Application Developers' Intro School

Specifications Binder

Segment 1

Table of Contents

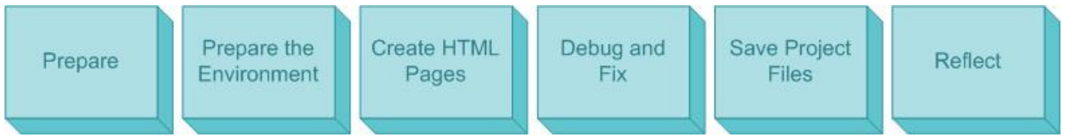
Activity Instructions

Overview

In this segment, you are expected to create all the related HTML pages for the usage inquiry module of the customer service online suite of applications. You will create the HTML pages according to the given specs, verify that it meets the requirements and debug the application accordingly.

This activity is designed to introduce you to the IntelliJ IDEA integrated development environment for application development, basic HTML and cascading style sheets, along with processes and standards for development and testing.

Activity Steps



To complete the activity you need to follow the steps above. These steps are explained later in greater detail in this section of the binder.

Key Learning Points

At the end of the segment, you need to learn and understand:

- How to build Web pages using HTML
- What technical specifications contain, and the importance of using technical specifications to code Web pages
- Using the IntelliJ toolset for application development
- What version control is, and how to use it in IntelliJ
- The basics of GUI development

Learning Materials

The following information may be found within this binder, the Reference Guide or the Performance Support Tool. You do not need to thoroughly understand everything in these resources at this time. Familiarize yourself with what the resources contain and how they will be used. This section describes the purpose and location of each document.

Input

What	Why	Where
Activity Instructions	Guides you through the process of completing the activity	Activity Instructions section in this binder
Technical Specifications	Used as a blueprint for the GUI design and page logic	Technical Specifications section in this binder

Reference Material

What	Why	Where
GUI Best Practices	Provides information on the common elements of a user-interface	GUI Best Practices in the Performance Support Tool
Getting Started with intelliJ IDEA	Provides basic background and "how to" information to get started using intelliJ	IntelliJ Help Prepare the Environment Activity Steps in this Binder
HTML & CSS Syntax	Provides HTML and CSS syntax and examples	Reference Guide
Frequently Asked Questions	Provides answers to questions found throughout the Activity Instructions	Frequently Asked Questions in the Performance Support Tool

Outputs

What	Why	Where
Input Page	Enable CSR to input account number of subscriber	Your desktop
Subscriber Info Page	Provide subscriber information for verification	Your desktop
Usage Page Monthly Usage Page	Provide usage information for a customer service call	Your desktop
Error Page No Account Page No Usage Page	Provides the CSR with feedback if the information he/she requests cannot be found	Your desktop

Step 1 – Prepare for this Activity

1. Review the information contained in each tab of the Activity Binder.

   Briefly review the Activity Overview. Browse the sections of the design specifications. Get an overview of the information they contain. You will use these sections extensively when you are coding and debugging so it is helpful to gain a general understanding of these materials.

2. Mark your Reference Guide and Performance Support Tool.

   Throughout this binder you will encounter questions inside text boxes. These are relevant to the tasks at hand. You will find the answers to these questions from the either the Reference Guide or Performance Support Tool provided to you.

3. Confirm your understanding of the application logic.

   To confirm your understanding of the application logic, try to answer the questions below:

   What page is displayed after the user clicks on the submit button from the input form page?

   Subscriber usage is derived from which table/s? What about subscriber monthly usage?

   What external file is used by all the HTML pages? Why?

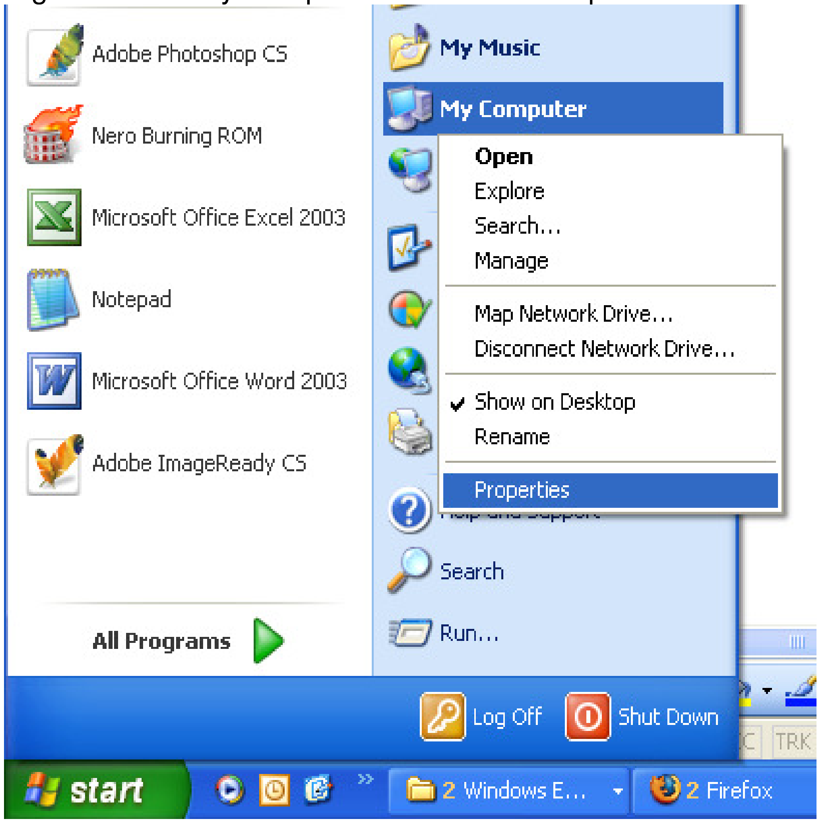
Step 2 – Prepare the Environment

- What is a development environment and how do I learn to use it?

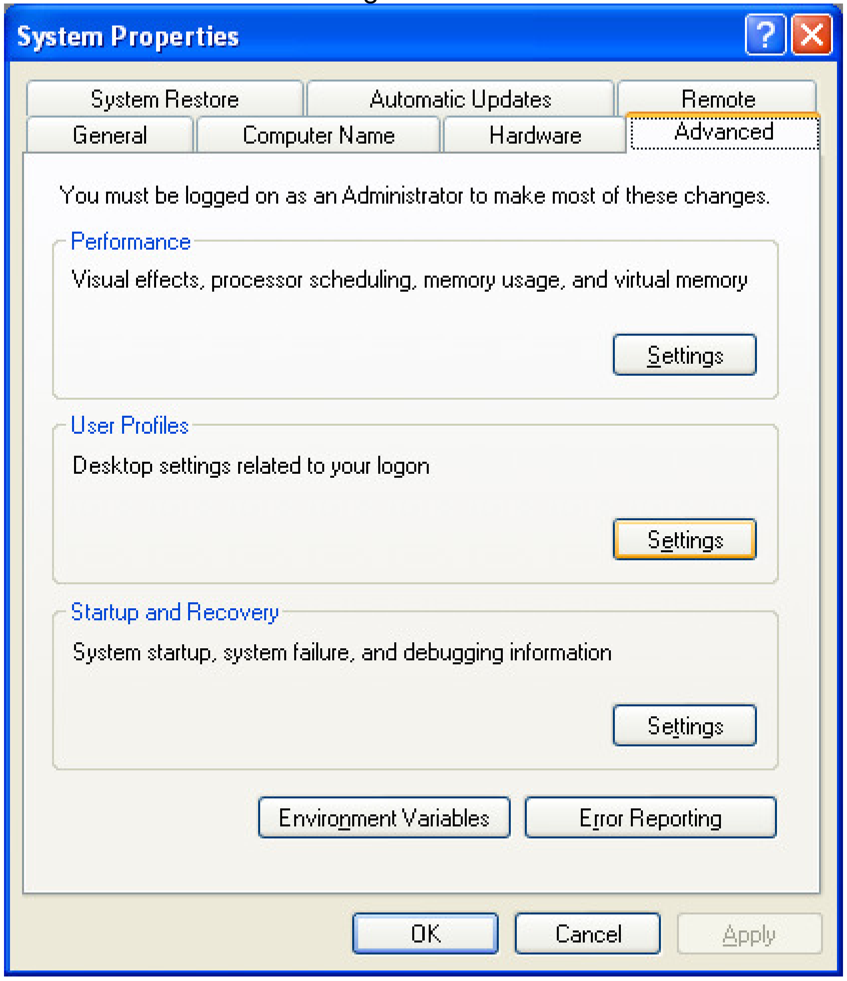
NOTE: It is important that the installation steps be done in the correct sequence.

1. Install the ***Java Development Kit (JDK)***
2. Modify Environment Variables for use by Tomcat.

a. Right click on “My Computer” and select “Properties”



b. Go to “Advanced” and change “Environment Variables”



System Properties

System Restore | Automatic Updates | Remote
General | Computer Name | Hardware | Advanced

You must be logged on as an Administrator to make most of these changes.

