Lesson 01 Demo 01

Configuring Spring Core in Java Project

Objective: To configure the Spring Core dependencies in a Java project using Maven and XML configuration

Tool required: Eclipse IDE

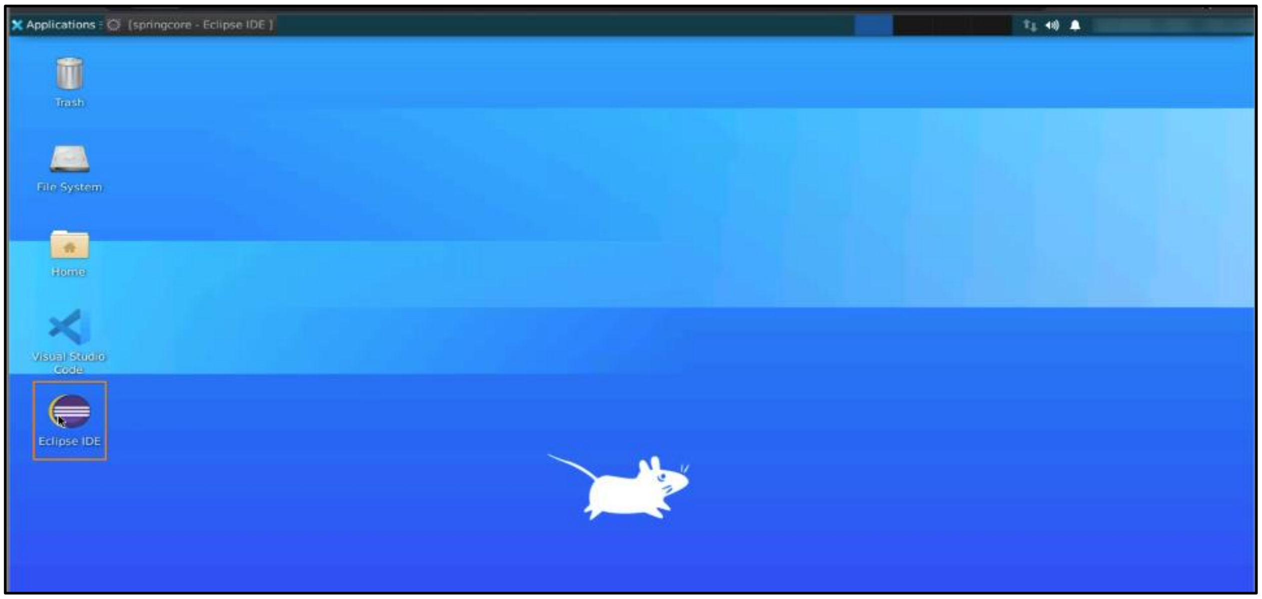
Prerequisites: None

Steps to be followed:

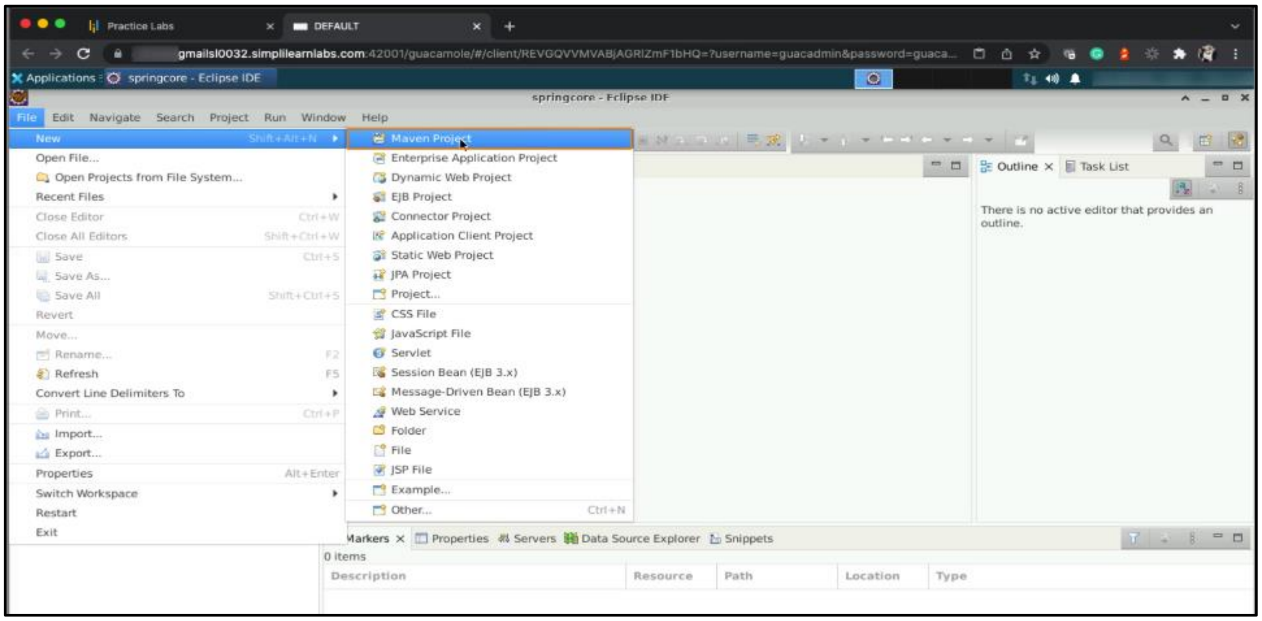
1. Creating a Maven project
2. Adding Spring Core dependency
3. Creating a bean class
4. Configuring XML file

Step 1: Creating a Maven project

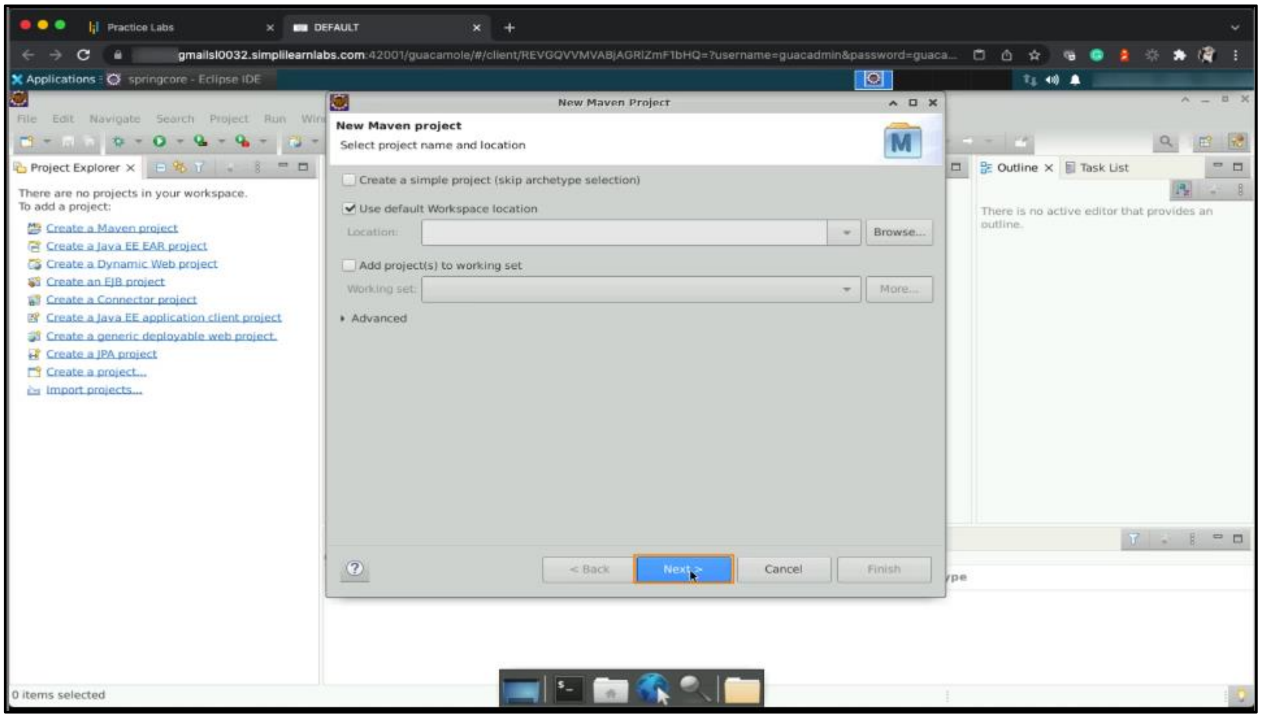
1.1 Open **Eclipse IDE**



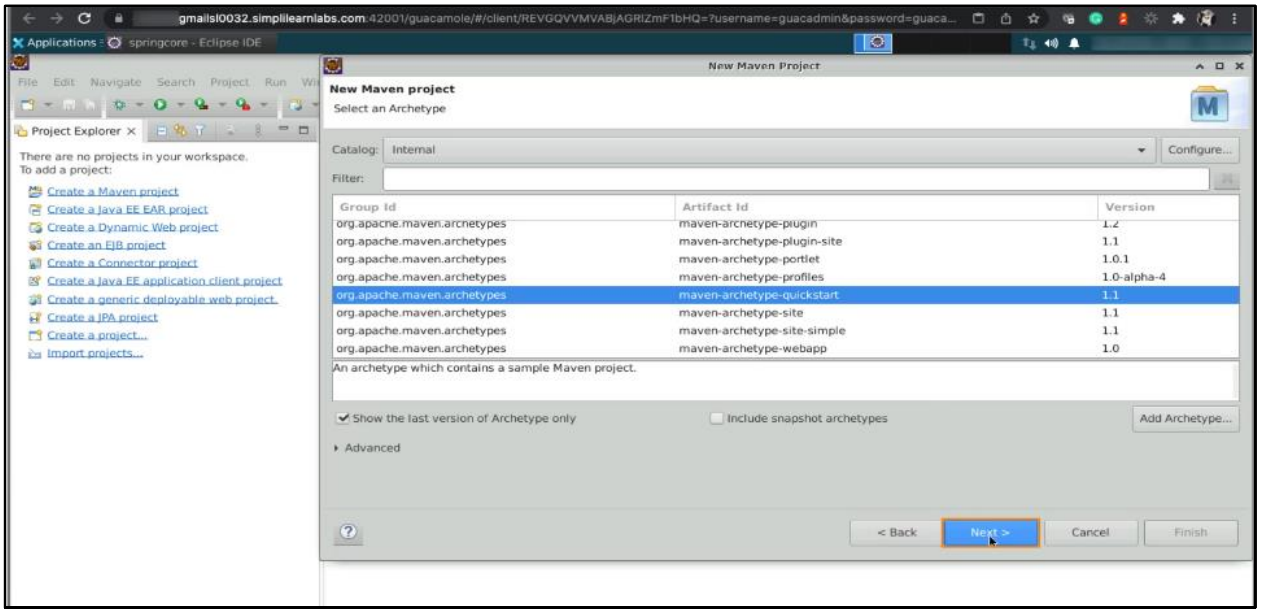
1.2 Click on **File** in the menu bar, select **New** and, choose **Maven Project**



