CS-321: Server-Side Web Development
Spring 2021
Northeastern Illinois University
Homework #3: Spring and Spring MVC
Due Date: Thursday, 02/04/21 by 6:00 p.m. CDT
Terminal Due Date: Thursday, 02/11/21 by 6:00 p.m. CDT

Homework Goals:

1. Build a basic Spring servlet application using the Servlet API and annotations.
2. Build a basic Spring project with annotations to demonstrate dependency injection.
3. Deadlines are moderately flexible. The suggested deadline is when I think you should submit your homework by to stay on track. You can submit before then of course! It takes me about 5-7 days to provide feedback.
4. The terminal due date is the last date/time that you can submit the assignment by to receive credit.

Part I: Basic Spring servlet application

- You need to have Apache Tomcat downloaded in order to do this!
- Launch IntelliJ and choose New Project. Click Java Enterprise on the sidebar and then click Next.
- Make sure the Servlet specification is checked and then click Next.
- Create a folder with the name `SpringServlet` (this will also be the name of the project). Fill in `edu.neiu` for GroupId. Click Finish.
- In the pom.xml file, add the following dependencies in the dependencies section underneath the version tags and click the little `m` icon that appears in the window to have Maven download the dependencies.

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.3</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.3.3</version>
</dependency>
```

- Delete the web.xml file located in the WEB-INF directory.
- In the `edu.neiu.springservlet` package, create a Java class named `ApplicationContextListener`. This class should do/have the following:

1. It should implement `ServletContextListener`.
2. Add a comment that describes the role of the `ServletContextListener`.
3. Add the `@WebListener` annotation above the class header.
4. Add a comment that describes the purpose of this annotation.
5. Add the following code:

```
@Override
public void contextInitialized(ServletContextEvent sce) {
    AnnotationConfigWebApplicationContext webContext = new AnnotationConfigWebApplicationContext();
    webContext.scan("edu.neiu.SpringServlet");
    webContext.refresh();
    sce.getServletContext().setAttribute("context", webContext);
}
```

6. As comments in this class, explain which lines in this code are analogous to the code required in the xml file for the corresponding lecture example that demonstrated an XML configuration for a servlet application.

- In the `edu.neiu.springservlet` package, create a Java class named `MessageBean`.
- Create a default constructor that does not do anything.
- Override the `toString` method to return the String "`MessageBean`.
- Add the `@Component` annotation above the class header.
- As a comment in this file, describe what line in the `ApplicationContextListener` searches for this bean (i.e. searches for this component. Remember, a component is a bean).
- In the `HelloServlet` class, delete everything except the `doGet` and `destroy` methods.
- In the class annotation, change `value = "/hello-servlet"` to `value = "/hello"`. Add a comment that describes what this does.
- In the `doGet` method, delete the existing code, and add the following:

```
ApplicationContext context = (ApplicationContext) request.getServletContext().getAttribute("context");
MessageBean bean = context.getBean(MessageBean.class);

response.setContentType("text/html");

PrintWriter out = response.getWriter();
out.println("<html><body>");
out.println("<h1>" + bean.toString() + "</h1>");
out.println("</body></html>");
```

- Add a comment that describes what the first two lines of this code does.
- Go to the `index.jsp` file and modify the `href` parameter to be "`hello`".

- Go to Edit Configurations at the top of IntelliJ. If Apache Tomcat is not configured as the Application Server, you will need to configure it. To do this, Click Configure, then the plus sign and name this Tomcat 9.0.41 and Navigate to where you downloaded, unzipped, and saved Apache Tomcat. Choose the apache-tomcat-9.0.41 directory for the home and base directories. Click ok.
- Uncheck the Launch checkbox.
- Click on the deployment tab and change the Application context to be just/.
- Click ok and then click the green play button to launch the application.
- Go to a browser and enter: `localhost:8080`. Then click on the link. Notice how your URL changes!
- Go back to IntelliJ and stop the application (note - you should always stop the application before exiting IntelliJ). Change the href parameter in the index.jsp to be `"foo"`. In order to make this URL work, what also needs to be changed in the `HelloServlet` class?
- Run the application again, refresh the browser at `localhost:8080` and test out your link.

Part II: Basic Spring MVC, with annotations

- As above, launch IntelliJ and choose New Project. Click Java Enterprise on the sidebar and then click Next.
- Make sure the Servlet specification is checked and then click Next.
- Create a folder with the name `SpringWebMvc` (this will also be the name of the project). Fill in `edu.neiu` for GroupId. Click Finish.
- In the pom.xml file, add the following dependencies in the dependencies section underneath the version tags and click the little `m` icon that appears in the window to have Maven download the dependencies.

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-context</artifactId>
  <version>5.3.3</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-web</artifactId>
  <version>5.3.3</version>
</dependency>

<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.3.3</version>
</dependency>
```

