Create Your First Spring Project

Estimated time needed: 20 minutes

Overview

In this lab, you will be working on Spring Framework. You will set up a project structure and then build and run it in the IntelliJ environment.

Learning objectives

After completing this lab, you will be able to:

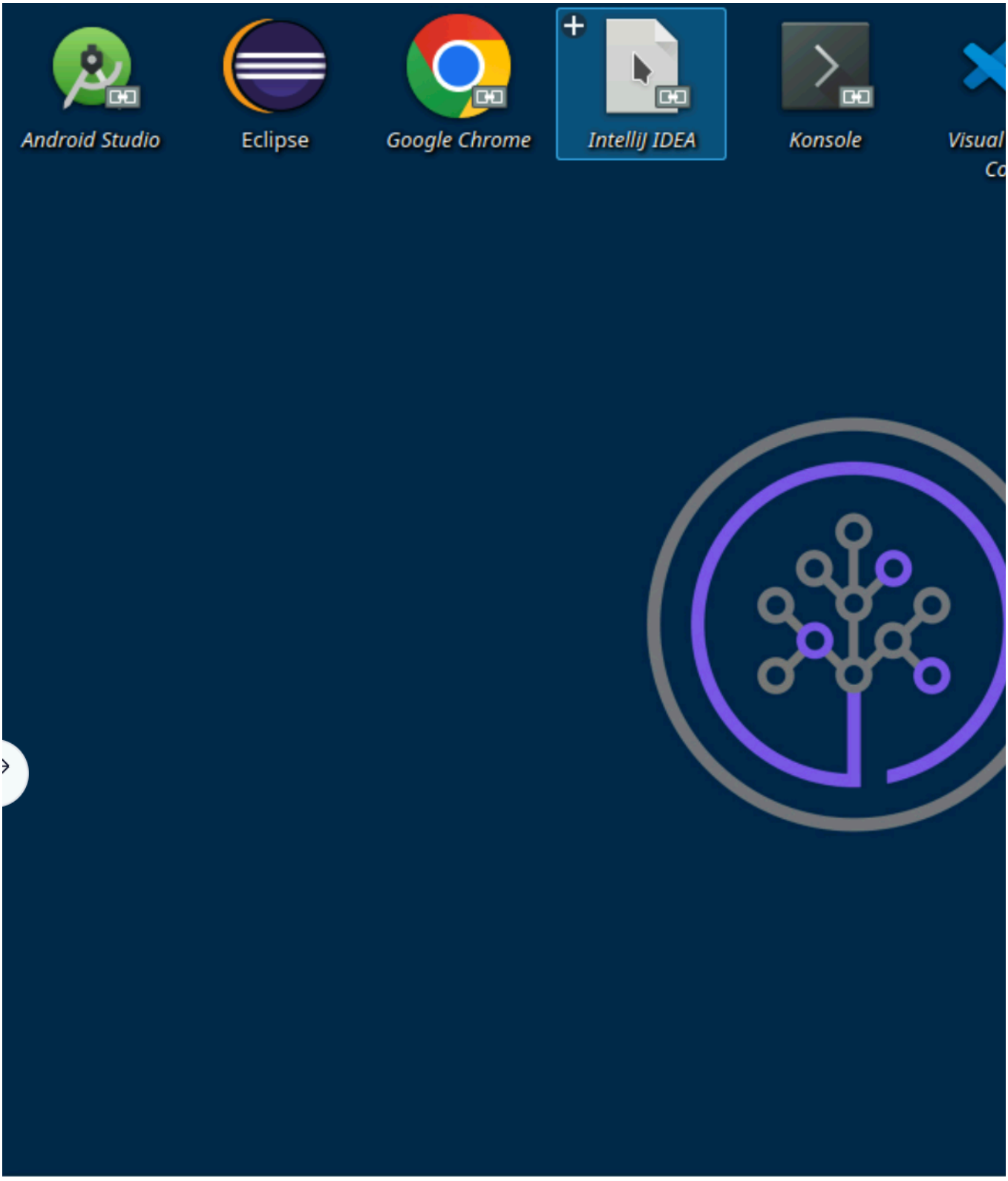
- Set up a Spring Framework project structure with Maven
- Edit the core configuration contained in `pom.xml`
- Create an `applicationContext.xml` file to define and configure Spring beans and their dependencies
- Create a bean class
- Build and run the spring application

