ENTWA Project Ideas

*Report: Helping MSc Students find suitable topics of study*

Table of Contents

Introduction 1

Design 1

Homepage Search 1

Register page 1

Overall flow 2

Implementation and testing 2

Netbeans, Derby and JSF 2

Internationalisation 2

Summary 3

References 3

# Introduction

The School of computing currently offers an MSc project ideas database, but it implemented in a non-MVC style that does not serve users particularly well. The aim of this project is to create a superior version of the current implementation.

The superior versions main objective is to provide a better experience by offering a search and a simple way to add new ideas. The design will be very simple with clear calls to action to make the user experience positive.

# Design

## Homepage Search

The homepage contains a simple search text box that allows users to easily search through the ideas. As this functionality is a key feature, I decided to give it a prominent position on the page. This makes finding ideas fairly straightforward.

[Image here]

## Register page

The page users can obtain an account via is super simple. It has been designed to only request the minimal information required to receive an account. For example, the user selects an organisation as their user type from the select box they are presented with more fields that are related to organisations.

[Pics]

## Overall flow

The main site is split into two main parts, the ideas and the people. The artefact was designed to be as simple to use as possible, so as a result it’s very stripped down to its basic functions. The main two tasks as a user needs to perform are either to submit an idea or apply for an idea. As the diagram below shows, a users flow through the site requires minimal effort in their part.

[flow diagram]

# Implementation and testing

Describe your experiences implementing the artefact. Here you should briefly discuss your choice of development tools, highlight areas of development that were particularly tricky, and explain how you overcame problems. You should also outline how you tested the components that make up your artefact.

## Netbeans, Derby and JSF

The main development tools were Netbeans as the IDE, Derby (Java DB) as the database engine and JSF and the Java EE framework.

Initially the artefact was developed using a MySQL database, but soon it was discovered that this could prove an unreliable solution, as it’s not included by default with Netbeans. Instead Derby (Java DB) was used as it comes with Netbeans. Derby works in a similar way to MySQL, but has slightly fewer features and treats variable types differently. For example, its integers are not suitable as an auto increment primary key as they do not count very high.

NetBeans has a wizard to create a skeleton application from the database using JSF. This feature allowed for a rapid development of the application so more focus could be placed on the complex business logic, such as creating Derby compatible search functions and login services.

One of the downsides of using JSF 2.0 was it’s inability to make use of HTML5 input elements and attributes out of the box.

## Internationalisation

Bundle!

* Show how the database works
  + Two tables, idea & people.
  + The database is simple with a handful of foreign keys to avoid a complex application.
* Describe the design of the model:
  + Controller (Business Logic & loading output in a view to put it into) > Facade (Beans) / Entity > (Mapping – Not implemented) > Data Source (Derby)  
    set up.
* Using internationalisation via bundle
  + Friggin cool.
* Using MVC & JSF
  + Auto generated the initial application structure from the database using tools within netbeans, on the assumption it followed best practise.
    - Link to tutorial
  + JSF 2.0 unable to take advantage of HTML5.
    - Found work around, but implemtation is tricky
* Netbeans requiring restart constantly?
* Working with Derby – Hard!
  + Originally connected to MySQL database, which was more flexible.
  + Figured out you can generate database from the entities, made it easier.
    - Though, you’re querying the entity not the database
* I didn’t take advantage of unit testing within Java EE instead I loaded up a page and ran it with expected input.
* Persistence is still proving tricky. A user needs to add the connection into the connection pool for it to work.

# Summary

Summarise what you have undertaken in the coursework. In particular highlight the good and bad design and technology decisions that you made. Conclude by briefly discussing an alternative approach that could have potentially been taken with the benefit of hindsight.

* Should have used derby from the start or used an external MySQL server.
* JSF was ok, but after researching the industry other frameworks are more agile.

# References