# 08.04.2020
# Servlets API

Намиот Д.Е.
dnamiot@gmail.com

# JSP директивы & теги

• Время действия:
директивы: на этапе компиляции
теги: на этапе выполнения

• Теги – динамические сущности

• <%@ directive attribute = "value" %>
Обрабатываются компилятором, в результирующем коде не остаются

• <jsp: tag_name attribute = "value" />
Компилятор вставляет вызовы, которые будут отрабатываться при выполнении

# JSP директивы

- <%@ page ... %>

Задает параметры страницы

- <%@ include ... %>

Включение файла (аналог #include )

- <%@ taglib ... %>

Определяет (подключает) пользовательскую библиотеку

# Page directive

- buffer

Размер буфера для вывода

- autoFlush

Политика сброса буфера

- contentType

Задает заголовок Content-type

- errorPage

Определяет страницу для переадресации при ошибках

# Page directive

• isErrorPage

Это страница для обработки ошибок

• extends

Определяет базовый класс для JSP страницы

• import

Подключение Java-пакетов

• info

Комментарий. Доступен через getServletInfo()

• session

Создавать или нет сессию

# Page directive

• isELIgnored

Игнорировать или нет EL expression within the JSP page will be

• isScriptingEnabled

Разрешать или нет Java фрагменты на странице.

# Page directive: примеры

<%@ page buffer = "none" %>

Не буферизованный вывод

<%@ page buffer = «16kb" autoflush=“true” %>

Буфер 16 kb

Контент:

<%@ page contentType = "text/xml" %>

<%@ page contentType = "application/msword" %>

Важно – нет пустых строк до этой директивы !

# Page directive: примеры

<%@ page import = "java.sql.*" %>

Если нужно импортировать несколько пакетов – то разделяются запятой

<%@ page import = "java.sql.*,java.util.*"  %>

По умолчанию импортируются

java.lang.*

javax.servlet.*

javax.servlet.jsp.*

javax.servlet.http.*.

# Include directive

• <%@ include file = "relative url" >

Добавляет (вставляет) содержимое файла на JSP страницу
Может быть jsp страница, может быть только язык разметки

Компилируется уже результирующая страница
Удобно для разработки, может затруднять поиск ошибок

# JSP теги

• jsp:include

Динамическое включение страницы

• jsp:useBean

Создание / переиспользование экземпляра

• jsp:setProperty

Установка значения свойства в JavaBean классе

• jsp:getProperty

Чтение свойства в JavaBean классе

# JSP теги

• jsp:forward

Переадресация запроса

• jsp:element

определение XML тега

• jsp:attribute

•Задание атрибута XML тега

• jsp:body

• задание тела XML тега

•jsp:text

Задание шаблона текста.

# JSP теги: include

• <jsp:include page = "relative URL" flush = "true" />

Код, который включает содержимое в момент выполнения

page – страница (сервлет) для включения

flush – сбрасывать буфер при включении

<% String s=“page.jsp”; %>

…

<jsp:include page=“<%=s%>” />

Используется текущее значение переменной *s*

Реализация RequestDispatcher.include()

# JSP теги: forward

<jsp:forward page = "relative URL" />

Код, который переадресует управление
page – страница (сервлет) для переадресации

<% String s=“page.jsp”; %>
…
<jsp:forward page=“<%=s%>” />
Используется текущее значение переменной *s*

Реализация RequestDispatcher.forward()

# JSP - понимание

-  before
  <% response.sendRedirect(“next.jsp”); %>
   after

-  before
  <% return; %>
   after

-  before
  <jsp:forward page=“next.jsp” />
 after

# JSP: body

```
<jsp:element name = "xmlElement">
        <jsp:attribute name = "xmlElementAttr">
           Value for the attribute
        </jsp:attribute>

       <jsp:body>
           Body for XML element
        </jsp:body>

 </jsp:element>

<xmlElement xmlElementAttr = "Value for the attribute">
        Body for XML element
  </xmlElement>
```

