

Java Server Faces Pages et Composants

Java EE 6 - Sommaire

- Introduction
- Pages web
 - HTML
 - XHTML
 - CSS
 - DOM
 - JavaScript
- JSP
- JSTL

Java EE 6 - Sommaire

- Facelets
- JSF 2.0

Introduction

- Avec le Web 2.0 et les Rich Internet Applications (RIA), les utilisateurs demandent des interfaces de plus en plus réactives et sophistiquées
- Aux débuts de Java, il fallait construire ses pages à la main depuis des Servlets. Java EE 6 propose JSF pour simplifier le développement web
- Dans ce chapitre nous verrons les différentes technologies de JSF pour créer des pages web

Introduction

- JSF héritant du passé, on peut encore utiliser les précédentes technologies comme JSP
- Nous nous concentrerons sur les nouvelles : Facelets, Expression Language et JSTL
- Nous verrons également comment créer ses propres composants JSF

Pages web

- Une page doit permettre d'afficher du contenu dynamique : le contenu d'un catalogue (CDs ou livres par exemple), les informations sur un client, le contenu d'un panier d'achats, ...
- Une page doit aussi permettre d'afficher des contenus qui changent rarement : une page d'information, les conditions de vente, des images, vidéos, ...
- Le but est d'afficher une page dans un navigateur, donc doit utiliser un des langages compris : HTML, XHTML, CSS et JavaScript

HTML

- Hypertext Markup Language est le langage des pages web
- Il est basé sur Standard Generalized Markup Language (SGML), métalangage pour définir des langages à base de balises
- Les document HTML ont généralement l'extension .html ou .htm

HTML

```
<H1>Create a new book</h1>
<hr>
<TABLE border=0>
  <TR>
    <TD>ISBN :</TD>
    <TD><input type=text/></td>
  </tr>
  <tr>
    <td>Title :</td>
    <TD><input type=text/></td>
  </tr>
  <tr>
    <td>Price :</td>
    <TD><input type=text/></td>
  </tr>
  <tr>
    <td>Description :
    <td><textarea name=textarea cols=20 rows=5></textarea>
  </tr>
  <TR>
    <TD>Number of pages :
    <td><input type=text/>
  </tr>
  <tr>
    <td>Illustrations :
    <td><input type=checkbox/>
  </tr>
</table>
<input type=submit value=Create>
<hr>
<i>APress - Beginning Java EE 6</i>
```

Create a new book

ISBN :	<input type="text"/>
Title :	<input type="text"/>
Price :	<input type="text"/>
Description :	<input type="text"/>
Number of pages :	<input type="text"/>
Illustrations :	<input type="checkbox"/>
<input type="button" value="Create"/>	
<i>APress - Beginning Java EE 6</i>	

XHTML

- XHTML a été créé peu après HTML 4.01
- Il s'appuie sur HTML mais dans une structure XML stricte
- Les documents XHTML ont l'extension .xhtml
- L'avantage de XHTML est qu'on peut utiliser les outils standard XML (XSL, XSLT, validation par schéma) et qu'on peut ajouter des balises personnalisées

XHTML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Creates a new book</title>
</head>
<body>
<h1>Create a new book</h1>
<hr/>
<table border="0">
  <tr>
    <td>ISBN :</td>
    <td><input type="text"/></td>
  </tr>
  <tr>
    <td>Title :</td>
    <td><input type="text"/></td>
  </tr>
  <tr>
    <td>Price :</td>
    <td><input type="text"/></td>
  </tr>
```

```

  <tr>
    <td>Description :</td>
    <td><textarea name="textarea" cols="20" rows="5"></textarea></td>
  </tr>
  <tr>
    <td>Number of pages :</td>
    <td><input type="text"/></td>
  </tr>
  <tr>
    <td>Illustrations :</td>
    <td><input type="checkbox"/></td>
  </tr>
</table>
<input name="" type="submit" value="Create"/>
<hr/>
<i>APress - Beginning Java EE 6</i>
</body>
</html>
```

CSS

- Cascading Style Sheet est un langage de style utilisé pour décrire la présentation d'un document écrit en HTML ou XHTML
- CSS est utilisé pour définir les couleurs, fontes, la mise en page et autres aspects liés à la présentation
- CSS permet la séparation entre le contenu d'un document (écrit en XHTML) et sa représentation (écrit en CSS)
- Comme HTTP, HTML et XHTML, la spécification CSS est maintenue par le World Wide Web Consortium (W3C)

CSS

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Creates a new book</title>
  <style type="text/css">
    .title {
      font-family: Arial, Helvetica, sans-serif;
      font-size: 22px;
      color: #000099;
      font-style: italic;
    }
    .row {
      font-family: Arial, Helvetica, sans-serif;
      color: #000000;
      font-style: italic;
    }
  </style>
</head>
<body>
  <h1 class="title">Create a new book</h1>
  <hr/>
  <table border="0">
    <tr>
      <td class="row">ISBN :</td>
      <td><input type="text"/></td>
    </tr>
    ...
```

Create a new book

ISBN :

Title :

Price :

Description :

Number of pages :

Illustrations :

☐

Create

APress - Beginning Java EE 6

DOM

- XHTML étant un document XML, possède une représentation Document Object Model (DOM)
- DOM est une spécification du W3C pour accéder et modifier la structure et le contenu d'un document XML ainsi que pour rechercher
- DOM correspond à la représentation en arbre de la structure d'un document

DOM

- DOM fournit un moyen standard d'interagir avec un document XML. En manipulant le DOM avec JavaScript, on peut donc introduire un côté dynamique aux pages Web
- Ajax est basé sur JavaScript pour interagir avec la représentation DOM d'une page web



JavaScript

- JavaScript permet de rendre dynamique l'affichage de contenus côté utilisateur
- On s'en sert notamment pour diminuer le travail du serveur en faisant des contrôles de saisie ou pour créer des composants dynamiques
- JavaScript est un langage de script utilisé principalement pour le développement web côté client
- Malgré son nom, il n'a pas de rapport avec le langage Java

JavaScript

- C'est un langage interprété et faiblement typé
- JavaScript est un langage très puissant pour écrire des applications dynamiques qui interagissent avec le DOM d'une page
- Le European Computer Manufacturers Association (ECMA) s'occupe des spécifications de JavaScript sous le nom ECMAScript

