

JAVA SERVER PAGES

Ненко Табаков

Пламен Танов

Технологическо училище “Електронни системи”

Технически университет – София

23 ноември 2008



ЛИТЕРАТУРА НЕОБХОДИМИ ПРОГРАМИ

- The Java EE 5 Tutorial -

<http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf>

- Java API документация - <http://java.sun.com/javase/6/docs/api/>

- Eclipse - www.eclipse.org

- Apache Tomcat - <http://tomcat.apache.org/>

- Step-by-step tutorial:

<http://www.java-tips.org/java-tutorials/tutorials/introduction-to-java-se1>

ВЪВЕДЕНИЕ

- *Java Server Pages (JSP)* позволяват лесно да се създава WEB съдържание, което има както статична така и динамична част
- *JSP* притежава всички динамични свойства на сървлет, но предоставя по – удобен начин за създаване на статично съдържание

ВЪВЕДЕНИЕ

Основните характеристики на *JSP* технологията са:

- Език за разработка на *JSP* страници – текстови документи, които описват как да се обработи заявката и как да се конструира отговорът
- Език за достъп до обекти от страна на сървъра
- Механизъм за разширяване на езика за разработка на *JSP*

КАКВО Е *JSP*

- *JSP* страница е текстов документ, който има две части
 - Статични данни, които могат да бъдат всякакъв тип текст (*HTML*, *XML*, *WML* и т.н.)
 - *JSP* елементи, които създават динамично съдържание
- Препоръчваното разширение на файл съдържащ *JSP* код е *.jsp*
- Една страница може да включва в съдържанието си друга страница. От своя страна тази друга страница може да бъде както цяла *JSP* страница, така и само фрагмент
- Препоръчваното разширение на файл съдържащ фрагмент е *.jspx*

ПРИМЕР

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"%>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Hello World!</title>
</head>
<body>
<%= "Hello World!" %>
</body>
</html>
```

ЖИЗНЕН ЦИКЪЛ НА *JSP*

- *JSP* страница се транслира до сървлет
- Когато има запитване към *JSP* страница, WEB контейнерът първо проверява дали транслираният сървлет е по – стар от страницата
- Ако е по – стар WEB контейнерът транслира страницата до сървлет
- По време на разработката, едно от предимствата на *JSP* страница пред сървлетите е, че процесът на създаване е автоматичен

ЖИЗНЕН ЦИКЪЛ НА *JSP*



Заявка към WEB
контейнер

Една *JSP*
страница не се
транслира до
сървлет всеки
път, а само ако
съдържанието ѝ
се е променило
спрямо
последния път,
когато е била
транслирана

Отговор на WEB
контейнера

Определяне на правилната *JSP* страница

Прочитане на *JSP* страницата

Транслиране на *JSP* страницата до сървлет

Създаване на обект от транслирания сървлет

ЖИЗНЕН ЦИКЪЛ НА *JSP*

- По време на транслиране статичните компоненти на *JSP* страницата се трансформират в *Java* код, който ще ги вмъкне в изходния поток на отговора
- Директивите се използват за контрол върху това как WEB контейнерът транслира и изпълнява *JSP* страницата
- Скриптови елементи се добавят в транслирания сървлет
- Изразите написани на *EL(Expression Language)* се подават като параметри на *JSP* процесора

ЖИЗНЕН ЦИКЪЛ НА JSP

- *jsp:[set|get]Property* елементите се преобразуват в извиквания към *JavaBeans* компоненти
- *jsp:[include|forward]* елементите се преобразуват в извиквания към сървлет процесора
- *jsp:plugin* елементът се преобразува в маркър за извикване на аplet
- Таговете, които са от външни библиотеки, се препращат към съответните процесори, които знаят как да ги обработят

ПРИМЕР

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib uri=http://java.sun.com/jsp/jstl/core prefix="c"%>
<%@ taglib uri="/functions" prefix="f"%>
<html>
<head>
<title>Localized Dates</title>
</head>
<body bgcolor="white">
<jsp:useBean id="locales" scope="application"
  class=" mypkg.MyLocales"/>
<form name="localeForm" action="index.jsp"
method="post"><c:set
  var="selectedLocaleString" value="\${param.locale}" />
<c:set
  var="selectedFlag" value=" \${!empty selectedLocaleString}"
  /></form>
</body>
</html>
```

