**JSP 应用框架**

**什么是应用框架：**

框架（framework）是可重用的，半成品的应用程序，可以用来产生专门的定制

程序。象人一样，软件应用的相似性比不同点要多。它们运行在相似的机器上，期望从相同的设备输入信息，输出到相同的显示设备，并且将数据存储到相同的硬盘设备。开发传统桌面应用的开发人员更习惯于那些可以涵盖应用开发同一性的工具包和开发环境。构架在这些公共基础上的应用框架可以为开发人员提供可以为他们的产品提供可重用服务的基础架构。

框架向开发人员提供一系列具有以下特征的骨架组件：

（1）已经知道它们在其它程序上工作得很好；

（2）它们随时可以在下一个项目中使用；

（3）它们可以被组织的其它团队使用；

对于框架是典型的构建还是购买命题。如果你自己构建它，在你完成时你就会

理解它，但是在你被融入之前又将花费多长时间呢？如果要购买，你必须得克服学习曲线，同样，在你可以用它进行工作之前又得花多长时间？这里没有所谓正确答案，但许多观察者都会同意，象 Struts 这样的框架能提供比从头开始开发更显著的投资回报，特别是对于大型项目来说。

**其它类型的框架：**

框架的概念不仅用于应用程序也可用于组件。在其它的资料里面，我们也介绍

了一些可以和 Struts 一起使用的框架。这些包括 Lucene 搜索引擎，Scaffold 工具包，Struts 验证器，以及 Tiles 标签库。与应用框架一样，这些工具也提供了一些半完成的版本，可以用在用户的定制组件之中。某些框架被限制于专门的开发环境中。Struts以及本文中涉及的组件却不是这样。你可以在很多环境中来开发 Struts： Visual Agefor Java, JBuilder, Eclipse, Emacs, 甚至使用 Textpad。对于你的工具，如果你可以用来开发 Java, 你就可以用它来开发 Struts。

**使用的技术：**

使用 Struts 的应用开发使用了大量的其他基础技术。这些技术并不是专门针对Struts，而是所有 Java web 应用都可以使用的。开发者使用 Struts 之类的框架是为了隐藏在诸如 HTTP，CGI，以及 JSP 之类技术后面的繁琐的细节。作为一个 Struts开发者，你并不需要知晓所有的相关知识，但是这些基本技术的工作原理可能有助于你针对棘手问题设计出创造性的方案。

**超文本传输协议 (HTTP)：**

当两个国家之间进行调解时，外交官们总是遵循一定的正式协议。外交协议主

要设计来避免误解，以及防止谈判破裂。同样，当计算机间需要对话，它们也遵循一个正式的协议。这个协议定义数据是如何传输，以及它们到达后如何进行解码。Web 应用程序就是使用 HTTP 协议在运行浏览器的计算机和运行的服务器的程序间传输数据。

很多服务器应用程序使用 HTTP 之外的其他协议。他们在计算机之间维护一个持久性的连接。应用服务器可以清楚的知道是谁连接上来，而且何时中断连接。因为它们知道每一个连接的状态，以及每一个使用它的人。这称之为状态协议。

相反， HTTP 是一个无状态协议。HTTP Server 可以接受来自于各种客户的各种请求，并提供各种响应，即使是这个响应仅仅是说 No。没有大量的协商和连接持久性，无状态协议可以处理大量的请求。这也是 Internet 可以扩展到很多计算机的原因。

HTTP 成为通用标准的原因是其简单性。HTTP 请求看起来就像一个平常的文本文档。这使应用程序很容易创建 HTTP 请求。你甚至可以通过标准的程序如 Telnet来手动传递一个 HTTP 请求。当 HTTP 响应返回时，它也是一个开发者可以直接阅读的平面文本。HTTP 请求的第一行包含方法，其后是请求的来源地址和 HTTP 版本。HTTP 请求头跟在首行后面，可以没有也可以有多个。HTTP 头向服务器提供额外的信息。可以包括浏览器的种类和版本，可接受的文档类型，浏览器的 cookies等等。7 种请求方法中， GET 和 POST 是用得最多的。

