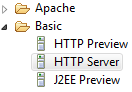
Creating the Web View

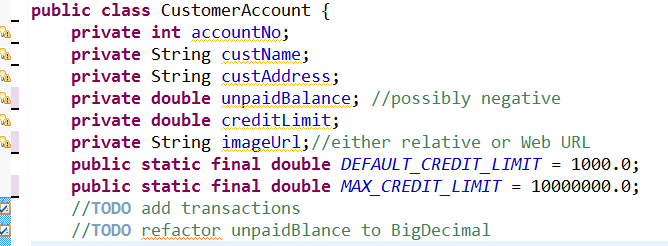
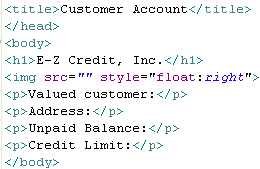
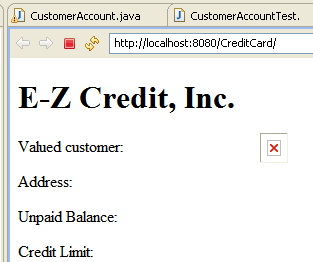
# CSCI 4300

Now we will show how to display a Java “business object” as a Web page. Before beginning, we need to make sure your computer has a Java Web server installed. Eclipse comes equipped with a built-in basic Web server called J2EE preview, but this server won’t handle some of the more sophisticated technologies we will use here.

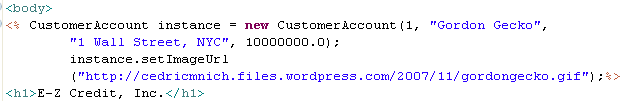
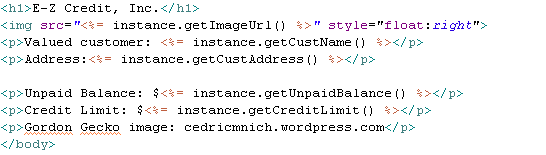
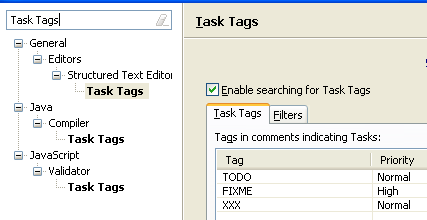
Before starting this exercise: You should have installed the Tomcat Web server on your computer. If this was unsuccessful, select **J2EE Preview** in step 2 below. At some point (as in very soon!) you will need to have a working Tomcat instance, either on your computer or on another server. If you are still struggling with this issue, please see me and we will try to get you up and running with Tomcat.

1. Launch Eclipse and create a Dynamic Web project called CreditCard. Check the package icon for the  project. Make sure you see the little globe above the folder, which means you are in a Web project. [[1]](#footnote-1) If you are in the J2EE perspective, then you should see the Servers tab. If not, use the **Window/Show View** menu item to open the Servers view.
2. In the Servers window, right-click the empty space and choose New/Server. If you were successful installing Tomcat, open the Apache folder and choose the Tomcat version that you installed (Tomcat v7.0 is the version we used the last time this exercise was updated). As part of the installation, you may be asked to choose the Tomcat home directory. This will be a directory whose name includes “apache-tomcat” and which contains subdirectories **bin**, **conf**, **logs**, and **webapps**.

Now that the configuration is set up, we are ready to actually write code!

