Coding Bootcamp

Introduction to
Java Server Pages (JSP)

Τι είναι η JSP;

• Η **JSP** είναι μια τεχνολογία η οποία χρησιμοποιείται για την ανάπτυξη (development) ιστοσελίδων (web pages) δυναμικού περιεχομένου.

• Μια σελίδα JSP είναι ένα αρχείο (.jsp) το οποίο περιέχει όλα όσα μπορεί να περιέχει μια σελίδα html (πχ html, CSS, Javascript) μαζί με JSP ετικέτες-στοιχεία (JSP tags-elements) μέσω των οποίων παράγεται το δυναμικό περιεχόμενο.

• Σε μια σελίδα JSP μπορούμε να έχουμε πρόσβαση σε όλο το JAVA API όπως για παράδειγμα το JDBC για πρόσβαση σε βάση δεδομένων.

• Για να εκτελέσουμε σελίδες JSP χρειαζόμαστε ένα web container (ή web Server) το οποίο διαθέτει Servlet & JSP engine (όπως πχ Tomcat).

JSP Processing

Την πρώτη φορά που καλείται (εκτελείται) μια σελίδα **JSP** το web container (Tomcat) ακολουθεί την παρακάτω διαδικασία:

• Όπως σε μια html σελίδα, ο client μέσω του web browser στέλνει ένα http request σε μια jsp σελίδα στον web server

• Το web container (Tomcat) «καταλαβαίνει» ότι του ζήτησε jsp σελίδα και προωθεί το request στο JSP engine.

• Το jsp engine φορτώνει από τον δίσκο την σελίδα JSP και την μετατρέπει σε Servlet. Αυτό γίνεται με το να μεταφέρει όλο τον «στατικό» κώδικα (δλδ html, css και javascript) μέσα σε out.println("….") και όλα τα JSP elements να μετατραπούν σε java κώδικα.

• Το JSP engine κάνει compile (παραγωγή του .class) το παραγόμενο Servlet και προωθεί το request στο Servlet engine για να το εκτελέσει.

• Το Servlet engine φορτώνει το Servlet class και το εκτελεί.

• Ο web servlet Προωθεί το output (http response) στον client.

Την 2η φορά και όσο δηλαδή η σελίδα JSP δεν αλλάζει ο web server φορτώνει το Servlet και όχι την JSP.

Κάθε φορά που η JSP αλλάζει ακολουθείτε ξανά η παραπάνω διαδικασία.

JSP Elements

Τα JSP στοιχεία-ετικέτες κατηγοριοποιούνται σε τρεις βασικές κατηγορίες:

- **JSP Directives**
- **JSP Scripting Elements**
  - **JSP Scriplets**
  - **JSP Declarations**
  - **JSP Expressions**
  - **JSP Comments**
  - **JSP Implicit Objects**
- **JSP Actions**

JSP Directives (<%@... %>)

Γενική σύνταξη: **<%@ directive-name attribute="value" %>**

Δίνουν πληροφορίες στο JSP engine για το τι περιέχει η σελίδα μας, τέτοιες πληροφορίες είναι η γλώσσα σελίδας και κωδικοποίηση (encoding) των χαρακτήρων, βιβλιοθήκες που θα χρησιμοποιήσουμε, session, ορισμός σελίδας σφαλμάτων κα. Στον παρακάτω πίνακα αναφέρονται με παραδείγματα τα πιο «συχνά» directives:

JSP directive	Περιγραφή
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>	Ορίζει πληροφορίες για την σελίδα, όπως γλώσσα (java), encoding UTF-8 (υποστήριξη Ελληνικών χαρακτήρων)
<%@ page import="java.sql.*, java.util.*" %>	Εισάγουμε διάφορες βιβλιοθήκες και πακέτα (στο παράδειγμα εισάγουμε το JDBC API και την java.util.*). Ισοδυναμεί με το import java.mysql.*; και import java.util.*; Αντίστοιχα.
<%@ page errorPage="MyErrorPage.jsp" %>	Ορίζουμε error page, την σελίδα δηλαδή που θα μεταφερθεί ο έλεγχος εάν συμβεί κάποιο σφάλμα.
<%@ page isErrorPage="true" %>	Δηλώνουμε ότι αυτή η σελίδα είναι error page.
<%@ page session="true|false" %>	Εάν πρόκειται να χρησιμοποιήσουμε το session object ή όχι (default is true)
<%@ include file="header.html" %>	«Στατικό» include, με αυτόν τον τρόπο ένα εξωτερικό αρχείο γίνεται μέρος της σελίδας μας την στιγμή της μετατροπής σε Servlet.

