Create Emitter (Twitter like app) with SPRING BOOT

Presented by ARMS（THAILAND）Co., Ltd.

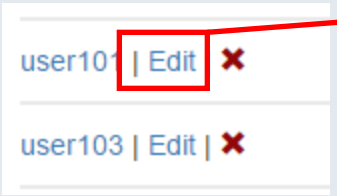
Index

1. Update and Delete

1-1. Update user

- In this section, we will edit user profile.

<a th:href="@{/user/edit/} + ${user.id}" th:text="'Edit'">



We have to edit a user by the user's id

Create an edit method in UserController and call edit.html from the edit method

```
@RequestMapping(value = "/user/edit/{userId}", method = RequestMethod.GET)
  public ModelAndView edit(@PathVariable int userId, ModelAndView modelAndView) {
    modelAndView.setViewName("user/edit");
    return modelAndView;
  }
```

"/user/edit" user folder, edit.html

value = "/user/edit/{userId}"

@PathVariable("userId") int userId

Continue on next page

1. Update and Delete

1-1. Update user

- We want to find user information by userId

Create a method in UserService to find one user by userId

UserService

```
public User findOne(int userId) {
      return userRepository.findOne(userId);
  }
```

Call the above method from the edit method in UserController

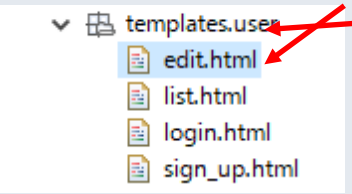
UserController

```
@RequestMapping(value = "/user/edit/{userId}", method = RequestMethod.GET)
  public ModelAndView edit(@PathVariable int userId, ModelAndView modelAndView) {
        modelAndView.addObject("user", userService.findOne(userId));
        modelAndView.setViewName("user/edit");
      return modelAndView;
  }
```

1. Update and Delete

1-1. Update user

- Create edit.html under templates.user



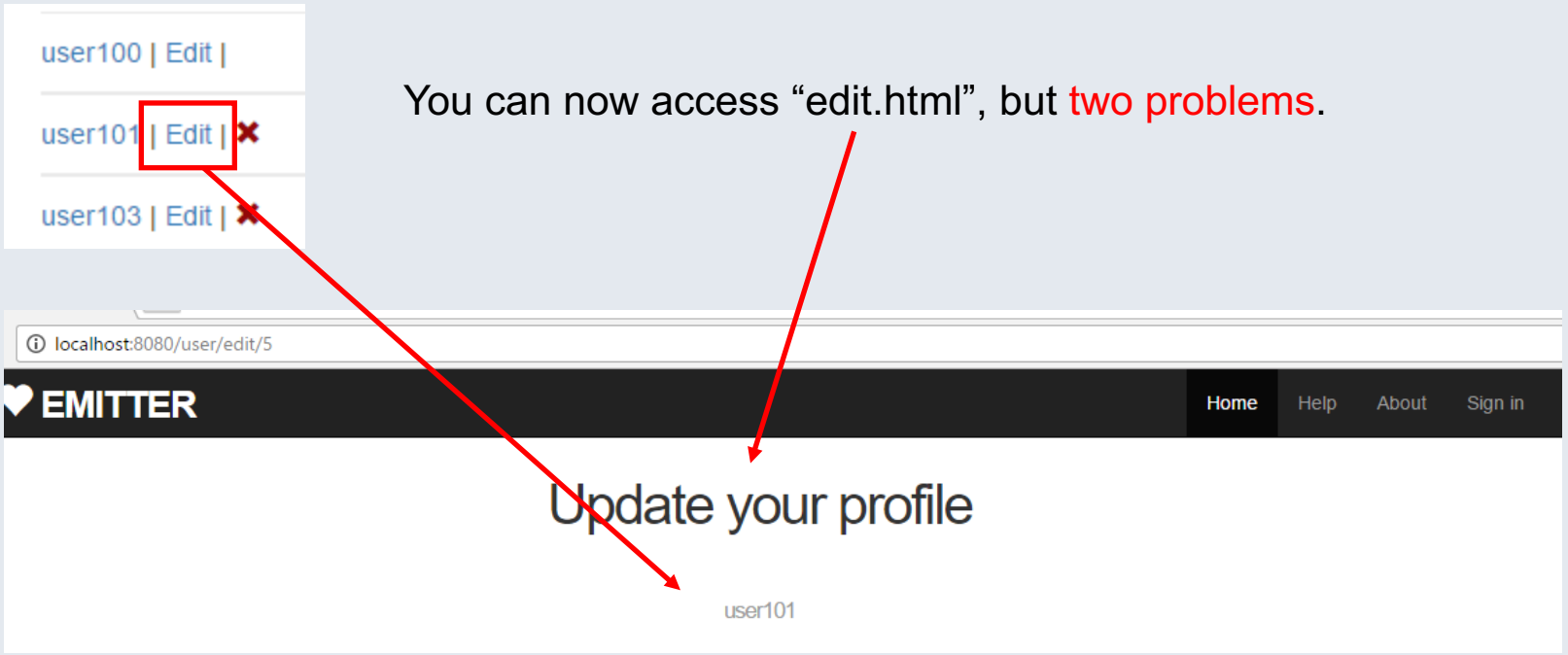
Add the following code to edit.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorator="layout/main"
>
<head>
        <title>Edit User</title>
</head>
<body>
        <div layout:fragment="content" th:remove="tag">
        <!-- Users edit.html inserted into main.html -->
                <h1>Update your profile</h1>
                <div class="row">
                   <div class="col-md-6 col-md-offset-3">
                        <h2 th:text="${user.name}">user name (replaced with image)</h2>
                   </div>
                </div>
        <!-- Users edit.html inserted into main.html End -->
        </div>
</body>
</html>
```

1. Update and Delete

1-1. Update user

- Restart your app and access ----------:8080/, and then go to a user edit page.



The two problems are “you can access without login” and “no form to edit user”

1. Update and Delete

1-1. Update user

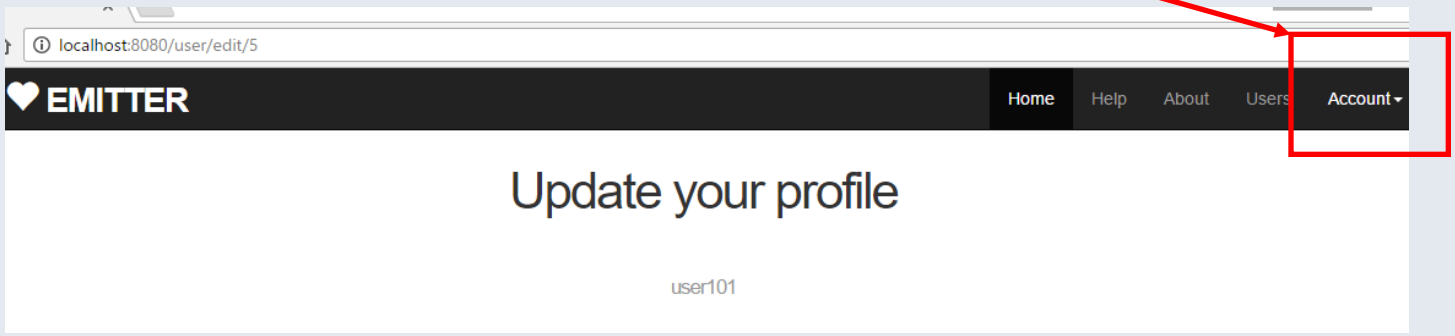
- You can access without “login”

UserController

```
@RequestMapping(value = "/user/edit/{userId}", method = RequestMethod.GET)
  public ModelAndView edit(@PathVariable int userId, ModelAndView modelAndView, Principal principal) {
        modelAndView.addObject("user", userService.findOne(userId));
        modelAndView.setViewName("user/edit");
    return modelAndView;
  }
```

Add Principal object as a method argument.

Restart your app and access ----------:8080/, and then go to a user edit page. Now only login users can access the edit page.



1. Update and Delete

1-1. Update user

- No form to edit users

Create a form object to receive user input from the View(html)

Create UserEditForm under com.arms.app.user

Add the following code to UserEditForm

```
package com.arms.app.user;

import lombok.Data;
import org.hibernate.validator.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
public class UserEditForm {

    @NotNull
    private Integer userId;

    @NotEmpty
    private String name;

    private String email;

    @NotEmpty
    private String password;
}
```

Make sure to add both of them

1. Update and Delete

1-1. Update user

- Add the following method to UserService

UserService

```
public UserEditForm setUserEditForm(int userId) {
      User user = userRepository.findOne(userId);
      UserEditForm userEditForm = new UserEditForm();
      userEditForm.setUserId(user.getId());
      userEditForm.setName(user.getName());
      userEditForm.setEmail(user.getEmail());
      return userEditForm;
  }
```

Call the above method from the edit method in UserController

UserController

```
@RequestMapping(value = "/user/edit/{userId}", method = RequestMethod.GET)
  public ModelAndView edit(@PathVariable int userId, ModelAndView modelAndView, Principal principal) {
      modelAndView.addObject("user", userService.findOne(userId));
      modelAndView.addObject("userEditForm", userService.setUserEditForm(userId));
      modelAndView.setViewName("user/edit");
      return modelAndView;
  }
```

1. Update and Delete

1-1. Update user

- Add the following form code to edit.html