JavaScript

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
  PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>Creates a new book</title>
  <script type="text/javascript">
    function priceRequired() {
      if (document.getElementById("price").value == "") {
        document.getElementById("priceError").innerHTML = ➡
          "Please, fill the price !";
      }
    }
  </script>
</head>
<body>
<h1>Create a new book</h1>
<hr/>
<table border="0">
  <tr>
    <td>ISBN :</td>
    <td><input type="text"/></td>
  </tr>
  <tr>
    <td>Title :</td>
    <td><input type="text"/></td>
  </tr>
```

```
<tr>
  <td>Price :</td>
  <td><input id="price" type="text" ➡
    onblur="javascript:priceRequired()"/>
    <span id="priceError"/>
  </td>
</tr>
<tr>
  <td>Description :</td>
  <td><textarea name="textarea" cols="20" rows="5"></textarea></td>
</tr>
<tr>
  <td>Number of pages :</td>
  <td><input type="text"/></td>
</tr>
<tr>
  <td>Illustrations :</td>
  <td><input type="checkbox"/></td>
</tr>
</table>
<input name="" type="submit" value="Create"/>
<hr/>
<i>APress - Beginning Java EE 6</i>
</body>
</html>
```

JSP

- Java Server Pages ont été introduites avec J2EE 1.2 en 1999 pour afficher des pages dynamiques
- Les pages JSP sont des pages HTML ou XHTML avec des balises spéciales contenant du code Java exécuté côté serveur
- Les pages JSP sont en fait compilées automatiquement en Servlets
- La spécification JSP définit 3 types d'éléments : directive elements, scripting elements et action elements

JSP

- Les directive elements fournissent des informations et ne produisent pas d'affichage :
 - page
 - include
 - taglib
- La syntaxe est la suivante :
 - <% **directive** attributes %>
 - <**jsp:directive** attributes />
- Exemples :

```
<%@ page contentType=" text/html; ISO-8859-1" language="java" %>
<jsp:directive.page contentType="text/html; ISO-8859-1" language="java"/>
<%@ include file="header.jsp"%>
<jsp:directive.include file="header.jsp" />
<%@ taglib uri="http://java.sun.com/jstl/core" prefix="c" %>
<jsp:root xmlns:c="http://java.sun.com/jstl/core">
```

JSP

- Les scripting elements permettent de manipuler les objets
- Il y a 3 catégories de scripting elements :
 - déclarations
 - scriptlets
 - expressions

```
<%! this is a declaration %>  
<jsp:declaration>this is a declaration</jsp:declaration>
```

```
<% this is a scriptlet %>  
<jsp:scriptlet>this is a scriptlet</jsp:scriptlet>
```

```
<%= this is an expression %>  
<jsp:expression>this is an expression</jsp:expression>
```

■ Exemples de scripting elements

```
<%! ArrayList books = new ArrayList(); %>
```

```
<jsp:declaration>ArrayList books = new ArrayList();</jsp:declaration>
```

```
<% books.add(new Book("H2G2", 12f, "Scifi IT book", "1234-234", 241, true)); %>
```

```
<jsp:scriptlet>books.add(new Book("H2G2", 12f, "Scifi IT book", "1234-234", 241, true));</jsp:scriptlet>
```

```
<%= book.getIsbn()%>
```

```
<jsp:expression>book.getIsbn()</jsp:expression>
```

```
<%! ArrayList<Book> books = new ArrayList<Book>(); %>
```

```
<jsp:declaration><![CDATA[  
    ArrayList<Book> books = new ArrayList<Book>();  
]]></jsp:declaration>
```

JSP

- Les action elements sont définis par la spécification JSP
- Ils ressemblent à des balises HTML :
<jsp:useBean>, <jsp:include>, ...

Action	Description
useBean	Associates an instance of an object within a given scope and available with a given ID.
setProperty	Sets the value of a property in a bean.
getProperty	Displays the value of a bean property.
include	Allows the inclusion of static and dynamic resources in the same context as the current page.

JSP

Action	Description
forward	Dispatches the current request to a static resource, a JSP, or a servlet in the same context as the current page.
param	Is used in the include, forward, and params elements. The included or forwarded page will see the original request object with the original parameters augmented with the new parameters.
plugin	Enables a JSP to generate HTML that contains the appropriate browser-dependent constructs (OBJECT or EMBED) that will result in the download of a plug-in.
params	Passes parameters. This is part of the plugin action.
element	Dynamically defines the value of the tag of an XML element.
attribute	Defines an XML attribute. This is part of the element action.
body	Defines the body of an XML element. This is part of the element action.

JSP

- Exemple de JSP :
 - On souhaite afficher la liste des livres

Lists all the books

ISBN	Title	Price	Description	Number of pages	Illustrations
1234-234	H2G2	12.0	Scifi IT book	241	true
564-694	Robots	18.5	Best seller	317	true
256-6-56	Dune	23.25	The trilogy	529	true

APress - Beginning Java EE 6

JSP

