DECLARATION

I, Marechal Adrienne, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of others authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

21/08/2014
Abstract

The aim of this project is to develop a deep understand of a range of web based technologies. Specifically, three technologies will be explored, i.e ASP, PHP and JSP. To facilitate this investigation a specific application will be developed – a web based application which helps people to cook easily. Those three alternatives of the web application will be a support for this study. Indeed, the implementation of those projects is necessary to compare the three languages on different points such as the difference in the way of development, in using design patterns, in developing web services and in the testing phase. The comparison will be done according to a number of criteria: code length and complexity, time of execution and deployment, database connectivity, the implementation of the design patterns and their structure, the frameworks used for testing, etc.. The first step of this project is to study the three languages in order to have a first idea of which language seems better. Then the three applications will be developed to confirm or invalidate this hypothesis.
Acknowledgement

First of all, I would like to thank my supervisor: Andrew Ireland for helping me in choosing and developing an interesting project. This dissertation could not be achieved without his help, guidance and feedback.

I would like to thank also my family and my friends to support me during this year and during the MSc Project.
Table of content

Abstract ................................................................................................................................. 2

Acknowledgement .................................................................................................................. 3

Chapter 1: Introduction ....................................................................................................... 6
  1.1. Project motivation ........................................................................................................ 6
  1.2. Aims and objectives .................................................................................................... 6
  1.3. Professional, legal, ethical and social issues ............................................................. 7
  1.4. The dissertation structure ......................................................................................... 8

Chapter 2: Background ....................................................................................................... 9
  2.1. A web based application ............................................................................................. 9
  2.2. Alternative implementation languages ..................................................................... 9
    2.2.1. PHP (Hypertext Preprocessor) ........................................................................ 10
    2.2.2. JSP (Java Server Pages) .............................................................................. 12
    2.2.3. ASP (Active Server Pages) ........................................................................... 14
  2.3. Unit testing .................................................................................................................. 15
  2.4. Design patterns .......................................................................................................... 16
    2.4.1. Overview .......................................................................................................... 16
    2.4.1.1. What is a pattern? .................................................................................... 16
    2.4.1.2. Categories of patterns .............................................................................. 17
    2.4.2. Patterns chosen .................................................................................................. 18
      2.4.2.1. Command pattern .................................................................................. 18
      2.4.2.2. Front controller pattern ......................................................................... 19
      2.4.2.3. Singleton pattern .................................................................................... 20
    2.4.2.4. MVC pattern ............................................................................................. 20
  2.5. Methodology .............................................................................................................. 21
  2.6. Summary .................................................................................................................... 22

Chapter 3: Requirements .................................................................................................... 23
  3.1. Generic requirements .................................................................................................. 23
  3.2. Case Study .................................................................................................................. 24
  3.3. Summary ..................................................................................................................... 25

Chapter 4: Application Design .......................................................................................... 26
  4.1. The website ................................................................................................................ 26
  4.2. Website requirements ............................................................................................... 29
    4.2.1. Focus on the main functionalities .................................................................... 32
    4.2.2. Data collection ................................................................................................. 34
  4.3. Summary ..................................................................................................................... 36

Chapter 5: Implementation .................................................................................................. 37
  5.1. Design ......................................................................................................................... 37
  5.1.1. Application style ................................................................................................. 37
    5.1.2. Database ...................................................................................................... 38
  5.2. Language specific design .......................................................................................... 39
    5.2.1. Alternative 1: PHP implementation ................................................................ 39
    5.2.2. Alternative 2: JSP implementation .................................................................. 40
    5.2.3. Alternative 3: ASP.NET implementation ......................................................... 42
Chapter 1: Introduction

1.1. Project motivation

When building a highly dynamic and interactive web application there is a broad range of technologies to choose from. In this dissertation I am going to explore those technologies to compare them. To achieve this, an application is created in different programming languages and by using tools and techniques needed.

Nowadays, for people who do not know so much about cooking, they can go in Google and write “find recipes”. They will find different links for recipes websites. Most of those websites provide a large number of recipes. It gives them a lot of wonderful ideas. This is excellent for people who know a little about cooking. Often, when you choose a recipe, most of ingredients are missing in your fridge. This is normal, those websites are done to give you ideas and then you have to buy everything to make the recipe properly. But, what about people who just want find recipes according to their ingredients already in their fridge?

The aim of the development part of this project is to help this people. To provide a web application which gives them ideas of what they can cook. In addition, it is good to give recipes according to ingredients but some criteria are also needed. Indeed, if you get a recipe with all ingredients present in your fridge but you do not have the oven or other facilities, it will be useless. So, the application should take care about that. This dissertation is supported by the application described. It is developed in three programming languages and used each point of comparison contained in this paper.

1.2. Aims and objectives

The aim of this project is to develop a deep understanding of a range of web based technologies in order to compare different web languages. It can break down into a number of objectives:

- Study the three web programming languages – an explanation of each language will be necessary to ensure the understanding of this study. All tools and software (for design patterns, testing, and web service) which will be used will be also explained.

- Develop a set of high level requirements – first, the requirement of this MSc project must be determined. It is a specification of the criteria by which the technologies will be compared. Then the requirements of the application should be defined. Those requirements must cover the generic requirements of the project.
• Develop the application using a number of alternatives – each application will be developed carefully, by using generic building blocks and exploring alternative implementations of the patterns. The application will be implemented in different languages: ASP.NET C#, Java/JEE, and PHP and will used patterns determined in this paper.

• Compare the three alternatives – the application will be developed with three approaches in order to compare them on different points such as the implementation, the patterns used, the web service created and the test phase. It seems relevant to demonstrate the worth of the application for this dissertation, the limitations of each alternative and the problems I encountered.

The three applications developed are a support for this dissertation. It helps to extend the discussion about web based applications developed with design patterns support. This paper shows how each web languages chosen are carried out with design patterns. It allows evaluating the effectiveness of design patterns, how web programming and design patterns fit together.

1.3. Professional, legal, ethical and social issues

Professional issues
BCS, the chartered institute for IT provides a code of good practice and a code of conduct for people who work in computer science. British Computer Society represents people who work in computer science in the UK or internationally. Those two codes will be following as much as possible in the development of this project. Indeed, it is a code often used in company and in teams. This project will be done by only one person and some practices cannot be applied.

Legal issues
In the creation of web sites, some legal issues are involved (BitLaw, n.d and Epsilis, n.d):

- Copyright: Pages of the web site will be built carefully without using, reproducing or deriving from the work made by other people. Pictures on the web site will be drawn or taken in web site which provides pictures that are shared for everyone. There are some libraries which provide licensed pictures. Text of the web site will be fully written. Text written by an author of a web site cannot be reused by someone else without copyright concerns.

- Domain name: The domain name of the web site needs to be checked before. It is forbidden to use a domain name which refers to something or someone which exist. However, in order to prevent other people taking the site it is better to get a multiple domain name (thesite.com, thesite.net, thesite.web, etc).

- Trademark: The web site will take care about words, pictures, slogan, etc which are similar to another product. Nothing in the web site should bring any confusion to customer. However, no links with the name or the logo of a product or a company can be used
- Defamation: The web site will not contain any false statement about a product, a company, a person or another web site.
- Contact details: The web site will provide a minimum of details such as: a name, an email or a phone number and an address
- Data protection: If the web site uses personal data, the information commissioner must be informed.

**Ethical issues**

Even if it is forbidden by laws, all legal issues above will be followed in an ethical manner. The application will be done with my own ideas and skills. Researches and development will be carried out by me only. Assistance will be sought if needed during the entire realisation of this project. The MSc Project will be conduct in the most ethical way in order to prove my integrity and knowledge. This project will demonstrate my capabilities and my motivation. The final application will try to contain all requirements defined and will be handed on time.

**Social issues**

Any culture, any companies, anybody or another will be affected by this project. It will not create any inequality, occupations or disorganization.

**1.4. The dissertation structure**

This paper will be structured into six chapters:

- This one introduces the project, the aims and objectives of the MSc Dissertation
- A second chapter will describe all the background material needed for the good understanding of the dissertation
- A third chapter will detail the generic and specific requirements of the dissertation
- Following, the design of the application will be described. All the requirements for the application will be developed in order to cover each generic requirement explained in the previous chapter
- A chapter will be dedicated in the description of how the application was developed. Each point will be represent in details
- Then, the last chapter will be the comparative evaluation of the application developed according to the generic requirements detailed in a chapter before
Chapter 2: Background

As it said above, this project involves three strands: an application in PHP, the same one in Java/JEE and a last one in ASP.NET C#. This section presents those three languages and shows their strengths and weaknesses. For each of those applications, we focus on implement them with the help of relevant design patterns which are described below. This web based application should be tested to ensure the customer’s satisfaction. A discussion is needed to make the best choice in choosing the most appropriate framework to handle unit tests.

2.1. A web based application

Nowadays, more and more applications are developed in order to propose different functionalities which are not just simple browsing. A web based application refers to all programs which communicate with users by using HTTP. It runs inside a web browser. According to Zhu, W. several categories of web usage exist like browsing, performing transactions or running applications.

Web based applications are used to allow users to interact with objects in the browser. In any web application, requests are processed and responses are generated by a server’s program. The navigation in this kind of application is quite simple for users, similar to traditional GUI (graphical user interface-based) applications. But, in comparing GUI and web based applications, interaction models for a web based application is more limited such as menu bars, manipulate directly an object, select it, etc (Zhu, W., n.d).

Even if web based applications are most of time dynamic and quick, some problems can appear. Indeed, nobody can control rapidity of network connections or the bandwidth. Those can cause a delay in the system response.

2.2. Alternative implementation languages

Several languages are used to develop dynamic web based applications which interact with databases and allow information exchange. Those languages are competing because they allow programming the same kind of application. However, they are not providing the same functionalities, advantages and can be complementary. Each technology has his own field of use depending on requirements, customer’s needs and expectations, and other factors.
The most popular scripting languages are ASP, PHP, JAP, Python, JSP, and others. But ASP, PHP, and JSP are mainly used because it is easier to understand and to learn them. Indeed, there are many famous books, references, bloggers, etc providing for those technologies. More the community which used a language is big, more it easy to understand it.

Rodriguez, H. (2003) states that “one of the most important features introduced in a web browser is a client-side scripting language” in order to allow interaction between the user and the system such as JavaScript or VBasic.

### 2.2.1. PHP (Hypertext Preprocessor)

PHP was created by Rasmus Lerdorf in 1994. It evolved through four iterations. This section introduces PHP, the last version of this language: PHP 5.

#### 2.2.1.1. What is PHP?

PHP is a server-side scripting language used to create dynamic web pages. PHP is an open source scripting language; it is not a proper web standard. PHP files contain a combination of HTML and tags and scripts which are executed on the web server. Using PHP allows programmers developing dynamic applications. Obviously, PHP has a wide range of dynamic features. Those allow delegating a precise behavior of the code at runtime, instead of calling a specific method in the code, a dynamic invocation can be used (Mulder C., 2013).

#### 2.2.1.2. Why use PHP?

To develop a web based application, some elements have to be chosen such as the operating system, the database management system, a programming language, etc. With PHP, it is not necessary. PHP is supported by almost platforms, web servers, databases, and operating systems like Windows, Linux, and Mac. PHP is very stable.

Moreover, with PHP, it is easy to connect a database. Indeed, PHP has excellent database connectivity with SQL database servers, for example using MySQL with PHP is easy and instinctive.
2.2.1.3. How to use PHP?

The PHP code is included in the HTML page and it will be executed when a customer visits the page. The server interprets the code and generates an HTML output. PHP is a language and doesn’t need anything else, another language to create a basic dynamic page. Just create an HTML page with PHP code inside and change the extension of the file in .php.

2.2.1.4. PHP advantages & disadvantages

Most relevant strengths for PHP:
- Provides a good performance like other scripting languages; it’s quick and easy to use
- Large number of functionalities into libraries: create PDF, manipulate cookies, send email, read XML document
- Free language and many documentation provided
- Most people use it, easy to find information or help
- Provides an object-oriented programming

But, some weaknesses exist:
- PHP executes slowly compared to C or other compiled language
- PHP is not good for creating Desktop applications
- To do more than HTML/CSS, JavaScript or other client-side languages are needed
- PHP is not suitable for applications which handle big databases or sensitive and calculation functionalities
2.2.2. JSP (Java Server Pages)

In 1999, Sun Microsystems, developer of J2EE, added something new in their Enterprise Java tool: Java Server Pages. This section introduces this new element.

2.2.2.1. What is JSP?

JSP was created to allow programmers to create web content. JSP are based on Java servlets. This is the latest version of Java for web development. JSP pages are built to develop dynamic or non-dynamic web pages contrary to HTML which allow creating only static pages. With JSP, the content of the page can be changed by using variables. Information can be shared between requests, pages and users. JSP Standard Tag Library (JSTL) specification exists to give to developers all the elements which are provided with this technology such as the access to the database, sending emails, etc.

2.2.2.2. Why use JSP?

Before the creation of JSP, the only way to develop dynamic web pages was the Common Gateway Interface (CGI) but it was not a reliable and efficient solution. Accordingly, when a request is done, the server creates a new operating system, and then, loads an interpreter and a script and executes it.

CGI alternatives have been created to avoid this such as Perl, ISAPI, Java servlets, and others but even if they provide a better performance and scalability, they generate web pages by embedding HTML in code. So, a better alternative, without those problems has been created, namely JSP.

JSP is similar to HTML but when a user calls a page the server execute Java program. Unlike the other languages cited above, it is the JSP pages where active elements are embedded into HTML pages. Moreover, JSP pages can contain directly Java code even if it is rarely needed and not a good solution, it is more difficult to maintain the code if JSP pages contain too much Java code. It is better to call methods from a Java file in the JSP page.

Another advantage of JSP is the compilation. JSP pages are always compiled before the server processes them, interpreter and script are not needed any more. JSP gets around this problem by checking if the page needs to be compiled. If it has never been or if it has been modified, the page is compiled.
2.2.2.3. How to use JSP?

