CS-321: Server-Side Web Development
Spring 2021
Northeastern Illinois University
Homework #4: Spring Boot
Due Date: Thursday, 02/11/21 by 6:00 p.m. CDT
Terminal Due Date: Thursday, 02/18/21 by 6:00 p.m. CDT

Homework Goals:

1. Understand how all the Spring terminology relates and ultimately ties in to Spring Boot
2. Build the Spring Boot framework for your project.
3. Create a GitHub repo and push your project to GitHub via GitHub desktop.

Part I: Understanding Spring terminology.

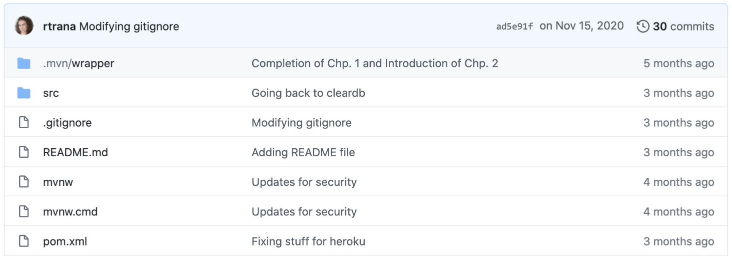
- Create a diagram that shows how the following terms are related to each other. Arrows from one term to another can also include "relationship" information.
- This diagram cannot be handwritten! You can use an editor such as Microsoft Word or https://app.diagrams.net. For an example of what a diagram might look like, please see here or here.
- Convert the diagram to a pdf. File types other than pdf will not be accepted!
- Terminology:
  - ☐ `Dependency Injection`
  - ☐ `ApplicationContext`
  - ☐ `@Bean, @Component, @Autowired`
  - ☐ Servlets
  - ☐ `ServletContext`
  - ☐ HTTP, `@WebListener, @WebServlet`
  - ☐ `WebApplicationContext, AnnotationConfigWebApplicationContext`
  - ☐ `XmlWebApplicationContext`
  - ☐ Spring Web, Spring Web MVC
  - ☐ `DispatcherServlet, AbstractAnnotationConfigDispatcherServletInitializer`
  - ☐ `ViewResolver`
  - ☐ `WebMvcConfigurer`
  - ☐ `@Controller, @RequestMapping, @EnableWebMv, @Configuration, @ComponentScan`
  - ☐ Spring Boot, `@SpringBootApplication, @SpringBootConfiguration, @EnableAutoConfiguration`

Part II: Spring Boot Framework for your project!

- Pick a domain name for your website. Go to https://www.domain.com to see if it's available (but don't purchase it because that's expensive!!). This should not be overly long.
- Create a Spring Boot project using IntelliJ and the Spring Initializr. Use the following for your Project Settings:
  - `Group`: This should be your domain name in reverse (Maven convention). For example: `racheltrana.com` would be `com.racheltrana`. This should be all lowercase.
  - `Artifact`: The name for your project, with words separated by hyphens. Don't make this too long, because it will be part of your base package name and also the name for application entry .java file. This should be all lowercase.
  - `Description`: Provide a short description specific to your project.
- Make sure to save your project in a folder with the same name as the Artifact. Note that wherever you save this project on your computer is going to end up being your Git repo (and you won't be able to move it without messing up your GitHub Desktop configurations!), so choose wisely.
- Make sure to choose the same Dependencies as described in lecture.
- Set up the Spring Boot Dev Tools (described in lecture).
- Change the `application.properties` file to an `application.yml` file.
- Create a `.gitignore` file (this is just a new file without an extension and it is literally named `.gitignore`) and add all the patterns that we want Git to ignore (this is in the lecture). This should go in the root project directory (i.e. not the base package. The top-most folder also referred to as your root directory).
- Create an HTML file named `index.html` in the correct location in your Spring Boot project. Add the Thymeleaf namespace. Add a title in the body of the HTML file (use the correct tags) that has the name of your project as described in your project proposal.
- Create a `controllers` directory inside the base package. Add a Java class named `HomeController`. Add the correct annotations for controllers! This controller should map to the path `"/"`. Create a method named `getHomePage` that does not take any parameters and returns a String. This method should return the name of the view as a String (you only have one view right now!). Note that when returning the name of a view, you do not include the file extension. Also make sure that this method has the correct annotation on it - it is handling GET requests.
- Run your project. Go to Chrome and enter `http://localhost:8080` and verify that you see the HTML page that you created. Note that I will be testing all code in Chrome, so your apps must work in Chrome!
- Always remember to stop your application before exiting IntelliJ.
- You must follow Java and HTML naming/coding conventions. You will be asked to re-do anything that is incorrect or does not follow these coding conventions.

Part III: GitHub!!

- If you do not already have a GitHub student education account, create one now. Make sure to set your name on your profile.
- Make sure you have Git installed first on your computer, and then GitHub Desktop.
- Following the example in lecture, turn your project folder into a local Git repo and then push it to GitHub. Verify that you see your code in GitHub.
- Your GitHub repo should look like the following (without the README.md file at this point):



rtrana Modifying gitignore — ad5e91f on Nov 15, 2020 — **30** commits

.mvn/wrapper	Completion of Chp. 1 and Introduction of Chp. 2	5 months ago
src	Going back to cleardb	3 months ago
.gitignore	Modifying gitignore	3 months ago
README.md	Adding README file	3 months ago
mvnw	Updates for security	4 months ago
mvnw.cmd	Updates for security	4 months ago
pom.xml	Fixing stuff for heroku	3 months ago

- Share your repository with me (i.e. invite me to your repo!). My GitHub handle is `@rtrana`.
- In the root directory of your project, add a new file named `README.md`. These are files that are written in markdown language (hence, the .md extension). Markdown language is very simple to get started with and can be written in a variety of simple text editors - including IntelliJ! See here for simple markdown language commands: https://www.markdownguide.org/cheat-sheet/s.
- In this file, include all the components of the project proposal that you turned in for Homework 1. Make it look nice! This is something for you to show off to future interviewers.
- Commit and push your updated project to GitHub. Verify that you see the `README.md` file in your GitHub repo.

Submitting your code to D2L:

1. Turn in the .pdf file to the D2L assignment folder.
2. Zip up the project folder and submit to the D2L assignment folder.