Performance

Visual effects, processor scheduling, memory usage, and virtual memory

Settings

User Profiles

Desktop settings related to your logon

Settings

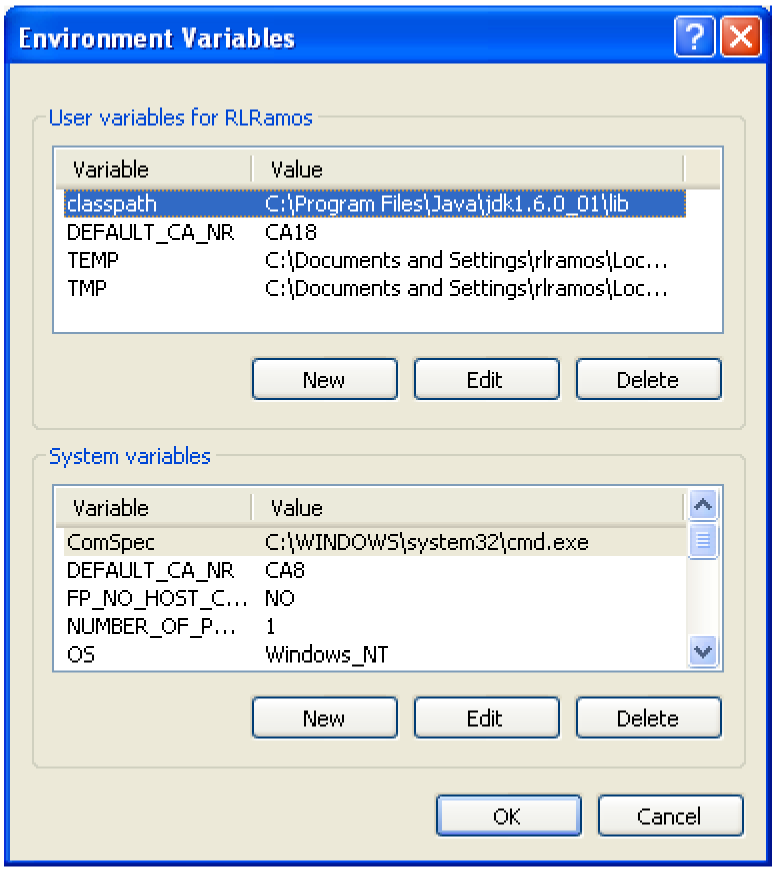
Startup and Recovery

System startup, system failure, and debugging information

Settings

Environment Variables | Error Reporting

OK | Cancel | Apply



Environment Variables

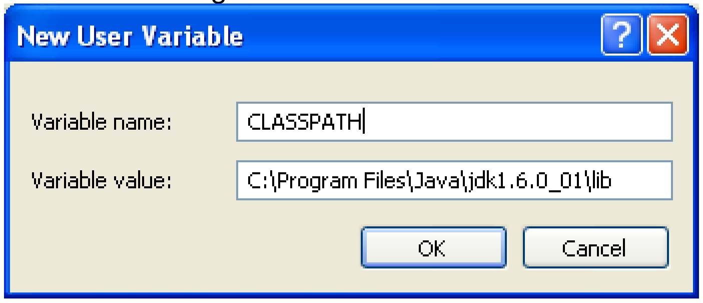
User variables for RLRamos

Variable	Value
classpath	C:\Program Files\Java\jdk1.6.0_01\lib
DEFAULT_CA_NR	CA18
TEMP	C:\Documents and Settings\rlramos\Loc...
TMP	C:\Documents and Settings\rlramos\Loc...

System variables

Variable	Value
ComSpec	C:\WINDOWS\system32\cmd.exe
DEFAULT_CA_NR	CA8
FP_NO_HOST_C...	NO
NUMBER_OF_P...	1
OS	Windows_NT

Add the following USER VARIABLES:

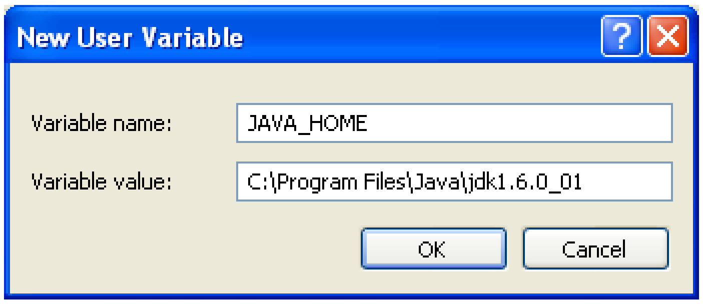


New User Variable

Variable name: CLASSPATH

Variable value: C:\Program Files\Java\jdk1.6.0_01\lib

where CLASSPATH is where your JDK's lib folder is found

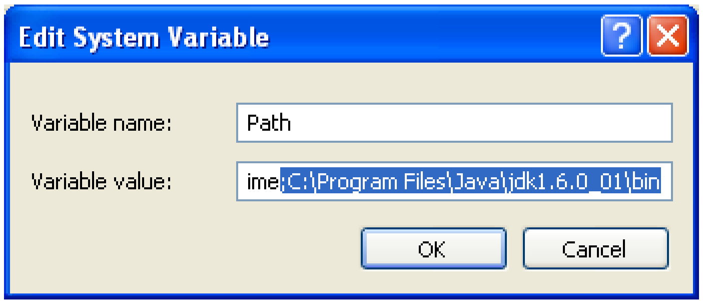

New User Variable

Variable name: JAVA_HOME

Variable value: C:\Program Files\Java\jdk1.6.0_01

where JAVA_HOME is your JDK install directory

Modify the SYSTEM VARIABLE PATH by adding your JDK install directory's bin folder


Edit System Variable

Variable name: Path

Variable value: ime;C:\Program Files\Java\jdk1.6.0_01\bin

NOTE: Be careful editing this value. You only add it at the end of the whole PATH value.

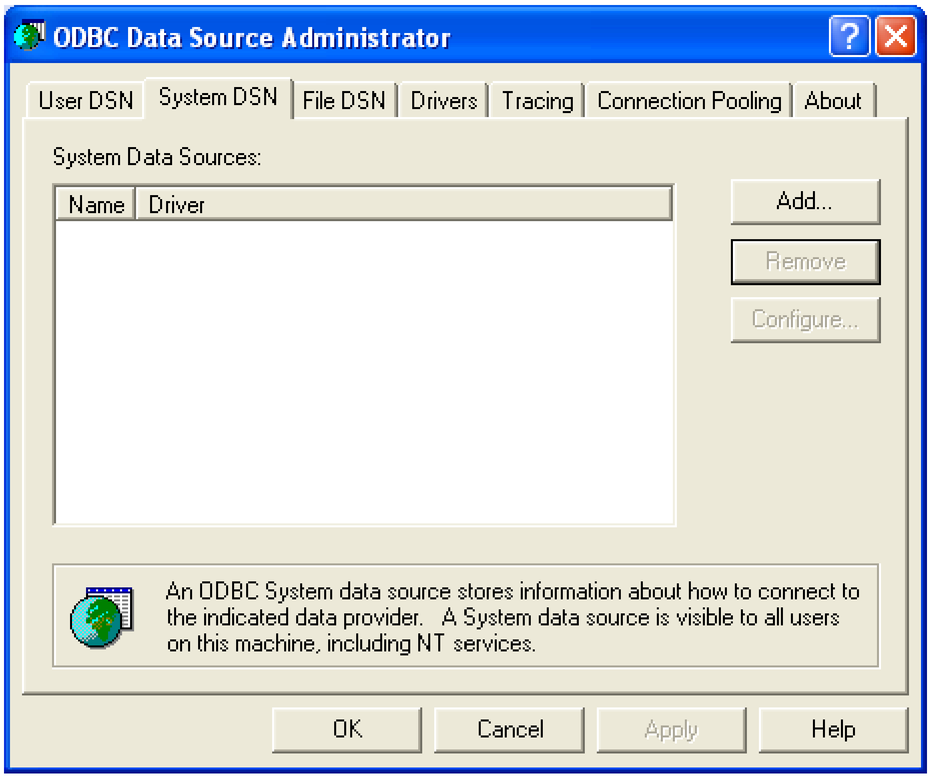
Step Two – Install ***Apache Tomcat***. Just use the default values.

Step Three – Install ***Winrar***. You will use this to extract Eclipse.

Step Four – Extract ***Eclipse*** to a location where you can easily find it.

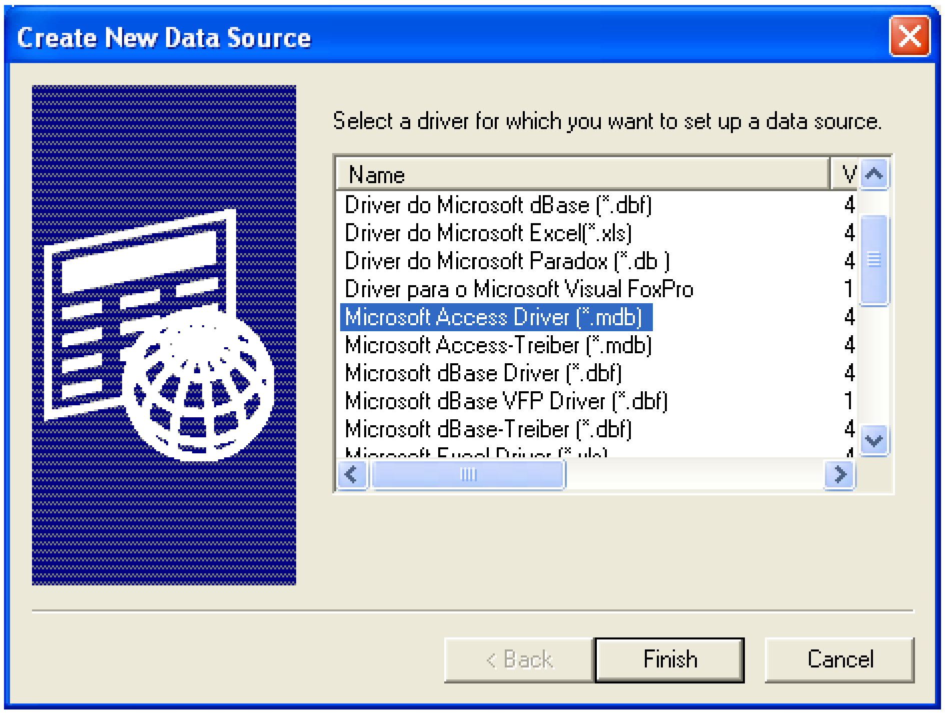
Step Five – Database Set-up

1. Copy the Access Database ***ADIS.mdb*** to your computer.
2. Go to Control Panel → Administrative Tools → Data Sources (ODBC) → System DSN

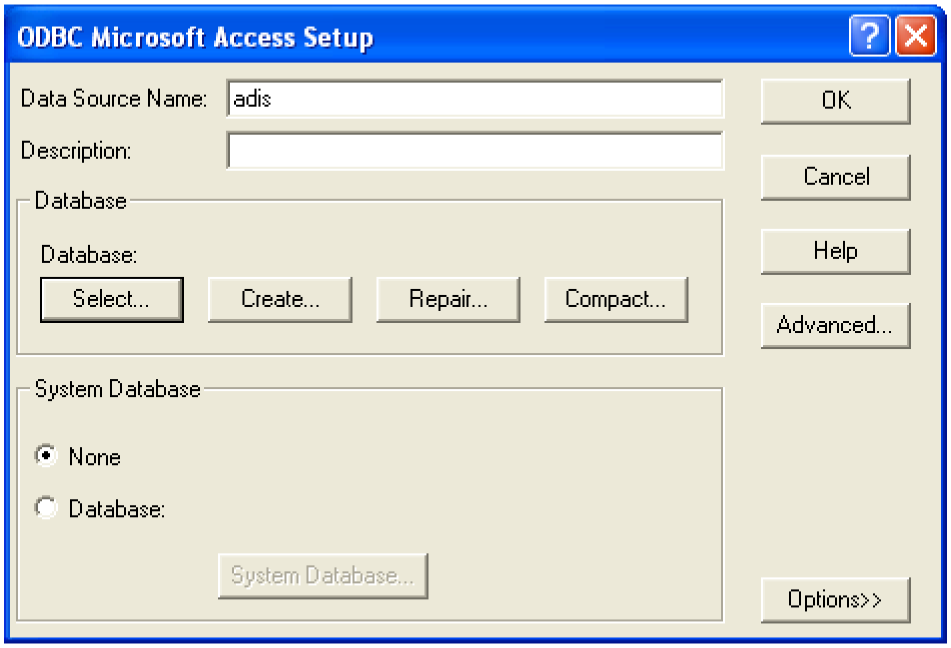


An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

3. Click on ADD and select the Microsoft Access Driver. Then click on Finish.


Create New Data Source
Select a driver for which you want to set up a data source.
Name
V
Driver do Microsoft dBase (*.dbf) 4
Driver do Microsoft Excel(*.xls) 4
Driver do Microsoft Paradox (*.db) 4
Driver para o Microsoft Visual FoxPro 1
Microsoft Access Driver (*.mdb) 4
Microsoft Access-Treiber (*.mdb) 4
Microsoft dBase Driver (*.dbf) 4
Microsoft dBase VFP Driver (*.dbf) 1
Microsoft dBase-Treiber (*.dbf) 4
< Back
Finish
Cancel

