Architecture

Architecture is about planning and designing a system such that it is easy to use, manage and maintain.

As with a building we want a building to be easy for us to use, we want the windows, doors and stairs to work well and be usable. We also want a building to be designed in such a way that if something does go wrong or need updating this may be done with minimum of disruption. Lastly, we want the design of a building to be such that it doesn’t start to fall apart as soon as people start to use it.

These points are equally true when building programs be they for the web, the desktop or a mobile phone / tablet.

To set the scene for the module we are going to start by looking at some of the many considerations when thinking about building software.

I want to start by examining a perfectly reasonable looking web application and think about the kinds of problems created by its faulty design.

Once we have done this we will look at an approach to structuring web applications in such a way that we stand a chance of avoiding these and other shortcomings.

What I want to do is to try and get across the importance of architecture and design when building software and we shall use this structure to guide us as we start to learn about programming.

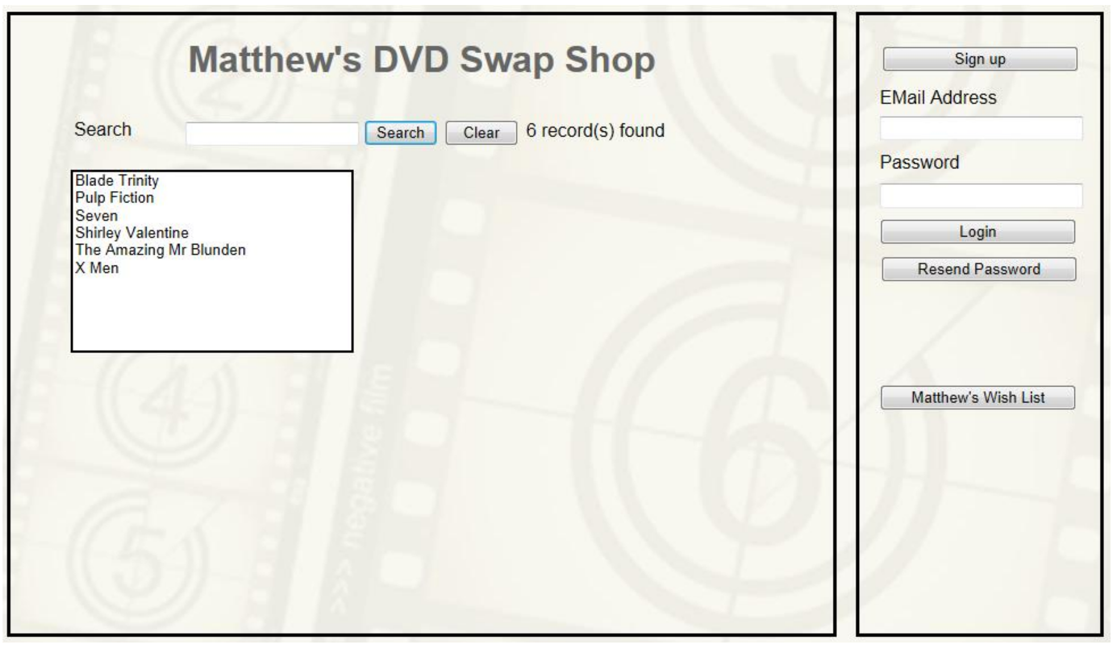
We will look at how our design should consider such things as security maintenance and scalability.

The DVD Swap Shop

Where I am going to start is to look at a program written in ASP & VB.NET. The program is a DVD swap shop.

I have placed a link to the program on the module web site so you may take a more detailed look at it.

If you download and run the program in Visual Studio you will see the following interface.



Blade Trinity
Pulp Fiction
Seven
Shirley Valentine
The Amazing Mr Blunden
X Men

The idea behind the program is that people may swap their DVDs with me on-line. When a person makes an offer on a DVD the application sends me a notification email and then I login to either accept or reject the offer.