1.3 Provide a workspace location that matches the selected workspace in Eclipse, and click **Next**


New Maven Project
New Maven project
Select project name and location
Create a simple project (skip archetype selection)
Use default Workspace location
Location:
Browse...
Add project(s) to working set
Working set:
More...
Advanced
< Back
Next >
Cancel
Finish
Project Explorer
There are no projects in your workspace.
To add a project:
Create a Maven project
Create a Java EE EAR project
Create a Dynamic Web project
Create an EJB project
Create a Connector project
Create a Java EE application client project
Create a generic deployable web project.
Create a JPA project
Create a project...
Import projects...
0 items selected

1.4 Select the **maven-archetype-quickstart** from the **Internal** catalogs and click **Next**

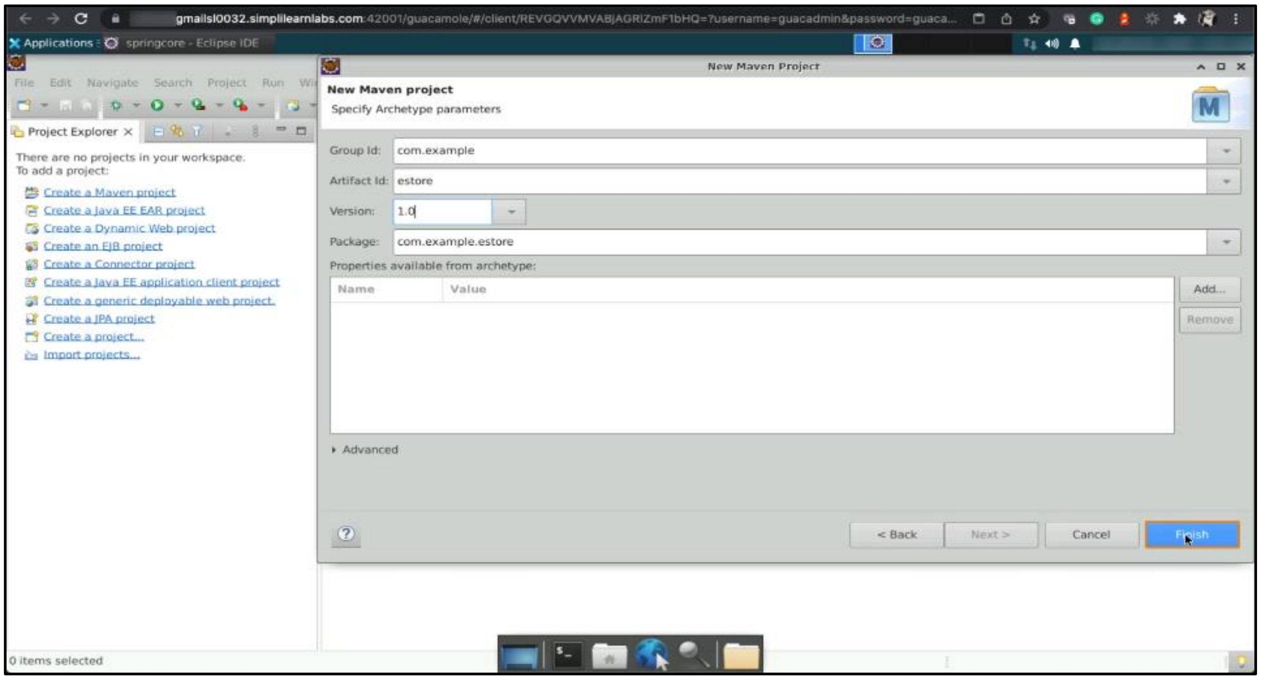


Group Id	Artifact Id	Version
org.apache.maven.archetypes	maven-archetype-plugin	1.2
org.apache.maven.archetypes	maven-archetype-plugin-site	1.1
org.apache.maven.archetypes	maven-archetype-portlet	1.0.1
org.apache.maven.archetypes	maven-archetype-profiles	1.0-alpha-4
org.apache.maven.archetypes	maven-archetype-quickstart	1.1
org.apache.maven.archetypes	maven-archetype-site	1.1
org.apache.maven.archetypes	maven-archetype-site-simple	1.1
org.apache.maven.archetypes	maven-archetype-webapp	1.0

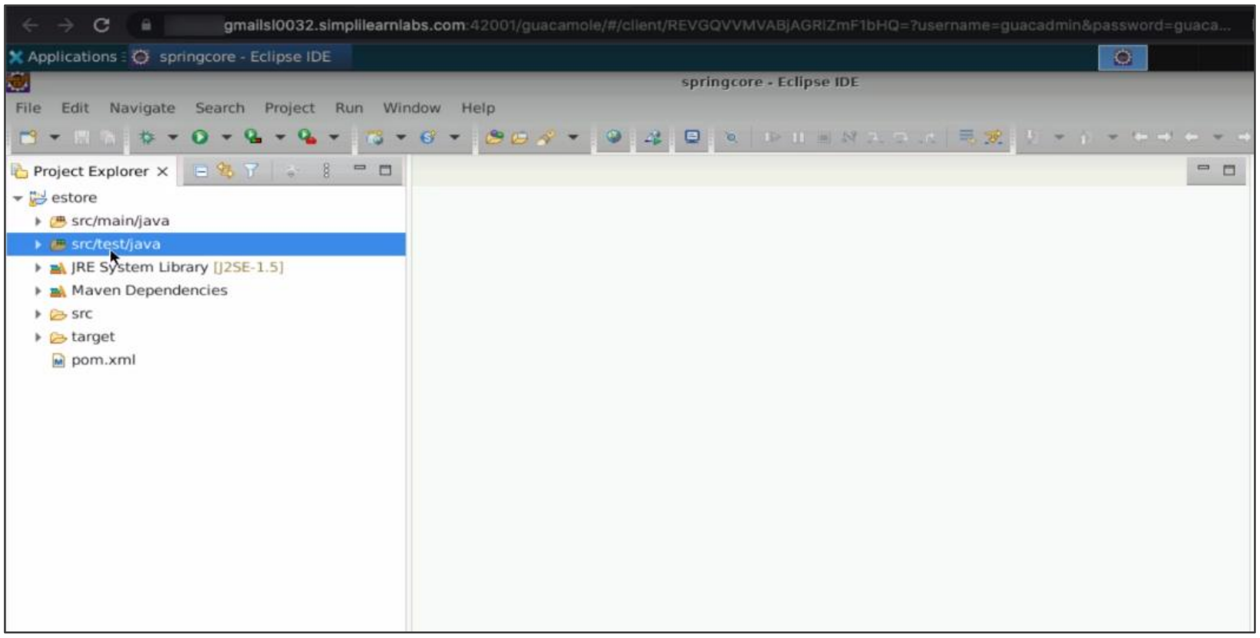
An archetype which contains a sample Maven project.

Show the last version of Archetype only
Include snapshot archetypes
Add Archetype...
Advanced
< Back
Next >
Cancel
Finish

1.5 Enter the Group Id, which typically follows the reverse order of the company's domain name. Also, provide the Artifact Id as **estore** and click **Finish.**



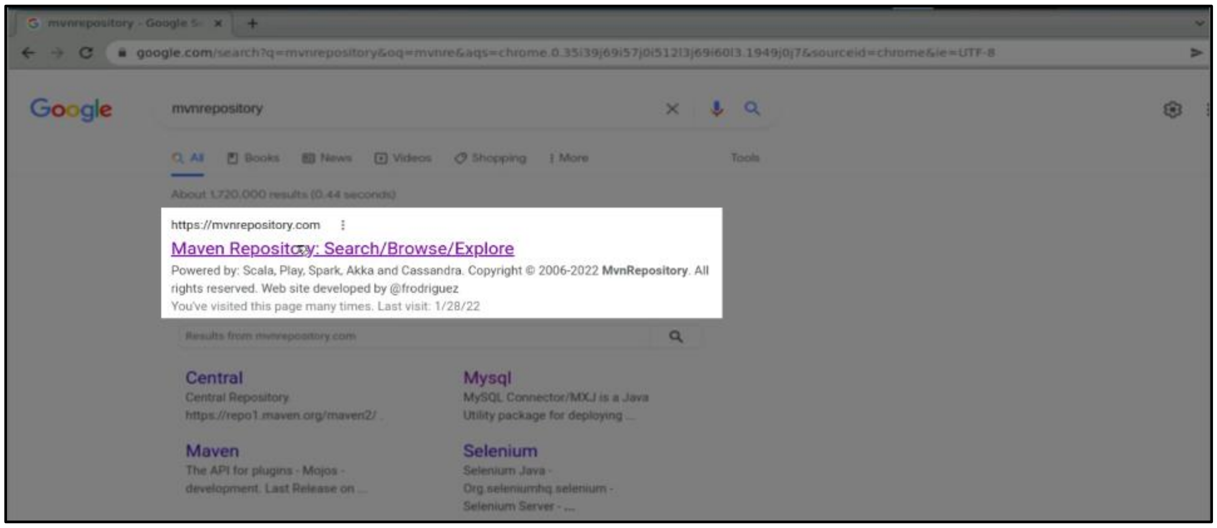
Group Id: com.example
Artifact Id: estore
Version: 1.0
Package: com.example.estore



In **Project Explorer**, you will see the newly created Maven project.