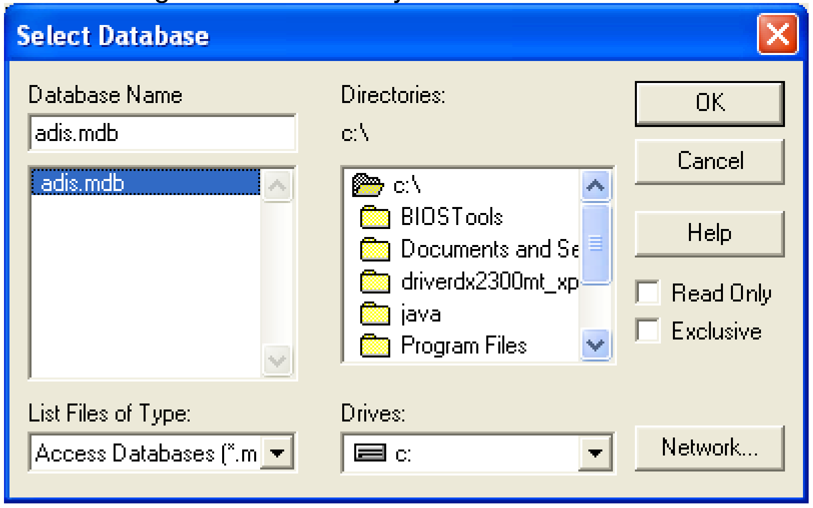
4. You will get the ODBC Set-Up window. Name the data source “adis”. Browse for your adis.mdb file by clicking on “Select”.


ODBC Microsoft Access Setup

Data Source Name: adis

Description:

After clicking “Select” look for your MDB file:



Database Name: adis.mdb

Directories: c:\

- c:\
- BIOSTools
- Documents and Se
- driverdx2300mt_xp
- java
- Program Files

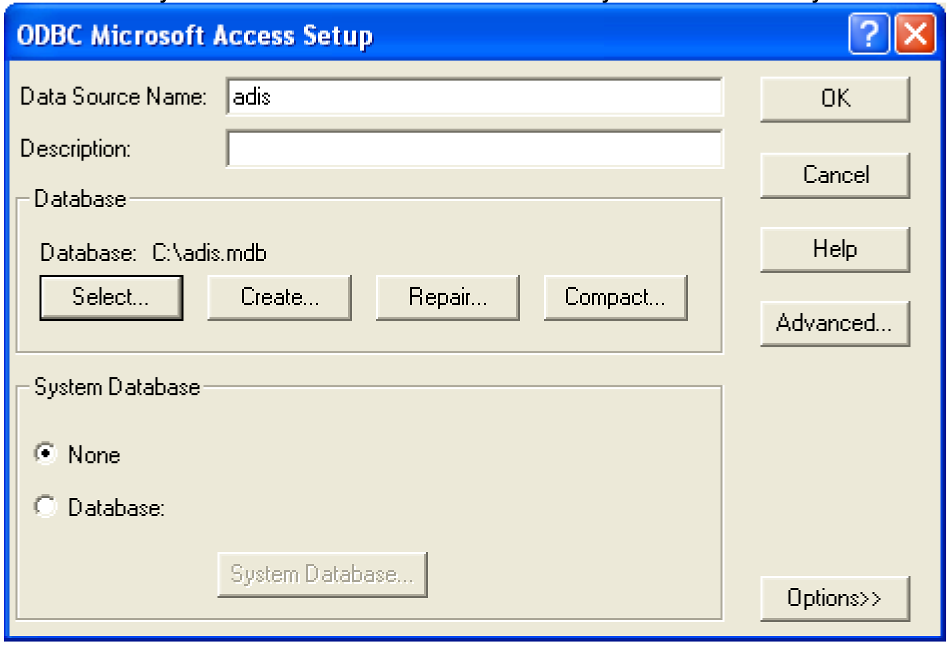
Read Only: [ ]

Exclusive: [ ]

List Files of Type: Access Databases (*.m

Drives: c:

5. This is how your screen should look like once you’ve selected your database:



Data Source Name: adis

Description:

Database

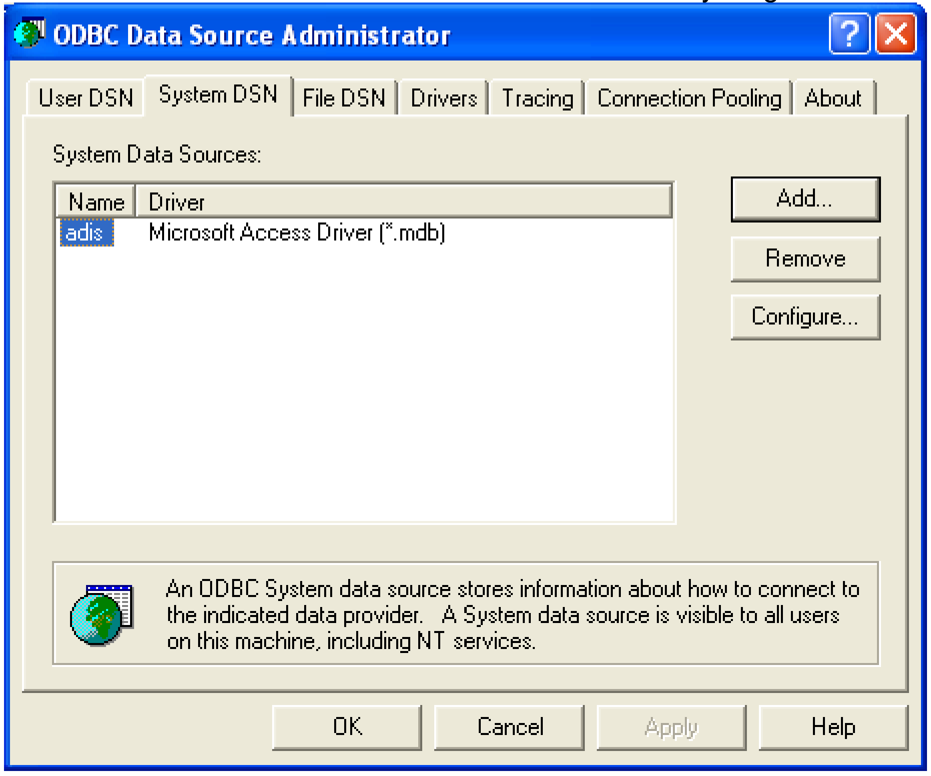
Database: C:\adis.mdb

System Database

[✓ None]

Database:

6. And this is how the ODBC screen looks like after everything has been done:


ODBC Data Source Administrator

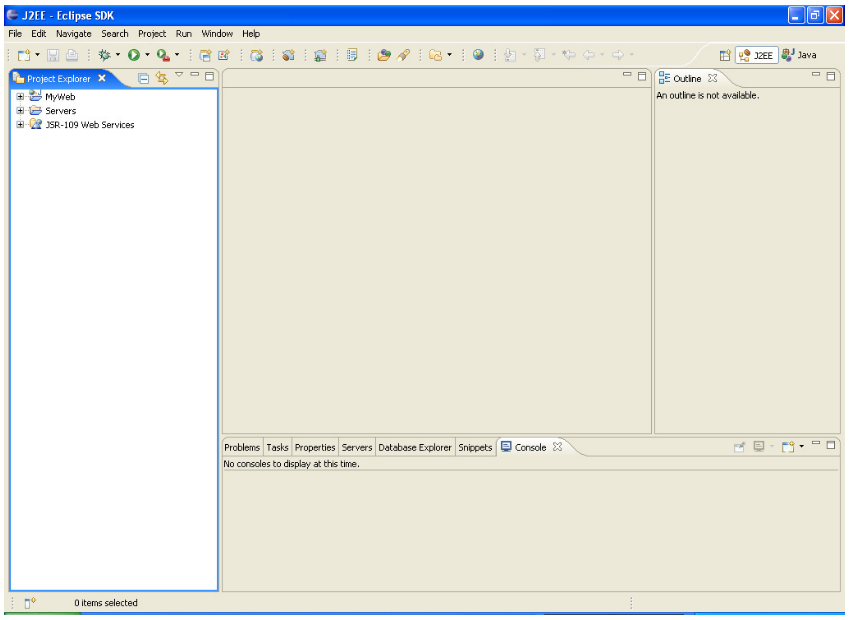
System Data Sources:

Name	Driver
adis	Microsoft Access Driver (*.mdb)

An ODBC System data source stores information about how to connect to the indicated data provider. A System data source is visible to all users on this machine, including NT services.

Preparing the Web Project.

1. Open Eclipse.

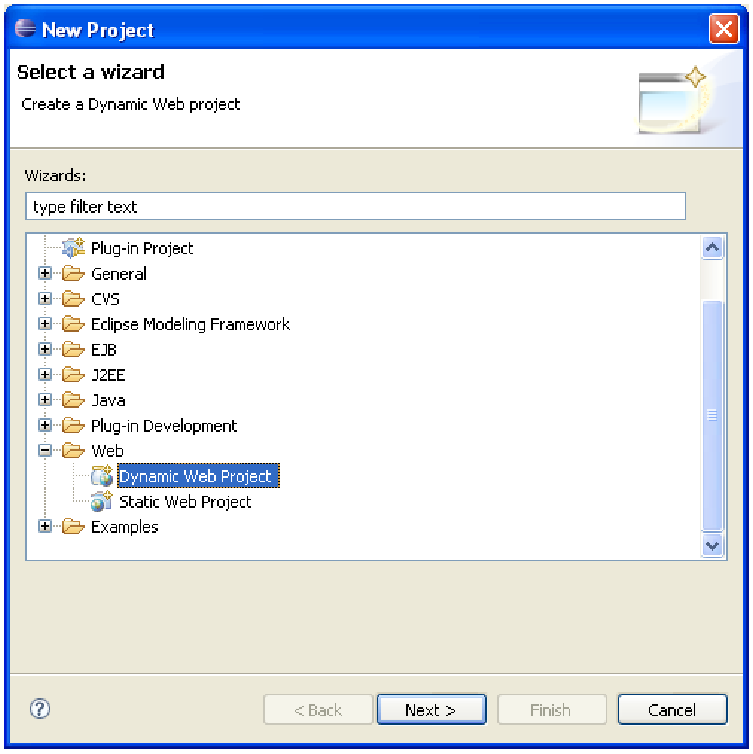


An outline is not available.

