

Web 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 LÁIRCE




Form Input

Web Development with Play

<input>

```
<form action="/register" method="POST">  
  
  <input type="text" name="firstName">  
  
  <input type="text" name="lastName">  
  
  <input type="text" name="email">  
  
  <input type="password" name="password">  
  
</form>
```



- Note the 'name' attributes

Register Route

POST /register

Accounts.register

- register() action will be responsible for:

- Recovering all of the fields “POST”ed by the user
- Save all these fields in a database

- Display the start screen again.

```
public class Accounts extends Controller
{
    //...

    public static void index()
    {
        render();
    }

    public static void register()
    {
        index();
    }
}
```

Controller Parameters

- Controllers can take parameters
- These will be passed from the form
- The names are highly significant

```
public static void register(String firstName, String lastName,  
                           String email,      String password)  
{  
    Logger.info(firstName + " " + lastName + " " + email + " " + password);  
  
    index();  
}
```

```
<form action="/register" method="POST">
```

```
    <input type="text" name="firstName">
```

```
    <input type="text" name="lastName">
```

```
    <input type="text" name="email">
```

```
    <input type="text" name="password">
```

```
</form>
```

- Direct mapping from 'name' attribute on input element to parameter name in controller/action

```
public class Accounts extends Controller
{
    //...
    public static void register(String firstName, String lastName,
                               String email,      String password)
    {
        Logger.info(firstName + " " + lastName + " " + email + " " + password);
    }
    index();
}
}
```

```
<form action="/register" method="POST">
```

```
<input type="text" name="firstName">
```

```
<input type="text" name="lastName">
```

```
<input type="text" name="email">
```

```
<input type="text" name="password">
```

```
</form>
```

- Direct mapping from 'name' attribute on input element to parameter name in controller/action

```
public class Accounts extends Controller
```

```
{
```

```
    //...
```

```
    public static void register(String firstName, String lastName,  
                               String email,      String password)
```

```
    {
```

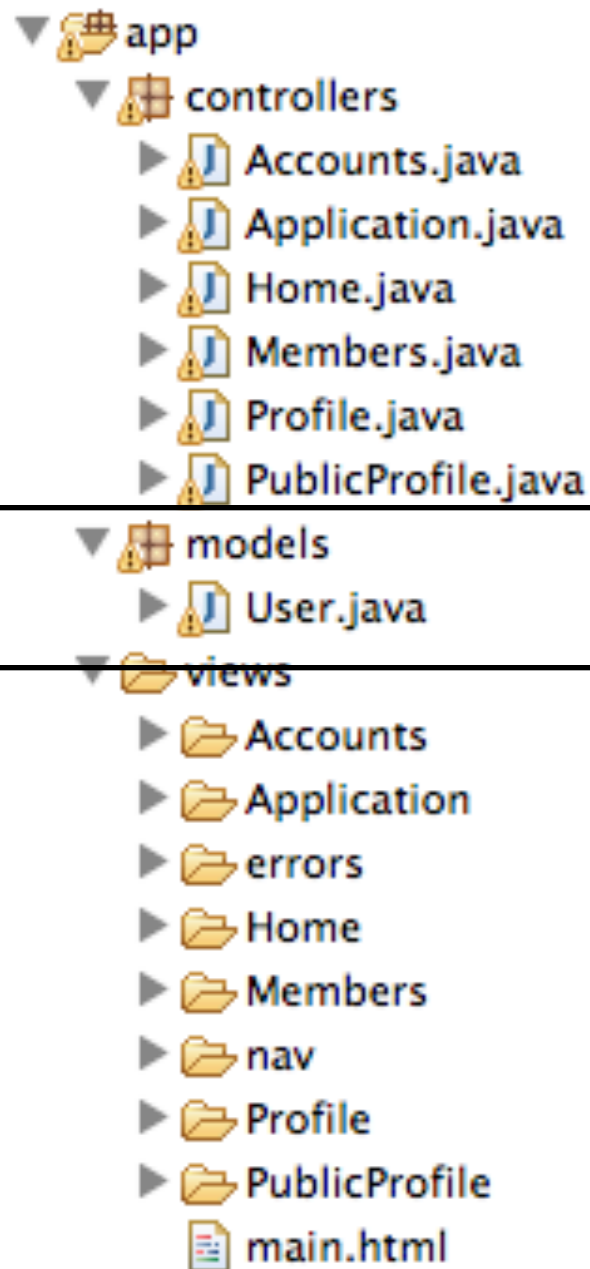
```
        Logger.info(firstName + " " + lastName + " " + email + " " + password);
```

```
        index();
```

```
    }
```

```
}
```

Database Models



- We would like to register new users in a database
- In Play, these are represented using ‘Models’
- Each table in a database can be represented by a java class
- Instances of this class (objects) will represent rows in the corresponding table

User Model

- Simple class to represent a user
- Public attributes represent fields
- Class 'extends' Model and is marked with @Entity annotation to indicate that it is to be saved to a database
- How this is done not our concern

```
package models;

import javax.persistence.Entity;

import play.db.jpa.Model;

@Entity
public class User extends Model
{
    public String firstName;
    public String lastName;
    public String email;
    public String password;

    public User(String firstName, String lastName,
                String email,      String password)
    {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
    }
}
```

