JSP程序设计教程

第9章 JSP高级程序设计

第 9章 JSP高级程序设计

- 9.1 JSP与Ajax技术 √
- 9.2 EL表达式及标签 √
- 9.3 JSP框架技术 √

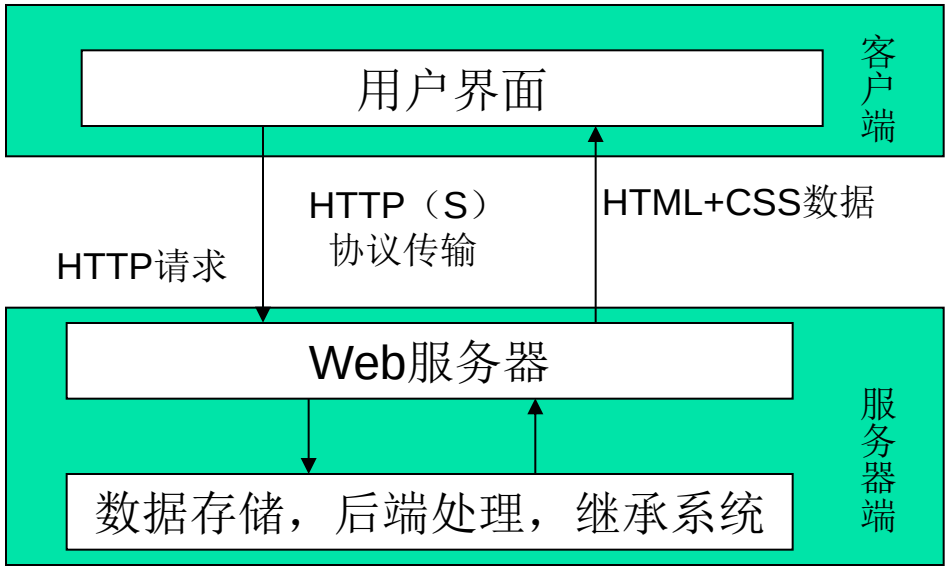
课件制作人：王国辉

9.1 JSP与Ajax技术

Ajax是Asynchronous JavaScript and XML的缩写，意思是异步的JavaScript与XML。Ajax并不是一门新的语言或技术，它是JavaScript、XML、CSS、DOM等多种已有技术的组合，它可以实现客户端的异步请求操作。这样可以实现在不需要刷新页面的情况下与服务器进行通信，从而减少了用户的等待时间。

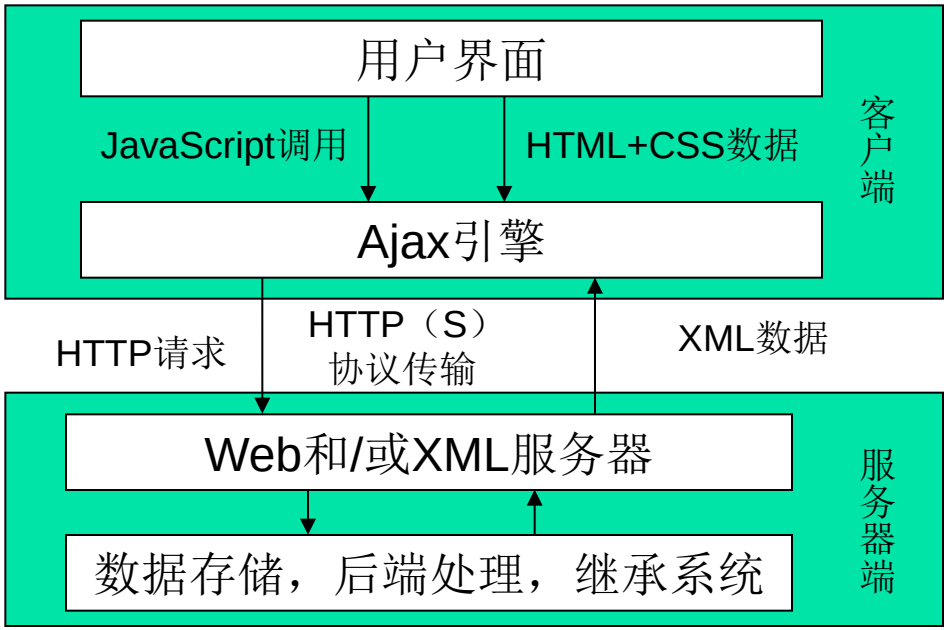
9.1.1 Ajax的开发模式

在传统的Web应用模式中，页面中用户的每一次操作都将触发一次返回Web服务器的HTTP请求，服务器进行相应的处理（获得数据、运行与不同的系统会话）后，返回一个HTML页面给客户端，如下图所示。



9.1.1 Ajax的开发模式

而在Ajax应用中，页面中用户的操作将通过Ajax引擎与服务器端进行通信，然后将返回结果提交给客户端页面的Ajax引擎，再由Ajax引擎来决定将这些数据插入到页面的指定位置，如下图所示。



9.1.1 Ajax的开发模式

从上面两个图中可以看出，对于每个用户的行为，在传统的Web应用模式中，将生成一次HTTP请求，而在Ajax应用开发模式中，将变成对Ajax引擎的一次JavaScript调用。在Ajax应用开发模式中通过JavaScript实现在不刷新整个页面的情况下，对部分数据进行更新，从而降低了网络流量，给用户带来了更好的体验。

9.1.2 Ajax使用的技术

1. JavaScript脚本语言

JavaScript是一种在Web页面中添加动态脚本代码的解释性程序语言，其核心已经嵌入到目前主流的Web浏览器中。虽然平时应用最多的是通过JavaScript实现一些网页特效及表单数据验证等功能，其实JavaScript可以实现的功能远不止这些。JavaScript是一种具有丰富的面向对象特性的程序设计语言，利用它能执行许多复杂的任务，例如，Ajax就是利用JavaScript将DOM、XHTML（或HTML）、XML以及CSS等技术综合起来，并控制它们的行为。因此要开发一个复杂高效的Ajax应用程序，就必须对JavaScript有深入的了解。

9.1.2 Ajax使用的技术

2．XMLHttpRequest

Ajax技术之中，最核心的技术就是XMLHttpRequest，它是一个具有应用程序接口的JavaScript对象，能够使用超文本传输协议（HTTP）连接一个服务器，是微软公司为了满足开发者的需要，于1999年在IE 5.0浏览器中率先推出的。现在许多浏览器都对其提供了支持，不过实现方式与IE有所不同。

通过XMLHttpRequest对象，Ajax可以像桌面应用程序一样只同服务器进行数据层面的交换，而不用每次都刷新页面，也不用每次都将数据处理的工作交给服务器来做，这样既减轻了服务器负担又加快了响应速度、缩短了用户等待的时间。

人民邮电出版社
POSTS & TELECOM PRESS

9.1.2 Ajax使用的技术

在使用XMLHttpRequest对象发送请求和处理响应之前，首先需要初始化该对象，由于XMLHttpRequest不是一个W3C标准，所以对于不同的浏览器，初始化的方法也是不同的。

（1）IE浏览器

IE浏览器把XMLHttpRequest实例化为一个ActiveX对象。具体方法如下：

```
var http_request = new ActiveXObject("Msxml2.XMLHTTP");
或者
var http_request = new ActiveXObject("Microsoft.XMLHTTP");
```

