NCC education

Awarding Great British Qualifications

Dynamic Websites

Topic 1:

Introduction to the Module

Scope and Coverage

This topic will cover:

- Introduction to
  - What the unit is all about
  - Web applications and their functions
  - Web development tools and frameworks
  - Client-server applications
  - Web service solutions
  - Static vs dynamic websites
  - Overview of terminology

Learning Outcomes

By the end of this topic students will be able to:

- Define and explain web applications and their functions
- Identify and evaluate appropriate web application development tools for a given scenario
- Identify and evaluate appropriate web application development techniques for a given scenario

Introduction the unit

- The aim of the unit is to enable you to:
  - Understand the different tools and techniques that are used for web application development
  - Develop data-driven websites
  - Apply tools and techniques to build data-driven websites
  - Understand the functions of web services
  - Be able to build and evaluate a dynamic website

The Old Internet

- Static web pages needed to be refreshed by the server every time the data changed
  - Almost everything was gone server side
- Working with web pages was very slow
  - Every action would require sending of user data, forming the page, downloading and parsing of that page
- Client computers fetch information from central servers to display web pages

The Modern Internet

- Web pages are fully functioning applications.
  - Facebook
  - Google apps
- Processing is done in real time on the client side
  - Web page contains instructions from the browser
  - The browser does the work of formatting the data
- This makes the web pages more interactive

Static –v- Dynamic Websites

- Static website tend to be basic pages with no scripting or interactivity except hyperlinks
- Fully static websites are uncommon but a small business with maybe only a one or two pages might still use these
- Dynamic websites includes interactive elements such as search boxes and allows html code to be shared between all pages of the site

Web Applications - 1

- A web application or web app is client-server software application that runs the user interface in a web browser
- The distinction between ‘web’ application and ‘desktop’ applications has become less meaningful as:
  - Most desktop applications incorporate internet functionality
  - Most web applications now imitate the look and feel of desktop applications

Web Applications - 2

- Google Chrome OS is an example of an operating system which is fully designed to work on the web
- Desktop applications still have benefits:
  - Bandwidth is a bottleneck in highly interactive performance
  - Privacy and security is easier when you have control over the data
  - They can be optimised for desktop content

Web Applications - 3

- It is now relatively easy to develop dynamic websites
  - Users expect more user-experience from websites and the site can make a difference between a sale or not
- Web applications have advantages to developers and users:
  - They are easily maintained
  - They can be easily deployed
  - They are always available (provided you have an internet connection)

The Parts of a Web Application - 1

- Web application and commonly e-commerce websites have three “layers” which is responsible for different part of the system:
  - Presentation layer – this handles the user frontend
  - Application layer (logic tier) – this handles the business logic
  - Data layer – this handles the storage and retrieval of data
- This is usually called a ***N-Tier architecture***, where N is the number of tiers.

The Parts of a Web Application - 2

- Website generally consists of:
  - Front-end web server
  - Middle dynamic content processing and generation level application server
  - Database management software

The Parts of a Web Application - 3

- Different tools are used to handle the complexities of the different tiers
  - These are separated out to ensure maximum flexibility when building applications
- At user level, languages such as Javascript and CSS are commonplace
- For application level, server side languages such as PHP and RUBY are used
- For data level, MySQL and Oracle are common.

Presentation

- At the presentation level, much of what is done is to do with the user experience, for example:
  - Formatting of the website properly
  - Handling simple data validation
  - Animation and interface flourishes – what else?
- It is more seamless for the user for this to be handed in the client's browser
- When updated data is required (to update the user interface), that request is sent to the application layer

Application

- At the application layer handles all the functionality
  - Things such as “programming”
  - Sometimes referred to as the “business logic”
- Most often handled on the server via a language such as PHP or RUBY.
- Also serves as the mediator between the presentation and the data.
  - All interaction between these two should be done through the application layer

Data

- The ***data layer*** is responsible for ***optimized data access***
  - This can be done in the application layer, but there are benefits to using a dedicated layer
- This is most often handled using a database management system such as MySQL
- The data layer can handle caching of data, and persistent storage
- It can handle load-balancing between servers

Communication Between Layers

- Communication between layers is handled via a platform independent protocol
  - Such as XML
- This allows for layers to be swapped in and out
  - Provided they can honor the format of communication packets
  - You can change the entire front-end without having to alter any of the other layers
- Communication in a formal format is important
  - The computers that control each layer may be clusters and located in entirely different continents

