CISC474 - Spring 2005

Advanced Web Technologies - Midterm Exam 1 4/14/2005

Name (as on official roster): ______________________________

Name you go by in class (if different): ______________________________

- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- You have 75 minutes
- You are permitted to use one 8.5 x 11 sheet of notes, (both sides) which you must turn in with your exam. Put your name on it. You won't get it back, so make a copy before you come to the exam. It *is* permitted for your notes to be a laser print or photocopy.
- **Pace Yourself!!!!!**

  Pay attention to the point values. When there are 10 minutes left, skim through and be sure you have at least written *something* for the questions that are worth many points.
- Read *all* the directions *carefully* on each problem.
- In your answers be as precise as possible. Be sure to answer the question that is *asked.* Don't just do a brain dump of everything you know.

Section 1. Short Answer

1. Illustrate how to write a comment in each of the following languages by putting the word “comment” inside a comment.

   - (2 pts)HTML
   - (2 pts) Java
   - (2 pts) JSP

2. (4 pts) Since HTTP is a stateless protocol, web applications have to do something “special” to maintain states. One way is using Cookies. What do technique do web servers that implement Java Servlets and JSPs use when cookies don’t work (e.g. the end user has disabled cookies in his/her browser)? Briefly describe this technique, and mention what “special extra thing” the web application programmer has to do to make it work.

3. (10 pts) Your textbook indicates that “Servlets don’t have a main() method. They’re under the control of another Java application...”.

   Briefly explain what this “other Java application” is.

   In your answer, address the following issue:

   > Suppose you could go hunting through source code at will (regardless of whether it is publicly available or not.) Where would you find the source code for the main() method that starts this “other application”?

4. (10 pts) Write a brief web page that allows someone to "sign up for a mailing list". The page should illustrate the proper use of the tags listed below (They are listed in alphabetical order, not the order in which you should use them.)

   The web page should include an input fields for "name" and "email address", and a submit button. The page should submit a form to a servlet called "signMeUp.do" using the POST method.

   You may use other tags if you like, but don't get too fancy: no extra credit here for making the page beautiful, and you are under a time-constraint.

   ```
   <body>
   <br>
   <form>
   <head>
   <html>
   <input>
   <p>
   <title>
   ```

   Add attributes to the tags as needed. Some attributes you might need are `method`, `action`, `type`, and `name` (you'll have to figure out which tags they belong with). Before you start, take a look at the following question also.

   Be sure all tags are closed and properly nested.

5. (10 pts) Now draw a picture of how a typical graphical browser (Firefox, IE) might render the web page you just created. Keep it simple; don't spend *too* much time on it, or you won't have time for the other questions.

6. (10 pts) Typically a web page that invokes a servlet does so either via a GET method or a POST method. The browser then communicates that choice to the web server somehow, which then invokes either the doGet() method or the doPost() method written by the web application designer.

   This raises a couple of questions. First, how does the web designer choose between GET and POST? Briefly explain the various issues involved. Don't go on for pages and pages, but be sure to at least mention *all* the relevant issues.

7. (15 pts) Your textbook indicates that “A Servlet can have THREE names”: a client-known URL name, a deployer-known *secret internal* name, and the actual file name.

   Give example that illustrates the purpose of all three of these names. Include each name, and a short rationale as to the purpose that name serves (as distinct from the other two.) Your example can be a “real one” from your Hilfbar, Beer or Product application, or you can make up a pretend one.

   Hint: the “actual file name” has both a directory part and a filename part; be sure to include both, and explain the relationship to a particular keyword in the Java programming language.

8. Related to the preceding question: if you are building a web application, where is the place that you encode the relationship among each of the three “names” that a servlet can have? Give me all of the following for full credit:

   - (3 pts) the correct “term” (two words) used for this part of the web application

   - (3 pts) the “language” or “format” in which this encoding is made (an abbreviation is fine)

   - (3 pts) the name of the directory and file in which you would encode this information. (In Resin there are at least two possible answers here; I’ll accept either for partial credit, but I’m looking for the one that is more common, and referred to by your textbook.)

9. Still related to the preceding question, answer each of the following questions about the MVC design pattern. Explain your answers with enough specific to make it clear to the grader that you understand the *role and purpose* of “M”, “V” and “C”, not *just* what they stand for—though you should mention what they stand for as well.

   - (4 pts) Which part (M, V, or C) would typically be represented by the “actual file name” that was part of the preceding question (the one about “three names for a Servlet?) Explain.