No consoles to display at this time.

0 items selected

2. Create a new Dynamic Web Project



Select a wizard

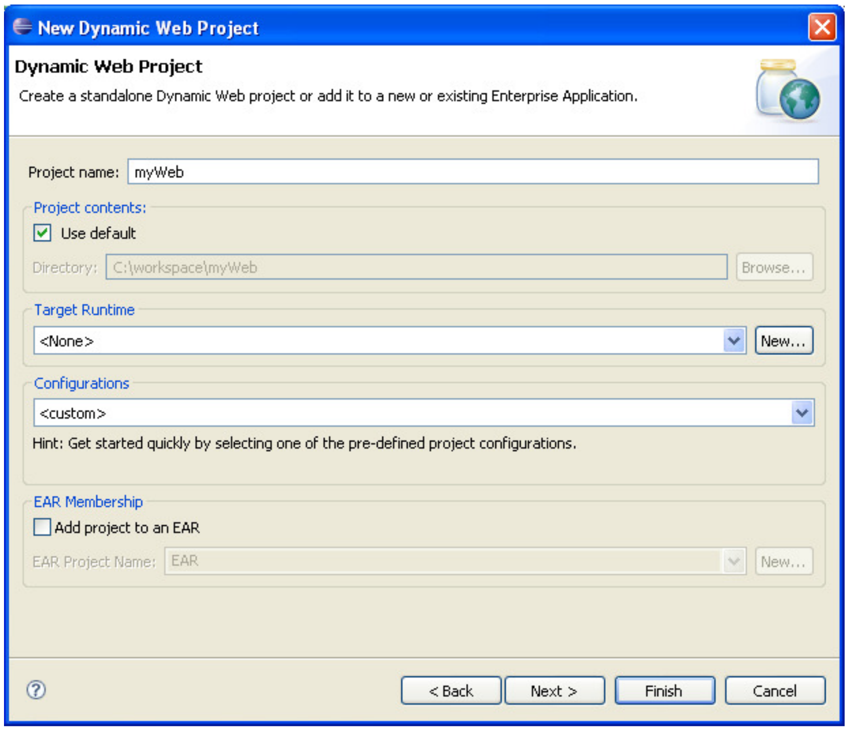
Create a Dynamic Web project

Wizards:

type filter text

- Plug-in Project
- General
- CVS
- Eclipse Modeling Framework
- EJB
- J2EE
- Java
- Plug-in Development
- Web
  - Dynamic Web Project
  - Static Web Project
- Examples

3. Fill in the name of the project.



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: myWeb

Project contents:

☑ Use default

Directory: C:\workspace\myWeb

Target Runtime: <None>

Configurations: <custom>

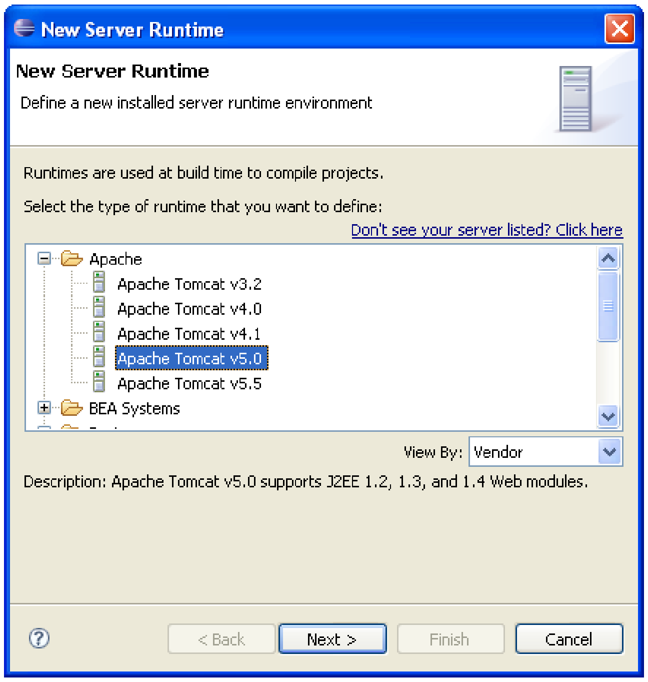
Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR Membership

☐ Add project to an EAR

EAR Project Name: EAR

4. Set-up the Target Runtime. This will be the webserver your project will use. Click on the "New" button to set this up.



New Server Runtime

Define a new installed server runtime environment

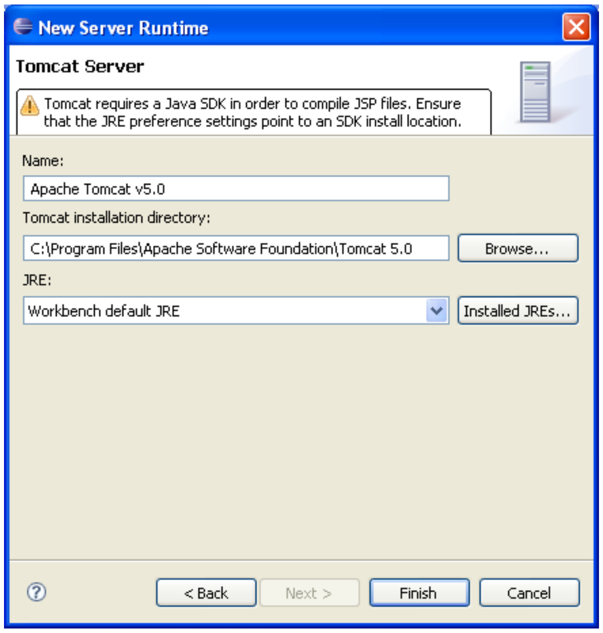
Runtimes are used at build time to compile projects.

Select the type of runtime that you want to define:

View By: Vendor

Description: Apache Tomcat v5.0 supports J2EE 1.2, 1.3, and 1.4 Web modules.

Make sure you select the correct version of Tomcat for this step. Click on "Next".



Tomcat Server

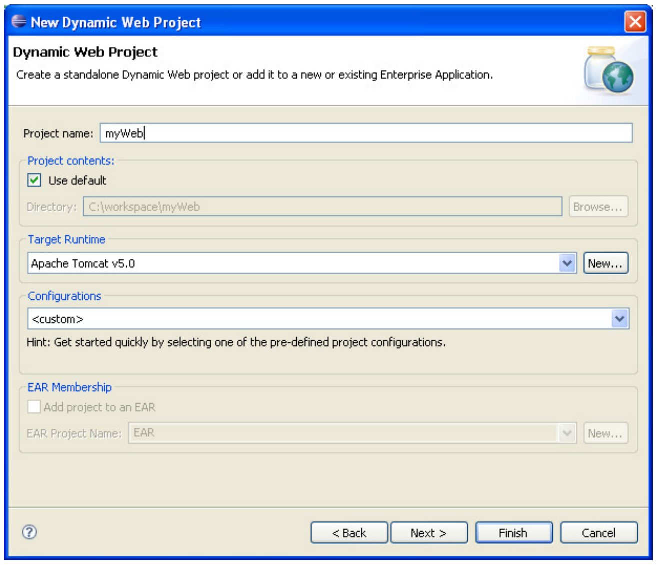
Tomcat requires a Java SDK in order to compile JSP files. Ensure that the JRE preference settings point to an SDK install location.

Name: Apache Tomcat v5.0

Tomcat installation directory: C:\Program Files\Apache Software Foundation\Tomcat 5.0

JRE: Workbench default JRE

Browse to the main folder of your Tomcat installation and click on "Finish".



New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: myWeb

Project contents:
☑ Use default
Directory: C:\workspace\myWeb

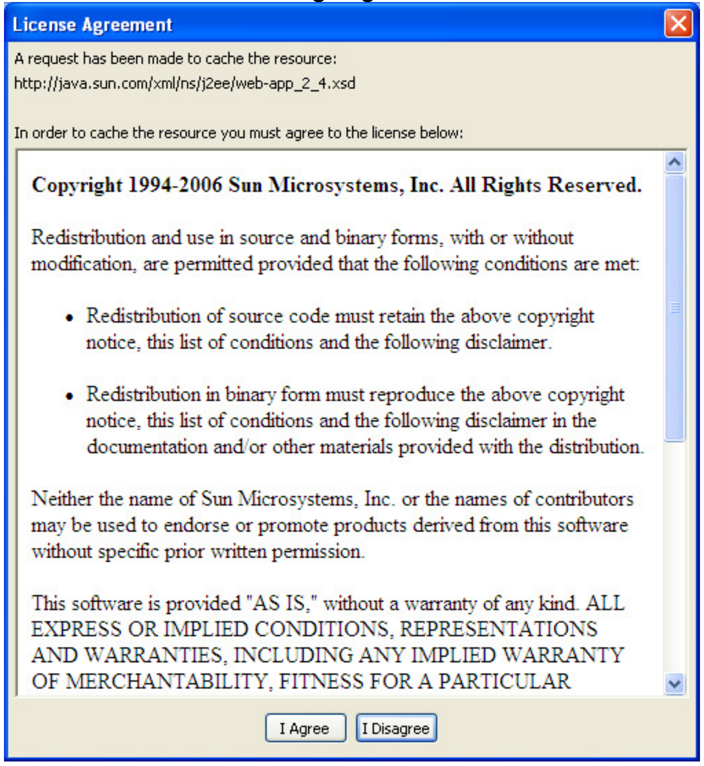
Target Runtime: Apache Tomcat v5.0

Configurations: <custom>

Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR Membership
☐ Add project to an EAR
EAR Project Name: EAR

5. Click on Finish. You might get the screen below. Click on "Agree".



License Agreement

A request has been made to cache the resource:
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd

In order to cache the resource you must agree to the license below:

Copyright 1994-2006 Sun Microsystems, Inc. All Rights Reserved.

