C# web pages

and C# database programming

Rich Hildred for

PROG1760 #1 (12F)

Advanced Programming w Visual Studio C#.NET

Study Notes

# Introduction

C# web programming can be performed in 3 basic ways:

1. Web Pages (with Razor syntax)
2. MVC (Model View Controller)
3. Web Forms (traditional ASP.NET)

These notes are for Web Pages (with Razor syntax). Instead of a .html file these web pages are contained in a .cshtml file that is a mixture of regular html and C# code that is turned into html on a web server running IIS and the .NET framework or Apache and Mono. This method of web programming is basic html5 css3 and javascript with one extra ingredient. The extra ingredient is the ability to have C# code mixed in with html that is run at the server to produce an html5 response that browsers can render.

In order to edit and test .cshtml pages in Visual Studio either Visual Studio 2012 must be used, or the Microsoft Web Platform Installer can be used to add MVC3 to Visual Studio 2010. Unfortunately Visual Studio 2008 can not be used as it works with the .NET 2.x framework, and .cshtml is dependent on the .NET 4.x framework. One can get the Microsoft Web Platform installer at: <http://www.microsoft.com/web/downloads/platform.aspx>.

A .cshtml page is different from a .html page in one way. A .cshtml page has C# code that is preceded by an “@” symbol. The code can be a single expression like “@DateTime.Now” in Figure 1



Figure

Or the C# code can be placed within a block as in Figure 2:

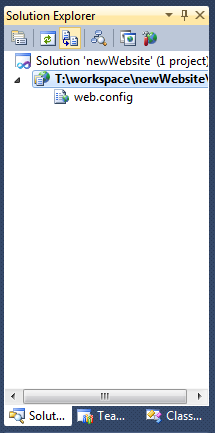


Figure

There is a longer example of C# code in a block in Figure 8.

## To Start a .cshtml page

1. If you already have a web site open it if need be by going to File/Open/Website and selecting the root directory of the site and proceed to step 6.
2. In Visual Studio go to File/New/Website.
3. In the New Web Site dialog, select Visual C# and then select ASP.net empty web site Visual C#.
4. Name your web site in the web location panel at the bottom of the dialog
5. Press ok and in your solution explorer a project should appear with a web.config file in it



Figure

1. On the project (second line in figure 3) click the right mouse button and in the context menu select Add New Item
2. In the Add New Item dialog select Web Page(Razor)
3. In the name panel at the bottom of the Add New Item dialog name your file something other than WebPage.cshtml

The ability to place C# code in a web page can be useful in itself for tasks like managing a catalogue, or some sort of data visualization task, but to write a web application it is also necessary to be able to act on data input from the internet.

## Input From the Internet

To understand how to get input from the Internet we need to understand a little about the http protocol. The http protocol is made up of requests (blue lines) from web browsers and responses from web servers (purple lines). The requests and responses are streams of text data that flow over the same TCP/IP connection. The requests and responses consist of specific headers that describe content in a body that follows the headers and 1 blank line.

