

JOSSE Final Exam

Please write all answers on one or more sheets of 8.5x11 paper. Please use a pencil and write legibly and neatly. Clarity of thought counts; if I cannot understand your answer, I cannot give you credit for it. You folks have all done a great job this semester and I look forward to seeing you again next Fall.

1. Please describe the essential aspects of the "model-view-controller" design pattern as it applies to Servlet and JSP-based web application request processing. Your description should include the principle components of MVC and how they are implemented, as well as a sample scenario which illustrates the data and control flow in MVC.

Model: one or more server-side classes that represent the state of the application and/or client workflow. View: JSP pages implement the user interface for the client without containing any navigation code. Controller: a single servlet that is invoked by all client requests. The controller, based upon the client request, additional data supplied by the request, and the current state of the application, decides what code to invoke to process the request and determines what page should be displayed next to the user.

A sample scenario from the "StackMVC" application: the user invokes the "Push" command with an optional parameter indicating the value to be pushed. This request goes to the "Controller" servlet, which gets the command and value to be pushed from request parameters, and dispatches to the PushCommand.process method based upon these values. The process method alters the state of the model by attempting to push the new value onto it, and returns the specific JSP page to be displayed next after augmenting the request object with additional data that enables the page to display the current value of the stack. The next page to be displayed could be either an error page or a "normal" JSP page depending upon the outcome of processing.

2. Although all of the web application code I distributed this semester used the MVC design pattern, it is not the only way to implement web application request processing. Please describe an alternative design for request processing and compare it to MVC. In what ways is your alternative superior to MVC? In what ways is it inferior?

Another way is to not use a controller servlet, but instead embed all logic and navigation directly in the JSP page. The "advantage" of this approach is its simplicity for trivial web applications. The disadvantage is that it is highly unmaintainable for non-trivial web applications since there is no separation of concerns. For example, every page needs to hardwire knowledge about every other possible page that could follow it and the circumstances under which each following page should be returned.

3. What is the difference between an "open source" software license and a GNU "free" software license? Why do advocates of GNU software licensing believe that "open source" is not "free"?

The GNU license is "viral": it requires all hackers of GNU licensed code to redistribute it and any changes made to it by them with the same GNU license. An "Open Source" compliant license does not require this "viral" clause. Thus, while all GNU licenses are Open Source, not all Open Source licenses are GNU. Folks in the GNU world think Open Source is not free because, without the viral clause, a "free" software can be converted to "non-free".

4. What is "Test First Design"? Compare and contrast TFD with the testing strategy you used in your projects this semester.

TFD requires that you write, implement, and run your test cases for a given functionality before starting implementation of the functionality. Obviously, the first time you run them, they will fail.

Most folks this semester implemented the functionality first and used testing as a regression prevention strategy. This generally meant that not every functionality had a corresponding test case.

The back of this paper provides a listing of a file index.jsp, which presents the user with two selection lists containing numbers and a button they can press to add the two numbers together and display the result at the bottom of the page

5. Assume the MVC infrastructure we used this semester is available, and implement the command class (edu.hawaii.final.control.command.AddCommand). Demonstrate that you understand how a command class in the MVC design pattern would support this JSP page.

Something like the following would be good enough. No, this won't compile, and that's OK.

```
public class AddCommand implements Command {
    public Page process(HttpServletRequest req) {
        int num1 = Integer.parseInt(req.getParameter("number1"));
        int num2 = Integer.parseInt(req.getParameter("number2"));
        req.setAttribute("addResult", num1 + num2);
        return Page.INDEX;
    }
}
```

6. Write a single HttpUnit test to test one aspect of the correctness of the web application represented by this index.jsp page.

Something like the following would be good enough. No, this won't compile, and that's OK.

```
public void testPage {
    WebConversation wc = new WebConversation();
    String url = getUrl();
    WebResponse response = c.getResponse(url);
    WebForm form = response.getFormWithID("AddForm");
    WebRequest req = form.getRequest();
    req.setParameter("number1", "2");
    req.setParameter("number2", "1");
    response = wc.getResponse(req);
    WebTable table = response.getTableWithID("AddResult");
    assertEquals("Testing 2+1", "3", table.getCellAsText(0,0));
}
```