- Replace the existing <build> tags in the pom.xml file with the following (this includes all the tags necessary to build/deploy an mvc war app). Note that you can also get this by following Lecture 3.3 MVC and creating a Maven project with a web app archetype:

```
<build>
    <finalName>springwebmvc</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (
        may be moved to parent pom) -->
        <plugins>
            <plugin>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.1.0</version>
            </plugin>
            <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.
                html#Plugin_bindings_for_war_packaging -->
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.0.2</version>
            </plugin>
            <plugin>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.0</version>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.1</version>
            </plugin>
            <plugin>
                <artifactId>maven-war-plugin</artifactId>
                <version>3.2.2</version>
            </plugin>
            <plugin>
                <artifactId>maven-install-plugin</artifactId>
                <version>2.5.2</version>
            </plugin>
            <plugin>
                <artifactId>maven-deploy-plugin</artifactId>
                <version>2.8.2</version>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
```

- Delete the web.xml file located in the WEB-INF directory.
- Delete the `HelloServlet` class.
- In the `edu.neiu.springwebmvc` package, create a Java class named `GatewayAppInitializer`. This class should do/have the following:
    1. It should extend `AbstractAnnotationConfigDispatcherServletInitializer`. (That's a heck of a name, right??)
    2. Add a comment in this class that describes the purpose of `AbstractAnnotationConfigDispatcherServletInitializer`.
    3. Right-click on `AbstractAnnotationConfigDispatcherServletInitializer` and choose Show Context Actions. Then choose Implement Methods and make sure that all 3 methods are selected. Click ok.
    4. Modify the `getServletMappings` method to be: `return new String[]{"/"};`
- In the `edu.neiu.springwebmvc` package, create a Java class named `GatewayAppWebConfig`. This class should do/have the following:

1. It should implement `WebMvcConfigurer`
2. Add a comment in this class that describes the purpose of `WebMvcConfigurer`.
3. Above the class header, add the `@Configuration`, `@EnableWebMvc`, and `@ComponentScan(basePackages = {"edu.neiu.springwebmvc"})` annotations.
4. Add comments that describes what each of these annotations does.
5. Add the following code:

```
@Override
public void configureViewResolvers(ViewResolverRegistry registry) {
    registry.viewResolver(jspViewResolver());
}

@Bean
public ViewResolver jspViewResolver() {
    InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
    viewResolver.setPrefix("/WEB-INF/jsp/");
    viewResolver.setSuffix(".jsp");
    return viewResolver;
}
```

6. Add comments that describe what each of these methods does.
7. Go back to the `GatewayAppInitializer` class and modify the `getServletConfigClasses` method to do the following:
   `return new Class[]{GatewayAppWebConfig.class};`

- In the WEB-INF directory, create a folder (i.e. directory) named `jsp`. Drag the `index.jsp` file into that new folder. Rename `index.jsp` to `home.jsp` (this is under Right-click -> refactor).
- Modify the `home.jsp` file to be:

```
<!DOCTYPE html>
<body>
<h2>${message}</h2>
</body>
</html>
```

- In the `edu.neiu.springwebmvc` package, create a Java class named `HelloWorldController`. This class should do/have the following:
  1. Above the class header, add the `@Controller` and `@RequestMapping("/")`
  2. Add comments that describe what each of these annotations does.
  3. Add the following to the class:

```
@GetMapping
public String homePage(Model model) {
    model.addAttribute("message", "Welcome to the Hello World homepage");
    return "home";
}
```

  4. Add a comment that describes in detail what this method and annotation do.

- As in Part I, make sure that Tomcat is set up so that you can run your app. Run the app and navigate to `localhost:8080`.
- Stop the app. In the controller class, modify the `@RequestMapping("/")` annotation to be: `@RequestMapping("/foo")`. Run the app - how do you navigate to the home page now?

Submitting your code to D2L:

1. Zip up both project folders, (`SpringServlet`, and `SpringWebMvc`. In other words, your zip file should contain 2 folders.
2. Submit this one zip file to D2L.