常都被打包为 web 应用程式，这种情况下，要利用 WebLogic Server 集群就无需修改应用程式。只要把应用程式部署到集群中的服务器上，就能获得增强的可伸缩性和可用性

10.3.2 Spring 会话复制

Spring Web 应用程式习惯在 HTTP 会话中保存信息，比如订单 ID 和用户信息。为了支持集群中 servlet 和 JSP 的自动复制和故障恢复，WebLogic Server 支持几种用于保持 HTTP 会话状态的机制。只要为应用程式提供正确的 weblogic.xml 部署描述符，Spring Web 应用程式就能非侵入性地使用这些机制。

10.3.3 集群化的 Spring 远程控制

Spring 提供功能强大的远程控制支持，允许用户轻松导出和使用远程服务，同时仍然能利用基于 POJO 的一致编程模型。通过一个接合到适当的 Spring bean 的 RMI 接口，Vanilla Spring 支持代理 POJO 调用。然而，这种支持仅限于 JRMP（Sun 的 RMI 实现），或通过 JndiRmiProxyFactoryBean 使用特定的远程接口。

借助于 Spring on WebLogic Server 认证，我们已扩展了 JndiRmiProxyFactoryBean 和相关的服务导出程式——这样他就能支持所有 J2EE RMI 实现的 POJO 代理，包括 RMI-IIOP 和 t3。

这方面的支持还包括一个 WebLogic RMI 部署描述符，他支持代理 RMI 接口上的集群化，所以 POJO 调用能跨一个 WebLogic Server 集群进行负载均衡。集群化的描述符是自动包含在内的，只需要以适当方式设置集群和将 Spring 应用程式部署到所有集群成员中。

10.3.4 对 Spring 组件的控制台支持

Spring on WebLogic Server 工具包中包含一个 WebLogic Server 控制台扩展，他显示了定义在应用程式中的 Spring bean、属性和操作。他构建在 WebLogic 控制台扩展门户框架之上，该框架能变换 WebLogic Administration 控制台的外观、功能和布局，而无需修改服务器或控制台代码。将控制台扩展复制到 yourdomain/console-ext 目录下，则重新启动服务器时就部署了控制台扩展。（参考 Spring on WebLogic Server 工具包）。

该扩展自动为不是 MBean 的 Spring bean（大多数 Spring bean）创建（JMX）管理接口，然后在 applicationContext.xml 中设置一个 MbeanExporter，并指定哪些 bean 要通过该 exporter 公开，这样控制台扩展就运行了。这项特性是 Spring 和 WebLogic Server 进行无缝和非侵入性合作的一个良好例证。要使应用程式支持 JMX，只需修改应用程式上下文部署描述符。要使控制台支持 Spring，只需将一个简单的 jar 部署到现有的域即可。

（另）WebLogic10.3添加支持 Spring 的控制台扩展

a. 登录到管理控制台。

b. 在控制台中，单击工具栏上的首选项 。

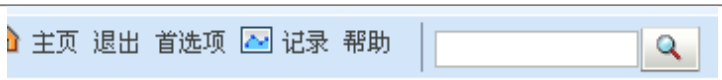


图 10-1

c. 在选项页，单击 扩展 。

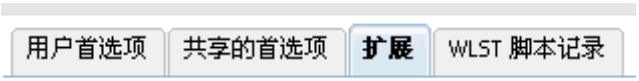


图 10-2

d. 选择复选框旁边的 Spring—console ，然后点击 启用 。

图 10-3

e. 停止服务器，然后重新启动使更改生效。

10.3.5 Web 服务支持

Spring 远程控制功能的另一个方面是他对 RPC 风格 Web 服务的支持。WebLogic Server 提供基于 Ant 的工具，用于基于 Web 服务的 WSDL 描述生成 JAX-RPC 存根。Web 服务客户端使用这些生成的存根来获取代表服务器端操作的一个远程接口。Spring 提供了一个 JaxRpcPortProxyFactoryBean 来简化了这个过程。我们发现，在 WebLogic Server 环境中设置 JaxRpcPortProxyFactoryBean 有些棘手，所以为了节约客户的时间，我们给出下面这个代码片断，演示怎么为一个包含复杂类型的 Document Literal 风格的 Web 服务设置代理生成。

大部分属性都是顾名思义自解释的。其中有一些属性比较重要：

1) serviceInterface 是 Spring 的 setter 注入的副产品。这个类将表示 Web 服务操作；

2) customProperties 属性支持制定的 WebLogic Server Web 服务存根属性；

3) jaxRpcService 值被设置为 WebLogic Server 生成的 JAX-RPC 实现服务。JAX-RPC 服务负责验证 Web 服务和加载复杂的类型映射。为了实现后者，必须把 WebLogic Server 的 JAX-RPC 服务实现设置为 Spring bean。这确保了 JAX-RPC 服务构造函数的执行，这也是加载类型映射文件的地方。