```
<form th:action="@{/user/edit}" th:object="${userEditForm}" id="userEditForm"
method="post" role="form" name="form1">
    <input type="hidden" th:field="*{userId}" />
    <div>
        <div class="form-group"
th:classappend="${#fields.hasErrors('name')}? 'has-error'">
            <label for="name">Name :</label>
            <input id="name" type="text" class="form-control"
th:field="*{name}" /><br/>
            <span th:if="*{#fields.hasErrors('name')}" th:errors="*{name}"
class="help-block">error!</span>
        </div>

        <div class="form-group"
th:classappend="${#fields.hasErrors('email')}? 'has-error'">
            <label for="email">E-mail :</label>
            <input id="email" type="text" class="form-control"
th:field="*{email}" readonly="readonly" /><br/>
            <span th:if="*{#fields.hasErrors('email')}" th:errors="*{email}"
class="help-block">error!</span>
        </div>
```

☆See the next page on where to insert this code

Continue on next page

1. Update and Delete

1-1. Update user

- Add the following form code to edit.html

```
<form th:action="@{/user/edit}" th:object="${userEditForm}" id="userEditForm"
    <div class="form-group" th:classappend="${#fields.hasErrors('password')}?
'has-error'">
        <label for="password">Password :</label>
        <input id="password" type="password" class="form-control"
th:field="*{password}" /><br/>
        <span th:if="*{#fields.hasErrors('password')}"
th:errors="*{password}" class="help-block">error!</span>
    </div>
</div>
<div class="text-center">
    <input class="btn btn-primary btn-large" type="submit"
name="submit" value="Update my account" /><br/><br/>
</div>
</form>
```

```
<h1>Update your profile</h1>
<div class="row">
    <div class="col-md-6 col-md-offset-3">
        <form th:action="@{/user/edit}" th:object="${userEditForm}" id="userEdit
            <input type="hidden" th:field="*{userId}" />
```

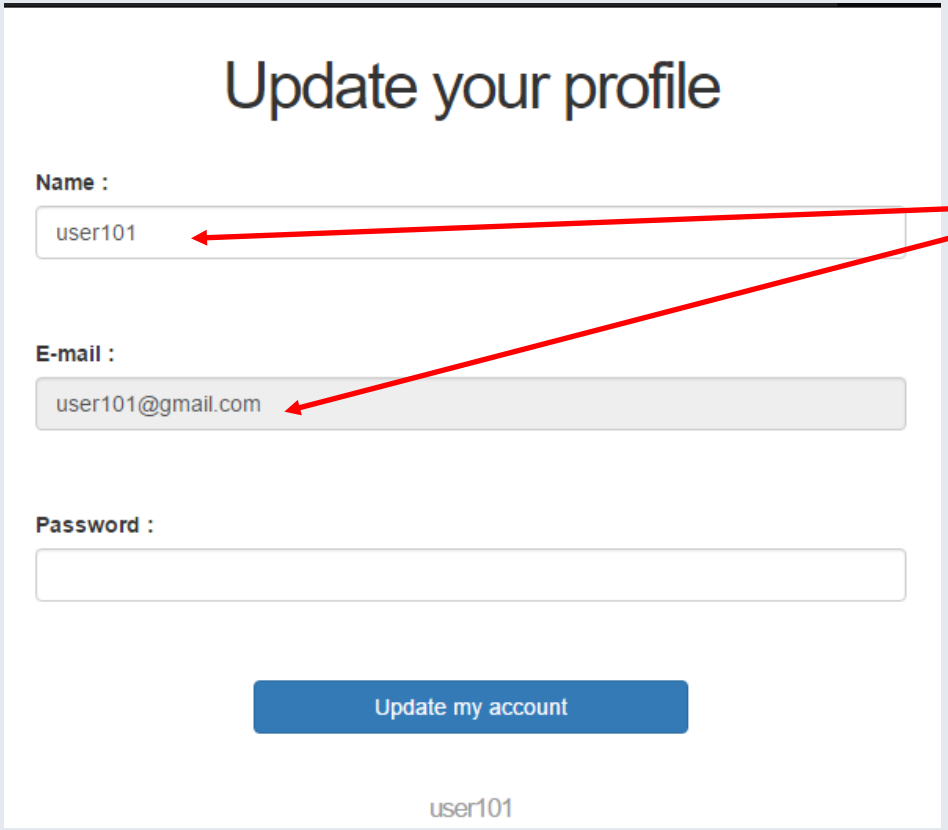
Insert the form tag here

```
            <input class="btn btn-primary btn-large" type="submit" name="sub
        </div>
    </form>
    <h2 th:text="${user.name}">user name (replaced with image)</h2>
```

1. Update and Delete

1-1. Update user

- Restart your app and access ----------:8080/, and go to the user edit page.



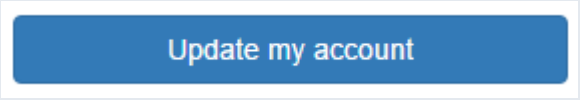
Now you can see edit form with Name and Email values

But when you update, you will see an error

1. Update and Delete

1-1. Update user

- Create a method in UserController to actually update your profile




```
<form th:action="@{/user/edit}" th:object="${userEditForm}"
id="userEditForm" method="post"
```

Create updateUser method in UserService

UserService

```
public void updateUser(UserEditForm userEditForm) throws NoSuchAlgorithmException {
        Date nowDate = Calendar.getInstance().getTime();
        User user = userRepository.findOne(userEditForm.getUserId());
        user.setName(userEditForm.getName());
        user.setPassword(passwordEncoder.hashMD5(userEditForm.getPassword()));
        user.setUpdated(nowDate);
        userRepository.save(user);
    }
```

In the next page, call the above method from UserController

1. Update and Delete

1-1. Update user

- Create an edit method with post in UserController

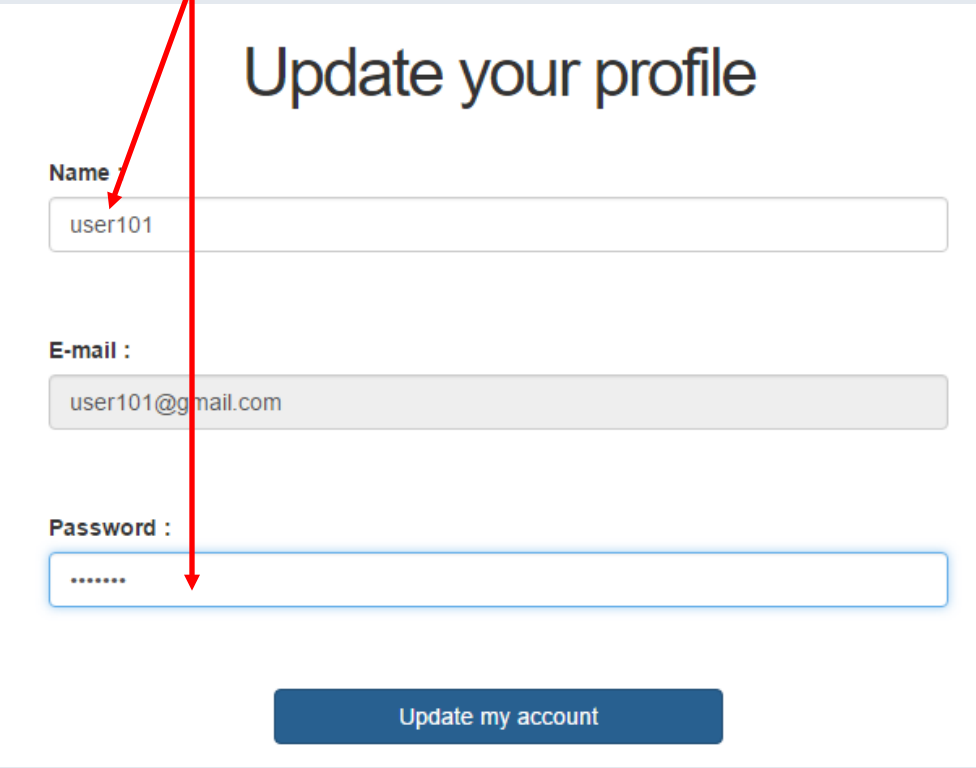
UserController

```
@RequestMapping(value = "/user/edit", method = RequestMethod.POST)
   public Object edit(@Validated UserEditForm userEditForm, BindingResult bindingResult,
RedirectAttributes redirectAttributes, ModelAndView modelAndView, Principal principal) throws
NoSuchAlgorithmException {

      if(bindingResult.hasErrors()) {
         modelAndView.addObject("user", userService.findOne(userEditForm.getUserId()));
         modelAndView.setViewName("user/edit");
         return modelAndView;
      }

      userService.updateUser(userEditForm);
      redirectAttributes.addFlashAttribute("message", "User was updated.");
      return "redirect:/";
   }
```

1. Update and Delete

1-1. Update user

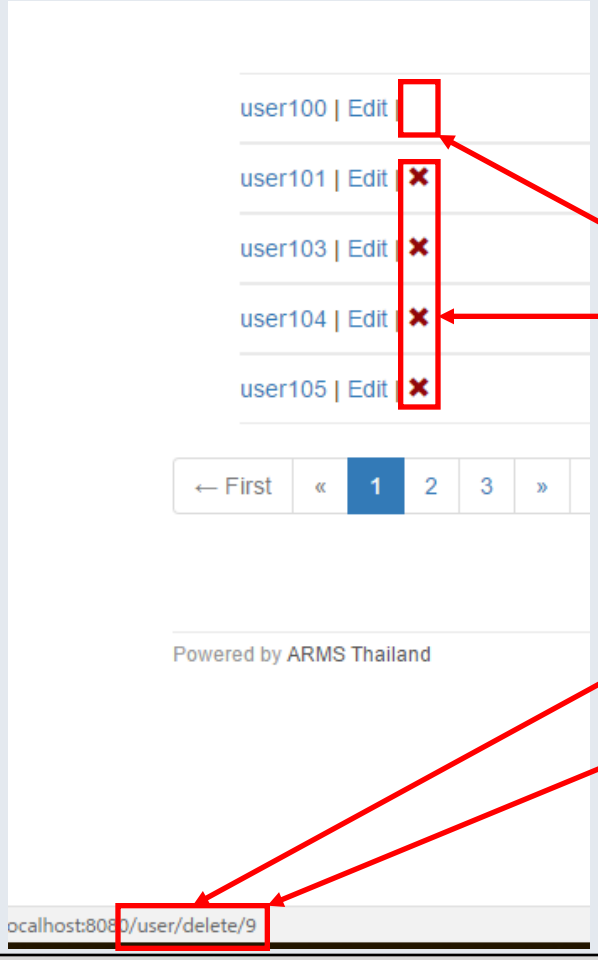
- Restart your app and access ----------:8080/, and see if you can update your profile



1. Update and Delete

1-2. Delete user

- In this section, we will delete users



Delete button(icon) specification

