Tasks to Solve During Seminar 3

Internet Applications, ID1354

Write a document with your answers to the following tasks, and upload it to Canvas at the end of the seminar. The format of the document is not important. The problems shall be solved by all group members together. You may also write the document together, but everyone must add their personal section describing what was learned during the seminar.

Mandatory Task

- All members of the group shall, in turn, explain and motivate their program, and also their report, to the other group members. Write a brief summary of differences, and comment on advantages and disadvantages of the different solutions. Do not write about minor layout differences.
- Go through all requirements specified in the lab tasks, and evaluate that your web sites meet them. Also evaluate whether the PHP code is well designed and has an appropriate layered architecture. It often happens that cohesion is lowered by placing tasks in the wrong layer. Check that the following rules are followed.
  - There shall not be anything related to HTTP or HTML outside the view. This means it shall be possible to change to a view that is not web-based, without having to change anything outside the view layer.
  - Nothing that is displayed on the screen shall be generated outside the view layer. This means it is a mistake if the view displays a string that is created in another layer. As an example, if the user tries to register and the model finds that the username is taken, it is not the task of the model to return a string saying 'please try another username', which is then just displayed by the view.
  - There shall not be anything related to databases outside the integration layer. This means it shall be possible to change between storing data in a text file or database, without having to change anything outside the integration layer.
  - There shall not be any logic in the integration layer. Typically, integration contains only methods that insert, read, update or delete data in the data storage, without bothering about the meaning of the data. It is for example not appropriate to check that username and password match in the integration layer. That is logic and belongs in the model.

- Evaluate that your reports meet the requirements in the report template and in the lab tasks.
- A good way to evaluate if the code is flexible and easy to understand is to actually try to change something. Select one or more of your web sites and try to change it *without the help of the author*. Try to find something that involves both PHP and HTML/CSS. It could for example be to display another page after the user has succeeded or failed to login, or to change the way the user is informed about the result of the login attempt. Another suggestion is to move the delete button for comments.

Optional Tasks

Solve as many of these tasks as time allows, and write your solutions in the same document as the mandatory task. If the solution is a piece of code, just paste the code in the document, without bothering about figures or formatting. The tasks may be solved in any order, you do not have to start from task one.

Task 1, Frameworks

a) It is almost always the case that a http request is received by one php file, which then includes another file containing the next view (otherwise the code will be terribly messy). This means that the URL does not match the path to the file with the view. If we are using a framework with routing functionality, the URL does not even match the path of the file handling the http request. *Make sure you understand why!*

Now, suppose a browser has made an HTTP request to **`http://www.myserver.se/abc/def`**. The server framework maps this URL to the file **`classes/ghi.php`**, which includes **`views/jkl.html`**. The view that is now displayed in the browser is the HTML code in **`views/jkl.html`**.

1) Which URL is now displayed in the browser's address field?
2) The layout of **`views/jkl.html`** is specified in **`resources/css/mno.css`**. **`jkl.html`** contains a link to **`mno.css`**, which path shall that link have?

b) Is there any feature you miss in the id1354-fw framework? Maybe handling request or application lifetime variables the same way session variables are handled? Or the possibility to include data in a view without writing php code? Or possibility to include fragments like header and footer without writing php? Try to find out how some missing feature could be implemented. You do not have time to implement it in code, just try to understand how it could be done. Ask the teacher for hints if you wish.

c) Try to understand how some existing functionality in the framework is implemented, for example session handling, routing or including views. You can try to figure out a possible implementation on your own or read the framework's source code. Ask the teacher for hints if you wish.

Task 2, Security

Try to perform an attack on one of your sites. You can, for example, try to impersonate another user, using one of the strategies below.

1. Perform a session hijacking attack, by reading the user's PHPSESSID cookie and setting your own cookie to that value. You can try to get the logged in user's cookie with an XSS attack, as suggested on slide 38 of the presentation from lecture 9.

2. If someone in the group has a packet sniffer tool (that can monitor network traffic), use it to read the user's password or PHPSESSID cookie. This is possible on KTHOPEN, but not on eduroam, where all communication is encrypted.

3. If you fail to steal the cookie, you can still check the outcome of a successful attack by just reading the PHPSESSID cookie on the logged in user's computer, and setting your own PHPSESSID to that value.