Note: the additional code for the project below is available through the shared project (it has been merged into it) at https://*your_username*@bitbucket.org/comp80052015/test-repo.git

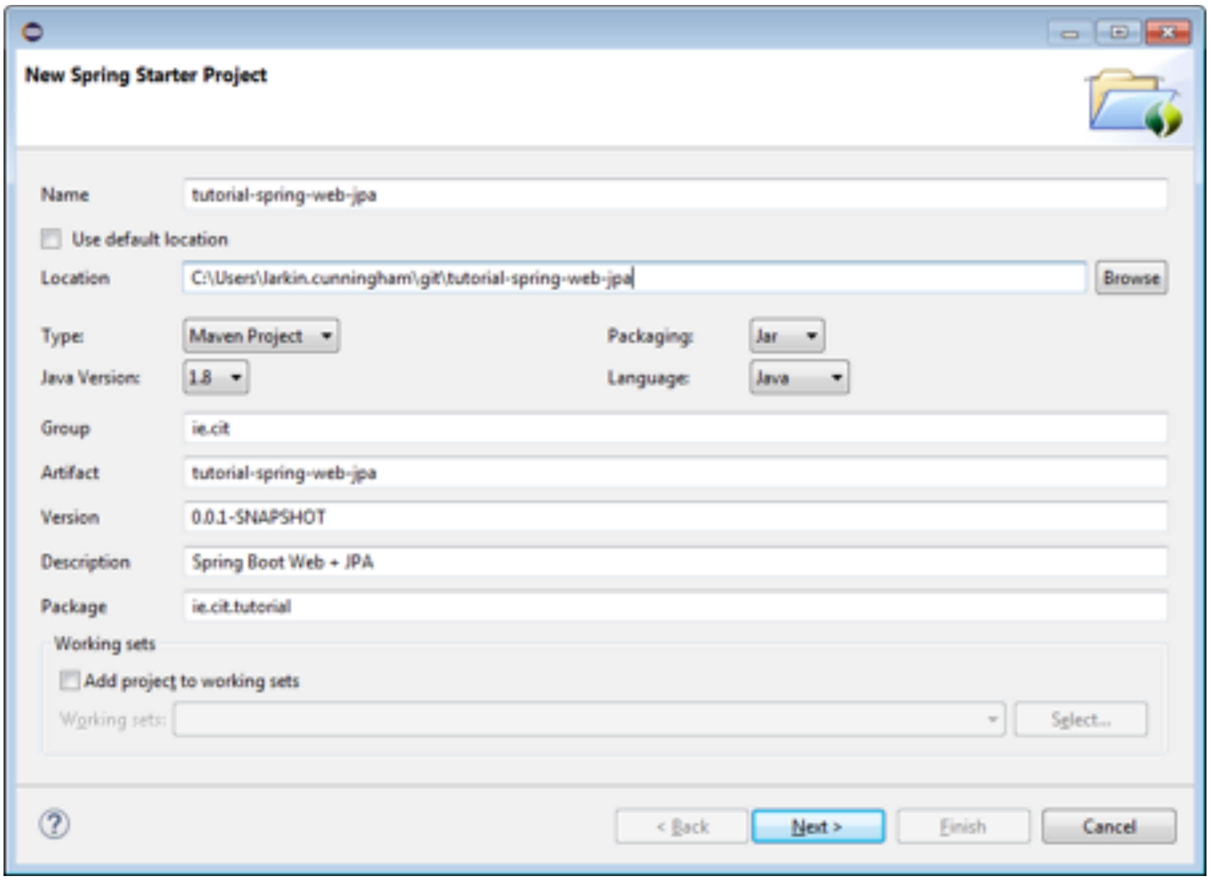
(Change your_username above to your own BitBucket username)

Part 1 – Create a new private repository

1. Use BitBucket to create a new git repository. It is ok to leave it public. Call it "tutorial-spring-web-jpa" and leave the other options at their default values.

Part 2 – Create a Spring Boot Project

1. Select File⌧New⌧Spring Starter Project and enter the following (use your own repository location within your local git repository folder). Then click Next.