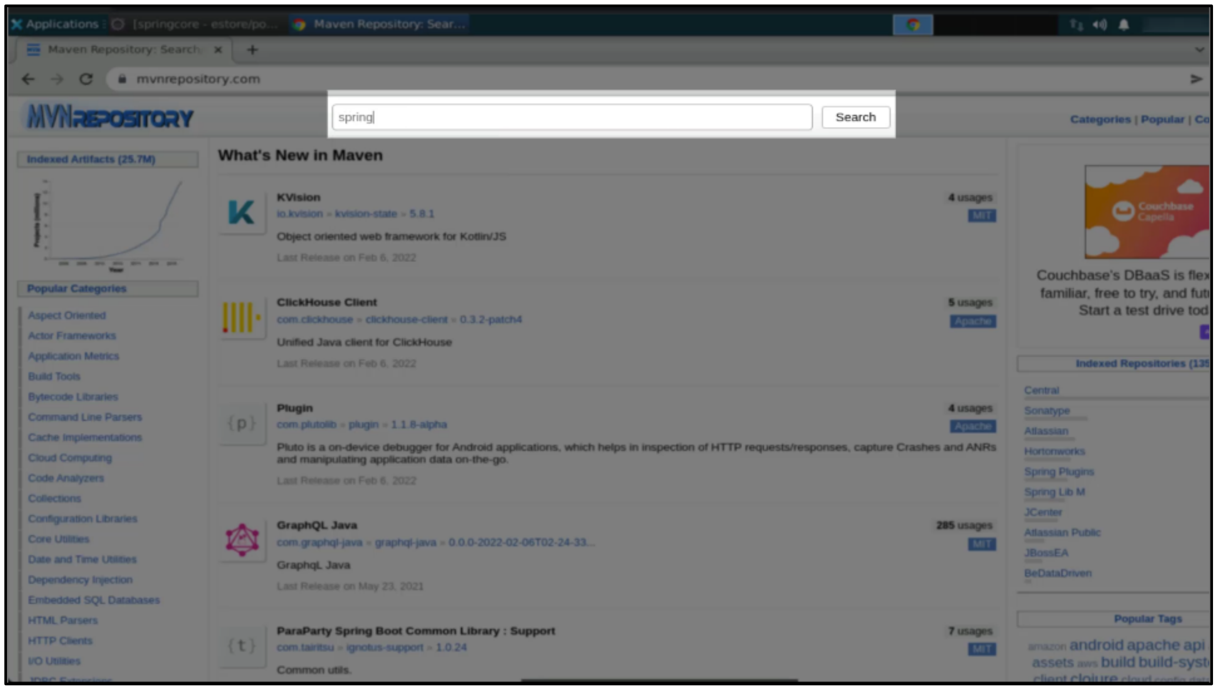
Step 2: Adding Spring Core dependency

2.1 Open a browser and search for **mvnrepository**


mvnrepository
https://mvnrepository.com
Maven Repository: Search/Browse/Explore
Central
Mysql
Maven
Selenium

Powered by: Scala, Play, Spark, Akka and Cassandra. Copyright © 2006-2022 MvnRepository. All rights reserved. Web site developed by @frodriguez
You've visited this page many times. Last visit: 1/28/22

2.2 Go to the **maven-repository.com** website and search for **Spring** in the repository search bar



What's New in Maven

KVision
io.kvision » kvision-state » 5.8.1
4 usages MIT
Object oriented web framework for Kotlin/JS
Last Release on Feb 6, 2022

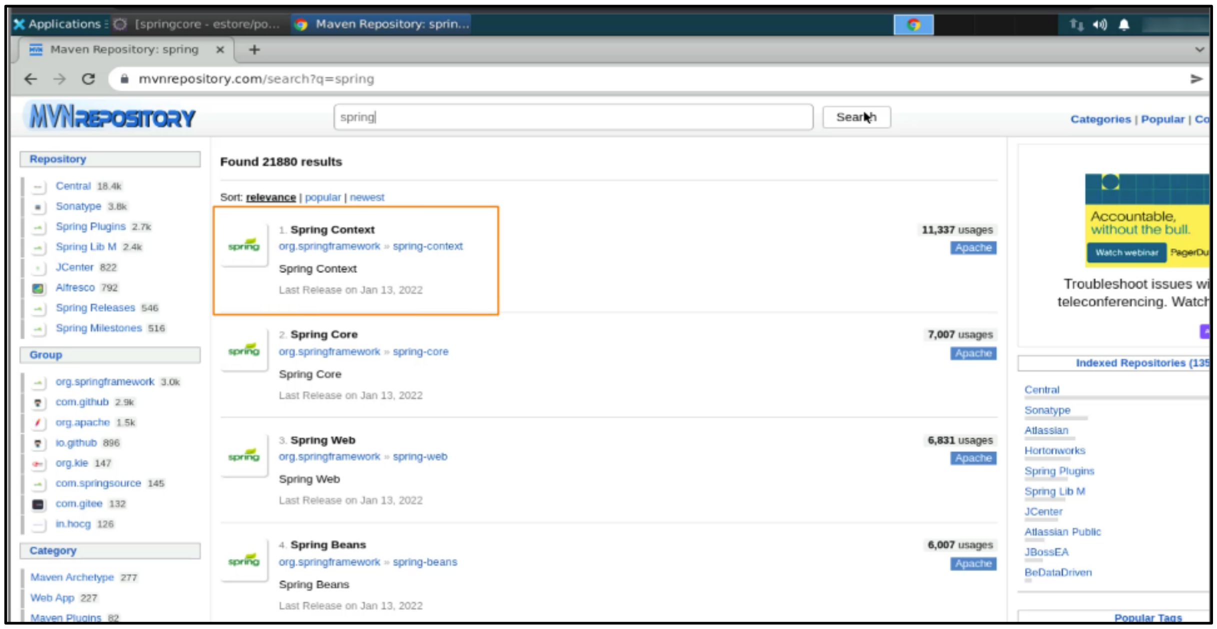
ClickHouse Client
com.clickhouse » clickhouse-client » 0.3.2-patch4
5 usages Apache
Unified Java client for ClickHouse
Last Release on Feb 6, 2022

Plugin
com.plutolib » plugin » 1.1.8-alpha
4 usages Apache
Pluto is a on-device debugger for Android applications, which helps in inspection of HTTP requests/responses, capture Crashes and ANRs and manipulating application data on-the-go.
Last Release on Feb 6, 2022

GraphQL Java
com.graphql-java » graphql-java » 0.0.0-2022-02-06T02-24-33...
285 usages MIT
GraphqL Java
Last Release on May 23, 2021

ParaParty Spring Boot Common Library : Support
com.tairitsu » ignotus-support » 1.0.24
7 usages MIT
Common utils.

2.3 Click on the **Spring Context** option from the search results



Found 21880 results

Sort: relevance | popular | newest

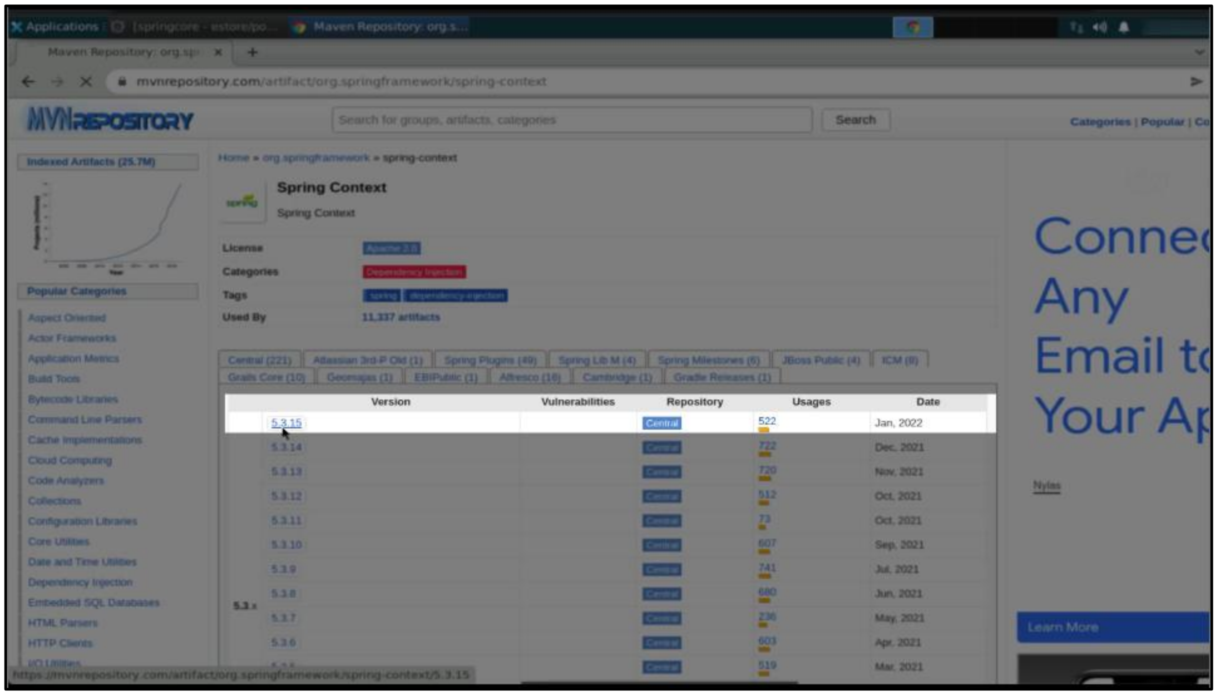
1. Spring Context
org.springframework » spring-context
11,337 usages Apache
Spring Context
Last Release on Jan 13, 2022

2. Spring Core
org.springframework » spring-core
7,007 usages Apache
Spring Core
Last Release on Jan 13, 2022

3. Spring Web
org.springframework » spring-web
6,831 usages Apache
Spring Web
Last Release on Jan 13, 2022

4. Spring Beans
org.springframework » spring-beans
6,007 usages Apache
Spring Beans
Last Release on Jan 13, 2022

2.4 Choose the latest version available


mvnrepository.com/artifact/org.springframework/spring-context

Home » org.springframework » spring-context

Spring Context

Spring Context

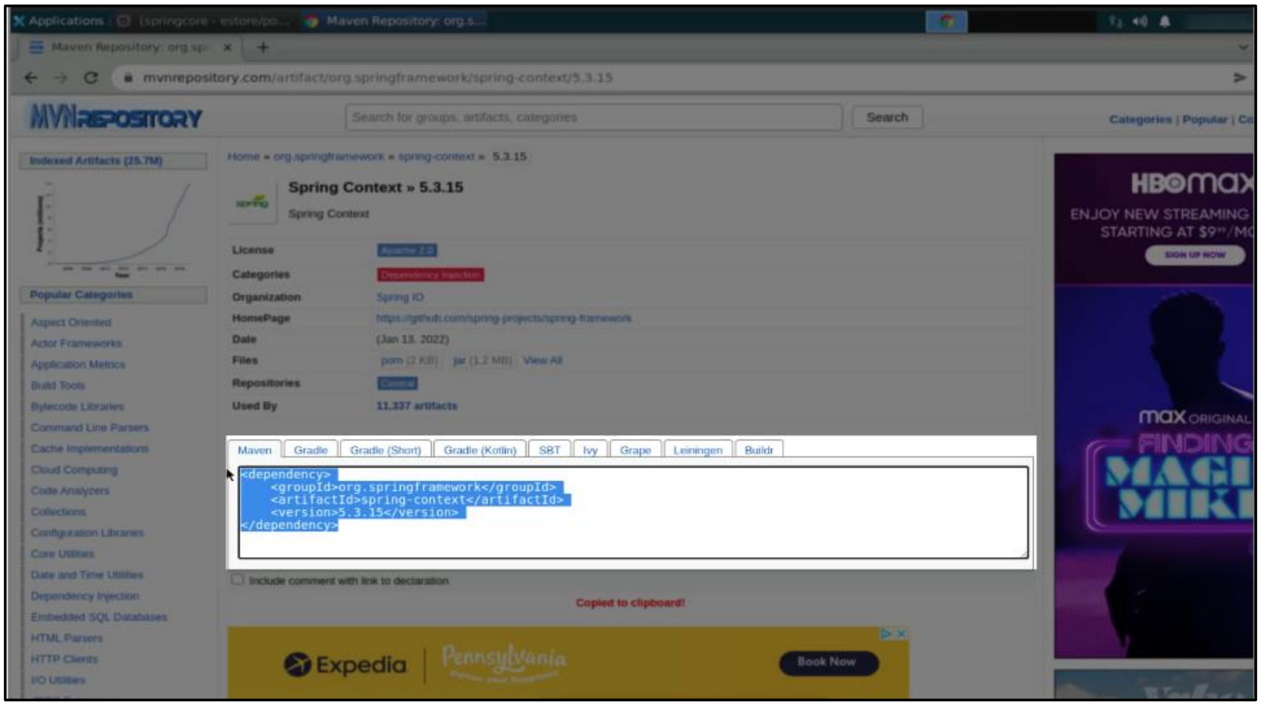
License	Apache 2.0
Categories	Dependency Injection
Tags	spring dependency-injection
Used By	11,337 artifacts