# JSP body

```
<jsp:element name = “<%=sName%>">
        <jsp:attribute name = “<%=sAttr%>">
          <%=sValue%>
        </jsp:attribute>

       <jsp:body>
          Body for XML element
        </jsp:body>

</jsp:element>
```

# JSP useBean

<jsp:useBean id="имя" class="class"  scope = " page | request | session | application"/>

page – объект на странице
request – объект в запросе
session - объект в сессии
application – объект в ServletContext

<jsp:useBean id=“MyObj” class = “com.mycompany.A”/>

com.mycompany.A MyObj = new com.mycompany.A();
pageContext.getAttribute(“MyObj”, MyObj, scope);

Или

com.mycompany.A MyObj
=(com.mycompany.A)pageContext.getAttribute(“MyObj”, scope);

# JSP useBean

<jsp:useBean id=“Cart" class=“MyCart"  scope = “session"/>

<jsp:setProperty name = “Cart” property=“qty” value=“5”/>

Эквивалент:

Cart.setQty(“5”);
И сохранение объекта Cart в сессии

Чтение:
<jsp:getProperty name=“Cart” property=“qty”/>

Эквивалентно:

<%=Cart.getQty()%>

# JSP setProperty

```
<jsp:setProperty name="instanceOfBean" property= "*"  />
```

Устанавливает значения свойств в соответствии с именами параметров

```
<jsp:setProperty name="instanceOfBean"
    property="propertyName" param="parameterName"  />
```

Устанавливает значение свойства в соответствии со значением параметра

# JSP setProperty

```
<form method=“post’ action=“proc.jsp”>
  <input type=“text” name=“user” size=“20”/>
  <input type=“submit” value =“Ok” />
</form>

proc.jsp :

<jsp:useBean id=“MyInput” class=“com.my.MyData”/>

<jsp:setProperty name=“MyInput"
    property=“userName" param=“user"  />
                              ||
                              \/
com.my.MyData MyInput = new com.my.MyData();
MyInput.setUserName(request.getParameter(”user”));
```

# JSP setProperty

```
package com.my;

public class myInput
{
  private String user;
  private String role;

  public void setUser (String user)   {
     this.user = user;
  }
  public String getUser () { return this.user; }

  public void setRole(String role)  {
     this.role = role;
  }
 public String getRole() { return this.role; }

}
```

# JSP: setProperty

```
<form method=“post’ action=“proc.jsp”>
  <input type=“text” name=“user” size=“20”/>
  <input type=“text” name=“role” size=“20”/>
  <input type=“submit” value =“Ok” />
</form>
```

proc.jsp :

```
<jsp:useBean id=“MyInput” class=“com.my.MyData”/>

<jsp:setProperty name=“MyInput“ property=“*"  />
                              ||
                              \/
com.my.MyData MyInput = new com.my.MyData();
MyInput.setUser (request.getParameter(”user”));
MyInput.setRole(request.getParameter(”role”));
```

# JSP: pageContext

pageContext.setAttribute(“key",value,PageContext.PAGE_SCOPE);
…
Object v = pageContext.getAttribyte(“key”, ,PageContext.PAGE_SCOPE);

pageContext.setAttribute(“key",value,PageContext.SESSION_SCOPE);
session.setAttrbiute(“key”, value);

pageContext.setAttribute(“key",value,PageContext.REQUEST_SCOPE);
request.setAttrbiute(“key”, value);

pageContext.setAttribute(“key",value,PageContext.APPLICATION_SCOPE);
application.setAttrbiute(“key”, value);

# JSP: EL

• The Expression Language (EL) – другая форма представления (записи) доступа  к классам Java Bean и объектам request, session, application

• Новые предопределенные объекты, операции, зарезервированные слова

• Добавлено в JSP version 2.0.

• Форма представления: ${ expression }

• Аналог: <%= expression%>

• параметр страницы: разрешить или запретить *isELIgnored*

# JSP:EL зарезервированные объекты

- pageScope атрибуты pageContext с областью PAGE_SCOPE
- requestScope атрибуты pageContext с областью REGUEST_SCOPE
- sessionScope атрибуты pageContext с областью SESSION_SCOPE
- applicationScope атрибуты pageContext с областью APPLICATON_SCOPE

- param параметр запроса
- paramValues массив значений параметров

- headerValues массив значений заголовков
- cookie куки

- initParam начальные параметры

- pageContext объект pageContext.