上面语法中的Msxml2.XMLHTTP和Microsoft.XMLHTTP是针对IE浏览器的不同版本而进行设置的，目前比较常用的是这两种。

9.1.2 Ajax使用的技术

（2）Mozilla、Safari等其他浏览器

Mozilla、Safari等其他浏览器把它实例化为一个本地JavaScript对象。具体方法如下：

```
var http_request = new XMLHttpRequest();
```

为了提高程序的兼容性，可以创建一个跨浏览器的XMLHttpRequest对象。创建一个跨浏览器的XMLHttpRequest对象其实很简单，只需要判断一下不同浏览器的实现方式，如果浏览器提供了XMLHttpRequest类，则直接创建一个实例，否则使用IE的ActiveX控件。具体代码如下：

9.1.2 Ajax使用的技术

```
if (window.XMLHttpRequest) { // Mozilla、Safari...
        http_request = new XMLHttpRequest();
} else if (window.ActiveXObject) { // IE浏览器
        try {
                http_request = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
                try {
                        http_request = new
ActiveXObject("Microsoft.XMLHTTP");
          } catch (e) {}
        }
}
```

说明：由于JavaScript具有动态类型特性，而且XMLHttpRequest对象在不同浏览器上的实例是兼容的，所以可以用同样的方式访问XMLHttpRequest实例的属性的方法，不需要考虑创建该实例的方法是什么。

9.1.2 Ajax使用的技术

下面对XMLHttpRequest对象的常用方法进行详细介绍。

（1）open()方法

open()方法用于设置进行异步请求目标的URL、请求方法以及其他参数信息，具体语法如下：

```
open("method","URL"[,asyncFlag[,"userName"[, "password"]]]);
```

在上面的语法中，method用于指定请求的类型，一般为get或post；URL用于指定请求地址，可以使用绝对地址或者相对地址，并且可以传递查询字符串；asyncFlag为可选参数，用于指定请求方式，同步请求为true，异步请求为false，默认情况下为true；userName为可选参数，用于指定求用户名，没有时可省略；password为可选参数，用于指定请求密码，没有时可省略。

9.1.2 Ajax使用的技术

（2）send()方法

send()方法用于向服务器发送请求。如果请求声明为异步，该方法将立即返回，否则将等到接收到响应为止。具体语法格式如下：

```
send(content);
```

在上面的语法中，content用于指定发送的数据，可以是DOM对象的实例、输入流或字符串。如果没有参数需要传递可以设置为null。

（3）setRequestHeader()方法

setRequestHeader()方法为请求的HTTP头设置值。具体语法格式如下：

9.1.2 Ajax使用的技术

```
setRequestHeader("label", "value");
```

在上面的语法中，label用于指定HTTP头；value用于为指定的HTTP头设置值。

注意：setRequestHeader()方法必须在调用open()方法之后才能调用。

（4）abort()方法

abort()方法用于停止当前异步请求。

（5）getAllResponseHeaders()方法

getAllResponseHeaders()方法用于以字符串形式返回完整的HTTP头信息，当存在参数时，表示以字符串形式返回由该参数指定的HTTP头信息。

9.1.2 Ajax使用的技术

XMLHttpRequest对象的常用属性如下表所示。

属　性	说　明
onreadystatechange	每个状态改变时都会触发这个事件处理器，通常会调用一个JavaScript函数
readyState	请求的状态。有以下5个取值： 0 = "未初始化" 1 = "正在加载" 2 = "已加载" 3 = "交互中" 4 = "完成"
responseText	服务器的响应，表示为字符串
responseXML	服务器的响应，表示为XML，这个对象可以解析为一个DOM对象
status	返回服务器的HTTP状态码，如： 200 = "成功" 202 = "请求被接受，但尚未成功" 400 = "错误的请求" 404 = "文件未找到" 500 = "内部服务器错误"
statusText	返回HTTP状态码对应的文本

9.1.2 Ajax使用的技术

3．XML语言

XML是Extensible Markup Language（可扩展的标记语言）的缩写，它提供了用于描述结构化数据的格式。XMLHttpRequest对象与服务器交换的数据，通常采用XML格式，但也可以是基于文本的其他格式。

4．DOM

9.1.2 Ajax使用的技术

DOM是Document Object Model（文档对象模型）的缩写，是表示文档（如HTML文档）和访问、操作构成文档的各种元素（如HTML标记和文本串）的应用程序接口（API）。W3C定义了标准的文档对象模型，它以树形结构表示HTML和XML文档，定义了遍历树和添加、修改、查找树的节点的方法和属性。在Ajax应用中，通过JavaScript操作DOM，可以达到在不刷新页面的情况下实时修改用户界面的目的。

9.1.2 Ajax使用的技术

5. CSS

CSS是Cascading Style Sheet（层叠样式表）的缩写，用于（增强）控制网页样式并允许将样式信息与网页内容分离的一种标记性语言。在Ajax出现以前，CSS已经广泛地应用到传统的网页中了。在Ajax中，通常使用CSS进行页面布局，并通过改变文档对象的CSS属性控制页面的外观和行为。

9.1.3 搭建Ajax开发框架

应用Ajax技术开发Web应用程序时，首先需要搭建Ajax开发框架。下面将通过具体实现介绍如何在JSP中搭建Ajax框架。

【例9-1】 通过Ajax实现不刷新页面，从另一个页面请求信息

9.1.4 Ajax开发需要注意的几个问题

1. 浏览器兼容性问题

Ajax使用了大量的JavaScript和Ajax引擎，而这些内容需要浏览器提供足够的支持。目前提供这些支持的浏览器有IE 5.0及以上版本、Mozilla 1.0、NetScape 7及以上版本。Mozilla虽然也支持Ajax，但是提供XMLHttpRequest对象的方式不一样。所以使用Ajax的程序必须测试针对各个浏览器的兼容性。

9.1.4 Ajax开发需要注意的几个问题

2．XMLHttpRequest对象封装

Ajax技术的实现主要依赖于XMLHttpRequest对象，但是在调用其进行异步数据传输时，由于XMLHttpRequest对象的实例在处理事件完成后就会被销毁，所以如果不对该对象进行封装处理，在下次需要调用它时就得重新构建，而且每次调用都需要写一大段的代码，使用起来很不方便。不过，现在很多开源的Ajax框架都提供了对XMLHttpRequest对象的封装方案，其详细内容这里不做介绍，请参考相关资料。

9.1.4 Ajax开发需要注意的几个问题

3. 性能问题

由于Ajax将大量的计算从服务器移到了客户端，这就意味着浏览器将承受更大的负担，而不再是只负责简单的文档显示。由于Ajax的核心语言是JavaScript，而JavaScript并不以高性能知名。另外，JavaScript对象也不是轻量级的，特别是DOM元素耗费了大量的内存。因此，如何提高JavaScript代码的性能对于Ajax开发者来说尤为重要。下面是3种优化Ajax应用执行速度的方法：