As it said above, JSP allows separating HTML and Java code. A good way to separate properly those elements is to use JSP with J2EE technologies.

The Java 2 platform, Enterprise Edition is a platform-independent used to develop web-based applications by using a combination of XML tags and Java servlets. J2EE is an extension of Java 2 and provides support for several components to ensure an easier way to develop web based applications. J2EE supports servlets and allows using JSP.

With J2EE, Java servlets can be used to handle inputs in the application; Enterprise JavaBean (EJB) is useful for the logic of the application and JSP can take care about the user interface.

2.2.2.4. JSP advantages & disadvantages

JSP has several strengths such as:
- Simple and easy to understand
- Easily combine static templates (like HTML) and code that generates dynamic content
- Use JavaBeans which manage completely components called in JSP tags
- Possible to customize JSP tags library
- Easy to access to standard objects and actions
- Support Java code and standard website development tools

Like everything, JSP has few weaknesses:
- Difficult to get a trace of errors occurred in JSP pages because the JSP pages are translated and compiled
- Requires more time for the first access as JSP pages are compiled on the server
- Requires more disk place to contain the JSP page
2.2.3. ASP (Active Server Pages)

2.2.3.1. What is ASP?

Microsoft’s Active Server Pages (ASP) is another popular technology to develop dynamic web applications. It is similar to server-side JavaScript, ASP is embedded in HTML pages. It is a compile-free application environment using HTML pages, scripts, and server components to create web-based business solutions.

The server executes the code in an ASP page after this one was requested. The result is created and copied in the page, and then the static and dynamic content is sent to the browser. ASP uses server-side scripting to build web pages and the type of the browser doesn't affect those web pages.

2.2.3.2. Why use ASP?

ASP provides a large range of functionalities. Microsoft built a class library in developing their new Framework .NET. With ASP.NET, displaying data is quite simple; no code needs to be written. A MSDN library is available for programmers. It contains all information and examples for ASP.NET.

ASP.NET is completely compatible. Indeed, languages can be mixed into pages; half can be in C# and the rest in VB.NET without creating any problems. All the programming languages are similar; it is just the syntax which changes. With ASP.NET, there are not language incompatibilities.

ASP.NET allows layers separation. It is a common use for companies in order to organize the code. It is easier to maintain and correct the code when it is organized.

The Framework .NET is often used for large application. For example, Amazon, eBay and others use ASP.NET.

2.2.3.3. How to use ASP?

ASP is not a language but a name of the process. It is most commonly used with Microsoft’s VBScript. Microsoft developed a framework called .NET to build dynamic web applications by using the language VB or C#. The Framework .NET is compiled in a Microsoft Intermediate Language (MSIL). This allows a better speed of the execution. MSIL allows loading, storing, initialising and calling methods on object.
2.2.3.4. **ASP advantages & disadvantages**

Nowadays, most of people use ASP.NET because:

- It reduces the code required for large application
- It is easy to perform common tasks such as client authentication, site configuration...
- The application is safe and secure
- The performance is better: just in time compilation, native optimization, etc
- It is easy to maintain: HTML and source code are together
- The application is always available to handle requests: if a process is dead, a new process is created
- It provides an easy use for Web Services and a great support for XML

However:

- ASP runs only on Windows platform and is supported only by Microsoft’s Web Servers
- Web server has to support ASP to implement pages which contain ASP code

2.3. **Unit testing**

A unit test is a part of code written by a programmer to test a specific functionality of an application. It can target a method, a class etc. Tests are often created in a separate folder in the project to ensure the source code and the test code are not mixed.

Nowadays, unit testing gained in popularity. Most of companies use automated tests to ensure the usability of their projects. Those kinds of tests allow executing a test regularly instead of manual testing which is irregularly and limited. Unit testing is often seen as a consuming task but it is the best way to improve the code quality and avoid bugs. They can be detected earlier and easier with unit testing (Zend; 2007).

For that, each language provides a framework to handle all the tests and prevent new changes in the code. Those frameworks are useful to reduce the time consumed in modifying the code, maintenance, refactoring and fixing bugs (Zend; 2007).

PHP can use PHPUnit to provide automated tests. PHPUnit is a framework which organises tests into cases. It is quite simple to run a test with PHP by using the console or a web browser. Then, unit testing in Java can be performed by JUnit. It is simple framework to write repeatable tests. It uses annotations to identify methods that specify a test (Vogel, L; 2007). Finally, there are many frameworks to test the C# solution. The framework chosen does not matter. For the recipe application, it was decided to use the Visual Studio Unit Testing Framework. It was simplest because it was directly provided into the IDE used to develop the ASP.NET C# alternative.
2.4. Design patterns

2.4.1. Overview

2.4.1.1. What is a pattern?

The term “Design patterns” is used to define elements which are reusable for object-oriented software; it is an algorithm which is repeated in a range wide of designs (Tignor W.W., Myrtveit M., 2000). Design patterns are useful for teaching how to design well, standardizing designs of application. It is used to further good design and for “a problem which occurs over and over again in our environment” written by Tignor W.W. and Myrtveit M. (2000) who cited other authors. When a software application is well designed, it has a positive impact for the rest of the development (Gamma E. et al, 1993).

For Gamma et al. (1993), design patterns may be seen as a micro-architecture and have a large number of advantages. Design patterns allow reusing successful designs and architectures, it provides the reuse of experience gained by experimented developers and building blocks to make its own reusable design. There are many specifications which allow a better maintenance of systems which exist. It is known by most of programmers and designers. It also provides an understandable vocabulary for all of them. Using design patterns at the beginning of a project permits to avoid refactoring the code after the development phase.

Gamma et al. (1995) and most of writers are agreeing there are four essential elements for a pattern:

- The pattern name: it allows facilitating dialog between designers; it is a word which describes a design problem
- The problem: it explains what the problem is, his context and when apply this pattern.
- The solution: it describes a general solution to handle a problem; it cannot be a specific design because a pattern can be applied in different situations.
- The consequences: it explains the benefits of using the pattern; it is a description which allows evaluating the pattern according to impacts caused by the pattern.

Web applications and other software applications are different due to their execution process. A web application is carried out by a server and by the user’s workstation, which means it is difficult to work with objects. In addition, Object-Oriented Programming does not support all software design languages for web application. For those reasons, it is harder to use design patterns with web applications (Paikens, A. and Arnicans, G., 2008).
2.4.1.2. Categories of patterns

Referring to Gamma et al. (1995), 23 patterns exist. Those patterns can be classified with two criteria: the scope and the purpose. The purpose describes what the design pattern does and the scope allows specifying if the pattern will be applied to classes or objects.

<table>
<thead>
<tr>
<th>Scope</th>
<th>Purpose</th>
<th>Class</th>
<th>Structural</th>
<th>Behavioral</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Factory Method</td>
<td>Adapter</td>
<td>Interpreter</td>
</tr>
<tr>
<td>Class</td>
<td></td>
<td></td>
<td></td>
<td>Template Method</td>
</tr>
<tr>
<td>Object</td>
<td>Abstract Factory</td>
<td>Adapter</td>
<td></td>
<td>Chain of Responsibility</td>
</tr>
<tr>
<td></td>
<td>Builder</td>
<td>Bridge</td>
<td></td>
<td>Command</td>
</tr>
<tr>
<td></td>
<td>Prototype</td>
<td>Composite</td>
<td></td>
<td>Iterator</td>
</tr>
<tr>
<td></td>
<td>Singleton</td>
<td>Decorator</td>
<td></td>
<td>Mediator</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Façade</td>
<td></td>
<td>Memento</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Flyweight</td>
<td></td>
<td>Observer</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Proxy</td>
<td></td>
<td>State</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Strategy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Visitor</td>
</tr>
</tbody>
</table>

Creational patterns describe the object creation mechanism. Those patterns are classified into creational class patterns which use subclasses to create an object (class instantiation) and creational object patterns which use another object to create a part of his object.

Structural patterns describe the composition of objects or classes. Those patterns identify an easy way to link entities. Structural class patterns compose classes by using inheritance whereas structural object patterns describe how to assemble objects.

Behavioral patterns describe the interaction between classes or objects and give the responsibility. It describes a process and explains how messages are sent to other classes and objects. Behavioral class patterns describe it by using inheritance while behavioral object patterns describe the process of performing a task with a group of objects.
2.4.2. Patterns chosen

To support this study, it is necessary to define the potential of those patterns discovered above. Most of patterns seen earlier are not useful for the kind of application. But some others seem relevant. Web applications involve user navigations, server transactions and database interactions and all of that imply many business rules. To avoid problem during the development phase or in the maintenance, the design approach of the application should be done carefully (Jacynho M.D. et al, 2002).

All patterns cited in the table are described in the table in the Appendix. This section presents only suitable patterns for the application which will be developed.

2.4.2.1. Command pattern

A pattern which seems useful for the future application is the command pattern. Indeed, this pattern provides a flexible approach to handle requests and their functionalities (Jacynho M.D. et al, 2002). This pattern is really useful when there are many requests to the same objects.

![Command pattern diagram](image)

Figure 1 – Process of the command pattern

The invoker which is optional controls the request. It can be also handled by the Client. The invoker receives a command and executes it. This is delegated to the Receiver and it performs the action. The Client represents the application. It creates a command which is sent to the Receiver.
2.4.2.2. Front controller pattern

An alternative of the mediator pattern seen in the table can be used to develop the web application, the front controller pattern. This pattern provides a centralized entry point for all requests in a single controller. It allows handling all requests by only one handler. This can be relevant for the maintainability of an application.

![Front controller structure](image1)

*Figure 2 – Front controller structure*

The handler is responsible for receiving parameters from the Web server and then selecting the right command. Those two responsibilities are shown in the sequence diagram below.

![Typical scenario of the front controller](image2)

*Figure 3 – Typical scenario of the front controller*

The commands are actions described in the Command pattern. The commands are represented as individual objects in order to facilitate interaction between commands and controller. The command completes an action and then transfers it to the corresponding view.
2.4.2.3. Singleton pattern

The Singleton pattern seems interesting to use in the recipe web application. This pattern allows checking if a class is instantiated only once and offers a unique access point to the created instance.

It can be applied to store the connection to the database, to provide a log file or another unique file, or to manage the configuration. It can be used to implement functionalities which would be called by each layer of the application.

In order to ensure the uniqueness of the Singleton, it is necessary to control the instantiation process. A new instance cannot be created; one is only created the first time. The Singleton pattern is composed by three features:

- A private and static attribute which keeps the unique instance of the class.
- A private constructor which avoids creating object from the outside the class
- A static method which allows returning the instance of the class or instantiate it if it is not

2.4.2.4. MVC pattern

This pattern is a common pattern used to develop web based application and other software development. It is not in the previous table because it is an architectural pattern. This pattern is a standard used to design which separate the project into three sections: the model, the view and the controller. The model handles states of the website; it allows notifying the presentation and controllers when a state is changed. The view is the presentation of the application. The controller handles actions sent by the view. User requests are sent to the controller which translates into actions for the model and finally the corresponding view handles the answer. This kind of architecture is relevant for web based applications because it allows separating data and business logic from presentation logic (Zhao W et al, n.d).

Morales-Chaparo et al describes two manners to use the MVC pattern: only server-side or mixed server-side and client-side. The server-side MVC is for an application which is poor in user interaction. The mixed MVC gives more functionality to the client like form validation or other JavaScript methods, this decrease bandwidth usage.

Most of programmers and researchers (Jacyntho M.D. et al, 2002; Sridaran R. et al, 2009; and others) are agreeing to say this pattern allows an easy maintenance of the application due to this separation and facilitate the understanding of the application.
2.5. Methodology

The methodology chosen to achieve this project in a best way as possible is an alternative of the Spiral Model. The Spiral Model is close to an incremental development. It divides into 4 steps:

- Determine objectives
- Identify and resolve risks
- Development and test
- Plan the next iteration

This approach requires user involvement during the development process. Requirements can be changed and validated during each cycle.

This methodology seems appropriate because after determining each requirement for the project and the application, the development phase of the project will start. And during this, the requirements of the application could be changed. Throughout this project, users’ expectations will be respected even if there are not existing users. All requirements are defined in customers’ mind and can evolve during this part of the project.

The project will be built each requirement one by one and will be developed as follow:

- Determine the objectives of the requirement: define the importance and the relevance of it
- Identify the reason of the requirement: define if it is relevant for the comparative evaluation
- Develop and test the requirement
- Plan the next iteration

Figure 4 – MVC Architecture
2.6. Summary

The application will be developed in three object oriented languages: PHP, JSP, ASP.NET C#. Each of those web programming languages has their own strengths and limitations. PHP is good for the performance of the application. It provides a large range of functionalities and it is easy and free to use it. However, PHP is a limited language compared to the two others. JSP supports Java code which allows developing more sensitive functionalities like ASP.NET C#. With those two languages, an application is more safe and secure. However, concerning JSP, the execution of the code or refreshing a JSP page can take some time. And ASP.NET C# is good if you work on a Windows platform. It can be expensive to use it for companies which are used to develop on Linux.

Moreover, the application will implement a number of patterns to ensure its maintainability by the future developers. The Singleton pattern will be used, probably to provide a log file which will make easier the finding of an error in the application. Then, the Front Controller pattern and the Command pattern will be implemented to allow handling the requests made to the database and to provide an entry point for all of them. The MVC pattern will structure the application for a better understanding for the programmers.

Finally, the application will be tested to ensure the quality of the functionalities developed. Each language provides a framework to manage unit tests. PHPUnit will be used for the PHP alternative; JUnit for the JSP solution and Visual Studio Unit Testing for the ASP.NET C# application. All those frameworks have to be installed and studied to implement the unit tests needed.
Chapter 3: Requirements

This study aims to develop a deep understand of the relative strengths and limitations of each technology seen earlier. It is important therefore that each criterion by which the technologies will be compared is mentioned and explained.