把 JaxRpcPortProxyFactoryBean 上的 lookupServiceOnStartup 设置为 false，能关闭启动期间的 JAX-RPC 服务查找。这样，查找将在首次访问时进行。这对于和 WebLogic Server 的可靠请求/响应 Web 服务通信的客户端来说是必须的，而且此处的客户端也必须是个 Web 服务。通常在这些情况下，始发客户端是和 Web 服务客户端一起部署的。因为直到应用程式部署完成才会激活 Web 服务，所以客户端 Web 服务对于 Spring 的上下文加载是不可用的。

10.3.6 安全性

WebLogic Server 安全系统支持和扩展了 J2EE 安全性，同时提供一组丰富的安全提供程式，你能对他们进行制定，然后使用他们来处理不同的安全性数据库或安全性策略。除了使用标准的 J2EE 安全性之外，应用程式程式员还能使用非常多专有扩展，这些扩展使应用程式能和安全系统紧密集成。

WebLogic Server 带有几个安全提供程式，例如，能选择包含大部分流行 LDAP 服务器的身份验证数据库、Active Directory、本地视窗系统和一个内置的身份验证解决方案。能使用制定的提供程式对内置的提供程式进行扩充，从而几乎能和任意身份验证数据库、授权机制和凭证映射服务相集成。因为部署为 webapp 的 Spring 应用程式使用的是 J2EE 安全性，所以无需修改应用程式就能获得 WebLogic Server 的安全性好处。

经验丰富的 Spring 用户还会熟悉 Acegi-Spring 自身的安全框架。目前，能在应用程式中使用 Acegi、WebLogic Server 安全性，或同时使用二者，因为他们是相互独立的。

10.3.7 分布式事务

Spring 为事务管理提供了基础架构。除了对各家数据库供给商提供支持之外，Spring 还通过一家 J2EE 供给商的 JTA 实现支持分布式事务。通过 WebLogicJtaTransactionManager，能把 Spring 的 JTA 管理器设置为和 WebLogic Server 的 JTA 实现一起工作。

WebLogicJtaTransactionManager 把责任直接委派给 WebLogic Server 的 Java Transaction API。WebLogic Server 的 JTA TransactionManager 接口能通过 JNDI 为客户端和 bean 提供者所用，而由 Spring 来管理这种交互。事务管理器还支持事务的作用域；事务能作用于集群和域内部或二者之间。

WebLogicJtaTransactionManager 最强大的特性是管理分布式事务的能力和用于企业应用程式的两阶段提交协议。通过采用 WebLogicJtaTransactionManager，应用程式能通过 WebLogic Administration Console 来进行事务监视。WebLogicJtaTransactionManager 还支持按数据库(per-database)隔离级别，这种级别支持复杂的事务设置。

10.3.8 Java Management Extension

Java Management Extension（Java 管理扩展，JMX）是用于监视和管理 Java 应用程式的规范。他使一般的管理系统能够监视应用程式，当应用程式需要注意时发出通知，并修改应用程式状态来补救问题。Spring 提供广泛的 JMX 支持，包括通过 Spring 的 MBeanServerConnectionFactoryBean 公开 WebLogic Server 的 MBeanServer 的能力。MBeanServerConnectionFactoryBean 是个使用方便的工厂，他附带了一个 MBeanServerConnection。在应用程式部署期间，连接被建立并缓存，以便稍后由引用 bean 对其进行操作。

能设置 MBeanServerConnectionFactoryBean，使其返回 WebLogic Server 的 Runtime MBean Server，他会公开特定 WebLogic Server 实例的监视、运行时控制和活动设置。这包括对 WebLogic Server 的 Diagnostics Framework 的访问。此外，Runtime MBean 还为当前服务器提供对运行时 MBean 和活动设置 MBean 的访问。

还能设置 MBeanServerConnectionFactoryBean，获得一个到 WebLogic Server 的 Domain Runtime MBean Server 的连接。Domain Runtime MBean Server 提供对域范围内服务的访问，比如应用程式部署、JMS 服务器和 JDBC 数据源。他还是访问域中所有服务器的所有运行时 MBean 和活动设置 MBean 层次结构的单点。这个 MBean Server 还用作访问位于托管服务器上的 MBean 的单点。

此外，能设置 MBeanServerConnectionFactoryBean，获得一个到 WebLogic Server 的 Edit MBean Server 的连接。Edit MBean Server 为管理当前 WebLogic Server 域设置提供入口点。

注意，WebLogic Server 的 Domain Runtime MBean Server 在部署期间不是活动的。因此，需要使用延迟初始化来设置 bean，他会在调用 bean 时获取该 bean。