```
<a th:if="${userId != user.id}"
th:href="@{/user/delete/} + ${user.id}"
```

1. Update and Delete

1-2. Delete user

- Add deleteUser method to UserService

UserService

```
public void deleteUser(int userId){
        userRepository.delete(userId);
   }
```

Call the above method from delete method in UserController

UserController

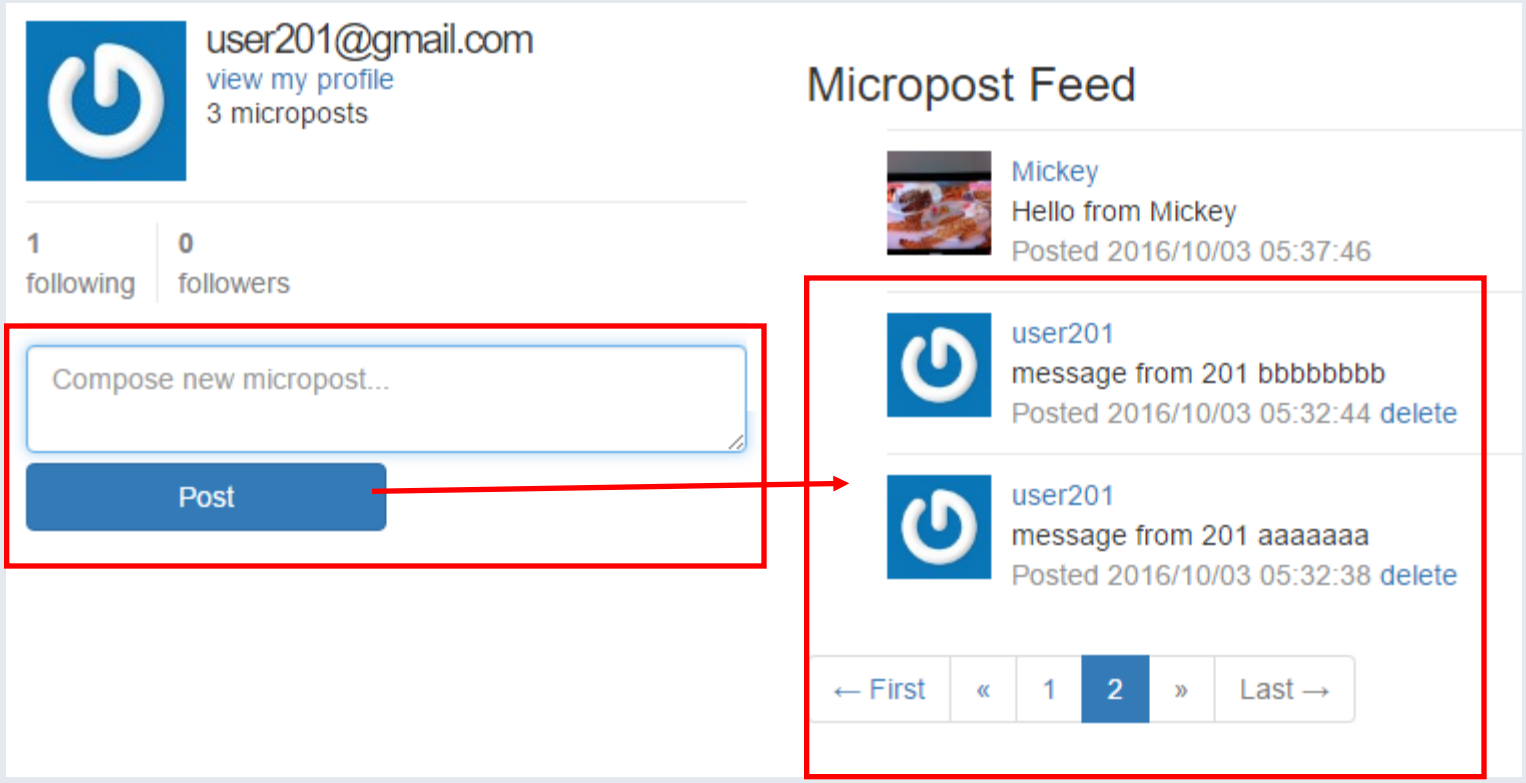
```
@RequestMapping(value = "/user/delete/{userId}")
   public String delete(@PathVariable int userId, RedirectAttributes redirectAttributes) {
       userService.deleteUser(userId);
       redirectAttributes.addFlashAttribute("message", "User was deleted.");
       return "redirect:/";
   }
```

Restart your app and see if you can delete users.!!!

2. Create Microposts

2-1. Summary

- We will create a micropost page like below.



2. Create Microposts

2-1. Summary

- We will also create a relationship between user and micropost table

user table

id	name	email	password
12	user200	user200@gmail.com	1d0258c2440a8d19e
13	user201	user201@gmail.com	1d0258c2440a8d19e
14	user202	user202@gmail.com	1d0258c2440a8d19e
15	user203	user203@gm	
16	user204	user204@gm	

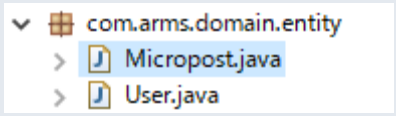
micropost table

id	content	user_id	created	updated
56	Message from 200 200000000000000	12	2016-10-03 08:31:14	2016-10-03 08:31:14
57	Message from 200 2000000000000	12	2016-10-03 08:31:26	2016-10-03 08:31:26
45	message from 201 aaaaaaa	13	2016-10-03 05:32:38	2016-10-03 05:32:38
46	message from 201 bbbbbbbb	13	2016-10-03 05:32:44	2016-10-03 05:32:44
48	message from 202 2bbbbbbbbbb	14	2016-10-03 05:33:36	2016-10-03 05:33:36
47	message from 202 2aaaaaaaaaaaaaa	14	2016-10-03 05:33:29	2016-10-03 05:33:29
49	message from 203 3aaaaaaaaaa	15	2016-10-03 05:34:22	2016-10-03 05:34:22
50	Message from 204 4aaaaaaaaaaaa	16	2016-10-03 05:34:47	2016-10-03 05:34:47
51	message from 205 5aaaaaaaaaaaaa	17	2016-10-03 05:35:39	2016-10-03 05:35:39
52	message from 206 6aaaaaaaaaaaaa	18	2016-10-03 05:36:01	2016-10-03 05:36:01
53	Message from 211 this should not be displayed on M...	24	2016-10-03 05:36:48	2016-10-03 05:36:48

2. Create Microposts

2-2. Create entity class

- Create an entity class which is mapped to the micropost table



Create Micropost under com.arms.domain.entity

Add the following code to Micropost

```
package com.arms.domain.entity;

import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import lombok.Data;

@Entity
@Data
public class Micropost {

    private int id;
    private String content;
    private Date created;
    private Date updated;
```

Continue on next page

2. Create Microposts

2-2. Create entity class

- Add the following code to micropost

```
@Id
@GeneratedValue
@Column(name = "id")
public int getId() {
    return id;
}
public void setId(int id) {
    this.id = id;
}

@Basic
@Column(name = "content")
public String getContent() {
    return content;
}
public void setContent(String content) {
    this.content = content;
}
```

Continue on next page

2. Create Microposts

2-2. Create entity class

- Add the following code to micropost

```
    @Basic
    @Column(name = "created")
    public Date getCreated() {
        return created;
    }
    public void setCreated(Date created) {
        this.created = created;
    }

    @Basic
    @Column(name = "updated")
    public Date getUpdated() {
        return updated;
    }
    public void setUpdated(Date updated) {
        this.updated = updated;
    }
}
```

2. Create Microposts

2-3. Create table relation

- In this section, we will create a relationship between user table and micropost table.

  One user can have many microposts.(OneToMany/ManyToOne), Create OneToMany relationship

Open User.java and add the following **field and its getter and setter**

```
@Entity
@Data
public class User {

        private int id;
        private String name;
        private String email;
        private String password;
        private Date created;
        private Date updated;
        private List<Micropost> micropostList;
```

The getter and setter are on next page

Continue on next page

2. Create Microposts

2-3. Create table relation

- A Open User.java and add the following field and **its getter and setter**

```
        private Date updated;
        private List<Micropost> micropostList;