Prerequisites

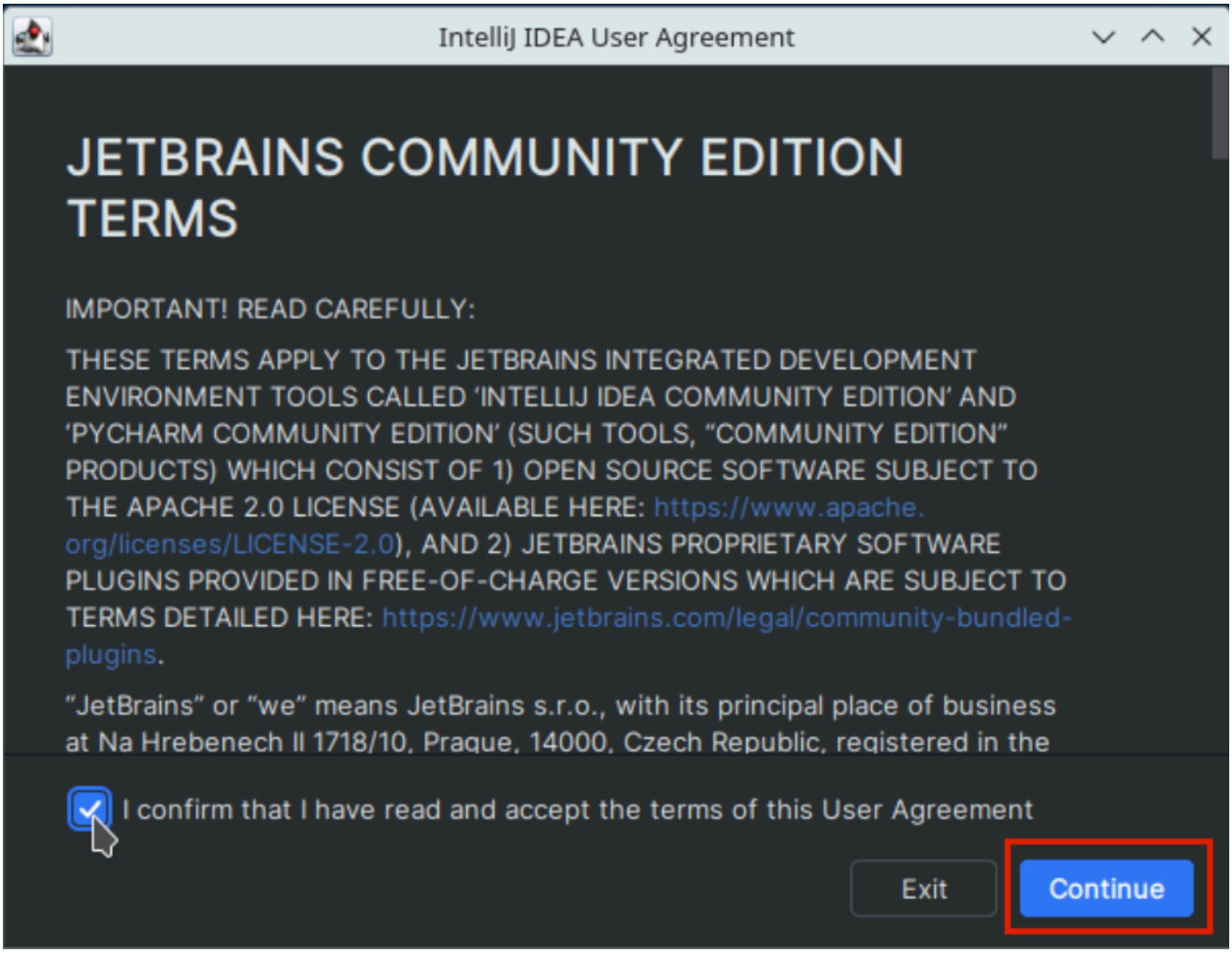
You should know basic Java programming before you get started with this lab.