（1）优化for循环；

（2）将DOM节点附加到文档上；

（3）尽量减少点“.”号操作符的使用。

9.1.4 Ajax开发需要注意的几个问题

4．中文编码问题

Ajax不支持多种字符集，它默认的字符集是UTF-8，所以在应用Ajax技术的程序中应及时进行编码转换，否则对于程序中出现的中文字符将变成乱码。一般情况下，有以下两种情况可以产生中文乱码。

（1）发送路径的参数中包括中文，在服务器端接收参数值时产生乱码。

将数据提交到服务器有两种方法，一种是使用GET方法提交；另一种是使用POST方法提交。使用不同的方法提交数据，在服务器端接收参数时解决中文乱码的方法是不同的。具体解决方法如下。

9.1.4 Ajax开发需要注意的几个问题

① 当接收使用GET方法提交的数据时，要将编码转换为GB2312，关键代码如下：

```
String name=request.getParameter("name");
out.println("姓名"+new String(name.getBytes("iso-8859-1"),"gb2312"));//解决中文乱码
```

② 由于应用POST方法提交数据时，默认的字符编码是UTF-8，所以当接收使用POST方法提交的数据时，要将编码转换为utf-8，关键代码如下：

```
String name=request.getParameter("name");
out.println("姓名"+new String(name.getBytes("iso-8859-1"), "utf-8"));//解决中文乱码
```

9.1.4 Ajax开发需要注意的几个问题

（2）返回到responseText或responseXML的值中包含中文时产生乱码。

由于Ajax在接收responseText或responseXML的值时是按照UTF-8的编码格式进行解码的，所以如果服务器端传递的数据不是UTF-8格式，在接收responseText或responseXML的值时，就可能产生乱码。解决的办法是保证从服务器端传递的数据采用UTF-8的编码格式。

9.2 EL表达式及标签

9.2.1 表达式语言 √

9.2.2 JSTL标准标签库 √

9.2.3 自定义标签库的开发 √

课件制作人：王国辉

9.2.1 表达式语言

表达式语言简称为EL（Expression Language），下面称为EL表达式，它是JSP2.0中引入的一种计算和输出Java对象的简单语言。EL为不熟悉Java语言的页面开发人员提供了一个开发JSP应用程序的新途径。EL表达式具有以下特点：

（1）在EL表达式中可以获得命名空间（PageContext对象，它是页面中所有其他内置对象的最大范围的集成对象，通过它可以访问其他内置对象）；

（2）表达式可以访问一般变量，还可以访问JavaBean类中的属性以及嵌套属性和集合对象；

（3）在EL表达式中可以执行关系、逻辑和算术等运算；

9.2.1 表达式语言

（4）扩展函数可以与Java类的静态方法进行映射；

（5）在表达式中可以访问JSP的作用域（request，session，application以及page）。

1．EL表达式的简单使用

在JSP2.0之前，程序员只能使用下面的代码访问系统作用域的值：

```
<%=session.getAttribute("name")%>
```

或者使用下面的代码调用JavaBean中的属性值或方法：

```
<jsp:useBean id="dao" scope="page" class="com.UserInfoDao"></jsp:useBean>
<%dao.name;%>                <!--调用UserInfoDao类中name属性-->
<%dao.getName();%>           <!--调用UserInfoDao类中getName()方法-->
```

9.2.1 表达式语言

在EL表达式中允许程序员使用简单语法访问对象。例如，使用下面的代码访问系统作用域的值：

```
${name}
```

其中${name}为访问name变量的表达式，而通过表达式语言调用JavaBean中的属性值或方法的代码如下：

```
<jsp:useBean id="dao" scope="page" class="com.UserInfoDao"></jsp:useBean>
${dao.name}                    <!--调用UserInfoDao类中name属性-->
${dao.getName()}               <!--调用UserInfoDao类中getName()方法-->
```

2．EL表达式的语法

EL表达式语法很简单，它最大的特点就是使用很方便。表达式语法格式如下：

```
${expression}
```

9.2.1 表达式语言

在上面的语法中，“${”符号是表达式起始点，因此，如果在JSP网页中要显示“${”字符串，必须在前面加上“\”符号，即“\${”，或者写成“${'${'}”，也就是用表达式来输出“${”符号。在表达式中要输出一个字符串，可以将此字符串放在一对单引号或双引号内。例如，要在页面中输出字符串“长亭外，古道边”，可以使用下面的代码：

```
${"长亭外，古道边"}
```

技巧：如果想在JSP页面中输出EL表达式，可以使用“\”符号，即在“${}”之间加“\”，例如“\${5+3}”，将在JSP页面中输出“${5+3}”，而不是5+3的结果8。

9.2.1 表达式语言

说明：由于在EL表达式是JSP2.0以前没有的，所以为了和以前的规范兼容，可以通过在页面的前面加入以下语句声明是否忽略EL表达式：

```
<%@ page isELIgnored="true|false" %>
```

在上面的语法中，如果为true，则忽略页面中的EL表达式，否则为false，则解析页面中的EL表达式。

3．EL表达式的运算符

在JSP中，EL表达式提供了存取数据运算符、算术运算符、关系运算符、逻辑运算符、条件运算符及Empty运算符，下面进行详细介绍。

9.2.1 表达式语言

（1）存取数据运算符

在EL表达式中可以使用运算符“[]”和“.”来取得对象的属性。例如，${user.name}或者${user[name]}都是表示取出对象user中的name属性值。

（2）算术运算符

算术运算符可以作用在整数和浮点数上。EL表达式的算术运算符包括加（+）、减（-）、乘（*）、除（/或div）、和求余（%或mod）等5个。

注意：EL表达式无法像Java一样将两个字符串用“+”运算符连接在一起（"a"+"b"），所以${"a"+"b"}的写法是错误的。但是，可以采用${"a"}${"b"}这样的方法来表示。

9.2.1 表达式语言

（3）关系运算符

关系运算符除了可以作用在整数和浮点数之外，还可以依据字母的顺序比较两个字符串的大小，这方面在Java中没有体现出来。EL表达式的关系运算符包括等于（==或eq）、不等于（!=或ne）、小于（<或lt）、大于（>或gt）、小于等于（<=或le）和大于等于（>=或ge）等6个。

注意：在使用EL表达式关系运算符时，不能够写成：

${param.password1} == ${param.password2}

或

${${param.password1} == ${param.password2}}

而应写成：

${param.password1 == param.password2}

9.2.1 表达式语言

（4）逻辑运算符

逻辑运算符可以作用在布尔值（Boolean），EL表达式的逻辑运算符包括与（&&或and）、或（||或or）和非（!或not）等3个。

（5）empty运算符

empty运算符是一个前缀（prefix）运算符，即empty运算符位于操作数前方，被用来决定一个对象或变量是否为null或空。

（6）条件运算符

EL表达式中可以利用条件运算符进行条件求值，其格式如下：

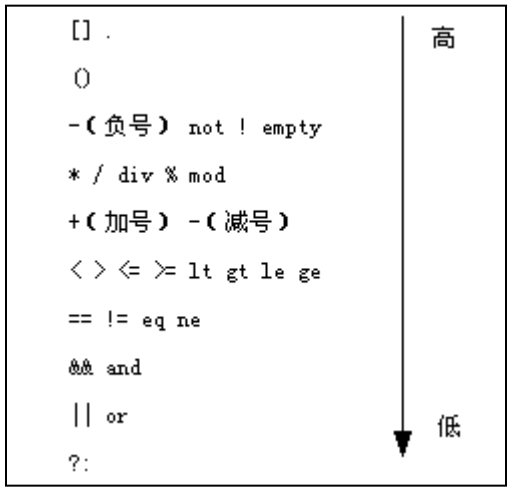
9.2.1 表达式语言

${条件表达式？计算表达式1：计算表达式2}

在上面的语法中，如果条件表达式为真，则计算表达式1，否则计算表达式2。但是EL表达式中的条件运算符功能比较弱，一般可以用JSTL（JSTL是一个不断完善的开放源代码的JSP标准标签库，主要给Java Web开发人员提供一个标准的通用的标签库，关于JSTL的详细介绍参见9.2.2节）中的条件标签<c:if>或<c:choose>替代，如果处理的问题比较简单也可以使用。EL表达式中的条件运算符唯一的优点是在于其非常简单和方便，和Java语言里的用法完全一致。