        ……
        ……
        ……

        public void setUpdated(Date updated) {
                this.updated = updated;
        }

        public List<Micropost> getMicropostList() {
                return micropostList;
        }
        public void setMicropostList(List<Micropost> micropostList) {
                this.micropostList = micropostList;
        }
}
```

2. Create Microposts

2-3. Create table relation

- Add @OneToMany annotation over List<Micropost>'s getter

User.java

```
@OneToMany(mappedBy="user", cascade = CascadeType.ALL)
public List<Micropost> getMicropostList() {
        return micropostList;
}
```

User has many microposts, and the micropost has a reference back to the user.

Open Micropost.java and Add user field and its getter and setter

```
@Entity
@Data
public class Micropost {

    private int id;
    private String content;
    private Date created;
    private Date updated;

    private User user;

    public User getUser(){
        return user;
    }
    public void setUser(User user){
        this.user = user;
    }
```

2. Create Microposts

2-3. Create table relation

- Add @ManyToOne annotation over User's getter

Micropost.java

```
@ManyToOne
@JoinColumn(name = "user_id", insertable = false, updatable = false)
public User getUser() {
    return user;
}
```

insertable: whether you want to include name = "user_id" in SQL insert statement.
updatable: whether you want to include name = "user_id" in SQL update statement.

2. Create Microposts

2-3. Create table relation

- Access “user_id” field by getter, add userId field and its getter and setter

Micropost.java

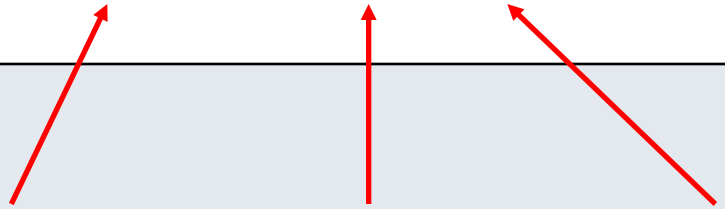
```
@Entity
@Data
public class Micropost {

    private int id;
    private String content;
    private Date created;
    private Date updated;

    private User user;
    private Integer userId;

    @Basic
    @Column(name = "user_id")
    public Integer getUserId() {
        return userId;
    }
    public void setUserId(Integer userId) {
        this.userId = userId;
    }
```

2. Create Microposts

2-4. Create repository

- Create MicropostRepository interface under com.arms.domain.repository

```
package com.arms.domain.repository;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.arms.domain.entity.Micropost;

@Repository
public interface MicropostRepository extends JpaRepository<Micropost, Integer> {

}
```

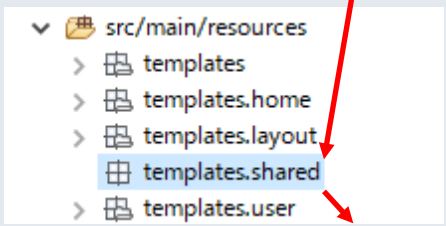


Make sure to extend JpaRepository and set entity type and primary key field type

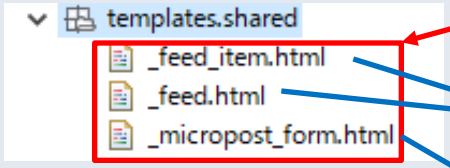
2. Create Microposts

2-5. Create microposts

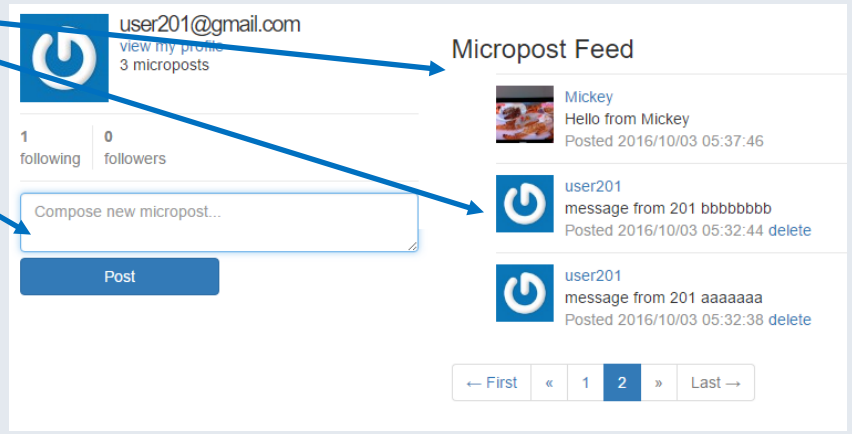
- Create templates.shared



Under templates.shared, create 3 html files,
_feed_item.html _feed.html _micropost_form.html



home.html (isLogin (true))



home.html *includes* _micropost_form.html

_micropost_form.html *includes* _feed.html

_feed includes _feed_item.html

2. Create Microposts

2-5. Create microposts

- Add the following code to home.html

home.html *includes* _micropost_form.html and _feed.html

```
<div th:if="${isLogin}" class="row">
                You have logged in
                <aside class="col-xs-4">
                        <section><div th:include="shared/_micropost_form :: micropost_form">micropost_form</div></section>
                </aside>
                <div class="col-xs-8">
                  <h3>Micropost Feed</h3>
                  <div th:include="shared/_feed :: feed">feed</div>
                </div>
</div>
```

2. Create Microposts

2-5. Create microposts

- Add the following code to _micropost_form.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
>
<head>
   <title>micropost_form</title>
</head>
<body>
        <div th:fragment="micropost_form" th:remove="tag">
                <!-- shared _micropost_form.html included into home.html -->
                <div th:if="${not #strings.isEmpty(message)}" class="alert alert-success alert-dismissible"
role="alert" th:text="${message}">
                <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span aria-
hidden="true">&times;</span></button>
                </div>

                <form th:action="@{/micropost/create}" th:object="${micropostCreateForm}" method="post"
role="form" name="form1">
                        <div class="field">
                                <textarea class="form-control" th:field="*{content}" placeholder="Compose new
micropost..."></textarea>
                        </div>
```

Continue on next page

2. Create Microposts

2-5. Create microposts

- Add the following code to _micropost_form.html

```
                    <input class="btn btn-primary btn-large" type="submit" name="submit" value="Post" />
                </form>

        <br>
                <br>
                <!-- shared _micropost_form.html included into home.html End -->
    </div>
</body>
</html>
```

2. Create Microposts

2-5. Create microposts

- Add the following code to _feed.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
>
<head>
   <title>feed</title>
</head>
<body>
          <div th:fragment="feed" th:remove="tag">
                    <!-- shared _feed.html included into home.html -->
                    <ol class="microposts">
                              <span th:each="micropost, stat : ${microposts}">
                                        <section><div th:include="shared/_feed_item :: feed_item(${micropost})">feed_item</div></section>
                              </span>
                    </ol>

                    <!-- shared _feed.html included into home.html End -->
   </div>
</body>
</html>
```

_feed includes _feed_item.html

2. Create Microposts

2-5. Create microposts

- Add the following code to _feed_item.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
>
<head>
   <title>feed_item</title>
</head>
<body>
          <div th:fragment="feed_item(micropost)" th:remove="tag">
                    <!-- shared _feed_item.html included into _feed.html -->
                    <li th:id="${micropost.id}">
                       <span class="user">
                          <a th:href="@{/user/show/} + ${micropost.user.id}" th:text="${micropost.user.name}">AAA</a>
                       </span>
                       <span class="content" th:text="${micropost.content}">BBB</span>
                       <span class="timestamp" th:text="'Posted ' + ${#dates.format(micropost.created, 'yyyy/MM/dd HH:mm:ss')}">
                             CCC
                       </span>
                             <span th:if="${userId == micropost.userId}">
                                       <a th:href="@{/micropost/delete/} + ${micropost.id}" onclick="return confirm('Are you sure?')">delete</a>
                             </span>
                    </li>
                    <!-- shared _feed_item.html included into _feed.html End -->
   </div>
</body>
</html>
```

2. Create Microposts

2-5. Create microposts

- Restart your app and access ----------:8080/, and login

After you login, you will see the following error

```
java.lang.IllegalStateException: Neither BindingResult nor plain target object for bean name 'micropostCreateForm' available as request attribute
        at org.springframework.web.servlet.support.BindStatus.<init>(BindStatus.java:144) ~[spring-webmvc-4.3.3.RELEASE.jar:4.3.3.RELEASE]
        at org.thymeleaf.spring4.util.FieldUtils.getBindStatusFromParsedExpression(FieldUtils.java:401) ~[thymeleaf-spring4-2.1.5.RELEASE.jar:2.1.5.RE
        at org.thymeleaf.spring4.util.FieldUtils.getBindStatus(FieldUtils.java:328) ~[thymeleaf-spring4-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.spring4.util.FieldUtils.getBindStatus(FieldUtils.java:294) ~[thymeleaf-spring4-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.spring4.processor.attr.AbstractSpringFieldAttrProcessor.processAttribute(AbstractSpringFieldAttrProcessor.java:98) ~[thymelea
        at org.thymeleaf.processor.attr.AbstractAttrProcessor.doProcess(AbstractAttrProcessor.java:87) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.processor.AbstractProcessor.process(AbstractProcessor.java:212) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.dom.Node.applyNextProcessor(Node.java:1017) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.dom.Node.processNode(Node.java:972) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.dom.NestableNode.computeNextChild(NestableNode.java:695) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.dom.NestableNode.doAdditionalProcess(NestableNode.java:668) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
        at org.thymeleaf.dom.Node.processNode(Node.java:990) ~[thymeleaf-2.1.5.RELEASE.jar:2.1.5.RELEASE]
```