To log in to the system you may use the following account

User name	mjdean@dmu.ac.uk
Password	password123

Maintenance - Switching the Database

This version of the DVD swap shop uses an Access database to store the details of all users and transactions.

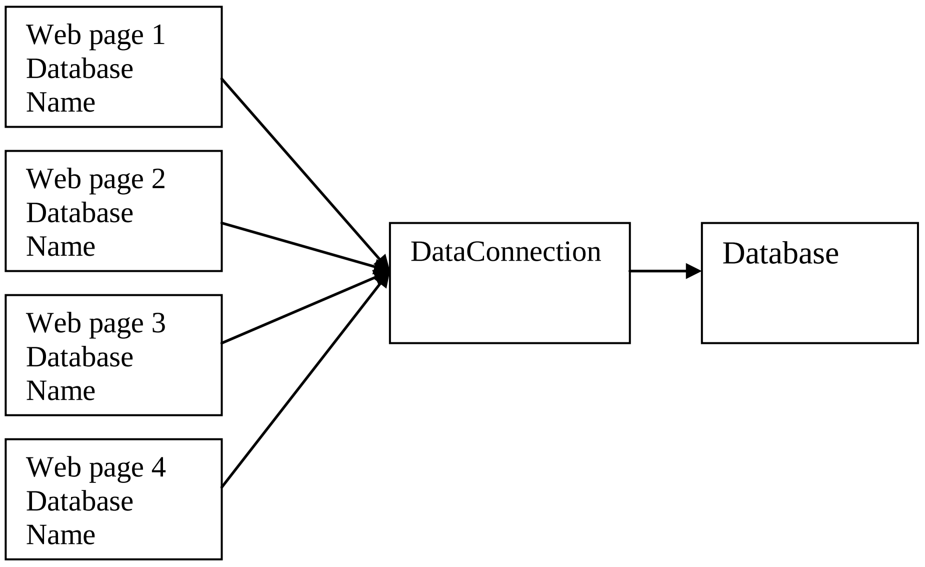
Access has its uses but it isn't the best tool for building commercial sites.

What if we wanted to change the underlying database from Access to say SQL server?

The current database name is DVD.mdb.

Changing to SQL server would result in the database file being called (probably) DVD.mdf

One thing to note about the application is that there are multiple connections across the application to the single database file.



The problem is that each of the connections to the database mentions the name of the database in each page.

What would happen if we had a 100-page site each page having 10 references to the database?

This would require 1000 changes to the program to cope with a single change of database name.

If we are designing a program then we also need to make sure that it is designed in such a way so that it is easy to upgrade specific parts of the system.

Scalability

How many of you here have FaceBook or Twitter on your mobile phone?

Think about what would be required to modify the DVD swap shop such that it works on a mobile phone?

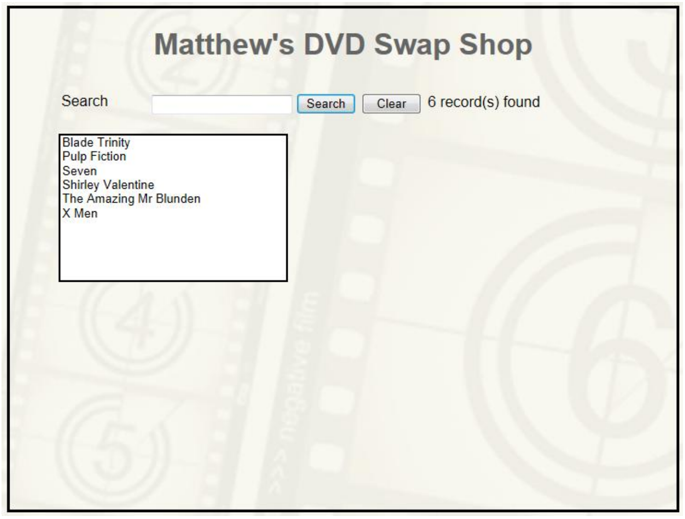
The most obvious thing to do is to re-design the layout of the pages so that it fits on a mobile screen.