   - (4 pts) Which part (M, V, or C) would typically be written as a JSP rather than as a Servlet? Explain *why* a JSP more appropriate for this part than a Servlet.

   - (4 pts) Which part (M, V, or C) would typically be written as a Java class NOT derived from HttpServlet? Explain why.

   - (4 pts) OOP and design patterns encourage code-reuse. Which part (M, V, or C), if written properly, could typically be re-used in a non-web-based application? Explain.

10. (5 pts) Chapter 5 of your textbook includes an example problem where you want to put an email address on each page of your web application. Since this email address might change, you don't want to hard code it on each page, and you don't want it to be in Java code that has to be recompiled.

    So it goes in a "special place" (I can't mention what that place is without giving away the answer to one of the other questions on the exam.. but you know where I mean, right?) Then, you can change it in one place only, and retrieve it whenever you need it using one of the following methods:

    ```
    getServletContext().getInitParameter("emailAddress");
    ```

    or

    ```
    getServletConfig().getInitParameter("emailAddress");
    ```

    Now, it may seem like a picky detail... and I told you those wouldn't be on the exam. However, this particular picky detail was *hammered* home in both Chapter 4 and 5 as something "really important".

    So I ask you: what's the difference between these two methods, and which one would you use for a JSP (as opposed to a Servlet?)

11. (5 pts) Chapter 5 talks a lot about "thread safety". Describe (very briefly) at least one example of a case of something you might want to do in a web-app where thread safety has to be considered.

    In your answer, mention why synchronizing on a particular Servlet (or, equivalently, using the "Single-Threaded Model", which your book describes as "evil") typically doesn't solve anything.

Total Points: 100

CISC474 - Spring 2005

Advanced Web Technologies - Midterm Exam 1 4/14/2005

Name (as on official roster): ______________________________

Name you go by in class (if different): ______________________________

- DO NOT WRITE YOUR NAME ON ANY PAGE EXCEPT THIS ONE!
- You have 75 minutes
- You are permitted to use one 8.5 x 11 sheet of notes, (both sides) which you must turn in with your exam. Put your name on it. You won't get it back, so make a copy before you come to the exam. It *is* permitted for your notes to be a laser print or photocopy.
- **Pace Yourself!!!!!**

  Pay attention to the point values. When there are 10 minutes left, skim through and be sure you have at least written *something* for the questions that are worth many points.
- Read *all* the directions *carefully* on each problem.
- In your answers be as precise as possible. Be sure to answer the question that is *asked.* Don't just do a brain dump of everything you know.

**********Answer Key**********

Section 1. Short Answer

1. Illustrate how to write a comment in each of the following languages by putting the word "comment" inside a comment.

   - (2 pts)HTML
   - (2 pts) Java
   - (2 pts) JSP

   Answer:
   - HTML:
     ```
     <!-- comment -->
     ```
   - Java:
     ```
     // comment
     ```
   - JSP:
     ```
     <%-- comment --%>
     ```

2. (4 pts) Since HTTP is a stateless protocol, web applications have to do something "special" to maintain states. One way is using Cookies. What do technique do web servers that implement Java Servlets and JSPs use when cookies don't work (e.g. the end user has disabled cookies in his/her browser)? Briefly describe this technique, and mention what "special extra thing" the web application programmer has to do to make it work.

   Answer: The technique is called URL rewriting. An extra parameter is added to each URL to identify the session. The requires that the URL the user clicks on to access subsequent pages in a session be encoded using the response.EncodeURL(); method when the HTML is added to the response by an out.println() in a Servlet, or the
   `<c:URL>` tag in a JSP.

3. (10 pts) Your textbook indicates that "Servlets don't have a main() method. They're under the control of another Java application...".

   Briefly explain what this "other Java application" is.

   In your answer, address the following issue:

   > Suppose you could go hunting through source code at will (regardless of whether it is publicly available or not.) Where would you find the source code for the main() method that starts this "other application"?

   Answer: The main method is part of the Container (HF p. 39). The main() method would be part of the java source code for the particular Container, be that Tomcat or Resin or whatever.

4. (10 pts) Write a brief web page that allows someone to “sign up for a mailing list”. The page should illustrate the proper use of the tags listed below (They are listed in alphabetical order, not the order in which you should use them.)

   The web page should include an input fields for “name” and “email address”, and a submit button. The page should submit a form to a servlet called “signMeUp.do” using the POST method.