10.3.9 WebLogic Server 上的 Spring Framework 版本兼容

下面分别对此进行详细的列表描述。

10.3.9.1 WLS9.x

编号	描述和变通方法或解决方案	相关版本
CR242675	在 RMI 类加载器中发生了 NullPointerException。 变通方法或解决方案： 请与 BEA 客户支持联系以获取 WebLogic Server/Spring 合并修补程序。	9.0、9.2
CR236708	在 Hibernate 3 和 WebLogic Server 之间存在 Antlr 冲突。 变通方法或解决方案： 将 Antlr2.7.5.jar 放在 CLASSPATH 中的 weblogic.jar 之前。	8.1SP05、9.0、9.2
CR242923	T3 运行时无法对包含基元类型的类描述符进行解码。 变通方法或解决方案： 请与 BEA 客户支持联系以获取 WebLogic Server-Spring 合并修补程序。	9.0、9.2
CR242883	IIOP 运行时无法对包含基元类型的类描述符进行解码。 变通方法或解决方案： 请与 BEA 客户支持联系以获取 WebLogic Server-Spring 合并修补程序。	9.0、9.2
CR237532	Spring Framework 存在 Web 应用程序类加载问题。 变通方法或解决方案： 请与 BEA 客户支持联系以获取 WebLogic Server-Spring 合并修补程序。	8.1SP05、9.0、9.2
CR241195	在 Spring Pet Clinic 示例应用程序中更新记录会导致以下错误： java.lang.IllegalStateException: Cannot access session scope since the requested page does not participate in a session. at weblogic.servlet.jsp.PageContextImpl.getAttribute(PageContextImpl.java:273) at javax.servlet.jsp.jstl.core.Config.get(Config.java	9.0、9.2

	:145) at javax.servlet.jsp.jstl.core.Config.find(Config.java:393) at org.apache.taglibs.standard.tag.common.fmt.TimeZoneSupport.getTimeZone(TimeZoneSupport.java:140) 变通方法或解决方案： 将 includes.jsp 文件中的第一行标记为注释。	
CR244683	HP-UX 需要 jdk150_01，而不是 jdk150_03。 变通方法或解决方案： 在 medrec-spring 目录中，使用 jdk150_01 替换 jdk150_03。	9.0、9.2
CR244693	当您从远程计算机上访问 MedRec-Spring 时，MedRec-Spring 退出功能不起作用。 变通方法或解决方案： 不从远程计算机访问 MedRec-Spring 应用程序，并且不将 localhost 用于请求重定向。	9.0、9.2
CR244691	对 WebLogic 管理控制台的 Spring 扩展仅支持 Web 应用程序（.war）文件，无法用于监视非 .war 文件（如 MedRec-Spring）中的 Spring Bean。	9.0、9.2
CR243957	使用 CTRL-C 关闭 WebLogic Server 时，如果正在破坏 bean domainMBeanServerConnection，则可能会发生关闭异常。 变通方法或解决方案： 使用标志 -Dweblogic.slc=true 以便确定启动和停止 domainRuntimeServerService 的时间。	9.0、9.2
CR280985	无法通过将 countries_mbeans.war 应用程序复制到 WebLogic Server 域目录的 autodeploy 目录来自动部署该应用程序。countries_mbeans.war Web 应用程序是一个 Spring 测试扩展应用程序。 变通方法或解决方案： 使用 WebLogic Server 管理控制台来部署 countries_mbeans.war Web 应用程序，而不是自动部署。	9.2

CR301115	在 Spring Pet Clinic 示例应用程序中运行单元测试会导致以下错误： 从 weblogic.xml.jaxp.RegistrySAXTransformerFactory 中找不到有效的处理器版本实现 变通方法或解决方案： 通过将以下条目添加到 $java.home/lib/jaxp.properties 文件来定义 XML 解析器类： ▪ javax.xml.transform.TransformerFactory=org.apache.xalan.processor.TransformerFactoryImpl ▪ javax.xml.xpath.XPathFactory=org.apache.xpath.jaxp.XPathFactoryImpl ▪ javax.xml.parsers.SAXParserFactory=org.apache.xerces.jaxp.SAXParserFactoryImpl ▪ javax.xml.parsers.DocumentBuilderFactory=org.apache.xerces.jaxp.DocumentBuilderFactoryImpl	9.2
CR300748	访问部署到 WebLogic Server 9.2 的 tiles-samples 时会出现异常。	9.2

表 10-1

10.3.9.2 WLS10.x

在该版本上 Spring Framework 2.0.2经过了官方认证。

编号	描述和变通方法或解决方案	相关版本
CR319968	使用 JRockit 时，OpenJPA 的 ClassFileTranformer 工作不正常。 方案： 在编译阶段进行 enhence 而不要使用 LoadTimeWeaver 在系统加载阶段 enhence.	10.0
CR320649	使用 JRockit 时，在 WebLogic 上部署 Spring Pet Clinic 样例程序失败，产生和 OpenJPA 相关的异常。 方案： 将 JRockit 版本从 R26.4升级到 R27.2.	10.0

表 10-2

10.3.10 Spring 中遇到的问题

抛砖引玉，这里先总结一个实际项目的测试结果。

测试环境：jdk1.6 、Spring2.0.5、Spring2.5；中间件：weblogic10.3.1、weblogic8.1

经过测试，Spring2.0.5 在 weblogic8.1 表现良好。在部署的过程中没有出现什么问题。Spring2.5 在 weblogic8.1 下就不能部署而且每次在报错，

Spring2.5 在 Weblogic10.3.1 表现良好。在部署的过程中没有出现什么问题，Spring2.0.5 在 weblogic10.3.1 下也会出现报错现象不能部署。