However, there is another big problem with the design of my program.

A large amount of the functionality for each page has been placed inside the web pages themselves.

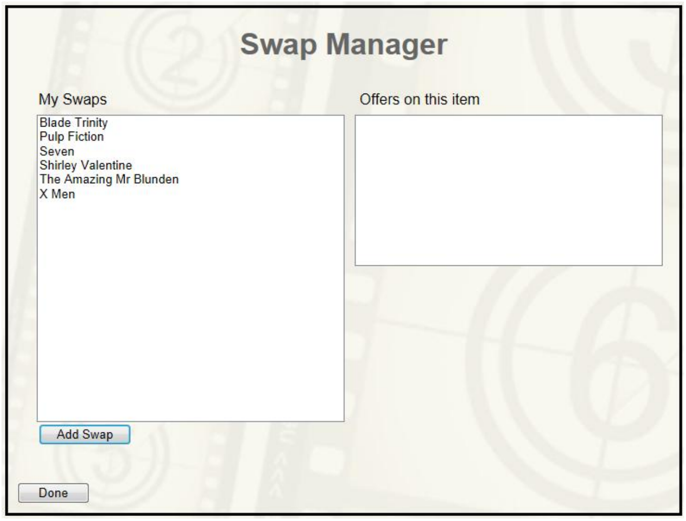
Two obvious places where functionality is both duplicated and not shareable are on the main screen and on the swap manager.

The list box on the main screen provides a list of available swaps...



Blade Trinity
Pulp Fiction
Seven
Shirley Valentine
The Amazing Mr Blunden
X Men

However, the list on the swap manager page...



Blade Trinity
Pulp Fiction
Seven
Shirley Valentine
The Amazing Mr Blunden
X Men

Has identical functionality. Note this is functionality that is <u>duplicated</u> and <u>unshared</u>!

If we wanted another list with the same features we would need to create it from scratch rather than re-using the list we have already created.

In designing our programs, we need to build them in such a way that we are not creating extra work for ourselves. When we design a useful bit of functionality we need to make sure that we may recycle that code when we need it again.

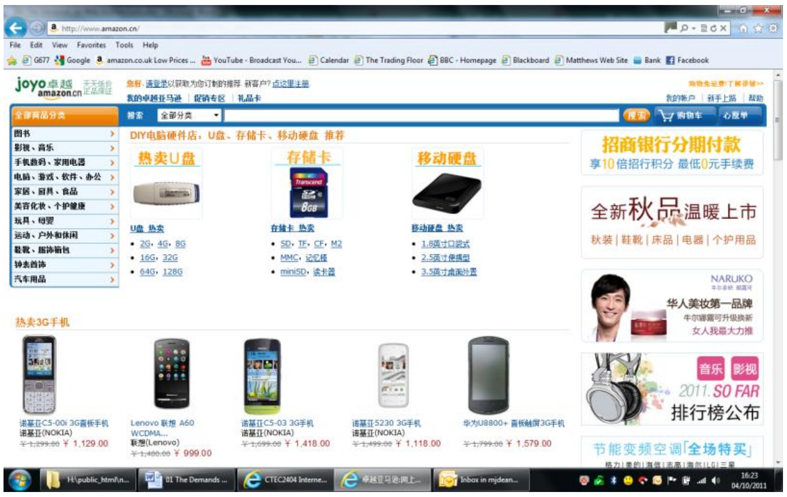
Some Other Issues to Think About

We have only talked about the tip of the iceberg when it comes to the many considerations in designing internet software.