   You may use other tags if you like, but don’t get too fancy: no extra credit here for making the page beautiful, and you are under a time-constraint.

```
<body>
<br>
<form>
<head>
<html>
<input>
<p>
<title>
```

   Add attributes to the tags as needed. Some attributes you might need are `method`, `action`, `type`, and `name` (you’ll have to figure out which tags they belong with). Before you start, take a look at the following question also.

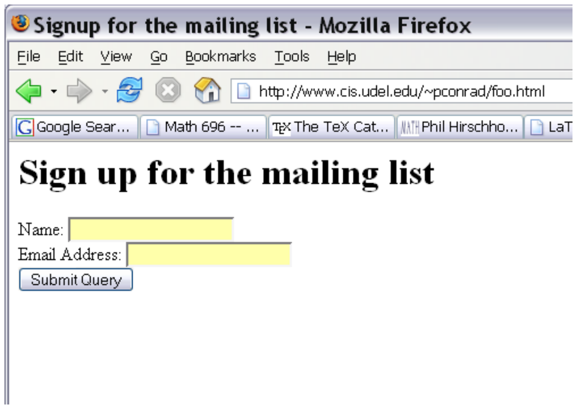
   Be sure all tags are closed and properly nested.

   Answer: See [HF p. 7-9, 75, 117-118].

```
    <html>
    <head><title>Signup for the mailing list</title>
    </head>
    <body>
    <h1>Sign up for the mailing list</h1>
    <form method="POST" action="signMeUp.do">
    Name: <input type="text" name="theName"/><br/>
    Email Address: <input type="text" name="theEmailAddress"/><br/>
    <input type="SUBMIT"/>
    </form>
    </body>
    </html>
```

5. (10 pts) Now draw a picture of how a typical graphical browser (Firefox, IE) might render the web page you just created. Keep it simple; don't spend *too* much time on it, or you won't have time for the other questions.

Answer:



Signup for the mailing list - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

http://www.cis.udel.edu/~pconrad/foo.html

Google Sear... Math 696 -- ... The TeX Cat... Phil Hirschho... LaT

Sign up for the mailing list

Name:

Email Address:

Submit Query

6. (10 pts) Typically a web page that invokes a servlet does so either via a GET method or a POST method. The browser then communicates that choice to the web server somehow, which then invokes either the doGet() method or the doPost() method written by the web application designer.

   This raises a couple of questions. First, how does the web designer choose between GET and POST? Briefly explain the various issues involved. Don't go on for pages and pages, but be sure to at least mention *all* the relevant issues.

   Answer: The issues involved in GET vs. POST include these:

   - Strictly speaking, except for a length restriction on the number of parameters that can be conveyed via a GET, there is no restriction on whether to use GET or POST; they can be used interchangeably. However, there are several matters of security, style, and usefulness that constrain the choice, as follows:
   - The parameter values in a GET are included in the URL. This allows a request to be bookmarked. That is useful for requests that a user might repeat (e.g. looking up a web search, looking up a list of open classes). However, if any of the parameters are sensitive, such as a password or credit card number, these parameters would be revealed in the URL line, which is not such a good thing. POST hides these for better security. Also, the length of a URL is restricted, so if the number or length of parameters is large, POST should be used instead of GET.
   - The Servlet specification says that all servlets that implement a GET method should be idempotent. That means that they should NOT make changes in the state of the application (e.g. updating a database). GET methods should "retrieve" information only. This doesn't include trivial changes such as updating access counters, for example. Anything that changes the database should be done with a POST. However, this is a matter of style only; there is nothing to prevent a designer from violating this spec.
   - Anything that can be done with GET can also be done with POST.

7. (15 pts) Your textbook indicates that “A Servlet can have THREE names”: a client-known URL name, a deployer-known *secret internal* name, and the actual file name.

   Give example that illustrates the purpose of all three of these names. Include each name, and a short rationale as to the purpose that name serves (as distinct from the other two.) Your example can be a “real one” from your Hilfbar, Beer or Product application, or you can make up a pretend one.

   Hint: the “actual file name” has both a directory part and a filename part; be sure to include both, and explain the relationship to a particular keyword in the Java programming language.

   Answer: A URL name is what is encoded in the web page link that brings up the Servlet. It is something that the application designers can give to the HTML coder (who might be a different person from the Java Servlet programmer), that the HTML coder can include in the page, without having to know anything about how the servlet is actually deployed. It is also publicly available in the source code of all web pages. It has no relationship to actual directories or file names on the server side.

   The actual filename of the servlet must be XXX.class, where XXX is the name of the Java class that extends HttpServlet. It is preceded by a directory path such as com/example/foobar which would correspond to a package name that would included with an import statement, such as import com.example.foobar.*;. This is a name that would need to be chosen by the Java programmer.