<form th:action="@{/micropost/create}" th:object="${micropostCreateForm}" method="post"

MicropostCreateForm object to receive data from form

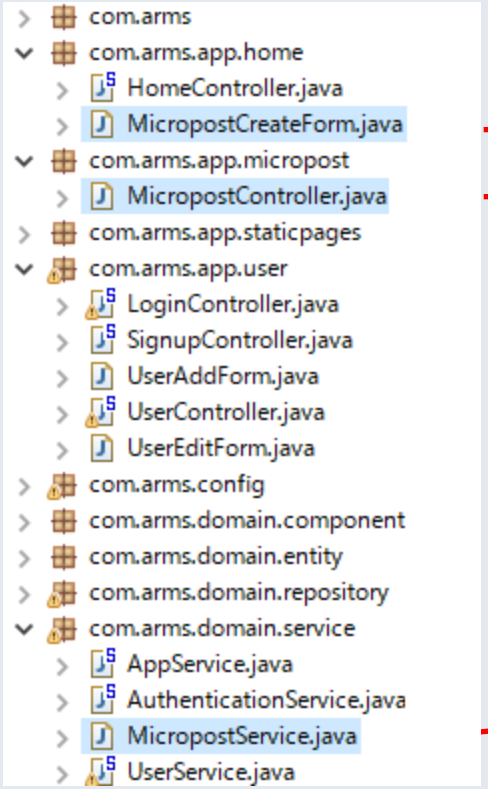
MicropostController to handle request from View

MicropostService to handle request between Controller and Repository

2. Create Microposts

2-5. Create microposts

- Create the following three java



MicropostCreateForm

MicropostController

MicropostService

2. Create Microposts

2-5. Create microposts

- Add the following code to MicropostCreateForm

```
package com.arms.app.home;

import lombok.Data;

@Data
public class MicropostCreateForm {

    private String content;

    private int userId;
}
```

2. Create Microposts

2-5. Create microposts

- Open AppService and add @Autowired to MicropostRepository

```
@Service
public class AppService {

    @Autowired
    UserRepository userRepository;

    @Autowired
    MicropostRepository micropostRepository;

    public Integer getUserId(Principal principal) {
……
……..…..;
```

2. Create Microposts

2-5. Create microposts

- Add the following code to MicropostService

```
package com.arms.domain.service;

import java.util.Calendar;
import java.util.Date;
import org.springframework.stereotype.Service;
import com.arms.app.home.MicropostCreateForm;
import com.arms.domain.entity.Micropost;

@Service
public class MicropostService extends AppService {

	public int createMicropost(MicropostCreateForm micropostCreateForm){
		Date nowDate = Calendar.getInstance().getTime();
		Micropost micropost = new Micropost();
		micropost.setContent(micropostCreateForm.getContent());
		micropost.setUserId(micropostCreateForm.getUserId());
		micropost.setCreated(nowDate);
		micropost.setUpdated(nowDate);
		micropostRepository.save(micropost);
		return micropost.getId();
	}
}
```

Make sure to extend AppService

2. Create Microposts

2-5. Create microposts

- Add the following code to MicropostController

```
package com.arms.app.micropost;

import java.security.Principal;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.servlet.mvc.support.RedirectAttributes;
import com.arms.app.home.MicropostCreateForm;
import com.arms.domain.component.ControllerAspect;
import com.arms.domain.service.MicropostService;

@Controller
public class MicropostController {

    @Autowired
    ControllerAspect controllerAspect;

        @Autowired
        MicropostService micropostService;
```

Continue on next page

2. Create Microposts

2-5. Create microposts

- Add the following code to MicropostController

```
        @RequestMapping("/micropost/create")
        public String create(@ModelAttribute MicropostCreateForm micropostCreateForm, Principal principal, RedirectAttributes
redirectAttributes){
                Integer userId = micropostService.getUserId(principal);
                micropostCreateForm.setUserId(userId);
                redirectAttributes.addFlashAttribute("message", "Feed was created");
                micropostService.createMicropost(micropostCreateForm);
                return "redirect:/";
        }
}
```

2. Create Microposts

2-5. Create microposts

- Add the following code to HomeController

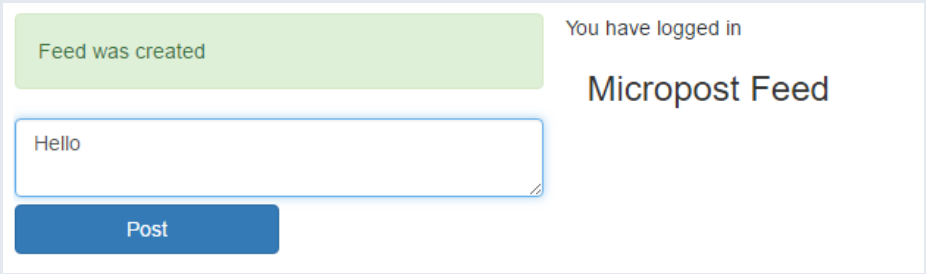
```
@Controller
public class HomeController {

   @Autowired
   ControllerAspect controllerAspect;

   @ModelAttribute
   MicropostCreateForm setMicropostCreateForm() {
         return new MicropostCreateForm();
   }
……..
……..
```

Note: Adding @ModelAttribute at method level
Initialize Form object and return it.
This @ModelAttribute method is called before methods with @RequestMapping (Regardless of the form object existence in the methods)
This means that the form object is automatically added to the model which thymeleaf can refer to. The default attribute name comes from the class name but the first letter is small. MicropostCreateForm (class name) → micropostCreateForm(first letter is small)

2. Create Microposts

2-5. Create microposts

- Restart your app and access ----------:8080/, and see if you can create a micropost



After you create a few microposts, go to phpMyAdmin and see micropost table.

user table

id	name	email	password
8	user106	user106@gmail.com	1d0258c2440a8d19e
9	user107	user107@gmail.com	1d0258c2440a8d19e

micropost table

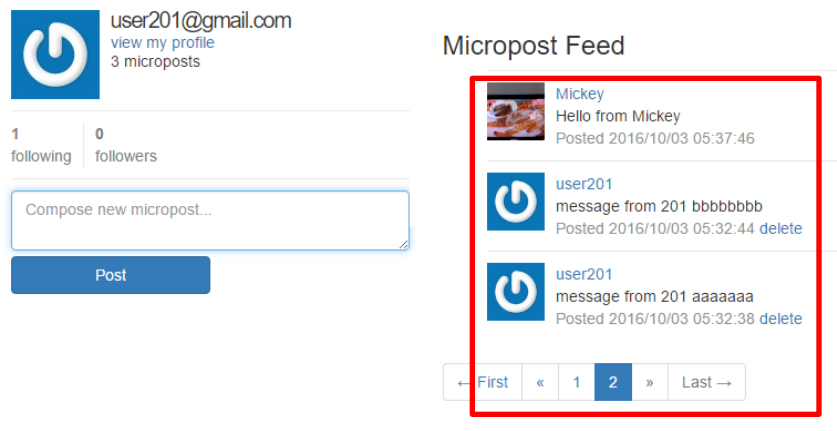
id	content	user_id	created
5	Hello	8	2016-11
6	How are you doing?	8	2016-11

You can see microposts with user_id as the foreign key.

2. Create Microposts

2-6. Create micropost feed

- In this section, we will create a list of microposts



2. Create Microposts

2-6. Create micropost feed

- Open MicropostRepository and add three query methods to it.

```
@Repository
public interface MicropostRepository extends JpaRepository<Micropost, Integer> {

    List<Micropost> findAllByUserId(int userId);
    Page<Micropost> findAllByUserIdOrderByUpdatedDesc(int userId, Pageable pageable);
    Page<Micropost> findByIdInOrderByUpdatedDesc(Collection id, Pageable pageable);
}
```

Find all micropost by userId

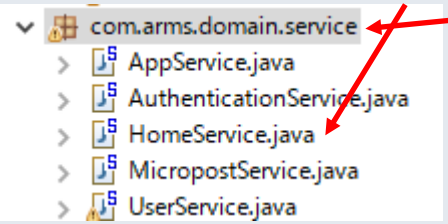
Find all microposts by userId and by page (Descending order by updateddate)

Find all microposts by Id List and by page (Descending order by updateddate)

2. Create Microposts

2-6. Create micropost feed

- Create HomeService under com.arms.domain.service



Add the following code to HomeService

```
package com.arms.domain.service;

import java.util.ArrayList;
import java.util.List;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;

import com.arms.domain.entity.Micropost;

@Service
public class HomeService extends AppService {
```

Make sure to extend AppService

Continue on next page

2. Create Microposts

2-6. Create micropost feed

- Add the following code to HomeService

```
        public List<Integer> getMyMicropost(int userId) {
        List<Micropost> micropostList = micropostRepository.findAllByUserId(userId);
        List<Integer> micropostIdList = new ArrayList<>();
        for(Micropost micropost : micropostList) {
            micropostIdList.add(micropost.getId());
        }
        return micropostIdList;
    }

    public Page<Micropost> findAllByIdList(List<Integer> micropostIdList, Pageable pageable) {
        return micropostRepository.findByIdInOrderByUpdatedDesc(micropostIdList, pageable);
    }
}
```

After you create following and follower function, the following users' feeds are also displayed