上面所介绍的各运算符的优先级如图9-4所示。

9.2.1 表达式语言



4．EL表达式中的隐含对象

为了能够获得Web应用程序中的相关数据，EL表达式中定义了一些隐含对象。这些隐含对象共有11个，分为以下3种。

9.2.1 表达式语言

（1）PageContext隐含对象

PageContext隐含对象可以用于访问JSP内置对象，例如，request、response、out、session、config、servletContext等，如${PageContext.session}。

（2）访问环境信息的隐含对象

EL表达式中定义的用于访问环境信息的隐含对象包括以下6个：

cookie：用于把请求中的参数名和单个值进行映射；

initParam：把上下文的初始参数和单一的值进行映射；

header：把请求中的header名字和单值映射；

param：把请求中的参数名和单个值进行映射；

headerValues：把请求中的header名字和一个Arrar值进行映射；

paramValues：把请求中的参数名和一个Array值进行映射。

9.2.1 表达式语言

（3）访问作用域范围的隐含对象

EL表达式中定义的用于访问环境信息的隐含对象包括以下4个：

applicationScope：映射application范围内的属性值；

sessionScope：映射session范围内的属性值；

requestScope：映射request范围内的属性值；

pageScope：映射page范围内的属性值。

5．EL表达式中的保留字

EL表达式中定义了如下表所示的保留字，当在为变量命名时，应该避免使用这些保留字。

9.2.1 表达式语言

EL表达式中的保留字

and	eq	gt	true	instanceof	div	or	ne
le	false	lt	empty	mod	not	ge	null

9.2.2 JSTL标准标签库

JSTL的全称是JavaServer Pages Standard Tag Library，是由Apache的Jakarta小组负责维护的，它是一个不断完善的开放源代码的JSP标准标签库，主要给Java Web开发人员提供一个标准的通用的标签库。通过JSTL，可以取代传统JSP程序中嵌入Java代码的做法，大大提高程序的可维护性。JSTL主要包括以下5种标签库。

（1）核心标签库

核心标签库主要用于完成JSP页面的基本功能，包含JSTL的表达式标签、条件标签、循环标签和URL操作共4种标签。

9.2.2 JSTL标准标签库

（2）格式标签库

格式标签库提供了一个简单的标记集合国际化（I18N）标记，用于处理和解决国际化相关的问题，另外，格式标签库中还包含用于格式化数字和日期的显示格式的标签。

（3）SQL标签

SQL标签封装了数据库访问的通用逻辑，使用SQL标签，可以简化对数据库的访问。如果结合核心标签库，可以方便地获取结果集、迭代输出结果集中的数据结果。

（4）XML标签库

XML标签库可以处理和生成XML的标记，使用这些标记可以很方便地开发基于XML的Web应用。

9.2.2 JSTL标准标签库

（5）函数标签库

函数标签库提供了一系列字符串操作函数，用于分解和连接字符串、返回子串、确定字符串是否包含特定的子串等。在使用这些标签之前必须在JSP页面的首行使用<%@ taglib%>指令定义标签库的位置和访问前缀。例如，使用核心标签库的taglib指令格式如下：

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

使用格式标签库的taglib指令格式如下：

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt"%>
```

9.2.2 JSTL标准标签库

使用SQL标签库的taglib指令格式如下：

```
<%@ taglib prefix="sql" uri="http://java.sun.com/jsp/jstl/sql"%>
```

使用XML标签库的taglib指令格式如下：

```
<%@ taglib prefix="xml" uri="http://java.sun.com/jsp/jstl/xml"%>
```

使用函数标签库的taglib指令格式如下：

```
<%@ taglib prefix="fn" uri="http://java.sun.com/jsp/jstl/functions"%>
```

下面将对JSTL中最常用的核心标签库的4种标签进行介绍。

9.2.2 JSTL标准标签库

1. 表达式标签

表达式标签包括<c:out>、<c:set>、<c:remove>、<c:catch>等4个标签，下面分别介绍它们的语法及应用。

（1）<c:out>标签

<c:out>标签用于将计算的结果输出到JSP页面中，该标签可以替代<%=%>。<c:out>标签的语法格式如下：

语法1：

```
<c:out value="value" [escapeXml="true|false"] [default="defaultValue"]/>
```

语法2：

```
<c:out value="value" [escapeXml="true|false"]>
        defalultValue
</c:out>
```

9.2.2 JSTL标准标签库

这两种语法格式的输出结果完全相同，它的属性说明如下表所示。

属　性	类　型	描　述	引用EL
value	Object	将要输出的变量或表达式	可以
escapeXml	boolean	转换特殊字符，默认值为true。例如“<”转换为“<”	不可以
default	Object	如果value属性值等于NULL，则显示default属性定义的默认值	不可以

【例9-2】 <c:out>标签示例

测试<c:out>标签的escapeXml属性及通过两种语法格式设置default属性时的显示结果。

9.2.2 JSTL标准标签库

（2）<c:set>标签

<c:set>标签用于定义和存储变量，它可以定义变量是在JSP会话范围内还是JavaBean的属性中，可以使用该标签在页面中定义变量，而不用在JSP页面中嵌入打乱HTML排版的Java代码。<c:set>标签有4种语法格式。

语法1：该语法格式在scope指定的范围内将变量值存储到变量中。

```
<c:set value="value" var="name" [scope="page|request|session|application"]/>
```

语法2：该语法格式在scope指定的范围内将标签主体存储到变量中。

```
<c:set var="name" [scope="page|request|session|application"]>
    标签主体
</c:set>
```

9.2.2 JSTL标准标签库

语法3：该语法格式将变量值存储在target属性指定的目标对象的propName属性中。

```
<c:set value="value" target="object" property="propName"/>
```

语法4：该语法格式将标签主体存储到target属性指定的目标对象的propName属性中。

```
<c:set target="object" property="propName">
      标签主体
</c:set>
```

以上语法格式所涉及的属性说明如下表所示。

9.2.2 JSTL标准标签库

属　性	类　型	描　述	引用EL
value	Object	将要存储的变量值	可以
var	String	存储变量值的变量名称	不可以
target	Object	存储变量值或者标签主体的目标对象，可以是JavaBean或Map集合对象	可以
property	String	指定目标对象存储数据的属性名	可以
scope	String	指定变量存在于JSP的范围，默认值是page	不可以

【例9-3】 <c:set>标签示例

应用<c:set>标签定义不同范围内的变量，并通过EL进行输出。

9.2.2 JSTL标准标签库

（3）<c:remove>标签

<c:remove>标签可以从指定的JSP范围中移除指定的变量，其语法格式如下：