3.1. Generic requirements

The previous chapter gives an overview of the generic requirements of this project. Indeed, most of criteria used for the comparative evaluation are developed and explained. The project could be split into two sections: the application itself and the dissertation where the website is the support of this paper.

The application will be developed in different programming web languages detailed earlier (PHP, J2EE and ASP.NET). This will be the first aspect studied for the comparative evaluation. It had seen that those three languages have their own strengths and weaknesses and the development of the applications will help to determine which advantages or limitations are effectively present in a real context.

Then, it seems a good point to develop the application by using some generic building blocks. There are some design patterns which are often used for web application such as Singleton, MVC, Command and Front Controller. Those patterns are detailed above and can be relevant for the comparative evaluation. Indeed, it could be interesting to see how the application can be built with those different patterns and how they are useful for the application depending on the language used.

Another interesting point in the comparative evaluation will be the tests of the application. Each programming language provides a framework or a tool to test the application. This is a way to test the application in an automated way. It could be suitable to see the effectiveness of automated test in each application. Unit testing is an important phase of a project.

Finally, something favorable for this paper will be the web services aspect. Indeed, most of time website needs to use other data or other functionality or service from another website. This is handled by the web services. So, it can be pertinent to use them in the recipe application to get more data and provide a better answer.
## 3.2. Case Study

Each generic requirement should have specific sub-requirements which will be a basis for the comparative evaluation. There are four main points in this part which are detailed below in the table:

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR1 – 1</td>
<td>Compare the application on the code itself</td>
</tr>
<tr>
<td></td>
<td>• Code length for the same function or method</td>
</tr>
<tr>
<td></td>
<td>• Complexity of the code</td>
</tr>
<tr>
<td></td>
<td>• Error handling</td>
</tr>
<tr>
<td></td>
<td>• Maintainability</td>
</tr>
<tr>
<td>GR1 – 2</td>
<td>Define which application is the quickest in terms of:</td>
</tr>
<tr>
<td></td>
<td>• Execution</td>
</tr>
<tr>
<td></td>
<td>• Deployment</td>
</tr>
<tr>
<td></td>
<td>• Database Connectivity</td>
</tr>
<tr>
<td></td>
<td>• Application interaction</td>
</tr>
<tr>
<td>GR1 – 3</td>
<td>Compare the installation of the server needed for each application:</td>
</tr>
<tr>
<td></td>
<td>• Time and difficulties to install it</td>
</tr>
<tr>
<td></td>
<td>• Ease of use</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GR2 – 1</td>
<td>The MVC Pattern:</td>
</tr>
<tr>
<td></td>
<td>• The structure of the project</td>
</tr>
<tr>
<td></td>
<td>• The facility to access to a folder from another</td>
</tr>
<tr>
<td>GR2 – 2</td>
<td>The Front Controller Pattern:</td>
</tr>
<tr>
<td></td>
<td>• How it is working depends on the application</td>
</tr>
<tr>
<td></td>
<td>• The complexity of the implementation</td>
</tr>
<tr>
<td></td>
<td>• Its interest for maintenance</td>
</tr>
<tr>
<td>GR2 – 3</td>
<td>The Singleton Pattern:</td>
</tr>
<tr>
<td></td>
<td>• How it is working depends on the application</td>
</tr>
<tr>
<td></td>
<td>• The complexity of the implementation</td>
</tr>
<tr>
<td>GR2 – 4</td>
<td>The Command Pattern:</td>
</tr>
<tr>
<td></td>
<td>• How it is working depends on the application</td>
</tr>
<tr>
<td></td>
<td>• The complexity of the implementation</td>
</tr>
<tr>
<td></td>
<td>• The interest for maintenance</td>
</tr>
<tr>
<td>ID</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| GR3 – 1 | Compare the code of the web services  
• Length of the code  
• Complexity of the code  
• The existing methods |
| GR3 – 2 | Compare how it is working depends on the application:  
• How the data are collected and treated |
| GR4 – 1 | Compare the code for a test:  
• Length of the code  
• Methods used  
• Ease of writing  
• Maintenance |
| GR4 – 2 | Compare the structure of a test:  
• Initialization  
• Unit test  
• Finality of the test |
| GR4 – 3 | Compare the frameworks used for unit test:  
• Installation  
• Time to execute a test  
• Visualisation of the test’s answer |

**GR**: Generic Requirement  

**GR?: ? is the number of the category (1: the languages; 2: the design patterns; 3: the testing; 4: the web services)**  

**GR1 – ?: ? is the number of the requirement**

### 3.3. Summary

The recipe application will contain different functionalities and requirements. Those requirements will be the basis for the comparative evaluation which is the aim of this project. To ensure the best results, the application will be developed carefully and by following some requirements such as developing the full website, using framework, developing a web service and testing the application. Each of those requirements has at least two others specific requirements which are necessary for the remainder of this paper. Different requirements, available in the table above, will be evaluated in order to define which web programming language is better in which case.
Chapter 4: Application Design

The design of the application is an important phase of the project because it allows defining all the necessary points to cover the requirements seen before. Each of those generic requirements has to be linked with an application requirement to ensure the best evaluation and comparison. First of all, the functionalities of the application must be defined. Then, the requirements for developing this application are needed with an explanation of what this requirement is necessary for this dissertation.

4.1. The website

The aim of a website is to have an impact on the public. Each website includes different features and important aspects. Haque, O. (2010) relates those aspects:

- **Accessibility:** all customers of the website must have an equal access to all functionalities provided by the site. It must be developed and designed in an easy and fast manner.
- **Compatibility:** the website must be developed and designed for all types of browser and not only in a particular platform.
- **Navigability:** the structure of the website has a significant importance; it must be easy to use for customers.
- **Readability:** the website must be readable, it is better to use common typefaces, colors, etc.
- **Usability:** information on the website must be easy to find, pages must be easy to access and attractive, etc in order to drive traffic to the website.

To ensure the respect of those points, the application should contain a menu with four tabs. It allows facilitating the access to each page and understanding easily how the application works. The four sections should provide a different functionality and interact with the database. Those interactions must be optimized in order to ensure the rapidness of the application.

![Example 1 – Home page of the application](image-url)
The first tab of the application should provide a research by categories: Drink, Starter, Dish, and Dessert. It could have a link on a picture which sends the results according to the category chosen. In this new page, each recipe display could be seen with more details (provided by a link also). The details should present the name of the recipe, the preparation and cooking time, and the procedures to do the recipe correctly. In the case of the customer is logged to the application, a button to save the recipe should be added. The button is available just if the customer has not saved the current recipe.

Example 2 – Page of results

The second tab should allow the customer looks for a particular recipe according to different criteria: ingredients, preparation or cooking time and facilities. The ingredients should be mandatory, at least two. Else it does not have any sense to use this page. This is the aim of the application. A list of ingredients could be displayed in which the customer should choose two of them. The other criteria could be optional. Three optional fields also could be added to allow the customer looks for a recipe by ingredients which are not in the list. Once the criteria given by the customer, a page with all the results should be displayed. Like above, each recipe could be seen in details. Moreover, a list with some other recipes which are not in the database could be displayed. Those results should come from another website which provides recipes as this one.

Example 3 – Page for searching recipes
The last one should be the login page. In this page, if the customer has an account he could just log in to the application. But if it does not, fields to register a new customer could be added. After, in the two cases, the customer is signed in to the application. Then, a new tab should appear. It should be a page which displays each recipe saved. The customer could provide a link to see the recipe in details or delete it from the list.

Example 4 – Log in/Register page
### 4.2. Website requirements

This part of the dissertation focuses on each requirement according to the application. In order to compare the three solutions the best as possible, it is necessary to define some points which will help in the comparison phase:

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>GR associated</th>
</tr>
</thead>
</table>
| AR1–1 | Providing a log in/log out to the application to access to a personalize account  
|       | • Check if the customer exists in the database  
|       |   o If yes: Redirection to the home page and add requirement R4  
|       |   o If no: Prevent the customer he made a mistake | GR1–1, GR1–2 |
| AR1–2 | Providing a registration functionality to the application  
|       | • Check if the customer exists in the database  
|       |   o If yes: Redirection to the home page and add requirement R4  
|       |   o If no: Prevent the customer he has already an account | GR1–1, GR1–2 |
| AR1–3 | Providing a research by categories  
|       | • Let the customer choose the category (Drink, Starter, Dish, Dessert)  
|       | • Get the data from the database and display the results in a new page | GR1–1, GR1–2 |
| AR1–4 | Providing a research by criteria such as ingredients, facilities, number of people, food category, preparation time, etc  
|       | • Check if the customer chose at least two ingredients from the list  
|       |   o If yes: Redirection to the pages with all results from the database and execute requirement R7  
|       |   o If no: Prevent the customer to choose two ingredients | GR1–1, GR1–2 |
| AR1–5 | Saving a recipe  
|       | • Check if the recipe is already saved  
|       |   o If no: Display a button “Save”  
|       |   o If yes: Display the text “Already saved” instead of the button  
<p>|       | • When the customer click on the button, the text “Already saved” is displayed and the recipe is added in the database in the preferences | GR1–1, GR1–2 |</p>
<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
<th>GR associated</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>About Data</strong></td>
<td></td>
</tr>
<tr>
<td>AR2</td>
<td>Developing a database</td>
<td>– / –</td>
</tr>
<tr>
<td></td>
<td>• Create the database with all the tables needed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Find many recipes for each categories and add</td>
<td></td>
</tr>
<tr>
<td></td>
<td>them into the database</td>
<td></td>
</tr>
<tr>
<td>AR2</td>
<td>Getting new recipes from the web</td>
<td>GR3 – 1</td>
</tr>
<tr>
<td></td>
<td>• Find a website which shares recipes</td>
<td>GR3 – 2</td>
</tr>
<tr>
<td></td>
<td>• Develop a web service to get those recipes and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>display them in the results page</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>About Testing</strong></td>
<td></td>
</tr>
<tr>
<td>AR3</td>
<td>Testing the main elements of the application</td>
<td>GR4 – 1</td>
</tr>
<tr>
<td></td>
<td>• Develop a test to check if the customer exists,</td>
<td></td>
</tr>
<tr>
<td></td>
<td>is added, is deleted</td>
<td>GR4 – 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GR4 – 3</td>
</tr>
<tr>
<td></td>
<td><strong>About Patterns</strong></td>
<td></td>
</tr>
<tr>
<td>AR4</td>
<td>Singleton Pattern</td>
<td>GR2 – 3</td>
</tr>
<tr>
<td></td>
<td>• Develop a class to create only once instance</td>
<td></td>
</tr>
<tr>
<td></td>
<td>of a file</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Write the log file of the application which</td>
<td></td>
</tr>
<tr>
<td></td>
<td>contains all the actions made by a customer</td>
<td></td>
</tr>
<tr>
<td>AR4</td>
<td>MVC Pattern</td>
<td>GR2 – 1</td>
</tr>
<tr>
<td></td>
<td>• Create three folders which allow structuring</td>
<td></td>
</tr>
<tr>
<td></td>
<td>the project with a model, a view and the logic</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Split the class and move them into the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>corresponding folder</td>
<td></td>
</tr>
<tr>
<td>AR4</td>
<td>Command Pattern</td>
<td>GR2 – 4</td>
</tr>
<tr>
<td></td>
<td>• Create an interface which has the role of the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>command</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Add the inheritance to all the actions classes</td>
<td></td>
</tr>
<tr>
<td>AR4</td>
<td>Front Controller Pattern</td>
<td>GR2 – 2</td>
</tr>
<tr>
<td></td>
<td>• Create an interface to centralize all the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>request to an entry point</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Add the inheritance to all the actions classes</td>
<td></td>
</tr>
</tbody>
</table>

**AR:** Application Requirement

**AR?: ? is the number of the category (1: the application; 2: the data; 3: the testing; 4: the design patterns)**

**GR1 – ?: ? is the number of the requirement**
Most of requirements for the application are mandatory because it is the heart of the application and can help to compare the three languages in terms of code length, time to deploy the application, platform usability. In addition as it is explained later in this paper, the data of the application can come from a database or a web service which gets data from the web. This requirement could be relevant to compare the effectiveness of the three alternatives. Moreover, it seems interesting to develop some unit tests for the application. Indeed, there are many frameworks which exist to write repeatable test. It is a good way to compare the interaction between automated tests and the different languages. Finally, it is mandatory to use patterns during the development phase. However, patterns could be changed during this phase depends on their utility. Consequently, The Command Pattern and the Front Controller Pattern are mandatory or optional depending on their effectiveness.

As it is shown in the table, there are different functionalities which cover the requirements GR1 – 1 and GR1 – 2. Indeed, to ensure the results of the comparison of the three web programming languages, it is necessary to develop enough functionality and use enough code.

There are not requirements linked to the development of the database (AR2 – 1) due to the uniqueness of it. Indeed, each application will use the same database (MySQL) and will be linked to exactly the same. Concerning the second requirements for the database which allows getting more data from the net, it will include all generic requirements for the web services. Indeed, a web service will be necessary to handle this requirement AR2 – 2.

Then, each application requirements for testing or for the design patterns will cover the corresponding generic requirement. Effectively, the development of a unit test allows ensuring the generic requirements concerning the testing phase or the use of the Singleton Pattern helps make sure the generic requirements for it. Same for the others design patterns. The MVC pattern will play a role in the design of the application. Indeed, this pattern is used to structure a project and to ensure their maintainability. The MVC pattern provides strict rules concerning the creation of a project. It has to contain three folders (Model, View, Controller) which handles different layer.
4.2.1. Focus on the main functionalities

The aim of this application is to propose to the customer to look for a recipe by ingredients or by criteria. The customer can search a recipe by different manners.

The application should look for recipes from the database and from other websites which provide the same kind of functionalities. The research can be done directly from the first page by choosing a category or from another page where the customer has to give some criteria. The customer can see all recipes available even if he is not logged into the application.

