Agile Software Development

Produced by

Eamonn de Leastar (edeleastar@wit.ie)

Department of Computing, Maths & Physics
Waterford Institute of Technology

http://www.wit.ie

http://elearning.wit.ie

Waterford Institute *of* Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

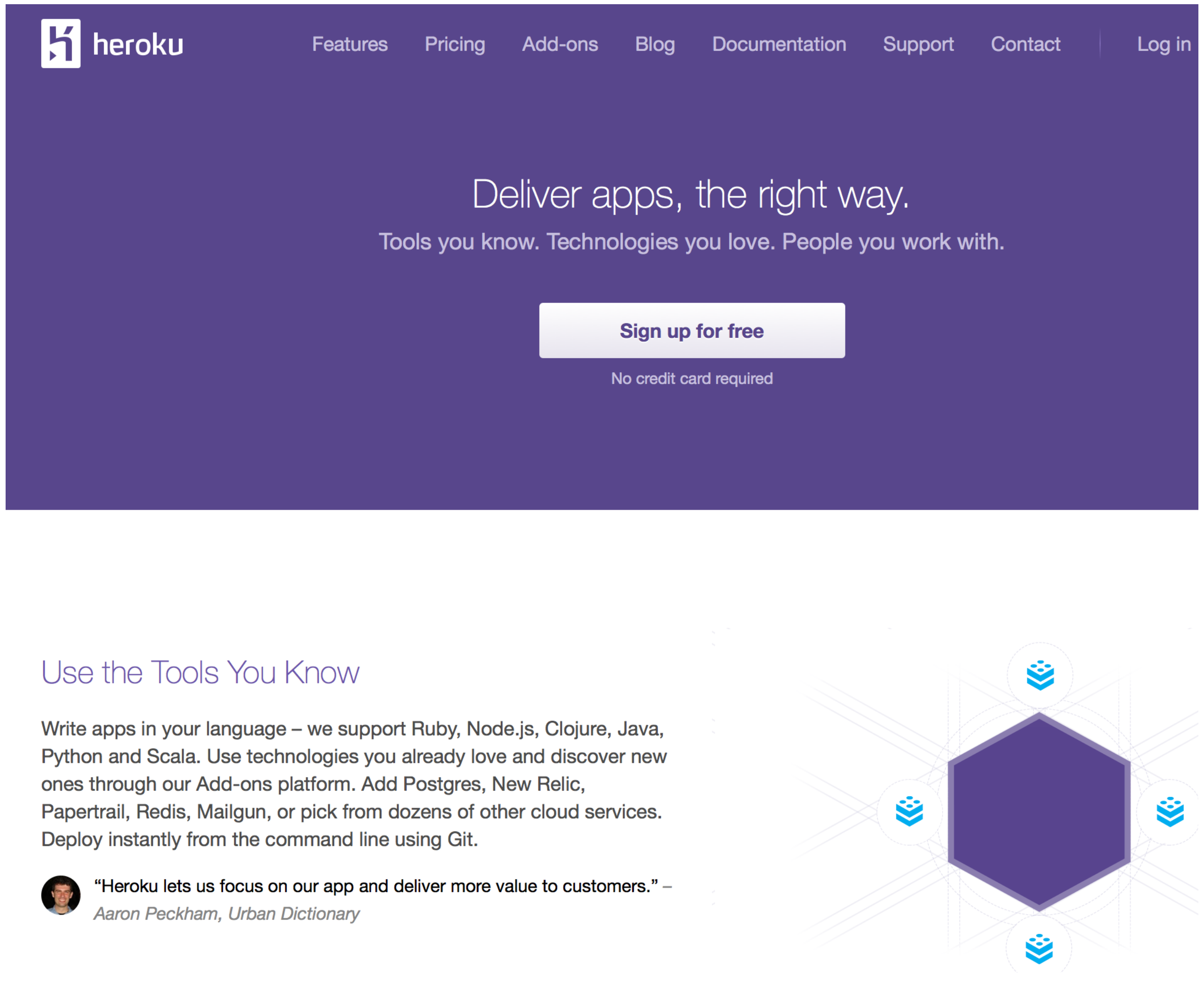
eLearning support unit

Pacemaker Cloud

Scope

- Refactor the pacemaker application as a cloud hosted service exposing a REST API
  - Use the Play Framework to provide sufficient (but not too much) abstraction layers
  - Use the Heroku cloud hosting service to deploy the application
  - Attempt to keep the much of the model and service implementations from the console version intact.
  - Keep the app ‘Reactive’

The High Velocity Web Framework For Java and Scala



Deliver apps, the right way.

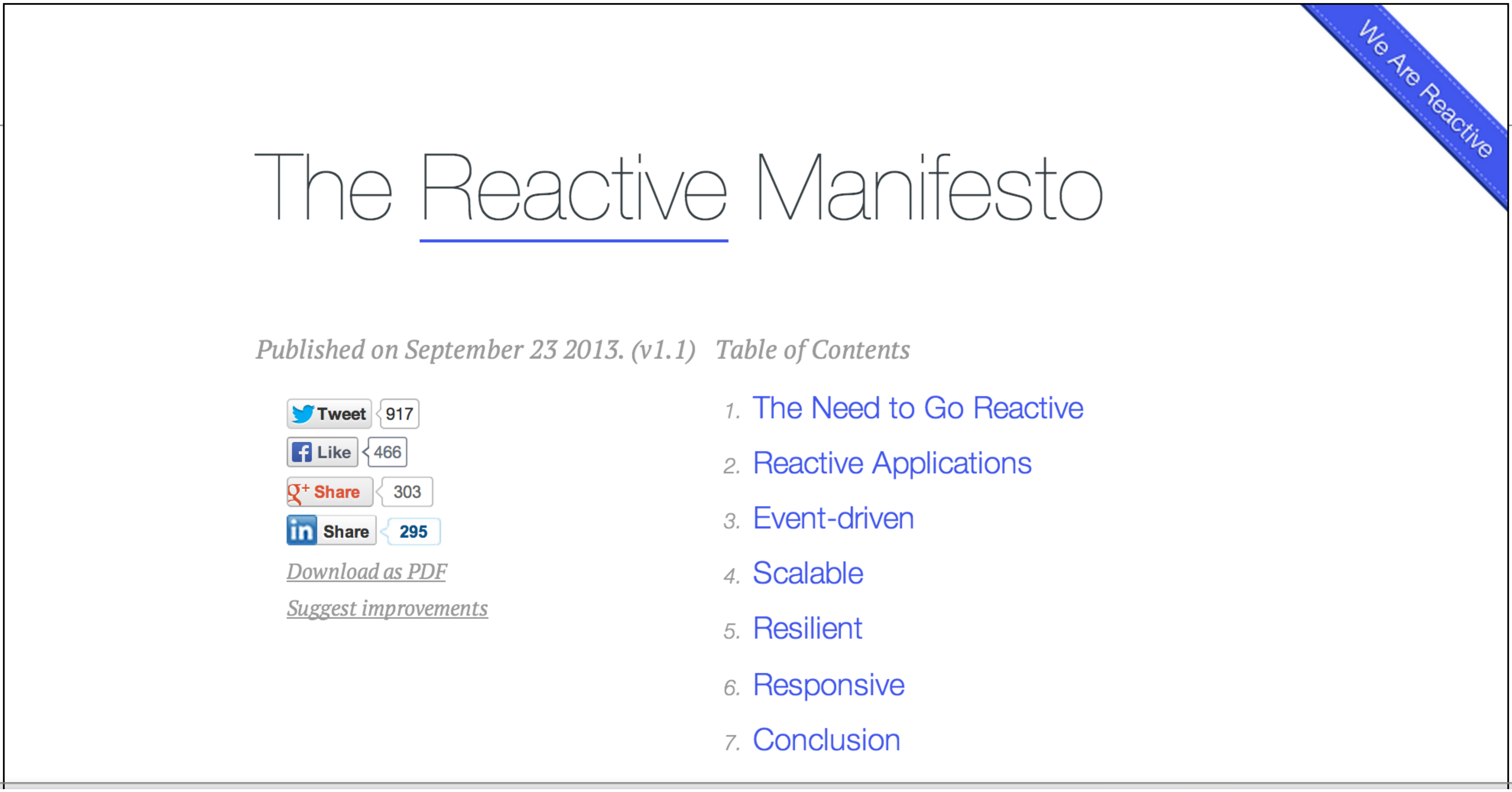
Tools you know. Technologies you love. People you work with.

Use the Tools You Know

Write apps in your language – we support Ruby, Node.js, Clojure, Java, Python and Scala. Use technologies you already love and discover new ones through our Add-ons platform. Add Postgres, New Relic, Papertrail, Redis, Mailgun, or pick from dozens of other cloud services. Deploy instantly from the command line using Git.