The Tools of this Module

- In this module, we are going to be developing N-Tier dynamic websites using the following tools:
  - ***Ajax*** and ***HTML*** to handle the presentation layer
  - ***PHP*** to handle the application layer
  - ***MySQL*** to handle the data layer
  - ***XML*** to handle generalize communication
- This is a very flexible set of tools
  - This combination is used for all sorts of real word web applications

Ajax

- Ajax (***Asynchronous Javascript and XML)*** is a set of client-side tools for creating interactive front ends.
- Ajax gets around the need for reloading web pages whenever something should change on the front-end.
  - Using the Ajax framework, we can manipulate web-pages as they sit in the browser
  - We can also send requests to the server to update partial parts of the content

PHP and RUBY

- PHP (originally this stood for ***personal home page***) is a server side scripting language
  - We embed PHP code into our HTML pages
  - The server processes this PHP code before it gets sent to the browser
- PHP incorporates a number of flexible tools that make it useful for implementing an application layer
  - Including native support for MySQL
- RUBY is another language or framework which can be used to make the pages interactive.

MySQL

- MySQL is a database management engine
  - It is popular because it is open source and reliable
- Within MySQL, we make use of relational database structures to store our data
  - We access it in our web pages via PHP
- MySQL lets us concentrate on the business logic and presentation without worrying about data structures

XML

- XML (eXtensible Markup Language) is a Unicode based data format
  - Its design incorporates both the data and the information for interpreting that data (metadata)
- It is the basis of most platform independent web protocols
- It is important to understand the relationship between XML and Ajax

The Client and the Server

- This type of development model is often referred to as a ***client-server application***
- This describes a ***distributed*** application – one where some of the work is handled locally (in your browser) and the rest is handled externally (on a server)
- Strictly speaking, both of these can be located on the same computer
  - And this is common for small-scale development.

Client-Server Architectures - 1

- There are advantages of this approach:
  - Highly maintainable
  - Centralised storage of data can (given good intentions) increase security of critical data
  - Data updates can be applied quickly and shared amongst all users of the data
  - Large-scale optimization can be performed
    - Load balancing and using clusters are common examples
  - Opportunities for interleaving data in ways that are not possible otherwise

Client-Server Architectures - 2

- There are disadvantages too:
  - If the server(s) go down, everyone loses access to the application
  - There are bandwidth considerations when working with large numbers of users
- Other models often used are peer to peer architectures (such as Bittorrent) and desktop deployments
  - Each has its advantages and disadvantages

Web Service Solutions

- A web service is software that allows programmes to talk to web pages instead of using a browser
- It uses XML format to represent the data
- It saves re-inventing the wheel

Evaluation - 1

- Choosing which architecture and tools to use is an important consideration
- In order to be able to assess if we made an appropriate choice, we have to be able to evaluate our success
  - This is made more complicated for web-based development
- We must be sure our tools are properly formed and work across multiple platforms.

Evaluation - 2

- Evaluation of our tools can be partially automated
  - There are many tools that will check to see if our web pages are well formed and conform to standards
- We also have to consider the potential userbase of web applications
  - This involves being mindful of issues such as accessibility
- We will discuss this in a later lecture

Our Task

- Our task then, as the developers of dynamic websites has several substantial sub-tasks associated:
  - Analyse and understand a problem
  - Access and understand our user-base
  - Design a solution
  - Decide on the appropriate tools with our design in mind
  - Develop the client-side applications
  - Develop the server-side applications
  - Evaluate our success

Reinventing the Wheel

- Platform independence reveals a benefit of dynamic web pages:
  - There is a lot of free functionality already out there
  - Existing services such as Google, Amazon and Facebook can be integrated as they use the same protocols and languages
- A lot of work can be done by using existing tools as part of standard ***Application Programming Interfaces (APIs).***

Conclusion

- You will need to learn how to build dynamic websites including:
  - Building client interfaces
  - Building server-side applications
  - Manipulating server-side database
- This will involve encountering a reasonable number of new languages, formats and frameworks
  - Ajax, PHP, MySQL and XML
- You will be able to create dynamic web pages that combine your code with existing APIs

Terminology - 1

- N-Tier Architecture
  - A software application that is explicitly designed in separate component layers
- Client-Server Architecture
  - A software application where a client communicates with a central server
- Ajax
  - A set of tools for delivering interactive user experiences in a browser

Terminology - 2

- PHP
  - A server side scripting suitable for providing the business logic of a software application
- My SQL
  - A database management engine
- XML
  - A strictly formed Unicode format for platform independent data transmission

Awarding Great British Qualifications

Topic 1 – Introduction to the Module

Any Questions?