# 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 LÁIRGE

eLearning support unit

# Models & Databases

## Web Development with Play

# Forms  & Databases

- Take the various parameters conveyed through a form and:
    - Formulate a Java class to hold these values
    - Create an instance of this class
        - Store this instance in a database
- In subsequent interaction consult this database an recover the instance if necessary
- These type of classes are called ‘Models’

# Purpose of the User Model

- A class that will represent details for a singe user.

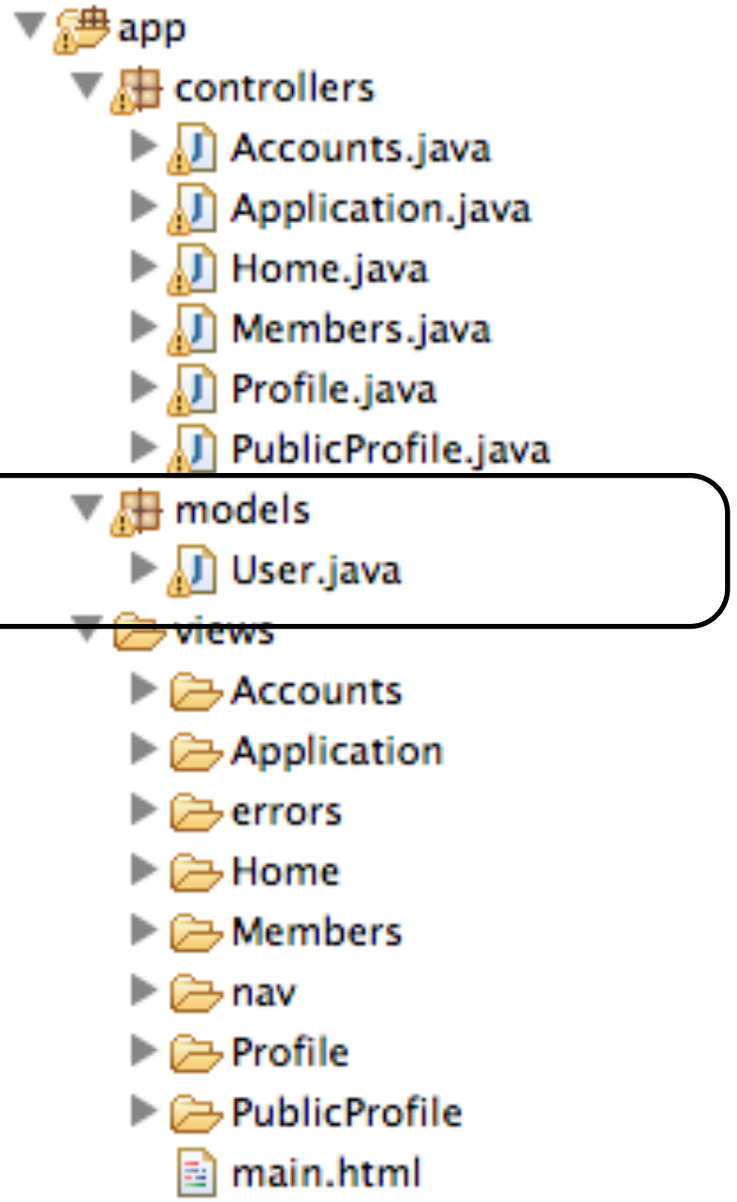
- Every time a new user ‘registers’ with the site, create a new User object and store in the database

- Every time a user tries to log in, ask the database if we have a matching User object (with same email/password).

- If we do, let this user in to the site.

- If not, then keep user out until correct ‘credentials’ provided.

# 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();
}
```

# Saving Objects to a Database

```
//...

  User user = new User (firstName, lastName, email, password);
  user.save();