```
<c:remove var="name" [scope="page|request|session|application"]/>
```

在上面语法中，var用于指定存储变量值的变量名称；scope用于指定变量存在于JSP的范围，可选值有page、request、session、application。默认值是page。

【例9-4】<c:remove>标签示例

应用<c:set>标签定义一个page范围内的变量，然后应用通过EL输出该变量，再应用<c:remove>标签移除该变量，最后再应用EL输出该变量

9.2.2 JSTL标准标签库

（4）<c:catch>标签

<c:catch>标签是JSTL中处理程序异常的标签，它还能够将异常信息保存在变量中。<c:catch>标签的语法格式如下：

```
<c:catch [var="name"]>
……存在异常的代码
</c:catch>
```

在上面的语法中，var属性可以指定存储异常信息的变量。这是一个可选项，如果不需要保存异常信息，可以省略该属性。

2．条件标签

条件标签在程序中会根据不同的条件去执行不同的代码来产生不同的运行结果，使用条件标签可以处理程序中的任何可能发生的事情。

9.2.2 JSTL标准标签库

在JSTL中，条件标签包括<c:if>标签、<c:choose>标签、<c:when>标签和<c:otherwise>标签等4种，下面将详细介绍这些标签的语法及应用。

（1）<c:if>标签

这个标签可以根据不同的条件去处理不同的业务，也就是执行不同的程序代码。它和Java基础中if语句的功能一样。<c:if>标签有两种语法格式。

语法1：该语法格式会判断条件表达式，并将条件的判断结果保存在var属性指定的变量中，而这个变量存在于scope属性所指定范围中。

```
<c:if test="condition" var="name" [scope=page|request|session|application]/>
```

9.2.2 JSTL标准标签库

语法2：该语法格式不但可以将test属性的判断结果保存在指定范围的变量中，还可以根据条件的判断结果去执行标签主体。标签主体可以是JSP页面能够使用的任何元素，例如HTML标记、Java代码或者嵌入其他JSP标签。

```
<c:if test="condition" var="name" [scope=page|request|session|application]>
        标签主体
</c:if>
```

以上语法格式所涉及的属性说明如下表所示。

9.2.2 JSTL标准标签库

属　性	类　型	描　述	引用EL
test	Boolean	条件表达式，这是<c:if>标签必须定义的属性	可以
var	String	指定变量名，这个属性会指定test属性的判断结果将存放在那个变量中，如果该变量不存在就创建它	不可以
scope	String	存储范围，该属性用于指定var属性所指定的变量的存在范围	不可以

【例9-5】 <c:if>标签示例

应用<c:if>标签判断用户名是否为空，如果为空则显示一个用于输入用户名的文本框及“提交”按钮。

9.2.2 JSTL标准标签库

（2）<c:choose>标签

<c:choose>标签可以根据不同的条件去完成指定的业务逻辑，如果没有符合的条件会执行默认条件的业务逻辑。<c:choose>标签只能作为<c:when>和<c:otherwise>标签的父标签，可以在它之内嵌套这两个标签完成条件选择逻辑。<c:choose>标签的语法格式如下：

```
<c:choose>
        <c:when>
                业务逻辑
        </c:when>
        …       <!--多个<c:when>标签-->
        <c:otherwise>
                业务逻辑
        </c:otherwise>
</c:choose>
```

9.2.2 JSTL标准标签库

<c:choose>标签中可以包含多个<c:when>标签来处理不同条件的业务逻辑，但是只能有一个<c:otherwise>标签来处理默认条件的业务逻辑。

<c:choose>标签中可以包含多个<c:when>标签来处理不同条件的业务逻辑，但是只能有一个<c:otherwise>标签来处理默认条件的业务逻辑。 <c:when>标签的语法格式如下：

```
<c:when test="condition">
    标签主体
</c:when>
```

在该语法中，test属性用于指定条件表达式，该属性为<c:when>标签的必选 属性，可以引用EL表达式。

（4）<c:otherwise>标签

<c:otherwise>标签也是一个包含在<c:choose>标签的子标签，用于定义<c:choose>标签中的默认条件处理逻辑，如果没有任何一个结果满足<c:when>标签指定的条件，

9.2.2 JSTL标准标签库

将会执行这个标签主体中定义的逻辑代码。在<c:choose>标签范围内只能存在一个该标签的定义。<c:otherwise>标签的语法格式如下：

```
<c:otherwise>
标签主体
</c:otherwise>
```

注意：<c:otherwise>标签必须定义在所有<c:when>标签的后面，也就是说它是<c:choose>标签的最后一个子标签。

【例9-6】<c:otherwise>标签示例

应用<c:choose>标签、<c:when>标签和<c:otherwise>标签根据当前时间显示不同的问候。

9.2.2 JSTL标准标签库

3．循环标签

JSP页面开发经常需要使用循环标签生成大量的代码，例如，生成HTML表格等。JSTL标签库中提供了<c:forEach>和<c:forTokens>两个循环标签。

（1）<c:forEach>标签

<c:forEach>标签可以枚举集合中的所有元素，也可以循环指定的次数，这可以根据相应的属性确定。

<c:forEach>标签的语法格式如下：

```
<c:forEach items="data" var="name" begin="start" end="finish"
step="step" varStatus="statusName">
    标签主体
</c:forEach>
```

9.2.2 JSTL标准标签库

<c:forEach>标签中的属性都是可选项，可以根据需要使用相应的属性。其属性说明如下表所示。

属　性	类　型	描　述	引用EL
items	数组、集合类、字符串和枚举类型	被循环遍历的对象，多用于数组与集合类	可以
var	String	循环体的变量，用于存储items指定的对象的成员	不可以
begin	int	循环的起始位置	可以
end	int	循环的终止位置	可以
step	int	循环的步长	可以
varStatus	String	循环的状态变量	不可以

9.2.2 JSTL标准标签库

【例9-7】<c:forEach>标签示例

应用<c: forEach >标签循环输出List集合中的内容，并通过<c: forEach >标签循环输出字符串“编程词典”6次。

（2）<c:forTokens>标签

<c:forTokens>标签可以用指定的分隔符将一个字符串分割开，根据分割的数量确定循环的次数。<c:forTokens>标签的语法格式如下：

```
<c:forTokens items="String" delims="char" [var="name"] [begin="start"]
[end="end"] [step="len"] [varStatus="statusName"]>
        标签主体
</c:forTokens>
```

<c:forTokens>标签的属性说明如下表所示。

9.2.2 JSTL标准标签库

属　性	类　型	描　述	引用EL
items	String	被循环遍历的对象，多用于数组与集合类	可以
delims	String	字符串的分割字符，可以同时有多个分隔字符	不可以
var	String	变量名称	不可以
begin	int	循环的起始位置	可以
end	int	循环的终止位置	可以
step	int	循环的步长	可以
varStatus	String	循环的状态变量	不可以

【例9-8】 <c: forTokens >标签示例

应用<c: forTokens >标签分割字符串并显示。

9.2.2 JSTL标准标签库

4．URL操作标签

JSTL标签库宝库<c:import>、<c:redirect>和<c:url>共3种URL标签，它们分别实现导入其他页面、重定向和产生URL的功能。

（1）<c:import>标签

<c:import>标签可以导入站内或其他网站的静态和动态文件到JSP页面中，例如，使用<c:import>标签导入其他网站的天气信息到自己的JSP页面中。与此相比，<jsp:include>只能导入站内资源，<c:import>的灵活性要高很多。<c:import>标签的语法格式如下。

9.2.2 JSTL标准标签库

语法1：

