Enterprise-Scale Software Development (COMP5348)

Semester 1, 2013

Tutorial Week 1 - Introducing the Video Store Application

1) In the lectures this week you were introduced to NFRs and some of the patterns used in software like MVC and layering. Let's take a look at some of these patterns in use in a sample enterprise application. We'll use the sample VideoStore application which you can download from eLearning for this exercise.

a. We'll first setup the database used by the Video Store application. To do this:

i. Open the VideoStore solution by double clicking on VideoStore.sln. This solution file should be under VideoStore/VideoStore.sln at the location where you unzip the download.

ii. Build the solution by right clicking on the top (solution) node in "Solution Explorer" and clicking on Build.

iii. Click on View >Other Windows>Server Explorer OR View>Server Explorer.

iv. On the "Server Explorer" tab, right click on Data Connections and click on create new sql server database.

v. In the dialog that pops up, select or type in your Sql server instance name (usually PCName\MSSQLSERVER2008 – but ask your tutor if this isn't working). Then choose "Use Windows Authentication", and type in "Videos" for the database name.

vi. This should create the Videos database and show you a new node under Data Connections in the Server Explorer tab. Right click on this new database node and click on properties. This should bring up a property grid. Copy the value specified for the "Connection String" property.

vii. Open the VideoStore.Application/VideoStore.Process/App.configfile and paste the value you just copied to replace the string "Data Source=MATT-PC\SQLEXPRESS;Initial Catalog=Videos;Integrated Security=True;MultipleActiveResultSets=True".

viii. Open the file under VideoStore.Business/VideoStore.Business.Entities/VideoStore.Entities Mode.edmx.sql. Right click this file and click on Execute sql. This will bring up a dialog to connect to sql server. Type in your sql server instance name (from 1 v) and use Windows Authentication. Once you

click ok, your Videos database will be generated. You can view the tables generated by going to the Server Explorer tab, expanding the Videos database you created earlier, and expanding the tables node.

If you take a look at the sql file you just executed, you'll notice that the tables and columns that are created by the script correspond to the entities, properties, and associations you'll see if you open the VideoStore.Business/VideoStore.Business.Entities/VideoStoreEntityModel.edmx file.

For new projects, one would usually create a conceptual definition as seen in VideoStore.Business/VideoStore.Business.Entities/VideoStoreEntityModel.edmx, and then generate the corresponding backing store script and c# classes, (see VideoStore.Business/VideoStore.Business.Entities/VideoStoreEntityModel.tt for c# classes), based on this definition. For existing databases, you can also generate the conceptual model based on that database definition.

b. The video store application lets registered customers log in, add items to a shopping cart, and submit a purchase order. The MVC pattern you looked at in lectures is used to implement the UI. Let's take a look at the controllers, views, and models used when a customer adds an item to a cart; and when a customer submits a purchase order:

    i. In Visual Studio, navigate to VideoStore.Presentation/VideoStore.WebClient/Controllers/CartController. In this class there is a method called AddToCart. When a customer clicks on the button "Add to cart" on the Browse Media page, a url like http://localhost:50575/Cart/AddToCart is requested, and as a result this AddToCart method is invoked. The AddToCart method uses the cart object that is passed in as a parameter so that it can add the requested item to the shopping cart. It does this by calling the AddItem method on the cart object. The AddToCart controller method then redirects the user to an Index page that lists all the items currently in the cart.

    ii. Let's take a look at the Cart class for the cart object which you saw passed into the AddToCart method in i. On Solution Explorer, navigate to VideoStore.Presentation/VideoStore.WebClient/Models/Cart. The Cart class serves to model information about the items that a customer would like to purchase when they check out. By looking at the instance variables of this class you can see that a list is used to encapsulate information about items that are about to be ordered. You can also see methods like AddItem which encapsulate the behaviour of the cart. For example, the AddItem method doesn't just add an item to a list. It first checks to see if the item has already been ordered, and

then adds an order item or increases the quantity associated with a pre-existing order item. You encapsulate this kind of logic in the model so that you don't need to duplicate it everywhere where you need to add an item to the cart.

iii. You've seen what the Model for the shopping cart looks like. Let's now take a look at what the view that's used to display the cart model looks like. Navigate to ViewStore.Presentation/VideoStore.WebClient/Views/Index.aspx. The first thing you notice on the top line of this view is that it's bound to the cart model via the Inherits="System.Web.Mvc.ViewPage<VideoStore.WebClient.Models.Cart>" declaration – this is what's known as a strongly typed view (it can only deal with models of this type). The view is created with an instance of the cart model at CartController::Index(), and it uses the model's property values to display items in the shopping cart. For example, you'll notice that the view defines a html table, and creates a new row (<tr>) in the table for each item that exists in the cart object. This mixing of code with html in the view allows the web server to return a dynamic view based on the parameters passed in with the user request, as well as based on what exists in a model.