Figure 5 – Look for a recipe
However, to save a recipe, the customer must login to the application. He can login and logout to the application whenever he wants. If he is registered in the database, the application will provide a new item to allow him seeing recipes that he has saved. If he is not, the application cannot login him and he has to register.
4.2.2. Data collection

The application will provide two kinds of information: some from a database created specifically for it and some directly from the web in order to give a wide range of recipes to the customer.

4.2.2.1. Database

The database will contain at least tables to handle customers and their preferences, details about recipes, and reviews made by customers. The database is represented in the schema below. A MySQL database will be used.

Figure 8 – General overview of the database
There are six tables in the database:

- Customer: all users registered in the application
- Recipe: all recipes available
- Category: Drink, Starter, Dish, Dessert
- Facility: a list of the facilities the most used
- Ingredient: a list with the ingredients the most used
- Preference: a link between a customer and their preference

### 4.2.2.2. Get data from other websites

The aim of this application is to give to the customer many different recipes corresponding to his research. For that, recipes in the database are not enough. The database contain a number of recipes but to ensure providing what the customer is looking for, it can be relevant to get recipes directly from the web also.

There are two ways to get data from the web: get an XML file or get a JSON file. For that, a web service is needed.

This part of the application will be developed to cover the generic requirements concerning the web services. The three alternatives will contain a web service which will get information about recipes from another website. There are different websites which are dedicated in providing data in XML or JSON. Once this website found, the data obtained will be treated and manipulated to give to the customer more recipes. Those other recipes could be displayed like the others if the details about them are similar to the recipes in the database or could be a link which sends the customer to another website containing the details about the recipe.
4.3. Summary

The application will provide four tabs with different functionalities: search by category, search by ingredient, login/logout or register to the application and see the recipes saved. All those functionalities will be developed to cover different requirements seen in the previous section. Each generic requirement will be treated and examined thanks to the application requirements. The application requirements are the way forward to develop the application by ensuring all the comparative points are overlaid.

The requirements concerning the application (AR1 – 1 to AR1 – 5) will ensure the comparison about the code itself. Then, the requirement AR2 – 2 will allow comparing the web service developed in the three alternatives. Regarding the requirement AR3 – 1, it will warrant that the comparative evaluation on the testing phase will be complete due to the unit test implemented. Finally, the requirements about the design patterns (AR4 – 1 to AR4 – 4) will cover the generic requirements about the design patterns chosen.
Chapter 5: Implementation

Once the design of the application made, the implementation phase can be started. To make sure all requirements are covered, the development phase was structured. First, the commonalities were developed (the look of the application and the database). Then, the functionalities of the application were implemented in parallel one by one for each alternative. After, the design patterns were included into each application. And finally, the unit tests of the application were developed for each alternative.

5.1. Design

5.1.1. Application style

The design is an important step of a project. It has to be done with the client or the customer depends on the project. In large or specific project, a prototype can be implemented without any functionality in order to ensure user’s satisfaction. They should be able to use the application efficiently. The interface must be simple, familiar and understandable. Each action of the website should be designed carefully and completely. In this project, there is no user or customer but the design of the recipe application was developed with the same rules. A view of the application is available in Appendix – “Application Design”.

For the three alternatives, the same design was developed in CSS with the help of JavaScript. Those two languages are compatible with the three web languages chosen for this project. Once the design made for the first solution, it was quite simple to transfer the files into the others solutions.

Before creating the CSS file, all the PHP pages were created and were empty. Then, each page was created one by one with the same header and footer. Three CSS file were needed: one for the general style of the application, one for the results page and one for the search by ingredients page. All CSS files are available in the View folder. This separation allows an easy maintenance and understanding of the look of the application.

- The application is available to:
  http://www2.macs.hw.ac.uk/~am978/MScProject1/index.php?action=home
- To log in to the application, you can create a customer or use:
  Login: a.miguel
  Password: antoine1234
5.1.2. Database

A unique database was necessary for the three alternatives. It was implemented with the help of phpMyAdmin. The six tables needed were developed according to the schema available in the Chapter 3.

![MySQL database on PhpMyAdmin](image)

Figure 9 – The MySQL database on PhpMyAdmin

It was necessary to insert data into the database. To ensure a good usability of the application, it was important to insert a consequent number of recipes into the application. Otherwise, the research by ingredient would be ineffective and uninteresting. So, it was decided to find a document on internet which contains a wide range of recipes because inserting data in the database manually was really long and boring. After lot of research, a document was finally found. It was not completely compatible with the database developed before, so some hours were spend in modifying the document. Even if it took time, it was less long than inserting data one by one.
5.2. Language specific design

At the beginning of the application, the requirements AR1 – 1 to AR1 - 5 were developed in the three languages without using any patterns. This first step allows defining and comparing the different programming languages according to the code itself (GR1 – 1; GR1 – 2). Each application was developed in keeping in mind the next phase. It was necessary to develop the code by ensuring the usability of the design patterns.

5.2.1. Alternative 1: PHP implementation

The first language which was used to develop the application was PHP. To develop this application, it was decided to use Geany as IDE. It was started with Eclipse for PHP but there were too many bugs with it.

PHP provides an object oriented programming. So, it was interesting to get some time to find out about this specificity. Indeed, in order to implement the patterns described in a section earlier, it was mandatory to develop the application with an object oriented languages. The application was implemented with classes to ensure the usability of design patterns.

After the creation of the CSS of the application, the structure of the project was made. The MVC pattern was followed. Three folders were created: Model, View, and Controller. In the View folder, three sub folders were also created: one for the CSS, one for the JavaScript files and one for the images into the application.

Then, in each folder, a class was implemented, called like the corresponding folder. The View class allow constructing a new class with the same header for each page. The Controller class is an interface which allows implementing the same method for each action. In an action, if after treating or receiving data it is necessary to create a new page the View class was called. The Model class contain the methods necessary to connect and disconnect the application to the MySQL database. Moreover, in the Model folder it was added six classes: Category.php, Customer.php, Facility.php, Ingredient.php, Preference.php and Recipe.php which correspond to the table in the database.
Once the structure and the model finished, the actions for each pages were started and achieved. An action is linked to the model and to the view. Indeed, most of action needs to get or insert data into the database. So, the constructor of the action calls the method wanted from the corresponding model. Then, it treats the data got and redirects the current page to the next one with an array of the data.

```php
public function __construct($customerLogin) {
    $this->recipes = Recipe::getRecipesSaved($customerLogin);
}

public function execute() {
    $view = new View("save");
    $view->generate(array('recipes' => $this->recipes));
}
```

Example 5 – The content of the controller for saving a recipe

Even if this programming language was a part of the last year in my studies, it was longer than it was planned to develop this first alternative. Classes with PHP were a new notion and it was complicated to understand how it works to create a new page for redirection, how the content of the page was created, where the data come from, how to send them to the new page... There were many different points which were not really clear. But it was not blocking. Once the process for the actions understood, it was quicker to develop the rest of the application.

5.2.2. Alternative 2: JSP implementation

After finishing the structure and the model of the PHP application, the second alternative was started with the JSP language. The best and easiest IDE found to develop this alternative was Eclipse which caused no errors unlike Eclipse for PHP.

5.2.2.1. Focus on the language

JSP is based on Java and is a part of J2EE programming language. To implement this version of the application, J2EE was used. This language includes many different technologies such as JSP, Servlets, JDBC, Web Services, etc and can be helped by the use of frameworks. In our case, JSP was necessary. Indeed, it handled the view of the application. Then after some research, it was decided to use two frameworks: Hibernate and Struts.

Hibernate is an open source Java package, more specifically an object relational mapping library which is used in Java. It allows mapping objects with a relational database. This framework was chosen because with him we just have to focus on the objects and features of the application without caring about how those objects are stored or how to find them.
Struts is an open source Java framework which is used to build web based applications with JSP. It cares about the actions of the application and allows linking the view with those actions. This framework was chosen because it represents the Front Controller Pattern for the J2EE alternative.

5.2.2.2. The implementation

The development of the second alternative was similar to the first one. The CSS was imported to the project and the PHP pages were adjusted and modified to be compatible with JSP pages. Then, the structure of the project was made by following the MVC pattern and by using the existing structure of a J2EE project. The View folder (named WebContent in J2EE application) is composed by the same view page and folders for CSS, JavaScript and images. Then, the same model classes were created.

Then, the framework Hibernate was implemented. All classes on the model were mapped by creating for each model an XML file. The XML file describes how to connect the model with the database. All those XML files are inserted into the file hibernate.cfg.xml and are available in the folder Hibernate. The hibernate XML file also contains the necessary elements to connect the application to the MySQL database.

![XML Files](image)

Example 6 – The models mapped

Once the database connectivity was made, the framework Struts was used. This framework allows handling the actions of the application. An XML file named struts-config.xml list all the actions needed. For each action which contains fields, a form is necessary. This form describes the element needed in the action. In the action class an answer is forwarded to know on which page the redirection will occur.

```xml
  <action path="/login" input="/login.jsp" name="LoginForm" type="action.LoginAction">
    <forward name="log_failed" path="/login.jsp" />
    <forward name="log_ok" path="/index.jsp" />
  </action>
```

Example 7 – Action mapped in the XML file
This alternative was the most complex to develop. For someone who never work with those technologies it can be long to learn and understand the frameworks used. Without studying carefully and deeply, it could be difficult to achieve this solution. It is really complicate to understand how Java fit with Hibernate and Struts if the developer is not familiar with Java first. Those two frameworks bring many errors without any explanation or stack trace to know where the error comes from. For example, if there are mistakes in the struts configuration files, or with a hibernate mapping, or if the server crashed... This alternative was long due to all configurations needed. However, there are lots of documentation for them on the web.

5.2.3. Alternative 3: ASP.NET implementation

Then, the last alternative was in ASP.NET. To develop this solution, there is an IDE provided named Visual Studio. The 2013 version was used.

This alternative can be used with C# or VBScript language. For our application C# was chosen. There is no specific reason for this choice; C# and VBScript have both strengths and weaknesses. But it seems interesting to learn C# because it is a common language nowadays in jobs applications. C# is a language close to Java and his operation is similar to J2EE. There are actions linked to each view pages.

The development of the last alternative was made like the first one. After modifying the PHP pages to use them as C# pages (.aspx), the models were developed and placed in the folder named App_Code instead of Model because this is a particularity of ASP.NET C#. The View folder is exactly the same than the two other applications. Then, the actions were developed in the Controller folder (.aspx.cs).

This solution was the most complicated due to its newness at the beginning. The IDE Visual Studio, or created an ASP.NET C# project was new. The linkage between the different folders was the most complex. Indeed, it is not like in Java where there is just to define the package, with C# the creation of namespace is mandatory. It can be complicate for someone not familiar with this notion. However, after some research and tests, it was found that namespace was not necessary if the Model folder was named App_Code. In addition, to link the view page with the controller only one line is needed at the top of the view page. There were many different small points which can be blocking for a novice developer. But once all this points were assimilated, the application was terminated quickly.
5.3. Pattern implementation

The second step in the development phase was the implementation of the design patterns. So, the requirements AR4 – 1, AR4 – 2 and AR4 – 4 were developed. Those three patterns was used in the same context but implemented totally differently in each alternative.

During the development phase, it was decided to give up the Command Pattern because these design patterns bring any utility to the kind of application which was developed. The Command Pattern and Front Controller Pattern are used together in general but for large application. There is not enough functionality in the recipe application to compare the three alternatives with this pattern. If this pattern was developed, the application would become too complicated. So, it would have been less maintainable and less understandable for future developers.

5.3.1. The MVC pattern

The MVC Pattern was the first used. Effectively, it allows organizing a project. So, it is necessary to separate the model, the view and the controller at the beginning of the project to ensure his effectiveness. It was done in parallel of the step one. Each alternative was developed in keeping in mind that the MVC pattern will be used. Without this, it would have been long and a waste of time to move to MVC architecture.

At the beginning of each project, the three folders needed to ensure this structure were created. They do not have the same name depending on the constraint of the alternative. The View for a J2EE project is called WebContent, or the Model for a C# application is named App_Code. Then, the appropriate classes were developed in each folder and communicate with the corresponding layer.

An example with the PHP alternative for the creation of the home page which allows searching recipes by category:

- The index.php page calls the controller needed. For that, the linkage is made at the top of the page with the line require_once('Controller/HomeController.php');

- The home page is created with the help of those lines:

```php
    if(isset($_POST['disconnect'])) {
        $controller = new HomeController($_POST['disconnect']);
        $controller->execute();
    } else {
        $controller = new HomeController('');
        $controller->execute();
    }
```
They allow calling the controller for the home page and checking if the customer wants to logout to the application. This controller gets all the categories from the database and then constructs the page by sending a table with them and the disconnection if needed.

```php
public function __construct($disconnect) {
    $this->categories = Category::getCategories();
    $this->disconnect = $disconnect;
}

public function execute() {
    $view = new View('home');
    $view->generate(array('categories' => $this->categories, 'disconnect' => $this->disconnect));
}
```

The content of the home page `home.php` is built by getting the information from the array sent by the controller.

```html
<div class="heading">Recipe by category</div>
<table id="home">
    <tr>
        <td width="25%">
            <div class="row">
                <?php foreach ($categories as $category): ?>
                    <td>
                        <a href="index.php?action=category&category=<?= $category->name; ?>" target="_blank">
                            <img src="/view/1img/<?= $category->name; ?>.jpg"/>
                        </a>
                    </td>
                <?php endforeach; ?>
            </div>
        </td>
    </tr>
</table>
```

Examples for the other alternatives are available in Appendix in the sections:
- Interaction between classes in PHP
- Interaction between classes in J2EE
- Interaction between classes in ASP.NET C#

### 5.3.2. The Front Controller pattern

Then, the Front Controller Pattern was implemented but only for the alternative 1 and 3 because for the second one, a framework which handles this was chosen before the development phase. If the Front Controller Pattern was not developed in same time of the first stage for the second alternative, it would have been a waste of time and complicated to change the code. Everything should have been changed. The research made before the developing phase showed that J2EE with servlets and J2EE with frameworks is totally different.
For the first alternative, a controller class was created which is an interface containing one function. Then, all the actions were developed by creating a class which implements the controller class. It allows developing the function defined in the interface. After, the index.php page calls the corresponding controller and executes it depending on the action sent. The index page knows which action is called due to the URL. Indeed, for each action made by the customer, a parameter named action in the URL is available. For example, the URL index.php?action=home sends the customer to the home page. All the actions are available in Appendix – The index.php page.

