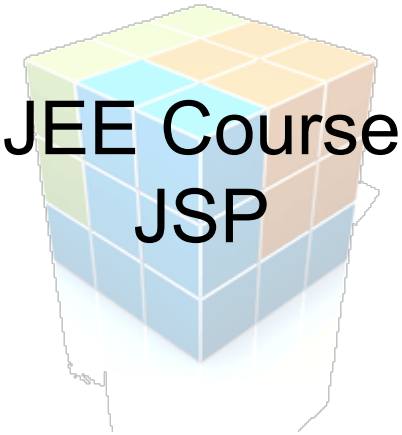


JEE Course
JSP

Márcio Fuckner

PUCPR - Pontifical Catholic University of Parana

ESIGELEC – Graduate School of Engineering

Content

- JSP definition
- JSP lifecycle
- JSP elements
- Servlet scopes
- *Forward* action
- Executing JSP and Servlets

Servlets/JSP timeline

Date	JEE	Servlets	JSP	Features
Dez/1999	1.2	2.2	1.1	
Set/2001	1.3	2.3	1.2	Security, EJB specification
Nov/2003	1.4	2.4	2.0	Web services (JAX-RPC), important changes in the JSP specification
Maio/2006	5	2.5	2.1	EJB 3, JPA, Java 5 and annotations, JSF
2010	6	3	2.2	Assincronous I/O for servlets, resource injection for servlets
2013	7	3.1	2.3	Minor changes

What is JSP ?

- JSP (*Java Server Pages*) is a Java specification and API based on **templates** to generate servlet code
- Emerged in 1998 as a “user-friendly” alternative for JEE interface development, if compared to the Servlet API
- In a JEE server, JSP processing is managed by an additional layer, which transforms the page into a servlet
- In other words, the entire page becomes a Java source code, compiled and maintained by the server

A warm-up example

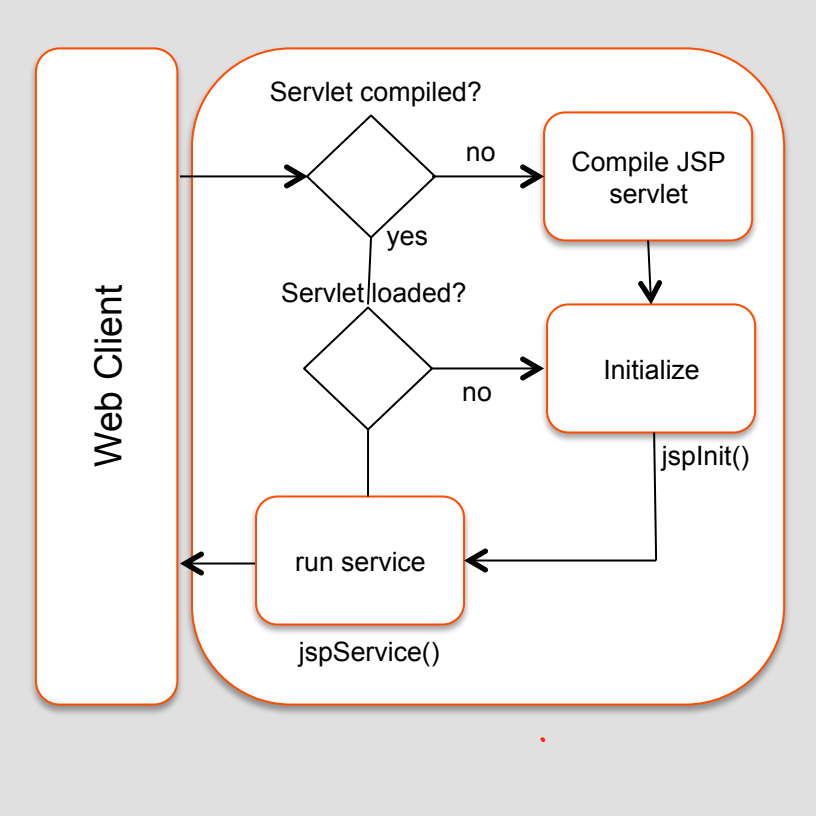
- A JSP page is a text document that contains two types of text: static data and JSP elements
- The recommended file extension for the source file of a JSP page is *.jsp*
- To run a JSP page, place the file into the web folder of a JEE project
- The compilation is made during the first access
- The request is redirected to the generated servlet

```
<p>Hello</p>
<% for(int i=0;i<10;i++){ %>
  <p>line <%=i %></p>
<% }%>
```


Example: module06/example01.jsp

JSP lifecycle

- When a request is mapped to a JSP file, the container:
  - Check if there is a corresponding servlet to attend the request
  - If there is no servlet or the servlet version is older than the JSPfile , then a new servlet is generated
  - From this point, the lifecycle is equivalent to the one found in the servlet specification
  - *jspInit* to load the JSP Servlet
  - *jspService* to attend requests
  - *jspDestroy* to destroy the instance



