Web development with the Play! framework

Java web development is fun again

Alexander Reelsen

alexander@reelsen.net

June 2010

Agenda

1. Introduction
   - Overview
2. Core concepts
   - Architecture
   - Conventions
   - Configuring routes
   - Model
   - Controller
   - View
   - Testing
   - Jobs
   - Running Play in production
3. Ongoing development
   - Features for Play 1.1
   - Module repository
   - Development forecast

Agenda

Speaking...

About me

- Studied information systems
- 10 years linux system engineering, converted to software engineering
- Just another java developer
- Web framework enthusiast
- Fed up with complex java environment for simple webapps

Non development life

- Basketball/Streetball
- Web 2.0

Speaking...

About me

- Studied information systems
- 10 years linux system engineering, converted to software engineering
- Just another java developer
- Web framework enthusiast
- Fed up with complex java environment for simple webapps

Non development life

- Basketball/Streetball
- Web 2.0

The web taking over

Ubiquitous web

- Web as ubiquitous platform
- Providing data, not web pages
- Browser as the operating system
- JavaScript gains steam, toolkits as well as engines
- Web `applications` are alive

Agenda

Play in 10 seconds

Quick overview

- No compile, deploy, restart cycle - Fix the bug and hit reload!
- Share nothing system - no state in the server
- Clean templating system - using groovy
- Exact errors (including line numbers, even for templates)
- Lots of builtin features for fast development
- Pure Java
- Starts fast, runs fast
- Extensible by modules

Roots and fundamentals

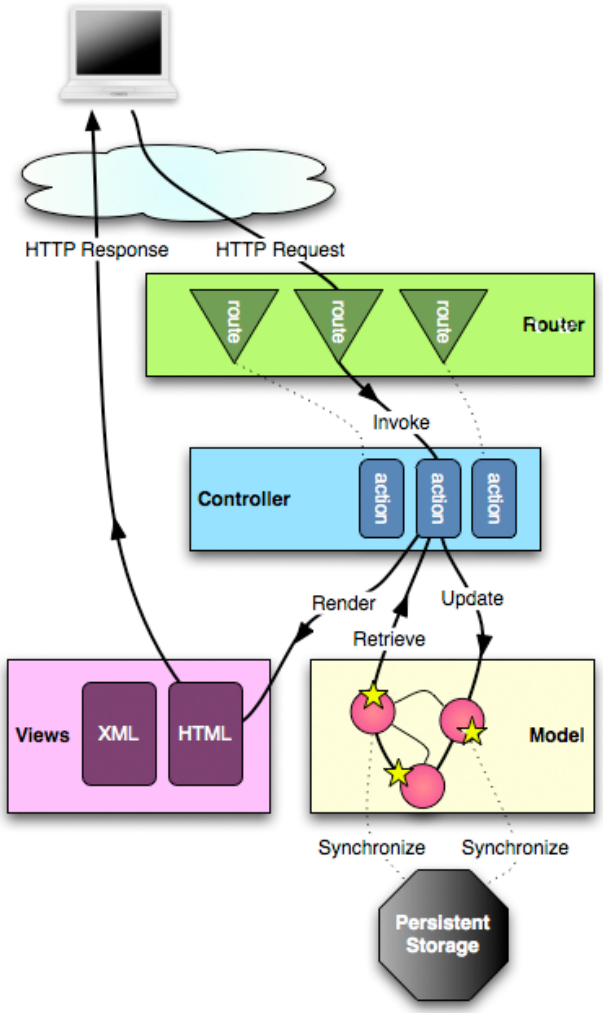
History

- Exists since 2008, by Guillaume Bort from zenexity
- Release 1.0 was in October 2009
- Current: 1.0.3 + development tree

A framework imposes its own style of architecture

- REST as architectural paradigm for ressources
- URLs are the entry point (and implicit interface) to your application
- Do not work against HTTP (stateless protocol)
- Convention over configuration
- Only fractions of differences between development and production mode