Central (221) | Atlassian 3rd-P Old (1) | Spring Plugins (49) | Spring Lib M (4) | Spring Milestones (6) | JBoss Public (4) | ICM (8) | Grails Core (10) | Geomajas (1) | EBIPublic (1) | Alfresco (16) | Cambridge (1) | Gradle Releases (1)

	Version	Vulnerabilities	Repository	Usages	Date
5.3.x	5.3.15		Central	522	Jan, 2022
	5.3.14		Central	722	Dec, 2021
	5.3.13		Central	720	Nov, 2021
	5.3.12		Central	512	Oct, 2021
	5.3.11		Central	73	Oct, 2021
	5.3.10		Central	607	Sep, 2021
	5.3.9		Central	741	Jul, 2021
	5.3.8		Central	680	Jun, 2021
	5.3.7		Central	236	May, 2021
	5.3.6		Central	603	Apr, 2021
	[illegible]		Central	519	Mar, 2021

https://mvnrepository.com/artifact/org.springframework/spring-context/5.3.15

2.5 Copy the dependency information provided for the selected version


mvnrepository.com/artifact/org.springframework/spring-context/5.3.15

Home » org.springframework » spring-context » 5.3.15

Spring Context » 5.3.15

Spring Context

License	Apache 2.0
Categories	Dependency Injection
Organization	Spring IO
HomePage	https://github.com/spring-projects/spring-framework
Date	(Jan 13, 2022)
Files	pom (2 KB) jar (1.2 MB) View All
Repositories	Central
Used By	11,337 artifacts

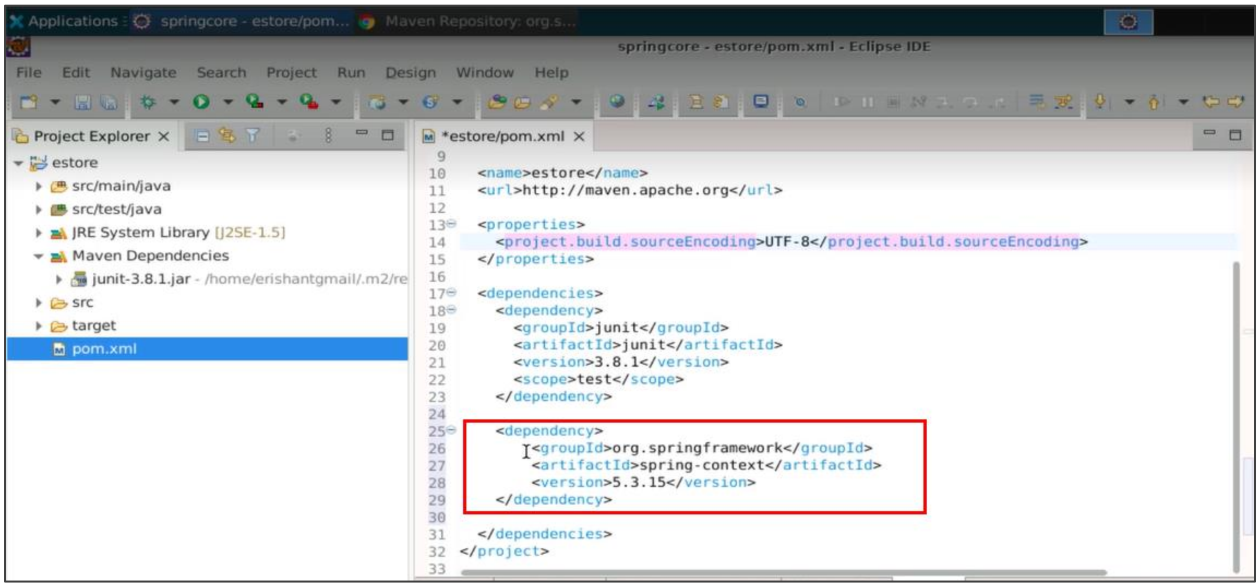
Maven | Gradle | Gradle (Short) | Gradle (Kotlin) | SBT | Ivy | Grape | Leiningen | Buildr

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.15</version>
</dependency>
```

Include comment with link to declaration

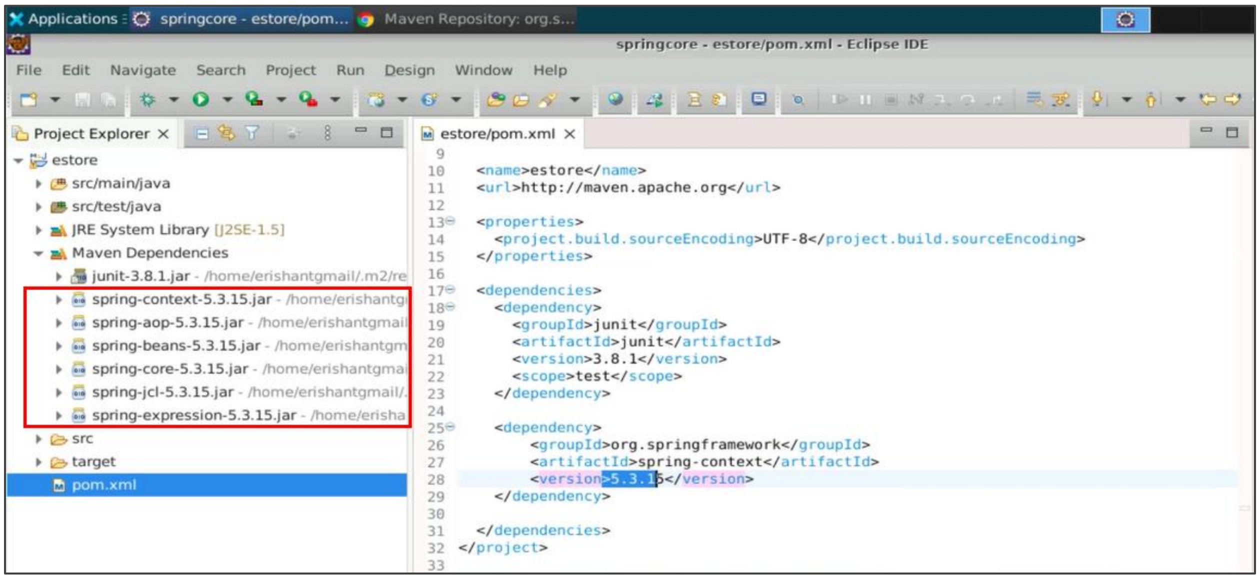
Copied to clipboard!

2.6 Go back to Eclipse and open the pom.xml file. Inside the <dependencies> section, paste the copied dependency for the Spring Context.


springcore - estore/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Design Window Help
Project Explorer
estore
src/main/java
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
junit-3.8.1.jar - /home/erishantgmail/.m2/re
src
target
pom.xml
*estore/pom.xml

```xml
  <name>estore</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.15</version>
    </dependency>

  </dependencies>
</project>
```

2.7 Save the file

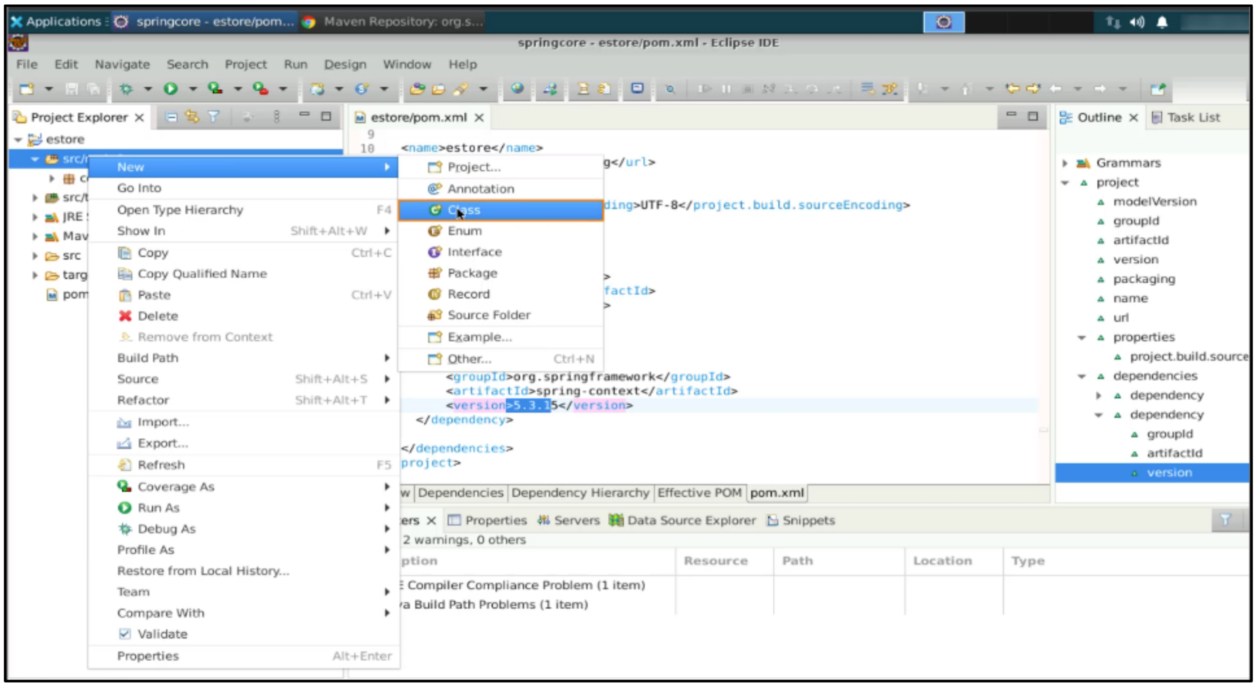

springcore - estore/pom.xml - Eclipse IDE
File Edit Navigate Search Project Run Design Window Help
Project Explorer
estore
src/main/java
src/test/java
JRE System Library [J2SE-1.5]
Maven Dependencies
junit-3.8.1.jar - /home/erishantgmail/.m2/re
spring-context-5.3.15.jar - /home/erishantg
spring-aop-5.3.15.jar - /home/erishantgmail
spring-beans-5.3.15.jar - /home/erishantgm
spring-core-5.3.15.jar - /home/erishantgmai
spring-jcl-5.3.15.jar - /home/erishantgmail/.
spring-expression-5.3.15.jar - /home/erisha
src
target
pom.xml
estore/pom.xml