“Heroku lets us focus on our app and deliver more value to customers.” – *Aaron Peckham, Urban Dictionary*

Reactive Application


We Are Reactive

The Reactive Manifesto

Published on September 23 2013. (v1.1)

Tweet 917
Like 466
Share 303
Share 295

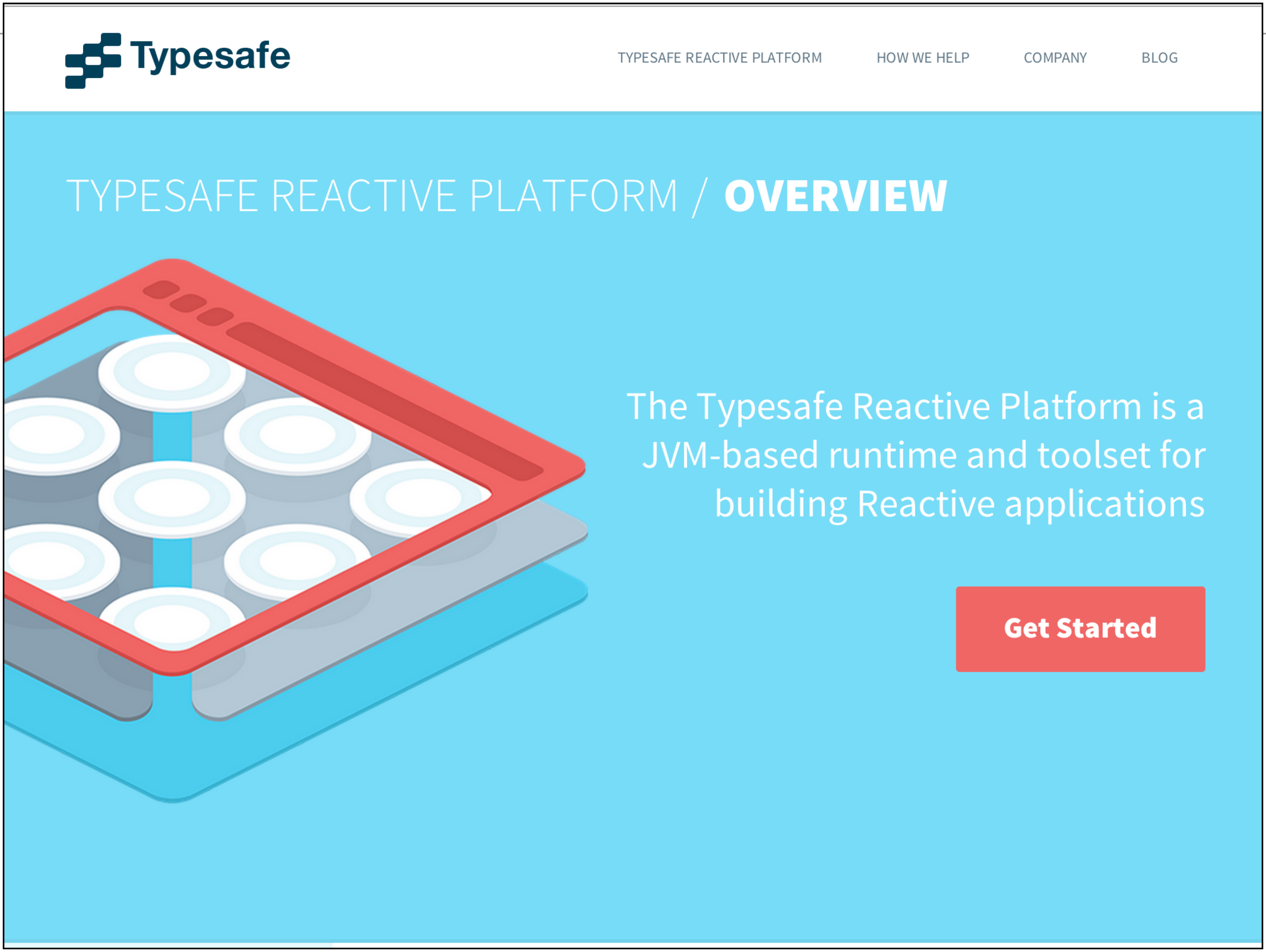
Download as PDF

Suggest improvements

Table of Contents

react to events	react to load	react to failure	react to users
the event-driven nature enables the following qualities	focus on scalability by avoiding contention on shared resources	build resilient systems with the ability to recover at all levels	honor response time guarantees regardless of load

Typesafe Stack



Typesafe

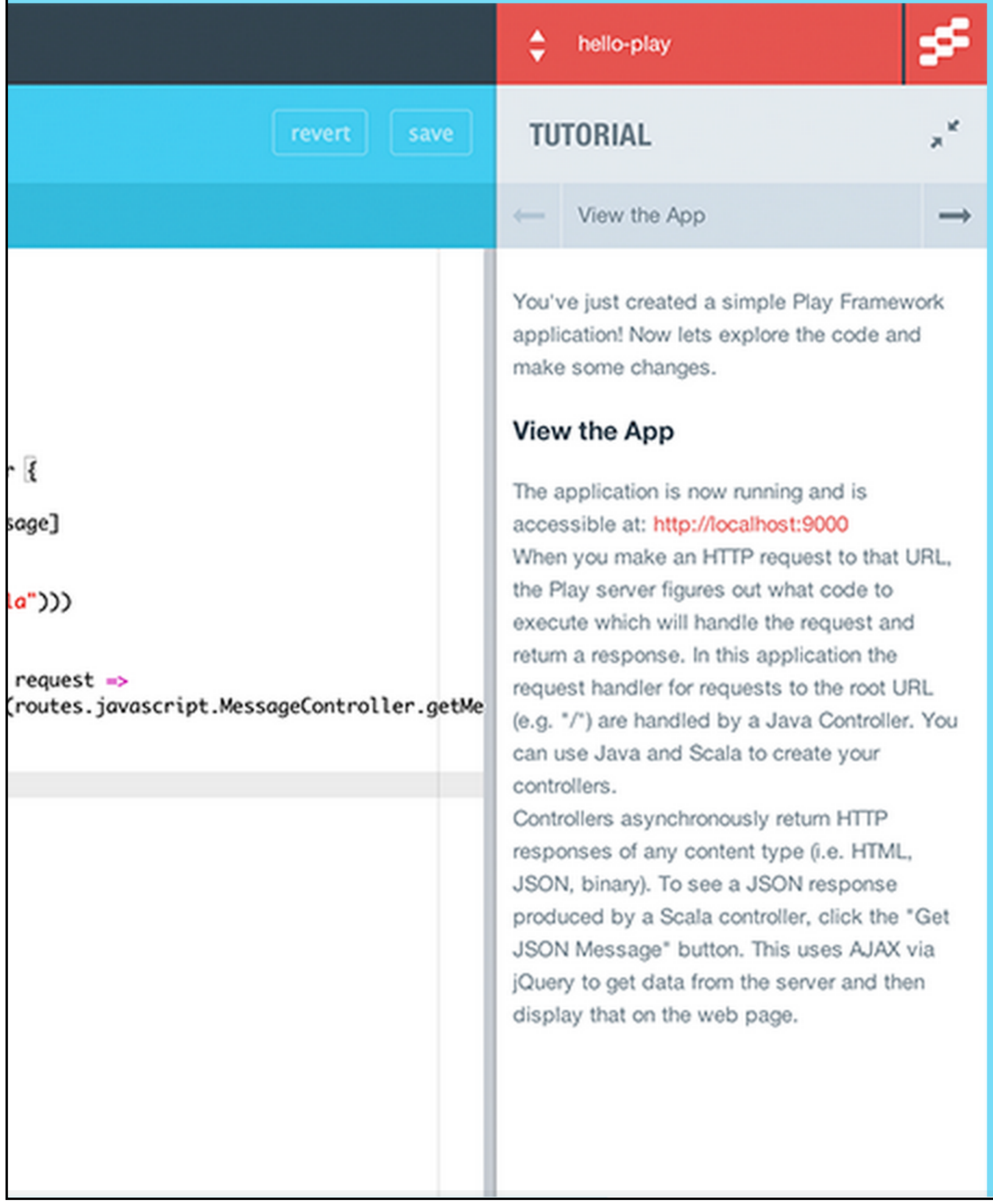
TYPESAFE REACTIVE PLATFORM HOW WE HELP COMPANY BLOG

TYPESAFE REACTIVE PLATFORM / **OVERVIEW**

The Typesafe Reactive Platform is a JVM-based runtime and toolset for building Reactive applications

Get Started

TYPESAFE ACTIVATOR / **OVERVIEW**



```
r {
sage]
la")))
 request =>
(routes.javascript.MessageController.getMe
```

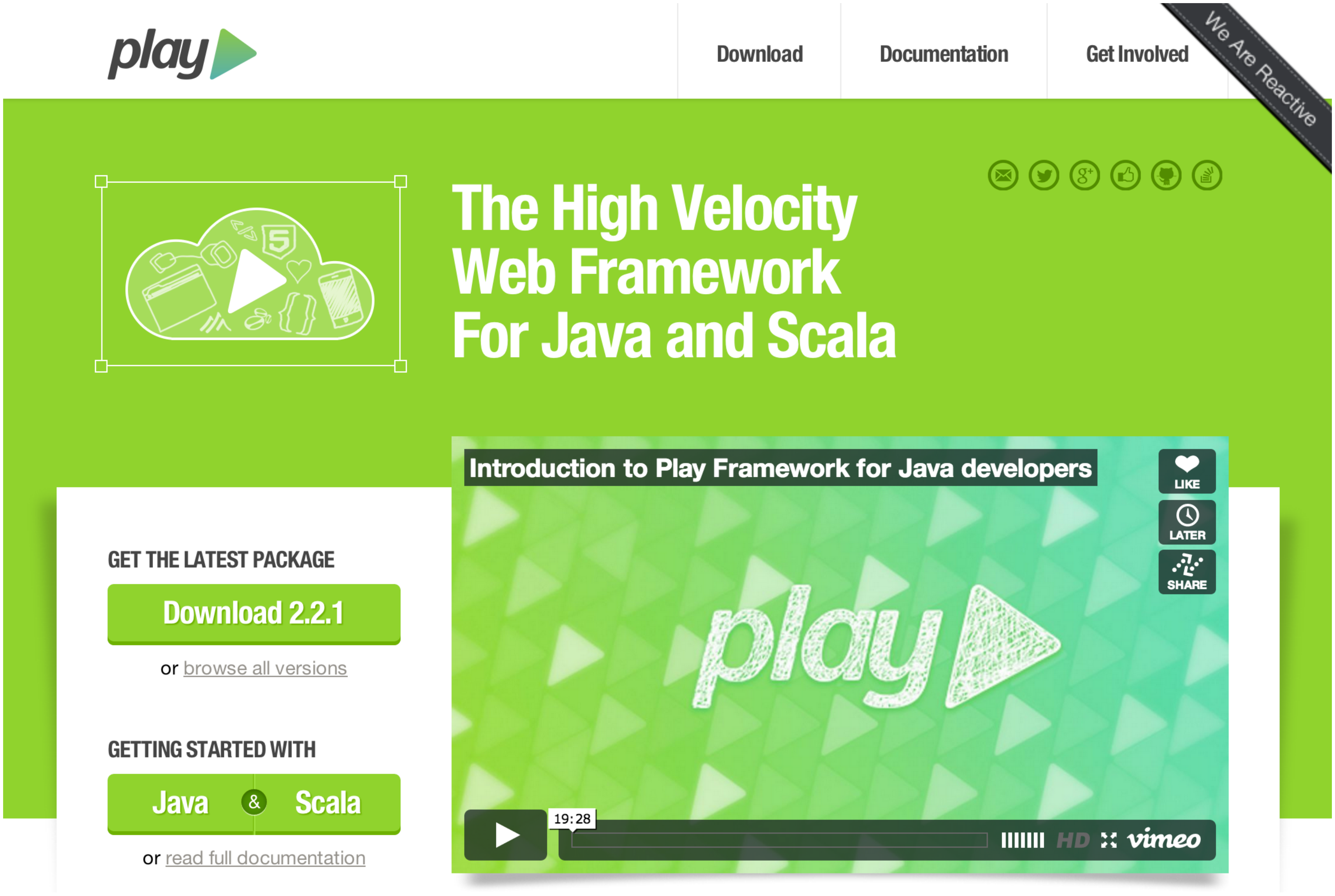
You've just created a simple Play Framework application! Now lets explore the code and make some changes.