Saving Objects to a Database

- In register (called when user 'submits' signup form):
 - Create a new User object
 - Save it!

```
public static void register(String firstName, String lastName,  
                           String email,      String password)  
{  
    Logger.info(firstName + " " + lastName + " " + email + " " + password);  
  
    User user = new User (firstName, lastName, email, password);  
    user.save();  
  
    index();  
}
```

Built in Database for test purposes

The screenshot displays the H2 database management tool interface. The main window shows a SQL statement: `SELECT * FROM USER`. The left sidebar shows the database structure, including a table named `user` with columns `id`, `email`, `firstname`, `lastname`, and `password`. The top toolbar includes buttons for `Auto commit`, `Max rows: 1000`, and `Auto complete`. A `Run (Ctrl+Enter)` button is visible next to the SQL statement.

Overlaid on the right is a `Login` dialog box with the following fields and buttons:

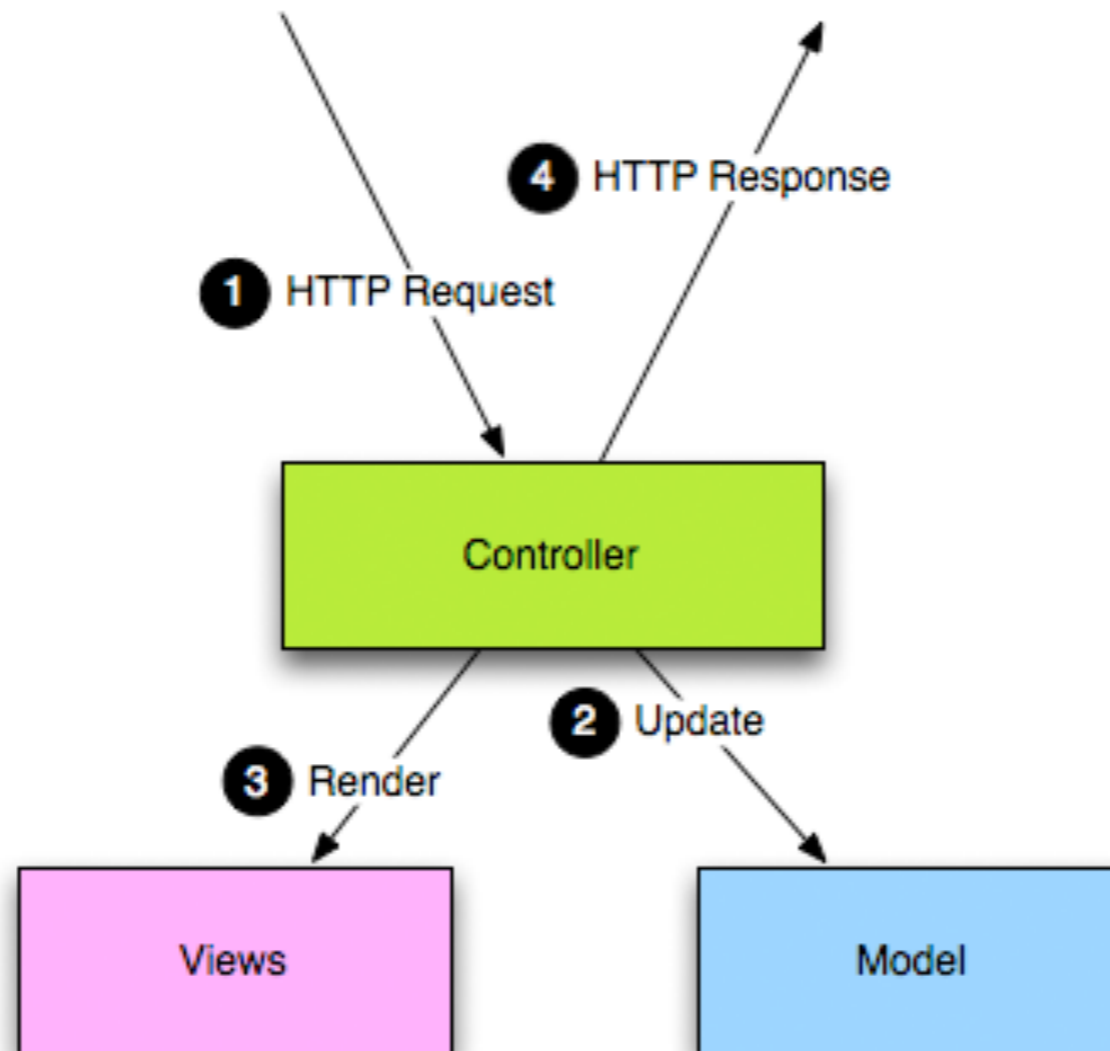
- `Saved Settings:` `Generic H2 (Embedded)` (dropdown)
- `Setting Name:` `Generic H2 (Embedded)` (text input) with `Save` and `Remove` buttons
- `Driver Class:` `org.h2.Driver` (text input)
- `JDBC URL:` `jdbc:h2:mem:play` (text input)
- `User Name:` `sa` (text input)
- `Password:` (empty text input)
- `Connect` and `Test Connection` buttons