一旦服务器接收到请求，它就要产生一个 HTTP 响应。响应的第一行称为状行，包含了 HTTP 协议的版本，数字型状态，以及状态的简短描述。状态行后，服务器将返回一个 HTTP 响应头，类似于 HTTP 请求头。如上所述，HTTP 并不在请求间保持状态信息。服务器接受请求，发出响应，并且继续愉快地处理文本请求。

因为简单和效率，无状态协议不适合于需要跟踪用户状态的动态应用。Cookies和 URL 重写是两个在请求间跟踪用户状态的方式。cookie 是一种特殊的信息包，存储于用户的计算机中。URL 重写是在页面地址中存储一个特殊的标记，Java 服务器可以用它来跟踪用户。这两种方法都不是无缝的，是用哪一个都意味着在开发时都要进行额外的工作。对其本身来说，标准的 HTTP web 服务器并不传输动态内容。它主要是使用请求来定位文件资源，并在响应中返回此资源。通常这里的文件使用Hypertext Markup Language (HTML) [W3C,HTML] 格式化，以使浏览器可以显示它们。HTML 页面通常包含一些到其他页面的超文本连接，也可以显示其他一些内容比如图像和视频等等。用户点击连接将产生另一个请求，就开始一个新的处理过程。标准 web 服务器处理静态内容处理得很好，但处理动态内容时则需要额外的帮助手段了。

定义静态内容直接来自于文本或数据文件，比如 HTML 或者 JPEG 文件。这些文件可以随时改变，但通过浏览器请求时，却不能自动改变。相反，动态内容是临时产生的，典型地，它是针对浏览器的个别请求的响应。

**公共网关接口(CGI)：**

第一个普遍用来产生动态内容的标准是通用网关接口（ Common Gateway

Interface (CGI)）。CGI 使用标准的操作系统特征，比如环境变量和标准输入输出，在 Web 服务器间以及和主机系统间创建桥接和网关。其他程序可以看到 web server传递过来的请求，并创建一个定制的响应。 web 服务器接收到一个对 CGI 程序的当请求时，它便运行这个程序并向其提供它请求里面所包含的信息。CGI 程序运行，并将输出返回给 Web server，web server 则将输出响应给浏览器。CGI 定义了一套关于什么信息将作为环境变量传递，以及它希望怎样使用标准输入和输出的惯例。与 HTTP 一样，CGI 是灵活和易于实现的，并且已经有大量现成的 CGI 程序。

CGI 的主要缺点是它必须为每个请求运行一个程序。这是一个相对昂贵的处理方法，对大容量站点来说，每分钟有数千个请求，有可能使站点瘫痪。CGI 程序的另一个缺点是平台依赖性，一个平台上开发的程序不一定在另一个平台上能运行。

**Java servlet：**

Sun 公司的 Java Servlet 平台直接解决了 CGI 程序的两个主要缺点：

首先，servlet 比常规 CGI 程序提供更好的性能和资源利用。其次，一次编写，随处运行的 JAVA 特性意味着 servlet 在有 JVM 的操作系统间是轻便的可移动的。Servlet 看起来好像是一个微小的 web server。它接受请求并产生响应。但，和常规web server 不同，servlet API 是专门设计来帮助 Java 开发人员创建动态应用的。

Servlet 本身是要编译成字节码的 Java 类，就像其他 Java 对象一样。Servlet 访问 HTTP 特定服务的 API，但它仍然是一个运行于程序之中的 Java 对象，并可以利用所有的 Java 资产。为了使常规 web servers 能访问 servlet，servlet 被安插在一个容器之中。Servlet 容器连接到 Web 服务器。每 servlet 都可以声明它可以处理何种样式的 URL。当符合所注册样式的请求到达，web server 将请求传递给容器，容器则调用响应的 servlet。但和 CGI 程序不同，并不是针对每个请求都要创建一个新的 servlet。一旦容器实例化了一个 servlet，它就仅为每个新的请求创建一个新的线程。Java 线程可比使用 CGI 程序的服务器处理开销小多了。