```
<c:import url="url" [context="context"] [var="name"]
 [scope="page|request|session|application"] [charEncoding="encoding"]
标签主体
</c:import>
```

语法2：

```
<c:import url="url" varReader="name" [context="context"]
[charEncoding="encoding"]
```

上面语法中涉及的属性说明如下表所示。

9.2.2 JSTL标准标签库

属　性	类型	描　述	引用EL
url	String	被导入的文件资源的URL路径	可以
context	String	上下文路径，用于访问同一个服务器的其他Web工程，其值必须以“/”开头，如果指定了该属性，那么url属性值也必须以“/”开头	可以
var	String	变量名称，将获取的资源存储在变量中	不可以
scope	String	变量的存在范围	不可以
varReader	String	以Reader类型存储被包含文件内容	不可以
charEncoding	String	被导入文件的编码格式	可以

9.2.2 JSTL标准标签库

（2）<c:redirect>标签

<c:redirect>标签可以将客户端发出的request请求重定向到其他URL服务端，由其他程序处理客户的请求。而在这期间可以对request请求中的属性进行修改或添加，然后把所有属性传递到目标路径。该标签有两种语法格式。

语法1：该语法格式没有标签主体，并且不添加传递到目标路径的参数信息。

```
<c:redirect url="url" [context="/context"]/>
```

语法2：该语法格式将客户请求重定向到目标路径，并且在标签主体中使用<c:param>标签传递其他参数信息。

```
<c:redirect url="url" [context="/context"]>
        ……<c:param>
</c:redirect>
```

上面语法中，url属性用于指定待定向资源的URL，它是标签必须指定的属性，可以使用EL；context属性用于在使用相对路径访问外部context资源时，指定资源的名字。

9.2.2 JSTL标准标签库

（3）<c:url>标签

<c:url>标签用于生成一个URL路径的字符串，这个生成的字符串可以赋予HTML的<a>标记实现URL的连接，或者用这个生成的URL字符串实现网页转发与重定向等。在使用该标签生成URL时还可以搭配<c:param>标签动态添加URL的参数信息。<c:url>标签有两种语法格式。

语法1：

```
<c:url value="url" [var="name"] [scope="page|request|session|application"]
[context=
    "context"]/>
```

该语法将输出产生的URL字符串信息，如果指定了var和scope属性，相应的URL信息就不再输出，而是存储在变量中以备后用。

9.2.2 JSTL标准标签库

语法2：

```
<c:url value="url" [var="name"] [scope="page|request|session|application"]
[context=
    "context"]>
            <c:param>
</c:url>
```

语法格式2不仅实现了语法格式1的功能，而且还可以搭配<c:param>标签完成带参数的复杂URL信息。

<c:url>标签的语法中所涉及的属性说明如下表所示。

9.2.2 JSTL标准标签库

属　性	类型	描　述	引用EL
url	String	生成的URL路径信息	可以
context	String	上下文路径，用于访问同一个服务器的其他Web工程，其值必须以“/”开头，如果指定了该属性，那么url属性值也必须以“/”开头	可以
var	String	变量名称，将获取的资源存储在变量中	不可以
scope	String	变量的存在范围	不可以
context	String	url属性的相对路径	可以

9.2.2 JSTL标准标签库

（4）<c:param>标签

<c:param>标签只用于为其他标签提供参数信息，它与本节中的其他3个标签组合可以实现动态定制参数，从而使标签可以完成更复杂的程序应用。<c:param>标签的语法格式如下：

```
<c:param name="paramName" value="paramValue"/>
```

在上面的语法中，name属性用于指定参数名称，可以引用EL；value属性用于指定参数值。

9.2.3 自定义标签库的开发

自定义标签是程序员自己定义的JSP语言元素，它的功能类似于JSP自带的<jsp:forward>等标准动作元素。实际上自定义标签就是一个扩展的Java类，它是运行一个或者两个接口的JavaBean。当多个同类型的标签组合在一起时就形成了一个标签库，这时候还需要为这个标签库中的属性编写一个描述性的配置文件，这样服务器才能通过页面上的标签查找到相应的处理类。使用自定义标签可以加快Web应用开发的速度，提高代码重用性，使得JSP程序更加容易维护。引入自定义标签后的JSP程序更加清晰、简洁、便于管理维护以及日后的升级。

9.2.3 自定义标签库的开发

1. 自定义标签的定义格式

自定义标签在页面中通过XML语法格式来调用的。它们由一个开始标签和一个结束标签组成，具体定义格式如下。

（1）无标签体的标签

无标签体的标签有两种格式，一种是没有任何属性的，另一种是带有属性的。例如下面的代码：

```
<wgh:displayDate/>                                          <!--无任何属性-->
<wgh:displayDate name="contact" type="com.UserInfo"/>  <!--带属性-->
```

在上面的代码中，wgh为标签前缀，displayDate为标签名称，name和type是自定义标签使用的两个属性。

9.2.3 自定义标签库的开发

（2）带标签体的标签

自定义的标签中可包含标签体，例如下面的代码：

```
<wgh:iterate>Welcome to BeiJing</wgh:iterate>
```

2. 自定义标签的构成

自定义标签由实现自定义标签的Java类文件和自定义标签的TLD文件构成。

（1）实现自定义标签的Java类文件

任何一个自定义标签都要有一个相应的标签处理程序，自定义标签的功能是由标签处理程序定义的。因此，自定义标签的开发主要就是标签处理程序的开发。

9.2.3 自定义标签库的开发

标签处理程序的开发有固定的规范，即开发时需要实现特定接口的Java类，开发标签的Java类时，必须实现Tag或者BodyTag接口类（它们存储在javax.servlet.jsp.tagext包下）。BodyTag接口是继承了Tag接口的子接口，如果创建的自定义标签不带标签体，则可以实现Tag接口，如果创建的自定义标签包含标签体，则需要实现BodyTag接口。

（2）自定义标签的TLD文件

自定义标签的TLD文件包含了自定义标签的描述信息，它把自定义标签与对应的处理程序关联起来。一个标签库对应一个标签库描述文件，一个标签库描述文件可以包含多个自定义标签声明。

9.2.3 自定义标签库的开发

自定义标签的TLD文件的扩展名必须是.tld。该文件存储在Web应用的WEB-INF目录下或者子目录下，并且一个标签库要对应一个标签库描述文件，而在一个描述文件中可以保存多个自定义标签的声明。

自定义标签的TLD文件的完整代码如下：

9.2.3 自定义标签库的开发

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee web-jsptaglibrary_2_0.xsd"
    version="2.0">
    <description>A tag library exercising SimpleTag handlers.</description>
    <tlib-version>1.2</tlib-version>
                <jsp-version>1.2</jsp-version>
     <short-name>examples</short-name>
<tag>
                <description>描述性文字</description>
     <name>showDate</name>
                <tag-class>com.ShowDateTag</tag-class>
                <body-content>empty</body-content>
                <attribute>
                            <name>value</name>
                            <required>true</required>
                </attribute>
</tag>
</taglib>
```