Below are few other issues to get you thinking

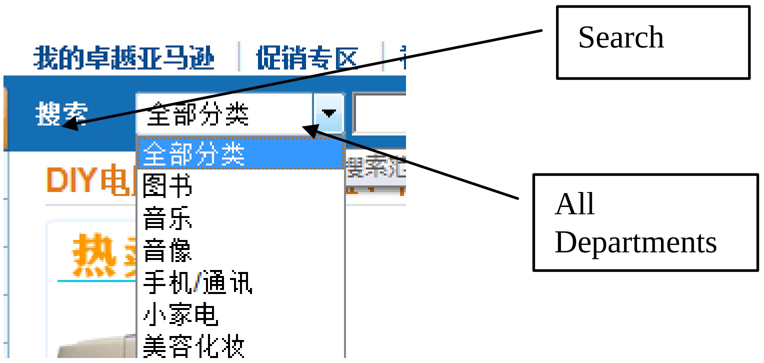
Dealing with International Markets

What do you think of the following page?



It’s all Chinese to me. It should be as this is the front page of the Chinese Amazon site.

Now if you look at the page you will note that the functionality is identical to the UK site.



In writing the code that drives the Amazon search facility does the server care what language it is searching in?

Certainly not!

What this means is that there must be some way of allowing the functionality to operate independently of the language that the user is interested in.

Dealing with Different Computer Platforms

One of the great achievements of the internet has been its ability to operate over a range of different hardware and software platform.

The same page may be rendered on a range of different hardware

- Mobile Apps Apple/Android
- Tablet computers e.g. Apple iPad
- Windows Computers
- Linux machines
- Servers running Apache / IIS

All with a range of operating system / hardware facilities

We need to think about the underlying technology that allows all of this to be possible.

Simple Three Layer Architecture

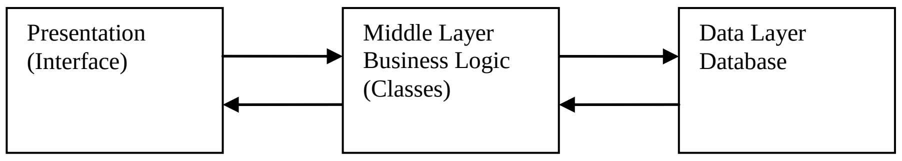
To address the issues mentioned above what we do is take our web applications and split them into layers.

Typically, we use the three-layer architecture.

The three layers are called

- The presentation layer or interface
- The middle layer or business logic
- The data layer

Simple Three Layer Architecture



One very important rule here is that the interface must have no knowledge of the structure of the database.

Communication of data between the presentation and data layers is handled by the middle layer code.

One big payoff of this design is that if we want to change the database then the interface will know nothing of the change.

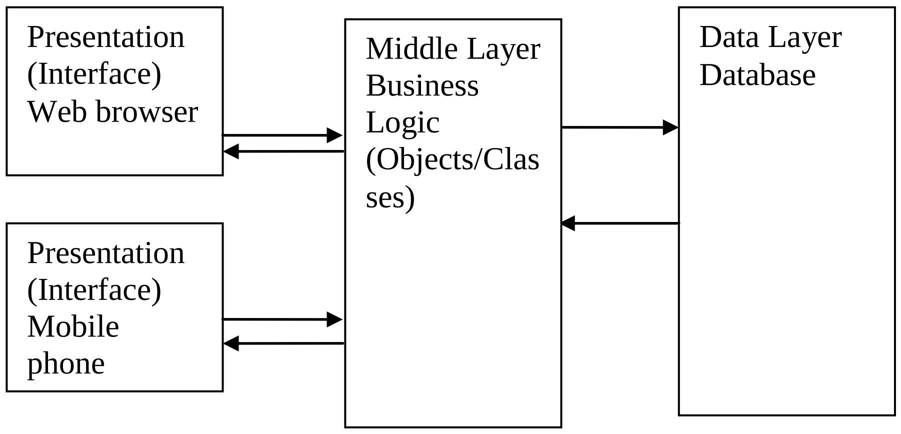
We change the database and then change the internal structure of the middle layer classes to cope with the change.

Communication between the interface and the middle layer remains unaffected.

Another big payoff of this design is that all the functionality of the system is handled by the middle layer classes.

The only functionality that resides in the presentation layer is code that connects the services provided by the middle layer to the interface the user sees.