View the App

The application is now running and is accessible at: http://localhost:9000
When you make an HTTP request to that URL, the Play server figures out what code to execute which will handle the request and return a response. In this application the request handler for requests to the root URL (e.g. "/") are handled by a Java Controller. You can use Java and Scala to create your controllers.
Controllers asynchronously return HTTP responses of any content type (i.e. HTML, JSON, binary). To see a JSON response produced by a Scala controller, click the "Get JSON Message" button. This uses AJAX via jQuery to get data from the server and then display that on the web page.

Typesafe Activator gets you started with the Typesafe Reactive Platform, Play Framework, Akka and Scala

Get Started



play

Download

Documentation

Get Involved

We Are Reactive

The High Velocity Web Framework For Java and Scala

GET THE LATEST PACKAGE

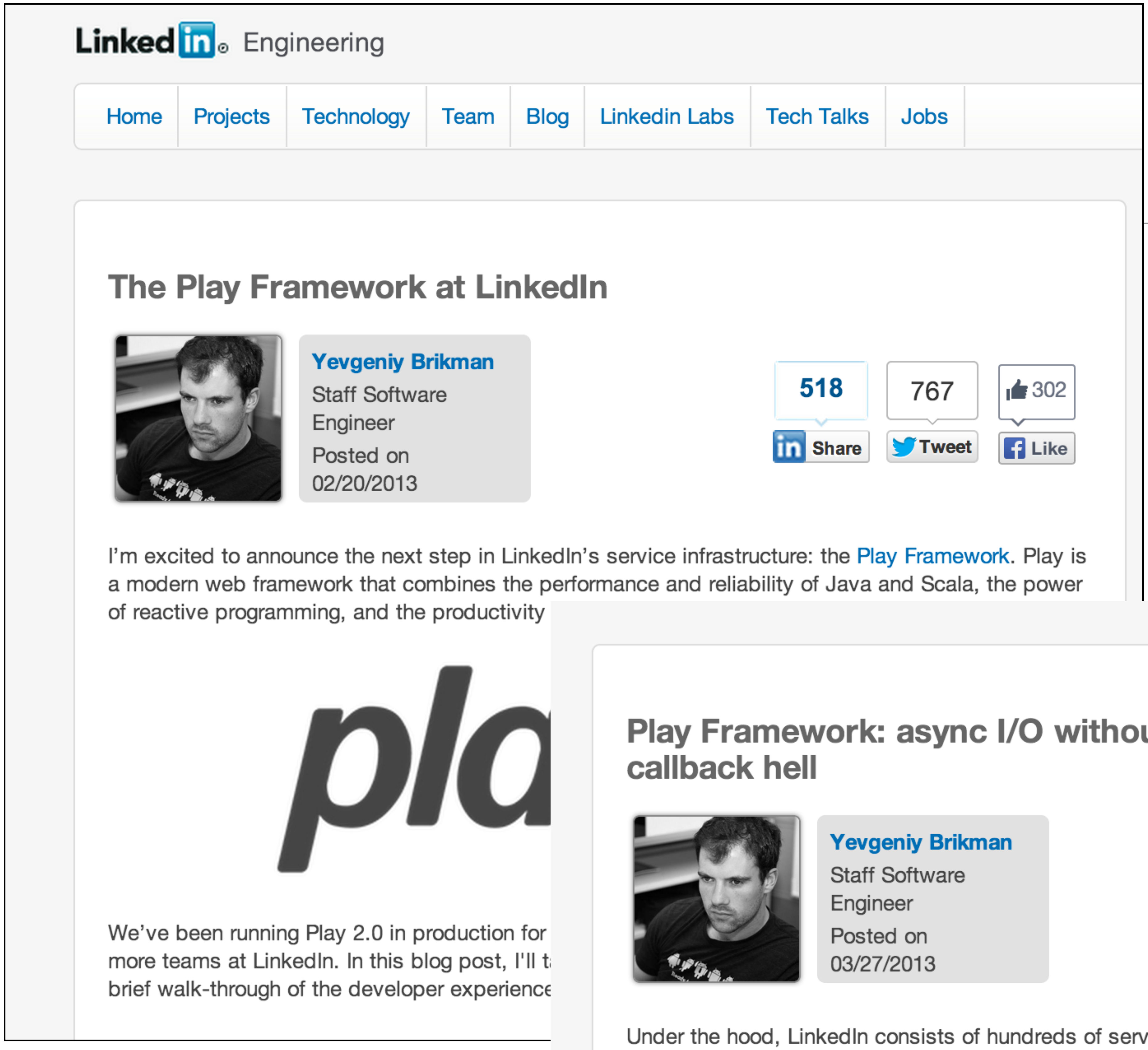
Download 2.2.1

or browse all versions

GETTING STARTED WITH

Java & Scala

or read full documentation



LinkedIn Engineering

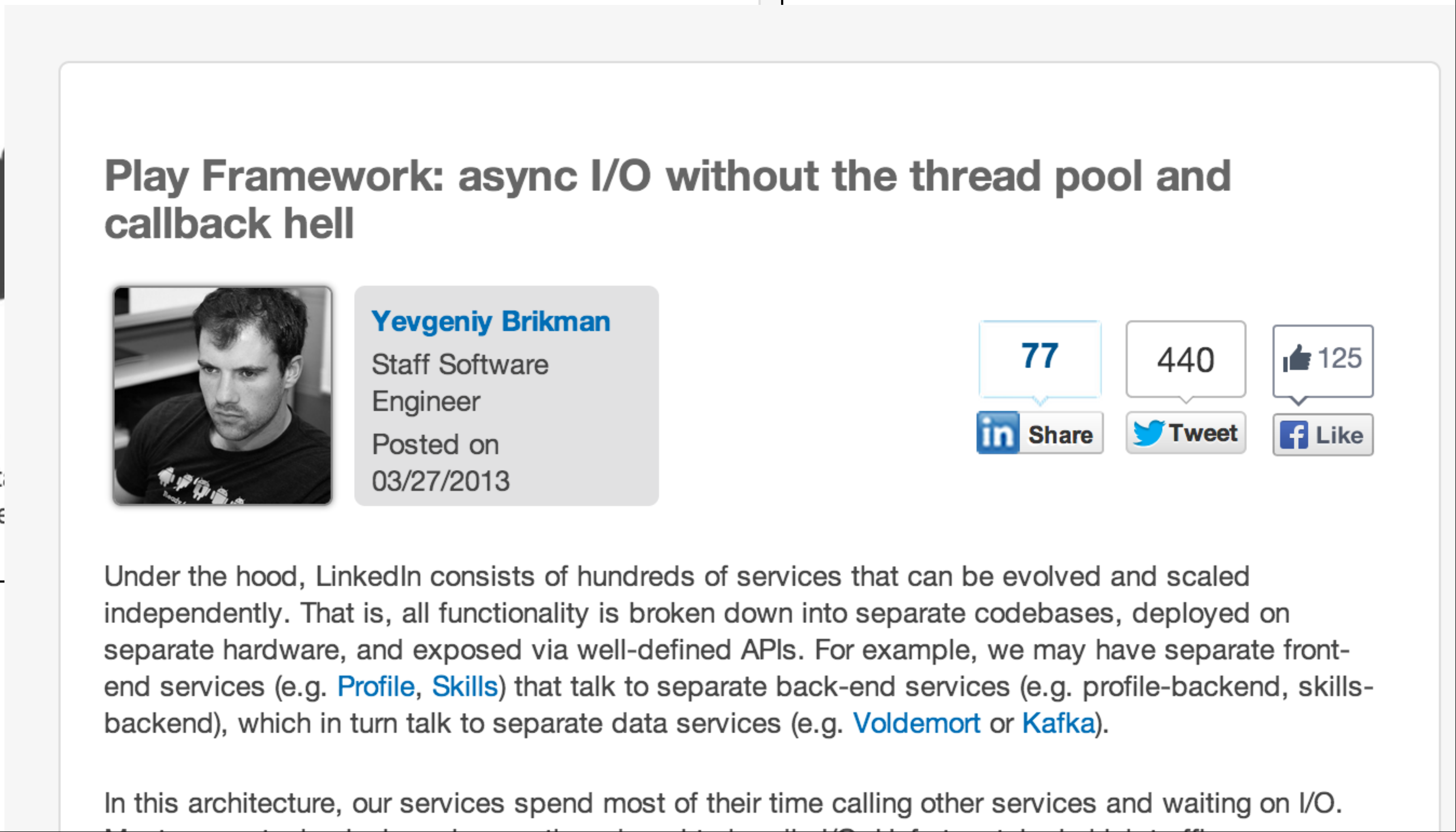
Home | Projects | Technology | Team | Blog | Linkedin Labs | Tech Talks | Jobs

The Play Framework at LinkedIn

Yevgeniy Brikman
Staff Software Engineer
Posted on 02/20/2013

I'm excited to announce the next step in LinkedIn's service infrastructure: the Play Framework. Play is a modern web framework that combines the performance and reliability of Java and Scala, the power of reactive programming, and the productivity

We've been running Play 2.0 in production for
more teams at LinkedIn. In this blog post, I'll t
brief walk-through of the developer experience



Play Framework: async I/O without the thread pool and callback hell

Yevgeniy Brikman
Staff Software Engineer
Posted on 03/27/2013

Under the hood, LinkedIn consists of hundreds of services that can be evolved and scaled independently. That is, all functionality is broken down into separate codebases, deployed on separate hardware, and exposed via well-defined APIs. For example, we may have separate front-end services (e.g. Profile, Skills) that talk to separate back-end services (e.g. profile-backend, skills-backend), which in turn talk to separate data services (e.g. Voldemort or Kafka).