一旦 servlet 被创建，使用它处理额外的请求仅带来很小的额外开销。Servlet开发人员可以使用 init() 方法保持对昂贵资源的引用，比如到数据库或者 EJB Home接口的连接,以便它们可以在不同的请求之间进行共享。获得这些资源要耗费数秒时间，这比大多数冲浪者愿意等的时间要长些。

Servlet 的另一个好处是，它是多线程的，servlet 开发人员必须特别注意确保它们的 servlet 是线程安全的。学习 servlet 编程，我们推荐 Java Servlets by Example, 作者 Alan R. Williamson[Williamson]。

**JavaServer Pages：**

虽然 servlets 对 CGI 程序来说前进了一大步，但它也不是万能灵药。为了产生响应，开发人员不得不使用大量的 println 语句来生成 HTML。比如这样的代码：

out.println("<P>One line of HTML.</P>");

out.println("<P>Another line of HTML.</P>");

在产生 HTTP 响应的 Servlet 中是很普遍的。也有一些库有助于你产生 HTML。随着应用越来越复杂，Java 开发人员将不再扮演 HTML 页面设计的角色。同时，大多数项目经理更喜欢将团队分成不同的小组。 它们喜欢 HTML 设计人员处理表现层的工作，而 Java 工程师则专注于业务逻辑。单独使用 servlet 的做法鼓励混合标记和业务逻辑，很难区分团队人员的专业工作。

为解决这个问题，Sun 提出了一个将脚本和模板技术结合到一个组件中的服务器页面技术（JavaServer Pages）。为创建 JSP 页面, 开发者按创建 HTML 页面类似的方式创建页面，使用相同的 HTML 语法。为将动态内容引入页面，开发人员可以将脚本元素置入页面之中。脚本元素是一些标记，封装了可以被 JSP 识别的逻辑。你可以在 JSP 页面中很容易的识别出脚本元素，他们被封装在一对<% 和 %>标记中。

为了识别 JSP 页面，文件需要保存为扩展名.jsp。当一个客户请求 JSP 页面时，容器将页面翻译成 Java servlet 源代码文件，并将它编译成 Java 类文件——就象你写的 servlet 文件一样。在运行时，容器也能检测 JSP 文件和相应的类的最后更新时间。如果，JSP 文件自上次编译以来被修改了，容器将重新翻译和编译 JSP 文件。项目经理现在可以将表现层分派给 HTML 开发人员，将业务逻辑工作分派给 JAVA开发人员。重要的是记住，JSP 页面事实上是一个 servlet。你可以在 servlet 做的，也可以在 JSP 中做。

**JavaBean：**

JavaBean 是一种 Java 类，它遵从一定的设计模式，使它们易于和其他开发工

具和组件一起使用。

定义 JavaBean 是一种 JAVA 语言写成的可重用组件。要编写 JavaBean，类必须是具体类和公共类，并且具有无参数的构造器（NON-ARGS CONSTRUCTOR）。JavaBean 通过提供符合一致性设计模式的公共访问方法将内部字段暴露称为属性。众所周知，属性名称也符合这种模式，其他 JAVA 类可以通过自省机制发现和操作这些 JavaBean 属性。

**我们必须做的如下：**

（1）编写一个类，通过实现doStart()或 者doEnd()方法来实现javax.servlet.jsp.tagext.TagSupport或者 javax.servlet.jsp.tagext.BodyTagSupport 接口。这些方法获得一个 JspWriter 对象，你可以用它来输出你需要的 HTML 内容。