在 orcale 的官方网站上可以确认：

- wls 10.3 支持的 Spring 2.5.x
- wls 10.0 支持的 Spring 2.0.x
- wls 9.2 支持的 Spring 1.2.8、2.0.x、2.5.x

10.3.10.1版本兼容性的问题

当 Spring 的版本升级到 2.5.6 的时候但是还是出现了报错现象错误提示是：

```
11:23:59,656 ERROR XFireServlet:51 - Error initializing XFireServlet.
org.springframework.beans.factory.BeanDefinitionStoreException: Unexpected exception parsing XML document from class path resource [org/codehaus/xfire/spring/xfire.xml]; nested exception is java.lang.ClassCastException: weblogic.xml.jaxp.RegistryDocumentBuilderFactory cannot be cast to javax.xml.parsers.DocumentBuilderFactory
```

示例 10-1

这个错误提示是说 WebLogic 的 RegistryDocumentBuilderFactory 类和 java 中 xml.parsers.DocumentBuilderFactory 类在编译的时候发生冲突。

这样的错误信息提示首先找到项目中的冲突包移除就没有什么了，所以我在项目中移除了两个 jar 包：wstx-asl-3.2.0.jar 和 xml-apis-1.0.b2.jar，这样以实现成功部署。

10.3.10.2关于项目开始出现的 nullbean 问题

这个问题主要是 Spring 的框架中的监听器没有加载，也就是在 web.xml 加载的时候就失败啦。在 web.xml 中加载监听 Spring 提供了两种方式，但是达到的效果是一样的。

首先介绍 Spring 的监听器作用：ContextLoaderListener 的作用就是启动 Web 容器时，自动装配 ApplicationContext 的配置信息。因为它实现了 ServletContextListener 这个接口，在 web.xml 配置这个监听器，启动容器时，就会默认执行它实现的方法。

在 ContextLoaderListener 中关联了 ContextLoader 这个类，所以整个加载配置过程由 ContextLoader 来完成。ContextLoader 创建的是 XmlWebApplicationContext 这样一个类，它实现的接口是 WebApplicationContext->ConfigurableWebApplicationContext->ApplicationContext-> BeanFactory 这样一来 spring 中的所有 bean 都由这个类来创建。

如果在 web.xml 中不写任何参数配置信息，默认的路径是"/WEB-INF/applicationContext.xml，在 WEB-INF 目录下创建的 xml 文件的名称必须是 applicationContext.xml。如果是要自定义文件名可以在 web.xml 里加入 contextConfigLocation 这个 context 参数：

```
<context-param>
  <param-name>contextConfigLocation</param-name>
```

```
    <param-value>/WEB-INF/classes/applicationContext-*.xml</param-value>
  </context-param>
```

示例 10-2

在<param-value> </param-value>里指定相应的 xml 文件名，如果有多个 xml 文件，可以写在一起并以“,”号分隔。上面的 applicationContext-*.xml 采用通配符，比如这那个目录下有 applicationContext-ibatis-base.xml，applicationContext-action.xml，applicationContext-ibatis-dao.xml 等文件，都会一同被载入。

10.4 WebLogic 与 Struts 的兼容性

在讨论完 WebLogic 与 Spring 框架的兼容性后，下面详述与 Struts 的兼容性。

10.4.1 调试和日志记录 Struts 应用程序

WebLogic 提供了它自己的日志子系统，该系统是 WebLogic 的缺省日志框架。不过，也可以将 WebLogic 与其它日志框架（如 Log4j）一起使用。缺省情况下，Struts 将把所有消息记录到 WebLogic 日志框架。要在开发 Struts 应用程序过程中进行最有效的记录日志，需要确保将“Logging Message”严重性级别设置为“INFO”；这样将记录所有 INFO 级别以上的日志，从而可以在出错时进行有效的调试。

还建议将所有 system.out.println() 消息都重定向到 WebLogic 日志，以方便调试。可以指定记录这些消息的 stdout 文件。只需编辑 WebLogic Server 脚本，使 JAVA_OPTIONS 变量做以下指定：

```
    -Dweblogic.Stdout="stdout-filename"
    -Dweblogic.Stderr="stderr-filename"
```

示例 10-3

要查看所有发出的 HTTP 请求的日志，可以参考 WebLogic Server 生成的访问日志（access log）。该日志提供了访问请求的一些详细信息，比如时间戳、访问的 URL 和 HTTP 返回代码等。

