

Application Development & Modelling

BSc in Applied Computing

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



Assignment 2 Extensions

Some Improvements

- Richer information in the data.yaml file
- Simplified Parameter passing from View to Action
- Preloading Images

data.yml: Preload Messages

```
Message(message1):  
  from : marge  
  to   : homer  
  messageText: 'how are things going?'  
  
Message(message2):  
  from : lisa  
  to   : homer  
  messageText: 'Get me outta here!'
```

- Specify 'Message' objects, including:
 - from and to objects (use pre-defined object names)
 - message Text field

data.yml: Preload Friendships

- Define object twice:
 - first time introduces essential attributes
 - second time integrated friendship relationships

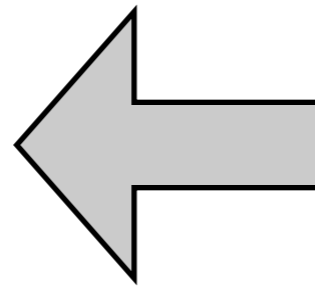
```
User(homer):
  firstName: homer
  lastName: Simpson
  email: homer@simpson.com
  password: secret
```

```
User(marge):
  firstName: Marge
  lastName: Simpson
  email: marge@simpson.com
  password: secret
```

```
Friendship(friend1):
  sourceUser: homer
  targetUser: marge
```

```
Friendship(friend2):
  sourceUser: homer
  targetUser: lisa
```

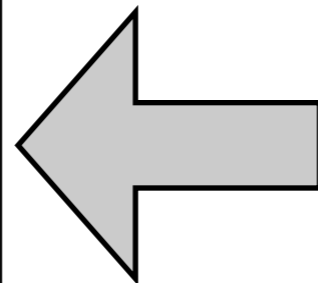
```
Friendship(friend3):
  sourceUser: marge
  targetUser: homer
```



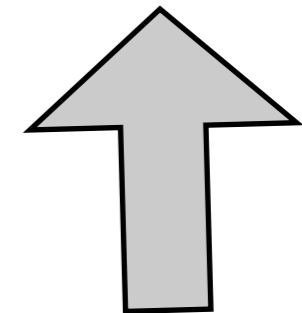
Define User Objects as usual

```
User(homer):
  firstName: Homer
  lastName: Simpson
  email: homer@simpson.com
  password: secret
  friendships:
    - friend1
    - friend2
```

```
User(marge):
  firstName: Marge
  lastName: Simpson
  email: marge@simpson.com
  password: secret
  friendships:
    - friend3
```



Introduce
'Friendship'
objects,
relating two
users



Refefine User objects
with Friendship list

```
User(homer):
  firstName: Homer
  lastName: Simpson
  email: homer@simpson.com
  password: secret

User(marge):
  firstName: Marge
  lastName: Simpson
  email: marge@simpson.com
  password: secret

Message(message1):
  from : marge
  to   : homer
  messageText: 'how are things going?'

Message(message2):
  from : lisa
  to   : homer
  messageText: 'Get me outta here!'

Friendship(friend1):
  sourceUser: homer
  targetUser: marge

Friendship(friend2):
  sourceUser: homer
  targetUser: lisa

Friendship(friend3):
  sourceUser: marge
  targetUser: homer

User(homer):
  firstName: Homer
  lastName: Simpson
  email: homer@simpson.com
  password: secret
  friendships:
    - friend1
    - friend2

User(marge):
  firstName: Marge
  lastName: Simpson
  email: marge@simpson.com
  password: secret
  friendships:
    - friend3
```

Simplified Parameter Passing

- 'name' field specified in 'input' parameter is matched to parameter to action

```
<form action="/register" method="POST">
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name="firstName">
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name="lastName">
    </div>
  </div>
  <div class="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name="nationality">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name="age">
    </div>
  </div>
  <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```


-
- Values are copied to the correctly named parameter

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

- Example: 'age'

- 'name' attribute in html element must be same as age parameter in action

```
<div class="two fields">  
  <div class="field">  
    <label>First Name</label>  
    <input placeholder="First Name" type="text" name="firstName">  
  </div>  
  <div class="field">  
    <label>Last Name</label>  
    <input placeholder="Last Name" type="text" name="lastName">  
  </div>  
</div>  
<div class="two fields">  
  <div class="field">  
    <label>Nationality</label>  
    <input placeholder="Nationality" type="text" name="nationality">  
  </div>  
  <div class="field">  
    <label>Age</label>  
    <input placeholder="Age" type="text" name="age">  
  </div>  
  <div class="field">  
    <label>Email</label>  
    <input placeholder="Email" type="text" name="email">  
  </div>  
<div class="field">
```

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

Simplification

- Prefix each name attribute with 'user.'

