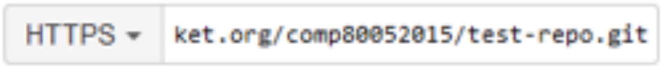
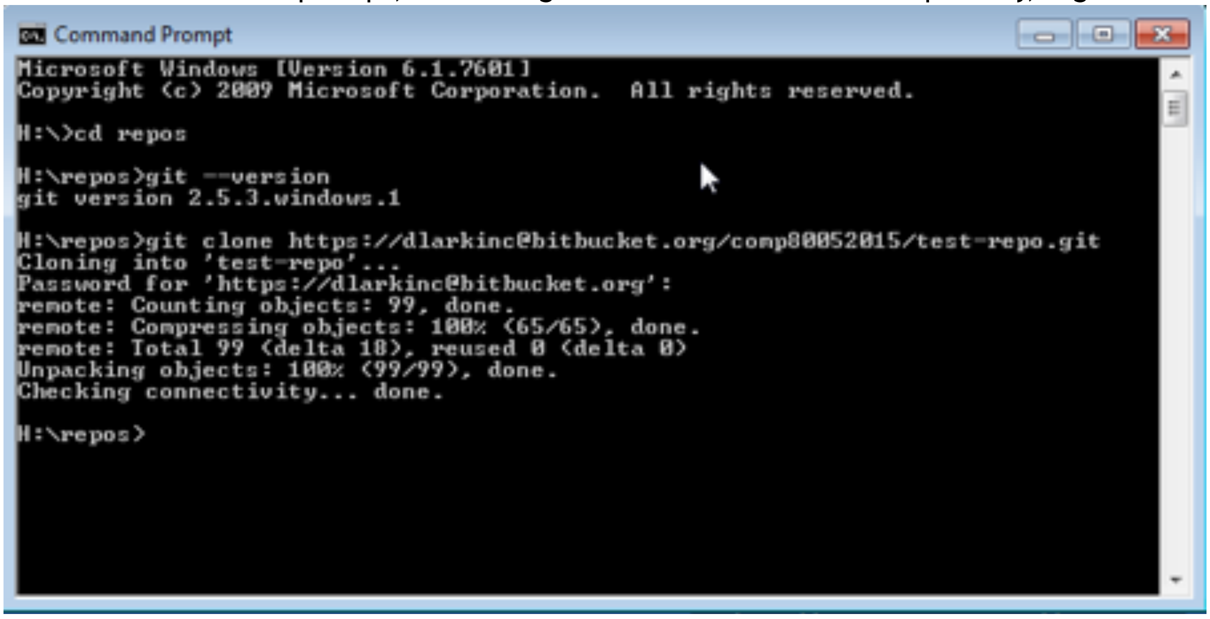
Part 1 – Create your local repository, clone the remote repository and get the project running in Eclipse / STS

1. Using the command prompt, change to your git repositories directory (should be on your network drive / H: ; if not, create one called something like git or repos)
2. Using the browser, log in to your BitBucket account and look for the test-repo on your dashboard; click into it
3. Look for the repository address and copy it to the clipboard, e.g.

