EE6052 Web-based Application Design
CE4208/EE4023/ED5022 Distributed Systems

Exercise Sheet 2

Always log on as user "puser" with password "PUSER", using the domain of the local machine (**not** the UL domain!).

Additional Information: A HTML reference guide can be found at `http://www.htmlref.com` (many others do exist and are equally helpful). Exercise 2.4: Sample code for file handling is provided on the module's web page.

Objectives:

Learn how to:

- Code and invoke HTTP servlets.
- Including other resources in HTTP servlets
- Using cookies and session objects to save user settings
- Using files in HTTP servlets to save user information.

Ex2.1: *Your first HttpServlet.*
Perform the following steps:

- Create a new web application project in NetBeans.
- Create a new package "httpServlets" in the "Source Packages" folder.
- Add a new servlet "MyHttpServlet" to the package "httpServlets" (don't forget to tick the "Add information to deployment descriptor (web.xml)" box).
- Remove the method `protected void processRequest(HttpServletRequest request, HttpServletResponse response)` and all its comments.
- Your servlet now contains a line "HttpServlet methods. Click on the + sign on the left to edit the code". Click on the + symbol and remove the comment-line that starts with "// <editor-fold defaultstate=". Also, remove the (now invalid) calls to `processRequest(...)` in the `doGet(...)` and `doPost(...)` methods.
- In the `doGet(...)` method add commands to create a form with three fields (choose any information you like). The action item of the form should point to MyHttpServlet using method POST.
- In the `doPost(...)` method add commands to display the values entered in the form of the GET method.
- Run your application and test its behaviour.

Ex2.2: *Including other resources.*
Expand the previous exercise: Create static html files (these need to be placed in the "Web Pages" folder) for a header banner and a footer banner (use the text "This is a header banner"/ "This is a footer banner" or any other text you like). Use the `include(...)` method from a `RequestDispatcher` to include both banners in your GET and POST method. Run your application and test its behaviour.

Ex2.3: *Using cookies/session objects to save user settings.*
Create a HTTP servlet that displays some arbitrary content, but uses a variable background colour, foreground colour and text size. Values for theses items should be retrieved from cookies. If no corresponding cookies are found, use some default values.

Depending on the solution (see below), either provide the page with a link to a user settings page (HTML code for a basic configuration form can be found at module's web page - you need to adjust it to your application) or provide the settings form on the same page. The configuration form allows the user to select different values for background colour, foreground colour and text size. When the selected configuration is submitted (to your HTTP servlet), the selected values are used to display the content and are also stored in cookies (one cookie per selected item, i.e. one for background colour, one for foreground colour and one for text size). Use persistent cookies, i.e. provide your cookies with a maximum age.

Provide three solutions for this exercise:

(a) Use a single serlvet. Follow the HTTP servlet template provided by NetBeans (that executes the method `processRequest(...)` for both, GET and POST requests. This servelt will provide some content (any arbitrary text) and also the configuration form to change text size and colours. Make sure to react properly to any changes to the user settings (the target for the configuration form is this servlet itself).

(b) Use two separate servlets:

   i. One serving GET request, which provides the content (again any arbitrary text) displayed in the selected colours and font size.

   ii. Another servlet that processes both, GET and POST requests. The GET request provides the form to select colours and font size, where the target is the same servlet itself (using POST). The POST request will evaluate the results from the form and stores the selected values in cookies. Then is uses the `sendRedirect(...)` method of a `HTTPServletResponse` object to pass control on to the content providing servlet.

(c) Use Session Objects rather than cookies (implement either the single servlet or the two servlet architecture - of course feel free to do both again ;-))

Test your solutions: Make sure that on the first request default settings are used. Change some settings and access the content again - now with the chosen settings. Close all your brower windows access the content servlet again - it should retain your settings. Clear all cookies from your browser and access the content again - have settings reversed to default?

Ex2.4: *More cookies ...*
Modify the previous exercise: rather than using mutliple cookies use a single cookie to identify the user and store the values in member variables of the servlet. Your servlet should be able to store settings for multiple users - use classes such as `java.util.ArrayList` to store information.
Advanced Option: Make sure that your servlet handles concurrent request correctly, i.e. make sure it synchronises access to the member variables.

Ex2.5: *Using files in Servlets*
Create a HTTP servlet and provide/override the following methods:

- `public void init()`: This method will open a file that can be written to during the lifetime of the servlet (example code of how to open files in Java are provided at the module's web page). Use the line `String filename = getServletContext().getRealPath( "/WEB-INF/FileServletLogFile.txt");` to create your filename - otherwise, you may end up with a location at which Tomcat is unable to access files. With this apporach, your file will be located at "*base*/build/web/WEB-INF", where *base* equals the folder in which you created your NetBeans project.

  Attention: If you override `void init(ServletConfig config)` instead, then your first statement should be `super.init(config)` to ensure proper setup of the base class component.
- `public void destroy()`: This method will close the file that has been opened in `init()` (example code of how to open files in Java are provided at the module's web page).
- `protected void processRequest(...)` This method simply displays some content. The content includes a link to the servlet itself to provoke GET access to the servlet and a form (all you need is a submit item) to provoke POST access to the servlet.
- `protected void doGet(...)` This method first prints a message to the file opened in `init()` of the form "GET method requested at XXXX", where XXXX is the time as obtained from `System.currentTimeMillis()` (the current time represented in miliseconds passed since 1 January 1970 UTC) and then calls the method `processRequest(...)`.
- `protected void doPost(...)` Equivalent to `doGet(...)`, just writing the message "POST method requested at XXXX"

Make sure that your servlet handles concurrent request correctly, i.e. make sure it synchronises access to the file.

Test your application - keep in mind that the `destroy()` method only gets called when the servlet is unloaded. To enforce this, stop the Tomcat server - it will start automatically again if you run another project/file.