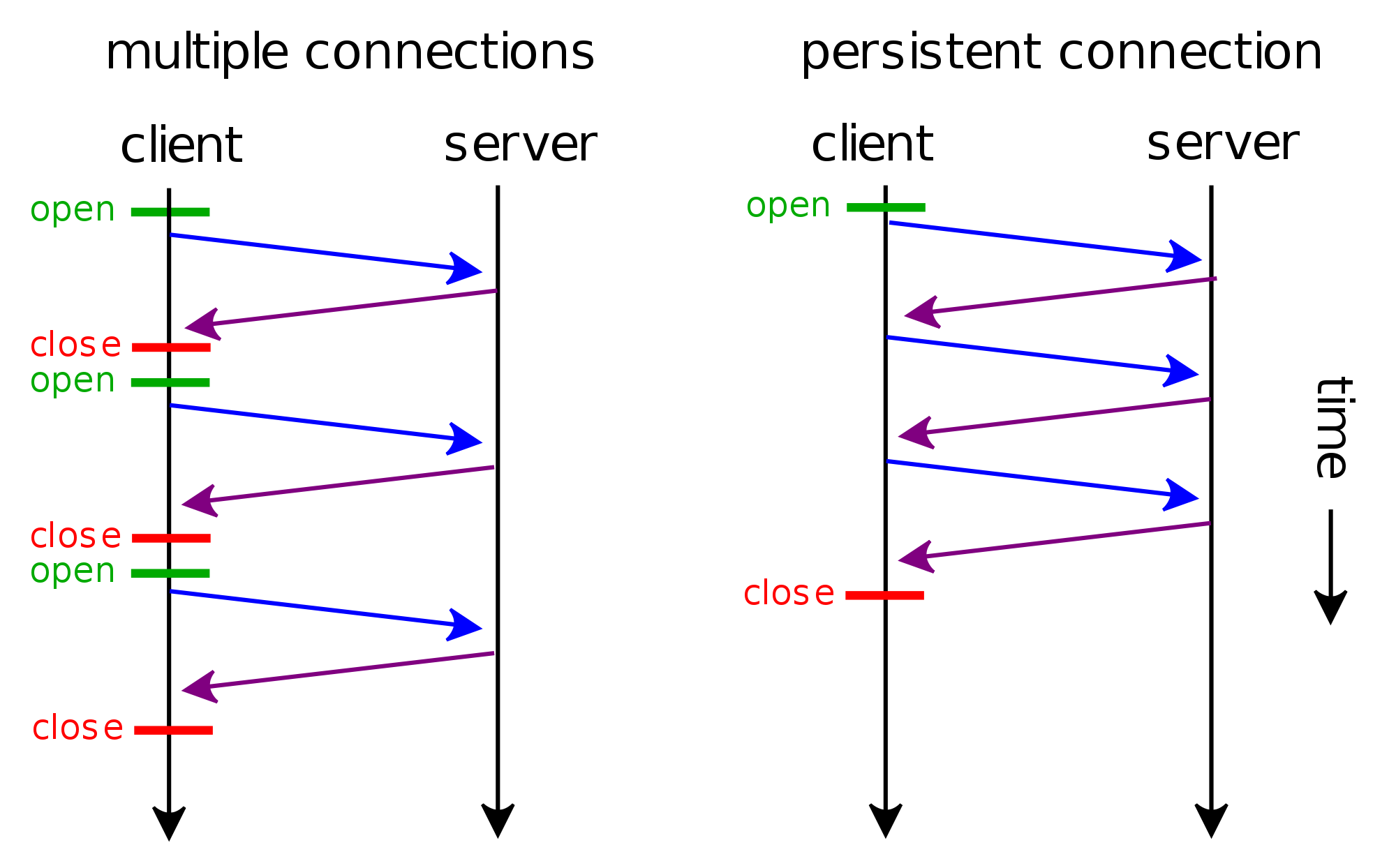


Figure [[1]](#endnote-1)

This is an http GET request (blue line in Figure 3)



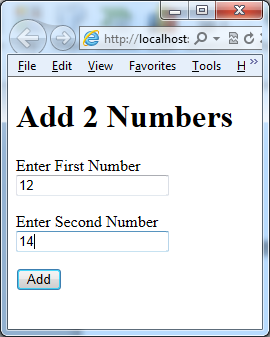
Figure

This is the http response to the same request (purple line in Figure 3):



Figure

The World Wide Web is built on these requests and responses. When the browser has rendered the form described in the html that is the body of the response in Figure 5 it will look something like the image in Figure 6.



Figure

When the user clicks on the add button in their browser, a new post request is generated. Notice the numbers 12 and 14 in the browser and in the request. In the request 12 and 14 are the values of name-value pairs that are named by the input name attributes in Figure 5.