```
<%@ page import="com.apress.javaee6.chapter11.Book" %>
<%@ page import="java.util.ArrayList" %>
<%!
    ArrayList<Book> books = new ArrayList<Book>();
%>
<html>
<head>
    <title>List all the books</title>
</head>
<body>
<h1>Lists all the books</h1>
<hr/>
<%
books.add(new Book("H2G2", 12f, "Scifi IT book", "1234-234", 241, true));
books.add(new Book("Robots", 18.5f, "Best seller", "564-694", 317, true));
books.add(new Book("Dune", 23.25f, "The trilogy", "256-6-56", 529, true));
%>
<table border="1">
    <tr>
        <td>ISBN</td>
        <td>Title</td>
        <td>Price</td>
        <td>Description</td>
        <td>Number of pages</td>
        <td>Illustrations</td>
    </tr>
    <%
        Book book;
        for (int i = 0; i < books.size(); i++) {
            book = books.get(i);
        %>
```

JSP

- Une page JSP peut rapidement devenir difficile à lire quand on mixe beaucoup de code HTML avec du code Java
- Il n'y a pas de séparation entre la logique métier et la présentation, ce qui rend la page difficile à maintenir
- On mélange deux langages faits pour des catégories d'utilisateurs différents : Java pour les développeurs métier et XHTML/CSS pour les web designers
- Pour résoudre ce problème, les JSPs peuvent utiliser les tag libraries et l'expression language (EL)

Expression Language

- Expression Language fournit une syntaxe plus simple que les scripting elements pour réaliser des actions similaires
- EL permet d'afficher des valeurs de variables ou les valeurs des attributs d'un objet
- EL a aussi un grand nombre d'opérateurs mathématiques, logiques et relationnels
- EL 2.2 permet aussi d'invoquer des méthodes

Expression Language

- La syntaxe est la suivante :

`${expr}`

- Depuis JSP 2.1 on peut aussi utiliser :

`#{expr}`

- Pour les JSP, il n'y a pas de différence entre les deux syntaxes.
- Pour JSF, `${expr}` contient les expressions évaluées une seule fois à la compilation de la JSP, alors que les expressions `#{expr}` sont évaluées au moment de l'utilisation

Expression Language

- Les expression EL permettent d'utiliser la plupart des opérateurs Java :
 - Arithmétiques : +, -, *, /, %
 - Relationnels : ==, !=, <, >, , <=, >=
 - Logiques : &&, ||, !
 - Autres : (), empty, []
- Exemples :

```
{empty book}  
{empty book.isbn}
```

```
{book.isbn}  
{book[isbn]}
```

```
{book.buy}  
{book.buy('EURO')}
```

JSP Standard Tag Library

- Il y a une série de tags appelés JSTL qui permettent de manipuler du contenu dynamique en utilisant des balises XML plutôt que du code Java
- Une tag library et une collection de fonctions qui peuvent être utilisées dans une page JSP ou JSF

Functional Area	URI	Commonly Used Prefix
Core	http://java.sun.com/jsp/jstl/core	c
XML processing	http://java.sun.com/jsp/jstl/xml	x
I18N and formatting	http://java.sun.com/jsp/jstl/fmt	fmt
Database access	http://java.sun.com/jsp/jstl/sql	sql
Functions	http://java.sun.com/jsp/jstl/functions	fn

JSP Standard Tag Library

- Pour utiliser ces library, il faut commencer par l'importer dans la page JSP :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
// or
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jstl/core" version="1.2">

// puis
  <c:set var="upperLimit" value="20"/>
```

JSP Standard Tag Library

■ Core Actions

Action	Description
<c:out>	Evaluates an expression and outputs the result of the evaluation
<c:set>	Sets a value for an object
<c:remove>	Removes a variable
<c:catch>	Catches a java.lang.Throwable thrown by any of its nested actions
<c:if>	Evaluates whether the expression is true
<c:choose>	Provides mutually exclusive conditions
<c:when>	Represents an alternative within a <c:choose> action
<c:otherwise>	Represents the last alternative within a <c:choose> action
<c:forEach>	Repeats the nested body over a collection of objects, or repeats it a fixed number of times
<c:forTokens>	Iterates over tokens, separated by the supplied delimiters
<c:import>	Imports a resource
<c:url>	Encodes a URL
<c:param>	Adds request parameters to a URL
<c:redirect>	Redirects to a specific URL

JSP Standard Tag Library

- Core Actions, exemple :

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
  xmlns:c="http://java.sun.com/jstl/core" version="1.2">
<html>
<body>
<c:set var="upperLimit" value="20"/>
<c:forEach var="i" begin="3" end="{upperLimit - 5}">
  <c:choose>
    <c:when test="{i%2 == 0}">
      <c:out value="{i} is even"/><br/>
    </c:when>
    <c:otherwise>
      <c:out value="{i} is odd"/><br/>
    </c:otherwise>
  </c:choose>
</c:forEach>
</body>
</html>
```

JSP Standard Tag Library

■ Formatting Actions

Action	Description
<code><fmt:message></code>	Internationalizes a JSP by pulling a message from a message bundle
<code><fmt:param></code>	Supplies a parameter for <code><fmt:message></code>
<code><fmt:bundle></code>	Determines the resource bundle
<code><fmt:setLocale></code>	Sets the locale
<code><fmt:requestEncoding></code>	Sets the request's character encoding
<code><fmt:timeZone></code>	Specifies the time zone in which time information is to be formatted
<code><fmt:setTimeZone></code>	Stores the specified time zone on a variable
<code><fmt:formatNumber></code>	Formats a numeric value (number, currency, percentage) in a locale-sensitive manner
<code><fmt:parseNumber></code>	Parses the string representation of numbers, currencies, and percentages
<code><fmt:formatDate></code>	Formats dates and times in a locale-sensitive manner
<code><fmt:parseDate></code>	Parses the string representation of dates and times

JSP Standard Tag Library

- Formatting Actions, exemple :

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt" %>
<html>
<body>
Dates
<c:set var="now" value="<%=new java.util.Date()%>"/>
<fmt:formatDate type="time" value="${now}"/>
<fmt:formatDate type="date" value="${now}"/>
<fmt:formatDate type="both" dateStyle="short" timeStyle="short" ➔
    value="${now}"/>
<fmt:formatDate type="both" dateStyle="long" timeStyle="long" ➔
    value="${now}"/>
Currency
<fmt:setLocale value="en_us"/>
<fmt:formatNumber value="20.50" type="currency"/>
<fmt:setLocale value="en_gb"/>
<fmt:formatNumber value="20.50" type="currency"/>
</body>
</html>
```

```
Dates
11:31:12
14 may 2009
14/02/09 11:31
14 may 2009 11:31:12 CET
Currency
$20.50
£20.50
```

JSP Standard Tag Library

■ SQL Actions

Action	Description
<code><sql:query></code>	Queries a database
<code><sql:update></code>	Executes a SQL INSERT, UPDATE, or DELETE statement
<code><sql:transaction></code>	Establishes a transaction context for <code><sql:query></code> and <code><sql:update></code> subtags
<code><sql:setDataSource></code>	Sets the datasource
<code><sql:param></code>	Sets the values of parameter markers (?) in a SQL statement
<code><sql:dateParam></code>	Sets the values of parameter markers (?) in a SQL statement for values of type <code>java.util.Date</code>

