ECM3408 Enterprise Computing Lecture Notes

Ogaday Willers Moore
Department of Computer Science
School of Engineering and Computer Science
University of Exeter

2015/04/09

1 Introduction

1.1 What is Enterprise Computing?

Enterprise computing is all about combining separate applications, services and processes into a unified system that is greater than the sum of its parts.

> "Enterprise Computing is seen as an integrated solution/platform for separate business problems, such as database management, analytics, reports, which have previously been treated as single business problems treated with specific software solutions."

- `http://www.techopedia.com/definition/27854/enterprise-computing`

1.2 Drivers and Motivations

Up-scaling Enterprises need to be able to upscale to suit an increase in customers.

Outsourcing Specialist service providers will be able to fulfill requirements cheaper and easier than in-house solutions. Circumvents the need to invest in hardware, software and human resources, which requires *Capital Investment*.

1.3 Assumptions

This course makes two axiomatic assumotions.

1. **Enterprise Computing is based on web technologies.** This is not a controversial claim because of the many examples are out there. Netflix, Google, Twitter, Facebook are all worth billions of dollars and are based upon web technologies. Businesses will all need to provide their services and reach their customers over the web in one capacity or another.
2. **Enterprise Computing is done on the Ruby on Rails framework.** This is a controversial claim because there are many models available for web enterprises and Ruby on Rails provides just one instance of that. However this course will only be concerned with Enterprise Computing with Ruby on Rails.

2 Three Tier Architecture

Web programs often follow a *Three Tier Architecture*.

Data Services Tier This provides database access. Having this tier provides abstraction from the database behind the business. This means that migrating services, expanding onto more suitable hardware and scaling up is easier.

Business Services Tier This provides the "business logic", which is the functionality of the enterprise. For instance, Amazon provides for items to be added to carts, carts to be checked out, products to be searched and displayed and many many more small services which comprise their webshop and are all functions of the business. This might require communicating with the data services tier and for the output to be sent to the presentation tier.

Presentation Services Tier This provides the user interface and means that the responses can be adjusted for each user. Specific users can be targeted differently with specialised content; either suitable presentation for a device and browser can be provided (see headers). For example, mobile websites for mobile devices and html generated for internet explorer instead of firefox can be sent to the client. Alternatively, specific demographics can be targeted with specific presentational services. Facebook, Google and Amazon all often provide experimental new interfaces to a subset of their users. This is made possible by the abstraction of the Presentation tier, which can be altered without affecting or interfering with the underlying business and data tiers.

2.1 HTTP: The Hypertext Transfer Protocol

Communication is provided through the Hypertext Transfer Protocol (HTTP). HTTP is a call and response protocol. The client makes a call (Request) and the server sends a response. The response is usually a web page or service. All HTTP communication is in form text (Strings), rather than ints, bools etc.

2.1.1 HTTP Requests

HTTP Requests consist of a request line, request headers and a request body.

The Request Line The request line consists of a method (or verb), a resource and a version. eg: METHOD RESOURCE VERSION GET /images/logo.png HTTP/1.1

The request methods are GET - for receiving data POST - for sending data/send amendment data PUT - for amending data/send replacement data DELETE - for deleting data N.B. GET requests print the data in the URL, which is really bad for anything except debugging. Therefore bank details which are sent with a GET request, for instance, will leave those details in the browser in plain text, so either the request can be intercepted and read, or the browser can later be scraped because there will be a record of the url, and the url, along with the details, will be recorded in server log files.

Request Headers Headers consist of Name: Value pairs on separate lines. eg. Content-Length : 255

The Request Body The request body is empty except for the case of POST requests, which send the data in the body.

N.B. The server can also deny your request! It is under obligation to follow the request, so for instance DELETE google.com HTTP/1.1 would not work.

2.1.2 HTTP Responses

HTTP responses consist of status line, response headers and a response body.

Status Line The status line consists of version, code and reason eg. HTTP/1.1 200 OK The string reason is determined by the code. The code indicates different responses from the server, and the first digit classifies the response. 2 - good 3 - relocation advice, resource has moved. 4 - bad ie. 404 resource not found. 5 - transient bad, server overloaded etc.

Response Header Response headers consist of Name : Value pairs again. Cache-control : caching allowed Connection : Connection preferred Content Length : (length of body)

A case where caching might not be preferred is a live score ticker, so it will continue to update and wont use the cached value of the score. N.B. Can get a snapshot of past browser activity by looking at the browser cache.

Response Body This is often in HTML, but can be anything else (javascript, xml, etc).

3 Introduction to Ruby

Ruby is duck typed, interpreted, high level, OOP capable language. It was invented/designed by Yukihiro "Matz" Matsumoto in Japan around two decades ago and has risen to popularity because of its expressivity and Ruby on Rails, a very successful web framework for building web apps. Ruby is introduced by countless high quality tutorials which I will not try to match.