Play architecture



Play is a lot of glue code

Includes the following libraries

- Hibernate
- Oval
- Lucene
- Google gson
- Eclipse compiler
- Commons: fileupload, httpclient, email, logging, beanutils
- MINA and Asyncweb
- EhCache
- JAMon
- Groovy

Play specialties

Interesting to know

- No support for servlet API (yes, in a web framework)
- Sharing objects via memcached through several nodes
- Everything is UTF-8
- Full text indexing with 2 annotations
- No anemic domain model - logic is in the object
- DAOs and finders are not external
- Textmate, eclipse bundles, also support for IDEA and netbeans

Application layout

Creating a new app

- `play new myapp`

Application structure

```
./conf
./conf/routes
./conf/application.conf
./conf/messages
./test
./lib
./public
./app
./app/models
./app/controllers
./app/views
```

Application layout

Creating a new app

- `play new myapp`

Application structure

```
./conf
./conf/routes
./conf/application.conf
./conf/messages
./test
./lib
./public
./app
./app/models
./app/controllers
./app/views
```

Application config file

conf/application.conf

- Configure database access
  - db=fs, db=mem
  - `db=mysql:user:pwd@database_name`
  - Any JDBC connection
- Specify modules
- Supported languages
- Logger
- memcached setup
- mail configuration
- mode/system specific settings

The conf/routes file

Interface contract to the outer world

```
GET     /                           Application.index
GET     /user/{username}            Application.showUser
POST    /user                       Application.createUser
DELETE  /user/{username}            Application.deleteUser

GET     /public                     staticDir:public
```

Designing a domain model

A simple user

```
@Entity
public class User extends Model {
        @Required
        @Column(unique = true)
        public String username;

        @Required
        @Email
        public String email;

        @Required
        public String password;

        public void setPassword(String password) {
                this.pass = Crypto.passwordHash(pass);
        }
}
```

The Model class is a helper

Finders and Entity actions

```
// Query by property
User user = User.find("byUsername", username).first();

// Calls a setter
user.password = "foobar";
user.save();

List<User> users = User.findAll();
users.get(0).delete();

// JPA queries are possible as well, so are joins
List<String> names = User.find("select␣u.username␣from␣User␣u
␣order␣by␣u.username␣desc").fetch();
```

Calling business logic

User controller

```
public class Application extends Controller {
        public static void index() {
                List<User> users = User.findAll();
                render(users);
        }

        public static void showUser(String username) {
                User user = User.find("byUsername", username).first();
                notFoundIfNull(user);
                render(user);
        }

...
```

Calling business logic

User controller (ctd.)

```
...

        public static void deleteUser(String username) {
                User user = User.find("byUsername", username).first();
                notFoundIfNull(user);
                user.delete();
                Application.index();
        }

        public static void createUser(@Valid User user) {
                if (validation.hasErrors()) {
                        flash.error("Invalid␣user␣data");
                        Application.index();
                }

                user = user.save();
                Application.showUser(user.username)
        }
}
```

Calling business logic

Example: Accessing the session

```
public class AuthController extends Controller {

        @Before(unless = "login")
        public static void checkSession() {
                if (!request.session.contains("username")) {
                        forbidden("You␣are␣not␣authorized");
                }
        }


        public void login(String username, String password) {
                String pass = Crypto.passwordHash(password);
                User user = User.find("byUsernameAndPassword",
                        username, pass).first();
                notFoundIfNull(user);
                request.session.put("username", user);
                Application.index();
        }
}
```

The templating system

List users (app/views/Application/index.html)

```
#{extends 'main.html' /}
#{set title:'Index' /}

<ul>
#{list items:users, as:'user'}
<li>
  #{a @Application.showUser(user.username)}
      ${user.username}
  #{/a}
  with email address ${user.email}
</li>
#{/list}
</ul>
```

The templating system

Add user

```
#{form @Application.createUser()}
<div>Username:
<input type="text" name="user.username" />
</div>

<div>Password:
<input type="pass" name="user.password" />
</div>

<div>Email:
<input type="text" name="user.email" />
</div>

<input type="submit" value="Add␣user" />
#{/form}
```