JSP Standard Tag Library

■ SQL Actions, exemple :

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
<html>
<head>
  <title>Lists all the books</title>
</head>
<body>
<h1>Lists all the books</h1>
<hr/>
  <sql:setDataSource dataSource="jdbc/___default"/>
  <sql:query var="books">
    select * from book
  </sql:query>
  <table border="1">
    <tr>
      <th>ISBN</th>
      <th>Title</th>
      <th>Price</th>
      <th>Description</th>
      <th>Number of pages</th>
      <th>Illustrations</th>
    </tr>
```

```
<c:forEach var="row" items="${books.rows}">
  <tr>
    <td><c:out value="${row.isbn}"/></td>
    <td><c:out value="${row.title}"/></td>
    <td><c:out value="${row.price}"/></td>
    <td><c:out value="${row.description}"/></td>
    <td><c:out value="${row.nbOfPage}"/></td>
    <td><c:out value="${row.illustrations}"/></td>
  </tr>
</c:forEach>
</table>
<hr/>
<i>APress - Beginning Java EE 6</i>
</body>
</html>
```

JSP Standard Tag Library

■ XML Actions

Action	Description
<code><x:parse></code>	Parses an XML document.
<code><x:out></code>	Evaluates an XPath expression and outputs the result.
<code><x:set></code>	Evaluates an XPath expression and stores the result into a variable.
<code><x:if></code>	Evaluates the XPath expression if the expression evaluates to true.
<code><x:choose></code>	Provides mutually exclusive conditions.
<code><x:when></code>	Represents an alternative within an <code><x:choose></code> action.
<code><x:otherwise></code>	Represents the last alternative within an <code><x:choose></code> action.
<code><x:forEach></code>	Evaluates an XPath expression and repeats its content over the result.
<code><x:transform></code>	Applies an XSLT style sheet transformation to an XML document.
<code><x:param></code>	Set transformation parameters. Nested action of <code><x:transform></code> .

JSP Standard Tag Library

■ XML Actions, exemple :

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
<html>
<body>
<table border="1">
  <tr>
    <th>ISBN</th>
    <th>Title</th>
    <th>Price</th>
    <th>Description</th>
    <th>Number of pages</th>
    <th>Illustrations</th>
  </tr>
  <c:import url="books.xml" var="bookUrl"/>
  <x:parse xml="${bookUrl}" var="doc"/>
  <x:forEach var="b" select="$doc/books/book">
    <tr>
      <td><x:out select="$b/@isbn"/></td>
      <td><x:out select="$b/title"/></td>
      <td><x:out select="$b/@price"/></td>
      <td><x:out select="$b/description"/></td>
      <td><x:out select="$b/@nbOfPage"/></td>
      <td><x:out select="$b/@illustrations"/></td>
    </tr>
  </x:forEach>
</table>
</body>
</html>
```

```
<?xml version='1.0' encoding='UTF-8'?>
<books>
  <book isbn='1234-234' price='12' nbOfPage='241' illustrations='true'>
    <title>H2G2</title>
    <description>Scifi IT book</description>
  </book>
  <book isbn='564-694' price='18.5' nbOfPage='317' illustrations='true'>
    <title>Robots</title>
    <description>Best seller</description>
  </book>
  <book isbn='256-6-56' price='23.25' nbOfPage='529' ➡
    illustrations='false'>
    <title>Dune</title>
    <description>The trilogy</description>
  </book>
</books>
```

JSP Standard Tag Library

- Des fonctions qui ne sont pas des tags sont définies dans la spécification de JSTL
- Elles peuvent être utilisées dans les expressions EL et servent pour la plupart à manipuler les chaînes de caractères

JSP Standard Tag Library

Function	Description
fn:contains	Tests whether a string contains the specified substring
fn:containsIgnoreCase	Tests whether a string contains the specified substring in a case-insensitive way
fn:endsWith	Tests whether a string ends with the specified suffix
fn:escapeXml	Escapes characters that could be interpreted as XML markup
fn:indexOf	Returns the index within a string of the first occurrence of a specified substring
fn:join	Joins all elements of an array into a string
fn:length	Returns the number of items in a collection or the number of characters in a string
fn:replace	Returns a string resulting from replacing an input string with all occurrences of a substring
fn:split	Splits a string into an array of substrings
fn:startsWith	Tests whether a string starts with the specified prefix
fn:substring	Returns a subset of a string
fn:substringAfter	Returns a subset of a string following a specific substring
fn:substringBefore	Returns a subset of a string before a specific substring
fn:toLowerCase	Converts all of the characters of a string to lowercase
fn:toUpperCase	Converts all of the characters of a string to uppercase
fn:trim	Removes white space from both ends of a string

JSP Standard Tag Library

- Fonctions, exemple :