Example: module06/example02.jsp

JSP elements

- JSP elements are server-side fragments mixed with markup code (HTML, XHTML, XML)
- All JSP elements are interpreted in the server side (they never reach the client)
- Types of JSP elements

Directives	<%@	…	%>
Declarations	<%!	…	%>
Expressions	<%=	…	%>
Scriptlets	<%	…	%>
Actions	<jsp:action	…	/>
Custom tags	<prefix:element	…	/>

Directives <%@ … %>

- Allows modifying the behavior of the JSP servlet generator:
- Syntax:

  <%@ directive attr=value attr=value ...%>

- Most important directives:
  - *page*: page settings
  - *include*: includes (statically other files on the page)
  - *taglib*: add custom tags

Page Directive

Some page directive attributes

info=”JSP description”	default: none
language="java“	(default)
contentType="text/html; charset=<charset>“	(default)
extends="acme.SourceJsp“	default: none
import="java.io.*, java.net.*”	default: java.lang
session="true“	(default)
buffer="8kb“	(default)
autoFlush="true“	(default)
isThreadSafe="true“	(default)
errorPage="/erros/404.jsp"	default: none
isErrorPage= "false“	(default)

Example: module06/example03.jsp

Declarations <%! … %>

- Give access to the body of the servlet class.
- Allows the declaration of variables and methods in a page
- Useful for:
  - Declaring instance variables and methods
  - Declaring static variables and methods
  - Declaring inner classes (both static and instance)
  - Declaring static blocks
  - Overriding JSP methods (i.e.: *jspInit*)

Example: module06/example04.jsp

Expressions <%= … %>

- Expressions
  - Use the return value of the expression and writes it using the response stream
  - Equivalent to *out.print (expression)*, so can not end with a semicolon
  - All values resulting from expressions are converted to String before being redirected to the standard output
  - Syntax:

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


Example: module06/example05.jsp

Scriptlets <% … %>

- Expressions
  - This java code snippet will be executed during the service execution.
  - This block has access to all declared methods and implicit objects (we will see them soon)
  - Syntax:

```
<% general java code %>
```


Example: module06/example06.jsp

JSP actions

- Predefined set of actions
- They use the jsp namespace
- List of JSP actions:
  - **<jsp:include>**
    - **Includes dinamically a page**
  - <jsp:forward>
    - Forward the request and response to another resource
  - <jsp:useBean> <jsp:getProperty> and <jsp:setProperty>
    - Bean management

Example: module06/example06.jsp

Implicit objects

- Predefined variables available for expression and scriptlets

Name	Description	Availability
config	Retrieves JSP configuration such as initialization params	Always
Page	Reference to the generated servlet. Equivalent to « this »	Always
request	Request object	Always
response	Response object	Always
out	PrintWriter object	Always
session	HttpSession object	when page session="true“
Application	Application object	Always
pageContext	Page context object	Always
exception	Exception object	When page isErrorPage=true

out object

- A reference to a *PrintWriter*
- Can be used to generate textual content

Example: module06/example07.jsp

request object

- A known instance of the *HttpServletRequest* object
- Can be used to:
  - Get query parameters
  - Get headers
  - Get request information
  - Get request body

Example: module06/example08.jsp

session object

- An known instance of the *HttpSession* object
- Can be used to
  - Add attributes
  - Get attributes
  - Remove attributes
  - Manage session lifecycle