Директиви

Статични данни

Динамични данни

JSP СТАНДАРТНИ ТАГОВЕ

Съществуват пет стандартни типа тагове за *JSP* страници

- Декларативни тагове
- Изрази
- Директиви
- Тагове за скриплети
- Тагове за действия

ДЕКЛАРАТИВНИ ТАГОВЕ

- Тези тагове позволяват да се декларират/дефинират променливи и методи
- Този тип тагове не генерират изходни данни
- Трябва да има ; на края на всяка декларация/дефиниция
- Използва се следният синтаксис - `<%! %>`

ПРИМЕР

декларативни тагове

```
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title>Localized Dates</title>
</head>
<body bgcolor="white">
<%!private int n = 10;

    private String getAccount() {
        return "5";
    }%>
</body>
</html>
```

ИЗРАЗИ

- Тези тагове позволяват да се добавят всякакви *Java* изрази
- Този тип тагове генерират изходни данни
- Изразите не завършват с ;
- Използва се следният синтаксис - `<%= %>`

```
<%@ page contentType="text/html; charset=UTF-8"%>
<html>
<head>
<title>Localized Dates</title>
</head>
<body bgcolor="white">
<%= new java.util.Date() %>
</body>
</html>
```

ДИРЕКТИВИ

- Тези тагове дават специална информация на *JSP* процесора
- Този тип тагове не генерират изходни данни, но променят начина, по който една страница се обработва
- Използва се следният синтаксис - `<%@ %>`

```
<%@ page contentType="text/html; charset=UTF-8"%>
```


ТАГОВЕ ЗА СКРИПЛЕТИ

- Позволяват да се използва *Java* код в страницата
- Използва се следният синтаксис - `<% %>`

```
<%  
    if (posts.size() < 1)  
        out.println("There are no posts!");  
    else  
        for (ForumPost post : posts) {  
            out.println(post);  
        }  
%>
```

ТАГОВЕ ЗА ДЕЙСТВИЕ

Използват се поради три основни причини:

- Позволяват да се използват *JavaBeans* компоненти от страната на сървъра
- Предават контрола между страници
- Поддръжка на аплети
- Използва се следният синтаксис - `<jsp: />`

```
<jsp:useBean id="locales" scope="application"  
  class=" mypkg.MyLocales" />
```

ИЗПЪЛНЕНИЕ НА JSP

Изпълнението на една *JSP* страница може да се контролира чрез набор от параметри (директиви на страницата)

Буфериране:

- Когато една *JSP* страница се изпълнява изходният поток, който се записва в отговора се буферира автоматично
- Размерът на буфера може да се определя чрез следната директива

```
<%@ page buffer="none|xxxkb"%>
```

ИЗПЪЛНЕНИЕ НА JSP

Управление на грешки:

- По време на изпълнение на една *JSP* страница може да настъпят грешки – да се генерират изключения
- За определяне коя страница да се зареди, когато възникне дадена грешка, се използва следната директива

```
<%@ page errorPage="filename"%>
```

- Ако определената страница за грешки е *JSP* страница то в началото ѝ трябва да се добави следната директива

```
<%@ page isErrorPage="true"%>
```

ИЗПЪЛНЕНИЕ НА JSP

Вмъкване на библиотеки става със следната директива

```
<%@ page import="library"%>  
<%@ page import="java.util.*, java.text.*"%>
```

СЪЗДАВАНЕ НА СТАТИЧНО СЪДЪРЖАНИЕ

- Статично съдържание се създава като просто се добавя към страницата
- Статичното съдържание може да бъде във всякакъв текстови формат – *HTML, WML, XML*
- За определяне на формата се използва следната директива:

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

- За повече информация:

<http://www.iana.org/assignments/media-types/>

СЪЗДАВАНЕ НА СТАТИЧНО СЪДЪРЖАНИЕ

За определяне на кодирането на символите (encoding) могат да се използват две директиви“

```
<%@ page contentType="text/html; charset=UTF-8" %>
```

```
<%@ page pageEncoding="UTF-8" %>
```

СЪЗДАВАНЕ НА ДИНАМИЧНО СЪДЪРЖАНИЕ

- Динамично съдържание се създава като се използват свойства на *Java* обекти
- Могат да бъдат достигани *EnterpriseBeans (EJB)* обекти и *JavaBeans* обекти
- *JSP* автоматично предоставя някои обекти за използване. Също могат да бъдат създавани и използвани обекти, които съдържат служебна информация

ОБЕКТИ СЪДЪРЖАЩИ СЛУЖЕБНА ИНФОРМАЦИЯ

Обектите съдържащи служебна информация се създават от WEB контейнера и предоставят данни за дадена заявка, страница, сесия или приложение

Такива обекти са:

- *application : ServletContext*
- *config: ServletConfig*
- *out: JspWriter*
- *pageContext: PageContext*
- *request: HttpServletRequest*
- *response: HttpServletResponse*
- *session: HttpSession*

ОБЕКТИ СЪДЪРЖАЩИ СЛУЖЕБНА ИНФОРМАЦИЯ

```
<%@page language="java"contentType="text/html;charset=UTF-8"%>
<html>
...
<body>
<% String servletInfo = application.getServerInfo();
   java.util.Enumeration enumAttrNames = application
       .getAttributeNames();%>
<p>Servlet Info: <%=servletInfo%> <br>
<%while (enumAttrNames.hasMoreElements()) {
   Object obj = enumAttrNames.nextElement();
   out.println(obj + " = ");
   out.println(application.getAttribute((String) obj));
}%>
</p>
</body>
</html>
```

СПОДЕЛЕНИ ОБЕКТИ

Със следната директива се определя дали достъпът до обекти от *JSP* страница

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

- Когато е *true*, WEB контейнерът може да избере да изпрати няколко заявки едновременно към *JSP* страница (**multithreaded**)
- Когато не е избрано друго, това е поведението по подразбиране
- В този случай трябва да сме сигурни, че достъпът до споделени обекти е синхронизиран
- Когато е *false* заявките се изпращат една след друга в реда, в който са получени

EXPRESSION LANGUAGE (EL)

- **EL** позволява да се използват прости изрази за динамично четене на данни от *JavaBean* компоненти
- С **EL** може динамично да се достигат данни от *JavaBean* компоненти, различни структури от данни и обекти съдържащи служебна информация
- С **EL** може динамично да се записват данни в *JavaBean* компоненти
- С **EL** може да се извикват статични и публични методи
- С **EL** може динамично да се извършват аритметични операции
- **EL** може да се разширява, така че да поддържа изрази, които не са част от езика

EL – НЕПОСРЕДСТВЕНО ИЗЧИСЛЯВАНЕ НА ИЗРАЗИ

- Непосредствено изчисляване (*immediate evaluation*) означава, че **JSP** процесорът изчислява израза и връща стойността му веднага след като страницата е показана
- Такива изрази използват `#{}` синтаксис
- Такива изрази могат да се използват само в текстови шаблони или като стойност в **JSP** таг, който може да приема изрази по време на изпълнение

```
<fmt:formatNumber value="#${sessionScope.cart.total}"/>
```

EL – ПРЕДИЗВИКАНО ИЗЧИСЛЯВАНЕ НА ИЗРАЗИ

- Предизвикано изчисляване (*deffered evaluation*) означава, че технологията използваща *EL* може да използва свой собствен механизъм за изчисляване на израза по някое време по – късно в жизнения цикъл на страницата
- Такива изрази използват `#{}` синтаксис

```
<h:inputText id="name" value="#{customer.name}" />
```