CISC474 - Spring 2006

Advanced Web Technologies - Final Exam 5/24/2006

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 two hours 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. In web applications, The technique known as “URL rewriting” is typically employed as a “backup strategy” in case a particular “preferred” technique fails.

    - (5 pts) What is the “purpose” of URL rewriting, and what is the preferred technique that is generally tried first?

2. (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”?

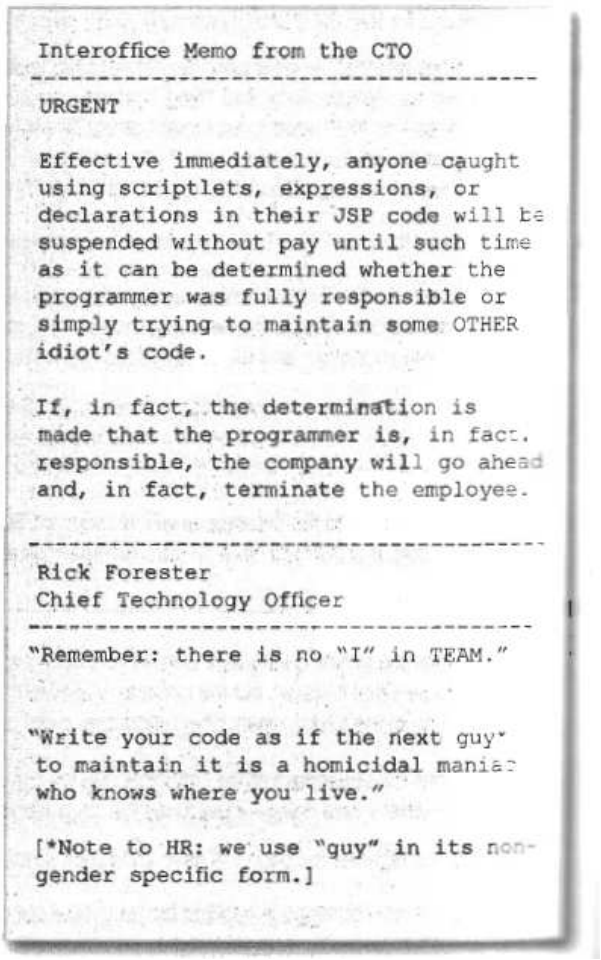
3. Consider the memo from the textbook, p. 314 (reproduced on the following page to refresh your memory).

   - (9 pts) The memo bans the use of Scriptlets, Expressions, and Declarations in JSPs. Give an example of each (in the context intended by this memo, i.e. as they would appear in a JSP.)

     If you need more room, continue on the space next to the memo on the following page.

   - (5 pts) Explain what would motivate a CTO to write such a memo, and what JSP designers ought to be using instead of Scriptlets, Expressions and Declarations.

     If you need more room, continue on the space next to the memo on the following page.



Interoffice Memo from the CTO

URGENT

Effective immediately, anyone caught using scriptlets, expressions, or declarations in their JSP code will be suspended without pay until such time as it can be determined whether the programmer was fully responsible or simply trying to maintain some OTHER idiot's code.

If, in fact, the determination is made that the programmer is, in fact, responsible, the company will go ahead and, in fact, terminate the employee.

Rick Forester
Chief Technology Officer

"Remember: there is no "I" in TEAM."

"Write your code as if the next guy* to maintain it is a homicidal maniac who knows where you live."

[*Note to HR: we use "guy" in its non-gender specific form.]

4. (10 pts) Albert Einstein is often quoted as having said: You do not really understand something unless you can explain it to your grandmother. This question is in that spirit.

   Your friend "Chris" has asked for your help. Chris is a smart businessman, but is not particularly tech-savvy. Chris wants to hire someone to do some maintenance on the web site for his company. He knows that the current site is based on "JSPs" but he doesn't really know what those are. He tried to hire you, but your work load is too much at the moment; you know you don't really have time to help him out. However, you do agree to sit in on the interviews of the web developers who have applied for the job.

   The first person to apply has a really nice looking portfolio. The web sites he says he developed look great. Your friend is very impressed as he looks through the sites the candidate shows him.

   So, your friend asks you if you have any questions for the candidate. You start with an easy one, just to break the ice: "Tell me, how did you learn HTML?"