The templating system

More tags

- `doLayout, extends, include`
- `if, ifnot, else, elseif`
- &{'i18nVariable'} out of `conf/messages.de`
- Always access to: `session, flash, request, params, lang, messages, out, play`

The templating system

Extending objects using mixins

```
public class SqrtExtension extends JavaExtensions {

  public static Double sqrt(Number number) {
     return Math.sqrt(number.doubleValue());
  }

}
```

The template code

```
<div>
Square root of x value is: ${x.sqrt()}
</div>
```

Testing

Providing test data

YAML formatted file provides testdata

```
User(aleree):
    username : alr
    password : test
    email : alexander@reelsen.net
```

Loading test data...

```
@Before
public void setUp() {
    Fixtures.deleteAll();
    Fixtures.load("data.yml");
}
```

Testing

Unit tests

- Standard junit tests
- Extend from UnitTest, which needs a JPA environment

Functional tests

- Integration tests
- Checks the external responses (http response)

Selenium tests

- GUI tests
- Very nice controllable, playback recorder
- Possibility of doing step-by-step slow debugging

Testing

CI with Calimoucho

- Poor mans hudson
- Checks out the project and runs `play auto-test`, which needs a graphical layer for selenium tests
- Check it under `http://integration.playframework.org`

Code coverage with cobertura

- Enable the cobertura module in `application.conf`
- Run the tests, check the results

Jobs - being asynchronous

Doing the right thing at the right time

- Scheduled jobs (housekeeping)
- Bootstrap jobs (initial data providing)
- Suspendable requests (rendering a PDF report without blocking the connection thread pool)

```
/* @Every("1h") */
@OnApplicationStart
public class LoadDataJob extends Job {

        public void doJob() {
                /* .. do whatever you want */
        }
}
```

Putting play into production

The setup

- A redirector like nginx or apache is preferred
- Also eliminates the need to serve static files
- Redirect to different nodes would be the main task
- Profile per nodes possible (very useful for server farms)

Monitoring play application

Partial Output of `play status`

```
 Monitors:

Application.showLatestRecipesRss(), ms. ->
4120 hits; 41.0 avg; 20.0 min; 260.0 max;
/app/views/Application/showLatestRecipesRss.html, ms. ->
4120 hits; 34.9 avg; 19.0 min; 235.0 max;

Datasource:
Job execution pool:
Scheduled jobs:
```

Agenda

Play 1.1 is on its way

Core features

- Bug fixes
- Scala support
- More modules through module repository

Useful modules

Slowly but steadily growing

- Scala, Scalate, Akka
- PDF, excel modules
- Guice and Spring modules
- Netty and grizzly support
- GWT support, GAE support
- Extended CSS, SASS
- Ivy and Maven support
- Siena, Ebean ORM, MongoDB
- Database migration module
- Hosting: stax, playapps

TODO

Open issues

- NoSQL support (Siena, MongoDB)
- Amazon Cloud Integration
- Hosting platform (playapps.net has just launched)
- Lucene Solr Support for shared environments
- Tighter integration with JavaScript Toolkits like Dojo
- Far more modules - check out the rich grails ecosystem
- The first book... is coming!

Done!

Thanks for listening.
Questions?

Job offers regarding Java Scalability or new technologies like Play and/or MongoDB?

Talk to me or send mail to alexander@reelsen.net

URLs

Further information

- This presentation: http://www.slideshare.net/areelsen/
- http://www.playframework.org
- http://twitter.com/playframework
- http://it-republik.de/jaxenter/artikel/Webanwendungen-mit-dem-Java-Framework-Play!-2813.html
- http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2009-11.pdf
- http://www.bookware.de/kaffeeklatsch/archiv/KaffeeKlatsch-2009-12.pdf
- http://www.zenexity.fr/frameworks/anatomie-dun-framework-de-developpement-web/