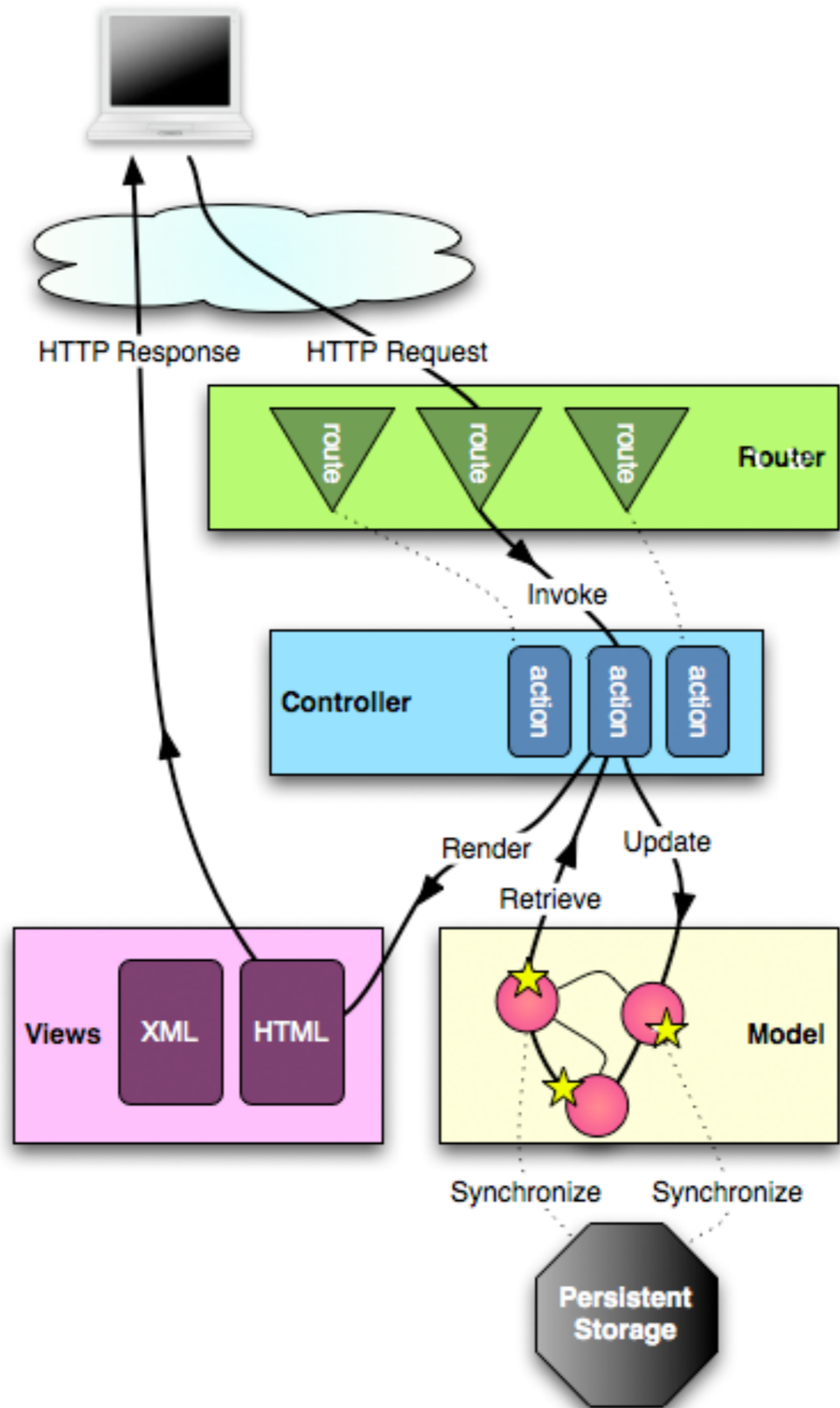
Below the SQL statement, there is an `Important Commands` section with a table of icons and descriptions:

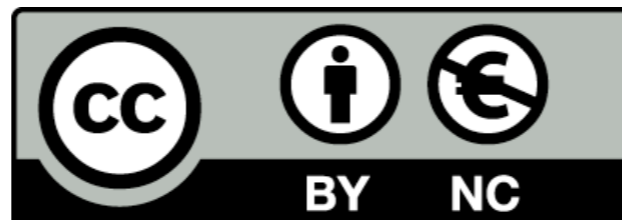
| | |
|--|------------------------------------|
| | Displays this Help Page |
| | Shows the Command History |
| | Executes the current SQL statement |
| | Disconnects from the database |

- Play comes with a database - which is a full relational db like MySQL
- 'Transient' - so all values are lost between program executions

HTTP Request/Response Cycle







Except where otherwise noted, this content is licensed under a Creative Commons Attribution-NonCommercial 3.0 License.

For more information, please see <http://creativecommons.org/licenses/by-nc/3.0/>