例如：

```
    127.0.0.1 - - [13/Jul/2004:21:39:46 -0615] "GET /struts-
sample/preregisterCab.do?IndiReport=true HTTP/1.1" 200 4173
    127.0.0.1 - - [13/Jul/2004:21:41:14 -0615] "POST /struts-
sample/getIndiData.do;jsessionid=A05R51Y5jLRJokg8BRMYgLwCOvfuVzC7KEE4AsCZijqD71
6A22Re!-237055073 HTTP/1.1" 200 4667
    127.0.0.1 - - [16/Jul/2004:00:07:58 -0615] "GET /struts-
sample/preregisterCab.do?IndiReport=true HTTP/1.1" 200 4173
    127.0.0.1 - - [16/Jul/2004:00:08:26 -0615] "POST /struts-
sample/getIndiData.do;jsessionid=A32LikMTeXrS5Bnbp7p1TASmcLHkuko9Sudr8LjatWw2Aq
eP6zkt!1759102893 HTTP/1.1" 200 8893
    127.0.0.1 - - [19/Jul/2004:04:39:36 -0615] "GET /struts-
sample/preregisterCab.do?IndiReport=true HTTP/1.1" 200 4173
    127.0.0.1 - - [19/Jul/2004:04:40:16 -0615] "POST /struts-
sample/getIndiData.do;jsessionid=A7yxT3KtTy01T7A39DLSup2RPqxvV9uWay325WxBLqtj1d
lx3Ewa!-431463598 HTTP/1.1" 200 5416
    127.0.0.1 - - [14/Aug/2004:22:36:24 -0615] "GET /struts-
sample/preregisterCab.do?IndiReport=true HTTP/1.1" 200 4173
```

```
    127.0.0.1 - - [14/Aug/2004:22:36:58 -0615] "POST /struts-
sample/getIndiData.do;jsessionid=Be2WKm1nP7DGdX1uIY8PraQdx51wBZTjVSswGHdcbL4njc
jXOdJh!1670519755 HTTP/1.1" 200 4684
    127.0.0.1 - - [14/Aug/2004:22:36:59 -0615] "GET /struts-
sample/pages/STFull.jpg HTTP/1.1" 304 0
```

示例 10-4

10.4.2 调试 WebLogic 类加载器

在开发 Struts 应用程序的过程中，必然会遇到类加载器问题。调试那些 NoClassDefFoundError 和 ClassNotFoundException 的确是一件麻烦事。无法在父类加载器和子类加载器中找到类时，会发生 ClassNotFoundException；这很可能是打包问题造成的。 加载了请求的类但无法找到依赖类时，会抛出 NoClassDefFoundError。父类加载器中的类从不引用子类加载器中的类。类加载器会先请求它们的父类加载器加载类，然后才会尝试自行加载类，因此在此类情况下可能会遇到 NoClassDefFoundError。

为调试 WebLogic 类加载器，WebLogic 提供了类加载器特定的调试标志。设置这些调试标志的方法：

- 编辑 StartWeblogic 脚本，添加以下内容作为启动 WebLogic 时的命令行参数。
  Dweblogic.Debug=debug.lineNumbers,debug.methodNames,weblogic.ClassLoaderVerbose,weblogic.ClassLoader
- 确保在控制台的“Logging”选项中将“Debug to Stdout”选项设置为“enabled”。

启用这些调试选项后，您就会注意到当应用程序加载任何类都将有相应的类似如下的日志：

```
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : ()
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@10d3f0d finder:
weblogic.utils.classloaders.MultiClassFinder@1510d96 annotation: )
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : ()
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@1ce64f6 finder:
weblogic.utils.classloaders.MultiClassFinder@52fecf annotation:
ApplicationClassLoader@)
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : (C:\APP-INF\classes)
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@1d6d61d finder:
weblogic.utils.classloaders.MultiClassFinder@d6ee28 annotation: struts-example@)
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
```

```
    [ChangeAwareClassLoader] :
weblogic.utils.classloaders.ChangeAwareClassLoader@18a6890 finder:
weblogic.utils.classloaders.MultiClassFinder@ad8bb4 annotation: struts-
example@struts-example about to loadClass(org.apache.struts.taglib.html.ImgTag)
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
```