   He says, "well, to be honest, I never took a formal class. I just did a 'view source' on a bunch of web pages that I thought looked good, and sort of figured out how they worked, and picked up HTML that way. As you can see, my web pages look pretty snazzy, even though I never took a formal class. Here see, you can do 'view source' on any web page and read how it is coded." At this the candidate shows a view source, and shows your friend how the tags line up with some of the elements on the rendered page. Your friend is getting more and more impressed.

   You reply: "Very clever. Tell me though, my friend also needs someone who can maintain Java Server Pages... you know JSPs. You've shown us a bunch of JSP based sites as part of your portfolio. Did you create those sites too?"

   The candidate replies, proudly: "Yes, I sure did".

   You respond: "How did you learn JSP?"

   He replies: " Exactly the same way... just found a bunch of sites that used JSPs, and then did "view source" and figured out they worked."

   You respond: "Really? You learned not just static HTML, but also *JSPs* by doing 'view source'? Are you sure?"

   He replies: "Yup, sure did. Took me a while to figure it out the JSP part, but eventually I got it."

   You reply: "Ah, very good... well I think that's all we need to know."

   Your friend is all smiles, and you and he both thank the candidate for stopping by. Once the candidate leaves the room, you turn to your friend and say: "That candidate is a bald faced liar, and technically incompetent. Under no circumstances should you hire him."

   Your friend is taken aback. "That's a pretty serious charge. How do you know this after only asking him four questions?"

   What do you tell your friend?

Note: If you understand how JSPs work, it pretty simple to work out that the candidate is lying, and furthermore that he is not technically competent enough to realize that the lie is easy to spot.

However, for this question, a quick explanation in technical terms is NOT sufficient for full credit... having the technical facts correct will only earn you half the points.

For full credit, you'll be graded not only on the technical content of your question, but how well you explain it in terms that a non-technical person can understand. Keep in mind that your friend won't understand terms like "compiler", "container", "source code", etc. You friend isn't stupid... he's a very successful businessman. But you won't communicate with him if you talk in "geek speak". And it is important that he fully understands... "trust me, I'm an expert" is not going to cut it here.

Let's assume, to make things easier, that your friend did take one computer class in programming in BASIC a zillion years ago, and vaguely remembers things like a "for loop" and "input" and "output". So you can use that as a point of reference. He's also seen the HTML code... the candidate did a view source, and showed him how the tags like <TABLE> correspond with the table on the page. He also understands that a computer has files on it, and that the files are stored on the hard disk. He sort of gets, too, that information on web pages comes from some place called a "server" out there on the network somehow, and makes its way to his computer, though he is very hazy on the details.

That's all you have to go on. Good luck!

There is extra space on the next page in case you need it.

Extra space in case you need it

5. (10 pts) Write the XHTML code for 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.

   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. You don't need to specify the DTD declaration at the top of the page, but you should follow the rules of strict XHTML (v1.0 strict, or v1.1).

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.

   - (3 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.

   - (3 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.

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

   - (3 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");
    ```

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

Total Points: 100

CISC474 - Spring 2006

Advanced Web Technologies - Final Exam 5/24/2006

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 two hours 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. In web applications, The technique known as “URL rewriting” is typically employed as a “backup strategy” in case a particular “preferred” technique fails.

   - (5 pts) What is the “purpose” of URL rewriting, and what is the preferred technique that is generally tried first?

   Answer:
   - URL rewriting is a backup to cookies for session management.

2. (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:

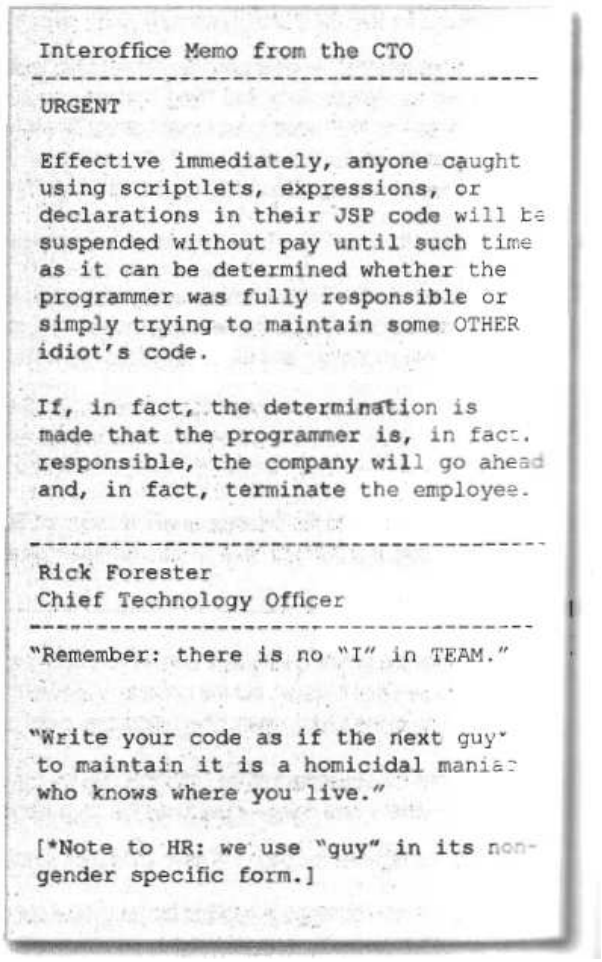
   > Suppose you could go hunting through source code at will (regardless of whether it is publicly available or not.) Where would you find the source code for the main() method that starts this “other application”?

