Introduction

This assignment is the fourth of four assignments, in which you will complete the web site for your term project. It is meant to give you practice in accessing a database to generate pages on your site.

Specifically, you will create a very simple database table listing the products/services/items your web site allows a user to choose, and link your “Products” page to that table.

Note that I am not requiring that everything about your site now involve a database (this would probably be too complex for a simple project like this). However, you will be required to modify some other part of your project to involve the database (more details are given below).

Basic Database Table Structure

Your “products” and “details” pages that I required in the first assignment currently display “hardwired” information about the choices available to a user of your site.

You are to create a simple database with at least one table, which stores this information. Each record should correspond to an item type the user can select from, and should have the following fields:

- A unique ID (the one you should already be using to store items in your cart).
- A name of some sort (which you probably already are using in your “products” page).
- An additional field for every other piece of information (price, etc.) that you display in the “product” or “details” page.

To simplify the grading for this project, you are to use the following databases, depending on what kind of project you are doing:

- A Derby database if you are using JSF.
- An Access database if you are using ASP.

Linking the “Products” Page to the Database Table

The main exercise for this project is to read the contents of your main “products” page (that is, the list of choice the user has) from the above database table.
These choices are currently “hardwired” in a support class you have already developed. In the previous assignment, you have implemented the display as either a DataTable or a GridView.

You are to modify this page to pull this list from your database table. Specifically:

- If you are working in JSF, you will need to use JDBC in one of your support classes to connect to the database, query for all products, and construct an object from each field of the corresponding record.

- If you are working in ASP, you will need to link your “products” gridview to the database table.

Note that if you display images as part of your “products” page, you are not required to do so in this version of your project (unless you decide to do so as your “additional database access”).

**Additional Database Access**

You are to implement one other type of database access in your project. The type of database access you create should be based on what is most appropriate for your project, but could include one of the following:

**Generating “Details” Page from the Database**

Your “details” pages display details about an “item” selected by the user, currently implemented as a set of “hardwired” pages.

You could replace this with a single page, generated from information read from the record corresponding to that “item”. More specifically, when you redirect to the details page, the ID of the corresponding item would be passed. The page would then use the ID to query for all of the details to display on the page.

**Storing the Cart in the Database**

In the previous assignment, you created a “cart”, where you store the items selected by the user. You could store this cart in your database as well, which would allow the user to retrieve it the next time they logged in.

See the lecture I gave on storing carts for more details on how this might be done.

**Retrieving Images from the Database**

A number of you have displayed product images in your “products” page (which is a really nice idea). You could continue to do so by storing the relative URL of the image as a string in an additional table field.
Your “products” page would need to then need to read the URL along with the rest of the product information, and then display it as an image:

- If you are using JSF, you will need to insert the URL into an IMG tag.
- If you are using ASP, you will to add an ImageField in the Column Editor with the DataImageURLField set to that URL field.

### Storing Other Information in the Database

Many of you are creating sites where the user is expected to enter information for future use (for example, a product review site).

You could store the text entered by the user in an **additional table**, one indexed by **user ID** (which the user would provide) and **question ID**. This information could then be displayed from the database table, by looking it up based on the user ID and the question ID.

### What to Turn In

You are to **email me** your web site files. My email is [jrsullins@ysu.edu](mailto:jrsullins@ysu.edu).

As before, you can find the source code as a subdirectory of the NewBeans Projects directory of Documents, or as a subdirectory of the Visual Studio 2013/WebSites directory of Documents. Please use a zip program such as gzip to compress the entire directory and send it to me.

You must also be sure to send your **database** as well:

- If you are working in ASP, it should be automatically included in the directory if you have put the Access database in the APP_DATA directory of the solution.

- If you are working in JSF, you can find your Derby database in the C:\Users\yourname\.netbeans-derby directory. Each database is a subdirectory of this. Make sure to compress the directory, and send it as an additional attachment.