示例 10-5

上例中 Struts 应用程序寻找的是 org.apache.struts.taglib.html.ImgTag 类。首先在父类加载器中寻找该类，然后在应用程序加载器中寻找，最后在 war 类加载器中寻找。如果在类加载器中找不到某个类，您会注意到日志中产生了下面这样的消息:

```
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : ()
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@10d3f0d finder:
weblogic.utils.classloaders.MultiClassFinder@1510d96 annotation: )
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : ()
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@1ce64f6 finder:
weblogic.utils.classloaders.MultiClassFinder@52fecf annotation:
ApplicationClassLoader@)
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : (C:\APP-INF\classes)
    Classloader object id
(weblogic.utils.classloaders.GenericClassLoader@1d6d61d finder:
weblogic.utils.classloaders.MultiClassFinder@d6ee28 annotation: struts-example@)
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    [ChangeAwareClassLoader] :
weblogic.utils.classloaders.ChangeAwareClassLoader@18a6890 finder:
weblogic.utils.classloaders.MultiClassFinder@ad8bb4 annotation: struts-
example@struts-example about to loadClass(org.apache.struts.taglib.html.ImgTag)
    [GenericClassLoader] : Looking for class:
org.apache.struts.taglib.html.ImgTag...
    With classpath of : (C:\bea-plat-
81sp3\user_projects\domains\sqlservdomain\myserver\.wlnotdelete\extract\myserve
r_struts-example_struts-example\jarfiles\WEB-INF\lib\log4j.jar;C:\bea-plat-
81sp3\user_projects\domains\sqlservdomain\myserver\.wlnotdelete\extract\myserve
r_struts-example_struts-example\jarfiles\WEB-INF\lib\log4j-core.jar)
    Classloader object id
(weblogic.utils.classloaders.ChangeAwareClassLoader@18a6890 finder:
weblogic.utils.classloaders.MultiClassFinder@ad8bb4 annotation: struts-
example@struts-example)
```

```
    [GenericClassLoader] : Class org.apache.struts.taglib.html.ImgTag not found.
    <Nov 17, 2004 4:37:52 PM PST> <Error> <HTTP> <BEA-101017>
<[ServletContext(id=28746180,name=struts-example,context-path=/struts-example)]
Root cause of ServletException
    java.lang.NoClassDefFoundError: org/apache/struts/taglib/html/ImgTag
    at jsp_servlet.__index._jspService(__index.java:434)
    at weblogic.servlet.jsp.JspBase.service(JspBase.java:33)
    atweblogic.servlet.internal.ServletStubImpl$ServletInvocationAction.run(Ser
vletStubImpl.java:996)
    at weblogic.servlet.internal.ServletStubImpl.invokeServlet
    (ServletStubImpl.java:419)
    at weblogic.servlet.internal.ServletStubImpl.invokeServlet
    (ServletStubImpl.java:463)
    at weblogic.servlet.internal.ServletStubImpl.invokeServlet
    (ServletStubImpl.java:315)
    at weblogic.servlet.internal.WebAppServletContext
    $ServletInvocationAction.run (WebAppServletContext.java:6452)
    atweblogic.security.acl.internal.AuthenticatedSubject.doAs
    (AuthenticatedSubject.java:321)
```

示例 10-6

10.5 WebLogic 与 Hibernate 的兼容性

下面我们再来看 WebLogic 同著名 Java 的 OR 映射框架 Hibernate 的兼容性，并逐步展开讨论。

10.5.1 Hibernate 中可能遇到的问题

系统抛出下边的异常，导致 WebLogic 异常退出：

```
    org.hibernate.hql.ast.HqlToken  org.hibernate.QueryException:
ClassNotFoundException: org.hibernate.hql.ast.HqlToken [from
com.dhcc.mm5.QbMm5List]
    at org.hibernate.hql.ast.HqlLexer.panic(HqlLexer.java:57)
    at antlr.CharScanner.setTokenObjectClass(CharScanner.java:340)
    at org.hibernate.hql.ast.HqlLexer.setTokenObjectClass(HqlLexer.java:31)
    at antlr.CharScanner.<init>(CharScanner.java:51)
    at antlr.CharScanner.<init>(CharScanner.java:60)
    at org.hibernate.hql.antlr.HqlBaseLexer.<init>(HqlBaseLexer.java:56)
    at org.hibernate.hql.antlr.HqlBaseLexer.<init>(HqlBaseLexer.java:53)
    at org.hibernate.hql.antlr.HqlBaseLexer.<init>(HqlBaseLexer.java:50)
    at org.hibernate.hql.ast.HqlLexer.<init>(HqlLexer.java:26)
    at org.hibernate.hql.ast.HqlParser.getInstance(HqlParser.java:44)
    at org.hibernate.hql.ast.QueryTranslatorImpl.parse
    (QueryTranslatorImpl.java:232)
    at org.hibernate.hql.ast.QueryTranslatorImpl.doCompile
    (QueryTranslatorImpl.java:155)
    at org.hibernate.hql.ast.QueryTranslatorImpl.compile
    (QueryTranslatorImpl.java:109)
    at org.hibernate.engine.query.HQLQueryPlan.<init>(HQLQueryPlan.java:75)
    at org.hibernate.engine.query.HQLQueryPlan.<init>(HQLQueryPlan.java:54)
```

```
at org.hibernate.engine.query.QueryPlanCache.getHQLQueryPlan
(QueryPlanCache.java:71)
at org.hibernate.impl.AbstractSessionImpl.getHQLQueryPlan
(AbstractSessionImpl.java:133)
at org.hibernate.impl.AbstractSessionImpl.createQuery
(AbstractSessionImpl.java:112)
at org.hibernate.impl.SessionImpl.createQuery(SessionImpl.java:1583)
at com.dhcc.mm5.QbMm5ListDAO.findAll(QbMm5ListDAO.java:141)
at jsp_servlet.__left._jspService(__left.java:114)
at weblogic.servlet.jsp.JspBase.service(JspBase.java:34)
at weblogic.servlet.internal.StubSecurityHelper$ServletServiceAction.run
(StubSecurityHelper.java:226)
at weblogic.servlet.internal.StubSecurityHelper.invokeServlet
(StubSecurityHelper.java:124)
at weblogic.servlet.internal.ServletStubImpl.execute
(ServletStubImpl.java:283)
at weblogic.servlet.internal.ServletStubImpl.onAddToMapException
(ServletStubImpl.java:391)
at weblogic.servlet.internal.ServletStubImpl.execute
(ServletStubImpl.java:309)
at weblogic.servlet.internal.ServletStubImpl.execute
(ServletStubImpl.java:175)
at weblogic.servlet.internal.WebAppServletContext
$ServletInvocationAction.run(WebAppServletContext.java:3370)
at weblogic.security.acl.internal.AuthenticatedSubject.doAs
(AuthenticatedSubject.java:321)
at weblogic.security.service.SecurityManager.runAs(Unknown Source)
at weblogic.servlet.internal.WebAppServletContext.securedExecute
(WebAppServletContext.java:2117)
at weblogic.servlet.internal.WebAppServletContext.execute
(WebAppServletContext.java:2023)
at weblogic.servlet.internal.ServletRequestImpl.run
(ServletRequestImpl.java:1359)
at weblogic.work.ExecuteThread.execute(ExecuteThread.java:200)
at weblogic.work.ExecuteThread.run(ExecuteThread.java:172)
```