In this architecture, our services spend most of their time calling other services and waiting on I/O.

Lab 08 - Pacemaker 2 (Play)

- Install Play
- User Model
- Parsers
- Controllers
- Routes
- Testing
- Deploy to Heroku
- Database on Heroku
- Database Evolutions

Install Play (1)

- Download and install the latest version of the Play Framework (currently 2.2.1)

http://www.playframework.com

- This will involve simply unzipping the archive, and placing the unzipped folder on the path.

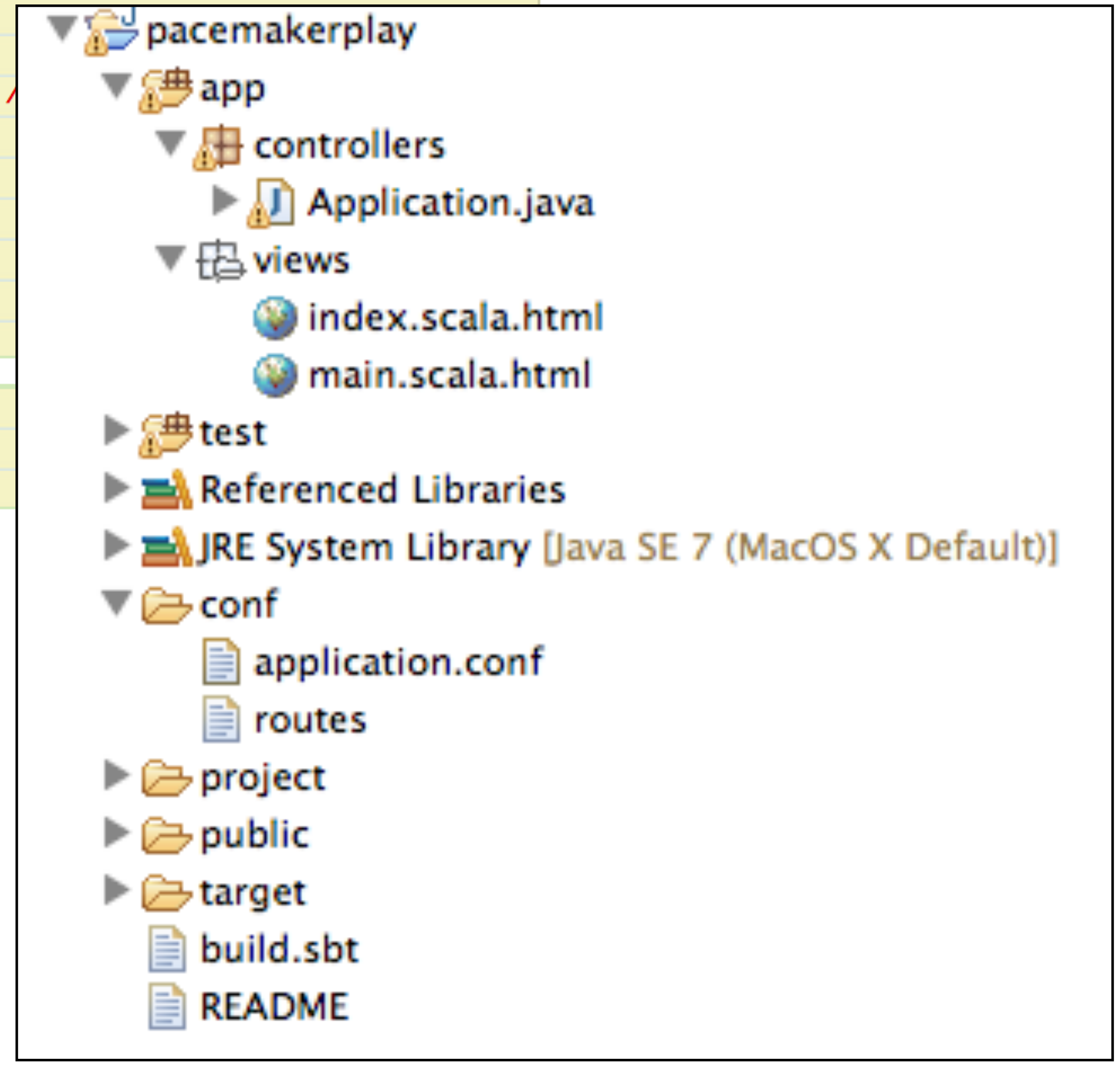
```
play new pacemakerplay
```

```
       _
 _ __ | | __ _ _  _
| '_ \| |/ _' | || |
|  __/|_|\____|\__ /
|_|            |__/

play 2.2.1 built with Scala 2.10.2 (running Java 1.7.0_40), http://www.playframework.com

The new application will be created in /Users/edeleastar/repos/modules/agile/pacemaker/pace
maker-1.0/pacemakerplay

What is the application name? [pacemakerplay]
>

Which template do you want to use for this new application?

  1             - Create a simple Scala application
  2             - Create a simple Java application

> 2
OK, application pacemakerplay is created.

Have fun!
```

Install Play (2)

```
...
       _
 _ __ | | __ _ _  _
| '_ \| |/ _' | || |
|  __/|_|\____|\__ /
|_|            |__/

play 2.2.1 built with Scala 2.10.2 (running Java 1.7.0_40), http:/

> Type "help play" or "license" for more information.
> Type "exit" or use Ctrl+D to leave this console.

[pacemakerplay] $
```

```
eclipse
```



```
▼ pacemakerplay
    ▼ app
        ▼ controllers
            ▶ Application.java
        ▼ views
              index.scala.html
              main.scala.html
    ▶ test
    ▶ Referenced Libraries
    ▶ JRE System Library [Java SE 7 (MacOS X Default)]
    ▼ conf
          application.conf
          routes
    ▶ project
    ▶ public
    ▶ target
      build.sbt
      README
```

Install Play (3)

In the play console, enter

```
run
```

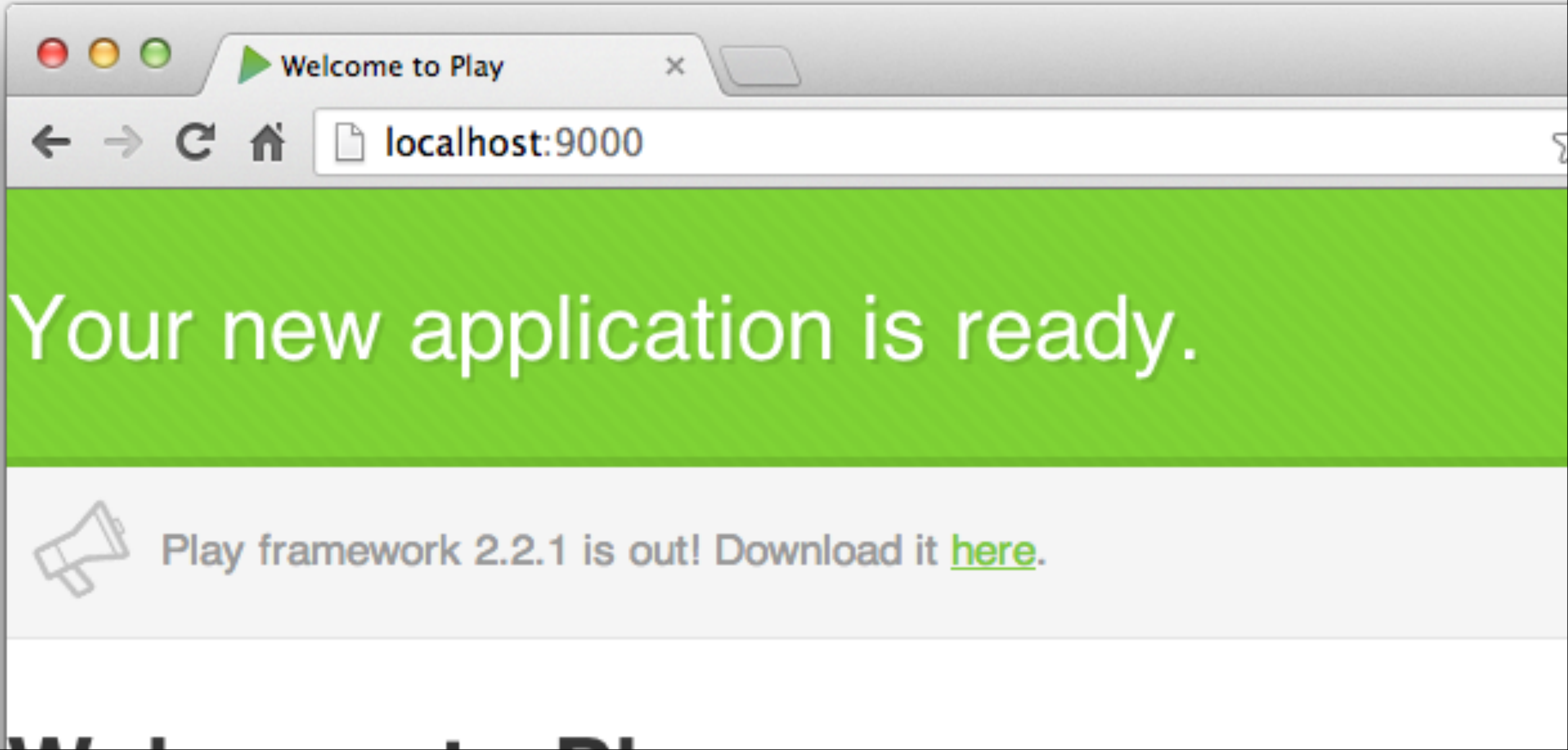
which should display:

```
--- (Running the application from SBT, auto-reloading is enabled) ---

[info] play - Listening for HTTP on /0:0:0:0:0:0:0:0:9000

(Server started, use Ctrl+D to stop and go back to the console...)
```

Browse to :

- http://localhost:9000

It should display a standard greeting page.



