

# Adv. Web Technologies 1) Servlets (introduction)

*Emmanuel Benoist*  
Fall Term 2016-17

## Introduction

## Java Servlets

- Introduction
- HttpServlets Class
  - HttpServletResponse
  - HttpServletRequest
  - Lifecycle Methods
- Session Handling
  - Example: Guess a Number

## Introduction to Servlets

- ▶ Modules inside Request/Response-oriented Servers
  - Can handle requests coming from HTML Forms and update the company DataBase.
- ▶ Servlet API : assumes nothing about how the servlet is used
  - Allows servlets to be embedded in many different Web Servers.
- ▶ Platform independant
- ▶ Base for Java and the Web
  - ▶ Java Framework for Web Applications
  - ▶ Specifications implemented in many web servers
  - ▶ Powerful and clean tool
- ▶ Servlets vs PHP
  - ▶ PHP is aimed for small to middle size projects (PHP 4) or pure web projects (PHP 5).
  - ▶ Java : Reusability of existing frameworks or packages, synergies with applications.
  - ▶ AJAX : Web Site has the same functionalities as an Application

# Java Servlets ?

- ▶ **Server gets a request**
  - ▶ Set of rules to send a request to a specific class
  - ▶ One URL → One java class (.class file)
- ▶ **Each class is instantiated only once**
  - ▶ One occurrence
  - ▶ Many requests: method executed many times (for the same object)

# Servlets Architecture Overview, The servlet API

- ▶ **Central Abstraction : Servlet interface**  
All servlets implement this interface
- ▶ **Servlet interface defines the methods for communication with clients**
- ▶ **Servlets receive two objects for any request:**
  - ▶ ServletRequest
  - ▶ ServletResponse
- ▶ **For the web (HTTP)**
  - ▶ A class to extend HttpServlet
  - ▶ Two classes for the parameters, HttpServletRequest and HttpServletResponse.

## HttpServlets Class

- ▶ **Servlets that specialize HttpServlets should override the methods designed to handle HTTP interactions**
- ▶ → **doGet** for handling the GET and HEAD requests
- ▶ → **doPost** for handling POST requests
- ▶ → **doPut** for handling PUT requests
- ▶ → **doDelete** for handling DELETE requests
- ▶ **By default these methods return a BAD\_REQUEST (400) error**
- ▶ **The service method also calls the doOptions and doTrace methods respectively as response to OPTIONS and TRACE requests**

# Parameters

- ▶ **The methods take two arguments :**
- ▶ **HttpServletRequest**  
Contains the request HTTP
  - ▶ Http method used (get, post, ...)
  - ▶ Http header informations (useragent, preferred language, ...)
  - ▶ Parameters sent by the client
  - ▶ Cookies
  - ▶ Session informations
- ▶ **HttpServletResponse**  
Used to store the informations to send to the client
  - ▶ Http status and header
  - ▶ Cookies
  - ▶ Body of the message



# The HttpServletResponse class

- ▶ **To return Text, use the method `getWriter`**
- ▶ **If you want to return data, use method `getOutputStream`**
- ▶ **You should set HTTP header data before sending data.**
  - ▶ `setStatus` Set the status code and message for this response
  - ▶ `sendError` Sends an error response to the client
  - ▶ `sendRedirect` Sends a temporary redirect response to the client
  - ▶ ...
- ▶ **In HTTP Header you can insert a Cookie with `addCookie`**

# Hello World

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

    public void doGet(HttpServletRequest request,
                      HttpServletResponse response)
        throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
```

```

out.println("<html>");
out.println("<head>");
out.println("<title>Hello_World!</title>");
out.println("</head>");
out.println("<body>");
out.println("<h1>Hello_World!</h1>");
out.println("</body>");
out.println("</html>");
}
}

```

## HttpServletRequest

## The HttpServletRequest class

### ▶ Access information in the Http header

- ▶ `getHeader(String name)`

```
request.getHeader("UserAgent");
```

### ▶ Access GET and POST parameters (independently of the method)

- ▶ `String getParameter(String name)` returns the value of name, or null if it is not defined.
- ▶ `Enumeration getParameterNames()` returns an Enumeration of the names
- ▶ `String[] getParameterValues(String name)`

## The HttpServletRequest (Cont.)

### ▶ Access to the cookies

- ▶ `getCookies()` returns an array of cookies sent by the client.

### ▶ For HTTP methods POST PUT and DELETE, you have the choice:

- ▶ `getReader()` : returns a `BufferedReader` to read directly the input if you expect text data
- ▶ `getInputStream()` : returns a `ServletInputStream` if you expect binary data.

# Hello Boss

```
public class HelloBoss extends HttpServlet {  
  
    public void doGet(HttpServletRequest request,  
        HttpServletResponse response)  
        throws IOException, ServletException {  
        PrintWriter out = response.getWriter();  
        out.println(...);  
        out.println("<h1>Hello World!</h1>");  
        String username = request.getParameter(  
            "username");  
  
        if(username==null) printForm(out);  
        else out.println(" Hello_" +username);  
        out.println(" ...");  
    }  
}
```

## Lifecycle Methods

# Hello Boss (Cont.)