（2）创建一个标签库描述文件(TLD)来将你的新建的类，映射到一个标签名称。

（3）在你的 Web 应用描述符(web.xml)中定义你的<taglib> 元素。通过在 JSP 页面的顶部放置下面的语句：<%@tagliburi="/tags/app.tld prefix="app" %>来告诉 JSP页面你将使用你自己的标签库。

（4）这个语句导入将在本页中使用的标签库，并分配给它一个前缀。关于更多细节，请参考 JSP 标签库技术页面。

JavaBean 设计模式提供两种类型的方式来访问 bean 的内部状态：访问器（accessor） 用来读 JavaBean 的状态，修改器（mutator ）用来改变 JavaBean 的状态。

Mutator 通常以小写的 set 前缀开始，后跟属性名。属性名的第一个字母必须大写。返回值通常是 void，因为 mutator 仅仅改变属性的值，而不返回它们。简单属性的 mutator 在其方法体中可能只有一个参数，该参数可以是各种类型。Mutator也可根据其前缀称为设置器 setters 。例如， Double 类型的属性 weight 的 mutator对方法体可能是：

public void setWeight(Double weight)

相似的设计模式也用于访问器方法的创建。Accessor 通常以小写的 get 为前缀，后跟属性名。属性名的第一个字母必须大写。返回值必须匹配相应的修改器方法的参数。简单属性的 Accessor 在其方法体中不能接受参数。同样，访问器 accessor 也经常称为获取器 getter。

属性 weight 的访问器方法体可能是：public Double getWeight()

如果访问器返回一个逻辑值，这种情况下有个变体模式。不使用小写的 get，逻辑属性的访问器可以使用小写的 is 前缀，后跟属性名。属性名的首字母必须大写。返回值肯定是逻辑值，不管是 boolean 还是 Boolean。逻辑访问器在其方法体中不能接受参数。On 属性的逻辑访问器的方法体可能是：public boolean isOn()在使用 JavaBean 时，规范的方法体签名扮演了极为重要的角色。其他组件可以使用 Java 的反射 API 通过查找前缀为 set、 或者 get 的方法来发现 JavaBean 的is属性。如果一个组件在一个 JavaBean 中发现一个这样的方法，它就知道这个方法可以用来访问或者改变 JavaBean 的属性。 引入 JavaBean 是为了用于 GUI 组件，Sun但它们已经用在 Java 开发的各个方面，包括 Web 应用。Sun 的工程师在开发 JSP标签的扩展类时，也被设计来可以和 JavaBean 一起工作。一个页面的动态数据可以使用一个 JavaBean 来传递，并且 JSP 标记可以随后使用 bean 的属性来定制页面的输出。

**Model 2：**

Servlet/JSP 规范的 0.92 版描述了在一个应用中使用 servlet 和 JSP 的架构。在其后的规范中，Model 2 这个叫法消失了，但它已经在 Java web 开发人员中非常通用了。根据 Model 2，servlet 处理数据存取和导航流，JSP 处理表现。Model 2 使 Java工程师和 HTML 设计者分别工作于它们所擅长和负责的部分。Model 2 应用的一部分发生改变并不强求其他部分也跟着发生改变。HTML 开发人员可以改变程序的外观和感觉，并不需要改变后端 servlet 的工作方式。Struts 框架是基于 Model 2 的架构。它提供一个控制器 servlet 来处理导航流和一些特殊类来帮助数据访问。随框架也提供一个丰富的标签库，以使 Struts 易于和 JSP 一起使用。

**小结：**

在本文中，我们介绍了 Struts 应用框架。并介绍了一些基本知识，关于 HTTP,CGI, Java servlet, JSP,以及 JavaBean。我们也说明了 Model 2 应用架构，以及它如何用来将 servlet 和 JSP 在结合在同一个应用之中。现在你已经有关于 Strtus Web 应用程序模样的初步印象，以后我们将更深入的讨论 Struts 框架的理论和具体实践