示例 10-7

10.5.2 问题原因分析

Hibernate3.0 采用新的基于 ANTLR 的 HQL/SQL 查询翻译器，然而在 weblogic.jar 中已经包含了 antrl 类库，但是版本并不一致，因此就会产生一些类加载的错误。出现这个错误之后，antlr 会调用 System.exit()，这样就会导致 WebLogic 中止服务。

需要补充说明一下，在 Hibernate 的配置文件中，hibernate.query.factory_class 属性用来选择查询翻译器。

（1）选择 Hibernate3.0 的查询翻译器：

```
hibernate.query.factory_class=
org.hibernate.hql.ast.ASTQueryTranslatorFactory
```

（2）选择 Hibernate2.1 的查询翻译器

hibernate.query.factory_class=
org.hibernate.hql.classic.ClassicQueryTranslatorFactory

为了使用 3.0 的批量更新和删除功能，只能选择（1）否则不能解释批量更新的语句，当使用的时候出现了不支持条件输入中文的情况。选择（2）可以支持输入中文，但没法解释批量更新语句了。

10.5.3 解决方法

解决办法是在 hibernate.properties 文件中增加属性 hibernate.query.factory_class，属性的值是 org.hibernate.hql.classic.ClassicQueryTranslatorFactory. 如果用的是 cfg.xml 文件，就在 hibernate.cfg.xml 中的<session-factory>下面添加一条声明:

```
<property name="query.factory_class">
  org.hibernate.hql.classic.ClassicQueryTranslatorFactory
</property>
```

示例 10-8

也可在 WebLogic 的启动脚本中，将 antlr-2.7.5H3.jar 放在最前面。

小结：以上主要提及了 Weblogic 与 spring 和 hibernate 的兼容性，一般情况下 Weblogic 与 SSH 兼容问题在于类（包）冲突，由于 Weblogic 自带了 classloader 会首先去找到自己的 jar 文件，然后加载项目的 jar，很多项目中的 jar 均已在 Weblogic 的 jar 中被加载了，直接导致无法在项目的 war 或者 ear 中找到 jar 文件。通常的解决办法就是：在 classpath 中配置让需要的包在 webservice.jar 之前加载。

10.6 Tomcat 开源项目移植入 WebLogic 问题总结

10.6.1 JDK 和 Servlet 版本问题

WebLogic 8.1 sp4以前（包括 sp4）只支持 JDK1.4，建议使用 JDK1.4进行编译代码，有时 JDK1.5编译的程序无法运行。由于 WebLogic 8.1不支持 J2EE1.4，不要使用 Servlet2.4和 JSP2.0进行编码。

10.6.2 Include 问题

在 WebLogic 中不允许在一个文件中出现一次以上类似<%@ page contentType="text/html; charset=GBK"%>的代码，所以使用 include file 时，一般将被 include 的文件中类似代码删除。

在 TOMCAT 时允许上述代码出现多次，并且使用 include file 时，若被 include 的文件中不包含上述代码，编译后客户端显示为乱码。最早 BEA 对此解释为 TOMCAT 不符合 J2EE 规范。

为了增加代码的通用性和可移植性，建议使用<jsp:include>方式。<jsp:include>将被 include 的 jsp 代码视为独立存在的文件，所以可以在不同文件内使用多个<%@ page contentType="text/html; charset=GBK"%>。<jsp:include>直接传参由<jsp:param>标签完成，在被 include 页面可以通过 request 得到传入的值，也可以通过 request.setAttribute()、request.getAttribute()进行内外文件参数传递。

10.6.3 打包后 Log4j 支持问题

有时候应用打包成.war 部署到 WebLogic 后，出现如下问题：

```
    Error: weblogic.management.DeploymentException: Cannot set web app root
system property when WAR file is not expanded - with nested exception:
    [java.lang.IllegalStateException: Cannot set web app root system property
when WAR file is not expanded]
```