Your new application is ready.

Play framework 2.2.1 is out! Download it here.

Pacemaker 1 User model

(removed activity for the moment)

```
public class User
{
  static Long   counter = 0l;

  public Long   id;
  public String firstName;
  public String lastName;
  public String email;
  public String password;

  public User()
  {
  }

  public User(String firstName, String lastName, String email, String password)
  {
    this.id        = counter++;
    this.firstName = firstName;
    this.lastName  = lastName;
    this.email     = email;
    this.password  = password;
  }
 }
 // equals, toString, hashCode
}
```

Pacemaker 2 User Model

- Uses JPA annotations to manage
  - DB Table generation
  - ID management
  - Relationships to other Models (not included yet)

```
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;

  public User()
  {
  }

  public User(String firstname, String lastname, String email, String password)
  {
    this.firstname = firstname;
    this.lastname  = lastname;
    this.email     = email;
    this.password  = password;
  }
//  same equals, toString, hashCode
}
```

Pacemaker 2 User Model

- Equip User class with simple database search and management methods
- All ‘static’ metods

```
public class User extends Model
{
  //…
  public static User findByEmail(String email)
  {
    return  User.find.where().eq("email", email).findUnique();
  }

  public static User findById(Long id)
  {
    return find.where().eq("id", id).findUnique();
  }

  public static List<User> findAll()
  {
    return find.all();
  }

  public static void deleteAll()
  {
    for (User user: User.findAll())
    {
      user.delete();
    }
  }

  public static Model.Finder<String, User> find
    = new Model.Finder<String, User>(String.class, User.class);
}
```

Parsers

transform the model into various formats

- Carry over general approach from pacemaker 1

```
public class JsonParser
{
  private static JSONSerializer  userSerializer = new JSONSerializer();

  public static User renderUser(String json)
  {
    return new JSONDeserializer<User>().deserialize(json, User.class);
  }

  public static String renderUser(Object obj)
  {
    return userSerializer.serialize(obj);
  }
}
```

Specialise serialisation for JSON

Pacemaker 1 - PacemakerAPI

- Responsible for :
  - maintaining data structures
  - exposing core features to clients

```
public class PacemakerAPI
{
  private Map<Long,   User>    userIndex       = new HashMap<>();
  private Map<String, User>    emailIndex      = new HashMap<>();
  private Map<Long, Activity> activitiesIndex = new HashMap<>();

  private Serializer serializer;

  public PacemakerAPI(Serializer serializer)
  {
    this.serializer = serializer;
  }

  @SuppressWarnings("unchecked")
  public void load() throws Exception
  {
    serializer.read();
    activitiesIndex = (Map<Long, Activity>) serializer.pop();
    emailIndex      = (Map<String, User>)   serializer.pop();
    userIndex       = (Map<Long, User>)     serializer.pop();
  }

  public void store() throws Exception
  {
    serializer.push(userIndex
    serializer.push(emailInde
    serializer.push(activitie
    serializer.write();
  }

  public Collection<User> getU
  {
    return userIndex.values()
  }

  public  void deleteUsers()
  {
    userIndex.clear();
    emailIndex.clear();
  }

  public User createUser(String firstName, String lastName, String
email, String password)
  {
    User user = new User (firstName, lastName, email, password);
    userIndex.put(user.id, user);
    emailIndex.put(email, user);
    return user;
  }
```

Implement the core application features as represented by the Model.

Pacemaker 2 - PacemakerAPI

- Data structures now in Database, so responsibilities simplified
- Logic very similar to pacemaker 1

```
public class PacemakerAPI extends Controller
{
  public static Result users()
  {
    List<User> users = User.findAll();
    return ok(renderUser(users));
  }

  public static Result user(Long id)
  {
    User user = User.findById(id);
    return user==null? notFound() : ok(renderUser(user));
  }

  public static Result createUser()
  {
    User user = renderUser(request().body().asJson().toString());
    user.save();
    return ok(renderUser(user));
  }

  public static Result deleteUser(Long id)
  {
    Result result = notFound();
    User user = User.findById(id);
    if (user != null)
    {
      user.delete();
      result = ok();
    }
    return result;
  }

  public static Result deleteAllUsers()
  {
    User.deleteAll();
    return ok();
  }
  //…
}
```

```java
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;

  public User()
  {
  }

  public User(String firstname, String lastname,
              String email,     String password)
  {
    this.firstname = firstname;
    this.lastname  = lastname;
    this.email     = email;
    this.password  = password;
  }
//  same equals, toString, hashCode
}
```

```java
public class JsonParser
{
  private static JSONSerializer  userSerializer = new JSONSerializer();

  public static User renderUser(String json)
  {
    return new JSONDeserializer<User>().deserialize(json, User.class);
  }

  public static String renderUser(Object obj)
  {
    return userSerializer.serialize(obj);
  }
}
```

```java
public class PacemakerAPI extends Controller
{
  public static Result users()
  {
    List<User> users = User.findAll();
    return ok(renderUser(users));
  }

  public static Result user(Long id)
  {
    User user = User.findById(id);
    return user==null? notFound() : ok(renderUser(user));
  }

  public static Result createUser()
  {
    User user = renderUser(request().body().asJson().toString());
    user.save();
    return ok(renderUser(user));
  }

  public static Result deleteUser(Long id)
  {
    Result result = notFound();
    User user = User.findById(id);
    if (user != null)
    {
      user.delete();
      result = ok();
    }
    return result;
  }

  public static Result deleteAllUsers()
  {
    User.deleteAll();
    return ok();
  }



  //…
}
```

NO MORE CODE !

(for this version)

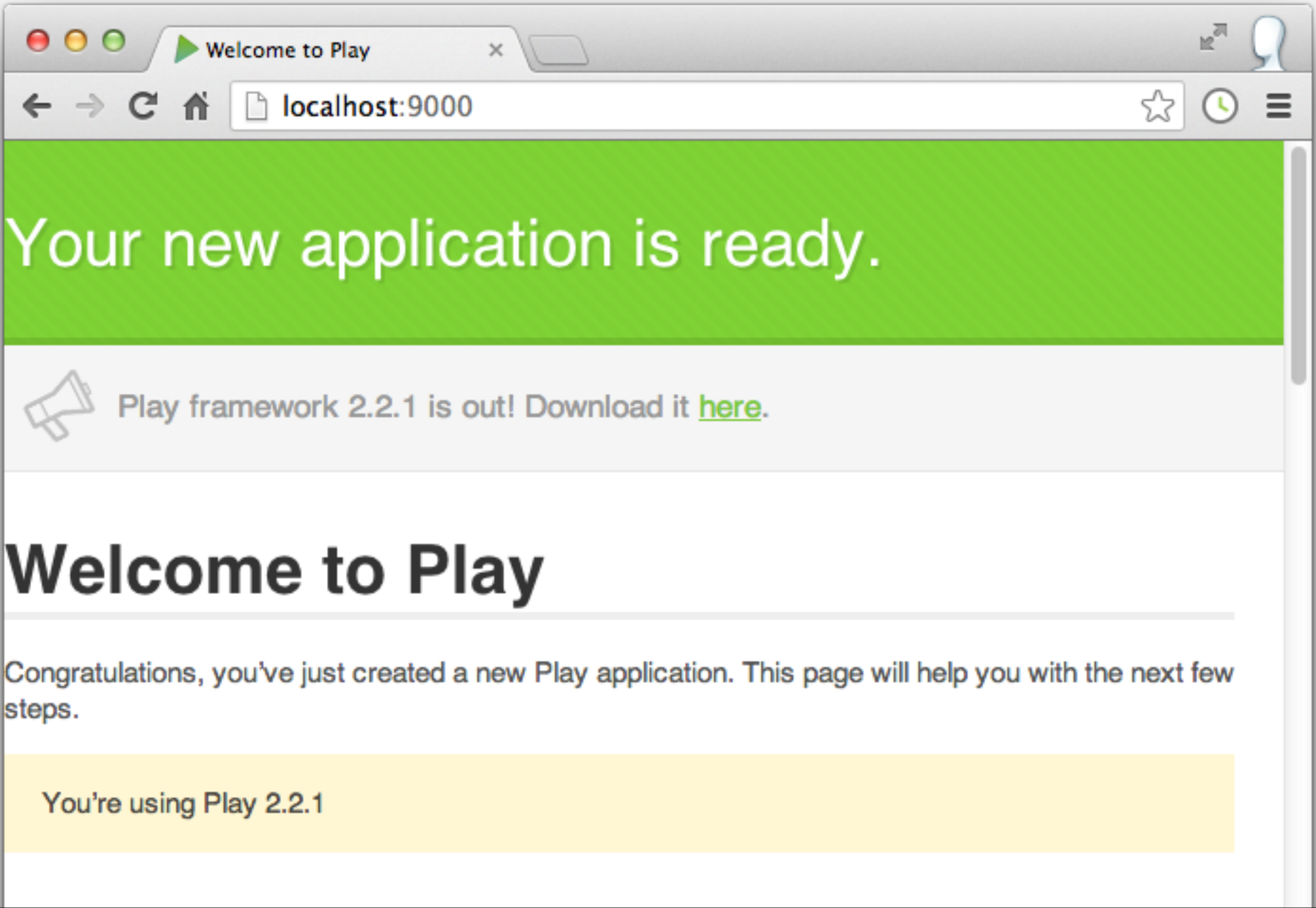
Routes

```
GET     /                                      controllers.Application.index()

GET     /api/users                             controllers.PacemakerAPI.users()
DELETE  /api/users                             controllers.PacemakerAPI.deleteAllUsers()
POST    /api/users                             controllers.PacemakerAPI.createUser()

GET    /api/users/:id                          controllers.PacemakerAPI.user(id: Long)
DELETE /api/users/:id                          controllers.PacemakerAPI.deleteUser(id: Long)
PUT    /api/users/:id                          controllers.PacemakerAPI.updateUser(id: Long)
```

- Defines HTTP routes that will be published by this app.
- Route matches http verb + url -> controller.method
- Any browser (or application that can ‘speak’ http) can access the application services through these routes.

```
GET      /                                     controllers.Application.index()
```

```
public class Application extends Controller
{
  public static Result index()
  {
    return ok(index.render("Your new application is ready."));
  }
}
```



Your new application is ready.