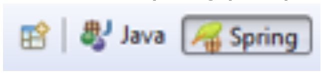


4. Back at the command prompt, issue the git clone command for the repository, e.g.



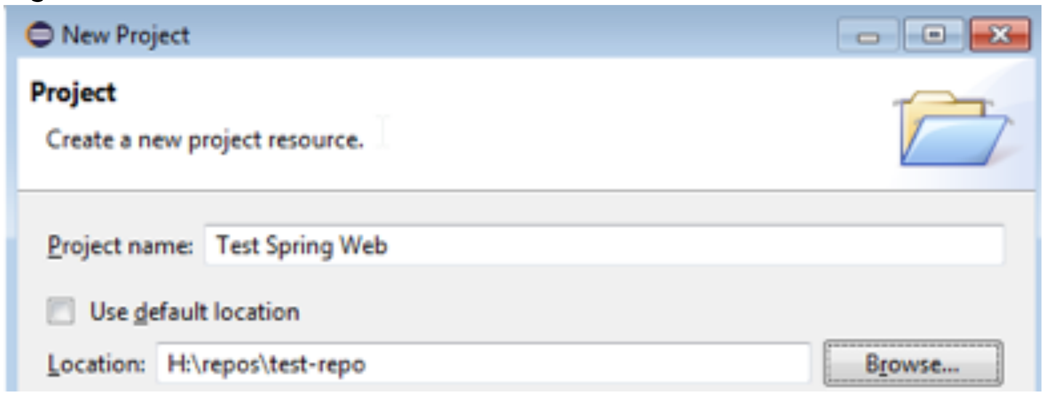
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

H:\>cd repos

H:\repos>git --version
git version 2.5.3.windows.1

H:\repos>git clone https://dlarkinc@bitbucket.org/comp80052015/test-repo.git
Cloning into 'test-repo'...
Password for 'https://dlarkinc@bitbucket.org':
remote: Counting objects: 99, done.
remote: Compressing objects: 100% (65/65), done.
remote: Total 99 (delta 18), reused 0 (delta 0)
Unpacking objects: 100% (99/99), done.
Checking connectivity... done.

H:\repos>
```

5. Open Eclipse.
6. Select the Spring perspective



7. Select File New Project…
8. Select General Project
9. Click Next

10. Name the project “Test Spring Web” and select the location of your git repository, e.g.

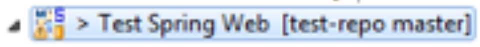


Project name: Test Spring Web

Use default location: unchecked

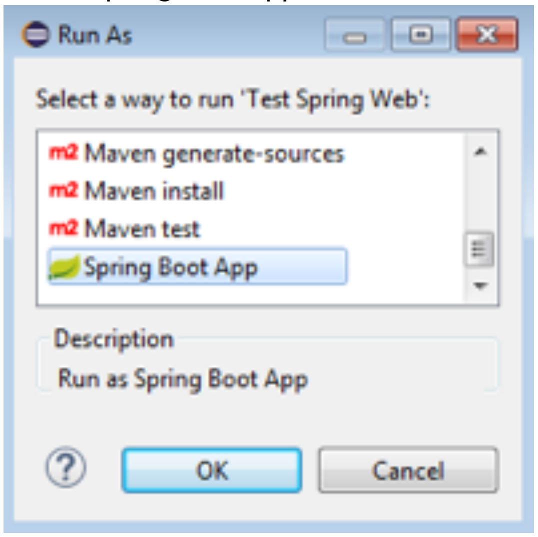
Location: H:\repos\test-repo

11. Click Finish
12. You may need to right-click on the project and select Configure ⌧ Convert to Maven Project
13. If it does not automatically add the Spring Project Nature, then right-click on the project and select Spring Tools ⌧ Add Spring Project Nature
14. You should see the following project in Eclipse (with letters M and S):



> Test Spring Web [test-repo master]

15. The project (barring issues with your JDK version) should run as is. Click the run button (or right-click on the project and select Run As ⌧ Spring Boot App) and select Spring Boot App:



16. A large amount of log output should be seen in the console window, ending with something like:

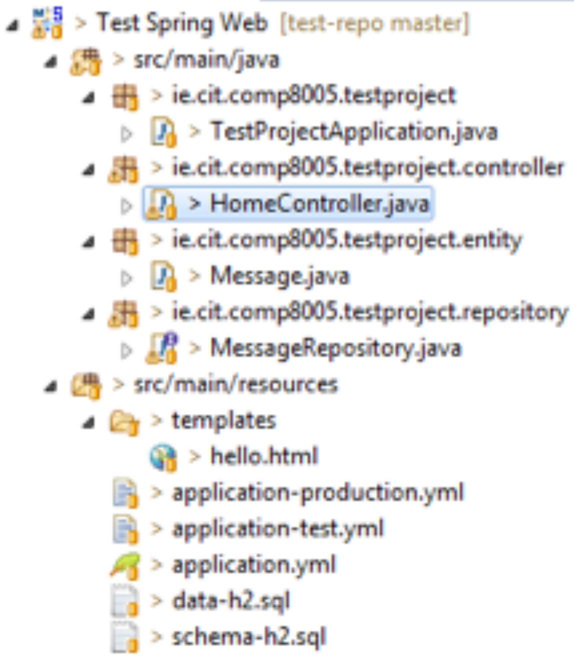
```
2015-09-21 15:39:48.523  INFO 11368 --- [           main]
i.c.c.t.TestProjectApplication            : Started
TestProjectApplication in 21.995 seconds (JVM running for 24.023)
```

17. Now you can access this basic web application at the address: http://localhost:8080/hello
18. You should see the message: “Hi there!”

19. Note: If you see a "port already in use" error at any point, you can click the following button to destroy the running application:

Part 2 - Adding your personalised web page and hello message

1. Examine the directory structure of the project, in particular what is shown here:



```
> Test Spring Web [test-repo master]
  > src/main/java
    > ie.cit.comp8005.testproject
      > TestProjectApplication.java
    > ie.cit.comp8005.testproject.controller
      > HomeController.java
    > ie.cit.comp8005.testproject.entity
      > Message.java
    > ie.cit.comp8005.testproject.repository
      > MessageRepository.java
  > src/main/resources
    > templates
      > hello.html
    > application-production.yml
    > application-test.yml
    > application.yml
    > data-h2.sql
    > schema-h2.sql
```

2. We will make a change to the HomeController.java class file to add a personalised route. Currently we just have the route /hello, but you will add one for your name in the format:
   /hello/from/lcunningham
   Just change the name to your own first initial and full last name.

3. Open the HomeController.java file.

4. Copy and paste the following code just below the hello method:

```
@RequestMapping("hello")
      public String hello(Model model) {

            Message message =
messageRepository.findByMessageKey("hello");
            model.addAttribute("message", message.getMessageText());

            return "hello";
      }
```

5. Make the indicated changes below, using your own name:

```
      @RequestMapping("hello/from/lcunningham")
      public String helloFromLCunningham(Model model) {

            Message message =
messageRepository.findByMessageKey("hello-lcunningham");
            model.addAttribute("message", message.getMessageText());
```

```
                return "hello ";
    }
```

6. Edit data-h2.sql and add a line like this with your own named message key:
   `INSERT INTO messages (message_key, message_text) VALUES ('hello-lcunningham', 'Hello folks... this is your lecturer speaking!');`
7. Restart the application (hit stop, then the run button). Visit the new route, e.g. http://localhost:8080/hello/from/lcunningham

Part 3 – Collaboration and Version Control

Several students have now been editing their local copies of the project. It is time to start pushing your changes to the central server so that everyone's changes can be pulled down to the local repositories.

Follow the instructions in the BitBucket 101 tutorial from last week (see link in Blackboard week 1) and add your changes.

"git add -A" within the repository directory will pick up the changed files. Use "git status" to view the changes, e.g.

```
H:\repos\test-repo>git add -A

H:\repos\test-repo>git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   src/main/java/ie/cit/comp8005/testproject/
controller/HomeController.java
        modified:   src/main/resources/data-h2.sql
```

Use:

```
git commit -a -m "enter a useful message here"
```

Then push it up to BitBucket:

```
H:\repos\test-repo>git push origin master
Password for 'https://dlarkinc@bitbucket.org':
Counting objects: 13, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (13/13), 988 bytes | 0 bytes/s, done.
Total 13 (delta 4), reused 0 (delta 0)
To https://dlarkinc@bitbucket.org/comp80052015/test-repo.git
   b8e063d..1ee6c46  master -> master
```

Try "git pull" every couple of minutes to see if you pull down any new changes from your classmates.

Try running the project (refresh it in Eclipse first) and see if the new routes added to HomeController are working or not.