JSP

Основные идеи JSP

1. Использовать обычные HTML теги для создания разметки.

2. Иметь возможность встраивать в HTML java код, с помощью специальных jsp специфических конструкций.

3. JSP страницу сервер транслирует в сервлет, как правило, при первом обращении к ней. Трансляция осуществляется один раз, выполнение кода результирующего сервлета – каждый раз в ответ на запрос.

Соотношение между сервлетами и JSP

С помощью сервлетов можно сделать все, что можно сделать с помощью JSP.

Основное предназначение JSP:
упрощение создания и поддержки View слоя (представления информации) в web приложениях.

Жизненный цикл JSP

После того, как JSP страница создана, при поступлении к ней первого запроса, сервер осуществляет следующие действия:

1) транслирует JSP в сервлет;
2) компилирует сервлет;
3) создает экземпляр сервлета;
4) инициализирует сервлет (метод init);
5) делает вызов метода service.

При поступлении второго запроса к той же JSP сервер делает следующие действия:

1) ~~транслирует JSP в сервлет;~~
2) ~~компилирует сервлет;~~
3) ~~создает экземпляр сервлета, и размещает его в памяти;~~
4) ~~инициализирует сервлет (метод init);~~
5) делает вызов метода service.

Если сервер будет перегружен, то при поступлении запроса к той же JSP, он осуществит следующие действия:

1) ~~транелирует JSP в сервлет;~~
2) ~~компилирует сервлет;~~
3) создает экземпляр сервлета;
4) инициализирует сервлет (метод init);
5) делает вызов метода service.

Если исходная страница JSP была изменена, то при поступлении к ней запроса, сервер сделает следующее:

1) транслирует JSP в сервлет;
2) компилирует сервлет;
3) создает экземпляр сервлета;
4) инициализирует сервлет (метод init);
5) делает вызов метода service.

Т.е. трансляция JSP будет осуществлена после изменения JSP страницы (как правило, при первом запросе к ней).

Элементы синтаксиса JSP

1) HTML код. Конструкции вида <h1>test</h1> будут вставлены в результирующий сервлет в виде:

```
out.print("<h1>test</h1>");
```

2) HTML комментарии.

```
<!-- это HTML комментарий -->
```

будет передан клиенту в результирующей HTML странице.

3) JSP комментарии.

```
<%-- это JSP комментарий --%>
```

клиенту передан не будет (остается на сервере, служит для комментирования JSP кода).

4) Экранирование конструкций <%, %>:

<% ==> <\%

%> ==> %\>

5) Скриптовые элементы JSP.

6) Директивы JSP.

7) Действия JSP.

8) Пользовательские теги.

Скриптовые элементы JSP

1) Декларации
Общий вид: <%! КОД_ДЕКЛАРАЦИИ %>

Вставляют в результирующий сервлет определенный Java код, причем он будет размещен непосредственно в теле класса (но не внутри его методов).

2) Выражения

Общий вид: <%= КОД_ВЫРАЖЕНИЯ %>
Вычисляет и вставляет в поток вывода сервлета соответствующее значение.

3) Скриплеты

Общий вид: <% КОД_СКРИПЛЕТА %>

Код, записанный в скриплете, будет вставлен в результирующий сервлет в метод _jspService, который вызывает метод service сервлета.

Выражения

Вид:

```
<%= Expression %>
```

Выражение вычисляется, конвертируется в String и результат появится в том месте, где расположена данная конструкция.

Пример:

```
Time: <%= new java.util.Date() %>
```

Замечание: точка с запятой в конце выражения не ставится.

При трансляции JSP в сервлет, выражение вида

```
<%= Expression %>
```

будет транслировано в код метода _jspService

```
out.println(Expression);
```

Неявные объекты, доступные на JSP странице

1) request - запрос к JSP странице.

2) responce - ответ клиенту.

3) out - поток вывода, связанный с ответом клиенту.

4) session - сессия, связанная с запросом.

5) application - сервлетный контекст.

Примеры выражений с использованием неявных объектов

Получить значение параметра запроса ParamName:

```
<%= request.getParameter("ParamName") %>
```

Получить атрибут запроса по имени:

```
<%= request.getAttribute("AttributeName");
```

Получить атрибут сессии по имени:

```
<%= session.getAttribute("AttributeName");
```

Скриплеты

Вид:

<% Java код %>

Содержимое скриплета будет записано внутри результирующего сервлета, внутри метода _jspService.

Пример JSP и результирующего сервлета

JSP:

```
<h1>text<h2>
<%=  getX() %>
<% m(); %>
```

_jspService сервлета:

```
out.println("<h1>text<h2>");
out.println("getX()");
m();
```

Декларации

Вид:

```
<%! КОД_ДЕКЛАРАЦИИ %>
```

Код декларации в результирующем сервлете будет вставлен на уровне элементов (class members) класса сервлета.

Примеры:

```
<%! private int x = 2; %>
<%!
    pirvate String m() { ... }
%>
```

Замечание: следует избегать объявления с помощью деклараций методов внутри JSP страницы; целесообразно выносить данную функциональность в отдельный Java класс и использовать его на JSP странице с помощью стандартных средств JSP.

Замечание: неявные объекты JSP страницы (request, session и т.д.) недоступны внутри объявляемых с помощью деклараций методов.

Директивы

Вид:

<%@ НАЗВАНИЕ_ДИРЕКТИВЫ АТРИБУТЫ %>

Директивы - это сообщения контейнеру JSP, которые дают возможность определить параметры страницы, подключение других ресурсов, использовать собственные библиотеки тегов.

Для JSP страниц существует три директивы:
1) page; 2) taglib 3) include

Директива page

Вид:

<%@ page АТРИБУТЫ %>

Атрибуты:

1) import; 2) contentType; 3) language
4) pageEncoding 5) session 6) isELignored
7) buffer 8) errorPage 9) extends
10) isThreadSafe

Атрибут import:

```
<%@ page import="pack1.class1, pack2.class2, ... " %>
```

Вставляет в код сервлета импорт соответствующих классов/пакетов.

Атрибут pageEncoding

```
<%@page pageEncoding="Encdogin" %>
```

Указывает, в какой кодировке записана данная JSP страница.

Атрибут contentType:

```
<%@ page contentType="MIME-T" %>
<%@ page contentType="MIME-T; charset=Encoding" %>
```

Устанавливает mime тип и кодировку html страницы, которую вернет сервлет - результат трансляции данной JSP страницы, в ответ на запрос к JSP.

Директива include

Вид:

<%@ include file="FILE_RELATIVE_ADDRESS" %>

Включает содержимое файла (jsp, html, просто текст и т.п.) в состав текущей JSP страницы **на этапе трансляции** JSP в сервлет.

Адрес подключаемого файла - относительный по отношению текущей JSP страницы. Если адрес начинается с **/**, то он будет определен относительно корня web приложения.

Директива taglib

Вид:

```
<%@ taglib prefix="PREFIX" uri="TAG_LIBRARY_URI" %>
```

Подключает к странице JSP библиотеку тегов.
TAG_LIBRARY_URI - идентификатор (уникальный) библиотеки тегов;
PREFIX - префикс для подключаемых тегов.

Пример:

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