1.  Define a class **CustomerAccount** in package **creditcard** with the fields shown at right*. Brain teaser quiz*: why are two of these fields declared as *static*?
2. Right-click the CustomerAccount.java node and choose Source/Generate Getters and Setters. Generate getter methods with public access for all of the non-static fields. (The static fields are already public!). These are necessary to allow the Web page that we are about to create to read the object’s data. Later we will discuss the setter methods that might be appropriate for this class – they are not what you might think!
3. We use the term ***view*** to denote a Web page that displays application data. In Project Explorer, right-click on the  project node and choose **New/JSP File**. Name the file index.jsp. This is the default welcome file, which is the Web page that will show up when we run the project as a Web app.
4. You should now see the HTML source code for the basic JSP page.( If you’re not familiar with HTML, just follow along and we’ll explain later.)There arise also a tag starting with <%@ on the first line. All tags starting with <% are **JSP tags**, which will be converted to Java code on the server. The Web browser never sees these tags.
5. As you can see, the <%@page> tag contains general information describing this page. As with all tags, the information is in the form of a set of name=value pairs, separated by spaces. Each of these pairs is called an **attribute**. The attribute name is not in quotes, but the value always is.
6. In order for this JSP page to recognize our CustomerAccount class, we need to import the class by adding the following attribute to the end of the <%@page > tag. Note that Java classes are referred to using the package name and class name. This type of compound name is called a **qualified name**.
7. Next, edit the title and body of the file to add basic HTML labels as shown. If you’re good at HTML and CSS, feel free to jazz the display up a bit. Save all the files when you are done.
8. Let’s see what this looks like! Run the application by right-clicking the project node and choosing Run As/ Run on Server. If you’re asked to choose a server, choose ***J2EE Preview*** (unless you have already installed the Tomcat server, in which case good for you!). This operation can take quite a few seconds to fire up the server.
9. The result should look something like this. Note the URL:
   1. localhost is your computer;
   2. 8080 is the TCP port number on which your Tomcat Web server is running (this might be different depending on your installation).

The red X represents the missing image.

1. Click the white X on the Web browser window to close the browser.
2. Now, let’s create the actual customer data, in the form of a CustomerAccount object, by adding a ***Java Scriptlet*** as shown below. Note the special <% and %> tags: these JSP tags will be consumed on the server and will not appear in the output code. 
3. Run the application again. You should see exactly the same output, since we have not specified that the data values in our CustomerAccount object should be displayed to the page.
4. Now we will add Java **Expression Scriptlets** to actually display the data. Note the <%= opening tag and the “object dot method” notation. Also note that Java Expressions, unlike statements, are not terminated by semicolons. Note finally that we have included an attribution for the image content. If this were a real application for public display, we should check for potential copyright violations.
5. Run the application again.[[2]](#footnote-2) Now you should see the actual customer data! Unfortunately Gordon Gecko’s pic is too big: you can fix this, if you like, by adding height and width attributes to the image tag. I chose these height and width values to be proportional to the original image dimensions, to avoid distorting the image.
6. We still have a bit of work to do:
   1. The credit limit is in a clunky exponential format;
   2. We don’t have the ability to add purchases to this account.
7. To leave these unfinished items for future iterations, we simply make sure there are TODO comments to indicate the undone task. We already have two Java //TODO comments in the CustomerAccount source code. Let’s add a JSP comment: to the index.jsp file:
8. Can you see the TODOs in your Task view? First, use the **Window/Show View** command to display the Task view – it should attach itself as a tag in the bottom window. If you don’t see all your TODO tags, you can configure the Task pane by using the Window/Preferences command and searching for Task Tags. Note that in this configuration, a //FIXME comment will rise to the top of your TODO list!
9. If you’ve got this done, think about how to address the TODO issue about the money formats. Hint: start by looking up

That’s it! Save your work and rest. If you have been working with a partner, ask in class about how to share your files among partners.

1. If you have created a Java Project for the Credit Card example in another class, it will not work on the Web.. You must create a new Dynamic Web project . In this case, declare your CustomerAccount and other classes in the new project, then copy the source code from the existing classes to your newly declared classes. [↑](#footnote-ref-1)
2. Eclipse has a known issue where sometimes after you have changed a JSP file and saved the changes, the server still displays the results of compiling the old JSP. When this happens, open the Project menu and choose clean, which should destroy all the old compiled code. If this still does not wor, the tried and true remedy is to stop and restart Eclipse. You needed a break anyway! [↑](#footnote-ref-2)