Figure

It is up to our .cshtml page to produce the response body for both the GET and the POST requests. In our C# code the textInput1 and textInput2 form values are contained in Request[“textInput1”] and Request[“textInput2”] respectively. In the case of the first get request there is no form input, so we must test those fields to see if they are null. In the 2nd post request the data is filled in so we can display the sum of the results like:

<p>

Sum of @nInput1 + @nInput2 = @nResult</p>

The complete page is listed in Figure 9.



Figure

With this approach we can capture all sorts of inputs from the internet. The other interesting thing from this approach is the way in which we handle bad data using structured exception handling. We parse the nInput1 and nInput2 Decimals inside a try block. That way if the user enters “asdf” in the textInput1 input, we can catch the format exception and display a message that is informative to our intended audience. At first our intended audience is us, so we just display e.Message, which describes the error to a programmer. The exciting thing about the structured exception handling is that it lets us handle the error at the correct level of abstraction. This leads us to another exciting point. In our .cshtml files we can take advantage of the whole C# .NET ecosystem, including adding our own classes.

## Adding and Instantiating Classes in the .cshtml Web Pages

When we used the Add Item dialog there were many choices such as “Style Sheet”, “XML File” and “HTML Page”. One of the choices is to add a Class. To Add a class:

1. Right mouse click over the project and from the context menu select Add New Item
2. From the Add New Item dialog select Class
3. If this is the first class that you are adding to the web site you will get a dialog asking you if you want to place the file in the App\_Code folder. Click Yes, and the empty class will be created for you to fill in.

Being able to add classes to work behind web pages is exciting for a number of reasons. Like css separates presentation from the content of an html file, classes take that one step further by being able to have multiple .cshtml views of the same business/domain model classes. The class in Figure 10 is pretty basic, but illustrates the point.



Figure

When the Demo class is called from the following .cshtml file in Figure 11:



Figure

The following output in Figure 12 is generated:



Figure

Placing business logic in classes allows us to instantiate objects that have domain behaviour like oMyaccount.credit or oMyAccount.debit as well as persist and retrieve themselves from the database per Michael Fowler’s Active Record pattern, “An object that wraps a row in a database table or view, encapsulates the database access, and adds domain logic on that data.”[[2]](#endnote-2) We can also instantiate C# ADO.NET objects to manipulate SQL data in a more traditional way.

## Accessing MySql from .cshtml Web Pages

ADO.NET is the basic way to access a database from C# as described in Murach. The basic pattern is to:

1. Add a reference to the provider that you are using … for mysql we browse to mysql.data.dll
2. Instantiate a connection object depending on the provider that you are using
3. Instantiate a command object from the sql String that you want to run and the connection object from step 1
4. Execute the command into a data reader if the command is a select, or to return the number of rows affected if the command is not a select

The following table names the specific classes for some of the ADO.NET providers that are used in this process.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SQL Server | OLE DB | ODBC | Oracle | MySQL |
| SQLConnection | OleDbConnection | OdbcConnection | OracleConnection | MySqlConnection |
| SQLCommand | OleDbCommand | OdbcCommand | OracleCommand | MySqlCommand |
| SqlDataReader | OleDbDataReader | OdbcDataReader | OracleDataReader | MySqlDataReader |

The following example in Figure 13 illustrates this pattern, by first connecting with the connection string "user=root;database=sample;convert zero datetime=true". In this case we have added “convert zero datetime=true to our connection string so that we can display rows with null MySql DATETIME values as C# DateTime objects. Next we create a command from the String "SELECT type, class, numberofcrew, commisioned, name FROM ships" and the connection object that we just created. Finally we execute the query into a MySqlDataReader and loop through the results with a results.Read().