示例 10-9

问题解决：通常您不需要亲自编写 servlet 或者 listener，比如直接利用 log4j 的 com.apache.jakarta.log4j.Log4jInit 类，Spring 的 org.springframework.web.util.Log4jConfigServlet 和 org.springframework.web.util.ServletContextListener 方式配置，找到 Log4jConfigServlet 和 ServletContextListener 的源码，他们都在适当的地方(callback method)调用了 Log4jWebConfigurer.initLogging(getServletContext());定位到这个方法，第一句就是:WebUtils.setWebAppRootSystemProperty(servletContext);再定位到该方法，方法很短：

```
    public static void setWebAppRootSystemProperty(ServletContext
servletContext) throws IllegalStateException {
      String param = servletContext.getInitParameter(WEB_APP_ROOT_KEY_PARAM);
      String key = (param != null ? param : DEFAULT_WEB_APP_ROOT_KEY);
      String oldValue = System.getProperty(key);
      if (oldValue != null) {
        throw new IllegalStateException(
        "WARNING: Web app root system property already set: "
        + key + " = " + oldValue
        + " - Choose unique webAppRootKey values in your web.xml files!");
      }
      String root = servletContext.getRealPath("/");
      if (root == null) {
        throw new IllegalStateException
        ("Cannot set web app root system property when WAR file is not
expanded");
      }
      System.setProperty(key, root);
      servletContext.log("Set web app root system property: "
                         + key + " = " + root);
    }
```

示例 10-10

系统需要读取 webAppRootKey 这个参数，所以在部署到 WebLogic 里的时候，在 web.xml 中手动添加如下代码：

```
    <context-param>
       <param-name>webAppRootKey</param-name>
       <param-value>webapp.root</param-value>
    </context-param>
```

示例 10-11

WebLogic 自身也包含对 Log4j 的支持，在打包部署（.war）的时候，会和 Spring 的 org.springframework.web.util.Log4jConfigListener 有冲突,所以改用 Servlet 加载。

web.xml 中删除下面代码：

```
<listener id="log4jConfigListener">
   <listener-class>org.springframework.web.util.Log4jConfigListener
   </listener-class>
</listener>
```

示例 10-12

将 Listener 加载改为通过 Servlet 加载，再在 web.xml 增加：

```
<servlet>
   <servlet-name>log4jConfigListener</servlet-name>
   <servlet-class>org.springframework.web.util.Log4jConfigServlet
   </servlet-class>
   <load-on-startup>0</load-on-startup>
</servlet>
```

示例 10-13

10.6.4 Hibernate3、Axis 部署问题

Hibernate3中 hibernate.query.factory_class 的默认值为 org.hibernate.hql.ast.ASTQueryTranslatorFactory，在 WebLogic 下系统运行时会抛出 org.hibernate.QueryException: ClassNotFoundException: org.hibernate.hql.ast.HqlToken 异常。

这个问题讨论得很多，解决方法也各式各样，其实很简单，WebLogic 系统默认加载 EJB-QL parser，存在重名类，所以使用时会出现 ClassNotFoundException。一般的简单修改方式(包括前述)都是修改 startWebLogic 运行的脚本，将 antlr-2.7.5H3.jar 文件优先加载。但这样的方法会带来一些其他问题，所以不推荐使用。

最好的方法是，在 WEB-INF 目录下建一个 weblogic.xml 文件，文件中加入如下代码：

```
<weblogic-web-app>
  <container-descriptor>
    <prefer-web-inf-classes>true</prefer-web-inf-classes>
  </container-descriptor>
</weblogic-web-app>
```

示例 10-14

说明：prefer-web-inf-classes=true 是 WebLogic's classloader 在有重名类时，优先加载 Web 应用中 WEB-INF 内的类。

Axis 部署同理。注意：

1、如果有包在通过修改 startWebLogic 启动脚本优先加载后，web 应用中有重复的包，并且将 prefer-web-inf-classes=true，BEA WebLogic 编译 JSP 时会报错。（直接设置 true 就可以，无需再修改脚本；如果已经修改过脚本，需要还原。）

2、可以有两种方式部署 Axis 包：一种为 prefer-web-inf-classes=true，另外一种将 saaj.jar 一个包在 webservices.jar 之前优先加载。经实际验证，只将 saaj.jar 一个包优先加载并不能解决全部问题，如果不使用前一种方法，请将 axis 全部的包加载在 webservices.jar 之前。

10.6.5 Axis 远程调用.net Web Service 接口找不到方法

调用时出现如下异常：

```
    java.lang.NoSuchMethodError:
javax.xml.namespace.QName.getPrefix()Ljava/lang/String
```

示例 10-15

应用系统需要调用远程.net 平台的 Web Service 接口，该程序在 Tomcat 和 Windows 下 BEA WebLogic 8.1 SP5下进行测试，全部正常使用，但移植到 HPUX 上，每次调用接口时都会找不到 javax.xml.namespace.QName.getPrefix()方法。

查明该方法存在于 jaxrpc.jar 文件中，而 webservices.jar 存在名为 javax.xml.namespace.QName 的重名类。在 startWebLogic.sh 文件中，手动将 jaxpc.jar 排在 webservices.jar 之前加载，即可解决该问题。