2. Create Microposts

2-6. Create micropost feed

- Modify HomeController as below,

```
@Controller
public class HomeController {

    @Autowired
    ControllerAspect controllerAspect;

    @Autowired
    HomeService homeService;

    @ModelAttribute
    MicropostCreateForm setMicropostCreateForm() {
        return new MicropostCreateForm();
    }

        @RequestMapping("/")
        public ModelAndView home(ModelAndView modelAndView, Principal principal, Pageable pageable){
            Integer userId = homeService.getUserId(principal);
```

Continue on next page

2. Create Microposts

2-6. Create micropost feed

- Modify HomeController as below,

```
                if(userId != null){
                        List<Integer> micropostIdList = homeService.getMyMicropost(userId);

                        Page<Micropost> micropostPage = homeService.findAllByIdList(micropostIdList, pageable);
                        PageWrapper<Micropost> page = new PageWrapper<>(micropostPage, "/");
                        modelAndView.addObject("microposts", page.getContent());
                        modelAndView.addObject("page", page);
                }
                modelAndView.setViewName("home/home");
                return modelAndView;
        }
}
```

2. Create Microposts

2-6. Create micropost feed

- Add the pagination code to _feed.html

```
                    <section><div th:include="shared/_feed_item :: feed_item(${micropost})">feed_item</div></section>
                </span>
            </ol>
            ……..add from here!!
            <!-- Start Of Pagination Bar -->
            <div th:fragment='paginationbar'>
                <div>
                    <ul class='pagination pagination-centered'>
                        <li th:class="${page.firstPage}?'disabled':''">
                            <span th:if='${page.firstPage}'>← First</span>
                            <a th:if='${not page.firstPage}' th:href='@{${page.url}(page=0,size=${page.size})}'>←
First</a>
                        </li>
                        <li th:class="${page.hasPreviousPage}? '' : 'disabled'">
                            <span th:if='${not page.hasPreviousPage}'>«</span>
                            <a th:if='${page.hasPreviousPage}' th:href='@{${page.url}(page=${page.number-
2},size=${page.size})}' title='Go to previous page'>«</a>
                        </li>
                        <li th:each='item : ${page.items}' th:class="${item.current}? 'active' : ''">
                            <span th:if='${item.current}' th:text='${item.number}'>1</span>
                            <a th:if='${not item.current}' th:href='@{${page.url}(page=${item.number-
1},size=${page.size})}'><span th:text='${item.number}'>1</span></a>
                        </li>
```

2. Create Microposts

2-6. Create micropost feed

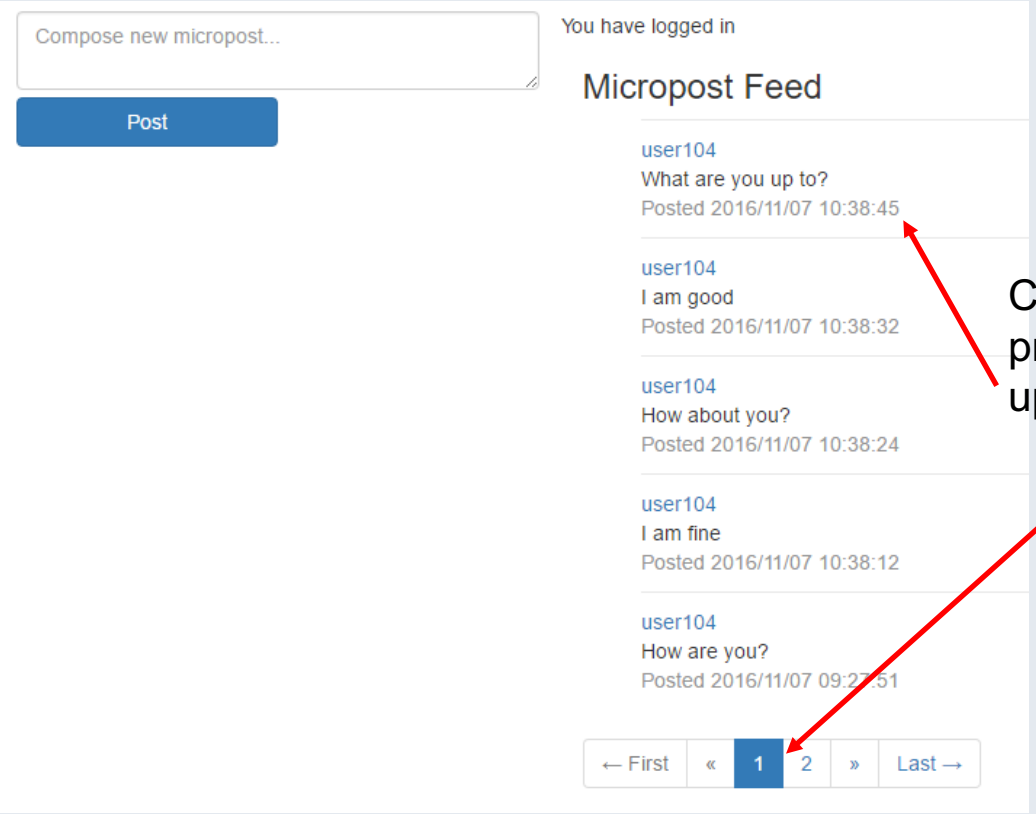
- Add the pagination code to _feed.html

```
                                        <li th:class="${page.hasNextPage}? '' : 'disabled'">
                                                <span th:if='${not page.hasNextPage}'>»</span>
                                                <a th:if='${page.hasNextPage}'
th:href='@{${page.url}(page=${page.number},size=${page.size})}' title='Go to next page'>»</a>
                                        </li>
                                        <li th:class="${page.lastPage}? 'disabled' : ''">
                                                <span th:if='${page.lastPage}'>Last →</span>
                                                <a th:if='${not page.lastPage}' th:href='@{${page.url}(page=${page.totalPages -
1},size=${page.size})}'>Last →</a>
                                        </li>
                                </ul>
                        </div>
                </div>
                <!-- End Of Pagination Bar -->
```

2. Create Microposts

2-6. Create micropost feed

- Restart your app and access ----------:8080/, and then login and add more than 6 microposts.

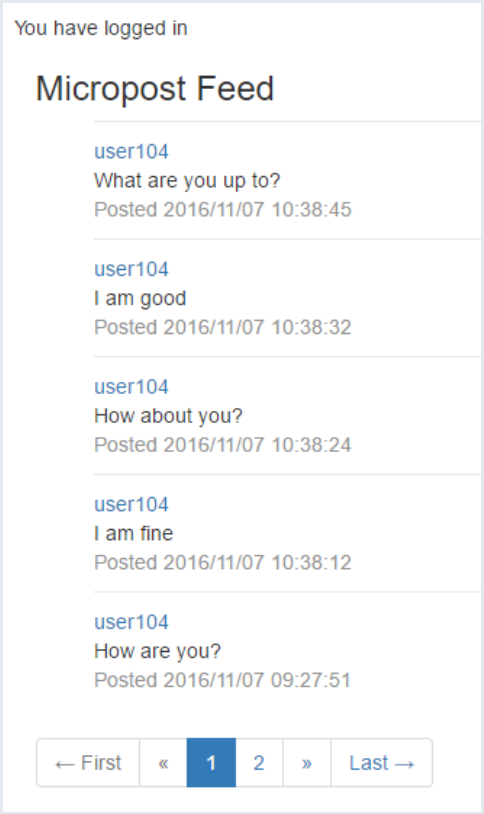


Check if pagination is working properly, and your post is in updatedDate order.

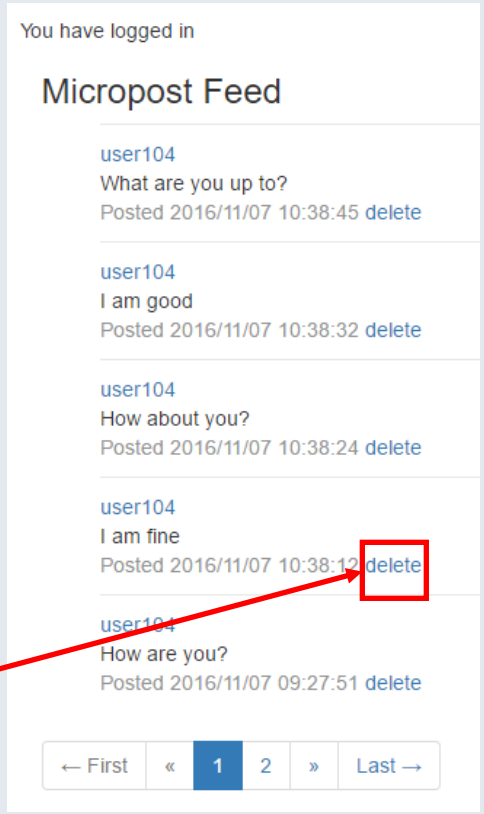
2. Create Microposts

2-7. Delete Micropost

- Add delete function to Micropost



Micropost should only be deleted by its created user
_feed_item.html




```
<span th:if="${userId == micropost.userId}">
 <a th:href="@{/micropost/delete/} + ${micropost.id}" onclick="return confirm('Are you sure?')">delete</a>
</span>
```

2. Create Microposts

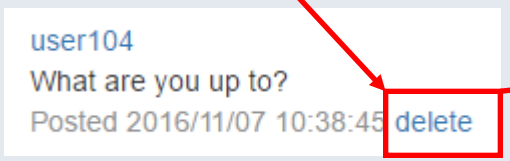
2-7. Delete Micropost

- Add the following code to HomeController and give userId parameter to the View

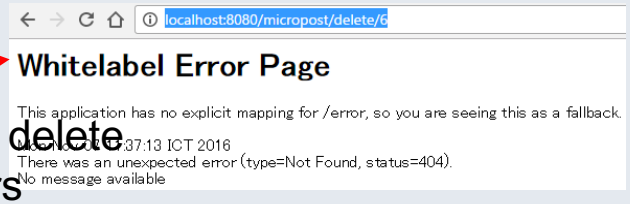
```
@RequestMapping("/")
public ModelAndView home(ModelAndView modelAndView, Principal principal, Pageable pageable){
        Integer userId = homeService.getUserId(principal);

        if(userId != null){
                modelAndView.addObject("userId", userId);
                List<Integer> micropostIdList = homeService.getMyMicropost(userId);
……
……
……
```

Get userId from principal and give its value as “userId” to the View

Restart your app and access ----------:8080/, and see if the delete button is displayed.