Play framework 2.2.1 is out! Download it here.

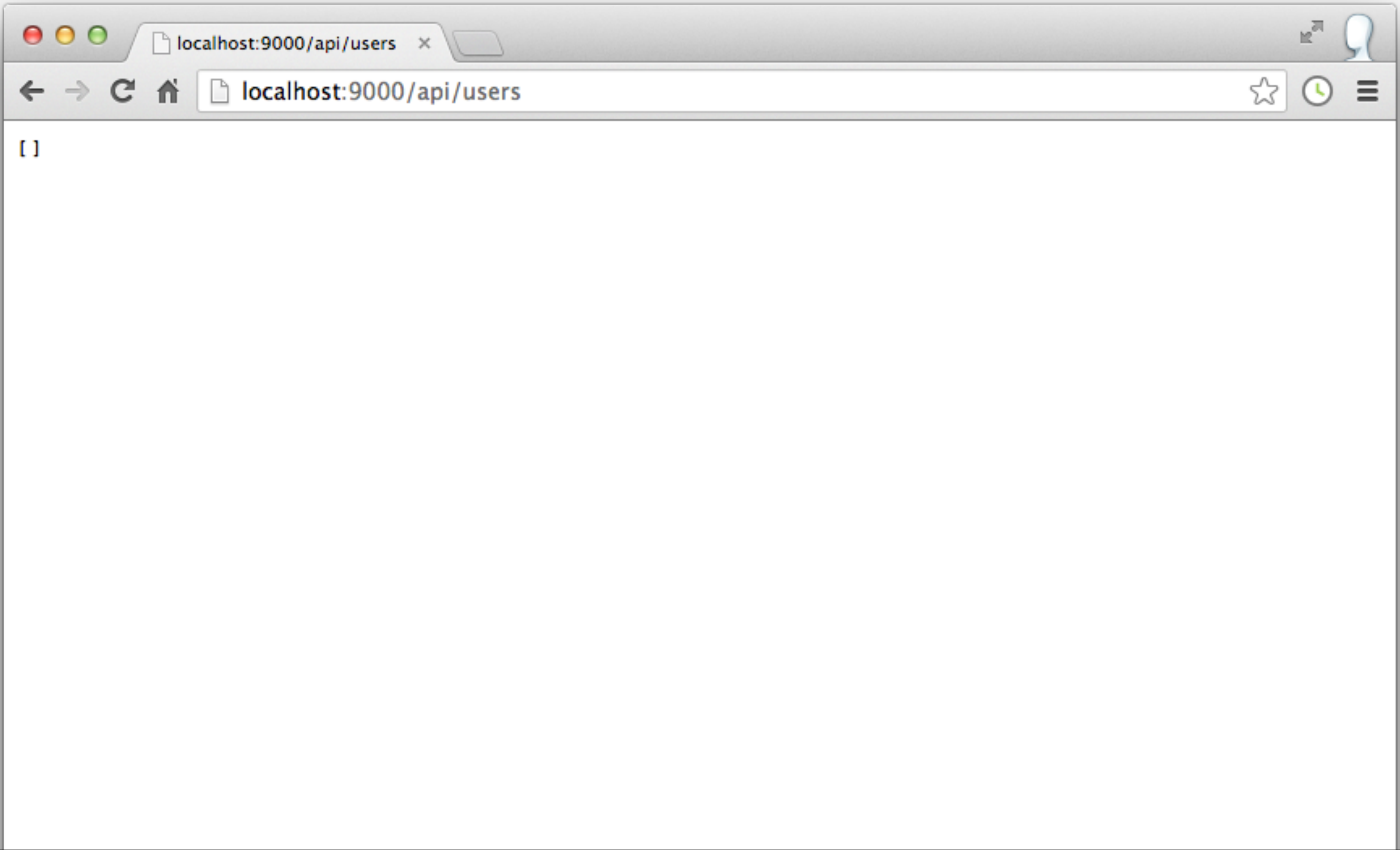
Welcome to Play

Congratulations, you've just created a new Play application. This page will help you with the next few steps.

You're using Play 2.2.1

```
GET      /api/users                              controllers.PacemakerAPI.users()
```

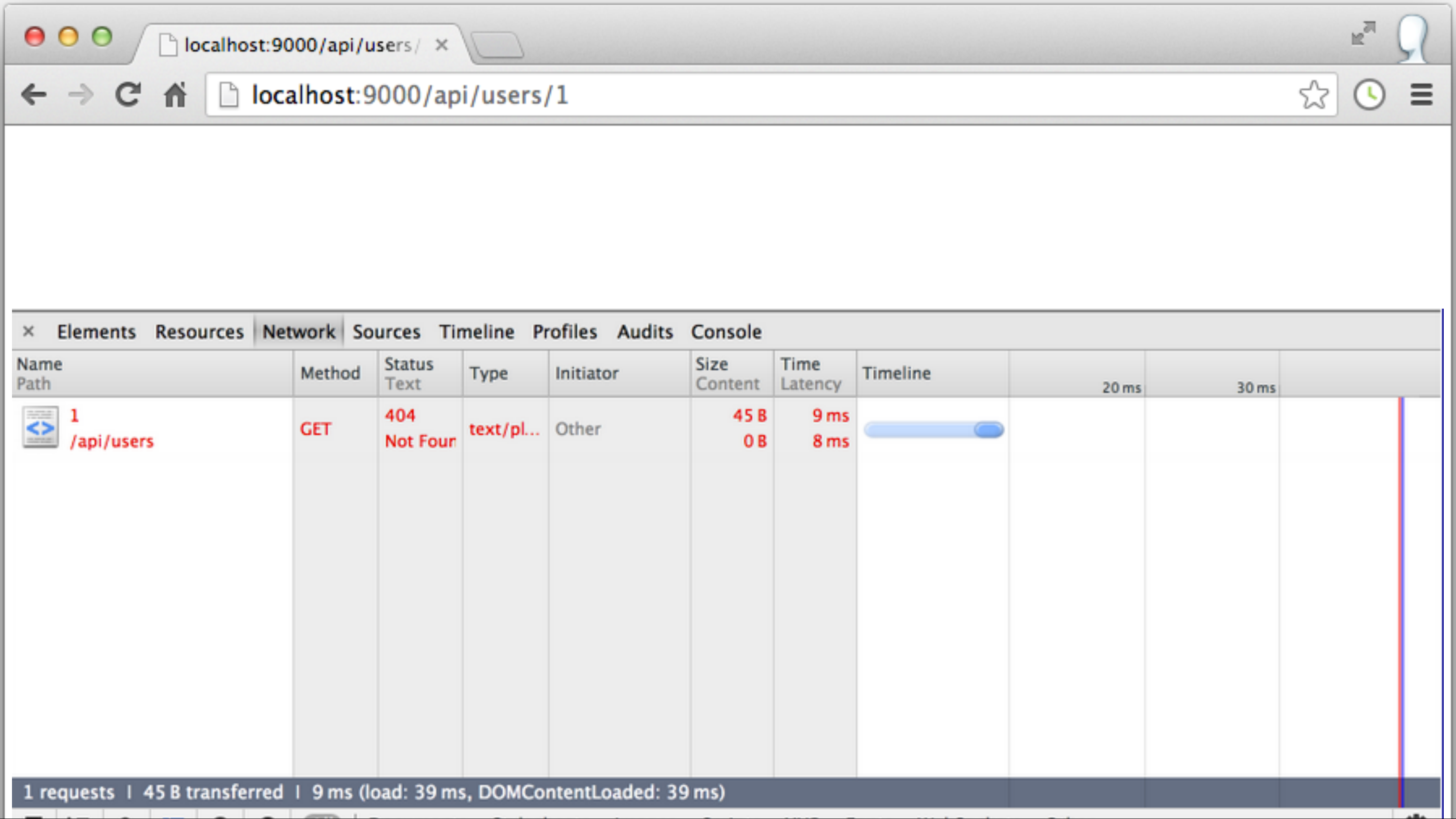
```java
public class PacemakerAPI extends Controller
{
  public static Result  users()
  {
    List<User> users = User.findAll();
    return ok(renderUser(users));
  }
…
}
```



```
[]
```

```
GET     /api/users/:id                                controllers.PacemakerAPI.user(id: Long)
```

```
public class PacemakerAPI extends Controller
{
  public static Result user(Long id)
  {
    User user = User.findById(id);
    return user==null? notFound() : ok(renderUser(user));
  }
…
}
```