Figure

## Parameters and SQL Injection Attacks from Web Input

If we want to be able to search our ships database the naïve approach would be to use the input technique from Figure 9. We could do something like in Figure 14:



Figure

Unfortunately this is a really bad idea, because if someone searches for a ship named "’);DROP TABLE test;#" and there is actually a test table in the database they can cause the table to be dropped. This is called a SQL injection attack, and is certain to happen, given enough time. The proper way to prevent this kind of attack is to use parameters like in Figure 15.



Figure

This important technique must also be used for inserting data from web input as in Figure 16.



Figure

When working with web input, an annoying artifact of the request/response architecture is that the input doesn’t display in the same page as the results. Consider the Add 2 Numbers dialog from Figure 6. If you can see the results of the addition the input fields no longer display the values being added.

## Using AJAX to Update Part of a Page

If we could update just the part of the page that displays the results of our input, the input would still be visible along with the results. Asynchronous Javascript And XML is a technique for updating only a part of a page. Although it is possible to use this technique with pure javascript, browser inconsistencies make using a library like jquery a less frustrating approach. We can get jquery when we load our page from the googleapis page by including this script tag in our script:

<script src="//ajax.googleapis.com/ajax/libs/jquery/2.0.0/jquery.min.js"></script>

By adding a div for the results to go in, and a div for the submit button we can progressively enhance the web page if javascript is enabled to only update the results div when the submit button is clicked. If javascript isn’t enabled the page will behave like it did before. Other than the div and the ids our .cshtml page is largely unchanged. The post request that we get is also unchanged from Figure 8.



The jQuery() selector method also has an alias of $(). You will see many examples where $() is used for jQuery. I have chosen to use the jQuery syntax, because $ is also used in Murach HTML5 as an alias for document.getElementById(). Hopefully explicitly using jQuery reduces any confusion that this causes.

## Uploading Images to a web site

Often we need visitors to our site to at least be able to upload an avatar to our site. Usually it is good for customers as well to be able to post new pictures to our site. This is a common use case, so for some time now there has been a way to do this with http. If we send an http request with content type = “multipart/form-data” we can include an image in that content.



A request like this can be constructed with a form or using html5 javascript. In the case of a form, the form tag must have the attribute enctype="multipart/form-data". There is a “file” type of input so that the user can select the file to upload, then when a regular submit type button is pressed the multipart/form-data request will be sent to the server. This is a form that we can use to send the proper type of image upload request to a server:



Your server code receives the request like any other web input POST request, but by the time your code gets the request the image is ready for further processing.



Note\* The default maximum image upload size is 4,096, 000 bytes (~4 mb). You can change this by editing your web.config file as follows:



## Using Ajax To Drag and Drop an Image into Part of a page:

You can also use Ajax to drag and drop an image in Firefox and Chrome. The code is the same on the server, so it is easy to offer drag and drop on browsers that support it and more conventional form generated requests for internet explorer. After adding the dataTransfer property to jQuery, we need to define 2 new event listeners. The following dragover event listener just stops event propagation and sets the effect to show that a file is being copied on to the server.



The interesting part is the drop event listener. In the drop event listener we go through a number of stages.

1. Turn off event propagation
2. Get the file that was dropped into a FormData object
3. Construct a request and send to the server
4. Listen for the result and display the image if it worked.



Using this approach we can do image upload to a directory that we have control over. This approach doesn’t work for appharbor though as it doesn’t scale particularly well on a cloud based server.

## Deployment

For your site to be viewable on the Internet, you will need to put it on a host that is accessible from the internet. The host that runs your web site will also need to support the resources it uses like database and server side code in a .cs or .cshtml file. A host that provides these resources, up to a point for free is <http://appharbor.com>. Appharbor also offers enhanced services for when your application moves into production and requires more resources. There are also Virtual Private Servers and more traditional hosting companies like GoDaddy.com that can host .cshtml files.