Γενικά τα directives τα βάζουμε στην αρχή της σελίδας μας (εξαίρεση το include που μπαίνει και σε άλλα σημεία) και εκτελούνται κατά την μετατροπή του JSP σε Servlet.

JSP Scriplets (<% … %>)

Σύνταξη: **<%** …κώδικας java … **%>**

- Τα **Scriplets** είναι αυτά που χρησιμοποιούμε κατά κόρον όταν συγγράφουμε μια σελίδα JSP.
- Μέσα σε αυτά βάζουμε μόνο κώδικα JAVA (πχ όπως στα Servlet).
- Μπορούμε να έχουμε όσα scriplets θέλουμε στην σελίδα μας.
- Το JSP engine «μαζεύει» όλα τα scriplets της σελίδας μας και τα τοποθετεί στην ίδια μέθοδο (_jspService) στο παραγόμενο Servlet, οπότε ότι μεταβλητές ορίσω σε ένα scriplet μπορώ να τις χειριστώ-προσπελάσω από οποιοδήποτε άλλο στην σελίδα μου.
- Ο κώδικας μέσα στα **scriplets** εκτελείται κάθε φορά που στέλνουμε request στην JSP σελίδα.
- Μέσα στα **Scriplets** δεν μπορούμε να ορίσουμε μεθόδους.

JSP Declarations (<%! … %>)

Σύνταξη: **<%!** …field or method declaration… **%>**

- Τα **JSP Declarations** χρησιμοποιούνται για να ορίσουμε μεθόδους και πεδία.
- Όποιες μεθόδους και πεδία ορίσουμε μέσα στα declarations θα τοποθετηθούν εκτός της (κύριας) μεθόδου _jspService() στο παραγόμενο Servlet.
- Από τα scriplets μπορούμε να χρησιμοποιήσουμε τις μεθόδους και τα πεδία που έχουμε ορίσει στα declarations.

Παράδειγμα:

```
<%!
private int x = 5;
private int addNumbers (int a, int b) { return a+b; }
%>
…
<% int c = addNumbers(3, 5) %>
```

JSP Expressions (<%= … %>)

Σύνταξη: <%= java κώδικας %>

- Τα **JSP expressions** χρησιμοποιούνται για να τυπώνουμε απευθείας (πχ χωρίς out.println()) τιμές που επιστρέφουν μέθοδοι ή μεταβλητές.
- Δεν προσθέτουμε ποτέ ερωτηματικό στο τέλος
- Τα Expressions τοποθετούνται στην ίδια κύρια μέθοδο του παραγόμενου Servlet μαζί με τα scriplets.
- Στο παραγόμενο Servlet μετατρέπονται σε String και μπαίνουν σε out.println()
- Μπορούμε να τυπώσουμε μεταβλητές που έχουμε (προηγουμένως) ορίσει σε scriplets ή declarations.

Παράδειγμα:

```
<%! private int addNumbers (int a, int b) { return a+b; } %>
...
<%= addNumbers(3, 5) %>
```

JSP Comments (<%-- … --%>)

Σύνταξη: **<%--** This is a comment…. **--%>**

- Τα JSP comments χρησιμοποιούνται για ορίσουμε σχόλια στον κώδικα μας
- Το JSP engine τα αγνοεί εντελώς, δηλαδή ότι κώδικας περιέχεται ανάμεσα στην ετικέτα έναρξης και την ετικέτα λήξης δεν εκτελείται.
- Δεν εμφανίζονται στο source της σελίδας, σε αντίθεση με τα σχόλια στην html (<!– this is an html comment -->)
- Είναι χρήσιμα γιατί μπορούμε να μαρκάρουμε και «απενεργοποιήσουμε» κώδικα στην σελίδα μας χωρίς να τον σβήσουμε
- Μπορούν να περιέχουν scriplets, directives, expressions, html κτλ

Παράδειγμα:

```
<%--
<%!  private int addNumbers (int a, int b) { return a+b; } %>
<p> <%= addNumbers(10, 20)  %></p>
--%>
```