But if you click the delete button, error occurs



2. Create Microposts

2-7. Delete Micropost

- Add the following code to MicropostService

MicropostService

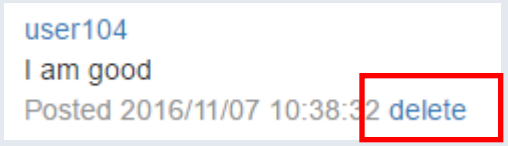
```
public void deleteMicropost(int micropostId){
        micropostRepository.delete(micropostId);
}
```

Call the above method from MicropostController

MicropostController

```
@RequestMapping("/micropost/delete/{id}")
  public String delete(@PathVariable int id, RedirectAttributes redirectAttributes) {
     micropostService.deleteMicropost(id);
     redirectAttributes.addFlashAttribute("message", "Feed was deleted");
     return "redirect:/";
  }
```

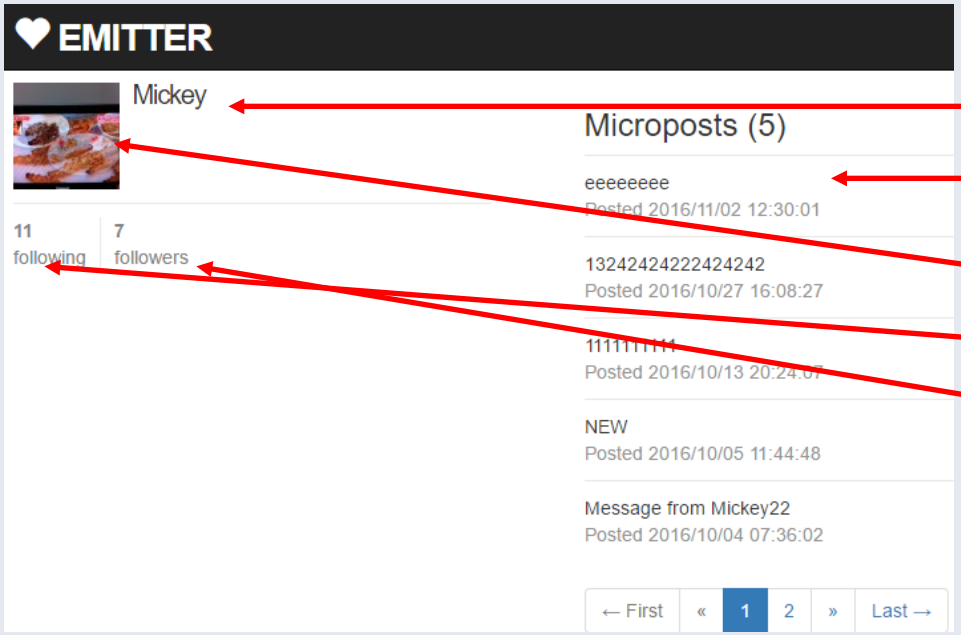
Restart your app and access --------:8080/, and see if you can delete microposts by delete button.



3. Create User Profile

3-1. Summary

- In this section, we will create the user profile page as below,



User name

List of microposts for the user

Avatar (not in this section)

Following (not in this section)

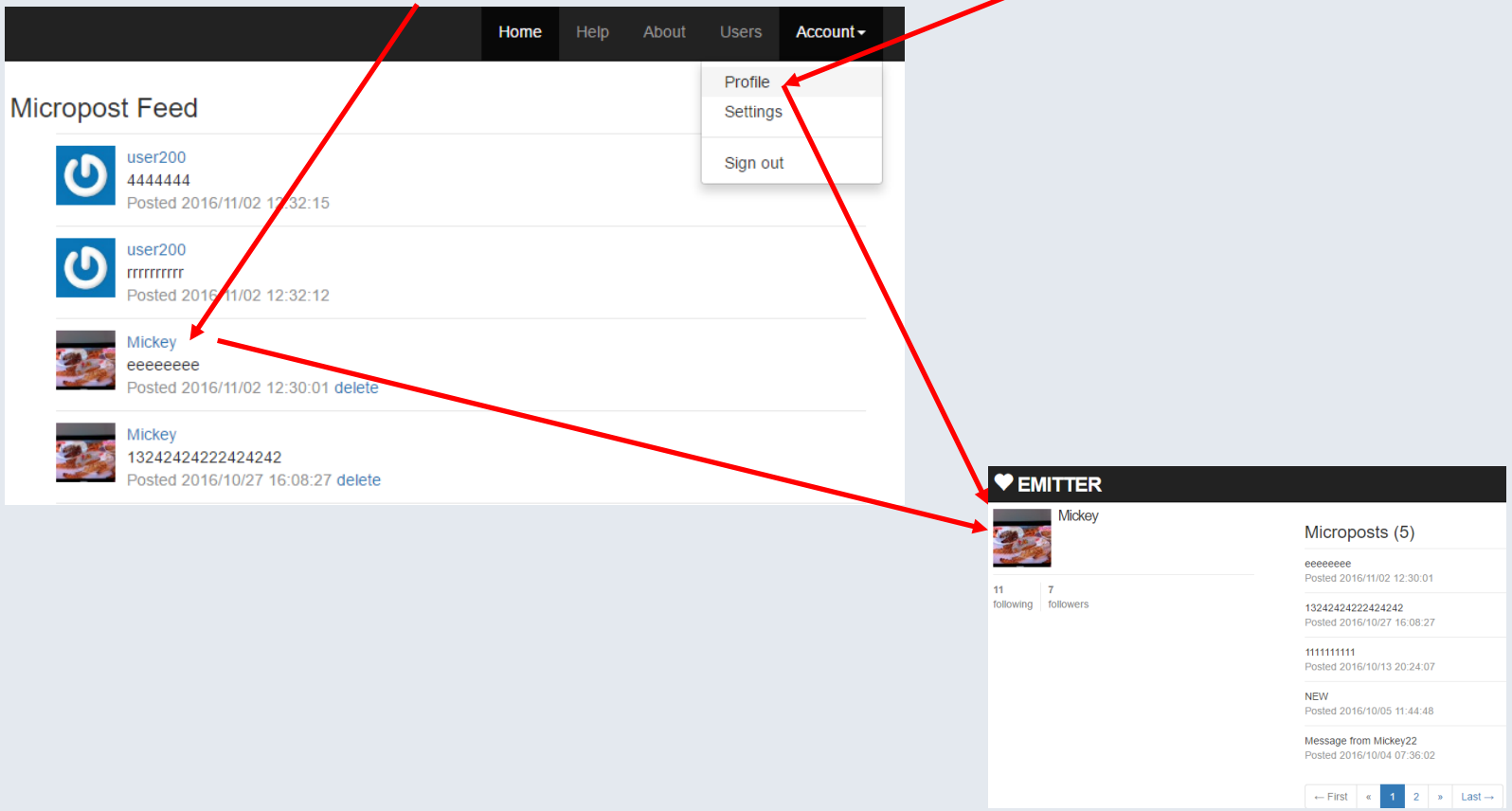
Followers (not in this section)

In this section, we only display user name and a list of microposts for the user

3. Create User Profile

3-1. Summary

- You can jump to user profile either from Account menu Profile or from username on Micropost Feed



3. Create User Profile

3-2. Micropost for each user

- Add the following method to UserService

UserService

```
public Page<Micropost> findAllMicropostByUserId(int userId, Pageable pageable) {
    return micropostRepository.findAllByUserIdOrderByUpdatedDesc(userId, pageable);
  }
```

Call the above method from UserController

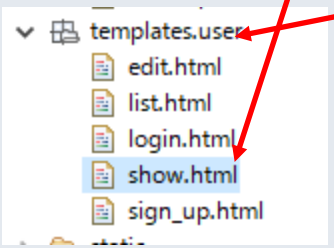
UserController

```
@RequestMapping("/user/show/{userId}")
  public ModelAndView show(@PathVariable int userId, ModelAndView modelAndView, Pageable pageable,
Principal principal){
      User user = userService.findOne(userId);
    modelAndView.addObject("user", user);
    Page<Micropost> micropostPage = userService.findAllMicropostByUserId(userId, pageable);
    PageWrapper<Micropost> page = new PageWrapper<>(micropostPage, "/user/show" + '/' + userId);
    modelAndView.addObject("microposts", page.getContent());
    modelAndView.addObject("page", page);
    modelAndView.setViewName("user/show");
    return modelAndView;
  }
```