The second alternative uses a framework to handle the Front Controller pattern: Struts. The actions are categorized into a configuration file. All the actions are available in Appendix – The struts configuration file. Each action needs a class to execute the commands necessary to made it. Then, when a customer wants to execute an action, the information is sent to the struts configuration file and this xml file redirects the customer to the new page wanted and executes the action class in parallel.

The last alternative works similarly to the first one. An interface exists which is used in each action classes. A function is redefined and contains the information and the commands needed to execute an action made by the customer. However, there is not a file like the index in PHP. Each view page contains a link to the controller at the top of the file like in Java with the notion of package. The action wanted is executed in the function redefined from the interface and allows building the new page and his content. An example of a controller is available in Appendix – Example of ASP.NET C# controller.

5.3.3. The Singleton pattern

And finally, the Singleton Pattern was used for the three alternatives. This pattern has no impact on the code developed before it. It is an independent functionality which cares about the log of all the actions made by a customer after his connection to the application.

For the three alternatives, a class namedLogFile was created. This class allows creating only one instance for the construction of the log file. To handle this, a file and an instance of the class were defined and initialised in each class. Then, a method to write in a file was implemented. The code of each LogFile class is available in Appendix – LogFile class.
5.4. **Web service development**

The next phase was the development of the web service to get data from the web (requirement AR2 – 2). This part was optional for this project. However after developing most of the functionalities of the website, it seems interesting to have something different to compare the three solutions.

First of all, some research was necessary in order to find a recipe website which provides data into XML or JSON. Three websites were found:

- [https://developer.yummly.com/](https://developer.yummly.com/)
- [http://food2fork.com/about/api](http://food2fork.com/about/api)
- [http://www.recipepuppy.com/about/api/](http://www.recipepuppy.com/about/api/)

It is the last one which was chosen to keep for developing the web services. The first one and the second one asked an inscription unlike the third one.

Once the website chosen, the web services have been developed. Some research was necessary for each programming languages because this notion of web services was completely new. After that, the first solution made was in J2EE which seems less complicated because it was the language I mastered the most. Then, the PHP solution was developed and finally the ASP.NET C# version. The code of each web services is available in Appendix – Web services. This functionality is well documented on internet and does not bring many difficulties except for the ASP.NET C# solution because it was completely new.

5.5. **Testing**

The last step was the testing phase, the requirement AR3 – 1 was implemented. This is the development of automated tests to check if the main functionalities of the application were working well. Before starting the code and installations necessary to execute the tests, it was important to think about the tests themselves. It seemed not relevant to develop the tests for all functionalities and actions of the application. The aim of this step is to provide enough tests to compare the three alternatives. So, it was decided to develop only one test which was necessary for the comparative evaluation.

First of all, an acceptance test plan was written in order to understand which test will be developed. Then, this test was implemented by following exactly the testing scenario. This phase was a little long because automated test was totally new. Some research had to be made to understand how each testing frameworks works with the corresponding language. Once the frameworks understood and the tests developed, they were executed step by step.
The tests were executed in command line or in the IDE directly. The tests can return two responses:

- OK: the test is working well and there are no errors in the code
- Failed: the test can contain an error or it is the method which is not working

This phase of the project was the most sensitive one. Two of the frameworks used were new. For the J2EE solution it was quite quick because JUnit was studied during this year and it was simple to reinstall it. But for the two others it was a little longer. PHPUnit asks lots of configuration and it is difficult to move to a Window’s terminal when you are familiar with Linux. The commands are not the same and it can be long to find out what you are looking for. Fortunately, later, a tutorial to explain how it was working on Windows was found. For the last alternative it was also complicated to know how install the framework to develop the unit test. It was simplest than PHP but asks time also. Once the installation made for each language the development of the test itself was simple because the three frameworks use similar methods and are well documented.
Chapter 6: Comparative Evaluation

When companies want to develop a new web based application they have to think about requirements of the application and then try to find the most relevant technical use. For that, they have to do some proof of concept for each requirements to ensure the best solution. This part of the project can be a help for people who have to choose between PHP, ASP.NET C# and J2EE to develop a web based application. After some research to have an overview of what searchers think, this chapter provides a comparison of those three web languages according to the application developed and the requirements defined earlier.

6.1. The three languages

6.1.1. Overview

This evaluation part is done according to review made by seekers after doing the same kind of projects. There are a wide range of debates in forums and articles about PHP, ASP.NET and PHP to define which one is a better platform. This part is made to study and understand each consideration in order to provide a comparison without considering a language more than another one. The table below is a conclusion about strengths and limitations of each language.

<table>
<thead>
<tr>
<th></th>
<th>ASP.NET</th>
<th>PHP</th>
<th>JSP</th>
</tr>
</thead>
</table>
| **Execution of the code** | When the URL of the file is requested, the server processes the script. The web browser generates an HTML page and sends it to the browser. ASP code is interpreted. | When the URL of the file is requested, PHP code is executed on the server and the browser only receives an HTML output. | When the URL of the file is requested, there is two solutions:  
  • If the JSP page depends on a non existing servlet class, the JSP page is translated to the servlet class before being executed. The JSP page processes HTTP requests and generates responses.  
  • If it does not exist an instance of the servlet class of the JSP page, servlet class is loaded, an instance is instantiated and the servlet is initialised |

<p>| <strong>Documentation</strong> | Those three technologies are well documented with a wide range of books, forums, and other references. There are the most web technologies used in business world. | | |</p>
<table>
<thead>
<tr>
<th><strong>Database Connectivity</strong></th>
<th>ASP uses a Microsoft database called MS-SQL.</th>
<th>PHP is flexible and can be connected to different databases such as Oracle, MySQL, etc.</th>
<th>JSP uses JDBC application to access to the database. It can be connected to any database: MySQL, Oracle, MS-SQL, etc.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Time to deploy</strong></td>
<td>ASP.NET is a compiled language which means the code needs to be compiled each time he is modified. It can take time. According to the Zdnet study, Microsoft ASP pumped out about 43 pages per second.</td>
<td>PHP is an interpreted language, so when a functionality is changed, changes can be seen without using additional steps. According to the Zdnet study, PHP pumped out about 47 pages per second.</td>
<td>JSP is interpreted and compiled into Java Servlets. According to the Zdnet study, JSP pumped out about 19 pages per second.</td>
</tr>
<tr>
<td><strong>Platform Compatibility</strong></td>
<td>ASP.NET runs only on Windows platform. It is a new language compared to the two others and has to prove his reliability and effectiveness. However, with ASP-Apache installed on the server, ASP can run on Linux.</td>
<td>PHP can run on any platform such as Linux, Unix, Mac, Solaris and Windows.</td>
<td>JSP is based in Java code which can be used over different platforms such as Windows, Unix, Solaris, etc. It has proven his effectiveness and scalability.</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>To develop with ASP.NET, some tools are needed such as Internet Information Services (IIS), MS-SQL database (Microsoft's product) and it has a cost. Moreover, some additional components have to be purchased (for emails, ftp, etc).</td>
<td>PHP is free. It is an open source and a lot of code can be found on forums. It runs on Linux and uses MySQL database and both are free.</td>
<td>JSP is free. But as it said above, using JSP involves J2EE and J2EE has a cost. Servers for J2EE are quite expensive.</td>
</tr>
<tr>
<td><strong>Base Language</strong></td>
<td>ASP.NET is based on Visual Basic language. C# can be used to develop the application.</td>
<td>PHP is based on C++ language. The syntax of PHP is quite the same than C++.</td>
<td>JSP is based on HTML and XML syntax. J2EE is based on Java language.</td>
</tr>
<tr>
<td><strong>Editor &amp; Tools</strong></td>
<td>.NET applications can be developed in an easy way with Microsoft Visual Studio.</td>
<td>PHP can use most of editors such as VI, VIM, Notepad++, etc.</td>
<td>JSP pages can be made in most of editors. But for Java the most used IDE are IBM’s VisualAge for Java, etc.</td>
</tr>
</tbody>
</table>
### Support & Resources

| ASP.NET compare to the two others languages is the less supported languages. There are few developers at Microsoft for making improvements and updates. | PHP is an open source, so there is a wide range of developers who make improvements and updates. Most of them provide also support for the platform. Many forums, articles, and others supports for developers are available online. | JSP/J2EE took more importance over the last years, so there is a large number of people who using it. This allows more support, improvements and updates. Support in J2EE is quite easy to find on the net. |

### Debugging

| ASP provides different ways to debug a code and this is supported by many tutorials. | PHP has many debugging tools and techniques. There are a couple configuration variables which need to be changed to debug a PHP application. | JSP has an API to debug Java code, namely JDB logger. Or a simple way for small bug is to print message to know where the error occurs. |

Concerning the scalability and maintenance, Kohan, B. (2010) shows this is not related to the programming language chosen. It depends on other factors:

- The experience of developers
- The use of the programming practice
- The use of a solid programming framework
- The compliance with guidelines and standards

So, this is not a criterion to define which programming language is the most reliable.

In addition, the performance and the speed of the language is not a good criterion because it depends also of the abilities and knowledge of the programmers. Many contradictions exist about this subject due to the evolution of programming language or people's point of view. Pires, H. (2005) states PHP code runs faster than ASP while according to Kohan, B. (2010), ASP is faster than PHP because of the Framework .NET. This framework uses C# which is faster than PHP. However, by referencing to the Nile benchmark, ASP.NET connected to SQL database is faster than PHP connected with MySQL or Oracle.

The solution performance and speed depends on which language is chosen, but also on database choice, how the program is developed, etc. Each choice has an impact on the rapidity of the application.

According to Miller, G. (2003), for companies which want to build a large application, they have to choose between .NET and J2EE because PHP is better for smallest solutions with less data. In term of cost, .Net is a better solution because it is less expensive to build, to deploy and to maintain. VB.NET developers and .NET servers are less expensive than Java developers and Java servers. Samtani, G & Sadhwani, D. supports also this idea in their published document.
6.1.2. About recipe application

The recipe application was implemented in the three web programming languages in order to compare them and explained which language is better in this context. This kind of application can be developed in PHP, ASP.NET C# or J2EE without any problem but it seems better to develop it in PHP or in ASP.NET C#.

6.1.2.1. The code (GR1 – 1)

The three alternatives were developed in the computer directly (localhost). They are developed with the use of classes. With the ASP.NET and J2EE projects there were no other solutions because they are object oriented languages but the PHP alternative is not. This choice was made for two reasons: to ensure the comparison according to how the application is built and to use design patterns. If classes were not used in the PHP solution, it would have not been relevant to compare it with object oriented languages. Moreover, to use design patterns in an application, the programming language has to be object oriented.

Another advantage of using object oriented language is the notion of inheritance and interface. The inheritance allows an object extending another object. It is used if several objects have the same behaviour, if they use the same implementation. It is a way for code reuse. An interface is a class which contains only the signature of the methods and functions. If a class implements this interface, it has to re-implement all the methods and functions defined in the interface.

The use of those functionalities is exactly the same in the three languages and probably in all object oriented languages. The only difference is the syntax. PHP and J2EE use the same key words:

- For an interface: `Class B implements Class A` where A is the interface
- For the notion of inheritance: `Class B extends Class A` where B is the subclass of A

In ASP.NET C# it is different but it is the same syntax for inheritance or interface `Class B : Class A` where B is the subclass of A or A an interface.

In the three solutions, the length of the code is quite similar for the same method. It depends on the method developed and on the existing functions in the language. Furthermore, the errors in the application are handled in a same way. The three web programming languages provide the try/catch functionality. Then the errors are displayed in a stack trace in the console of the IDE used or directly on the website. There is not a language more complex than another. The three are free documented and it depends on the familiarity of each person with the language.

To conclude, regarding to the code itself, there is not a language better than another. The three can be used. It depends on the resources available in the project, if programmers are more confident with a language than another. The three languages are closed to each other. Even if, one is unknown it is quite simple and quick to learn the basis for a novice developer due to all the documentation and tutorials provided on internet.
6.1.2.2. The quickest solution (GR1 - 2)

J2EE and ASP.NET C# are two similar programming languages. In terms of compilation, they are working in the same way. Both use classes and sub-classes of System.Object. It is not always the same syntax for functionality like the inheritance or the package but they can be used in the two languages similarly. The deployment of both alternatives can take some time because the program needs to be executed before launched in the server unlike the PHP solution. So for the deployment, PHP is the quickest application. Moreover, when a modification is made in the PHP code, the program does not need to be re-executed like the two other alternatives. The execution of the program is simplest with PHP.