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fn" prefix="fn" %>
<html>
<body>
  <c:out value="{fn:toLowerCase('AbcEfGH')}}" />
  <c:if test="{fn:length('H2G2')} == 4">
    H2G2 is four characters long
  </c:if>
</body>
</html>
```

Facelets

- Quand JSF a été créé, l'idée était de réutiliser JSP comme PDL (Page Definition Language) puisque JSP faisait déjà partie de Java EE
- Puisque les JSP utilisaient les technologies EL et JSTL, l'idée était aussi de réutiliser ces technologies dans JSF
- JSP est un langage de pages, alors que JSF est une couche au dessus orientée composant
- Cependant, les cycles de vie des pages JSP et de JSF ne correspondent pas

Facelets

- Les balises de JSP sont traitées une seule fois, de haut en bas, pour produire une réponse
- JSF a un cycle de vie plus complexe où la génération de l'arbre de composants et le rendu se produisent à des étapes différentes
- C'est là qu'intervient Facelets : pour coller au cycle de vie de JSF
- Facelets est une alternative Open Source (Fondation Apache) à JSP

Facelets

- Contrairement à JSP, EL et JSTL, Facelets n'a pas de JSP et ne fait pas partie de Java EE
- Facelets est un remplaçant à JSP et fournit une alternative basée sur XML (XHTML) pour les pages d'une application JSF
- Facelets a été conçu avec JSF en tête, c'est pourquoi il fournit un modèle de développement plus simple que JSP
- Facelets fournit une librairie de balises pour écrire l'interface et supporte partiellement JSTL

Facelets

- Le point important de Facelets, c'est son système de templates de pages plus flexible que celui de JSP. Il permet de créer des composants personnalisés à utiliser dans l'arbre de composants JSF

Facelets

- La taglib de Facelets est définie à l'URI <http://java.sun.com/jsf/facelets> et a généralement le préfixe ui

Tag	Description
<code><ui:composition></code>	Defines a composition that optionally uses a template. Multiple compositions can use the same template.
<code><ui:component></code>	Creates a component.
<code><ui:debug></code>	Captures debugging information.
<code><ui:define></code>	Defines content that is inserted into a page by a template.
<code><ui:decorate></code>	Allows you to decorate some content in a page.
<code><ui:fragment></code>	Adds a fragment of a page.
<code><ui:include></code>	Encapsulates and reuses content among multiple XHTML pages, similar to JSP's <code><jsp:include></code> .
<code><ui:insert></code>	Inserts content into a template.
<code><ui:param></code>	Passes parameters to an included file (using <code><ui:include></code>) or a template.
<code><ui:repeat></code>	Serves as an alternative to <code><c:forEach></code> .
<code><ui:remove></code>	Removes content from a page.

JavaServer Faces

- Comprendre JSP, JSTL et Facelets est essentiel pour comprendre JSF 2.0 : il faut choisir un PDL avec JSF. JSP était le PDL par défaut jusqu'à JSF 1.2. Avec JSF 2.0, le PDL par défaut est Facelets.
- Cela ne signifie pas qu'on ne peut plus utiliser les JSP avec JSF 2.0, mais les balises et fonctionnalités seront limitées

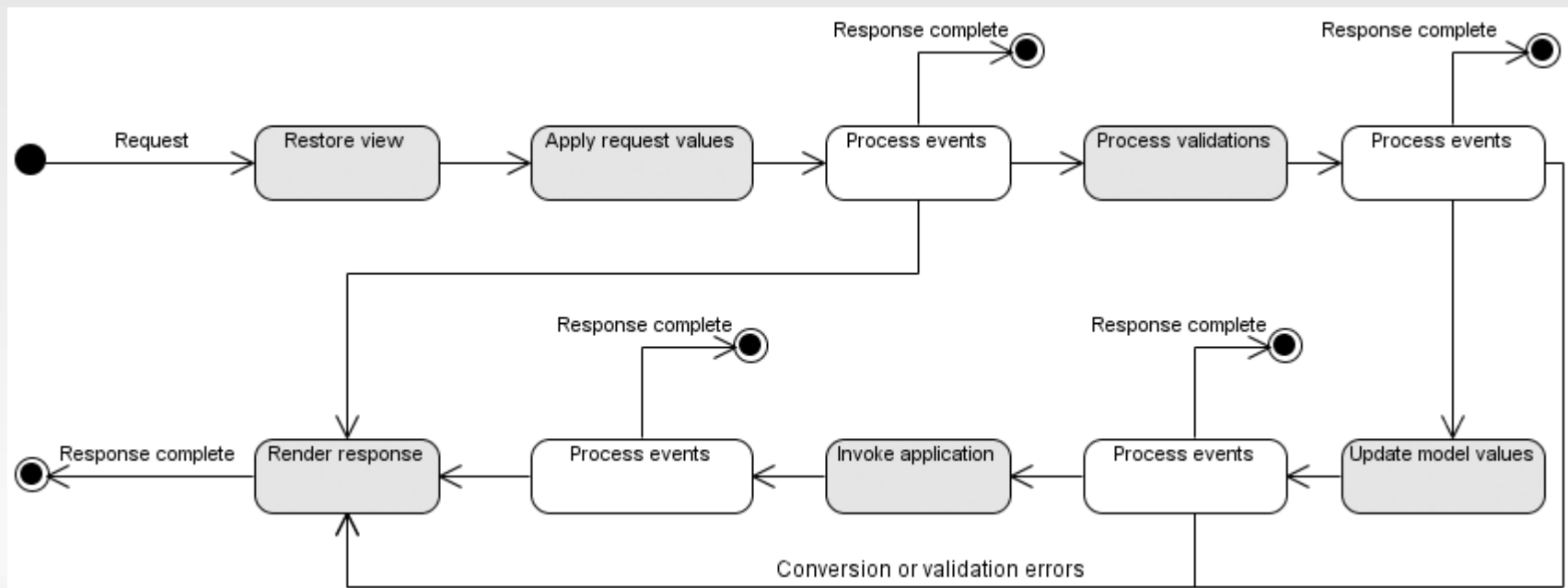
JavaServer Faces

- Taglib utilisables dans une page Facelets

URI	Common Prefix	Description
http://java.sun.com/jsf/html	h	This tag library contains components and their HTML renderers (h:commandButton, h:commandLink, h:inputText, etc.).
http://java.sun.com/jsf/core	f	This library contains custom actions that are independent of any particular rendering (f:selectItem, f:validateLength, f:convertNumber, etc.).
http://java.sun.com/jsf/facelets	ui	Tags in this library add templating support.
http://java.sun.com/jsf/composite	composite	This tag library is used for declaring and defining composite components.
http://java.sun.com/jsp/jstl/core	c	Facelets pages can use some of the core JSP tag libraries (<c:if/>, <c:forEach/>, and <c:catch/>).
http://java.sun.com/jsp/jstl/functions	fn	Facelets pages can use all the function JSP tag libraries.

JavaServer Faces

- Cycle de vie
 - Une page JSF est un arbre de composants avec un cycle de vie spécifique



JavaServer Faces

- Le cycle JSF est divisé en 6 phases distinctes :
 - Restore view : JSF cherche la vue correspondant à la requête. Si c'est la première visite, la vue est instanciée, sinon elle est réutilisée
 - Apply request values : les composants de la page reçoivent les valeurs contenues dans la requête
 - Process validations : JSF parcourt l'arbre de composants et demande à chaque composant de vérifier si la valeur assignée est correcte
 - Update model values : les valeurs des composants servent à mettre à jour les managed beans associés

JavaServer Faces

- Invoke application : la logique métier est exécutée par les managed beans et la navigation est déterminée
- Render response : renvoi de la réponse à l'utilisateur. L'état de la vue est sauvé pour la prochaine phase de "Restore view"

JavaServer Faces

- Composants HTML Standard :
 - JSF est indépendant du protocole et du langage PDL, mais il fournit une implémentation pour HTTP et HTML
 - JSF fournit une bibliothèque de composants standards qui couvrent la plupart des besoins

JavaServer Faces

- Commands : pour déclencher une action

Tag	Description
<code><h:commandButton></code>	Represents an HTML input element for a button of type submit or reset.
<code><h:commandLink></code>	Represents an HTML element for a hyperlink that acts like a submit button. This component must be placed inside a form.