在上面的代码中，<tag>标签用来提供在标签内的自定义标签的相关信息，在<tag>标签内包括许多子标签，如下表所示。

9.2.3 自定义标签库的开发

标 签 名 称	说　明
description	标签的说明（可省略）
display-name	供工具程序显示用的简短名称（可省略）
icon	供工具程序使用的小图标
name	标签的名称，在同一个标签库内不可以有同名的标签，该元素指定的名称可以被JSP页面作为自定义标签使用
tag-class	映射类的完整名称，用于指定与name子标签对应的映射类的名称
tei-class	标签设计者定义的javax.servlet.jsp.tagext.TagExtraInfo的子类，用来指定返回变量的信息（可省略）
body-content	Body内容的类型，其值可以为empty、scriptless、tagdependent其中之一，其值为emply时，表示body必须是空；其值为tagdependent时，表示body的内容由标签的实现自行解读，通常是用在body内容是别的语言时，例如SQL语句
variable	声明一个由标签返回给调用网页的EL变量（可省略）
attribute	声明一个属性（可省略）
dynamic-attributes	此标签是否可以有动态属性，默认值为false；若为true，则TagHandler必须实现javax.servlet.jsp.tagext.DynamicAttrbutes的接口
example	此标签的使用范例（可省略）
tag-extension	提供此标签额外信息给程序（可省略或多于一个此标签）

9.2.3 自定义标签库的开发

说明：通过<name>子标签和<tag-class>子标签可以建立自定义标签和映射类之间的对应关系。

3．在JSP文件中引用自定义标签

JSP文件中，可以通过下面的代码引用自定义标签：

```
<%@ taglib uri="tld uri"  prefix="taglib.prefix"%>
```

上面语句中的uri和prefix说明如下。

（1）uri属性

uri属性指定了tld文件在Web应用中的存放位置，此位置可以采用以下两种方式指定。

① 在uri属性中直接指明tld文件的所在目录和对应的文件名，例如下面的代码：

9.2.3 自定义标签库的开发

```
<%@ taglib uri="/WEB-INF/showDate.tld"  prefix="taglib.prefix"%>
```

② 通过在web.xml文件中定义一个关于tld文件的uri属性，让JSP页面通过该uri属性引用tld文件，这样可以向JSP页面隐藏tld文件的具体位置，有利于JSP文件的通用性。例如在Web.xml中进行以下配置：

```
<taglib>
        <taglib-uri>showDateUri</taglib-uri>
        <taglib-location>/WEB-INF/showDate.tld</taglib-location>
</taglib>
```

在JSP页面中就可应用以下代码引用自定义标签：

```
<%@ taglib uri="showDateUri "  prefix="taglib.prefix"%>
```

9.2.3 自定义标签库的开发

（2）prefix属性

prefix属性规定了如何在JSP页面中使用自定义标签，即使用什么样的前缀来代表标签，使用时标签名就是在tld文件中定义的<tag></tag>段中的<name>属性的取值，它要和前缀之间用冒号“:”隔开。

【例9-9】 自定义标签示例

创建用于显示当前系统日期的自定义标签，并在JSP页面中调用该标签显示当前系统日期。

9.3 JSP框架技术

在开发JSP程序时，采用合适的开发框架可以很好地提高开发效率。Struts、Spring和Hibernate是Java Web开发中比较优秀的开源框架，这些框架各有特点，各自实现了不同的功能，在开发的过程中，如果能够将这些框架集成起来，将会使开发过程大大简化，很大程序上降低开发成本。

9.3.1 Struts框架 √

9.3.2 Spring框架 √

9.3.3 Hibernate技术 √

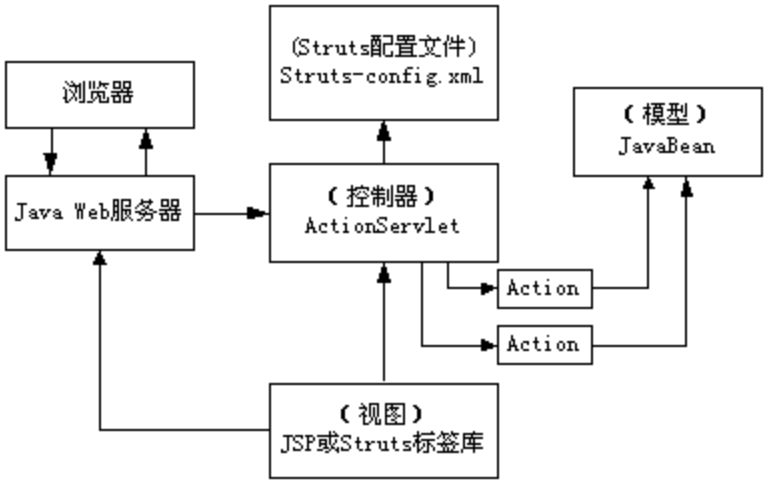
9.3.1 Struts框架

Struts框架是Apache组织的一个开放源代码项目，它是采用Java Servlet和JSP技术来构建基于MVC体系结构的Web应用程序的框架。Struts框架具有良好的架构和设计、可重用、模块化、扩展性强等特点，因此已经被广泛应用于Web应用开发。

1. Struts实现MVC的机制

Struts框架是目前非常流行的基于Java技术开发的JSP Web应用开发框架，它遵循了MVC设计模式。在Struts框架中，模型由实现业务逻辑的JavaBean组件构成，控制器由ActionServlet和Action来实现，视图由一组JSP文件与Struts标签库构成，如下图所示。

9.3.1 Struts框架



（1）视图

Struts中的视图部分依然可以采用JSP来实现。在这些JSP文件中没有业务逻辑，也没有模型信息，只有标签，这些标签可以是标准的JSP标签或客户化标签，如Struts标签库中的标签。

9.3.1 Struts框架

当用户通过视图向Servlet发送数据时使用了Struts中的ActionForm组件，该组件通常也被归于视图。ActionForm的作用就是将用户提交的数据编译成Bean对象，除了基本的getXxx()和setXxx()方法外，它还提供了另外两种特殊的方法用于对用户提交的数据进行一些初始化以及验证操作。

（2）模型

模型表示应用程序的状态和业务逻辑。对于大型应用，业务逻辑通常采用EJB或其他对象关系映射工具（如Hibernate）来实现模型组件。

9.3.1 Struts框架

（3）控制器

Struts提供了一个控制器组件ActionServlet，它继承自HttpServlet，并重载了HttpSerlvet的doGet()、doPost()方法，可以接受HTTP响应并进行转发；同时还提供了使用XML进行转发Mapping（映射）的功能。

2．Struts框架的工作流程

当启动一个采用Struts框架开发的Web应用程序时，ActionServlet就会被加载并被初始化。然后ActionServlet读取Struts配置文件中的信息，并根据文件中的各模块配置来初始化相应的配置对象。如对Action的映射信息，都保存在ActionMapping对象中。当用户的请求属于ActionServlet所处理请求的模式时，ActionServlet被调用，Struts的处理工作开始。