In terms of database connectivity, there is not a language quickest than another. The requests made for this project are not enough complex to provide errors or a deceleration to get data in the database. It is the same point for the rapidness of the interaction into the application. There is no real sensitive or considerable functionality. However, concerning the database, there is a major difference in coding the connection between it and the application. For the PHP solution, a function is available which only need the information concerning the database. In ASP.NET C#, a driver needs to be installed and then the connection can be opened with the database details. And for the J2EE alternative, the connection is handled by hibernate which uses a driver JDBC.

```csharp
$db_host = "localhost";
$db_user = 'root';
$db_pass = '';
$db_name = 'MScProject';
$dsn = "mysql://$db_user:$db_pass@unix+$db_host/$db_name";
$conf = mysql_connect($db_host, $db_user, $db_pass);
Example 8 – Database connection in PHP
```

```csharp
MySQLDriverCS.MySqlConnection conn = new MySQLDriverCS.MySqlConnection( 
    new MySQLDriverCS.MySQLConnectionString("localhost", "mscproject", "root", ").AsString); 
conn.Open();
Example 9 – Database connection in ASP.NET C#
```

```xml
<property name="hibernate.connection.driver_class">com.mysql.jdbc.Driver</property>
<property name="hibernate.connection.password"></property>
<property name="hibernate.connection.username">root</property>
<property name="hibernate.current_session_context_class">thread</property>
<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>
<property name="hibernate.show_sql">false</property>
<property name="hibernate.transaction.factory_class">org.hibernate.transaction.JDBCTransactionFactory</property>
<property name="hibernate.session_factory_name">hibernateHibernatePlugins</property>
Example 10 – Database connection in J2EE (Hibernate)
ASP.NET C# is used to make a wide variety of things such as mobile applications, web servers, large web applications, games, etc. The recipe application does not contain enough sensitive functionality to use ASP.NET C# to my mind. PHP is more limited than the other languages and cannot be used in an application which needs lot of database interactions or big database requests. It is good for the application like this one or for simple website which gives only information and has some interactions with the user. It seems that ASP.NET C# and J2EE are better in more complex application.

6.1.2.3. The server (GR1 – 3)

PHP and ASP.NET C# do not need a lot of manipulation to install a server instead of J2EE. There are too many installations which have to be done before coding in J2EE for this kind of application. PHP is much more flexible. It is supported in any platforms.

ASP.NET C# solution forced me to change the operating system. For developing I am using Linux on my computer, but to implement the ASP.NET C# alternative I had to use Windows. It was a constraint because I had to switch between the two operating systems to develop the functionality one by one. To facilitate the development of the functionalities (which were made one by one on each application) it was decided to develop everything from Windows. Java and PHP are multi-platform which is a significant strength. ASP.NET C# can be good for this kind of application only if the operating system used is Windows.

J2EE is definitively not appropriate for this application because it is a waste of time to install all the processes and the server whereas another language could be easier and quicker. ASP.NET C# could be used depending to the developer’s preferences and the evolution of the application in the future.
6.2. Design patterns

6.2.1. MVC pattern

6.2.1.1. Application context

The MVC pattern is used to structure the project and separate the view from the logic and the actions:

- Each page of the website (home, research, log in, recipes saved...) are in the view section
- Each database table is represented as a model in the logic section
- Each action made (with the database, on internet, between pages) is handled by a class in the controller section

This pattern was useful in this project for the maintainability of the solutions and because it is a good point of comparison. The implementation of this pattern is completely different between the three alternatives.

6.2.1.2. Comparison (GR2 – 1)

The difference between the three alternatives in the implementation of the MVC patterns comes from the name of the folder for the model, the view and the controller. In each application, the controller contains only the actions executed by the customer; the view contains the pages of the application and the model contains the elements of the database. The view calls the controller which interacts with the model. Then the database can be treated in the controller or in the model depends on the application.

For the PHP solution the Model-View-Controller is made easily. Below a simple example of those interactions is done. This example shows how the login action works. The code is available in Appendix – “Interaction between classes in PHP”.

![Figure 10 – Interaction between classes in PHP](image-url)
In the second alternative, the MVC pattern is implemented differently. Indeed, a J2EE project has a mandatory structure composed by three folders: Java Resources which contains the source code, the build which contains the executable and the WebContent which contains the view pages and the web configuration. The model and the controller (represented by the action folder) are in the Java Resources folder and the view is represented by the WebContent folder. Below a simple example of those interactions is done. This example shows how the login action works. The code is available in Appendix – “Interaction between classes in J2EE”.

![Diagram](image)

**Figure 11 – Interaction between classes in J2EE**

The MVC pattern in the ASP.NET C# solution is close to the PHP alternative. The model is represented by the App_Code folder. Below a simple example of those interactions is done. This example shows how the login action works. The code is available in Appendix – “Interaction between classes in PHP”.

![Diagram](image)

**Figure 12 – Interaction between classes in ASP.NET C#**
Figure 13 – Architecture of the three alternatives
6.2.2. Front controller pattern

6.2.2.1. Application context

It was decided to use the Front Controller design pattern in order to provide an entry point for handling the requests made by the user. This pattern is used or represented differently in each application but has the same utility. Once a request is made by the customer via the interface of the application, a query is executed which allows getting data needed from the database. Then the response can be sent to give the answer to the customer. With this pattern, if a method of the application needs to change, only a small part of the code has to be changed. This is the same thing if a page needs to be added or deleted. It is a benefit for long term.

6.2.2.2. Comparison (GR2 – 2)

In the PHP solution, to implement the Front Controller pattern, an interface was created. This interface, called Controller.php in the Controller folder, defines a function named execute(). All controllers used after in the application had to redefine this function. The method execute() allows defining which page will be build after the execution of the request. The interface is created in order to centralize the requests in a single component.

Figure 14 – Front Controller and PHP

In the J2EE application, the Front Controller pattern was represented by the Struts Framework. The struts action classes are the controller. It allows determining the next step in the process of the request. Each controller contains a method called execute(). A file (struts-config.xml) present in WebContent/WEB-INF is present in order to determine which action the request is forwarded to. Each view possesses its own controller. The Struts framework follows the MVC pattern architecture. Effectively, the “C” can be represented by this framework which handles the incoming requests.
For the ASP.NET C# alternative, each controller re-implements a method named `Page_Load()` which works like the method `execute()` in the two others solutions. This method allows calling the necessary functions in the model of the application in order to get data from the database.

This pattern was really useful in each alternative. The most complicated and long was in the J2EE alternative. The Struts framework is a little hard to understand if you are not familiar with it. It brings lots of errors of configuration. Indeed, there are many rules to know how to create the `struts-config.xml` file.

In the two others solutions, it was quite the same thing. An interface handles the Front Controller Pattern. Then the structure is similar. The view call the action which is executed after getting data needed from the model.
To conclude in terms of Front Controller Pattern, the alternatives in PHP and ASP.NET C# are better for this kind of application. J2EE, using Struts, could be used in project which provides many actions. It will be easier for maintenance because due to the xml configuration file it is quicker to access to the code wanted.

6.2.3. Singleton pattern

6.2.3.1. Application context

As it said in a previous section, Singleton pattern is useful to save the logs of an application. For this project, this pattern is used as an alternative of a log file. Once the customer is logged in to the recipe application, each of his actions will be saved in the log file. This is a way to check what the customer did if he has any request at any time. It can be used in a real context as a customer service to ensure the customer’s satisfaction and avoid mistakes.

In the same way, it would have been developed a real log file useful for debugging and monitoring. It is exactly the same thing. Only the content of the log file would change in that case. It was not necessary to implement it because there is no added value for this paper. But in a real application it could be really important.

6.2.3.2. Comparison (GR2 – 3)

The singleton pattern was developed in the same manner in each application. A class named LogFile was created in the controller folder. This class allows creating only one instance of a file called logFile.txt. This file contains all actions made by the user.

```java
<----- adriamarec Connection the 23/07/2014----->
Search by category: Starter
Details about the recipe : 40
Search by category: Starter
Details about the recipe : 56
Search by category: Starter
Search by criteria: lemon juice,pepper,Rice
Search by criteria: lemon juice,Farmesan,Rice
<----- Deconnection ------>
```

*Example 11 – A short example of the log file*

There is no better language to implement the Singleton pattern. It is exactly the same thing but not in the same language. The ASP.NET C# solution was a little longer because it was a web programming language totally unknown for me before the project.
The development of this pattern in the three alternatives shows that it is not an effective point for the comparison of the three languages. But it was interesting to implement it to see how it is working to have only one instance of an object. Moreover, it was a good point to see the utility of this pattern. It can be really useful in all kind of application and not only in the web oriented applications.

6.3. Web Services

6.3.1. Overview

Web Services took an important place over the years such as in the e-business, in software development, and in communication industries. Web Services allow exchanging and communicating easily between companies (Zhang, L.; 2004). Nowadays, Web Services are commonly used to share data or information. Web Services are independent from the others platforms. So, they can provide standard interfaces for business services. It is useful in all kind of companies to help them to perform business processes more strongly and efficiently.

More specifically, a Web Service is a way of communication between two electronic machines over a network. They are often provided on the internet as a software function and provide an output in JSON or in XML. Two software systems can exchange or share data more easily and safely.

![Figure 17 – A Web Service (Brittenham, P; 2002)](image)

The service provider allows the Web Service available on internet. Then the service requestor can access to it by opening a network connection and sending a request in XML. The service registry is a central point where programmers can publish new services or find one.
6.3.2. Application context

In this project, the application needs a wide range of recipes to be useful and efficient. It seems interesting to use a Web Service to get more data from the Web. It was clear that it was impossible to create a database which contains all recipes in the world. It would have been long and tiresome. Thus in addition of the recipes from the MySQL database a website http://www.recipepuppy.com was used to get other recipes. This other website provides many recipes which can get by a JSON output. It is not the only one but the aim of this functionality is to compare how it works in the different alternatives and the website used is not relevant.

6.3.3. Comparison (GR3 – 1 and GR3 – 2)

A Web Service can be used with any programming language that allows handling HTTP requests. So, the three alternatives of the recipe website can build the Web Service section. Each language handles a Web Service differently.

The easiest and shortest alternative to implement a Web Service was the PHP solution. There is a function which allows getting and decoding a JSON text from an URL. To handle this, the URL path is treated and then decoded. In ASP.NET C# and J2EE it was a little longer. Indeed, in the ASP.NET C# solution the class DynamicJsonConverter had to be re-implemented in order to read the JSON data. Once this class added, two functions were available to decode those data. The longest and the most complicate one was the J2EE alternative Web Service. The JSON response from the URL needed to be read and treated before parsed. Then for each web service, the recipes are read and added to a list or an array which is sent to the view page. The code is available in Appendix – “Web Services”.
6.4. Testing

6.4.1. Application context

Test an application is an essential aspect of developing in a project. Most of people which are inexperienced in IT do not know the impacts of automated tests. Without any unit tests in an application, it could appear the problems as follow:

- The code is not tested completely: it can bring errors after deploying the project on the net. It is often minor errors but could be problematic for the customer
- The code is not robust: refactoring the code or add a new functionality could bring errors
- The code is not reusable: it could be difficult for another developer to make changes in the code without any test
- The code is not scalable: more the application is large; more is difficult to find a problem inside

All those points have an impact on the program reliability and also on the time allocated to develop and maintain a project. It could be really expensive to move a project.

For those reasons, it seems interesting to provide unit tests for each application in order to compare how they are used in each web languages context.

Before developing the test, a testing scenario was written. The test is described as follow:

- Step 1: Initialize a new customer
- Step 2: Check if this customer exist
- Step 3: Insert the new customer
- Step 4: Check if the customer was inserted correctly
- Step 5: Delete the new customer to ensure the test several times
- Step 6: Check if the customer was deleted correctly

6.4.2. Comparison (GR4 – 1 to GR4 – 3)

The attributes of the customer were defined and then the test was made. Each test was implemented with the method `assertEquals(existCustomer, boolean)` for the J2EE and PHP alternatives or `AreEquals(existCustomer, boolean)` for the ASP.NET C# solution. They test if the customer exists or not. So, the step 2 and 6 should return false and the step 4 should return true. It was implemented a little different between the three alternatives. JUnit provides annotations (@Before, @Test, @After) to structure the execution of the method. The complete code of each alternative is available in Appendix – “Testing.”
With the PHPUnit framework, the class implemented inherits of the class `PHPUnit_Framework_TestCase`. The test was executed in the terminal and gives a written answer.

Example 12 – PHPUnit answer

The JUnit framework provides an answer as a digital image. If the colour is green, that means the test is OK. If the test fails, the colour will be red.

Example 13 – JUnit answer

With Visual Studio Unit Testing, a combination of JUnit and PHPUnit is made. There is the colour green and a line which says that the test is OK.

Example 14 – Visual Studio Unit Testing answer

The results give other relevant information which is the time of the execution. PHP test needs 424ms; JUnit test takes 0.84s and 19ms for the Visual Studio Unit Testing test.
In terms of structure, JUnit is better with annotations. It allows a better maintainability and understanding of the code. For the two other solutions, the initialisation was done directly in the definition of the attributes. Only one function is defined to care about the initialization, the test and the end of the test.

Moreover, JUnit and Visual Studio Unit Testing are two frameworks simple to install due to the corresponding IDE (Eclipse and Microsoft Visual Studio) provided for those languages. With JUnit, only one library was needed to execute the test and one reference for the ASP.NET C# solution. However, in the PHP solution, there is no IDE which helps in the installation of PHPUnit. There were many command lines needed. First, install PEAR (PHP Extension and Application Repository) and then phpunit command (execution of the command: `phpunit C:\wamp\www\MScProjectAlternative1\test\TestCustomer.php`). It is not really documented. It can be a problem.

As it shown above, in terms of rapidity, the best solution for testing is Visual Studio Unit Testing framework, then PHPUnit framework and finally JUnit framework.

To conclude, to choose the best solution it depends on what is needed in the application. To my mind, the most important point is the structure of the test. Indeed, once the installation made the most relevant point will be the maintainability and comprehension of the code for the future programmers. The tests are presents to ensure that the code is working but also to find where the problem comes from, if there is. The rapidity of the execution is also significant. If the code takes too much time to be executed, it can be a waste of time. A study had to be made before the project start in order to know what is needed in this one: the best time, the best structure, the simplest installation... But for this kind of project, the best solution is JUnit, so the J2EE alternative even if the two other frameworks are also a good solution.
6.5. Summary

The aim of this project is to compare three alternatives to build a website in terms of the programming language, the design patterns used, the web services implementation and the testing phase. One alternative was developed in PHP, another one in ASP.NET C# and the last one in Java/JEE with the help of framework (Hibernate, Struts).

After studying the three web programming languages, it seems that the recipe application could be developed with all of them. To confirm this hypothesis the three websites were achieved with all the functionalities requested. Once, it was done, it appears clearly that the recipe application could be implemented in those three programming languages easily because they are all used in a web context. But it was interesting to define which one will be the more relevant for this kind of application. This dissertation shows that the best alternative was the PHP approach. This alternative was quicker and easier in terms of deployment and installation.

This project allows demonstrating that PHP is better for web oriented applications which have limited functionalities and access to the database. Otherwise, if the application needs many connections to a big database, and sensitive functionalities, it can be used to implement it in Java/J2EE which allows handling errors easily and avoid slow database answer. ASP.NET C# is good in the same context depends on your choice and is often used for mobile apps, destocks apps... If it is a large project, the most and easiest maintainable programming language has to be chosen. So, ASP.NET C# or J2EE will be studied. For that, there are many proofs of concepts which are done. That allows ensuring the best solution. Then, it depends on what the client wants. Moreover, often in companies, the language chosen to develop a project comes from the employees’ skills. Indeed, if they are better in J2EE, the application will be in J2EE. This is an important point in the choice of the language.

In terms of designs patterns, it depends on which pattern is used. For the MVC pattern or the Singleton pattern, there is no better solution. The MVC pattern can be used in the three alternatives without any difficulties. It is just necessary to know that for the J2EE and ASP.NET C# alternative there are some mandatory folders which play the role of the controller or the view. Concerning the Singleton pattern, the only thing which changes is the language used. There are no differences between the three languages. It was used similarly. However for the Front Controller pattern, even if the result is exactly the same, the way to implement it was different in the three applications. The J2EE alternative used a framework to handle it and the two others implemented an interface. With this pattern, it is clear that the J2EE application is too complicated for this kind of application.
The testing phase was a good point to compare the three applications. Each application is different from each other. It showed that the best solution for the recipe application was the J2EE alternative. However, this is not true for all kind of application. The framework JUnit is slower than PHPUnit or Visual Studio Unit Testing. Each framework has strengths and weaknesses. Even if JUnit is slow, it is good to structure the testing code with its annotation. Visual Studio Unit Testing framework is the quickest solution. JUnit and Visual Studio Unit Testing are quite simple to install them whereas PHPUnit is more difficult and less documented. It seems that PHP unit testing is less used in projects.

To conclude, each project needs to be studied and evaluate before starting it. The first phase of a project is really important. If the objectives of a new project are not clearly defined at the beginning, it can bring many confusions and errors in the next phases and for developers. Programmers have to know what the client want exactly to ensure developing the best solution.
Chapter 7: Conclusion

The goal of this project was to study three web programming languages in order to define which language is the most appropriate in which context. Each feature, strengths and limitations of those languages were investigated to get a first idea of which language is better than another. Then, to ensure the validity of the results, a web based application was developed in those three different programming languages.

The first weeks of the project was focused on developing the application. Then, this paper was started in parallel of the development. And finally the last weeks were concentrated in the redaction of the dissertation. The plan made at the beginning of the project was almost respected. This plan is available in Appendix – “Project Plan”. The only difference is that the three alternatives were not developed one by one but in parallel. It was easiest to take notes for the comparative evaluation. Furthermore, the three applications were almost achieved at the middle of July, so this paper was started and a first version was handed in August 4th.

During the development phase, after creating the design of the application and the database, the implementation of the website was started. Each functionality of the application was developed one by one in each language. Once a functionality was finished in an alternative, it was developed in the two other languages. It allows facilitating the comparative evaluation. All the mandatory requirements were carefully developed. Once all those requirements were created, some optional requirements were also developed. The development phase allowed determining which language was the most appropriate for the recipe application. The PHP alternative was the simplest and quickest one. The two other programming languages should be used for more complex application.

This project was a good choice. It allows me learning many things I did not know before. During my studies I learnt many programming languages including J2EE and PHP. I had some notions of how start those two alternatives; it was a good point to practice with those languages. Finally, I found the PHP alternative more difficult than I thought because I had to develop it with classes and it was a notion I never used before. For the J2EE alternative, I think I would never finish it if I have not work with the frameworks used in this project during one of my work placement. I found interesting to develop applications in those languages to apply what I learnt. Concerning ASP.NET, it was totally new and I was not really confident about using this language but it was necessary to develop this alternative to ensure a good comparative evaluation. Moreover, I never used design patterns, web services and unit tests before. This project was a challenge for me. The web services and the testing phase were also interesting for my personal skills. This project should be really instructive for my future work.
With more time, it could be interesting to add some features to the application. Indeed, the actual website is better in PHP because there are not enough sensitive functionalities. It could be relevant to extend the website in another way. Most of large website such as Amazon, CDiscount, TripAdvisor... are created in J2EE or ASP.NET C#. There are more interactions with database, with other websites. For an e-Commerce website it is better to use one of those two languages. To extend the comparative evaluation of this dissertation it would be interesting to develop an e-Commerce application, something similar to Amazon. An application like this should show the PHP limitations.
References


Samtani, G. and Sadhwani, (n.d) D. *Web Services and Application Frameworks (.NET and J2EE)*.


Zhao W., Kearney D., Gioiosa G. (n.d) *Architectures for Web Based Applications*, University of South Australia.

**Internet resources:**


Bibliography


Internet resources:


Anonymous, Literature review. [Online] Available at <http://briggs.myweb.port.ac.uk/jimapp/SUMS/opms/Literature%20review.htm>


Anonymous, What Are Active Server Pages (Classic ASP)? [Online] Available at <http://www.webwiz.co.uk/kb/asp-tutorials/what-is-asp.htm>


Epsilis, Legal issues concerning websites. [Online] Available at <http://www.epsilon.co.uk/features/legal_issues_for_web_sites.aspx#contactdetailsmust>

Appendix

Application Design

Home Page

Copyright: The images used on this page were taken in Google Image.
- Drink: [http://hqdesktopwallpapers.net/drink/](http://hqdesktopwallpapers.net/drink/)
- Dish: [http://www.free-picture.net/food/barbecue/china-chicken.jpg.html](http://www.free-picture.net/food/barbecue/china-chicken.jpg.html)
- Dessert: [http://naenis.wordpress.com/](http://naenis.wordpress.com/)
<table>
<thead>
<tr>
<th>Date</th>
<th>Meal</th>
</tr>
</thead>
<tbody>
<tr>
<td>28/07/2014</td>
<td>French Onion Soup</td>
</tr>
<tr>
<td>17/08/2014</td>
<td>RICE SOUP</td>
</tr>
</tbody>
</table>

See more...  ☑
Interaction between classes in PHP

**login.php**