```xml
  <name>estore</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.15</version>
    </dependency>

  </dependencies>
</project>
```

Maven will automatically sync the Spring Context and its related jar files into your project.

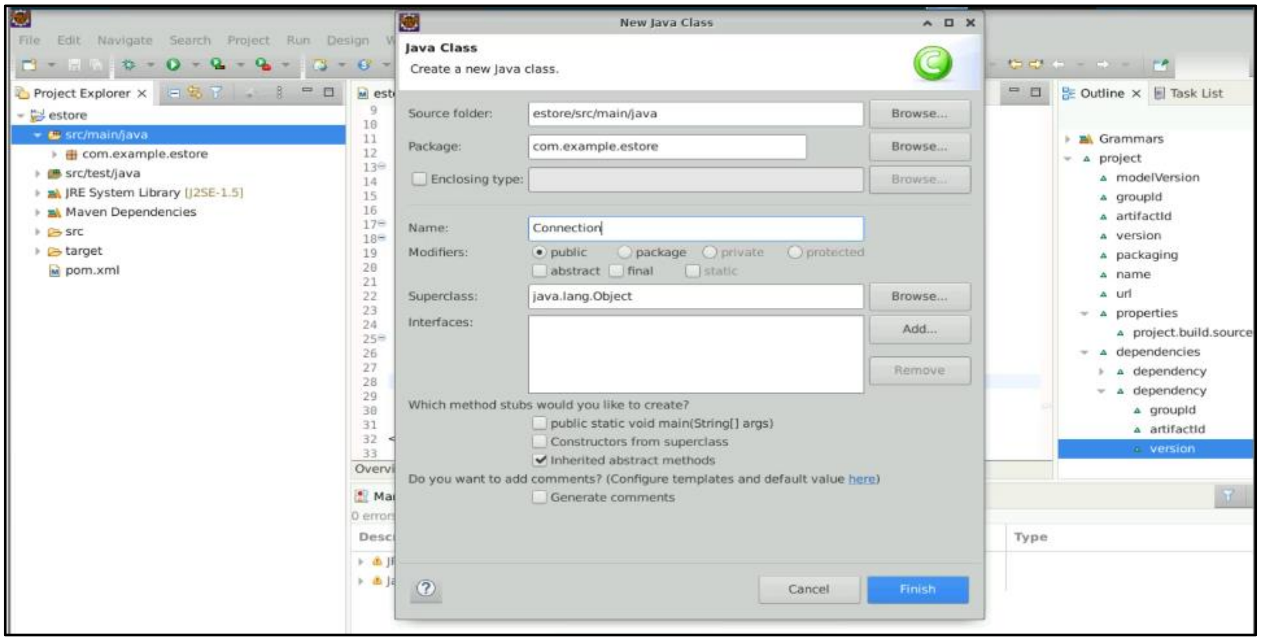
Step 3: Creating a bean class

3.1 In Eclipse, navigate to the Java package where you want to create the bean class. Right-click on the package, select **New**, and click on **Class.**



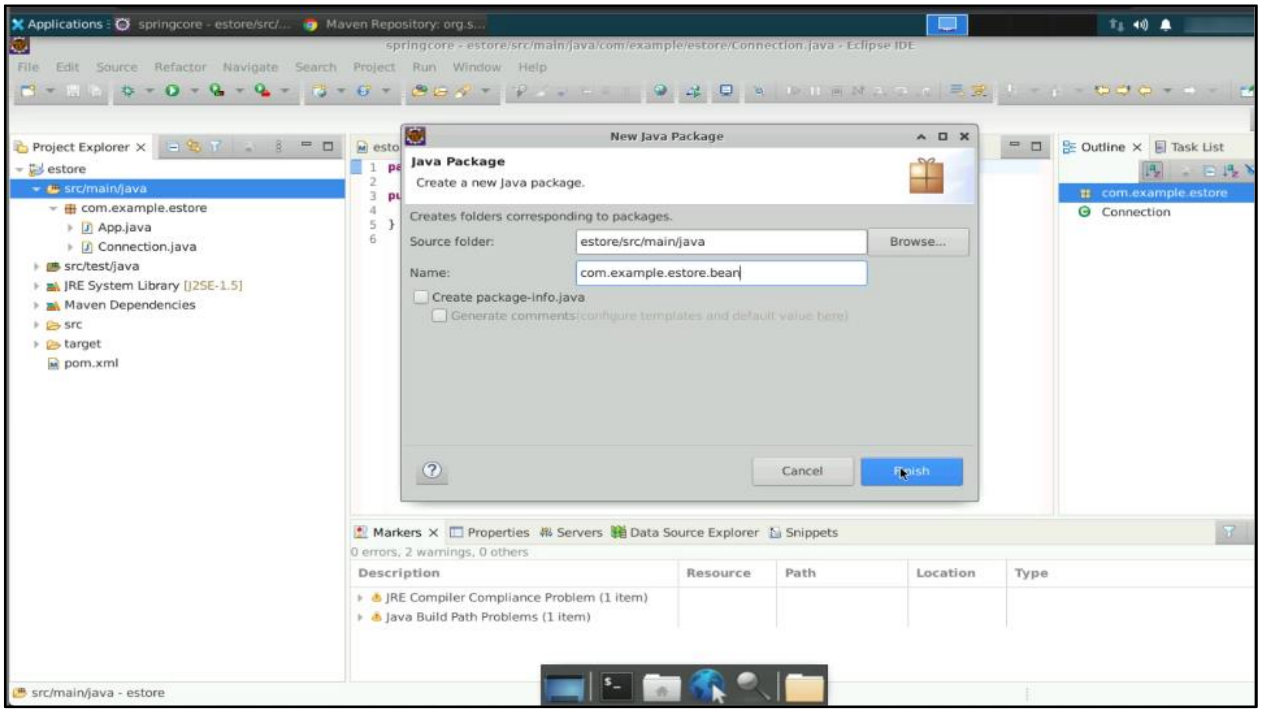
```
<name>estore</name>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>5.3.15</version>
</dependency>
</dependencies>
project>
```

3.2 Enter **Connection** in the **Name** field and click **Finish**



Source folder: estore/src/main/java
Package: com.example.estore
Enclosing type:
Name: Connection
Modifiers: [✓ public] package private protected; abstract final static
Superclass: java.lang.Object
Interfaces:
Which method stubs would you like to create?
public static void main(String[] args)
Constructors from superclass
[✓] Inherited abstract methods
Do you want to add comments? (Configure templates and default value here)
Generate comments

3.3 Move the newly created class to the **com.example.estore.bean** package to represent the **bean** package structure.



New Java Package

Java Package

Create a new Java package.

Creates folders corresponding to packages.

Source folder: estore/src/main/java

Name: com.example.estore.bean

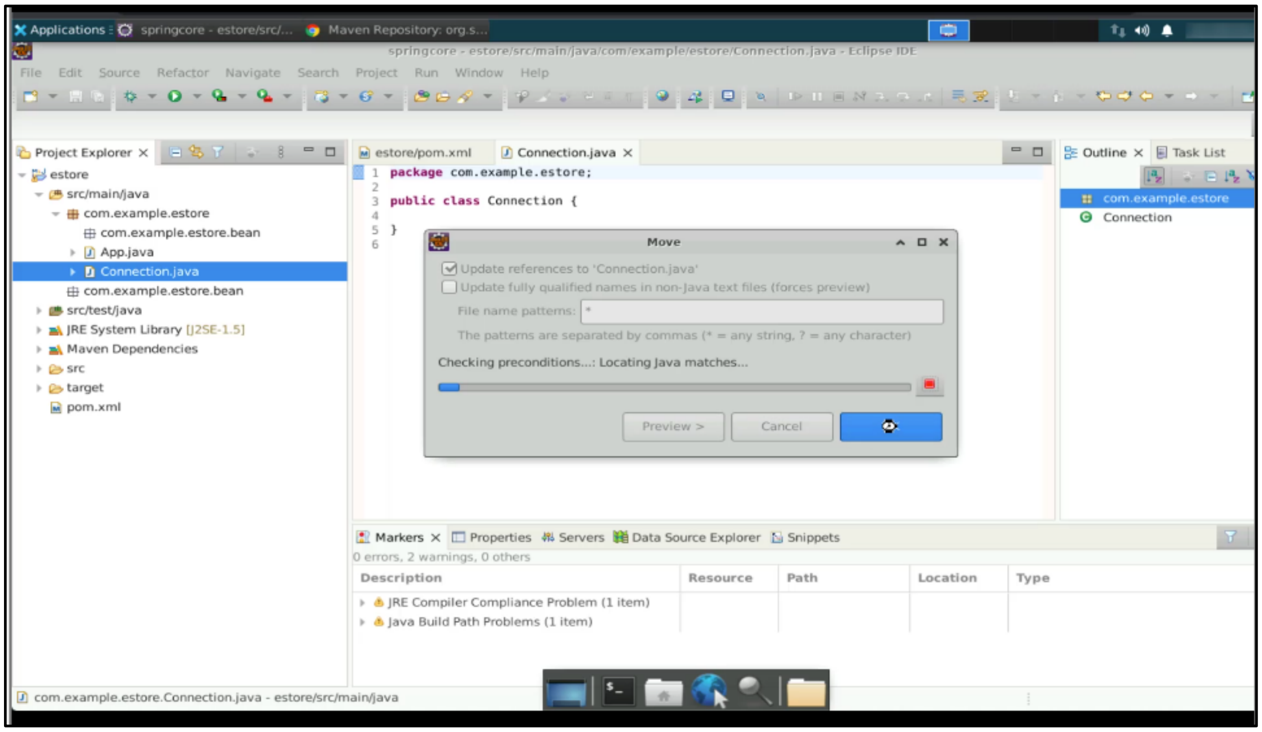
Create package-info.java

Generate comments(configure templates and default value here)

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
JRE Compiler Compliance Problem (1 item)				
Java Build Path Problems (1 item)				

src/main/java - estore