```
<h:commandButton value="A submit button"/>
```

```
<h:commandButton type="reset" value="A reset button"/>
```

```
<h:commandButton image="javaee6.gif" title="A button with an image"/>
```

```
<h:commandLink>A hyperlink</h:commandLink>
```

```
<h:commandLink action="#{bookController.doNew}">
```

Create a new book

```
</h:commandLink>
```

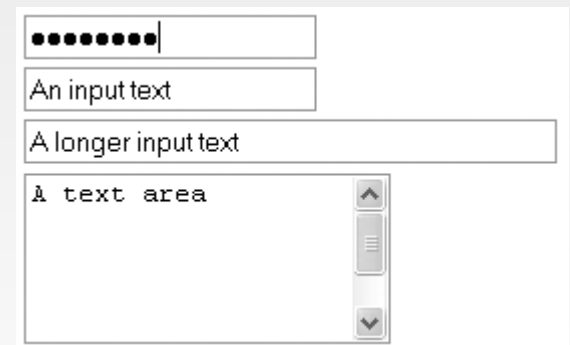


JavaServer Faces

- Inputs : pour l'affichage et la saisie de texte

Tag	Description
<code><h:inputHidden></code>	Represents an HTML input element of type hidden.
<code><h:inputSecret></code>	Represents an HTML input element of type password. On a redisplay, any previously entered value will not be rendered (for security reasons) unless the redisplay property is set to true.
<code><h:inputText></code>	Represents an HTML input element of type text.
<code><h:inputTextarea></code>	Represents an HTML text area element.

```
<h:inputHidden value="Hidden data"/>  
<h:inputSecret maxlength="8"/>  
<h:inputText value="An input text"/>  
<h:inputText size="40" value="A longer input text"/>  
<h:inputTextarea rows="4" cols="20" value="A text area"/>
```



The screenshot displays a vertical stack of four input fields. The top field is a hidden input, shown as a series of dots. The second field is a password input, also shown as dots. The third field is a standard text input containing the text 'An input text'. The bottom field is a text area containing the text 'A text area'.

JavaServer Faces

- Outputs : pour l'affichage simple de texte

Tag	Description
<code><h:outputLabel></code>	Renders an HTML <code><label></code> element
<code><h:outputLink></code>	Render an HTML <code><a></code> anchor element
<code><h:outputText></code>	Outputs text

```
<h:outputLabel value="#{bookController.book.title}"/>  
<h:outputText value="A text"/>  
<h:outputLink value="http://www.apress.com/">A link</h:outputLink>
```



```
<label>The title of the book</label>  
A text  
<a href="http://www.apress.com/">A link</a>
```


JavaServer Faces

- Selections : sélection d'une ou plusieurs valeurs

Tag	Description
<code><h:selectBooleanCheckbox></code>	Renders one check box representing a single Boolean value. The check box will be rendered as checked or not based on the value of the property.
<code><h:selectManyCheckbox></code>	Renders a list of check boxes.
<code><h:selectManyListbox></code>	Renders a multiple-selection component where one or more options can be selected.
<code><h:selectManyMenu></code>	Renders an HTML <code><select></code> element.
<code><h:selectOneListbox></code>	Renders a single-selection component where only one available option can be selected.
<code><h:selectOneMenu></code>	Renders a single-selection component where only one available option can be selected. It only shows a single available option at a time.
<code><h:selectOneRadio></code>	Renders a list of radio buttons.

JavaServer Faces

```
<h:selectOneMenu>
  <f:selectItem itemLabel="History" />
  <f:selectItem itemLabel="Biography"/>
  <f:selectItem itemLabel="Literature"/>
  <f:selectItem itemLabel="Comics"/>
  <f:selectItem itemLabel="Child"/>
  <f:selectItem itemLabel="Scifi"/>
</h:selectOneMenu>
```

Tag	Rendering
h:selectBooleanCheckbox	<input type="checkbox"/>
h:selectManyCheckbox	<input type="checkbox"/> History <input type="checkbox"/> Biography <input type="checkbox"/> Literature <input type="checkbox"/> Comics <input type="checkbox"/> Child <input type="checkbox"/> Scifi
h:selectManyListbox	<div>History Biography Literature Comics Child Scifi</div>
h:selectManyMenu	History
h:selectOneListbox	<div>History Biography Literature Comics Child Scifi</div>
h:selectOneMenu	History
h:selectOneRadio	<input type="radio"/> History <input type="radio"/> Biography <input type="radio"/> Literature <input type="radio"/> Comics <input type="radio"/> Child <input type="radio"/> Scifi

JavaServer Faces

- Graphics : composant pour afficher des images

```
<h:graphicImage value="book.gif" height="200" width="320"/>
```

JavaServer Faces

- Grid and Tables : représentation sous forme de listes

Tag	Description
<code><h:dataTable></code>	Represents a set of repeating data that will be rendered in an HTML <code><table></code> element
<code><h:column></code>	Renders a single column of data within an <code><h:dataTable></code> component
<code><h:panelGrid></code>	Renders an HTML <code><table></code> element
<code><h:panelGroup></code>	Is a container of components that can be embedded in an <code><h:panelGrid></code>

JavaServer Faces

```
<h:panelGrid columns="3" border="1">  
  <h:outputLabel value="One"/>  
  <h:outputLabel value="Two"/>  
  <h:outputLabel value="Three"/>  
  <h:outputLabel value="Four"/>  
  <h:outputLabel value="Five"/>  
  <h:outputLabel value="Six"/>  
</h:panelGrid>
```

One	Two	Three
Four	Five	Six