```php
<form action="index.php?action=login" class="group" method="post">
  <input name="login" type="text" class="text" placeholder="Username" required autofocus />
  <input name="password" type="password" class="text" placeholder="Password" required />
  <button class="button" type="submit">Sign in</button>
</form>

<?php
if(isset($_POST['login'])) {
    echo '<span id="error_login" class="error">Error Login - Try Again</span>';
    echo '<script>setTimeout("revertErrorLogin()",5000);</script>';
}
?>
```

**LoginController.php**

```php
class LoginController implements Controller {
    private $resp;
    private $login;

    public function __construct($login,$pass) {
        if ((isset($login) && !empty($login)) && isset($pass) && !empty($pass)) {
            $this->resp = Customer::existCustomer($login,$pass);
            $this->login = $login;
            LogFile::writeLog("<----- Connection ------->");
        }
    }

    public function execute() {
        if($this->resp) {
            session_start();
            $_SESSION['login'] = $this->login;
            header('Location: index.php?action=home');
            exit;
        } else {
            $view = new View('login');
            $view->generate(array());
        }
    }
}
```

**Customer.php**

```php
public static function existCustomer($login,$pwd) {
    parent::connect();
    $sql = "SELECT * FROM customer WHERE login='$login' AND password='$pwd'";
    $resp = mysql_query($sql);
    if (!$resp) {
        echo 'Error query for customer : '.mysql_error();
    }
    $num_rows = mysql_num_rows($resp);
    parent::disconnect();
    return $num_rows == 1;
}
```
Interaction between classes in J2EE

login.jsp

```html
<div id="login_div">
  <h3>beam:message key='login_already'/></h3>
  <form action="/login">
    <input name="name" type="text" class="text" placeholder="Username" required autofocus />
    <input name="pass" type="password" class="text" placeholder="Password" required />
    <input type="submit" style="background:" style="width:70px;">
    <bean:message key='login.send'/></form>

    <script>
    if(request.getSession().getAttribute("login_failed") != null &
      (Integer) request.getSession().getAttribute("login_failed") == 1) {
      span id="error_login" class="error">
        bean:message key='error.login'/
    </script>

    </div>
</form>

LoginAction.java

```java
public class LoginAction extends Action {
  public ActionForward execute(ActionMapping mapping, ActionForm form,
    HttpServletRequest request, HttpServletResponse response)
    throws Exception {
    String send = "login_failed";
    boolean found = false;
    LoginForm loginForm = (LoginForm)form;
    ServletContext context = request.getSession().getServletContext();
    SessionFactory factory = (SessionFactory) context.getAttribute(HibernatePlugin.SESSION_FACTORY_KEY);
    Session session = factory.openSession();
    List<Customer> listPerson = session.createQuery("From Customer").list();
    for (int i=0; i<listPerson.size(); i++) {
      if (listPerson.get(i).getPass().equals(loginForm.getPass()) &
          listPerson.get(i).getLogin().equals(loginForm.getName())) {
        send = "log_ok";
        found = true;
        request.getSession().setAttribute("login", listPerson.get(i));
      }
    }
    if(!found) {
      request.getSession().setAttribute("login_failed", 1);
    }
    session.close();

    // write in the log file
    try {
      File report = new File("logFile.FILE_LOG");
      report.delete();
      logFile.getInstance().writeLog("----- Connection -----");
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }

    return mapping.findForward("send");
  }
}
Interaction between classes in ASP.NET C#

**login.aspx**

```html
<div id="login_div">
    <h2>Already registered</h2>
    <asp:TextBox ID="login" placeholder="Username" runat="server" />
    <br />
    <asp:RequiredFieldValidator ID="login_required" runat="server" ControlToValidate="login" ValidationGroup="login" BackColor="Red" />
    <asp:TextBox ID="password" TextMode="Password" placeholder="Password" runat="server" />
    <br />
    <asp:RequiredFieldValidator ID="password_required" runat="server" ControlToValidate="password" ValidationGroup="login" BackColor="Red" />
    <asp:Button ID="signin" CssClass="button" Text="Sign in" Width="80px" ValidationGroup="login" runat="server" />
    <span id="error_login" class="error">
        <asp:Label ID="error" runat="server" />
    </span>
    <script>setTimeout("revertErrorLogin()",5000);</script>
</div>
```

**login.aspx.cs**

```csharp
// control for "already registered"
string login = Request.Form["login"];
string pass = Request.Form["password"]

if (login != null & pass != null)
{
    if (IsExist(login, pass))
    {
        Session["login"] = login;
        Server.Transfer("index.aspx", true);
    }
    else
    {
        error.Text = "Error Login - Try Again";
    }
}
```

**Customer.cs**

```csharp
private static Boolean IsExist(string login, string pass)
{
    Boolean yes = false;
    MySqlConnection conn = new MySqlConnection("localhost", "msproject", "root", ").AsString();
    conn.Open();
    MySqlCommand cmd = new MySqlCommand("SELECT * FROM Customer", conn);
    MySqlDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        {
            yes = true;
        }
    }
    reader.Close();
    cmd.Dispose();
    conn.Close();
    return yes;
}
```
The index.php page

```php
public function execute() {
    try {
        session_start();
        if (isset($_GET['action'])) {
            if ($_GET['action'] == 'home') {
                if (isset($_POST['disconnect'])) {
                    $controller = new HomeController($_POST['disconnect']);
                    $controller->execute();
                } else {
                    $controller = new HomeController('**');
                    $controller->execute();
                }
            } else if ($_GET['action'] == 'search') {
                $criteria = new CriteriaController();
                $criteria->execute();
            } else if ($_GET['action'] == 'searching') {
                $criteria = new SearchController($_POST['ingredients'], $_POST['ingredient1'],
                                                  $_POST['ingredient2'], $_POST['ingredient3'],
                                                  $_POST['facility'], $_POST['preparation'], $_POST['cooking']);
                $criteria->execute();
            } else if ($_GET['action'] == 'details') {
                $details = new DetailsController($_GET['recipe']);
                $details->execute();
            } else if ($_GET['action'] == 'category') {
                if (isset($_GET['cate'])) {
                    $searchCat = new CategoryController($_GET['cate']);
                    $searchCat->execute();
                } else {
                    header('Location: index.php?action=home');
                }
            } else if ($_GET['action'] == 'login') {
                if(isset($_POST['login']) && isset($_POST['password'])) {
                    $custom = new LoginController($_POST['login'], $_POST['password']);
                    $custom->execute();
                } else {
                    $custom = new LoginController('**', '**');
                    $custom->execute();
                }
            } else if ($_GET['action'] == 'logout') {
                session_unset();
                session_destroy();
                header('Location: View/login.php');
                exit;
            } else if ($_GET['action'] == 'register') {
                if(isset($_POST['firstname']) && isset($_POST['lastname']) && isset($_POST['username'])
                    && isset($_POST['email']) && isset($_POST['password'])) {
                    $add_custom = new RegisterController($_POST['firstname'],
                                                      $_POST['lastname'],
                                                      $_POST['username'], $_POST['email'], $_POST['password']);
                    $add_custom->execute();
                } else {
                    $add_custom = new RegisterController('**', '**', '**', '**', '**');
                    $add_custom->execute();
                }
            } else if ($_GET['action'] == 'save') {
                if(isset($_SESSION['login'])) {
                    $save = new SaveController($_SESSION['login']);
                    $save->execute();
                } else {
                    header('Location: index.php?action=home');
                }
            } else if ($_GET['action'] == 'deletePref') {
                if(isset($_SESSION['login'])) {
                    $delete = new DeletePrefController($_GET['id']);
                    $delete->execute();
                } else {
                    header('Location: index.php?action=save');
                }
            } else if ($_GET['action'] == 'savePref') {
                if(isset($_SESSION['login'])) {
                    $delete = new SavePrefController($_GET['recipeid']);
                    $delete->execute();
                } else {
                    header('Location: index.php?action=home');
                }
            } else {
                throw new Exception('Invalid action');
            }
        } else {
            header('Location: index.php?action=home');
            exit;
        }
        catch (Exception $e) {
            echo $e->getMessage();
        }
    }
```
The struts configuration file