# JSP EL примеры

```
<form action="process.jsp">
  Enter Name:<input type="text" name="guest" /><br/><br/>
  <input type="submit" value="go"/>
  </form>
```

process.jsp:

```
  Значение параметра ${ param.guest }
```

# JSP EL примеры

```
<h3>welcome to index page</h3>
    <%
    session.setAttribute("user","sonoo");
    %>

 <a href="process.jsp">visit</a>
```

process.jsp

```
<p>Value is ${ sessionScope.user }
```

<p> или

```
<p>Value is <%=session.getAttribute(“user”)%>
```

# JSP EL примеры

Index.jsp

```
<h1>Main page</h1>
<%
Cookie ck=new Cookie("name",“guest");
response.addCookie(ck);
%>
<a href="process.jsp">click</a>
```

process.jsp

```
Hello, ${cookie.name.value}
```

# JSP EL примеры

Операторы и приоритеты:

[] .
()
-(unary) not ! empty
* / div % mod
+ - (binary)
< <= > >= lt le gt ge
== != eq ne
&& and
|| or
?:

# JSP EL зарезервировано

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

# JSP EL примеры

${initParam.AppID + 200}

${initParam.AppID < 200}

${header["Accept-Encoding"]}

${sessionScope.employee.address.zipCode}

${1<2}
${1+2+3}

${empty param.username}

# Servlets security

```
<web-app>
  ...
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>SecuredBookSite</web-resource-name>
      <url-pattern>/secured/*</url-pattern>
      <http-method>GET</http-method>
      <http-method>POST</http-method>
    </web-resource-collection>

    <auth-constraint>
      <description>Let only managers use this app</description>
      <role-name>manager</role-name>
    </auth-constraint>
  </security-constraint>

  <security-role> <role-name>manager</role-name> </security-role>

  <login-config> <auth-method>BASIC</auth-method> </login-config>
  ...
</web-app>
```