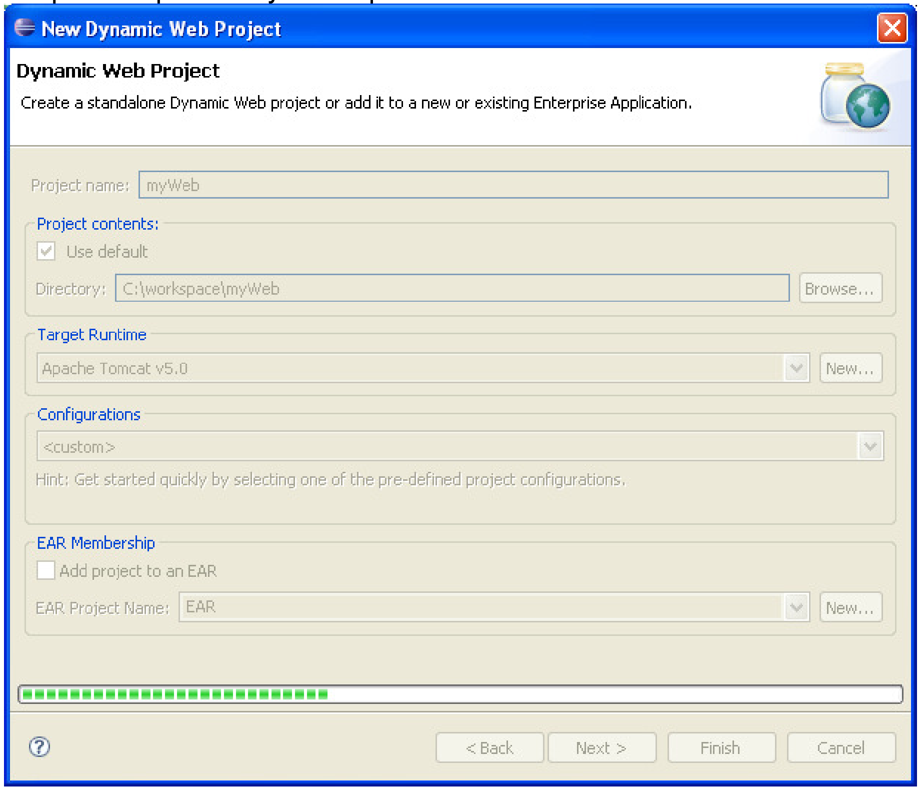
Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of Sun Microsystems, Inc. or the names of contributors may be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided "AS IS," without a warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR

6. Eclipse will process your request.


New Dynamic Web Project

Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Project name: myWeb

Project contents:

Use default

Directory: C:\workspace\myWeb

Target Runtime: Apache Tomcat v5.0

Configurations: <custom>

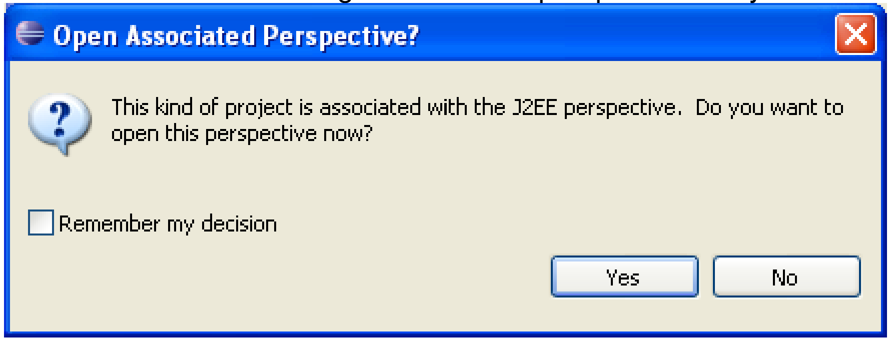
Hint: Get started quickly by selecting one of the pre-defined project configurations.

EAR Membership

Add project to an EAR

EAR Project Name: EAR

7. You will be asked to change to the J2EE perspective. Say “Yes”.


Open Associated Perspective?

This kind of project is associated with the J2EE perspective. Do you want to open this perspective now?

Remember my decision

8. You are now done with the initial steps.

Step 3 – Paint and Code the HTML Pages

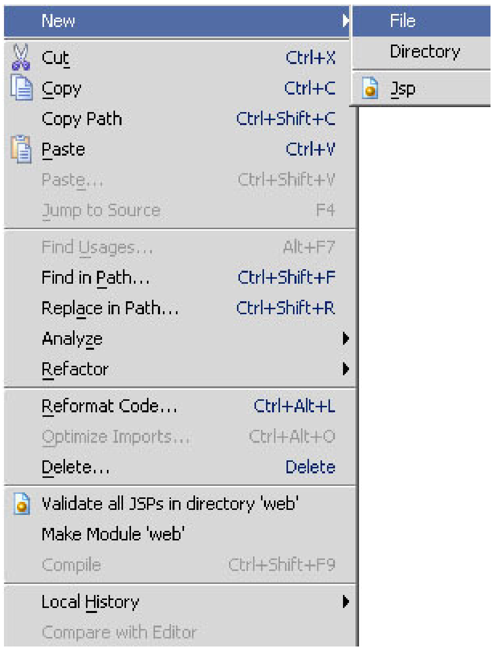
During this step you will build the basic structure of the HTML pages, and create the page's controls (that is, text boxes, labels, etc.). To perform this step, refer to the Page Layout and Page Attributes of the Technical Specifications section.

You may code the following HTML pages in any order and save them in the web resources folder - `c:\appdevschool\data\web`

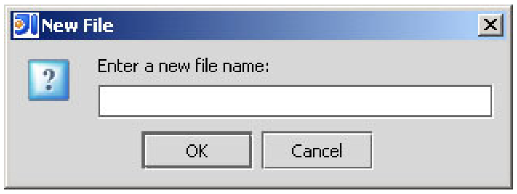
- input_form.html
- subscriber_info.html
- subscriber_usage.html
- subscriber_monthly_usage.html
- sub_info_no_data.html
- sub_usage_no_data.html
- error.html

For every HTML file that you create, do the following steps:

1. In the open IntelliJ web project, right-click on the `web` folder and choose **New-File** from the shortcut menu.



2. Type the specified filename including the filename extension (e.g. error.html) and click **OK**.



3. Type the HTML tags as you would in a word processor.
4. From the **File** menu, select **Save All** to save your work.

- What is HTML?
- Where do I get help on HTML syntax?
- Why am I using a form tag?
- How is an HTML page structured?
- How do I link files to my page?

Step 4 – Debug and Fix

Since HTML is a markup language that is interpreted rather than compiled, the simplest way to debug it is to open the files and see whether it looks like the way you intended it to be.

1. Open Windows Explorer and navigate to the folder where you have saved your HTML files are stored: `c:\appdevschool\data\web`
2. Double-click on the files to open them. It will be opened in the default browser (i.e. Internet Explorer).

Note: Project standards will dictate which browsers will be used to check your HTML. For this training, we will limit our testing to IE.

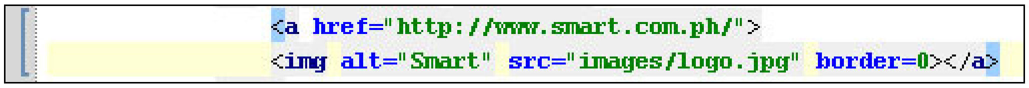
3. Compare your coded page to the Page Layout and Page Description sections of this manual.
4. Edit your code in IntelliJ and save it.
5. Refresh the browser to reflect the changes you have made.

However, IntelliJ has helpful features that will help you easily detect syntax errors in your HTML files while you are still typing them. This is why some programmers prefer to work with an IDE rather than code their HTML in notepad. Some features you may find useful are:

- **Color-coded text** – enables you to differentiate the different page elements such as:
    - dark blue = tags

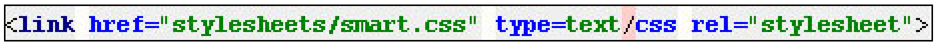
  - light blue = attributes
  - green = attribute values
  - gray = comments, and
  - black = text that will be displayed as is

- **Gutter symbols** – after you finish typing your ending tag or when your cursor is placed next to an ending tag, a blue bracket will appear on the gutter to indicate which start tag it is closing.



```
<a href="http://www.smart.com.ph/">
<img alt="Smart" src="images/logo.jpg" border=0></a>
```

- **Highlights** – special characters not coded as character entities or enclosed in quotes are highlighted.



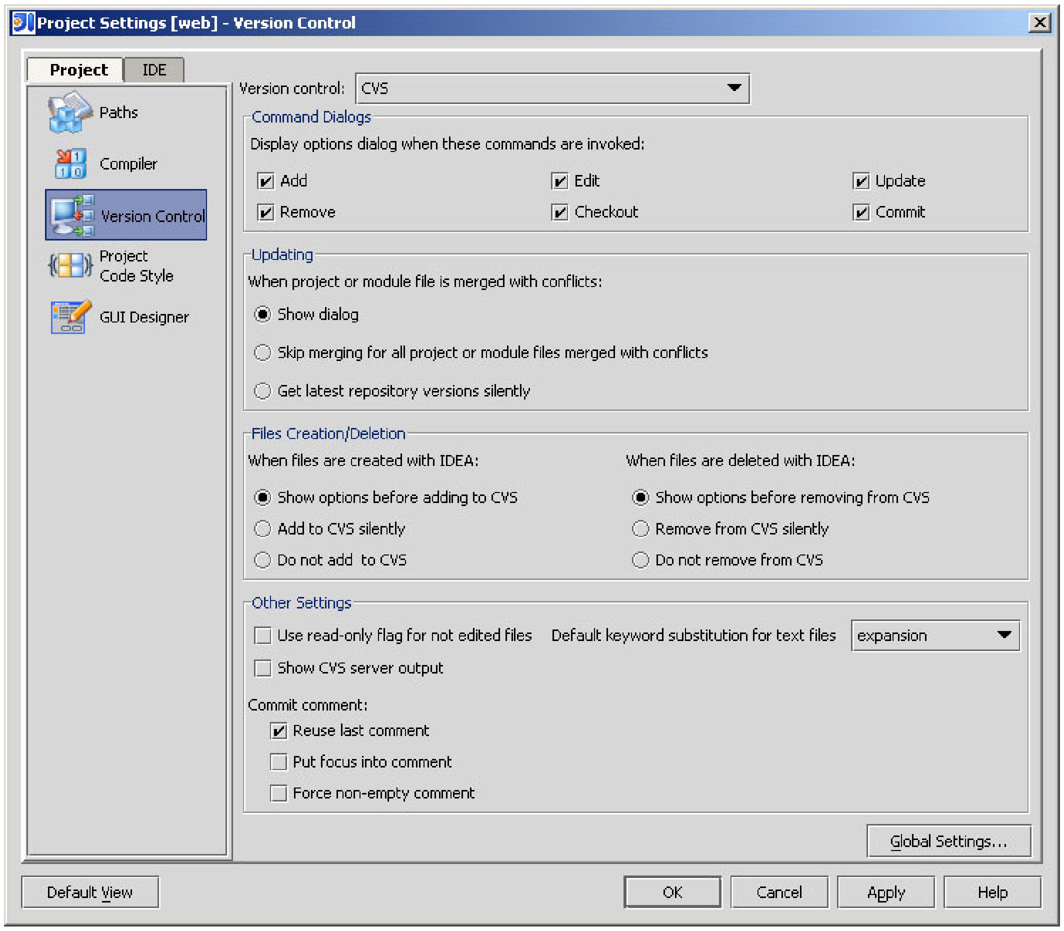
```
<link href="stylesheets/smart.css" type=text/css rel="stylesheet">
```

Step 5 – Save Project Files

- How do I check in files?
- How do I add files into CVS?