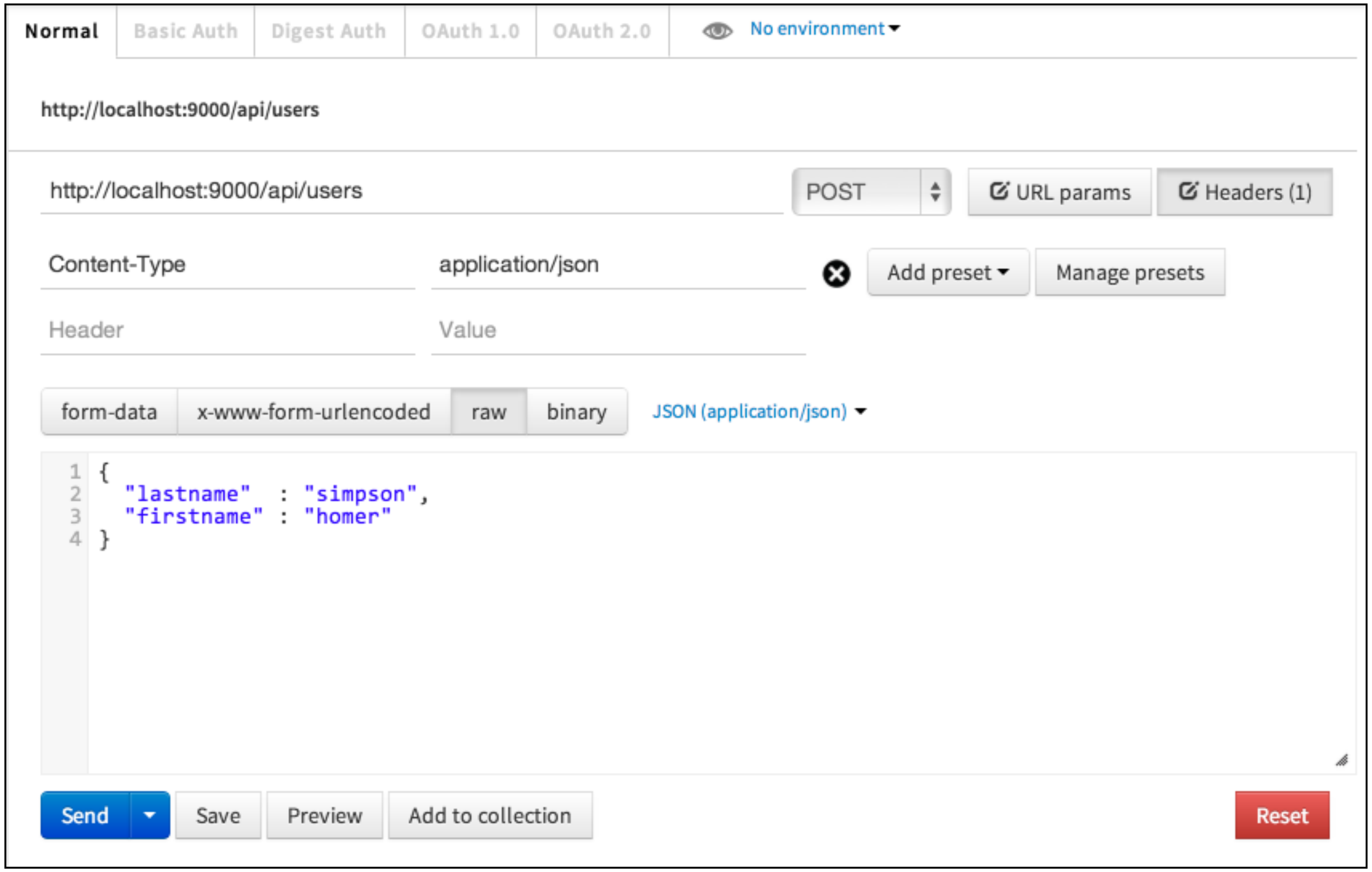


```
Name        Method  Status    Type      Initiator  Size     Time     Timeline
Path                Text                           Content  Latency
1           GET     404       text/pl…  Other      45 B     9 ms
/api/users          Not Foun                       0 B      8 ms

1 requests | 45 B transferred | 9 ms (load: 39 ms, DOMContentLoaded: 39 ms)
```

```
POST     /api/users                                    controllers.PacemakerAPI.createUser()
```

```
public class PacemakerAPI extends Controller
{
  public static Result createUser()
  {
    User user = renderUser(request().body().asJson().toString());
    user.save();
    return ok(renderUser(user));
  }
…
}
```

‘Postman’ Chrome extension



http://localhost:9000/api/users

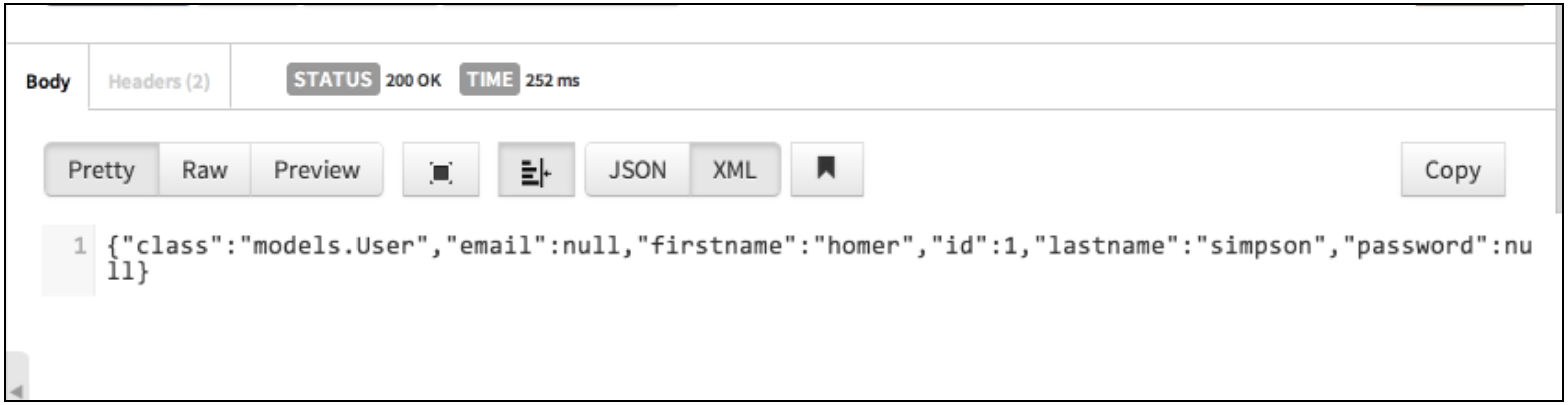
http://localhost:9000/api/users POST

Content-Type: application/json

```
{
  "lastname"  : "simpson",
  "firstname" : "homer"
}
```

```
GET     /api/users/:id                          controllers.PacemakerAPI.user(id: Long)
```

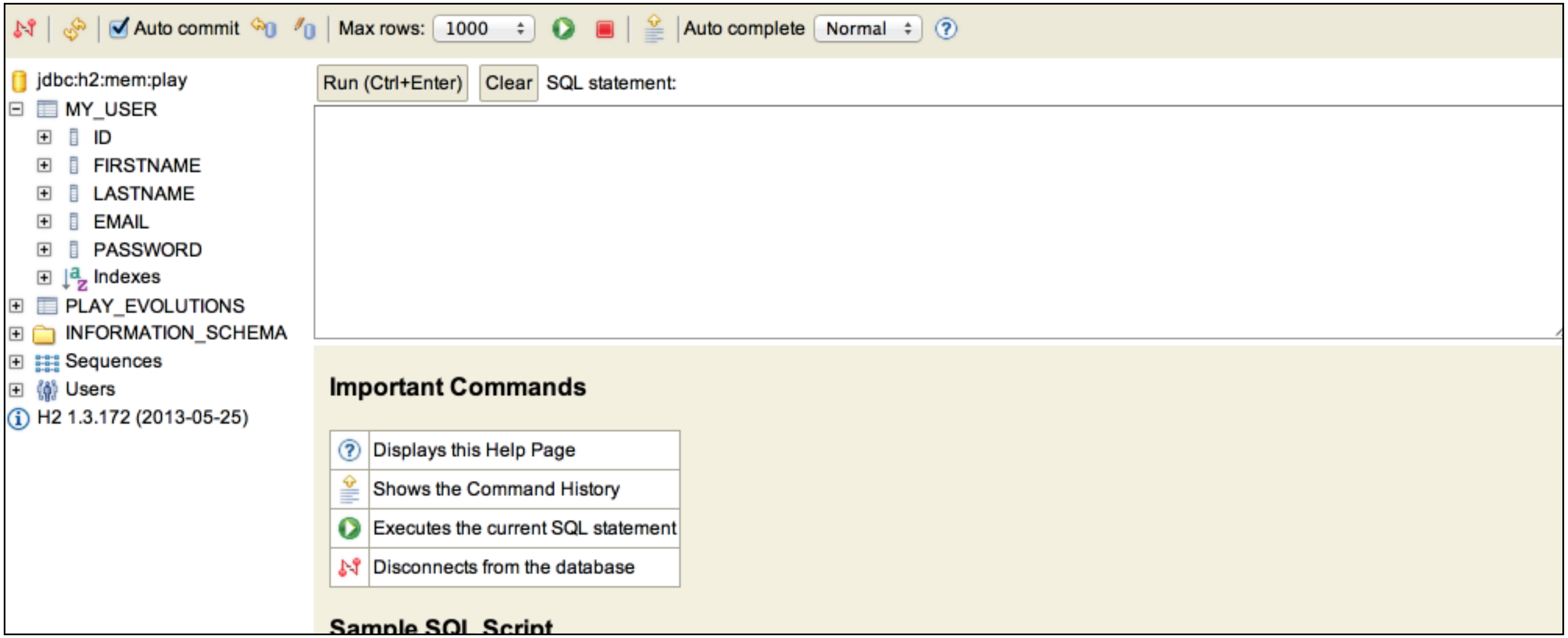
```
public class PacemakerAPI extends Controller
{
  public static Result user(Long id)
  {
    User user = User.findById(id);
    return user==null? notFound() : ok(renderUser(user));
  }
…
}
```



```
{"class":"models.User","email":null,"firstname":"homer","id":1,"lastname":"simpson","password":nu
ll}
```

‘Postman’
Chrome
extension

Browse Database



jdbc:h2:mem:play
- MY_USER
  - ID
  - FIRSTNAME
  - LASTNAME
  - EMAIL
  - PASSWORD
  - Indexes
- PLAY_EVOLUTIONS
- INFORMATION_SCHEMA
- Sequences
- Users

H2 1.3.172 (2013-05-25)

Important Commands

?	Displays this Help Page
	Shows the Command History
	Executes the current SQL statement
	Disconnects from the database

Sample SQL Script

- h2 database browser
- Be able to browse tables dynamically

Deployment

Change Database Connection Strings

```
db.default.driver=org.postgresql.Driver
db.default.url=${DATABASE_URL}

#db.default.driver=org.h2.Driver
#db.default.url="jdbc:h2:mem:play"
#db.default.user=sa
#db.default.password=""
```

Commit application to (local) git repository

```
$ git init
$ git add .
$ git commit -m "init"
$ heroku create
```