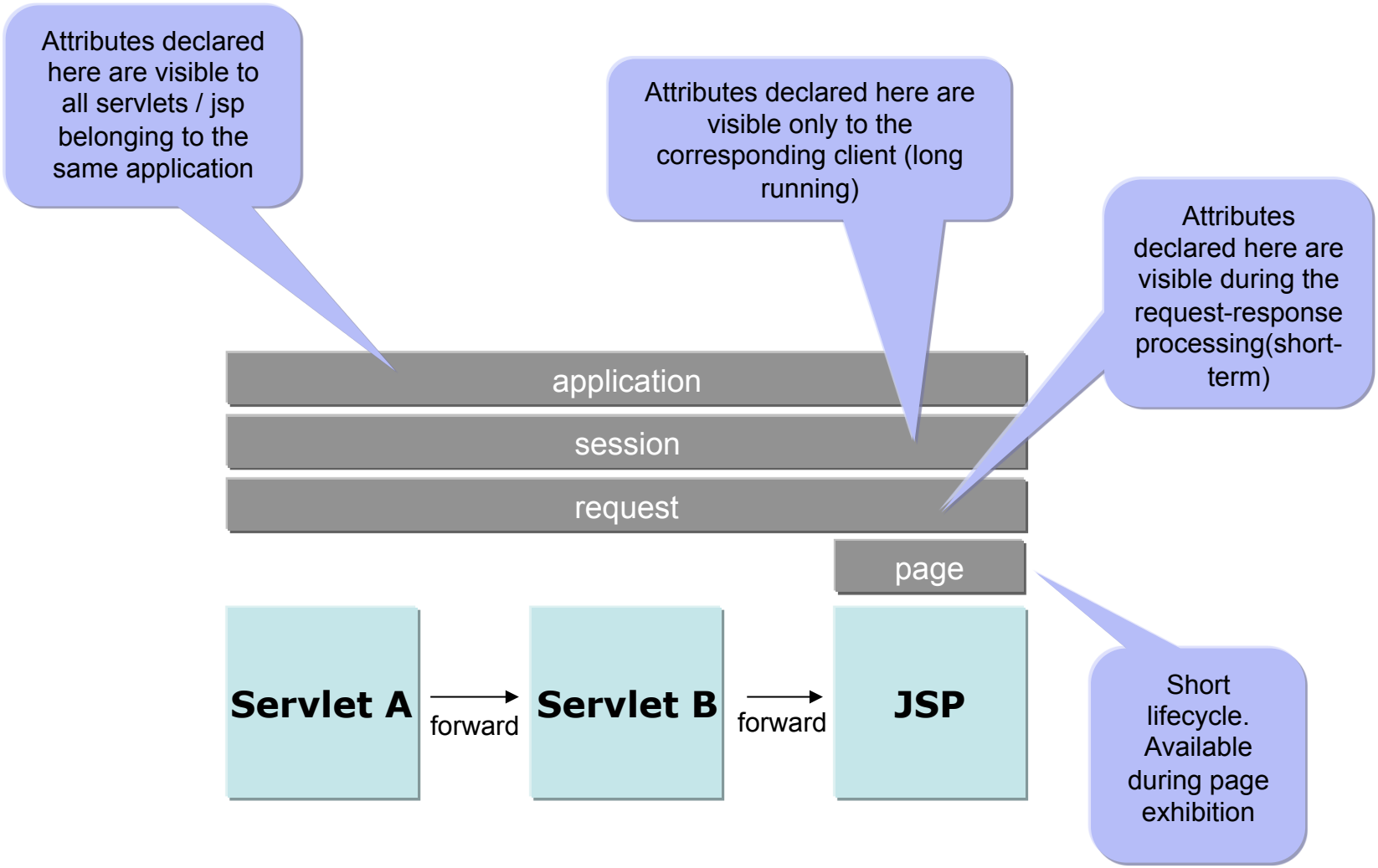
Example: module06/example09.jsp

application object

- An instance representing the application state.
- An instance of *ServletContext* object
- Can be used to:
  - Manage application elements (new servlets, new filters, etc)
  - Manage common attributes

Example: module06/example10.jsp

Servlet scopes



JSP *forward* action

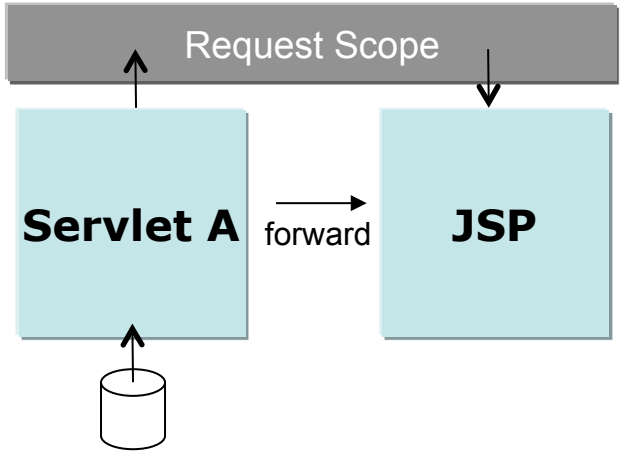
- Forwards the request and response to another URI
- Syntax

```
<jsp:forward page=<pageName>/>

or

<jsp:forward page=<pageName>>
    <jsp:param name=<paramName> value=<paramValue>/>
    …
</jsp:forward>
```


Example: module06/example11.jsp

Using JSP and Servlets

- MVC Web frameworks such as JSF, Spring and others enforce a clear separation between layers.
- Servlets are candidate to perform service-based tasks, such as connect databases, execute other services, connect legacy applications
- JSP and JSF are candidates to perform user-interface tasks (rendering content, input controls and basic event-handling)
- Example12 gives a basic idea of this separation
- In the second week, we will explore this separation using patterns like Front Controller to clearly separate business, control and interface layers



Example: module06/example12.jsp

Further reading

JSP Technology

http://www.oracle.com/technetwork/java/javaee/jsp/index.html