```
<h:panelGrid columns="3" border="1">  
  <f:facet name="header">  
    <h:outputText value="Header"/>  
  </f:facet>  
  <h:outputLabel value="One"/>  
  <h:outputLabel value="Two"/>  
  <h:outputLabel value="Three"/>  
  <h:outputLabel value="Four"/>  
  <h:outputLabel value="Five"/>  
  <h:outputLabel value="Six"/>  
  <f:facet name="footer">  
    <h:outputText value="Footer"/>  
  </f:facet>  
</h:panelGrid>
```

Header		
One	Two	Three
Four	Five	Six
Footer		

JavaServer Faces

- Error messages : permettent d'afficher des messages à l'utilisateur

Tag	Description
<code><h:message></code>	Renders one error message
<code><h:messages></code>	Renders all the enqueued error messages

```
<h:messages style="color:red"/>
<h:form>
  Enter a title:
  <h:inputText value="#{bookController.title}" required="true"/>
  <h:commandButton action="#{bookController.save}" value="Save"/>
</h:form>
```



Validation Error: Value is required.

Enter a title:

JavaServer Faces

- Miscellaneous : balises sans représentation graphique

Tag	Description
<code><h:body></code>	Renders an HTML <code><body></code> element
<code><h:head></code>	Renders an HTML <code><head></code> element
<code><h:form></code>	Renders an HTML <code><form></code> element
<code><h:outputScript></code>	Renders the markup for a <code><script></code>
<code><h:outputStylesheet></code>	Render the markup for a <code><link></code> element

JavaServer Faces

- Templating : permet de définir une mise en page utilisée par plusieurs pages

Tag	Description
<code><ui:composition></code>	Defines a composition that optionally uses a template. Multiple compositions can use the same template.
<code><ui:define></code>	Defines a set of content to be inserted into a corresponding <code><ui:insert></code> element in a template.
<code><ui:decorate></code>	Allows you to decorate some content in a page.
<code><ui:fragment></code>	Adds a fragment of a page.
<code><ui:insert></code>	Defines an insertion point in a template where content can be inserted using <code><ui:define></code> tags around the actual content.
<code><ui:param></code>	Passes parameters to an included file (using <code><ui:include></code>) or a template.

JavaServer Faces

layout.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xml:lang="en" lang="en">
<head>
  <title><ui:insert name="title">Default title</ui:insert></title>
</head>
<body>
  <h1><ui:insert name="title">Default title</ui:insert></h1>
  <hr/>
  <ui:insert name="content">Default content</ui:insert>
  <hr/>
  <i>APress - Beginning Java EE 6</i>
</body>
</html>
```

JavaServer Faces

newBook.xhtml

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xml:lang="en" lang="en">
<ui:composition template="layout.xhtml">
  <ui:define name="title">Create a new book</ui:define>
  <ui:define name="content">
    <table border="0">
      <tr>
        <td>ISBN :</td>
        <td><input type="text"/></td>
      </tr>
      <tr>
        <td>Title :</td>
        <td><input type="text"/></td>
      </tr>
      <tr>
        <td>Price :</td>
        <td><input type="text"/></td>
      </tr>
      <tr>
        <td>Description :</td>
        <td><textarea name="textarea" cols="20" rows="5">
          </textarea>
        </td>
      </tr>
    </table>
  </ui:define>
</ui:composition>
```

```
<tr>
  <td>Number of pages :</td>
  <td><input type="text"/></td>
</tr>
<tr>
  <td>Illustrations :</td>
  <td><input type="checkbox"/></td>
</tr>
</table>
<input name="" type="submit" value="Create"/>
</ui:define>
</ui:composition>
</html>
```

Create a new book

ISBN :	<input type="text"/>
Title :	<input type="text"/>
Price :	<input type="text"/>
Description :	<input type="text"/>
Number of pages :	<input type="text"/>
Illustrations :	<input type="checkbox"/>
<input type="button" value="Create a book"/>	

JavaServer Faces

- Gestion des ressources
 - La plupart des composants peuvent utiliser des ressources externes : `<h:graphicImage>`, `<h:commandButton>`, `<h:outputScript>`, ...
 - En JSF, les ressources sont des éléments statiques qui peuvent être transmis aux composants pour leur rendu
 - Les précédentes versions de JSF ne fournissaient pas de solution pour la gestion des ressources
 - JSF 2.0 permet même de packager les ressources dans un jar séparé avec une version et une locale

JavaServer Faces

- La ressource packagée peut être placée dans l'application web dans le dossier

`resources/<resourceIdentifier>`

OU

`META-INF/resources/<resourceIdentifier>`

- `<resourceIdentifier>` est de la forme :

`[localePrefix/][libraryName/][libVersion]resourceName[/resourceVersion]`

JavaServer Faces

- Par exemple :

book.gif
en/book.gif
en_us/book.gif
en/myLibrary/book.gif
myLibrary/book.gif
myLibrary/1_0/book.gif
myLibrary/1_0/book.gif/2_3.gif