   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.

3. Consider the memo from the textbook, p. 314 (reproduced on the following page to refresh your memory).

   - (9 pts) The memo bans the use of Scriptlets, Expressions, and Declarations in JSPs. Give an example of each (in the context intended by this memo, i.e. as they would appear in a JSP.)
     If you need more room, continue on the space next to the memo on the following page.
   - (5 pts) Explain what would motivate a CTO to write such a memo, and what JSP designers ought to be using instead of Scriptlets, Expressions and Declarations.

If you need more room, continue on the space next to the memo on the following page.



Interoffice Memo from the CTO

URGENT

Effective immediately, anyone caught using scriptlets, expressions, or declarations in their JSP code will be suspended without pay until such time as it can be determined whether the programmer was fully responsible or simply trying to maintain some OTHER idiot's code.

If, in fact, the determination is made that the programmer is, in fact, responsible, the company will go ahead and, in fact, terminate the employee.

Rick Forester
Chief Technology Officer

"Remember: there is no "I" in TEAM."

"Write your code as if the next guy* to maintain it is a homicidal maniac who knows where you live."

[*Note to HR: we use "guy" in its non-gender specific form.]

Answer:

- Scriplets, Expressions and JSPs:
  - Scriptlets are bits of Java embedded in a JSP. They appear between <% and %> tags.
  - Expressions are Java Expressions embedded in a JSP that get passed into out.println and placed in the response stream. They appear between <%= and %> tags.
  - Declarations are declartions of instance variables for the Servlet class that is created from the JSP by the Container. They appear between <%! and %> tags.
- A CTO would write such a memo because Java code inside a JSP is difficult to test and maintain. There may be legacy JSPs that contain scriptlets, etc., but new code development should use the Expression Language (EL) and tag libraries (e.g. the JSTL). This keeps Java code in Java files, and keeps web files containing only things that look like tags. The EL and tag libraries can easily be tied to Java classes, and are much more easily understood by Web designers that are used to HTML like syntax, not program syntax.

4. (10 pts) Albert Einstein is often quoted as having said: You do not really understand something unless you can explain it to your grandmother. This question is in that spirit.

   Your friend "Chris" has asked for your help. Chris is a smart businessman, but is not particularly tech-savvy. Chris wants to hire someone to do some maintenance on the web site for his company. He knows that the current site is based on "JSPs" but he doesn't really know what those are. He tried to hire you, but your work load is too much at the moment; you know you don't really have time to help him out. However, you do agree to sit in on the interviews of the web developers who have applied for the job.

   The first person to apply has a really nice looking portfolio. The web sites he says he developed look great. Your friend is very impressed as he looks through the sites the candidate shows him.

   So, your friend asks you if you have any questions for the candidate. You start with an easy one, just to break the ice: "Tell me, how did you learn HTML?"

   He says, "well, to be honest, I never took a formal class. I just did a 'view source' on a bunch of web pages that I thought looked good, and sort of figured out how they worked, and picked up HTML that way. As you can see, my web pages look pretty snazzy, even though I never took a formal class. Here see, you can do 'view source' on any web page and read how it is coded." At this the candidate shows a view source, and shows your friend how the tags line up with some of the elements on the rendered page. Your friend is getting more and more impressed.

   You reply: "Very clever. Tell me though, my friend also needs someone who can maintain Java Server Pages... you know JSPs. You've shown us a bunch of JSP based sites as part of your portfolio. Did you create those sites too?"