Create a Maven project in IntelliJ

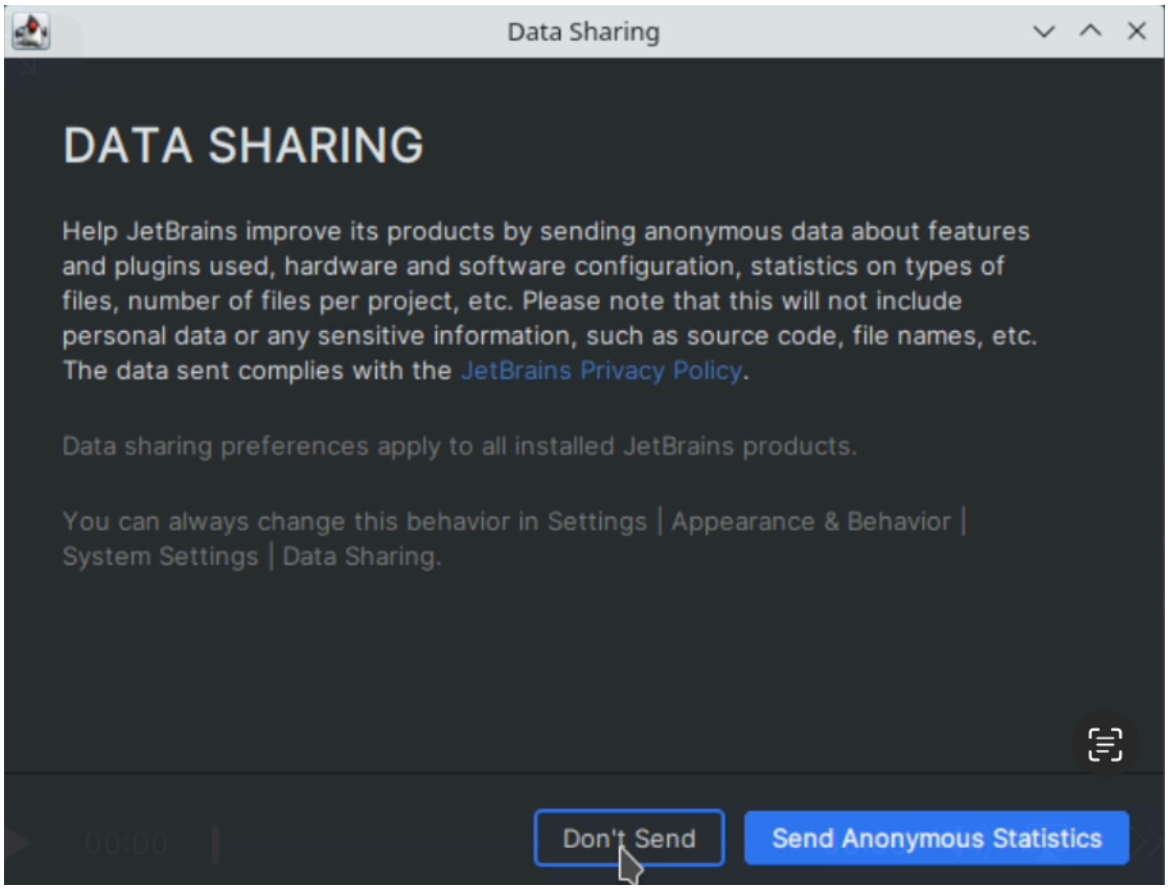
1. Open the IntelliJ IDE in the lab environment provided.



2. A `User Agreement` containing the terms and conditions for using the IntelliJ community edition will be displayed. Agree by selecting the checkbox and clicking `Continue`.