```
<form action="/register" method="POST">
  <div class="two fields">
    <div class="field">
      <label>First Name</label>
      <input placeholder="First Name" type="text" name="user.firstName">
    </div>
    <div class="field">
      <label>Last Name</label>
      <input placeholder="Last Name" type="text" name="user.lastName">
    </div>
  </div>
  <div class="two fields">
    <div class="field">
      <label>Nationality</label>
      <input placeholder="Nationality" type="text" name="user.nationality">
    </div>
    <div class="field">
      <label>Age</label>
      <input placeholder="Age" type="text" name="user.age">
    </div>
  </div>
  <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="user.email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="user.password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

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

- Action now take an object!
- Fields already filled with data from form

```
<div class="field">
  <label>First Name</label>
  <input placeholder="First Name" type="text" name="user.firstName">
</div>
<div class="field">
  <label>Last Name</label>
  <input placeholder="Last Name" type="text" name="user.lastName">
</div>
</div>
<div class="two fields">
  <div class="field">
    <label>Nationality</label>
    <input placeholder="Nationality" type="text" name="user.nationality">
  </div>
  <div class="field">
    <label>Age</label>
    <input placeholder="Age" type="text" name="user.age">
  </div>
  <div class="field">
    <label>Email</label>
    <input placeholder="Email" type="text" name="user.email">
  </div>
  <div class="field">
    <label>Password</label>
    <input type="password" name="user.password">
  </div>
  <button class="ui blue submit button">Submit</button>
</form>
```

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

Preloading Images

- Images cannot be loaded from the data.yaml file
- Recall, the yaml file is loaded in bootstrap:
- We can intervene here, and load specific extra information - including images!

```
@OnApplicationStart
public class Bootstrap extends Job
{
    public void doJob()
    {
        Fixtures.deleteDatabase();
        Fixtures.loadModels("data.yaml");
    }
}
```

```
@OnApplicationStart
public class Bootstrap extends Job
{
    public void doJob() throws FileNotFoundException
    {
        Fixtures.deleteDatabase();
        Fixtures.loadModels("data.yml");

        String photoName = "homer.gif";
        Blob blob = new Blob ();
        blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
        User homer = User.findByEmail("homer@simpson.com");
        homer.profilePicture = blob;
        homer.save();
    }
}
```

Preloading Images Example

The image file must be in the folder of your app

```
String photoName = "homer.gif";
```


Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

```
String photoName = "homer.gif";
```

```
Blob blob = new Blob ();
```

Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

```
String photoName = "homer.gif";
```

```
Blob blob = new Blob ();
```

```
blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
```

Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

```
String photoName = "homer.gif";
```

```
Blob blob = new Blob ();
```

```
blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
```

```
User homer = User.findByEmail("homer@simpson.com");
```

Locate a specific model object

```
;
```

Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

```
String photoName = "homer.gif";
```

```
Blob blob = new Blob ();
```

```
blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
```

```
User homer = User.findByEmail("homer@simpson.com");
```

```
homer.profilePicture = blob;
```

Locate a specific model object

Set the blob in that mode

Preloading Images Example

The image file must be in the folder of your app

Create an empty Blob object

Read the image and store in the blob

```
String photoName = "homer.gif";
```

```
Blob blob = new Blob ();
```

```
blob.set(new FileInputStream(photoName), MimeTypes.getContentType(photoName));
```

```
User homer = User.findByEmail("homer@simpson.com");
```

```
homer.profilePicture = blob;
```

Locate a specific model object

```
homer.save();
```

Set the blob in that mode

Save the model



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/>