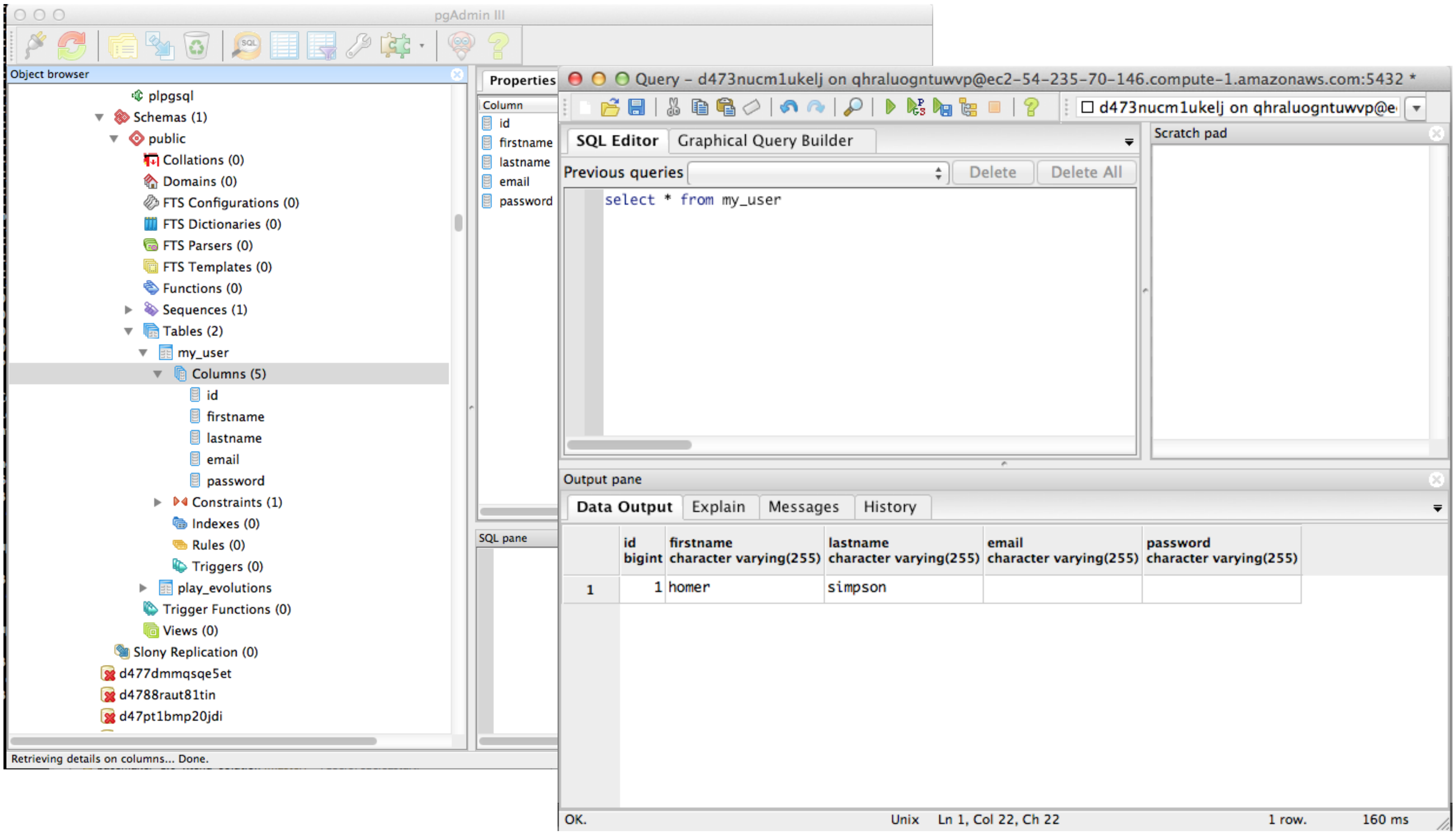
Push to heroku

```
git push heroku master
```

Test using generated heroku hosted public url

```
-----> Compiled slug size: 84.4MB
-----> Launching... done, v6
       http://polar-basin-1694.herokuapp.com deployed to Heroku

To git@heroku.com:polar-basin-1694.git
 * [new branch]      master -> master
```

Browse Database on Heroku



pgAdmin III

Object browser

- plpgsql
- Schemas (1)
  - public
    - Collations (0)
    - Domains (0)
    - FTS Configurations (0)
    - FTS Dictionaries (0)
    - FTS Parsers (0)
    - FTS Templates (0)
    - Functions (0)
    - Sequences (1)
    - Tables (2)
      - my_user
        - Columns (5)
          - id
          - firstname
          - lastname
          - email
          - password
        - Constraints (1)
        - Indexes (0)
        - Rules (0)
        - Triggers (0)
      - play_evolutions
    - Trigger Functions (0)
    - Views (0)
- Slony Replication (0)
- d477dmmqsqe5et
- d4788raut81tin
- d47pt1bmp20jdi

Retrieving details on columns... Done.

Properties

Column
id
firstname
lastname
email
password

SQL pane

Query - d473nucm1ukelj on qhraluogntuwvp@ec2-54-235-70-146.compute-1.amazonaws.com:5432 *

d473nucm1ukelj on qhraluogntuwvp@e

SQL Editor | Graphical Query Builder

Previous queries | Delete | Delete All

```
select * from my_user
```

Scratch pad

Output pane

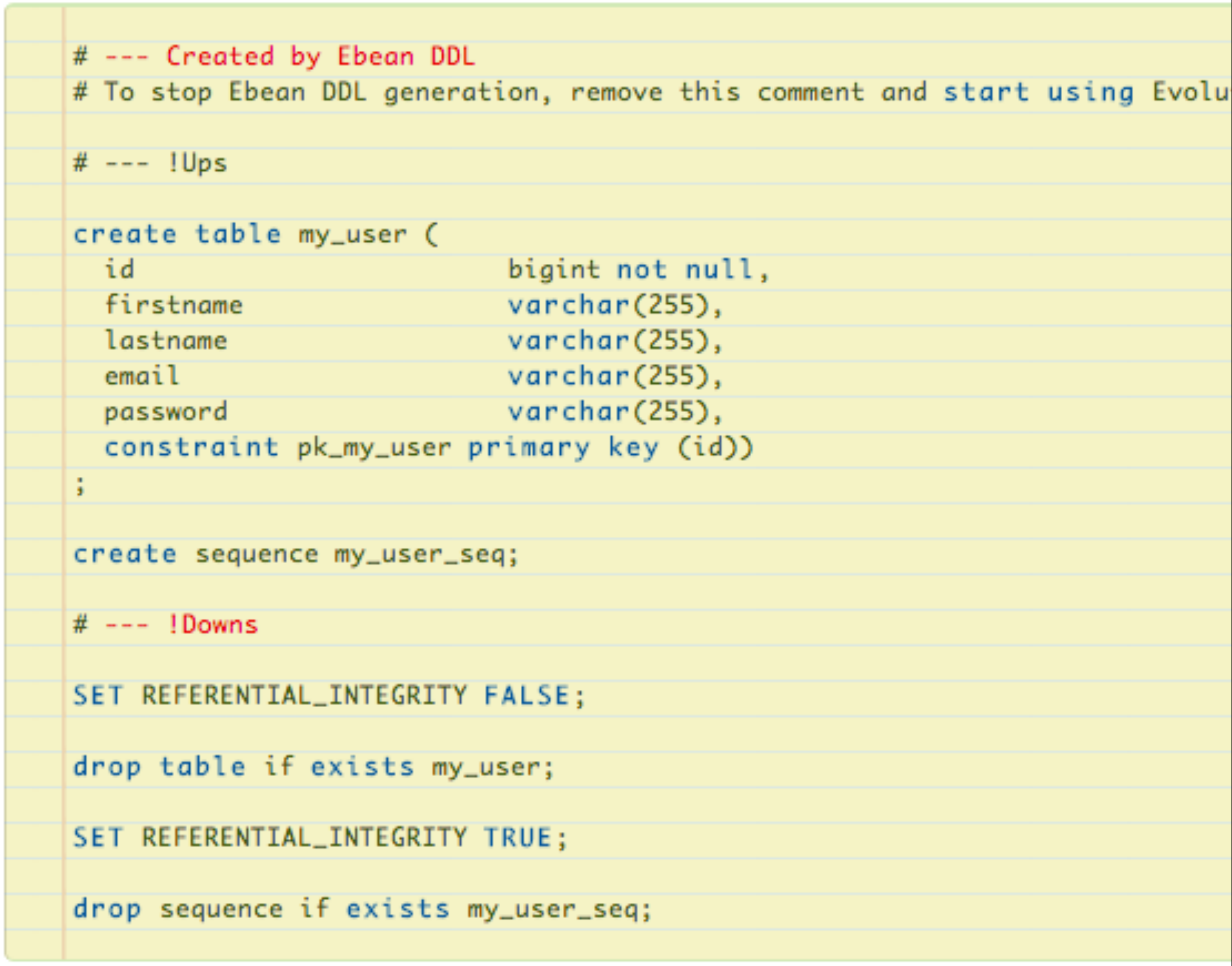
Data Output | Explain | Messages | History

	id bigint	firstname character varying(255)	lastname character varying(255)	email character varying(255)	password character varying(255)
1	1	homer	simpson		

OK. Unix Ln 1, Col 22, Ch 22 1 row. 160 ms

Database Evolutions

- Every time to make a change to the model, the database must be ‘evolved’
- This is done via play generated evolution scripts
- These scripts must be run before application starts.
- Multiple dialects of SQL - **Fun and Games!** (see lab exercises)



```
# --- Created by Ebean DDL
# To stop Ebean DDL generation, remove this comment and start using Evolu

# --- !Ups

create table my_user (
  id                        bigint not null,
  firstname                 varchar(255),
  lastname                  varchar(255),
  email                     varchar(255),
  password                  varchar(255),
  constraint pk_my_user primary key (id))
;

create sequence my_user_seq;

# --- !Downs

SET REFERENTIAL_INTEGRITY FALSE;

drop table if exists my_user;

SET REFERENTIAL_INTEGRITY TRUE;

drop sequence if exists my_user_seq;
```