    Because MVC separates out business logic and business concerns from presentation, you can also write more than one view that uses the same Model, without needing to duplicate business related concerns across these multiple views. Can you find the other view in the VideoStore.Presentation/VideoStore.WebClient uses the Cart model?

iv. When a user checks out, it means that the order items that have been added to the cart thus far are ready to be submitted as part of a single purchase order. Let's take a look at what happens at checkout:

    In Solution Explorer, navigate to VideoStore.Presentation/VideoStore.WebClient/CartController. In this class there is a method called CheckOut. Again, the same type of mapping between url and controller exists, and this controller method is invoked when a url like http://localhost:50575/Cart/CheckOut is requested. When this happens, the cart object is used to submit an order. Navigate to the SubmitOrderAndClearCart method by right clicking on it and clicking Go to definition. This method encapsulates the logic required to create a new order containing all of the order items added thus far to the cart; submitting this order to the application server; and clearing the contents of the Cart.

2) In the first exercise you looked at the controllers, views and models associated with submitting an order. We'll now look at how the state of the cart model changes as customers add items to the cart, and how the state of the database changes as customers are created and orders are submitted.

a. You'll first need to start the application server and the web server. To do this, right click on the VideoStore.Application/VideoStore.Process and click Debug>Start New Instance.

b. Once you start the application server, there should be a speech bubble that pops up on the right hand corner telling you that the ASP development server has started, and giving you the port number to use if you want to access the video-store web server. If the port is 50575, you would type the address http://localhost:50575 in your browser to access the video store website on your PC. Go ahead and access the video store web site.

c. If you wish to debug through web-server code, once you have accessed the main page, right click on the VideoStore.WebClient project and select Debug>Start New Instance.

d. When you start the web client, you should be presented with a login page. Register for a new customer account on this login page by clicking on the Register hyperlink. Once you have registered with a customer account you should be taken to the home page.

e. Once you register for a customer account, the details you entered should now be stored in the database. Click on View>Server Explorer, expand the Videos database instance you created, and browse to the Customer table under the tables folder. When you right click the customer table and click on Show Table Data you should see a record that contains the details you entered in b). If you also look at the records in LoginCredential, you should see the user name and a hashed version of the password for the customer you created.

f. Once you've done this, add some items to the shopping cart by clicking on Browse Media, and clicking on one of the "+ Add to Cart" buttons.

g. After you add some items to the cart, the instance of the cart object you saw earlier in 1 ii), (which exists from when you login to when you log off), should contain some order items. To see the state of the cart object:

    i. Navigate to VideoStore.Presentation/VideoStore.WebClient/Controllers/StoreController. In this class, place a breakpoint on the single line in the ListMedia method.

    ii. Go back to the web-browser and click on Browse Media. This should hit the breakpoint you defined in i. When this happens, go to the watch window in Visual Studio, and add a watch for "this.Session["_cart"]".

Expanding this will show you the cart object stored as session state. The session variable, which you defined a watch for holds information about session state. It's usually stored on the web server, but you can configure your application to store session state other on another machine or on in a database (http://msdn.microsoft.com/en-us/library/ms178586.aspx).

3) We'll now take a look at measuring the performance of VideoStore. There are a number of tools that let you record a certain test case, (like that described in Listing 1 below), and play that test case back as if it were executed by N clients concurrently. You can specify how many clients will run the test case concurrently. In today's tutorial we'll look at the response time of VideoStore associated with request/response pairs that occur in Listing 1.

i. Logon
ii. Browse Media
iii. Add Item to Cart
iv. Check out
v. Log Off

Listing 1

More specifically, we'll look at the average response time associated with retrieving the list of media items. We'll use a tool called JMeter to do this (see Appendix for a list of some other load test tools available).

i) Download the tool from http://jmeter.apache.org/download_jmeter.cgi

ii) Unzip the file, and browse to the bin folder. Double click on jmeter.bat.

iii) Once Jmeter starts up:

a. Ensure you have server and client running (as described in 2a - c)

b. Right click on Test Plan on the top left and Add>Threads>Thread Group

c. Right click on Thread Group and click on Add>Config Element>Http Request Defaults

d. On the detail view that shows up for this node, in the server name box type localhost. In the Port Number box type in the port which the VS web server mentions it's running on (usually 50575). For the protocol textbox type in http.