   The internal secret name is a way of mapping the two together (HF. 46-47). It allows the name in the HTML and/or the name in the Java code to change independently of one another, without having to recompile the Java code if the change is on the HTML side, and without having to change the HTML if the change is on the Java side.

8. Related to the preceding question: if you are building a web application, where is the place that you encode the relationship among each of the three “names” that a servlet can have? Give me all of the following for full credit:

   - (3 pts) the correct “term” (two words) used for this part of the web application
   - (3 pts) the “language” or “format” in which this encoding is made (an abbreviation is fine)
   - (3 pts) the name of the directory and file in which you would encode this information. (In Resin there are at least two possible answers here; I’ll accept either for partial credit, but I’m looking for the one that is more common, and referred to by your textbook.)

   Answer: The answers are:

   - deployment descriptor
   - XML
   - WEB-INF/web.xml under the main directory for the particular web app.

9. Still related to the preceding question, answer each of the following questions about the MVC design pattern. Explain your answers with enough specific to make it clear to the grader that you understand the *role and purpose* of "M", "V" and "C", not *just* what they stand for—though you should mention what they stand for as well.

   - (4 pts) Which part (M, V, or C) would typically be represented by the "actual file name" that was part of the preceding question (the one about "three names for a Servlet?) Explain.
   - (4 pts) Which part (M, V, or C) would typically be written as a JSP rather than as a Servlet? Explain *why* a JSP more appropriate for this part than a Servlet.
   - (4 pts) Which part (M, V, or C) would typically be written as a Java class NOT derived from HttpServlet? Explain why.
   - (4 pts) OOP and design patterns encourage code-reuse. Which part (M, V, or C), if written properly, could typically be re-used in a non-web-based application? Explain.

   Answer:
   - Actual file name: C, controller. This is a servlet that takes the input parameters, passes to the model to compute some information (e.g. an answer to the end user's question), and then passes that computed answer to the view for formatting (the view might be a JSP, for example.)
   - Which part a JSP: V, view. This is because the "view" is responsible only for formatting information nicely. JSPs allow you to write a page that is mostly HTML with a little bit of Java code to access the things that need to be formatted (e.g. iterate through a list.)
   - Which part NOT derived from HttpServlet: M. The model is typically code that is independent of the fact that this is a web app; its methods are called by the controller to look up some information, compute an answer, update a database, etc.
   - Which part could typically be re-used in a non-web-based application? M (same explanation as previous answer.)

10. (5 pts) Chapter 5 of your textbook includes an example problem where you want to put an email address on each page of your web application. Since this email address might change, you don't want to hard code it on each page, and you don't want it to be in Java code that has to be recompiled.

    So it goes in a "special place" (I can't mention what that place is without giving away the answer to one of the other questions on the exam.. but you know where I mean, right?) Then, you can change it in one place only, and retrieve it whenever you need it using one of the following methods:

    ```
    getServletContext().getInitParameter("emailAddress");
    ```

    or

    ```
    getServletConfig().getInitParameter("emailAddress");
    ```

    Now, it may seem like a picky detail... and I told you those wouldn't be on the exam. However, this particular picky detail was *hammered* home in both Chapter 4 and 5 as something "really important".

    So I ask you: what's the difference between these two methods, and which one would you use for a JSP (as opposed to a Servlet?)

    Answer: getServletContext().getInitParameter(); retrieves init parameters for the entire web app as opposed to an individual servlet. It should have been named "appContext". That's the one you'd use for a JSP. getServletConfig().getInitParameter(); retrieves init parameters for a specific servlet only. Both retrieve value/string pairs that can be defined in the Deployment Descriptor, i.e. the WEB-INF/web.xml file.

11. (5 pts) Chapter 5 talks a lot about "thread safety". Describe (very briefly) at least one example of a case of something you might want to do in a web-app where thread safety has to be considered.

    In your answer, mention why synchronizing on a particular Servlet (or, equivalently, using the "Single-Threaded Model", which your book describes as "evil") typically doesn't solve anything.

    Answer: An example is accessing an attribute that is part of the ServletContext (which, of course, we know, should be called the AppContext). Because there can be more than one Servlet in a web-app, synchronizing on a particular servlet doesn't help. Instead, you need to synchronize, for example, on the getServletContext method. The Single Threaded model ensures that each servlet can have only one thread, but doesn't ensure that a particular web-app can have only one thread. So, it impairs performance without actually helping race conditions between and among servlets that are part of the same web app.

Total Points: 100