```xml
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation/DTD Struts Configuration 1.1//EN" "http://jakarta.apache.org/struts/DTDs/struts-config_1_1.dtd">
<struts-config>
  <!-- Data Source Configuration -->
  <data-sources />

  <!-- Form Bean Definitions -->
  <form-beans>
    <form-bean name="LoginForm" type="form.loginForm">
      <form-property name="pass" type="java.lang.String" />
      <form-property name="name" type="java.lang.String" />
    </form-bean>

    <form-bean name="SearchForm" type="Form.SearchForm">
      <form-property name="ingredients" type="java.lang.String" />
      <form-property name="ingredient1" type="java.lang.String" />
      <form-property name="ingredient2" type="java.lang.String" />
      <form-property name="ingredients" type="java.lang.String" />
      <form-property name="preparation" type="java.lang.Integer" />
      <form-property name="cooking" type="java.lang.Integer" />
      <form-property name="facility" type="java.lang.String" />
    </form-bean>

    <form-bean name="RegisterForm" type="form.RegisterForm">
      <form-property name="username" type="java.lang.String" />
      <form-property name="password" type="java.lang.String" />
      <form-property name="email" type="java.lang.String" />
      <form-property name="firstname" type="java.lang.String" />
      <form-property name="lastname" type="java.lang.String" />
    </form-bean>

    <form-bean name="DynaListForm" type="org.apache.struts.action.DynaActionForm">
      <form-property name="list" type="java.util.List" />
    </form-bean>
  </form-beans>

  <!-- Global Exception Definitions -->
  <global-exceptions />

  <!-- Global Forward Definitions -->
  <global-forwards />

  <!-- Action Mapping Definitions -->
  <action-mappings>
    <action path="/login" input="/login.jsp" name="LoginForm" type="action.LoginAction">
      <forward name="log_failed" path="/login.jsp" />
      <forward name="log_ok" path="/index.jsp" />
    </action>

    <action path="/disconnect" type="action.DisconnectAction">
      <forward name="disconnect" path="/index.jsp" />
    </action>

    <action path="/save" type="action.SaveAction">
    </action>

    <action path="/delete" type="action.DeletePrefAction">
      <forward name="delete" path="/save.jsp" />
    </action>

    <action path="/searchCat" type="action.SearchCatAction">
      <forward name="search" path="/results.jsp" />
    </action>

    <action path="/search" input="/search.jsp" name="SearchForm" type="action.SearchCatAction">
      <forward name="search" path="/resultsCriteria.jsp" />
    </action>

    <action path="/signup" input="/login.jsp" name="RegisterForm" type="action.RegisterAction">
      <forward name="signup_failed" path="/login.jsp" />
      <forward name="signup_ok" path="/index.jsp" />
    </action>
  </action-mappings>

  <!-- Controller Configuration -->
  <controller />

  <!-- Message Resources Definitions -->
  <message-resources />

  <!-- Plug Ins Configuration -->
  <plug-in class="Hibermate.HibermatePlugin"/>
</struts-config>
```
Example of ASP.NET C# controller

```csharp
protected void Page_Load(object sender, EventArgs e)
{
    // control for "already registered"
    string login = Request.Form["login"];
    string pass = Request.Form["password"];
    if (login != null && pass != null)
    {
        if (Customer.isExist(login, pass))
        {
            Session["login"] = login;
            Server.Transfer("index.aspx", true);
        }
        else
        {
            error.Text = "Error Login - Try Again";
        }
    }

    // control for "new customer"
    string firstname = Request.Form["firstname"];
    string lastname = Request.Form["lastname"];
    string username = Request.Form["username"];
    string email = Request.Form["email"];
    string password = Request.Form["password1"];
    if (firstname != null && lastname != null && username != null && email != null
        && password != null && firstname != "" && lastname != "" && username != "" && email != "" && password != "")
    {
        if (!Customer.isExist(username, password))
        {
            Customer.addCustomer(firstname, lastname, username, email, password);
            if (Customer.isExist(username, password))
            {
                Session["login"] = username;
                Server.Transfer("index.aspx", true);
            }
            else
            {
                error1.Text = "Registering error - Try again";
            }
        }
        else
        {
            error1.Text = "Registering error - Username/Password already used";
        }
    }
    signup.Attributes.Add("onclick", "javascript:addOnClick()"窗外光景 mildness 后退前退 增加
LogFile class

ASP.NET C#

```csharp
public class LogFile
{
    public static string FILE_LOG = "../logFile.txt";

    private static LogFile instance;
    private FileStream f;

    private LogFile()
    {
        if (!File.Exists(FILE_LOG))
        { File.Create(FILE_LOG);
        }

        // Use the singleton pattern to create only one log file
        public static LogFile getInstance()
        {
            if (instance == null)
            {
                instance = new LogFile();
                return instance;
            }
        }

        public FileStream getLogFile()
        {
            return f;
        }

        public void writeLog(string line) {
            try
            {
                StreamWriter monStreamWriter = new StreamWriter(FILE_LOG);
                monStreamWriter.WriteLine(line);
                monStreamWriter.Close();
            }
            catch (Exception ex)
            {
            }
        }
    }
```
**PHP**

```php
class LogFile {
    private $file;
    private static $instance = NULL;

    private function __construct() {
        $this->file = fopen ("logFile.txt", "a");
    }

    public function getInstance() {
        if (NULL == self::$instance) {
            self::$instance = new LogFile();
        }
        return self::$instance;
    }

    public function writeLog($line) {
        $line = $line." \\
";
        fputs ($this->file, $line);
        fclose ($this->file);
    }
}
```

**Java/J2EE**

```java
public class LogFile {

    // Path to the log text file
    public static final String FILE_LOG = ".../logFile.txt";

    private static LogFile instance;
    private static File f;

    private LogFile(File f) {
        f = new File(FILE_LOG);
    }

    // Use the singleton pattern to create only one log file
    public static LogFile getInstance() {
        if (instance == null)
            instance = new LogFile(f);
        return instance;
    }

    public File getLogFile() {
        return f;
    }

    /**
     * This method is used to add a line to the log file
     * @param line line to add to the report file
     * @throws FileNotFoundException
     */
    public void writeLog(String line) throws FileNotFoundException {
        FileWriter fw;
        try {
            fw = new FileWriter(FILE_LOG, true);
            fw.write(line + "\\n");
            fw.close();
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```
/**
 * WEB SERVICE
 * @param ingredients list of ingredients
 * @return
 * @throws IOException
 */

public static List<Recipe> getOtherRecipes(String ingredients) throws IOException {
    List<Recipe> recipes = new ArrayList<Recipe>();

    try{
        String myurl = "http://www.recipepuppy.com/api/?i="+ingredients;
        URL url = new URL(myurl);
        URLConnection conn = url.openConnection();
        // check if error 500
        if (conn instanceof HttpURLConnection) {
            int statusCode = conn.getResponseCode();
            if (statusCode == 200) {
                Scanner scan = new Scanner(url.openStream());
                String str = new String();
                while (scan.hasNext()) str += scan.nextLine();
                scan.close();

                // decode the data
                JSONParser parser=new JSONParser();
                Object obj = parser.parse(str);
                JSONObject obj2 = (JSONObject) obj;

                // treat the results and add them into a list
                JSONArray array = (JSONArray) obj2.get("results");
                Iterator i = array.iterator();
                Recipe recipe;
                while (i.hasNext()) {
                    JSONObject elements = (JSONObject) i.next();
                    recipe = new Recipe((String) elements.get("title"), (String) elements.get("href"),
                                        (String) elements.get("ingredients"), (String) elements.get("thumbnail"));
                    recipes.add(recipe);
                }
            }
        }
        catch (ParseException pe) {
            System.out.println("Position: "+ pe.getPosition());
            System.out.println(pe);
        }
        catch (MalformedURLException mue) {
            System.out.println("MalformedURLException happened.");
            mue.printStackTrace();
        }
        catch (IOException e) {
        }
    }

    return recipes;
ASP.NET C#

```csharp
// WEB SERVICE
protected List<Recipe> getOthersRecipes(String ingredients)
{
    // create the web client
    WebClient client = new WebClient();
    // go to the URL
    string Json = client.DownloadString("http://www.recipepuppy.com/api/?i=" + ingredients);

    // decode the data
    JavaScriptSerializer js = new JavaScriptSerializer();
    js.RegisterConverters(new[] { new DynamicJsonConverter() });
    dynamic obj = js.Deserialize(Json, typeof(object));

    // treat the results and put them into a list
    List<Recipe> recipes = new List<Recipe>();
    foreach (var recipe in obj["results"])
    {
        recipes.Add(new Recipe { title = recipe.title, href = recipe.href });
    }

    return recipes;
}
```

PHP

```php
// WEB SERVICE
function file_contents($path) {
    $str = @file_get_contents($path);
    if ($str === FALSE) {
        throw new Exception("Cannot access 'spath' to read contents.");
    } else {
        return $str;
    }
}

public function getOtherRecipes($ingredients) {
    $myurl = "http://www.recipepuppy.com/api/?i='.\$ingredients; $recipes = array();
    try {
        // go to the URL
        $json = file_contents($myurl);

        // decode the data
        $obj = json_decode($json);

        // treat the results and add them to an array
        foreach($obj->results as $o) {
            $recipes[] = array($o->title, $o->href);
        }
    } catch (Exception $e) {
    }
    return $recipes;
```
Testing

**PHP: TestCustomer.php**

```php
<?php
if (!defined('PHPUnit_MAIN_METHOD')) {
    define('PHPUnit_MAIN_METHOD', 'TestCustomer::main');
}

require_once 'Model/Customer.php';

class TestCustomer extends PHPUnit_Framework_TestCase {
    var $fname = 'Anne';
    var $lname = 'Jouli';
    var $email = 'anne.jouli@gmail.com';
    var $pwd = 'anne1234';
    var $login = 'annejouli';

    public function testCustomerAdded() {
        $this->assertEquals(Customer::existCustomer($this->login, $this->pwd), false);
        Customer::addCustomer($this->fname, $this->lname, $this->login, $this->email, $this->pwd);
        $this->assertEquals(Customer::existCustomer($this->login, $this->pwd), true);
        Customer::deleteCustomer($this->fname, $this->lname, $this->login, $this->email, $this->pwd);
        $this->assertEquals(Customer::existCustomer($this->login, $this->pwd), false);
    }
}
?>
```

**J2EE: TestCustomer.java**

```java
public class TestCustomer {
    Customer c = new Customer();

    @Before
    public void init() {
        c.setName("Anne");
        c.setSurname("Jouli");
        c.setLogin("annejouli");
        c.setEmail("anne.jouli@gmail.com");
        c.setPassword("anne1234");
    }

    @Test
    public void testCustomerAdded() {
        assertEquals(Customer.isExist(c.getLogin(), c.getPassword()), false);
        Customer.addCustomer(c.getFirstName(), c.getLastName(), c.getLogin(), c.getEmail(), c.getPassword());
        assertEquals(Customer.isExist(c.getLogin(), c.getPassword()), true);
    }

    @After
    public void reinit() {
        Customer.deleteCustomer(c.getLogin(), c.getEmail(), c.getPassword());
    }
}
```
using System;

class TestCustomer
{
    [TestClass]
    public class TestCustomer
    {
        private String fname = "Anne";
        private String lname = "Jouli";
        private String email = "anne.jouli@gmail.com";
        private String pwd = "annel234";
        private String login = "annejouli";

        [TestMethod]
        public void TestCustomerAdded()
        {
            Assert.AreEqual(Customer.isExist(login, pwd), false);
            Customer.addCustomer(fname, lname, login, email, pwd);
            Assert.AreEqual(Customer.isExist(login, pwd), true);
            Customer.deleteCustomer(fname, lname, login, email, pwd);
            Assert.AreEqual(Customer.isExist(login, pwd), false);
        }
    }
}
## Project Plan

<table>
<thead>
<tr>
<th>Task Name</th>
<th>Time</th>
<th>Deliverable</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Requirements Engine</td>
<td>2 days</td>
<td>27/05/14</td>
</tr>
<tr>
<td>Database design</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developing phase</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Alternative 1</strong></td>
<td>13 days</td>
<td>13/06/14</td>
</tr>
<tr>
<td>Define and install the technical environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define frameworks needed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define and implement the architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement the design (CSS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understand how apply design patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop main functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test all functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliver a prototype of the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write notes for the dissertation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Alternative 2</strong></td>
<td>15 days</td>
<td>04/07/14</td>
</tr>
<tr>
<td>Define and install the technical environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define and implement the architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement the design (CSS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understand how apply design patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop main functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test all functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliver a prototype of the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Start the comparison of the two alternatives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write notes for the dissertation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Alternative 3</strong></td>
<td>20 days</td>
<td>01/08/14</td>
</tr>
<tr>
<td>Learn the language</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define and install the technical environment</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Define and implement the architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Implement the design (CSS)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Understand how apply design patterns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Develop main functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Test all functionalities</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deliver a prototype of the application</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Write notes for the dissertation</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Write the dissertation</strong></td>
<td>14 days</td>
<td>21/08/14</td>
</tr>
<tr>
<td>Resume my notes and compare the three alternatives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand in a draft</td>
<td></td>
<td>08/08/14</td>
</tr>
<tr>
<td>Read and correct the dissertation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand in a new version of the draft</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finalize the dissertation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hand in</td>
<td></td>
<td>21/08/14</td>
</tr>
</tbody>
</table>