```
package com.example.estore;

public class Connection {

}
```

Move

Update references to 'Connection.java'

Update fully qualified names in non-Java text files (forces preview)

File name patterns: *

The patterns are separated by commas (* = any string, ? = any character)

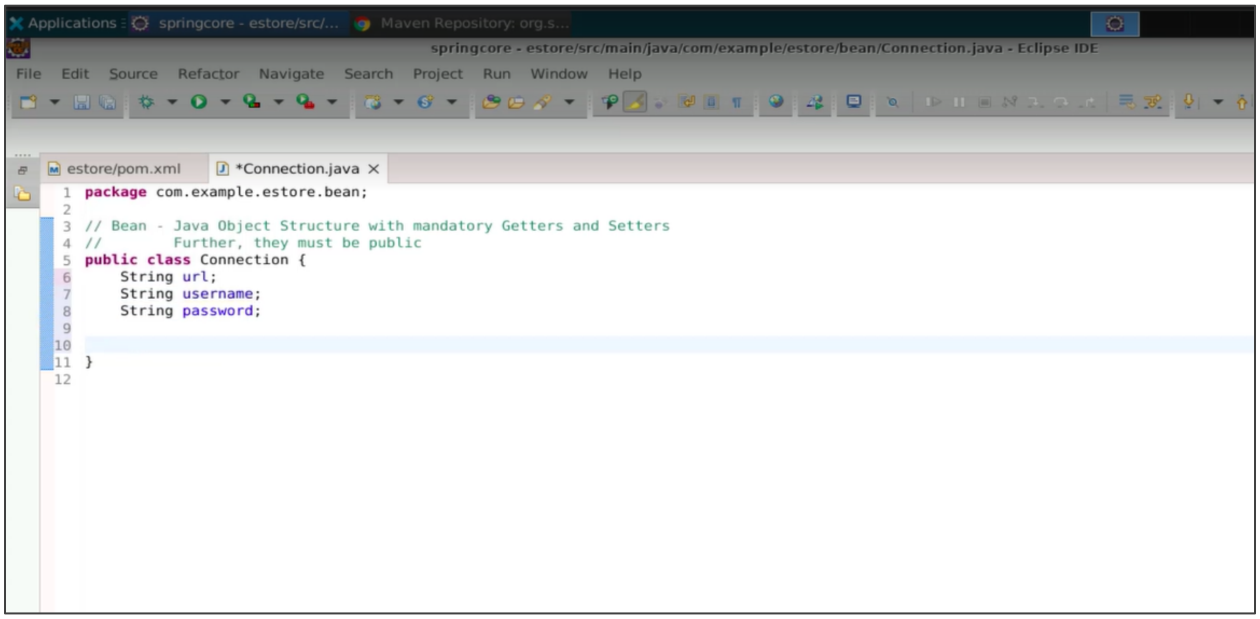
Checking preconditions...: Locating Java matches...

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
JRE Compiler Compliance Problem (1 item)				
Java Build Path Problems (1 item)				

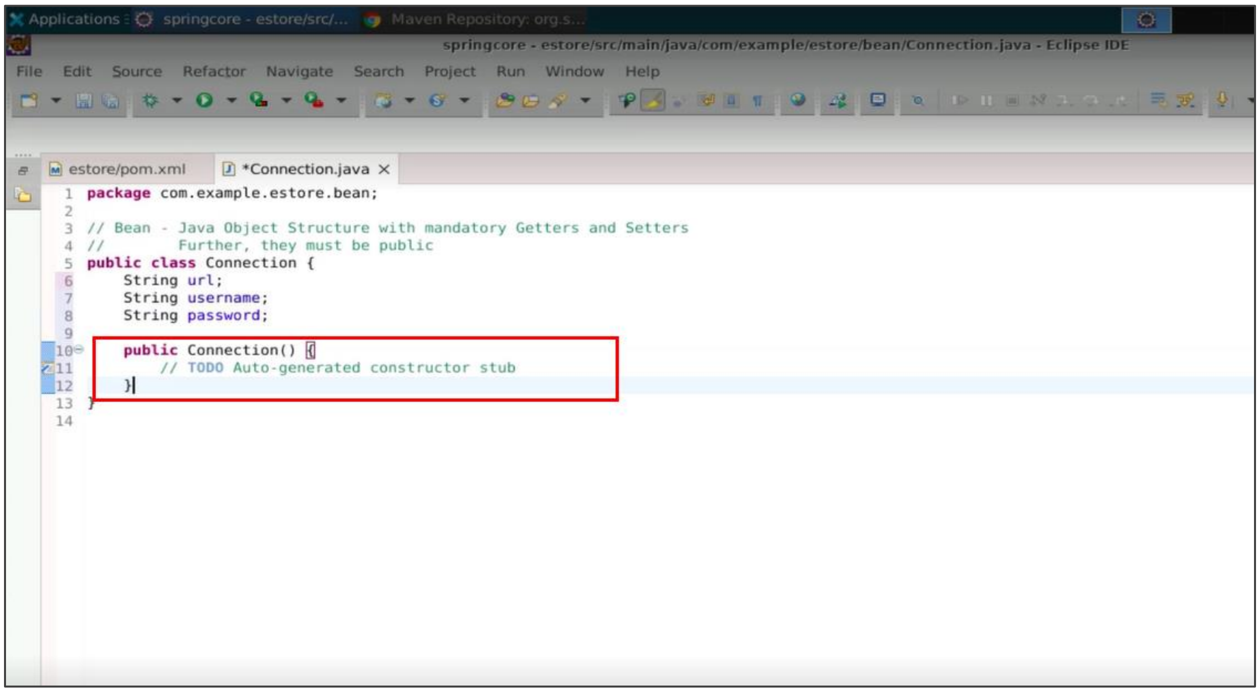
com.example.estore.Connection.java - estore/src/main/java

3.4 Inside the **Connection** class, define the necessary attributes such as **url, username**, and **password**


Applications
springcore - estore/src/...
Maven Repository: org.s...
springcore - estore/src/main/java/com/example/estore/bean/Connection.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
estore/pom.xml
*Connection.java

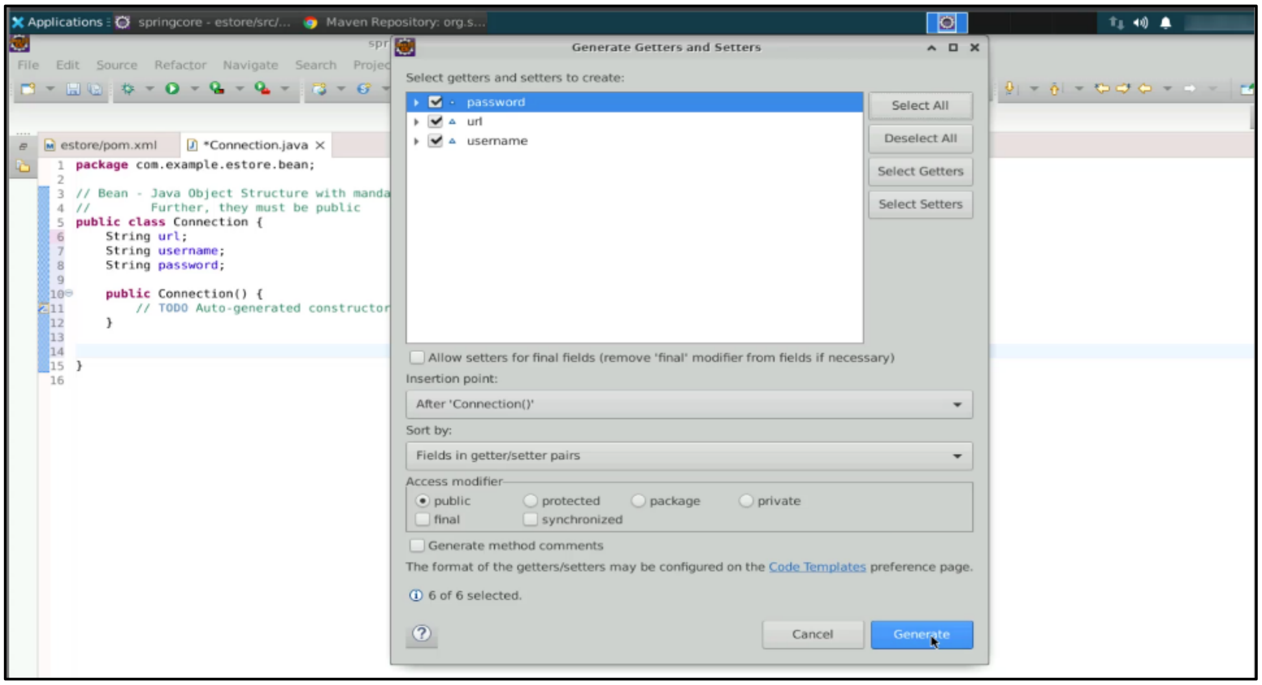
```java
package com.example.estore.bean;

// Bean - Java Object Structure with mandatory Getters and Setters
//        Further, they must be public
public class Connection {
    String url;
    String username;
    String password;


}
```

3.5 Create a default constructor for the class to initialize the attributes


Applications
springcore - estore/src/...
Maven Repository: org.s...
springcore - estore/src/main/java/com/example/estore/bean/Connection.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
estore/pom.xml
*Connection.java

```java
package com.example.estore.bean;

// Bean - Java Object Structure with mandatory Getters and Setters
//        Further, they must be public
public class Connection {
    String url;
    String username;
    String password;

    public Connection() {
        // TODO Auto-generated constructor stub
    }
}
```

3.6 Generate getters and setters for the attributes and implement a **toString()** method for the class.