   The candidate replies, proudly: "Yes, I sure did".

   You respond: "How did you learn JSP?"

   He replies: " Exactly the same way... just found a bunch of sites that used JSPs, and then did "view source" and figured out they worked."

   You respond: "Really? You learned not just static HTML, but also *JSPs* by doing 'view source'? Are you sure?"

   He replies: "Yup, sure did. Took me a while to figure it out the JSP part, but eventually I got it."

   You reply: "Ah, very good... well I think that's all we need to know."

   Your friend is all smiles, and you and he both thank the candidate for stopping by. Once the candidate leaves the room, you turn to your friend and say: "That candidate is a bald faced liar, and technically incompetent. Under no circumstances should you hire him."

   Your friend is taken aback. "That's a pretty serious charge. How do you know this after only asking him four questions?"

   What do you tell your friend?

Note: If you understand how JSPs work, it pretty simple to work out that the candidate is lying, and furthermore that he is not technically competent enough to realize that the lie is easy to spot.

However, for this question, a quick explanation in technical terms is NOT sufficient for full credit... having the technical facts correct will only earn you half the points.

For full credit, you'll be graded not only on the technical content of your question, but how well you explain it in terms that a non-technical person can understand. Keep in mind that your friend won't understand terms like "compiler", "container", "source code", etc. You friend isn't stupid... he's a very successful businessman. But you won't communicate with him if you talk in "geek speak". And it is important that he fully understands... "trust me, I'm an expert" is not going to cut it here.

Let's assume, to make things easier, that your friend did take one computer class in programming in BASIC a zillion years ago, and vaguely remembers things like a "for loop" and "input" and "output". So you can use that as a point of reference. He's also seen the HTML code... the candidate did a view source, and showed him how the tags like <TABLE> correspond with the table on the page. He also understands that a computer has files on it, and that the files are stored on the hard disk. He sort of gets, too, that information on web pages comes from some place called a "server" out there on the network somehow, and makes its way to his computer, though he is very hazy on the details.

That's all you have to go on. Good luck!

There is extra space on the next page in case you need it.

Extra space in case you need it

Answer: The technical explanation is that when you do a “view source” on a JSP, you aren’t seeing the source code of the JSP, but rather the HTML that is “computed” by that JSP. So it is impossible to learn how to code a JSP from doing “view source”... you never actually get to see the JSP code, only the computed HTML. That technical explanation is worth half credit.

A non-technical explanation might proceed as follows:

There are two types of web pages. Some web pages are “static”—every time you look at them, they are the same, while other web pages are dynamic—every time you look at them they are different.

The static ones have HTML codes that just sit there on the hard disk on the server. When you ask for the page, the server gives you that HTML code, exactly. When you do a view source, you see exactly what was on the server. So, it is reasonable that a really clever person MIGHT be able to learn static HTML coding from doing lots of view source.

Dynamic pages, though, come from a computer program that is written in a computer language other than HTML. It is called JSP, and it is actually a mixture of a couple of different languages.. HTML plus a language called Java. Java is a lot like the BASIC programs you wrote back in school. Every time you go to the dynamic page, the program runs–on the server, not on your local machine—and spits out some HTML. That HTML is sent to your computer and displayed.

You can do a view source on your local machine, but if you do, you are only seeing what the program spit out. You are NOT seeing the instructions in the program itself.. the JSP instructions. Those are on the server, and are hidden from view. There is no way to get access to them over the network unless you actually break into the server. But your candidate never claimed he has access to the server... he said that he learned JSP from doing “view source”. You definitely CAN’T see them with a “view source”.

So this programmer claims to have learned JSPs by doing view source. But that is impossible. So he is lying. Furthermore, he doesn’t even realize how easy his lie is to spot.. anyone who did understand how JSPs work would realize how stupid it is to claim that you can learn JSP coding from doing a view source. That’s why I claim that he is technically incompetent.

So sorry. Hope your next candidate is better!”

5. (10 pts) Write the XHTML code for 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.

   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. You don't need to specify the DTD declaration at the top of the page, but you should follow the rules of strict XHTML (v1.0 strict, or v1.1).

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

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.

   - (3 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.
   - (3 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.
   - (3 pts) Which part (M, V, or C) would typically be written as a Java class NOT derived from HttpServlet? Explain why.
   - (3 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");
    ```

    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.

Total Points: 100