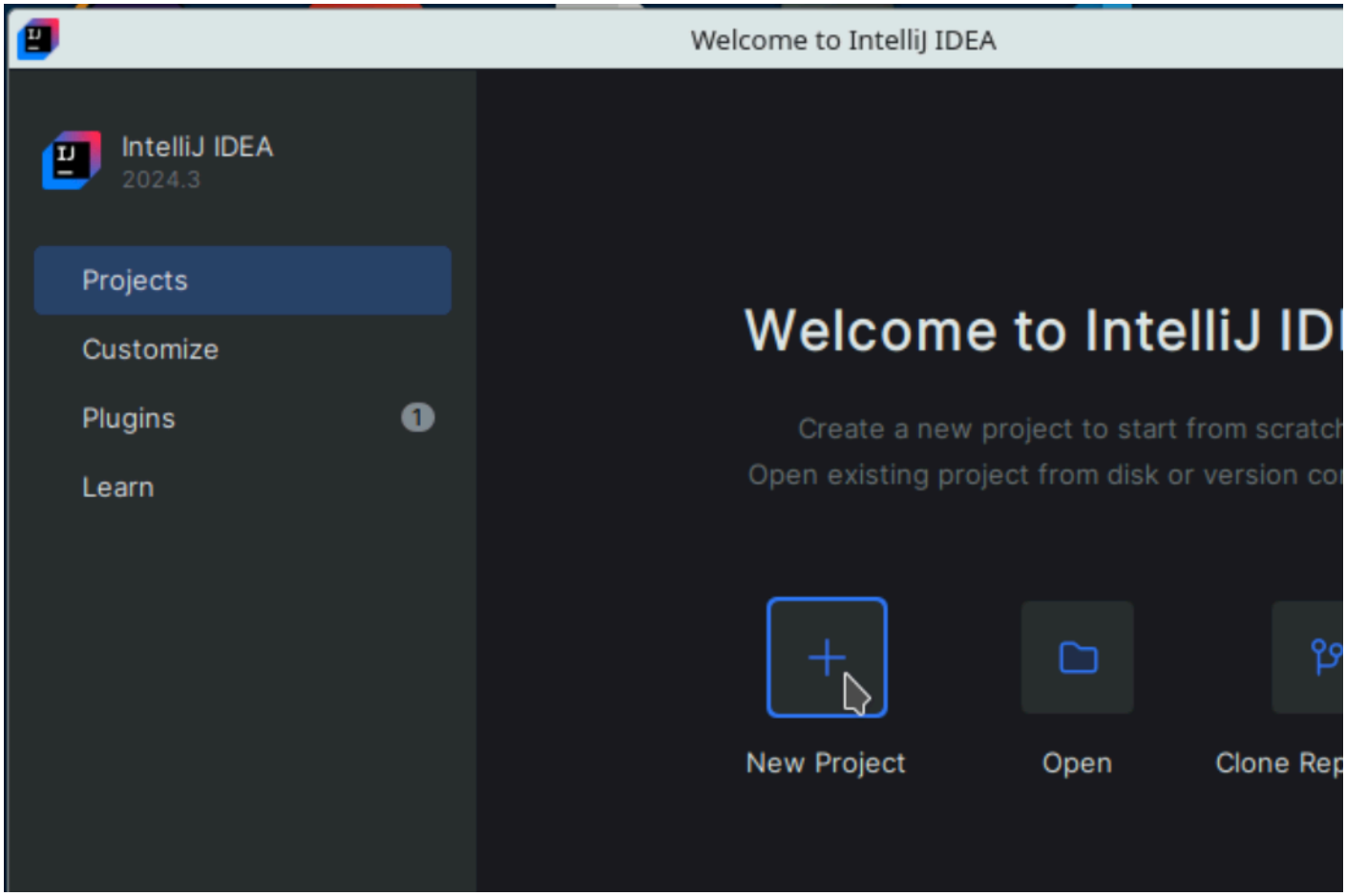


3. A window asking if you want to share your data on usage statistics will be displayed. Click `Don't Send` option.



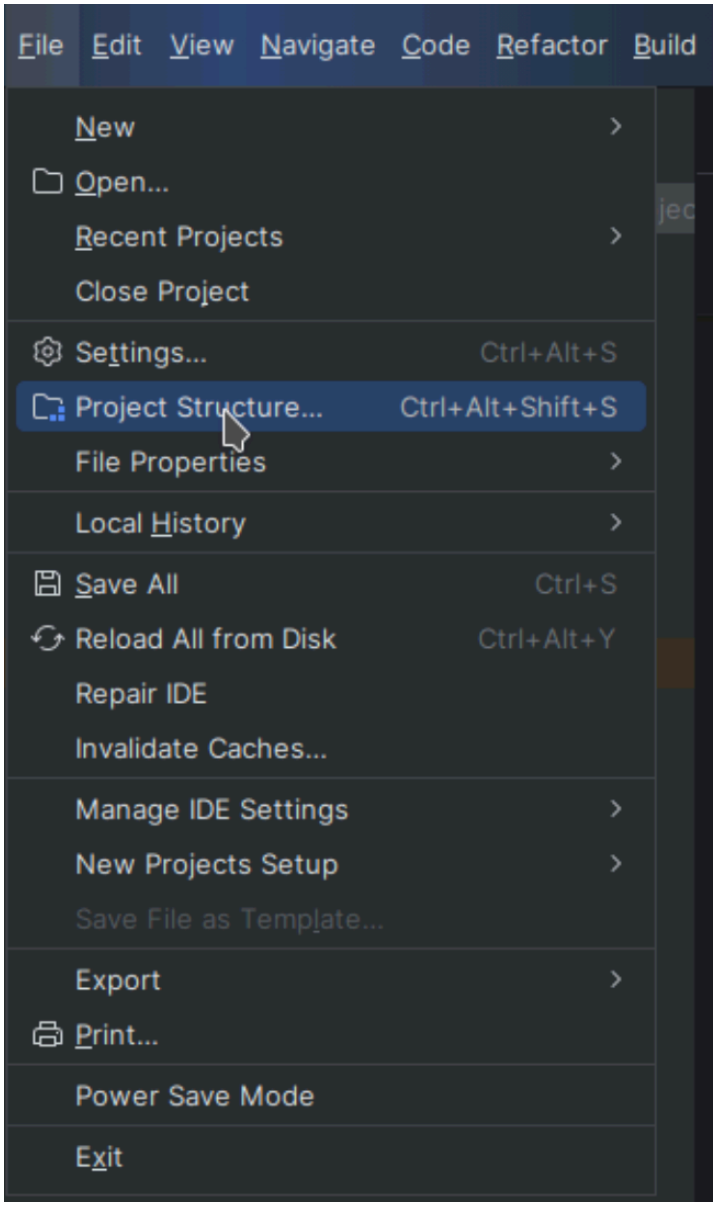
4. IntelliJ will open and provide you the options for creating a new project, opening an existing project from the Linux environment, or cloning from a repository.

Choose the option to create a new project.