e. Right click WorkBench and Add>Non Test Element>HTTP Proxy Server. In the detail view that shows up for this node, change the value in the port box to 9090. In the Target Controller dropdown choose TestPlan>Thread Group. Click the "Add" button under patterns to include, and on the row

that appears type ".*"

f. Click on start at the bottom of this detail view.

g. Start Internet Explorer. Go to Internet Options, then go to the connections tab. Click on LAN settings, and then check the box "Use a proxy server for your LAN".

h. Uncheck "Automatically Detect Settings"

i. For the address type in localhost, and use the port 9090.

j. Click ok, and apply changes.

k. Now, when you type in a url in internet explorer, or chrome, all interaction is recorded in Jmeter.

l. In Chrome, navigate to the url of videostore (http://localhost:portNumber). (portNumber is usually 50575)

m. Log in.

n. Browse Media

o. Add an item to the cart

p. Check out

q. Log out

r. Close the browser.

s. Right click on Thread Group in JMeter, and Add>Config Element HTTP cookie manager. For the cookie policy drop down choose compatibility.

t. Right click on Thread Group again, and Add>Listener>Aggregate Report. This report will show you the outcome of load tests.

u. Click on Thread Group and set threads and loop count to 1.

v. Click on the Aggregate Report you created, then click on Run and start.

w. Once your test completes you should see a table of results like the one shown below:

Label	# Samples	Average	Median	90% Line	Min	Max
/Account/LogOn	4	142	140	146	139	146
/Content/Site.css	6	4	4	5	4	5
/favicon.ico	6	59	16	146	15	147
/Account/LogOn?Ret...	1	437	437	437	437	437
/	2	122	100	145	100	145
/Store/ListMedia	1	125	125	125	125	125
/Cart/AddToCart	1	99	99	99	99	99
/Cart	1	73	73	73	73	73
/Cart/CheckOut	1	120	120	120	120	120
/Account/LogOff	1	156	156	156	156	156
TOTAL	24	92	99	147	4	437

To list media items, it took an average of 125ms from when the request was issued to when the response was received.

x. Delete the Aggregate Report from the tree, and create another aggregate report. Under thread group change thread count to 10. Click on Run>Start. You should notice that your response times are now slower.
y. Try the test with a larger thread count. Response should now perceivably slower.

Once you've finished experimenting with JMeter, ensure you restore all the settings for Internet Explorer (Check "Automatically detect settings" and uncheck "Use a proxy server for your LAN" on the LAN Settings page of Internet Options in Internet Explorer).

You now have an idea of how load testing tools for web applications operate. Testers usually try to test their applications to the limits (when throughput saturates (tapers off)). When they find the system limits they identify bottlenecks that enforce these limits, and try to see if changes in system configuration would improve performance. You'll learn about scaling up and scaling out in later lectures. Incidentally, you'd never host a web application using the VS integrated web server. You might use something like IIS, which dramatically improves the response times you saw today. For example, instead of a 4 second response time for List Media with 100 concurrent threads, when running IIS the author observed an average response time of only ~0.5s.

Homework

1. Identify the layers used in the Video Store application. Why is layering important? Why shouldn't a lower level layer point to an upper level layer? Are there any disadvantages in the use of layers?

    a. Take a look at the service layer guidelines from Microsoft at: http://msdn.microsoft.com/en-us/library/ee658090.aspx . Now take a look at the VideoStore.Services and VideoStore.Services.Interfaces projects. Notice that there are no dedicated message types or DTOs, and that business entities are exposed through the service interfaces. What implications does this have?

Appendix

Some Notes on the Data Layer

Microsoft Entity Framework allows you to use any backing store that you can find a provider for. A list of existing providers can be found at http://msdn.microsoft.com/en-us/data/dd363565. NET includes providers for SQL Server, and SQL Azure out of the box.

The plug-in/provider model prevents you from being locked in to using a specific data store. If you want your application to also be agnostic to the Object Relational Mapping Framework you are using, (for example if you wish to switch to NHibernate from MS Entity Framework later on), you may also wish to make use of the repository pattern described at:

http://martinfowler.com/eaaCatalog/repository.html

http://msdn.microsoft.com/en-us/library/ff649690.aspx

http://blogs.microsoft.co.il/blogs/gilf/archive/2010/01/20/using-repository-pattern-with-entity-framework.aspx

Other Web Stress Tools

- Microsoft Web Application Stress Tool
- Grinder
- Mercury LoadRunner