After you have completed the coding and debugging of your HTML files locally, you must upload the final documents to CVS.

1. Configure your project to connect to CVS.
   a. Click on **File – Settings**. The **Project Settings** dialog box appears.
   b. Click on **Version Control** from the **Project** tab.
   c. Choose CVS from the **Version control** drop-down list box.


Project Settings [web] - Version Control

Version control: CVS

Command Dialogs
Display options dialog when these commands are invoked:
[✓] Add [✓] Edit [✓] Update
[✓] Remove [✓] Checkout [✓] Commit

Updating
When project or module file is merged with conflicts:
[✓ Show dialog] Skip merging for all project or module files merged with conflicts / Get latest repository versions silently

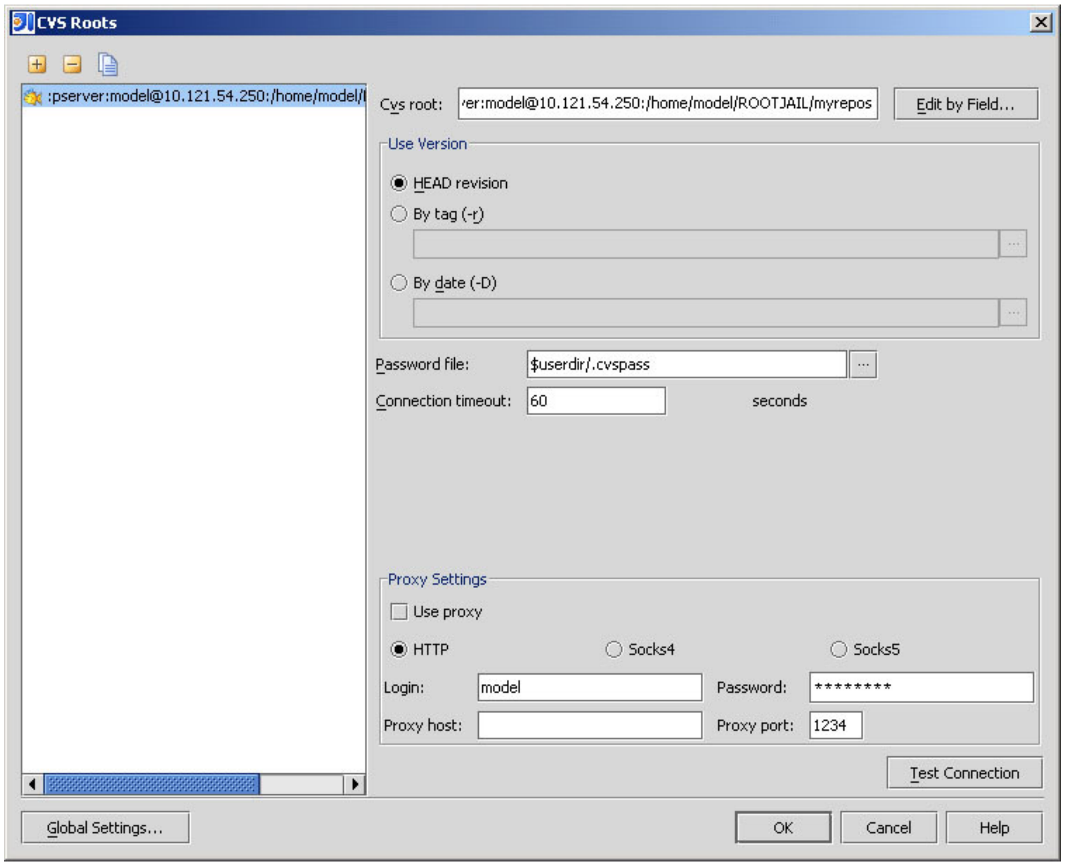
Files Creation/Deletion
When files are created with IDEA: [✓ Show options before adding to CVS] Add to CVS silently / Do not add to CVS
When files are deleted with IDEA: [✓ Show options before removing from CVS] Remove from CVS silently / Do not remove from CVS

Other Settings
[ ] Use read-only flag for not edited files
Default keyword substitution for text files: expansion
[ ] Show CVS server output
Commit comment:
[✓] Reuse last comment
[ ] Put focus into comment
[ ] Force non-empty comment

d. Leave the default settings as is and click **OK.**

e. Click on **File - Browse CVS Repository**. The **Select CVS Root Configuration** dialog box appears.

f. Click **Configure.** The **CVS Roots** dialog box appears.



:pserver:model@10.121.54.250:/home/model/ROOTJAIL/myrepos

Cvs root: ver:model@10.121.54.250:/home/model/ROOTJAIL/myrepos

● HEAD revision
○ By tag (-r)
○ By date (-D)

Password file: $userdir/.cvspass

Connection timeout: 60 seconds

☐ Use proxy

● HTTP ○ Socks4 ○ Socks5

Login: model Password: ********

Proxy host: Proxy port: 1234

g. In the **Cvs root** text box, type `:pserver:<username @10.121.54.250:/home/model/ROOTJAIL/myrepos`.

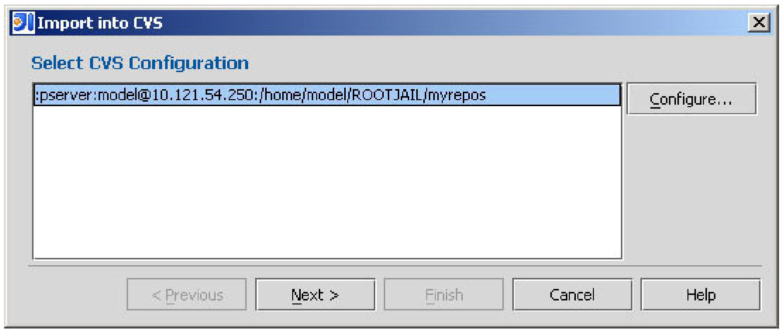
h. In the **Login** and **Password** text boxes, type the username and password given to you by your instructor respectively.

i. Click **OK** to return to **Select CVS Root Configuration** dialog box.

j. Click **OK** to connect to the root.

2. Import the project files to the CVS repository. You will only do this once.

   a. Click on **File – Import into CVS**. The **Import into CVS** dialog box appears.

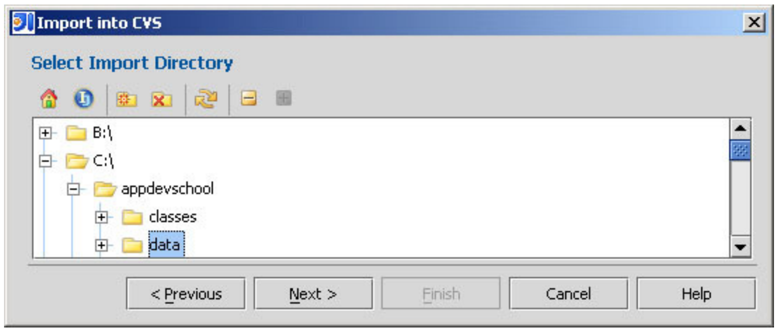


:pserver:model@10.121.54.250:/home/model/ROOTJAIL/myrepos

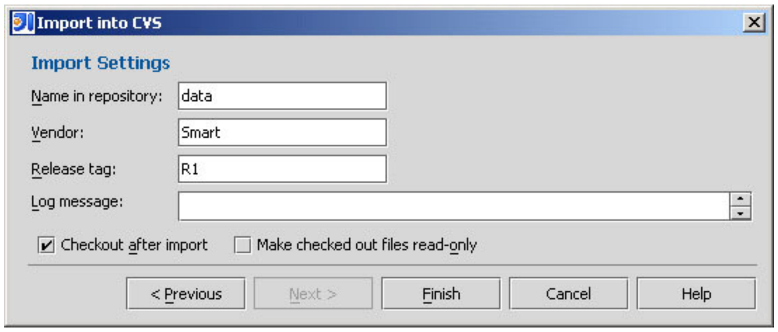
b. The CVS root you configured is selected by default. Click **<u>N</u>ext >**.

c. Choose your username in the list and click **<u>N</u>ext >**.

d. In the **Select Import Directory** list box, navigate to `c:\appdevschool\data` and click **<u>N</u>ext >**.



e. You are then asked to **Customize Keyword Substitutions**. Leave this as is and click **<u>N</u>ext >**.



Name in repository: data
Vendor: Smart
Release tag: R1
Log message:
☑ Checkout after import ☐ Make checked out files read-only

f. Type your name in the **<u>V</u>endor** text box.

g. Make sure that the **Checkout <u>a</u>fter import** checkbox is checked.

h. Click **<u>F</u>inish**.

3. After you import your files into CVS, concurrent access and modifications to it will now be handled by the CVS application. For any changes that you make, update your saved files in the repository.
    a. Click on **CVS – Commit Project**.

Step 6 – Reflect on Key Learning Points

Inform your instructor that you have completed the coding and testing activities of this segment. He/she will review your work.

Review the key learning points for this segment. Answer the following questions and discuss them with your co-participants and faculty.

General

1. What did you learn from today's experience that will help you through the rest of the course?
2. What would you do differently if you had to repeat today's activity?
3. What did you learn from today's assignment that will help you orient yourself in your next project?
4. How did your co-participants assist you in your learning? What are some of the ways that you assisted them?
5. What kind of activity instructions or guidance can you expect on the job?

Specific

1. Name three things you learned today.
2. Identify the method(s) you used to learn the coding syntax for HTML.
3. What strategy did you find most useful for identifying defects in your code?

Technical Specifications

Purpose

This document details the processing logic for the HTML pages and their respective controls. Sufficient information is provided to write the code for the pages. This technical specifications document is a hybrid of sections derived from the full-length Functional Specifications and DB Design Documents. In the future (depending on the scope of your project), you may encounter full versions of these documents.