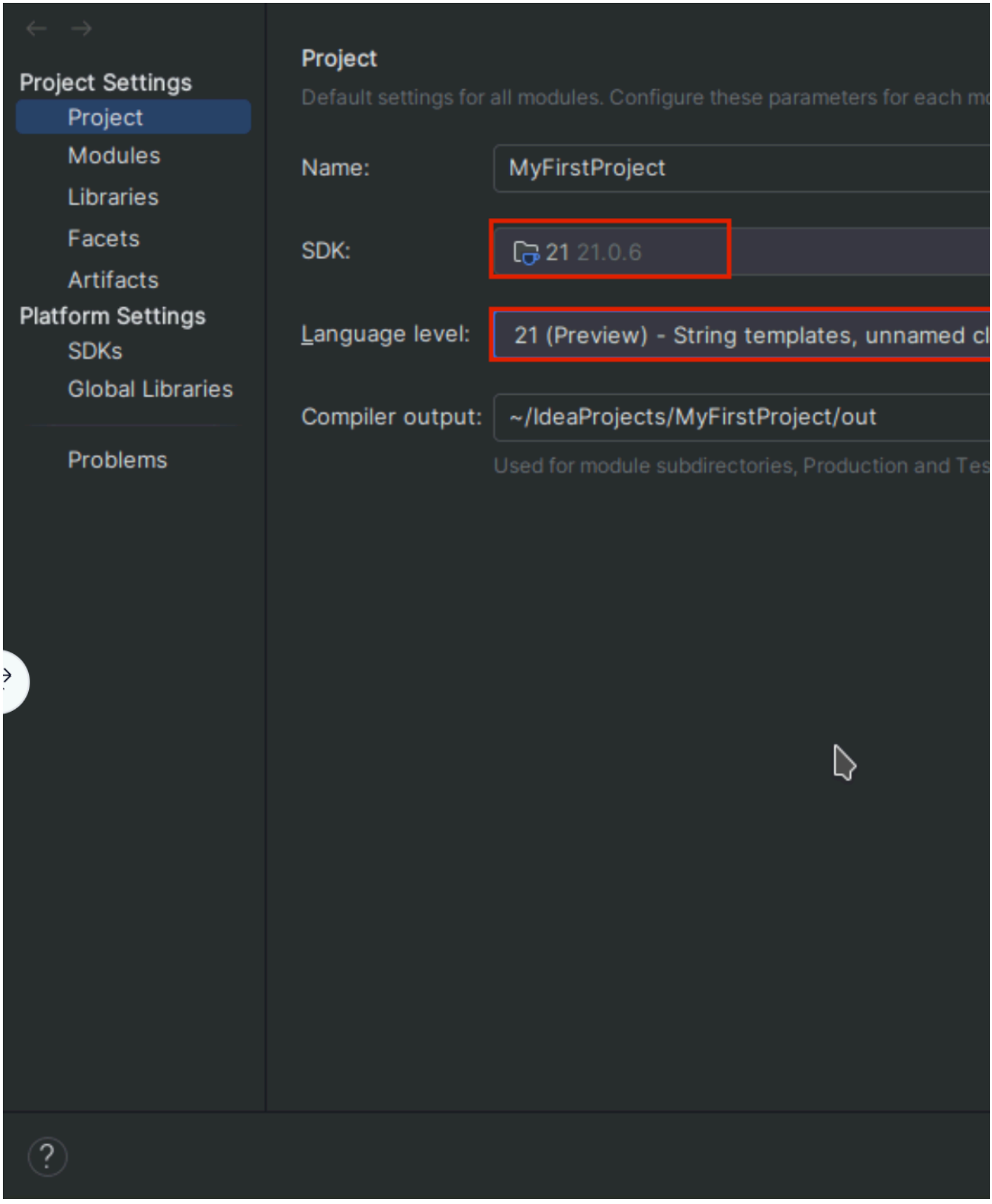


5. Enter the name of the project in the Name text field, choose the `Maven` option in the Build system field, choose the JDK version (*21.0.6, that's installed in the lab environment*), and click `Create`.

6. Click the `File` menu and select `Project Structure` to open the project settings.



7. Set the SDK and Language level to the version of Java installed (21 in this case), as shown in the image.



Configure the project

1. Open the `pom.xml` and paste the following content in place of the default content.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>MyFirstProj</artifactId>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <maven.compiler.source>21</maven.compiler.source>
        <maven.compiler.target>21</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <repositories>
        <repository>
            <id>mvn_repo_spring_context</id>
            <url>https://mvnrepository.com/artifact/org.springframework/spring-context</url>
            <releases>
                <enabled>false</enabled>
            </releases>
            <snapshots>
                <enabled>true</enabled>
            </snapshots>
        </repository>
    </repositories>
    <dependencies>
        <!-- Spring Context Dependency -->
        <!-- https://mvnrepository.com/artifact/org.springframework/spring-beans -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-beans</artifactId>
            <version>6.2.3</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>6.2.3</version> <!-- Use the latest stable version -->
        </dependency>
    </dependencies>
</project>
```

This describes the Spring class source and target JDK versions, the repositories from which the spring classes need to be downloaded, and the dependencies.

2. The environment needs to resync for the new configuration to be recognized. Right-click the project, choose `Maven` from the menu, and click `Sync Project`.

3. Create a new file named `applicationContext.xml` under `resources` directory.