```
<h:graphicImage value="book.gif" />  
<h:graphicImage value="book.gif" library="myLibrary" />  
<h:graphicImage value="#{resource['book.gif']}" />  
<h:graphicImage value="#{resource['myLibrary:book.gif']}" />
```

JavaServer Faces

- Composants Composite
 - JSF permet de créer et d'intégrer vos propres composants ou des librairies tierces
 - Des balises particulières sont disponibles

JavaServer Faces

Tag	Description
<code><composite:interface></code>	Declares the contract for a component.
<code><composite:implementation></code>	Defines the implementation of a component.
<code><composite:attribute></code>	Declares an attribute that may be given to an instance of the component. There may be zero or many of these inside of the <code><composite:interface></code> section.
<code><composite:facet></code>	Declares that this component supports a facet.
<code><composite:insertFacet></code>	Is used in the <code><composite:implementation></code> section. The inserted facet will be rendered in the component.
<code><composite:insertChildren></code>	Is used in the <code><composite:implementation></code> section. Any child components or template within the component will be inserted into the rendered output.
<code><composite:valueHolder></code>	Declares that the component whose contract is declared by the <code><composite:interface></code> in which this element is nested exposes an implementation of <code>ValueHolder</code> .
<code><composite:editableValueHolder></code>	Declares that the component whose contract is declared by the <code><composite:interface></code> in which this element is nested exposes an implementation of <code>EditableValueHolder</code> .
<code><composite:actionSource></code>	Declares that the component whose contract is declared by the <code><composite:interface></code> in which this element is nested exposes an implementation of <code>ActionSource</code> .

JavaServer Faces

Create a new CD	Create a new book
<p>Title : <input type="text"/></p> <p>Price : <input type="text"/></p> <p>Description : <input type="text"/></p>	<p>Title : <input type="text"/></p> <p>Price : <input type="text"/></p> <p>Description : <input type="text"/></p>
<p>Music company : <input type="text"/></p> <p>Number of CDs : <input type="text"/></p> <p>Total duration : <input type="text"/></p> <p>Gender : <input type="text"/></p> <p><input type="button" value="Create a cd"/></p>	<p>ISBN : <input type="text"/></p> <p>Number of pages : <input type="text"/></p> <p>Illustrations : <input type="checkbox"/></p> <p><input type="button" value="Create a book"/></p>
<p><i>APress - Beginning Java EE 6</i></p>	<p><i>APress - Beginning Java EE 6</i></p>

JavaServer Faces

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:composite="http://java.sun.com/jsf/composite">
<composite:interface>
    <composite:attribute name="item" required="true"/>
    <composite:attribute name="style" required="false"/>
</composite:interface>
<composite:implementation>
    <tr style="#{compositeComponent.attrs.style}">
        <td>Title :</td>
        <td>
            <h:inputText value="#{compositeComponent.attrs.item.title}"/>
        </td>
    </tr>
    <tr style="#{compositeComponent.attrs.style}">
        <td>Price :</td>
        <td>
            <h:inputText value="#{compositeComponent.attrs.item.price}"/>
        </td>
    </tr>
    <tr style="#{compositeComponent.attrs.style}">
        <td>Description :</td>
        <td>
            <h:inputTextarea
                value="#{compositeComponent.attrs.item.description}"
                cols="20" rows="5"/>
        </td>
    </tr>
</composite:implementation>
</html>
```

resources/apress/newItem.xhtml

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://java.sun.com/jsf/html"
    xmlns:ago="http://java.sun.com/jsf/composite/apress">
<h:head>
    <title>Creates a new book</title>
</h:head>
<h:body>
    <h1>Create a new book</h1>
    <hr/>
    <h:form>
        <table border="0">
            <ago:newItem item="#{itemController.book}"/>
            <tr>
                <td><h:outputLabel value="ISBN : "/></td>
                <td><h:inputText value="#{itemController.book.isbn}"/></td>
            </tr>
            <tr>
                <td><h:outputLabel value="Number of pages : "/></td>
                <td><h:inputText value="#{itemController.book.nbOfPage}"/></td>
            </tr>
            <tr>
                <td><h:outputLabel value="Illustrations : "/></td>
                <td><h:selectBooleanCheckbox
                    value="#{itemController.book.illustrations}"/></td>
            </tr>
        </table>
        ...
    </h:form>
</h:body>
```

newBook.xhtml

JavaServer Faces

- Objets implicites
 - Un certain nombre d'objets sont disponibles sans qu'il faille les déclarer
 - Ces objets sont appelés objets implicites
 - On utilise ces objets dans les expressions EL

JavaServer Faces

Implicit Object	Description	Returns
application	Represents the web application environment. Used to get application-level configuration parameters.	Object
applicationScope	Maps application-scoped attribute names to their values.	Map
component	Indicates the current component.	UIComponent
compositeComponent	Indicates the current composite component.	UIComponent
cookie	Specifies a Map containing cookie names (the key) and Cookie objects.	Map
facesContext	Indicates the FacesContext instance of this request.	FacesContext
header	Maps HTTP header names to a single String header value.	Map
headerValues	Maps HTTP header names to a String[] of all values for that header.	Map
initParam	Maps context initialization parameter names to their String parameter values.	Map
param	Maps request parameter names to a single String parameter value.	Map
paramValues	Maps request parameter names to a String[] of all values for that parameter.	Map
request	Represents the HTTP request object.	Object
requestScope	Maps request-scoped attribute names to their values.	Map
resource	Specifies the resource object.	Object
session	Represents the HTTP session object.	Object
sessionScope	Maps session-scoped attribute names to their values.	Map
view	Represents the current view.	UIViewRoot
viewScope	Maps view-scoped attribute names to their values.	Map

JavaServer Faces

- Objets implicites, exemple :

```
<h3>headerValues</h3>
<c:forEach var="parameter" items="#{headerValues}">
  <h:outputText value="#{parameter.key}"/> =
  <c:forEach var="value" items="#{parameter.value}">
    <h:outputText value="#{value}" escape="false"/><br/>
  </c:forEach>
</c:forEach>
```



headerValues

```
host=localhost:8080
user-agent=Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.0.6) Gecko/2009011913 Firefox/3.0.6
accept=text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-language=fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
accept-encoding=gzip,deflate
accept-charset=ISO-8859-1,utf-8;q=0.7,*;q=0.7
keep-alive=300
connection=keep-alive
referer=http://localhost:8080/chapter11-1.0/
cookie=com.sun.faces.extensions.flash.PostbackRequest=1; JSESSIONID=0a5ac1217660ad44e45d7956ec6e;
com.sun.faces.extensions.flash.PostbackRequest=123; JSESSIONID=eedf80c16a1382991d11cd67460a
cache-control=max-age=0
```

Conclusion

- On peut créer des pages web en utilisant des langages statiques tels que HTML, XHTML, CSS ou dynamiques tels que JavaScript ou les technologies serveur
- Avec Java EE 6, il est possible d'utiliser plusieurs spécifications dont JSP, EL, JSTL
- Bien que JSP puisse servir de JSF PDL, il vaut mieux utiliser Facelets
- JSF fournit un ensemble de composants standards et une gestion des ressources avec version et localisation