```
void printForm(PrintWriter out){  
    out.println("<form>");  
    out.println("What is your name?");  
    out.println("<input type=\"text\" +  
        \" name=\"username\" >");  
    out.println("</form>");  
    out.println("");  
}  
  
}
```

## Lifecycle methods : *init*

- ▶ **Initialisation** : the server should prepare the resources it manages
- ▶ **Init method takes a ServletConfig object as a parameter**  
The method have to save this object, so it can be returned by the *getServletConfig* method : call the *super.init* method.

## Lifecycle methods : *init* (Cont.)

```
public void init(ServletConfig config)
    throws ServletException
{
    super.init(config);

    //Store the directory that will hold the
    //survey-results files
    resultsDir = getInitParameter("resultsDir");

    //If no directory was provided,
    if (resultsDir == null) {
        throw new UnavailableException (this,
            "Not_given_a_directory!");
    }
}
```

## Lifecycle methods : *destroy*

- ▶ **This method is called when the server unloads the servlet**
- ▶ **It should undo any initialization and synchronize persistent state with current in-memory state.**
- ▶ **The service method might still be running when destroy is called**
- ▶ **It is often used to close connection to a server (for instance Data Base)**

## Lifecycle methods : *destroy* (Cont.)

```
/** Cleans up database connection */
public void destroy() {
    try {con.close();}
    catch (SQLException e) {
        while(e != null) {
            log("SQLException:_" + e.getSQLState() +
                e.getMessage() + '\t' +
                e.getErrorCode() + '\t');
            e = e.getNextException();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

## Providing information about the Servlet

- ▶ **Some applets and applications display information about a servlet**  
For instance:
  - ▶ short description
  - ▶ purpose of the servlet
  - ▶ author
  - ▶ version number
- ▶ **The servlet API provides the method `getServletInfo` to return this information**

# Providing information about the Servlet

```
public class SimpleServlet extends HttpServlet {  
    ...  
    public String getServletInfo() {  
        return "A_simple_servlet";  
    }  
}
```

## Session Handling

- ▶ **Session are everywhere on the internet**
  - ▶ Authorisation
  - ▶ Basket for e-commerce
  - ▶ Statistics
  - ▶ ...
- ▶ **Get the current Session**  
Session session = request.getSession();
- ▶ **Get attributes of a session**
  - ▶ `getAttribute(name)` returns the Object stored under the given name
  - ▶ `setAttribute(name, value)` affects an Object to a string
  - ▶ `removeAttribute(name)` to end the action.

## Session Handling

## Example: Guess a Number

# Example: Guess a Number

- ▶ Game for kids
  - ▶ The user has to find a number
  - ▶ The computer choose a number, it answers *higher or lower*
- ▶ Java Structures
  - ▶ Form to type the guess
  - ▶ Tests if valid
  - ▶ If not : error
  - ▶ If OK, compares with the hidden number

# Guess.java

- ▶ Access the GET parameter number

```
String number = request.getParameter("number");
```

- ▶ Creates a new number to find if needed

```
if(request.getSession().getAttribute("TOGUESS")  
    ==null){  
    int answer = Math.abs(new Random().nextInt()  
        % 100) + 1;  
    request.getSession().setAttribute(  
        "TOGUESS", new Integer(answer));  
}  
int toGuess = ((Integer)request.getSession().  
    getAttribute("TOGUESS")).intValue();  
int intGuess = -1;
```

# Guess.java (Cont.)

- ▶ Tests the inputs:

```
if(number==null){  
    out.println("Wellcome_in_our_game,");  
    out.println("choose_a_number");  
}  
else{  
    try { intGuess = Integer.parseInt(number); }  
    catch (Exception e) {  
        error(out,"Malformed_number");  
        return;  
    }  
    if(intGuess < 0 || intGuess > 100){  
        error(out,"Number_out");  
        return;  
    }  
}
```

# Guess.java (Cont.)

- ▶ Test the value to return the message

```
if(intGuess==toGuess){  
    out.println("Congratulation_you_found_it");  
    request.getSession().setAttribute("TOGUESS",null);  
}  
else if(intGuess < toGuess){  
    out.println("Nice_Try,_it's_larger<br>");  
}  
else{  
    out.println("Not_so_bad,_it's_smaller<br>");  
}
```



# Conclusion

- ▶ Java Servlets Engine handles HTTP Requests and generate Responses
- ▶ Servlet Class receives `HttpServletRequest` and `HttpServletResponse`
- ▶ Servlet can use built-in mechanism for dealing with
  - ▶ Sessions,
  - ▶ Configuration of the application
- ▶ Servlet contain a lot of `out.println`
  - ▶ Layout is developed in Dreamwaver
  - ▶ Then transformed in Java
  - ▶ What about a modification?
- ▶ Idea: could we integrate Java inside a HTML page like for PHP
  - ▶ Answer Yes, it is called JSP (Java Server Pages)