//...
```

- Create new Java Object of type User, initialized with appropriate attributes

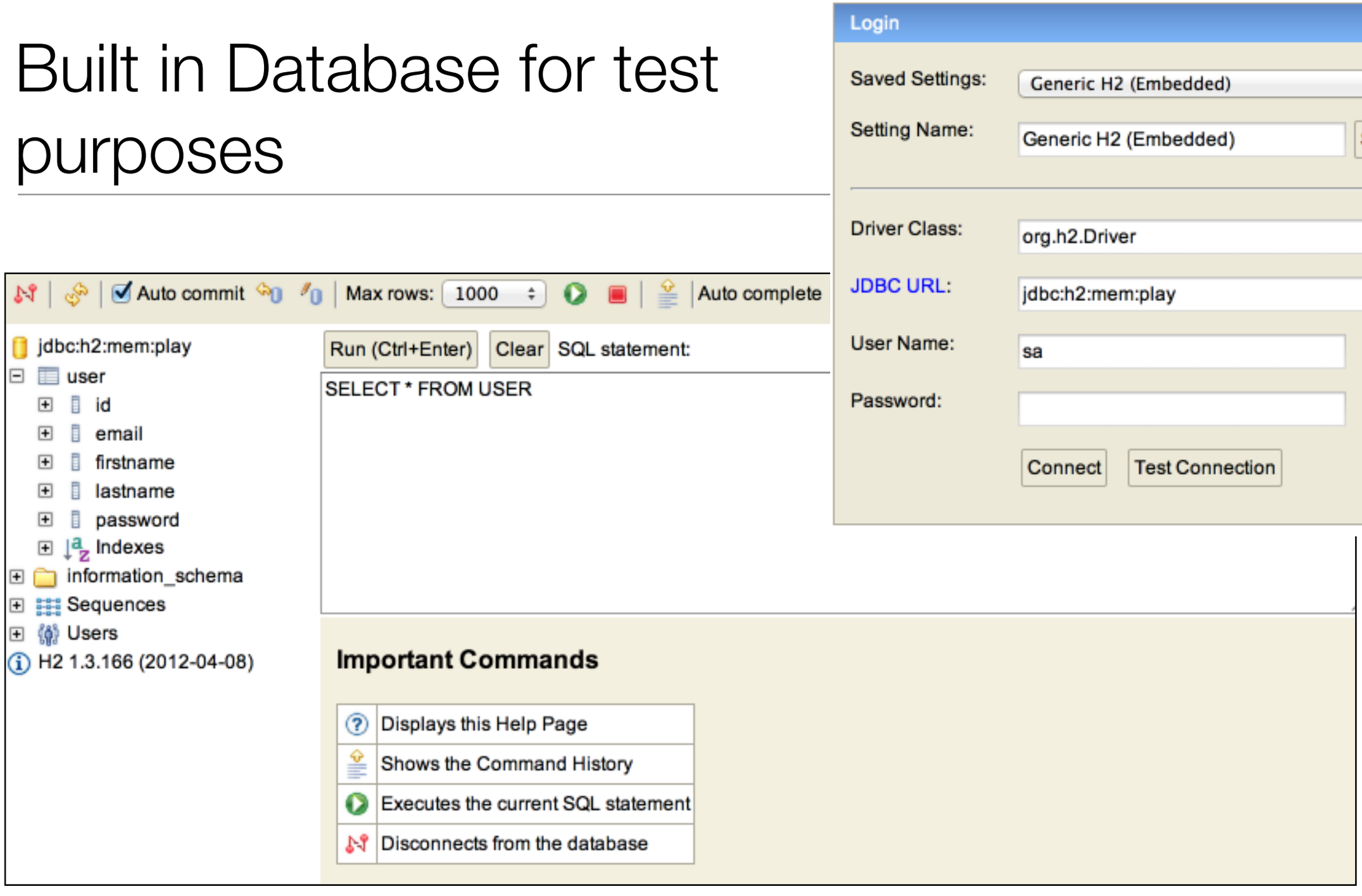
- Save this new object in the database

# Database Configuration

- The Database will be ‘in memory’
- Specified in ‘conf/application.conf’

```
# Database configuration
# ~~~~~
# Enable a database engine if needed.
#
# To quickly set up a development database, use either:
#   - mem : for a transient in memory database (H2 in memory)
#   - fs  : for a simple file written database (H2 file stored)

db=mem
```

# Built in Database for test purposes




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

# Browse/Edit/update...

- Enable in configuration:
  - db=mem
- This means ‘in memory’ database
- Then just browse to:
  - http://localhost:9000/@db
- when application is running

Max rows: 1000 | Auto complete Normal

Run (Ctrl+Enter) | Clear | SQL statement:

```
SELECT * FROM USER
```

SELECT * FROM USER;

| ID | EMAIL | FIRSTNAME | LASTNAME | PASSWORD |
|---|---|---|---|---|
| 2 | homer@simpson.com | Homer | Simpson | secret |
| 3 | marge@simpson.com | Marge | Simpson | secret |

(2 rows, 2 ms)

Edit

@edit SELECT * FROM USER;

| Action | ID | EMAIL | FIRSTNAME | LASTNAME | PASSWORD |
|---|---|---|---|---|---|
| | 2 | homer@simpson.com | Homer | Simpson | secret |
| | 3 | marge@simpson.com | Marge | Simpson | secret |
| | | | | | |

(2 rows, 2 ms)

@edit SELECT * FROM USER;

| Action | ID | EMAIL | FIRSTNAME | LASTNAME | PASSWORD |
|---|---|---|---|---|---|
| | 2 | homer@simpson.com | Homer | Simpson | secret |
| | 3 | marge@simpson.com | Marge | Simpson | secret |
| | | | | | |

(2 rows, 2 ms)

# User Model

- import libraries containing ‘annotations’ which are used to ‘mark’ classes with specific database-aware features

- Model base class ensures each object will have a unique ID + general purpose methods for :
  - find
  - save

- ‘@Entity’ implies that this class will be represented by a table in the database, with individual objects occupying each row

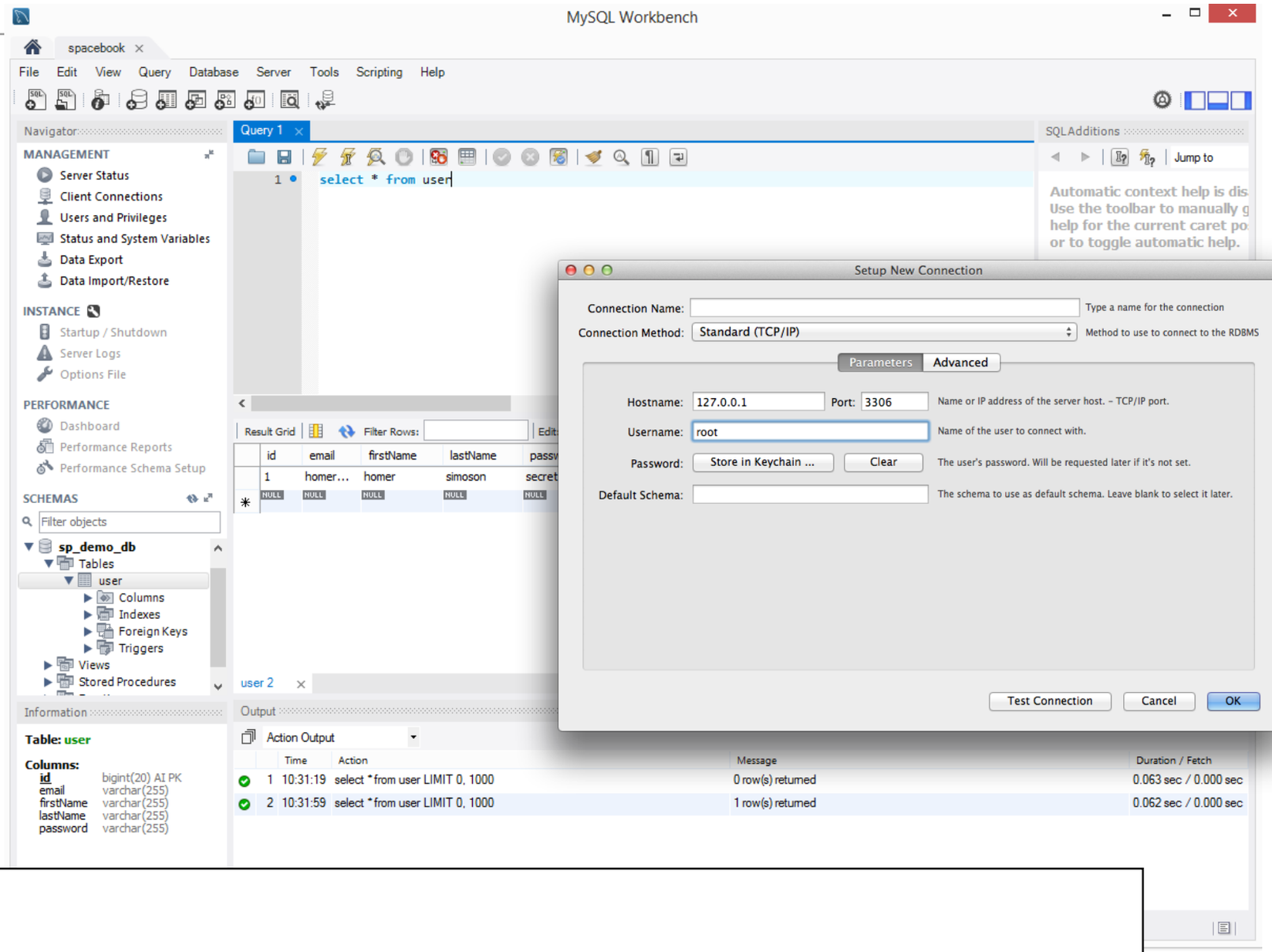
```
package models;

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;
  }
}
```

# Cloudbees Database

- When you deploy the app, it will not connect to this external database

- In Cloudbees you can create a database that will not me ‘transient’

- This will require new configuration on the conf/application.conf file:




```
#db=mem

db.url=jdbc:mysql://ec2-176-34-253-124.compute.amazonaws.com:3306/spacebook_db
db.driver=com.mysql.jdbc.Driver
db.user=spacebook_yourusername
db.pass=secret
jpa.ddl=create
```