Чтобы облегчить внедрение динамической структуры, JSP использует ряд тегов, которые дают возможность проектировщику страницы вставить значение полей объекта JavaBean в файл JSP.

Содержимое Java Server Pages (теги HTML, теги JSP и скрипты) переводится в сервлет код-сервером. Этот процесс ответствен за трансляцию как динамических, так и статических элементов, объявленных внутри файла JSP. Об архитектуре сайтов, использующих JSP/Servlet-технологии, часто говорят как о thin-client (использование ресурсов клиента незначительно), потому что большая часть логики выполняется на сервере.

JSP составляется из стандартных HTML-тегов, JSP-тегов, action-тегов, JSTL и пользовательских тегов. В спецификации JSP 2.0 существует пять основных тегов:

<%@ *директива* **%>** – используются для установки параметров серверной страницы JSP.

<%! *объявление* **%>** – содержит переменные Java и методы, которые вызываются в expression-блоке и являются полями генерируемого сервлета. Объявление не должно производить запись в выходной поток **out** страницы, но может быть использовано в скриптлетах и выражениях.

<% *скриптлет* **%>** – вживление Java-кода в JSP-страницу. Скриптлеты обычно используют маленькие блоки кода и выполняются во время обработки запроса клиента. Когда все скриптлеты собираются воедино в том порядке, в котором они записаны на странице, они должны представлять собой правильный код языка программирования. Контейнер помещает код Java в метод **_jspService()** на этапе трансляции.

<%= *вычисляемое выражение* **%>** – операторы языка Java, которые вычисляются, после чего результат вычисления преобразуется в строку **String** и посылается в поток **out**.

<%-- *JSP-комментарий* --%> – комментарий, который не отображается в исходных кодах JSP-страницы после этапа выполнения.

Стандартные элементы action

Большинство тегов, объявленных выше, применяются не так уж часто. Наиболее используемыми являются стандартные действия версии JSP 2.0. Они позволяют создавать правильные JSP –документы с помощью следующих тегов:

- **jsp:declaration** – объявление, аналогичен тегу **<%! ... %>**;
- **jsp:scriptlet** – скриптлет, аналогичен тегу **<% ... %>**;
- **jsp:expression** – скриптлет, аналогичен тегу **<%= ... %>**;
- **jsp:text** – вывод текста;
- **jsp:useBean** – позволяет использовать экземпляр компонента Java Bean. Если экземпляр с указанным идентификатором не существует, то он будет создан с областью видимости **page** (страница), **request** (запрос), **session** (сессия) или **application** (приложение). Объявляется, как правило, с атрибутами **id** (имя объекта), **scope** (область видимости), **class** (полное имя класса), **type** (по умолчанию **class**).

```
<jsp:useBean id="ob"
             scope="session"
                   class="test.MyBean" />
```

Создан объект **ob** класса **MyBean**, и в дальнейшем через этот объект можно вызывать доступные методы класса. Специфика компонентов JavaBean в том, что если компонент имеет поле **field**, экземпляр компонента имеет параметр **field**, а метод, устанавливающий значение, должен называться **setField(type value)**, возвращающий значение – **getField()**.

```
package test;
public class MyBean {
     private String field = "нет информации";
     public String getField() {
           return info;
     }
     public void setField(String f) {
           field = f;
     }
}
```

- **jsp:setProperty** – позволяет устанавливать значения полей указанного в атрибуте **name** объекта. Если установить значение **property** в «*», то значения свойств компонента JavaBean будут установлены таким образом, что будет определено соответствие между именами параметров и именами методов-установщиков (setter-ов) компонента:

```
<jsp:setProperty name="ob"
                        property="field"
                        value="привет" />
```

- **jsp:getProperty** – получает значения поля указанного объекта, преобразует его в строку и отправляет в неявный объект **out**:

```
<jsp:getProperty name="ob" property="field" />
```

- **jsp:include** – позволяет включать файлы в генерируемую страницу при запросе страницы:

```
<jsp:include page="относительный URL"
                        flush="true"/>
```

- **jsp:forward** – позволяет передать запрос другой странице:

```
<jsp:forward page="относительный URL"/>
```

- **jsp:plugin** – замещается тегом **<OBJECT>** или **<EMBED>**, в зависимости от типа браузера, в котором будет выполняться подключаемый апплет или Java Bean.
- **jsp:params** – группирует параметры внутри тега **jsp:plugin**.
- **jsp:param** – добавляет параметры в объект запроса, например в элементах **forward**, **include**, **plugin**.
- **jsp:fallback** – указывает содержимое, которое будет использоваться браузером клиента, если подключаемый модуль не сможет запуститься. Используется внутри элемента **plugin**.

В качестве примера можно привести следующий фрагмент:

```
<jsp:plugin type="bean | applet"
                  code="test.com.ReadParam"
```

```
                    width="250"
                    height="250">
            <jsp:params>
               <jsp:param name="bNumber" value="7" />
               <jsp:param name="state" value="true" />
            </jsp:params>
            <jsp:fallback>
                  <p> unable to start plugin </p>
            </jsp:fallback>
      </jsp:plugin>
```

Код апплета находится в примере 5 главы 11, и пакет, в котором он объявлен, должен быть расположен в корне папки **/WEB-INF**, а не в папке **/classes**.

Элементы **<jsp:attribute>**, **<jsp:body>**, **<jsp:invoke>**, **<jsp:doBody>**, **<jsp:element>**, **<jsp:output>** используются в основном при включении в страницу пользовательских тегов.

JSP-документ

Предпочтительно создавать JSP-страницу в виде JSP-документа – корректного XML-документа, который ссылается на определенное пространство имен, содержит стандартные действия JSP, пользовательские теги и теги ядра JSTL, XML-эквиваленты директив JSP. В JSP-документе вышеперечисленные пять тегов неприменимы, поэтому их нужно заменять стандартными действиями и корректными тегами. JSP-документы необходимо сохранять с расширением **.jspx**.

Директива **taglib** для обычной JSP:

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

для JSP-документа:

```
<jsp:root xmlns:c="http://java.sun.com/jsp/jstl/core"/>
```

Директива **page** для обычной JSP:

```
<%@ page contentType="text/html"%>
```

для JSP-документа:

```
<jsp:directive.page contentType="text/html" />
```

Директива **include** для обычной JSP:

```
<%@ include file="file.jspf"%>
```

для JSP-документа:

```
<jsp:directive.include file="file.jspf" />
```

Ниже приведены два примера, демонстрирующие различие применения стандартных действий и тегов при создании JSP-страниц и JSP-документов.

```
<!--пример # 1 : обычная jsp-страница: page.jsp -->
<%@ page contentType="text/html; charset=Cp1251" %>
<html><head><title>JSP-страница</title></head>
<%! private int count = 0;
      String version = new String("J2EE 1.5");
      private String getName(){return "J2EE 1.6";} %>
```