# Servlets security

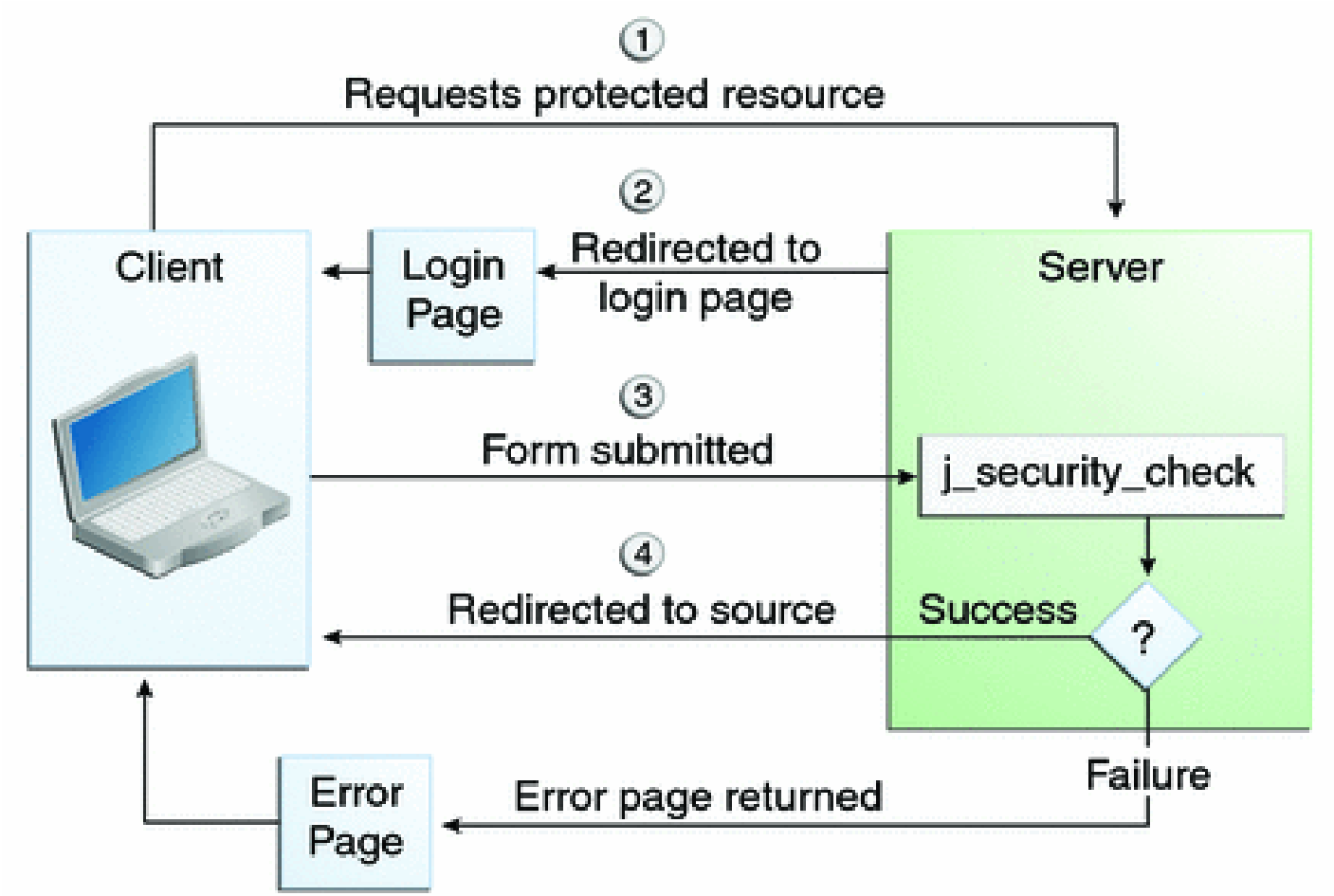
```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>wholesale</web-resource-name>
        <url-pattern>/company/wholesale/*</url-pattern>
    </web-resource-collection>

    <auth-constraint>
        <role-name>PARTNER</role-name>
    </auth-constraint>

    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>

</security-constraint>
```

# Servlets security



# Servlets security

```
<form method="POST" action="j_security_check">
<input type="text" name="j_username">
<input type="password" name="j_password">
</form>
```

```
<login-config>
    <auth-method>FORM</auth-method>
    <realm-name>file</realm-name>
    <form-login-config>
        <form-login-page>/login.xhtml</form-login-page>
        <form-error-page>/error.xhtml</form-error-page>
    </form-login-config>
</login-config>
```

# Servlets security

tomcat-users.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<tomcat-users>
   <role rolename="Admin"/>
   <user username="Jane" password="password" roles="Admin"/>
   <role rolename="manager"/>
   <user username="tomcat" password="tomcat"
      roles="manager"/>
</tomcat-users>
```

# Servlets security

Authenticating Users Programmatically

Методы HttpServletRequest для программной аутентификации:

authenticate  диалог для ввода имени/пароля

login – альтернатива для form-based authentication

logout – сброс идентификации

# Servlets security

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            out.println("<html>");
            out.println("<body>");
                        request.login(“guest", “12345");
            out.println("<h1>Ok</h1>");
            out.println("</body>");
            out.println("</html>");
        } catch (Exception e) {
            throw new ServletException(e);
        } finally {
            request.logout();
            out.close();
        }     }
```

# Servlets security

```
public class TestServlet extends HttpServlet {

    protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
                        request.authenticate(response);
            out.println("Authenticate Successful");
        } finally {
            out.close();
        }
    }
```

# Servlets security

request.getRemoteUser()
Returns the login of the user making this request, if the user has been authenticated, or **null** if the user has not been authenticated.

request.login(user, pwd) – устанавливает значение
request.logout()  - сбрасывает это значение в null

boolean isUserInRole(java.lang.String role) – проверка роли пользователя