JSP Implicit Objects

Τα **JSP implicit objects** είναι (9 συνολικά) αντικείμενα της Java τα οποία μας παρέχονται από την JSP αυτόματα (από default), δηλαδή μπορούμε να τα χρησιμοποιήσουμε χωρίς να χρειάζεται να τα ορίσουμε στην σελίδα μας. Παρακάτω αναφέρουμε αυτά που χρησιμοποιούνται πιο συχνά :

request

Αντιστοιχεί στο HttpServletRequest του Servlet

response

Αντιστοιχεί στο HttpServletResponse του Servlet

out

(JspWriter) «Όμοιο» με τον PrinterWriter του Servlet

session

Αντιστοιχεί στο HttpSession του Servlet

exception

Χρησιμοποιείται μόνο από την error page

Τα **implicit objects** δεν μπορούμε να τα χρησιμοποιήσουμε από τα declarations (δηλαδή σε κώδικα μέσα σε <%! … %>) αλλά μόνο σε Scriplets (δλδ μέσα σε <% … %>)

JSP Actions (1)

Τα **JSP Actions** είναι υψηλού επιπέδου στοιχειά τα οποία μας επιτρέπουν να είτε να εισάγουμε (καλέσουμε) δυναμικά άλλες σελίδες ή αρχεία (include), είτε να προωθήσουμε τον χρήστη (request) σε άλλες σελίδες (forward).

Τα JSP actions δεν τοποθετούνται μέσα σε άλλα JSP elements (όπως πχ scriptlets, directives κτλ)

Παρακάτω αναφέρουμε το include και το forward τα οποία είναι αντίστοιχα του RequestDispatcher που είδαμε στα Servlets.

- **jsp:forward** action (χωρίς παραμέτρους)

Σύνταξη: **<jsp:forward page**="relativeURL | <%= expression %> " **/>**

- **jsp:forward** action (με παραμέτρους)

Σύνταξη:

```
<jsp:forward page="relativeURL | <%= expression %>" >
        <jsp:param name="parametername" value="parametervalue | <%=expression%>" />
</jsp:forward>
```

JSP Actions (2)

Με το **JSP include** actions μπορούμε να εισάγουμε στην σελίδα μας στατικές σελίδες (html) ή/και δυναμικές (JSP ή Servlets) κατά την ώρα που στέλνουμε το request στην σελίδα μας (on request time).
Για αυτό τον λόγο χρησιμοποιούμε το JSP include συνήθως για να εισάγουμε δυναμικές σελίδες σε αντίθεση με το include directive που το χρησιμοποιούμε κυρίως για στατικές σελίδες.

- **jsp:include** action (χωρίς παραμέτρους)

Σύνταξη: **<jsp:include page**="relativeURL | <%= expression %> " **/>**

- **jsp:include** action (με παραμέτρους)

Σύνταξη:

```
< jsp:include page="relativeURL | <%= expression %>" >
        <jsp:param name="parametername" value="parametervalue | <%=expression%>" />
</ jsp:include >
```

Χρήση «Custom» κλάσεων από την JSP

Για να μπορέσετε στις JSP σελίδες να έχετε πρόσβαση σε (δικές σας) Java κλάσεις (να φτιάχνετε αντικείμενα, χρησιμοποιείτε μεθόδους κτλ), ακολουθείτε τα παρακάτω βήματα:

• Χρησιμοποιείτε πακέτα (packages) στις κλάσεις σας

Πχ:

```
package mypackage;
public class Test {
 …
}
```

• Στον δίσκο **W** και στην τοποθεσία WEB-INF\classes φτιάχνετε φάκελο και του δίνετε το ίδιο όνομα με το όνομα του πακέτου της κλάσης σας, στην συνέχεια τοποθετείτε μέσα εκεί την κλάση σας.

• Κάνετε import την κλάση στην σελίδα JSP

(πχ **<%@ page import="mypackage.Myclass" %>**)

Χρήση «Custom» κλάσεων από την JSP (2)

Παράδειγμα

Βήμα 1: Δημιουργία της κλάσης

```
package lesson7;

public class Customer {
        private String name;

        public void setName(String name) {
                this.name = name;
        }
        public String getName() {
                return name;
         }
}
```

Χρήση «Custom» κλάσεων από την JSP (3)

Βήμα 2: Δημιουργία φακέλου

Κάνουμε compile την κλάση (Customer) και στην συνέχεια πάμε στην τοποθεσία **W:\WEB-INF\classes** και φτιάχνουμε τον φάκελο με όνομα **lesson7** (ίδιο όνομα με το όνομα του πακέτου της κλάσης) και βάζουμε μέσα το αρχείο **class** (**Customer.class**)

Βήμα 3: Κάνουμε import την κλάση στην JSP

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

<%@ page import="lesson7.Customer" %>
…
<%
Customer customer = new Customer();
customer.setName("Jim Morrison");
%>
<p> Hello <%= customer.getName() %> </p>
```