

Introduction to Play! framework

Jonathan Pastor

September 3, 2013

1 Introduction

The objective of this course is to introduce the basic features of a modern web framework, through the study of a java compatible Model-View-Controller ¹ (MVC) web framework: Play! ².

During this course, the student will learn how to develop a small web application. This application will use a database for storing objects, and users will interact with it through using web forms.

Students will be asked to work in pairs: each pair will develop a dynamic website: the specifications are given in the section 3. The specifications are a set of constraints: Each constraint that is satisfied gives points, These constraints will be evaluated by a jury during the oral defence of the project.

2 The Subject of the project

The pairs are free to find a subject that will drive their project. Students are free to use this example of a website that sells sandwiches:

A sandwich shop asks Students of Ecole des Mines de Nantes to develop an online shop. Customers will be able to order sandwich via a front-end website. When an order is validated by a customer, it is processed by an administrator that will prepare the order. The administrator can create or modify sandwiches.

Project subject for year 2012-2013.

A customer will create orders. Orders belong to one customer. An order is a set of association between a sandwich and a number. A sandwich is composed by a set of ingredients.

Example of data structure for year 2012-2013.

¹cf. <http://en.wikipedia.org/wiki/Model-view-controller>

²cf. <http://www.playframework.com/documentation/1.2.5/home>

3 The Specifications

Constraint	Description	Points
Controller-1	Develop a controller for the frontend	1 pts
Controller-2	Develop a controller for the backend	1 pts
Controller-3	At least one controller asks data with a custom SQL query	1 pts
View-1	Display data (a list) with an HTML table	1 pts
View-2	Display data (a list) with an HTML list	1 pts
View-3	Display data (an element) with details	1 pts
View-4	Create a transition between 2 views	1 pts
Database-1	Database contains at least one complex entity (that contains a set of other entities, cf appendix A)	3 pts
Database-1	The database is in third normal form ³	1 pts
CRUD	At least one complex data entity have create, read, update and delete actions. These actions are located in the backend.	3 pts
AJAX-1	At least one controller have a method that produce JSON or XML	1 pts
AJAX-2	At least one view gets data dynamically with an AJAX request	1 pts
Design-1	Use a CSS framework (Bootstrap ⁴ , Zurb Foundation ⁵ , PureCSS ⁶ , ...)	1 pts
Design-2	The application has a clean design	1 pts
Design-2	The application has a good ergonomoy	1 pts
Design-3	The application has been developed around prototyping methodology ⁷	1 pts
Total		20 pts
Bonus-1	Display statistics with a framework like D3.js ⁸	1 pts

4 Evaluation

During the oral defence of the project, each constraint will be evaluated. The jury will check if the students have a good comprehension of the technologies seen during this course. The sum of the points given by the satisfaction of the constraints will give an idea of the final mark.

³cf. http://en.wikipedia.org/wiki/Third_normal_form

⁴cf. <http://getbootstrap.com/>

⁵cf. <http://foundation.zurb.com/>

⁶cf. <http://purecss.io/>

⁷cf. http://en.wikipedia.org/wiki/Software_prototyping

⁸cf. <http://d3js.org/>

Appendices

A Complex Data Entity

Here is an example of complex data (entity Sandwich):

```
class Sandwich {  
    String name;  
    List<Ingredient> ingredients;  
}  
  
[...]  
  
class Ingredient {  
    String name  
}
```

In this example a sandwich contains a set of ingredients.