- What are functional specifications? Why do we need it?

Project Description

Users of the Smart Online facility for subscribers have been clamoring for an additional feature that would enable them to look up their usage history to track their calls and manage their subscription plan.

Management decided to pilot test this feature by adding the usage inquiry module to the current suite of applications available to our customer service representatives.

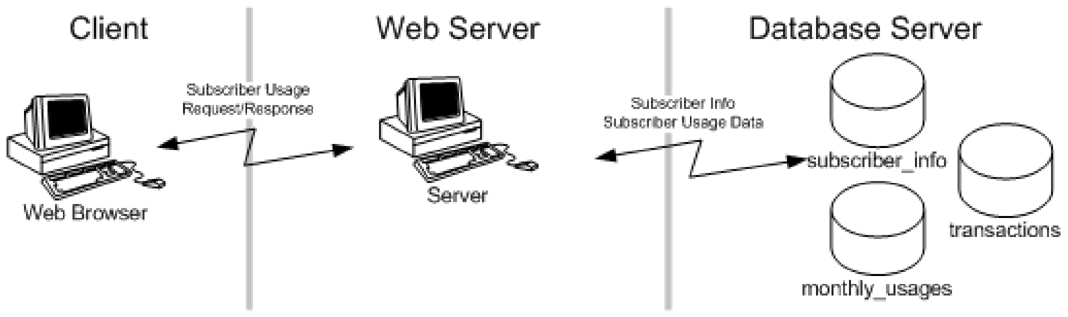
It will be used in the following customer service interaction:

- The customer calls up the customer service hotline and gives his/her account number to the CSR.
- The CSR inputs the account number into the system. If entry is successful, will be shown the account information of the subscriber for verifying the caller's identity.
- Once the caller's identity has been verified manually, the CSR can look up the subscriber's current month's usage or the year-to-date usage history.

To aid us in defining the requirements, we are going to create mock up screens of the system. This will allow the users to see the system before we actually do any codes.

Architectural Overview

The usage inquiry module that you will be developing is a web application. The diagram below illustrates the system components involved in the proposed system architecture.



Database Tables

Though the report pages display information from the database, database access is abstracted from the report pages by several classes called managers. The managers retrieve information from three tables in the database. The Table Definitions for these tables are included below for informational purposes only.

SUBSCRIBER_INFO

SI_IMSI	SI_NAME	SI_MSISDN	SI_IMSI	SI_ACC_NUM	SI_BIRTH DAY	SI_MAIDEN _NAME
Primary Key	varchar 30	char 20	char 20	char 20	varchar2 14	varchar 20

TRANSACTIONS

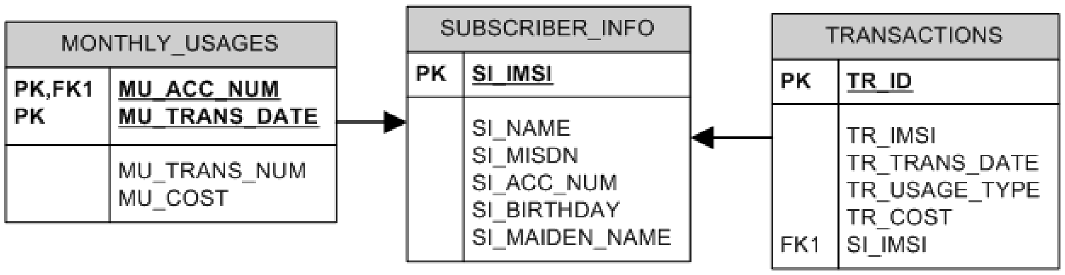
TR_ID	TR_IMSI	TR_TS	TR_UT	TR_ID	TR_COST
Primary Key	char 20	char 14	char 5	char 20	decimal 10,2

MONTHLY_USAGES

MU_ACC_NUM	MU_TS	MU_ACC_ NUM	MU_TS	MU_TRANS_ NUM	MU_COST
Primary Key		char 20	char 6	int	decimal 10,2

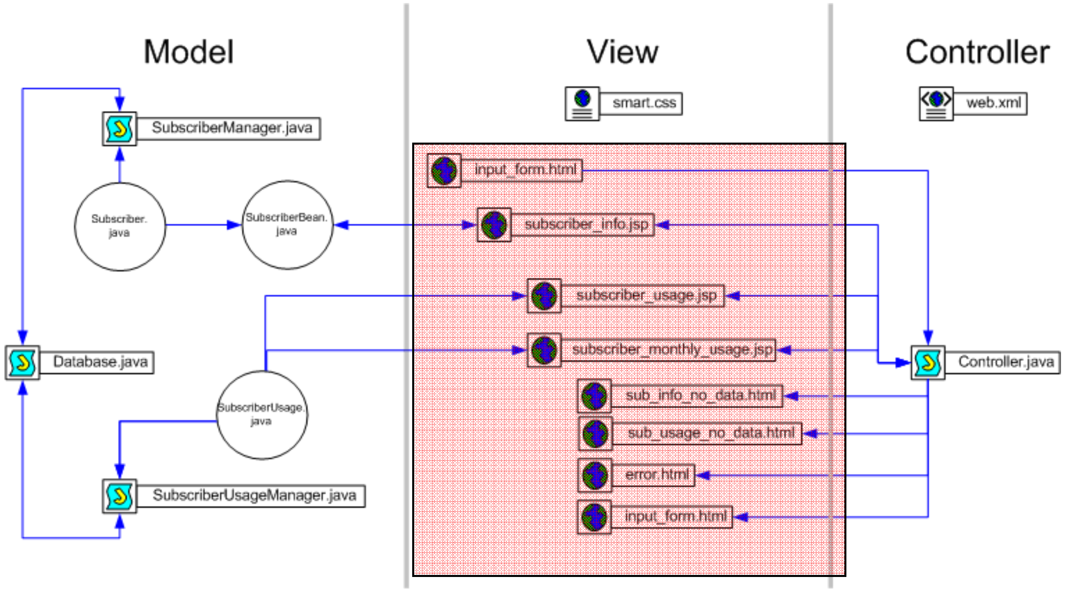
Database Architecture

The tables used in the project are related to each other thus:



Calling Sequence

All items except the presentation layer (items surrounded by box overlay) of the project have been pre-coded. The diagram below illustrates how page request and response are fulfilled within the web server.



Systems Integration

The HTML pages use the following external files, functions and/or procedures to accomplish the stated project requirements.

External File, Function or Procedure	Description or Purpose	Type
smart.css	Defines the formatting styles of the elements of the company website. also creates classes that can be applied to elements	Cascading Style Sheet
verify.js	Validates data input for correct data type and format	JavaScript function.

Call Pattern References

Among the HTML files you will create today, the report pages are the only ones who will display information from the database.

To enable you to incorporate data returned by the SQL calls, you will be asked to convert these HTML pages into JSP pages in the following segment.

Page Description

input_form.html

The input form page contains a single text box where the user can enter a *subscriber account number*. When the user clicks on a *submit* button, it should post the *subscriber account number* to the subscriber information page or return an error if the account number does not exist in the database.

subscriber_info.html

This page should display the following relevant subscriber information: *subscriber name*, *MSISDN*, *IMSI*, *account number*, *birthday* and *mother's maiden name*.

From this page the user can click on:

- *change account* button to return to the input form page and enter another subscriber account.
- *get subscriber usage* button to show usage history year-to-date

- *get subscriber monthly usage* button to show usage history for the current month in a vertical table.

The page should follow a top-to-bottom-left-to right tab sequence. Images should not be part of the tabbing sequence.

The Title bar should display "Subscriber Information"

subscriber_usage.html

This page should show usage history year-to-date in a vertical table with the following columns: *IMSI, transaction date, usage type, transaction ID* and *cost.*

From this page the user can click on:

- *change account* button to return to the input form page and enter another subscriber account.
- *get subscriber info* button to display the subscriber information
- *get subscriber monthly usage* button to show usage history for the current month in a vertical table.

The page should follow a top-to-bottom-left-to right tab sequence. Images should not be part of the tabbing sequence.

The Title bar should display "Subscriber Usage"

subscriber_monthly_usage.html

This page shows usage history for the current month in a vertical table with the following columns: *account number, transaction date, number of transactions for the day,* and *cost.*

From this page the user can click on:

- *change account* button to return to the input form page and enter another subscriber account.
- *get subscriber info* button to display the subscriber information
- *get subscriber usage* button to show usage history year-to-date

The page should follow a top-to-bottom-left-to right tab sequence. Images should not be part of the tabbing sequence.

The Title bar should display "Subscriber Monthly Usage"

sub_info_no_data.html

This page shows the following error message if subscriber information cannot be retrieved from the database.

"The subscriber account number you entered does not exist or there was an error in fulfilling your request."

The Title bar should display "Subscriber Unknown"

There should be a link to the input form page so that they can re-enter the subscriber account number.

sub_usage_no_data.html

This page shows the following error message if no usage records exists for a valid account number:

"The subscriber account has no usage records or there was an error in fulfilling your request."

The Title bar should display "Subscriber Usage Unknown"

There should be a link to the input form page so that they can re-enter the subscriber account number.

error.html

This page displays the following catch all error message for any other error encountered by the controller:

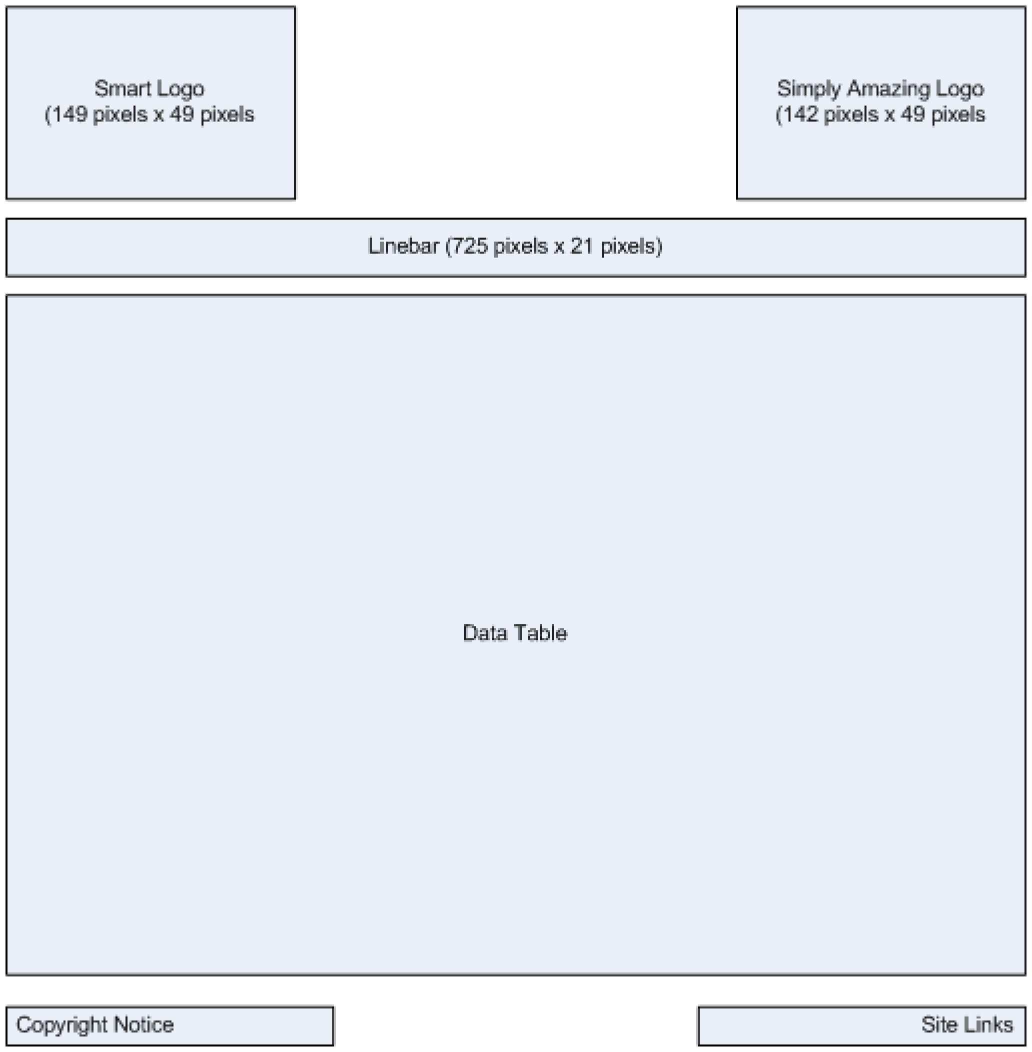
"The resource you are looking for does not exist or there was an error in fulfilling your request."

The Title bar should display "Error"

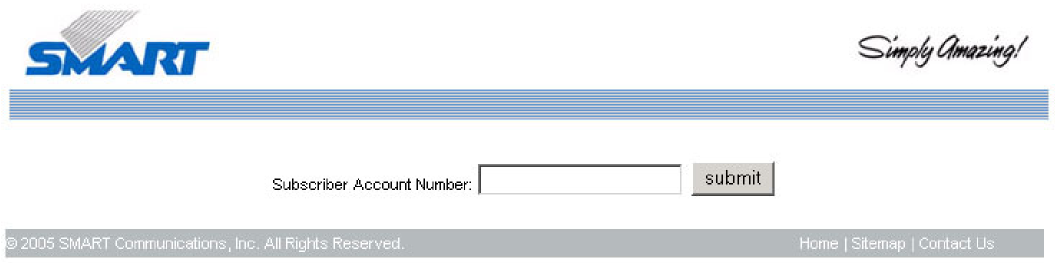
There should be a link to the input form page so that they can re-enter the subscriber account number.

Page Layout

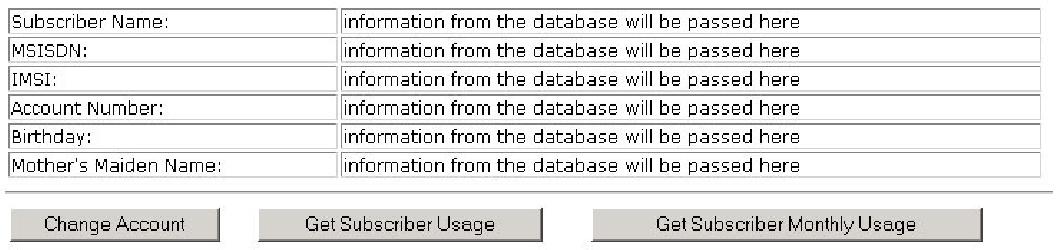
The HTML pages conform to the general layout below:



Data Entry Page: input_form.html

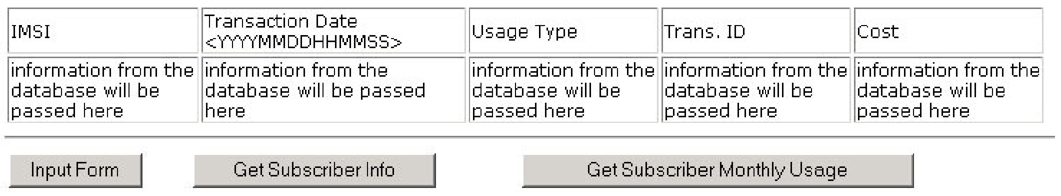

Simply Amazing!
Subscriber Account Number:
submit
© 2005 SMART Communications, Inc. All Rights Reserved.
Home | Sitemap | Contact Us

Report Page: subscriber_info.html



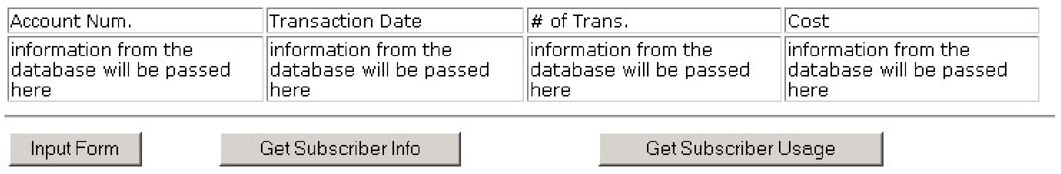
Subscriber Name:	information from the database will be passed here
MSISDN:	information from the database will be passed here
IMSI:	information from the database will be passed here
Account Number:	information from the database will be passed here
Birthday:	information from the database will be passed here
Mother's Maiden Name:	information from the database will be passed here

Report Page: subscriber_usage.html



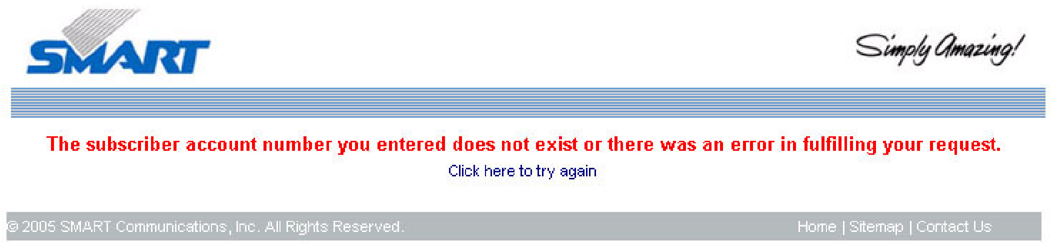
IMSI	Transaction Date <YYYYMMDDHHMMSS>	Usage Type	Trans. ID	Cost
information from the database will be passed here	information from the database will be passed here	information from the database will be passed here	information from the database will be passed here	information from the database will be passed here

Report Page: subscriber_monthly_usage.html



Account Num.	Transaction Date	# of Trans.	Cost
information from the database will be passed here	information from the database will be passed here	information from the database will be passed here	information from the database will be passed here

Error Page: sub_info_no_data.html

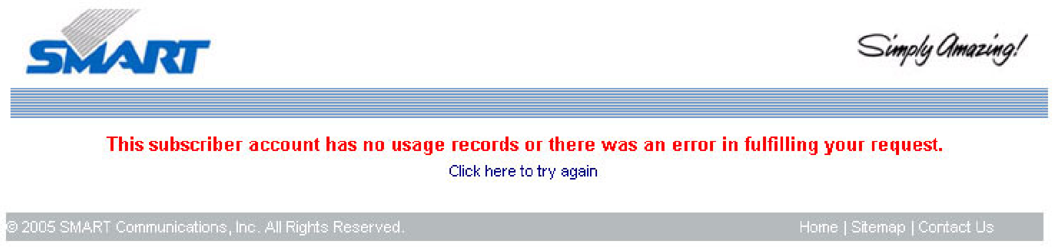


The subscriber account number you entered does not exist or there was an error in fulfilling your request.

Click here to try again

© 2005 SMART Communications, Inc. All Rights Reserved. Home | Sitemap | Contact Us

Error Page: sub_usage_no_data.html

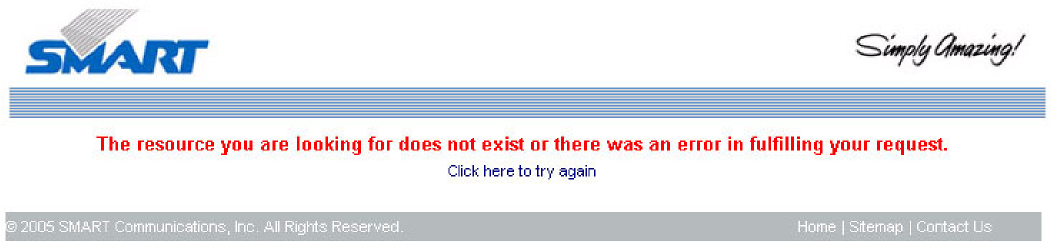


This subscriber account has no usage records or there was an error in fulfilling your request.

Click here to try again

© 2005 SMART Communications, Inc. All Rights Reserved. Home | Sitemap | Contact Us

Error Page: error.html



The resource you are looking for does not exist or there was an error in fulfilling your request.

Click here to try again

© 2005 SMART Communications, Inc. All Rights Reserved. Home | Sitemap | Contact Us

Application Security Requirements

Although you are not required to implement any security controls in this particular exercise, it is very important to note that you put input validation routines in your user interfaces when building actual application systems.

Input validation is the first line of defense of our application system as it ensures that the input data received by the system is valid and 'safe'. 'Safe' means that the data inputs are free from:

- Malformed characters, which could cause buffer overflows (e.g., excessive number of characters), thus, could make the application crash which may eventually result to “Denial of Service”.
- Malicious entries such as SQL Commands in unfiltered form fields. Common attacks nowadays includes SQL query strings entered through the form fields, by which data manipulation can be done as if directly accessing the database. This attack is also referred to as “SQL Injection”.
- Invalid input characters and/or format for a certain data field, thus, allowing the system to process unnecessary information.

The following are the best practices in designing data input validation as a security control:

1. System-wise, do not trust user inputs. ALWAYS VALIDATE user inputs as to integrity, validity and accuracy.
2. Input validation and filtering should be done **both** at the client side **and** the server side, especially for critical fields such as *Username* & *Password*.
3. Filter invalid inputs prior to actual processing.
4. Verify maximum character prior to processing, to prevent buffer overflow attacks.
5. Do not use unrestricted input fields that accept any data formats.