

Pre-Class Assignment

Dependency Injection, An Introduction to Spring

Here is a list of materials you will need to read in preparation for the next lab period. Read everything carefully to make sure you understand the concepts presented here. There will be a reading quiz at the beginning of class.

What to learn

- How Spring can simplify Java Development
- Dependency Injection: Use cases and benefits
- Elimination of boilerplate code with Spring
- How beans are wired by Spring using XML

Readings

Spring in Action, 4th Edition by Craig Walls

Note: In the following readings, focus mainly on the topics regarding dependency injection, wiring via XML, and JDBC templates. Other topics touched upon in this reading, such as AOP, will be for part 2 of this lab

<https://learning.oreilly.com/library/view/spring-in-action/9781617291203/>

Read sections 1.0 - 1.2 (you can skim section 1.1.3), 2.0 - 2.2.1, and 2.4

Pre-Class Assignment

Complete the setup below to prepare for the tutorial.

Import a Maven Spring Project

1. Download the [project source files](#) and unzip them to a working directory.
2. Open IntelliJ and select the option to import a project from existing sources (If you see a welcome screen, just click on "Import Project"). A pop up window to select what project to import will show. Navigate to the folder where you unzipped the project files and select the pom.xml file.
3. All the files in the project should be imported. The program will not run yet since you need to include the required dependencies in the Maven pom.xml file. We will be using

Spring 5 for this tutorial. Add the dependencies for the latest production version of **spring-context**, **spring-jdbc**, **spring-test**, **sqlite-jdbc**, **mockito-core**, and **junit-jupiter-api** to the Maven **pom.xml** file. Refer to the dependency management tools tutorial if you need help with this step. Look at the website here for the latest versions of these tools:

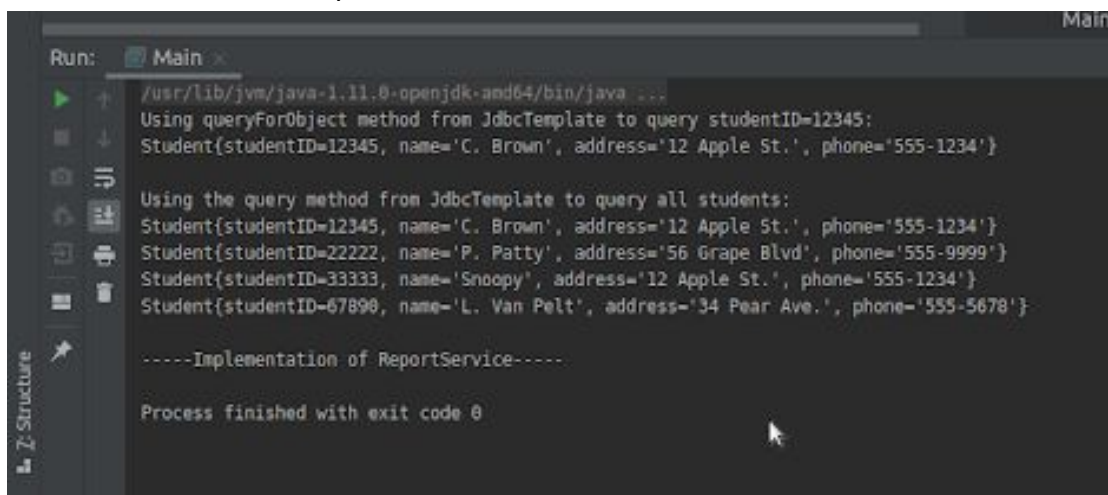
[org.springframework](https://springframework.org) (spring-context, spring-jdbc, and spring-test are found here)

[org.xerial](https://xerial.org) » [sqlite-jdbc](https://xerial.org/sqlite-jdbc)

[org.mockito](https://mockito.org) » [mockito-core](https://mockito.org/mockito-core)

[org.junit.jupiter](https://junit.org) » [junit-jupiter-api](https://junit.org/junit-jupiter-api)

4. Remember to enable Auto-Import of Maven dependencies. After all the dependencies are imported, you should not be getting any errors.
5. Open the `src/test/java` folder to find the `ReportServiceTest.java` file. Run the `ReportServiceTest`. **All the tests should run, but fail.**
6. Open the `src/main/java` folder to find the `Main.java` file. Run the main method. It should be able to output a query of a student and query of all students in the database. You should see this as the output:



```
Run: Main x
/usr/lib/jvm/java-11.0-openjdk-amd64/bin/java ...
Using queryForObject method from JdbcTemplate to query studentID=12345:
Student{studentID=12345, name='C. Brown', address='12 Apple St.', phone='555-1234'}

Using the query method from JdbcTemplate to query all students:
Student{studentID=12345, name='C. Brown', address='12 Apple St.', phone='555-1234'}
Student{studentID=22222, name='P. Patty', address='56 Grape Blvd', phone='555-9999'}
Student{studentID=33333, name='Snoopy', address='12 Apple St.', phone='555-1234'}
Student{studentID=67890, name='L. Van Pelt', address='34 Pear Ave.', phone='555-5678'}

-----Implementation of ReportService-----

Process finished with exit code 0
```

Install DB Browser for SQLite

1. If not yet installed, install DB Browser for SQLite. Download it here:
<https://sqlitebrowser.org/>
2. To do the tutorial, you must be able to look at what is in the database provided in the project's root folder. The database is named `data.sqlite`. Open DB Browser and click on "Open Database". A window will popup. Navigate to the project's root folder and select `data.sqlite`. Open it up and click on the "Browse Data" tab to view the contents of the database.