This makes it reasonably simple to implement the following system.


Presentation (Interface) Web browser
Presentation (Interface) Mobile phone
Middle Layer Business Logic (Objects/Classes)
Data Layer Database

What we have here is an architecture supporting a standard web browser with the addition of a specialised interface for a mobile phone.

The big plus here is that if we change the functionality of the middle layer, any applications that are built on it instantly benefit.

Of course, as with most things in life it is never quite that simple.

Overview of the Address Book Program

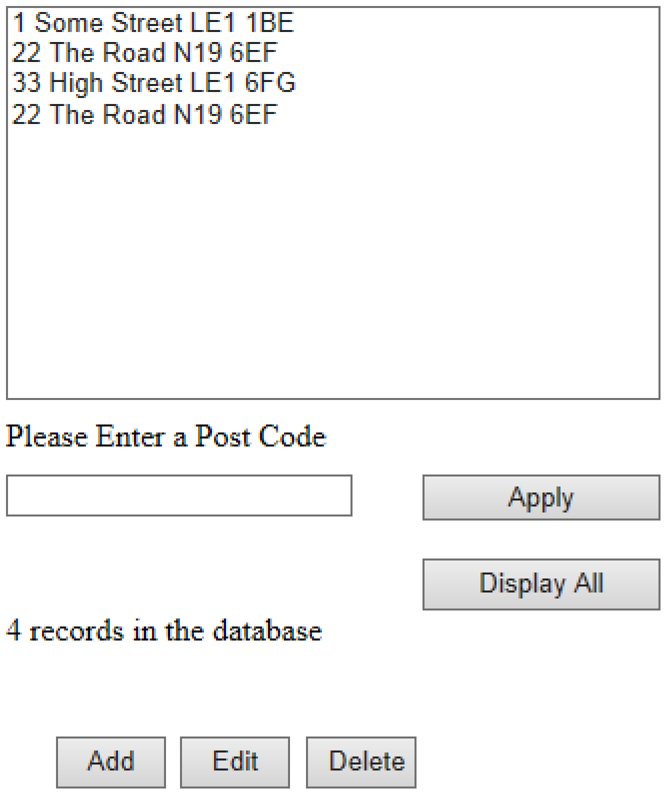
This application is available from the module web site and will be used as the reference standard for all the work on the module.

The address book application is what you might describe as an application with delusions of grandeur.

What I mean is this. The functions of the program are simple, list, add, edit, delete etc. The architecture of the system is not simple. If we followed a simpler architecture we could probably build this system in a day.

By learning about a complex architecture on a simple application it should help you understand why these things matter and more importantly help you to scale up what you learn to a more sophisticated system.

When you run the program, you will see a main screen like this one...



1 Some Street LE1 1BE
22 The Road N19 6EF
33 High Street LE1 6FG
22 The Road N19 6EF

Please Enter a Post Code

4 records in the database

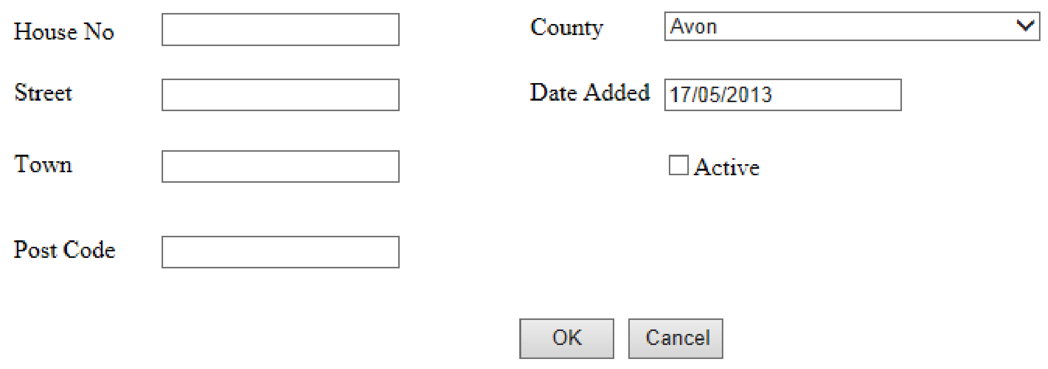
In my example the program takes the form of a simple address book.