3. Create User Profile

3-2. Micropost for each user

- Create show.html under templates.user



Add the following code to show.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"
    xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
    layout:decorator="layout/main">
<head>
        <title>Users Details</title>
</head>
<body>
        <div layout:fragment="content" th:remove="tag">
                <!-- Users show.html inserted into main.html -->
                <div class="row">
                  <aside class="col-xs-4">
                    <section>
                      <h1>
                        <span th:text="${user.name}">AAA</span>
                      </h1>
                    </section>
```

Continue on next page

3. Create User Profile

3-2. Micropost for each user

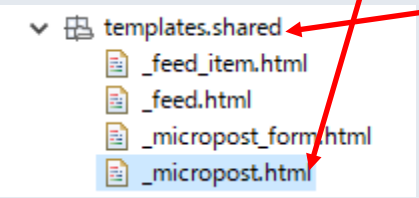
- Add the following code to show.html

```
                    </aside>
                    <div class="col-xs-8">

                                <span th:if="${microposts.size() > 0}">
                                    <ol>
                                        <h3 th:text="'Microposts (' + ${microposts.size()} + ')'">CCC</h3>
                                    </ol>
                            <ol class="microposts" th:include="shared/_micropost :: micropost">micropost</ol>
                                </span>
                    </div>

                </div>
                <a class="btn btn-primary btn-small" href="/">
                    Back to Home
                </a>
                <!-- Users show.html inserted into main.html End -->
            </div>
</body>
</html>
```

3. Create User Profile

3-2. Micropost for each user

- Create _micropost.html under templates.shared



Add the following code to _micropost.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:th="http://www.thymeleaf.org"
>
<head>
   <title >micropost</title>
</head>
<body>
          <div th:fragment="micropost" th:remove="tag">
                    <!-- shared _micropost.html included into show.html -->
                    <span th:each="micropost, stat : ${microposts}">
                              <li>
                                        <span class="content" th:text="${micropost.content}">CCC</span>
                                        <span class="timestamp" th:text="'Posted ' + ${#dates.format(micropost.created, 'yyyy/MM/dd
HH:mm:ss')}">AAA</span>
                              </li>
                    </span>
```

Continue on next page

3. Create User Profile

3-2. Micropost for each user

- Add the following code to _micropost.html

```
<!-- Start Of Pagination Bar -->
<div th:fragment='paginationbar'>
    <div>
        <ul class='pagination pagination-centered'>
            <li th:class="${page.firstPage}?'disabled':''">
                <span th:if='${page.firstPage}'>← First</span>
                <a th:if='${not page.firstPage}'
th:href='@{${page.url}(page=0,size=${page.size})}'>← First</a>
            </li>
            <li th:class="${page.hasPreviousPage}? '' : 'disabled'">
                <span th:if='${not page.hasPreviousPage}'>«</span>
                <a th:if='${page.hasPreviousPage}' th:href='@{${page.url}(page=${page.number-
2},size=${page.size})}' title='Go to previous page'>«</a>
            </li>
            <li th:each='item : ${page.items}' th:class="${item.current}? 'active' : ''">
                <span th:if='${item.current}' th:text='${item.number}'>1</span>
                <a th:if='${not item.current}' th:href='@{${page.url}(page=${item.number-
1},size=${page.size})}'><span th:text='${item.number}'>1</span></a>
            </li>
            <li th:class="${page.hasNextPage}? '' : 'disabled'">
                <span th:if='${not page.hasNextPage}'>»</span>
                <a th:if='${page.hasNextPage}'
th:href='@{${page.url}(page=${page.number},size=${page.size})}' title='Go to next page'>»</a>
            </li>
```

Continue on next page

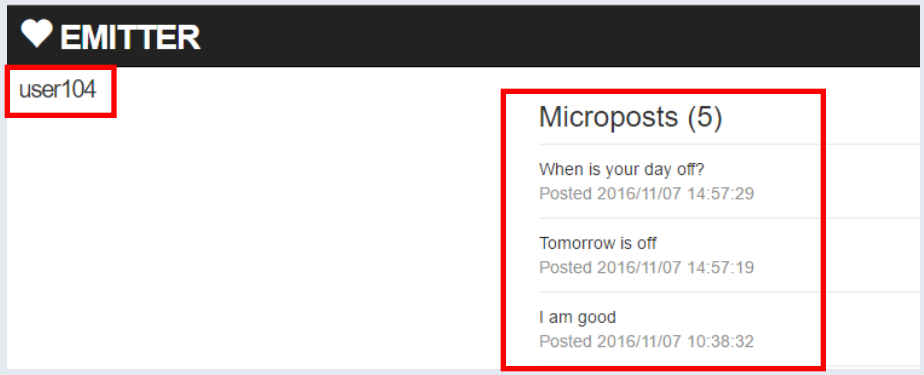
3. Create User Profile

3-2. Micropost for each user

- Add the following code to _micropost.html

```
                                        <li th:class="${page.lastPage}? 'disabled' : ''">
                                            <span th:if='${page.lastPage}'>Last →</span>
                                            <a th:if='${not page.lastPage}' th:href='@{${page.url}(page=${page.totalPages -
1},size=${page.size})}'>Last →</a>
                                        </li>
                                    </ul>
                                </div>
                            </div>
                            <!-- End Of Pagination Bar -->
                            <!-- shared _micropost.html included into show.html End -->
    </div>
</body>
</html>
```

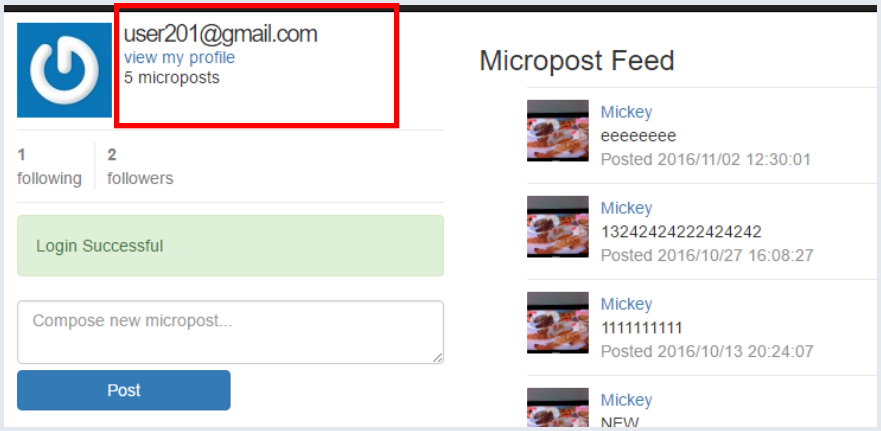
Restart your app and access --------:8080/, and see if you can see the page like below,



3. Create User Profile

3-3. Create user info

- In this section, we will create user info on home.html



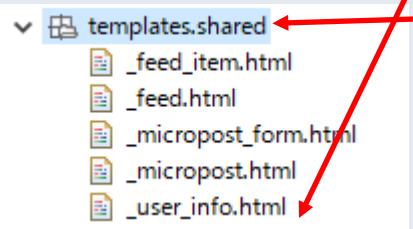
Add the following code to home.html, and delete “You have logged in” message

```
……
<div th:if="${isLogin}" class="row">
        You have logged in
        <aside class="col-xs-4">
          <section><div th:include="shared/_user_info :: user_info">user_info</div></section>
          <section><div th:include="shared/_micropost_form :: micropost_form">micropost_form</div></section>
        </aside>
……
```

3. Create User Profile

3-3. Create user info

- Create _user_info.html under templates.shared



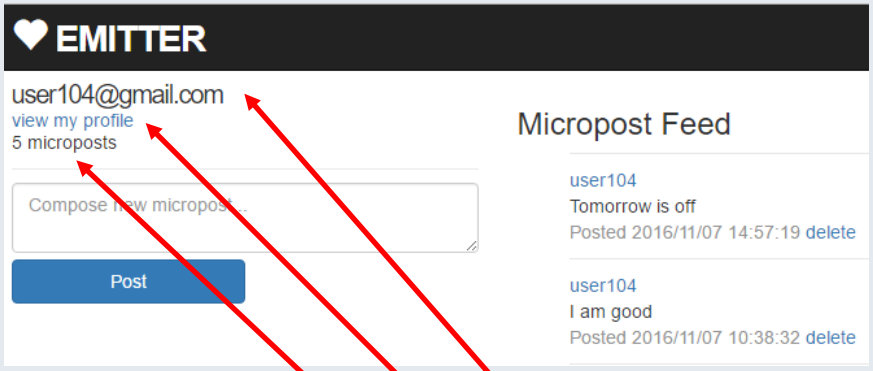
Add the following code to _user_info.html

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
    xmlns:th="http://www.thymeleaf.org"
>
<head>
  <title>User Info</title>
</head>
<body>
        <div th:fragment="user_info" th:remove="tag">
                <!-- shared _user_info.html included into home.html -->
                <h1 th:text="${userInfo.username}">AAA</h1>
                <span>
                        <a th:href="@{/user/show/} + ${userId}">view my profile</a>
                </span>
                <span th:text="${microposts.size()} + ' microposts'">
                        AAA
                </span>
                <!-- shared _user_info.html included into home.html End -->
  </div>
</body>
</html>
```

3. Create User Profile

3-3. Create user info

- Restart your app and access -------:8080/, and check if you can see user info



_user_info.html

```
<h1 th:text="${userInfo.username}">AAA</h1>
<span>
   <a th:href="@{/user/show/} + ${userId}">view my profile</a>
</span>
<span th:text="${microposts.size()} + ' microposts'">
   AAA
</span>
```

Your Idea Leads Your Ideals

homepage: http://arms-asia.com/
facebook: https://www.facebook.com/arms.asia?fref=ts