4. Paste the following content in `applicationContext.xml`.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
    <!-- Define a bean for HelloWorld -->
    <bean id="helloWorld" class="com.example.HelloWorld">
        <property name="message" value="Hello, World from XML!"/>
    </bean>
</beans>
```

5. Create a class named `com.example.HelloWorld` under the project's `src/main` directory. Paste the content in `HelloWorld.java`.

```
package com.example;
public class HelloWorld {
    private String message;
    // Setter method for dependency injection
    public void setMessage(String message) {
        this.message = message;
    }
    // Method to print the message
    public void printMessage() {
        System.out.println("Message: " + message);
    }
}
```

6. Modify `Main.java` to have the following code.

```
package com.example;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Main {
    public static void main(String[] args) {
        // Load XML-based configuration
        ApplicationContext xmlContext = new ClassPathXmlApplicationContext("applicationContext.xml");
        HelloWorld helloWorldXml = (HelloWorld) xmlContext.getBean("helloWorld");
        helloWorldXml.printMessage(); // Output: Message: Hello, World from XML!
    }
}
```

Build and run

1. Right-click the project and `Build` or `Rebuild` the module.

2. Once the build is successful, you get a confirmation message to this extent.

If the build is unsuccessful, repeat the `sync` and `rebuild`.

3. Right-click the `Main` class and click `Run`.

You will see the output in the window.

Practice Exercise

1. In `Main.java` change the message passed to HelloWorld class and rebuild the project. Run it once the build is successful.

Conclusion

In this lab you you have:

- Set up a Spring Framework project structure with Maven
- Edited the core configuration contained in `pom.xml`
- Defined and configured Spring beans and their dependencies in applicationContext.xml
- Created `HelloWorld` bean class
- Built and ran the spring application

Author(s)

Lavanya