New Spring Starter Project

Name: tutorial-spring-web-jpa
Use default location: ☐
Location: C:\Users\Jarkin.cunningham\git\tutorial-spring-web-jpa
Type: Maven Project
Packaging: Jar
Java Version: 1.8
Language: Java
Group: ie.cit
Artifact: tutorial-spring-web-jpa
Version: 0.0.1-SNAPSHOT
Description: Spring Boot Web + JPA
Package: ie.cit.tutorial
Working sets
Add project to working sets: ☐

2. You are now faced with a wide choice of dependencies (libraries or sets of libraries) to add to the project. Select the following:
   Data: JPA
   Database: H2, MySQL
   Template Engines: Thymeleaf
   Web: Web

3. Click Finish.

Part 3 – Set up the local repository for the project

1. Add a new text file to the project called .gitignore. This will allow us to only commit non-Eclipse files. When cloning to other computers, it is best to avoid having Eclipse files being cloned from another computer. Add the following to it:

   ```
   bin
   .metadata
   .settings
   target
   ```

2. Using the command prompt, CD into your project folder.

3. Initialise it as a git repository:
   ```
   git init
   ```

4. Set the remote origin to your BitBucket repository:

   ```
   git remote add origin https://your_username@bitbucket.org/
   your_username/tutorial-spring-web-jpa.git
   ```

5. Add your files to the local repository, then commit and push:
   ```
   git add —A
   git status
   git commit —a —m "Adding new project"
   git push origin master
   ```

6. View your project source at:
   https://bitbucket.org/*your_username*/tutorial-spring-web-jpa/src

7. Note the absence of the bin, target and .settings folders.

Part 4 – Add Components to the project

Controller

1. Using the previous shared tutorial project as a template (such as adding similar packages), add a controller called MessageController that handles the routes "/message" and "/message/{key}".
2. The "/message" URL will list all messages. The "/message/{key}" URL will display details for a single message according to the key parameter supplied in the URL.

3. This will require 2 methods, one with a RequestMapping annotation for “/message” and another for “/message/{key}” – give these methods the names *index* and view*ByKey*.

4. Full Code:

```
@Controller
@RequestMapping("/message")
public class MessageController {

      @Autowired
      MessageRepository messageRepository;

      @RequestMapping("")
      public String index(Model model) {
            Iterable<Message> messages = messageRepository.findAll();
            model.addAttribute("messages", messages);
            return "message/index";
      }

      @RequestMapping("/{key}")
      public String viewByKey(@PathVariable String key, Model model)
{
            Message message =
messageRepository.findByMessageKey(key);
            model.addAttribute("message", message);
            return "message/view";
      }
}
```

Repository and Settings

1. Copy the MessageRepository and Message class from the shared project to your new project.

2. Copy the .yml and .sql files from the shared project. This will copy all of the personalised message SQL statements.

Views

1. Under src/main/resources/templates, add a new folder message.

2. Within the new folder add the file index.html to list all messages:

```
<!DOCTYPE HTML>
      <html xmlns:th="http://www.thymeleaf.org">
      <head>
          <title>Messages</title>
          <meta http-equiv="Content-Type" content="text/html;
      charset=UTF-8" />
      </head>
      <body>
      <table>
            <tr>
              <th>KEY</th>
            <th>MESSAGE</th>
        </tr>
        <tr th:each="message: ${messages}">
              <td><a th:href="@{|/message/${message.messageKey}|}" th:text="${message.messageKey}">Key</a></td>
              <td th:text="${message.messageText}">Message</td>
          </tr>
</table>
</body>
</html>
```

```
Note 1 Loop through each message in messages

Note 2 Add a link to each message to view more detail (if there
were any). The bars in th:href allow for an expression within
an expression to be evaluated (i.e. the ${message.messageKey}
within the @{ section of code).
```

3. Also add the file view.html:

```
<!DOCTYPE HTML>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title th:text="${message.messageKey}"></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
</head>
<body>
<p th:text="${message.messageText}">Message Text</p>
</body>
</html>
```

Now… run the project and go to http://localhost:8080/message. This should list all messages and allow you to link to a view page for each message.

Additional work: You could consider extending the message table and class to add more columns, then try to display them on the view web template.

When you have the project working, add files to your local repository, check them into your local repository and push to your remote repository.