4 Model-View-Controller

4.1 Design Patterns

Design patterns come about when people find a solution to a challenge and encode them in a pattern. What makes you stand out is the ability to not only learn from your own mistakes, but from others' mistakes as well by using their design patterns.

In the three tier architecture, a common design pattern is the Model-View-Controller which is a pattern upon which the Business tier can be built. In this course, the Model-View-Controller design pattern principles are the rails upon which we are sat.

4.2 The Model

The model is responsible for database access. Why use a model to access the db? Because then the pattern is independent of database architecture, or at least insulated from the

databse and then the enterprise is more portable to different architectures and DB services.

4.3 The View

Responsible for data presentation. By separating the presentation from the data handling, it is easier to focus on creating the best presentation. It is possible to have visualisation experts to specifically work on the view. The view is very important because gives customers strong impressions of the enterprise, and can influence customer decision. For instance, Amazon and Netflix provide different views in some cases in order gauge the effect on customers. Facebook and Google can upgrade and modify their interfaces without risking the other aspects of their service.

4.4 The Controller

Responsible for data processing and business logic. This section sends emails, adds things to cart, does programmatic things.

4.5 The MVC Pattern

This pattern is also reproduced in other languages and frameworks, such as PHP, and most e-commerce platforms follow this design pattern.

See the lecture slides for an example of an implementation of the MVC design pattern in Ruby on Rails.

5 Service Oriented Architecture

Pull away from traditional web development course to think about what makes enterprise web systems as opposed to simple web applications.

The whole thrust of enterprise computing is: What happens when things get big? When you have a large enterprise, you cannot have a single one of anything anymore.

In the three tier architecture, the Presentation and the Business tiers wont do much hard work - All the heavy lifting is done by the Data services tier. Thats why a load balancer is introduced, to make sure the work is evenly distributed across the multiple databases.

Business need multiple databases as they scale for two reasons:

Performance As customer numbers increase, the business must have increased capacity to serve their customers.

Redundancy Business need to protect against fault or attacks which bring down their services. If a server fails they will be unable to serve their customers, which is a cardianl sin in the world of business. As a business scales and gets more custom, redudancy needs to be provided.

5.1 Websites and Web services

Websites act as you expect: human users can navigate their browsers to them and interact in the browser. Web services are strongly correlated with APIs, as they can be queried directly, rather than being rendered with html like a website.

Decompose the services in order to provide understandability, isolation, uniform access(Everyone knows how it works, should all work the same way between services. Should be predictable. This provides a contract between user and service provider - cannot change API functionality once it is released or user will be upset and confused), scalability, and redundancy and failover.

Examples: Both models use the Service Oriented architecture with services decomposed. The advantages are written above, plus with the modularity of the structure of the business, it would be easy to expand to offer a new service as the business develops, such as offering a new insurance field, or a new service, plus if one service fails, the others will be unaffected. Another good example is Google; Google offers many diverse and well developed services (Ads, search, maps, mail, drive, google+, calendar, scholar, videos, the play store and many more) which are well separated and modulated. This is not always possible though, sometimes services are impossible to extract and insulate from the other services. A good example is the login service offered by a lot of enterprises. If that fails, a lot of the other services may be rendered unavailable.

As you can see the paragraphs are separated by any number of blank lines. You can easily get a bulleted list of items with itemize:

- First item Help me!
- Second item
- Third item

A related environment is 'enumerate':

1. First item
    - First
    - Second
    - Third
2. Second item
3. Third item

The last useful one like this is 'description':

First First item

Second Second item

Last Third item

You can have different fonts *italics*, `teletype`, **bold**, sans serif, SMALL CAPS.

5.2 Mathematics

This is a new section instead of 1. You can put mathematics in the text by enclosing it in dollar signs: $x + y_i^{2 \log a} = e^{\pi}$. Here is an equation:

$$e^{i\pi} = -1 \tag{1}$$

Here is a displayed equation:

$$\sum_{n=1}^{\infty} \tan(\theta_n + \alpha) \gg \frac{\phi}{49} \tag{2}$$

That was in equation (**??**).

$$\mathcal{N} = \begin{cases} 1 & \text{if } q < 0 \\ -\pi + \phi^2 & \text{otherwise} \end{cases} \tag{3}$$

Here are vectors and matrices

$$\mathbf{x} = \begin{pmatrix} 1 \\ 2 \\ a \end{pmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 1 & 45 & 23 \\ 2 & 12 & y \end{bmatrix} \tag{4}$$

This is a bit of writing to finish the section before the paragraph.

References The url is `http://www.dcs.ex.ac.uk/~somebody`.