MVC and MVP patterns with Play! Framework and Backbone.js

Alberto García García
< *agg180@alu.ua.es* >

May 7, 2014

TABLE OF CONTENTS

INTRODUCTION
- Trends
- Challenges
- Addressing the challenges

PLAY! FRAMEWORK
- What is Play! Framework?
- RESTful Architecture
- Project layout
- The MVC application model
- Request/Response path

BACKBONE.JS
- What is Backbone.js?

INTRODUCTION

TRENDS

- Enterprises's needs lead the market.
- Offering services: SOA wins.
- The web changes the status quo.
- SOA is not web compliant.
- Exposing services through the web requires extra effort.
- The game changes: new possibilities and challenges.

CHALLENGES

- Real time data has to be pushed.
- Huge amounts of data.
- Need for scalability and integration.
- Easy integration and accessibility.
- Interoperability.

ADDRESSING THE CHALLENGES

- Embrace the internet.
  - HTTP Protocol
  - HTML5
  - XML/JSON
  - Javascript
  - CSS
- Paradigm shift: client-side.
- Simplicity.
- A framework to rule them all.

PLAY! FRAMEWORK

WHAT IS PLAY! FRAMEWORK?

- A web framework focused on:
  - Simplicity.
  - Productivity.
  - Scalability.
  - Designed for the modern web.
    - Concentrate on server-side.
    - Delegate AMAP to the client.
  - Embrace internet standards.
  - Java and Scala.
  - RESTful architecture web applications.
  - Model-View-Controller.

RESTful architecture

- Implemented using HTTP and REST principles.
- Representational state transfer (REST) principles:
  - Uniform interface.
  - Stateless.
  - Caching.
  - Layers.
  - Code on demand.
- Goals:
  - Performance.
  - Scalability.
  - Portability.
  - Reliability.
  - SIMPLICITY.

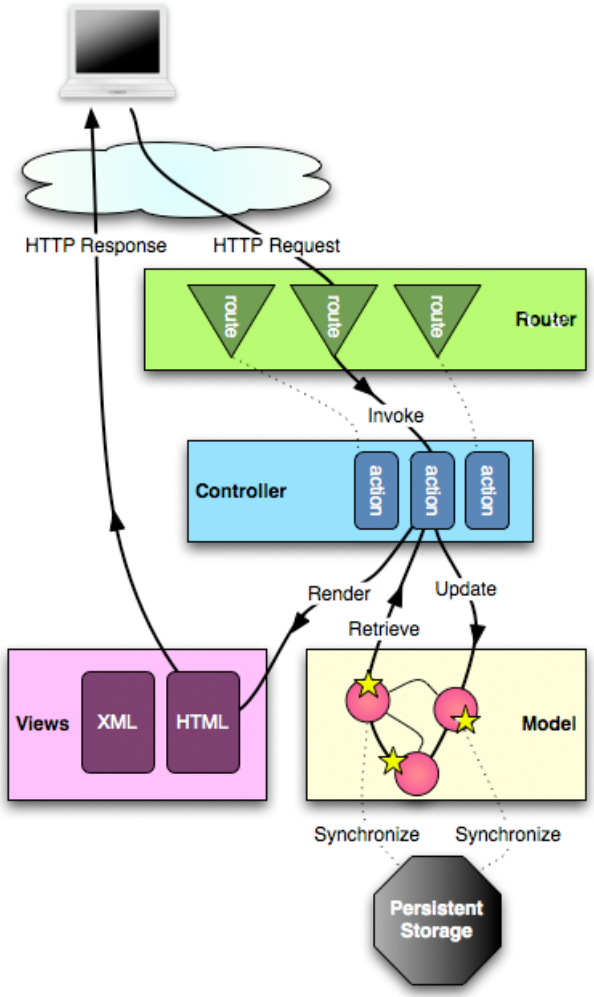
PROJECT LAYOUT

```
app                     → Application sources
 └ assets               → Compiled asset sources
    └ stylesheets       → Typically LESS CSS sources
    └ javascripts       → Typically CoffeeScript sources
 └ controllers          → Application controllers
 └ models               → Application business layer
 └ views                → Templates
build.sbt               → Application build script
conf                    → Configurations files and other non-compiled resour
 └ application.conf     → Main configuration file
 └ routes               → Routes definition
public                  → Public assets
 └ stylesheets          → CSS files
 └ javascripts          → Javascript files
 └ images               → Image files
project                 → sbt configuration files
 └ build.properties     → Marker for sbt project
 └ plugins.sbt          → sbt plugins including the declaration for Play its
lib                     → Unmanaged libraries dependencies
logs                    → Standard logs folder
 └ application.log      → Default log file
target                  → Generated stuff
 └ scala-2.10.0
    └ cache
    └ classes           → Compiled class files
    └ classes_managed   → Managed class files (templates, ...)
    └ resource_managed  → Managed resources (less, ...)
    └ src_managed       → Generated sources (templates, ...)
test                    → source folder for unit or functional tests
```

THE MVC APPLICATION MODEL

- Models in app/models
  - Java/Scala classes.
  - Data + Operations, mainly object-oriented.
  - Business logic and storage.
- Views in app/views
  - HTML/XML/JSON/Scala templates.
  - Directives as placeholders for data.
  - Render models to user interfaces.
- Controllers in app/controllers
  - Java/Scala classes.
  - Methods as actions, mainly procedural.
  - Receive requests, act (update models + render views) and response.

REQUEST/RESPONSE PATH



BACKBONE.JS

WHAT IS BACKBONE.JS?

REFERENCES

REFERENCES I


Nicolas Leroux and Sietse de Kaper, *Play for java*, Manning Publications, 2014.


Erik Bakker Peter Hilton and Francisco Canedo, *Play for scala*, Manning Publications, 2014.


Alexander Reelsen, *Play framework cookbook*, Packtpub Publications, 2014.