9.3.1 Struts框架

下面将介绍Struts具体的工作流程。

（1）中央控制器根据用户的请求，在Struts配置文件中的<action-mappings>元素中查找匹配该请求的<action>子元素。如果不存在，则返回类似于下面的异常代码：

```
Cannot retrieve mapping for action /logon
```

（2）在匹配的<action>元素中查找由scope属性指定的范围中是否存在由name属性指定的ActionForm Bean。如果不存在，就创建一个新的ActionForm对象。然后将用户提交的表单中的数据保存在该ActionForm对象中，并将该ActionForm对象存入scope属性指定的范围内。

9.3.1 Struts框架

（3）如果<action>元素中validate属性值为true，则调用ActionForm的validate()方法，进行表单验证。

（4）若ActionForm的validate()方法返回null或返回的ActionErrors对象中不包含任何ActionMessage对象，则表单验证成功，进行第（5）步；否则表单验证失败，ActionServlet将请求转发给input属性指定的JSP页面。

（5）中央控制器将控制权转交给<action>元素中type属性指定的Action类。如果相应的Action类对象不存在，则创建该类对象，接下来Action类的execute()方法被调用。

9.3.1 Struts框架

（6）在Action类的execute()方法中进行业务逻辑处理，并返回一个ActionForward对象。控制权被交回ActionServlet，ActionServlet将返回的ActionForward对象与<action>元素中的<forward>子元素进行匹配，并将请求转发给指定的JSP组件。

（7）如果ActionForward对象指定的是另外的Action动作（如*.do），则返回第（1）步再次执行该流程，否则生成动态网页返回给用户。

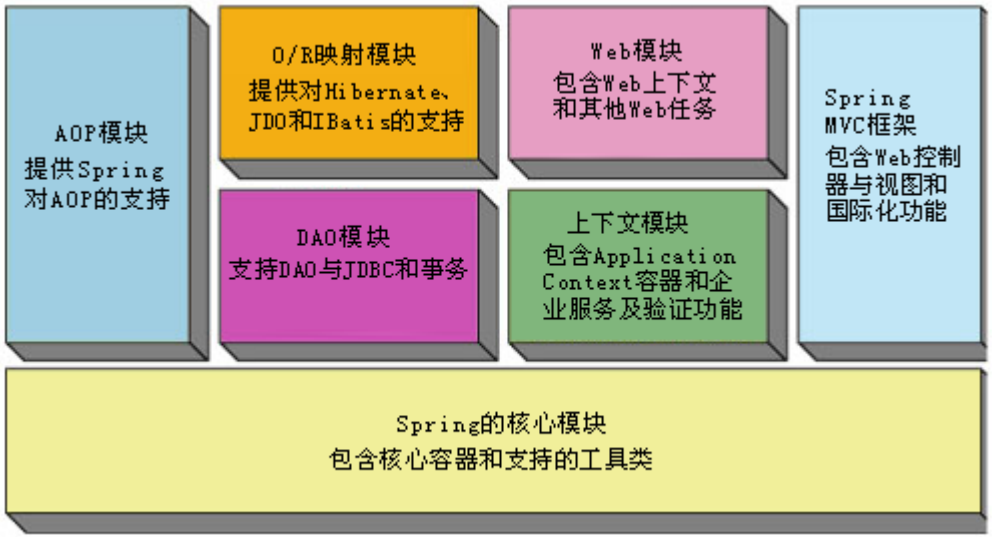
9.3.2 Spring框架

Spring是一个开源的框架，由Rod Johnson创建，从2003年初正式启动。它能够降低开发企业应用程序的复杂性，可以使用Spring替代EJB开发企业级应用，而不用担心工作量太大、开发进度难以控制和复杂的测试过程等问题。它以IoC（反向控制）和AOP（面向切面编程）两种先进的技术为基础，完美地简化了企业级开发的复杂度。

Spring框架主要由核心模块、上下文模块、AOP模块、DAO模块、Web模块等7大模块组成，它们提供了企业级开发需要的所有功能，而且每个模块都可以单独使用，也可以和其他模块组合使用，灵活且方便的部署可以使开发的程序更加简洁灵活。

9.3.2 Spring框架

Spring的7个模块的部署如下图所示。

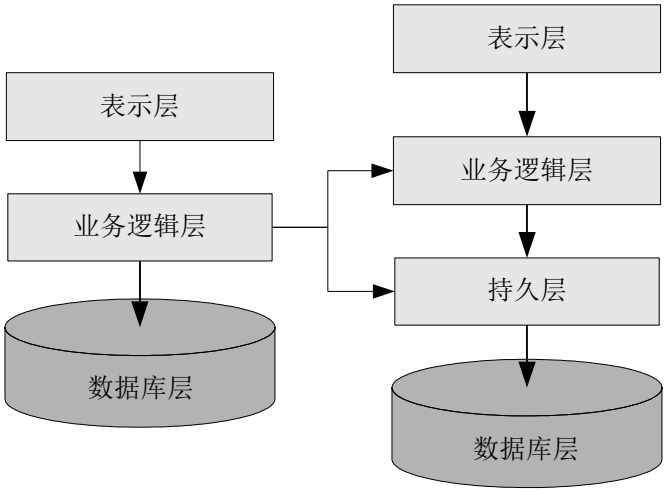


9.3.3 Hibernate技术

Java是一种面向对象的编程语言，但是通过JDBC方式操作数据库，运用的是面向过程的编程思想，为了解决这一问题，提出了对象—关系映射（Object Relational Mapping，ORM）模式。通过ORM模式，可以实现运用面向对象的编程思想操作关系型数据库。Hibernate技术为ORM提供了具体的解决方案，实际上就是将Java中的对象与关系数据库中的表做一个映射，实现它们之间自动转换的解决方案。

Hibernate在原有3层架构（MVC）的基础上，从业务逻辑层又分离出一个持久层，专门负责数据的持久化操作，使业务逻辑层可以真正地专注于业务逻辑的开发，不再需要编写复杂的SQL语句，增加了持久层的软件分层结构如下图所示。

9.3.3 Hibernate技术



Hibernate在Java对象与关系数据库之间起到了一个桥梁的作用，负责两者之间的映射，在Hibernate内部还封装了JDBC技术，向上一层提供面向对象的数据访问API接口。Hibernate特点如下：

9.3.3 Hibernate技术

（1）它负责协调软件与数据库的交互，提供了管理持久性数据的完整方案，让开发者能够专著于业务逻辑的开发，不再需要考虑所使用的数据库及编写复杂的SQL语句，使开发变得更加简单和高效；

（2）应用者不需要遵循太多的规则和设计模式，让开发人员能够灵活的运用；

（3）Hibernate支持各种主流的数据源，目前所支持的数据源包括DB2、MySQL、Oracle、Sybase、SQL Server、PostgreSQL、WebLogic Driver和纯Java驱动程序等；

（4）它是一个开放源代码的映射框架，对JDBC只做了轻量级的封装，让Java程序员可以随心所欲地运用面向对象的思想操纵数据库，无需考虑资源的问题。