Appharbor uses a program called git to synchronize changes between your personal development computer and the host that is visible from the internet. Git is a popular tool for concurrent programming among multiple developers. It was developed and designed by Linus Torvalds for Linux development. In the AppHarbor context of putting data on a server Git is used somewhat like the ftp program that is described in Murach HTML5. Git, however, has much more functionality to deal with synchronization among multiple users. Git’s synchronization is like optimistic locking in databases. It looks to see if you were working on the latest copy, before accepting changes. If you weren’t working on the latest copy, git has functionality to help resolve the issue. You can get git from <http://sourceforge.net/projects/mingwbuilds/files/external-binary-packages/>. In this package git comes with some other handy unix like tools like wget.

If you are working on your own project in AppHarbor you shouldn’t have any concurrency/synchronization issues so your workflow should be like this:

### To Start a Blank Project:

1. In your browser surf to <https://appharbor.com/>, sign in and click on “Your Applications.”
2. In the “Create New Application” dialog in the center of the screen supply a unique name for your application.
3. Click the “Create New” button
4. In the project panel, on the left hand side of your screen click on the Repository URL button.
5. If a url is displayed copy that URL, otherwise it will say Repository URL copied to clipboard
6. In your file explorer, go to the directory where you installed msys, and click on msys.bat
7. In the Msys shell, type “cd /e/workspace” where “e” is the drive letter that you are working from and “workspace” is where you are keeping your web sites. Press enter.

\*Note – in the MSys shell you can type part of a path and then press the tab key and shell will try to complete your path.

1. Type “git clone <https://rhildred@appharbor.com/richssite2.git>” where the url is supplied from the url copied in step 5. Press enter.
2. You will be prompted for your appharbor password. Note that there is no feedback when you type it in and press enter. You should get a warning that you have cloned and empty repository.
3. In visual studio start a new solution. For the location enter the directory from step 7. For the name you will use the application name from step 2.
4. Create some content (a simple hello world index.html file will do)
5. In your msys window change to the application name directory
6. Type “git add richssite2.sln” and “git add richssite2” using your application name and pressing enter after each step to add the files to revision control
7. Type ‘git commit –am “initial commit” ’ where the message “initial commit” refers to what you just did. Press Enter.
8. Type “git push origin master” to put the code on appharbor. Press Enter. You will need to enter your password when prompted. Remember that there is no feedback for password typing.
9. Go to appharbor. If you click on the name of your application you should be looking at a screen that says your application is active. There is a link on the top right that you can follow to view your app.

## To Add Database to your project:

1. On App Harbor under your application click “addons”, and scroll down to mysql. Click on See More and click to select the free instance Yocto
2. Click on Configuration variables to get the mysql connection string. Click the copy to clipboard icon to the right to get your connection string.
3. In Visual studio paste the connection string into the web.Config in the following manner:



You can also use the connection string with mysql workbench to put data in the database.

## To Add Mysql and Razor to your Project

MySql.Data

MySql.Data.Entities

Mysql.Web

Microsoft.aspnet.razor

## To edit the new database:

1. Open MySQLWorkBench.exe, Click on New Connection
2. In the host field of the Setup New Connection dialog, paste the server from your connection string in your web.Config, UserName=uid, DefaultSchema=database and click on “Store In Vault” to enter the connection string pwd value for your password.
3. Click on Test Connection to make sure that you have done this correctly.
4. Use the SQL Workbench window to edit the new database

In summary, web programming with .cshtml seems like a nice way to introduce the remaining more advanced techniques of file input/output, XML, LINQ and deployment. Java Server pages (.jsp) and .php are also very similar, and a lot of the same techniques apply. C Sharp html pages (.cshtml) is also a powerful method in it’s own right for implementing web applications.

1. Helix84 HTTP persistent connection, client-server connection schema, <http://commons.wikimedia.org/wiki/File:HTTP_persistent_connection.svg> retrieved (April 28) [↑](#endnote-ref-1)
2. Fowler, Martin “Active Record”, <http://www.martinfowler.com/eaaCatalog/activeRecord.html> retrieved (April 29th) [↑](#endnote-ref-2)