In designing your program, you will need to think of something that isn't an address book. You might create a web site that allows you to manage mobile phones or DVDs. It is entirely up to you but you must discuss your idea with your lab tutor before carrying on with the work.

The interface above allows the user of the site to perform the following tasks:

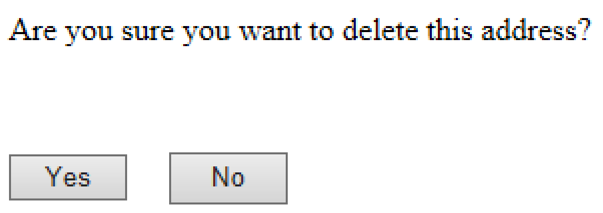
- View a list of addresses in the system
- Filter the list based on a partial post code
- Add a new address to the system
- Edit an existing address
- Delete an address

When an address is added, or edited, the details entered must be validated to make sure that they have been entered correctly.



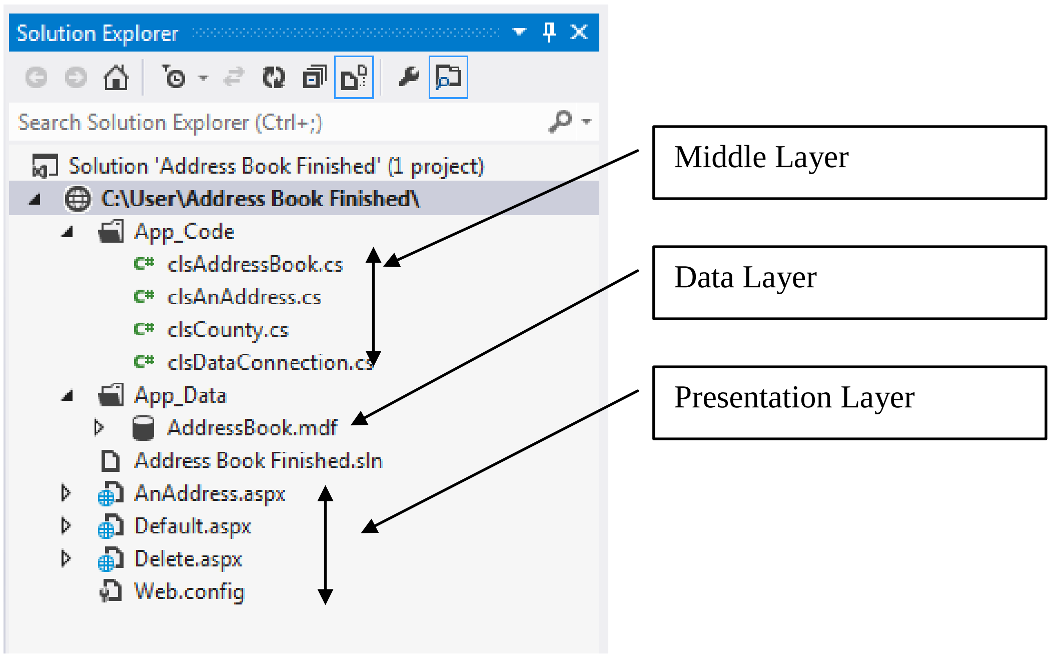
House No		County	Avon
Street		Date Added	17/05/2013
Town			☐ Active
Post Code			

Whenever a record is deleted from the system a confirmation message must be displayed to avoid accidental deletion of data...



Are you sure you want to delete this address?

It is also important to see the different components of the three-layer model within Visual Studio...



This week in the lab you will make a start at building the data layer for your assignment.

Next week you will build the presentation layer and the following week you will link the two together by building the middle layer.