```
package com.example.estore.bean;

// Bean - Java Object Structure with manda
//       Further, they must be public
public class Connection {
    String url;
    String username;
    String password;

    public Connection() {
        // TODO Auto-generated constructor
    }

}
```

Generate Getters and Setters

Select getters and setters to create:

- [✓] password
- [✓] url
- [✓] username

[ ] Allow setters for final fields (remove 'final' modifier from fields if necessary)

Insertion point: After 'Connection()'

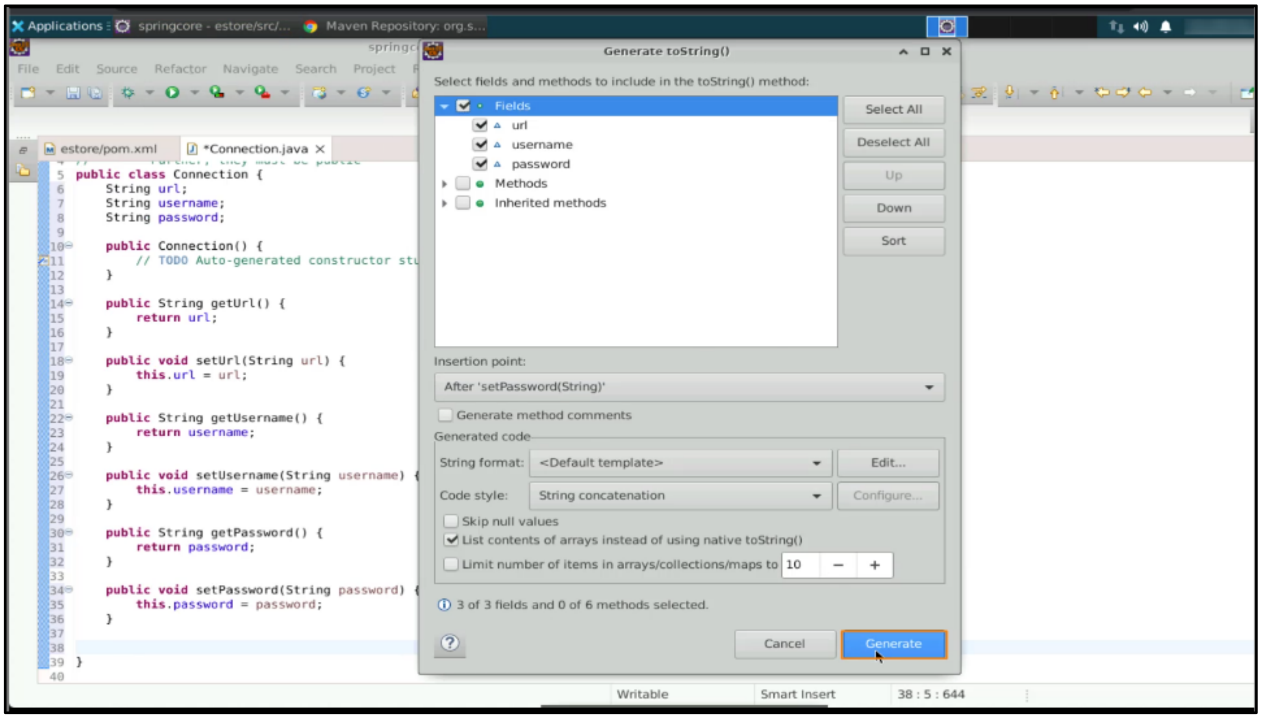
Sort by: Fields in getter/setter pairs

Access modifier: [✓ public] protected package private; [ ] final [ ] synchronized

[ ] Generate method comments

The format of the getters/setters may be configured on the Code Templates preference page.

6 of 6 selected.



```
public class Connection {
    String url;
    String username;
    String password;

    public Connection() {
        // TODO Auto-generated constructor stu
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

}
```

Generate toString()

Select fields and methods to include in the toString() method:

- [✓] Fields
  - [✓] url
  - [✓] username
  - [✓] password
- [ ] Methods
- [ ] Inherited methods

Insertion point: After 'setPassword(String)'

[ ] Generate method comments

Generated code

String format: <Default template>

Code style: String concatenation

[ ] Skip null values

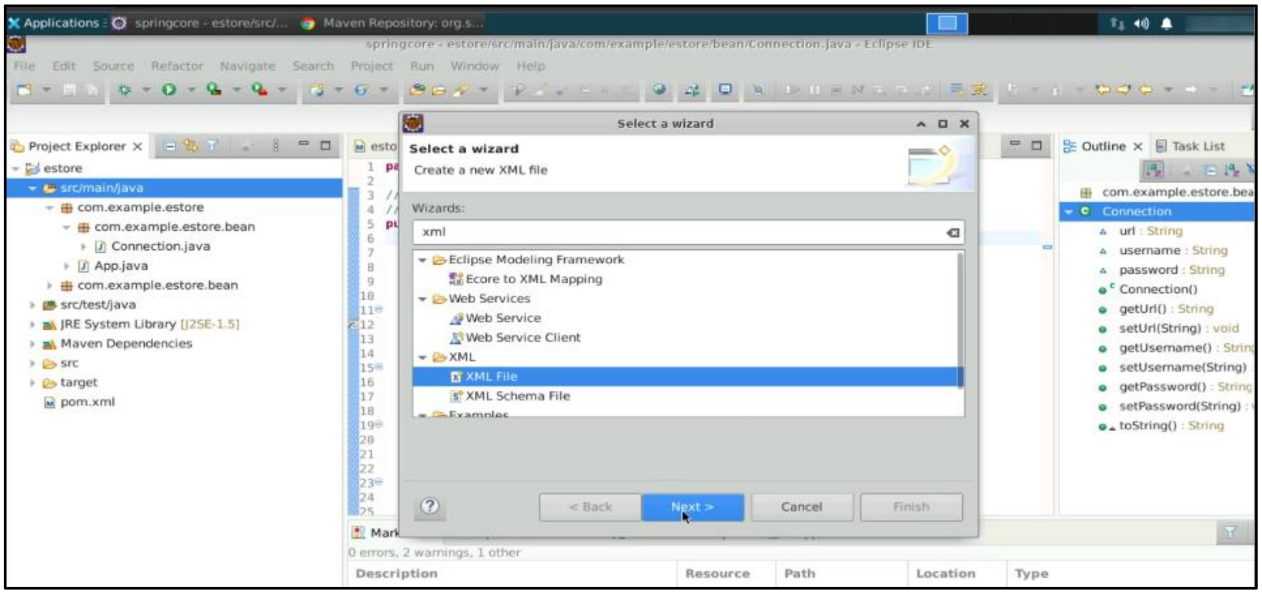
[✓] List contents of arrays instead of using native toString()

[ ] Limit number of items in arrays/collections/maps to 10

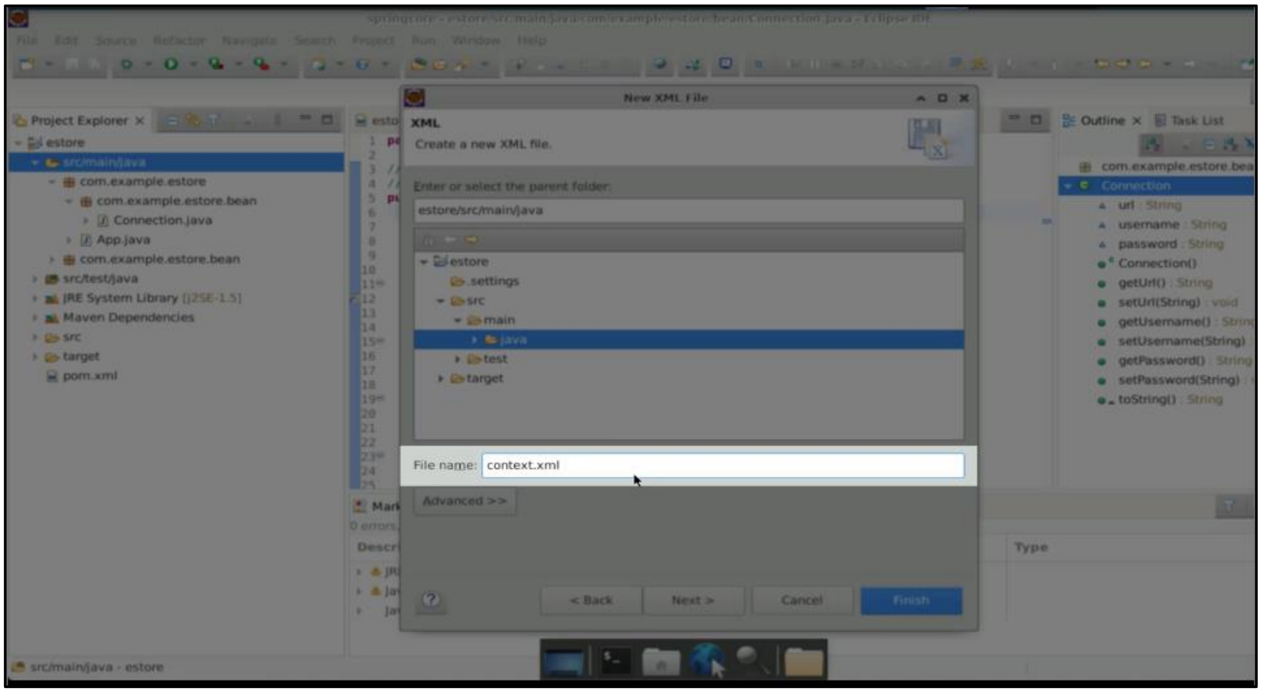
3 of 3 fields and 0 of 6 methods selected.

Step 4: Configuring XML file

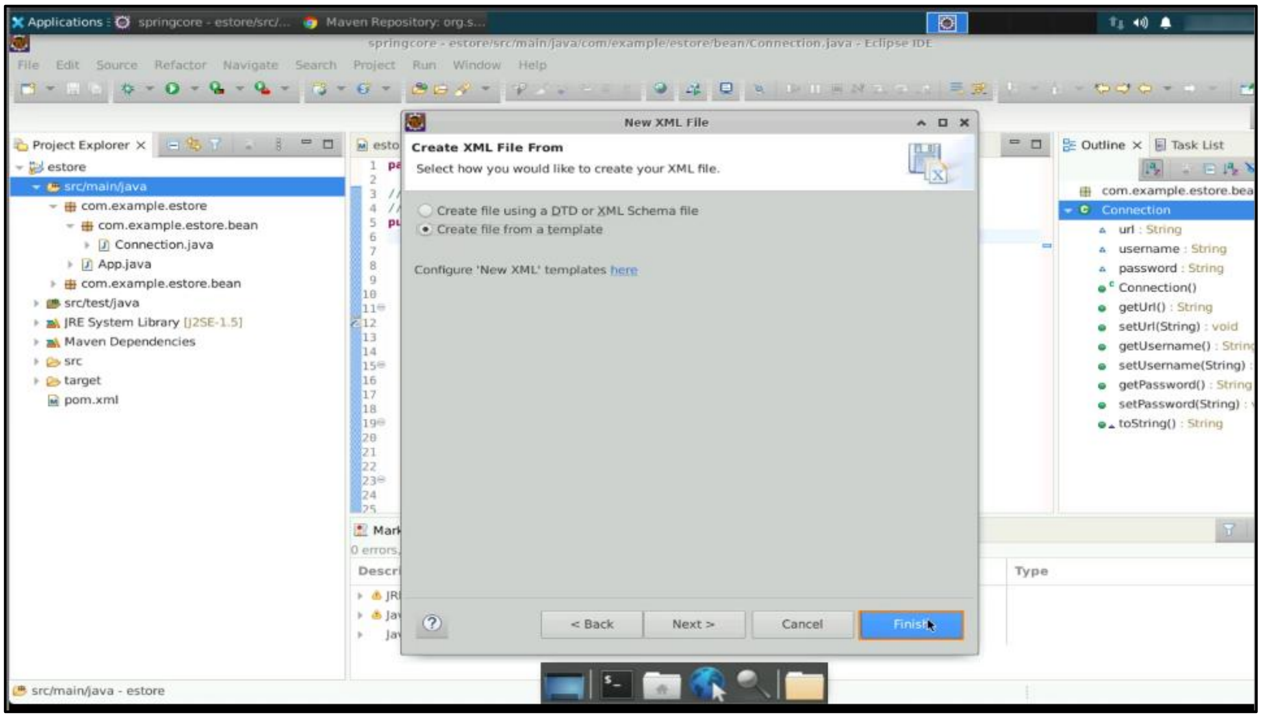
4.1 Right-click on the **src** folder in the project and select **New** > **Other**. In the **New dialog**, select **XML File** under the XML category and click **Next**



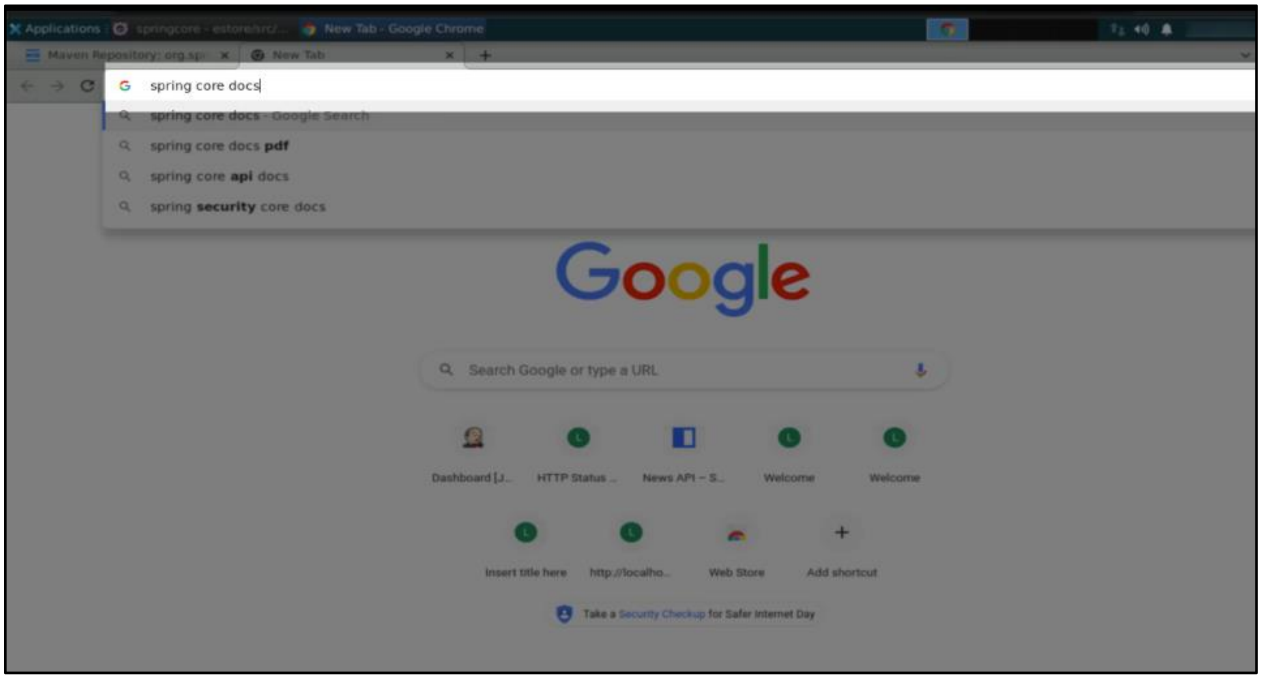
4.2 Enter **context.xml** in the File name field



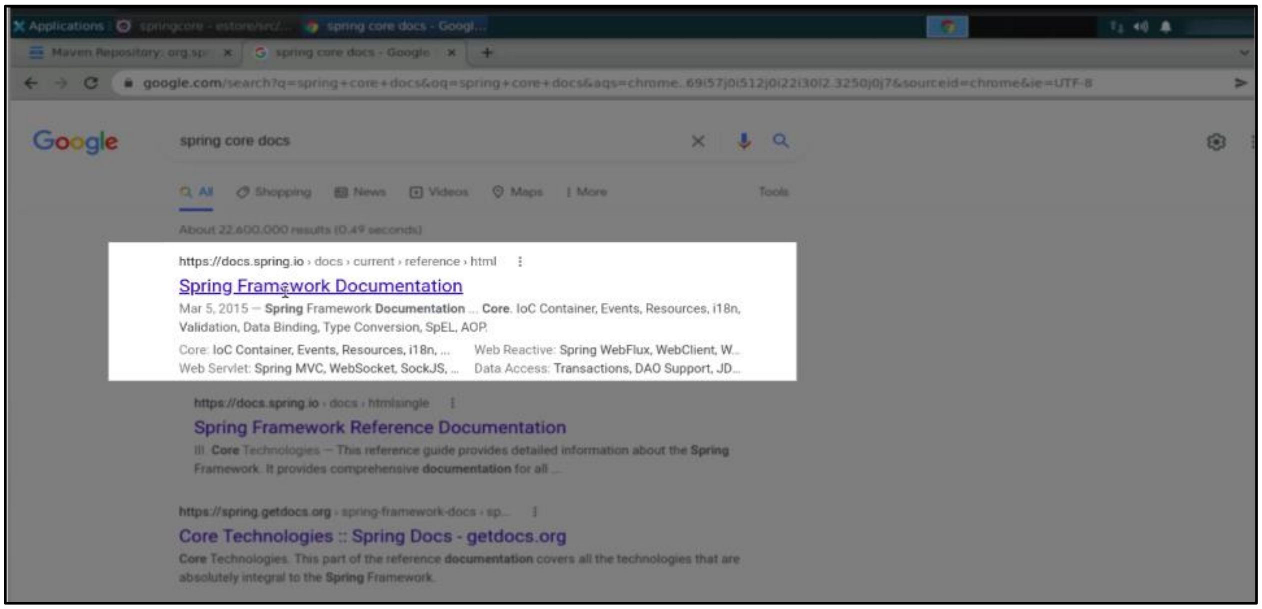
4.3 Click **Finish** to create the XML file


New XML File
Create XML File From
Select how you would like to create your XML file.
Create file using a DTD or XML Schema file
Create file from a template
Configure 'New XML' templates here
< Back
Next >
Cancel
Finish

4.4 Return to the browser and search for the Spring Core documentation


spring core docs
spring core docs - Google Search
spring core docs pdf
spring core api docs
spring security core docs
Google
Search Google or type a URL

4.5 Select the first link from the search results to access the documentation.



spring core docs

About 22,600,000 results (0.49 seconds)

https://docs.spring.io › docs › current › reference › html

Spring Framework Documentation

Mar 5, 2015 — Spring Framework Documentation ... Core. IoC Container, Events, Resources, i18n, Validation, Data Binding, Type Conversion, SpEL, AOP.

Core: IoC Container, Events, Resources, i18n, ... Web Reactive: Spring WebFlux, WebClient, W...
Web Servlet: Spring MVC, WebSocket, SockJS, ... Data Access: Transactions, DAO Support, JD...

https://docs.spring.io › docs › htmlsingle

Spring Framework Reference Documentation

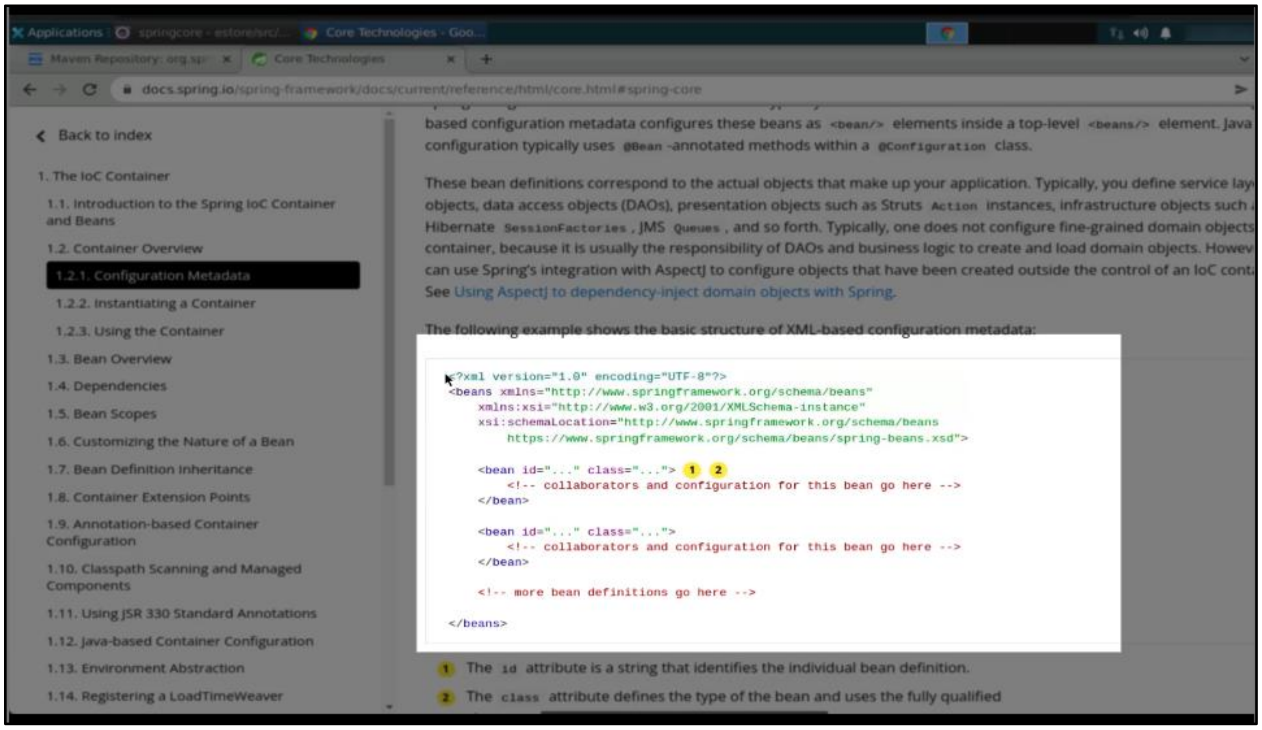
III. Core Technologies — This reference guide provides detailed information about the Spring Framework. It provides comprehensive documentation for all ...

https://spring.getdocs.org › spring-framework-docs › sp...

Core Technologies :: Spring Docs - getdocs.org

Core Technologies. This part of the reference documentation covers all the technologies that are absolutely integral to the Spring Framework.

4.6 Copy the sample XML configuration provided in the documentation.



docs.spring.io/spring-framework/docs/current/reference/html/core.html#spring-core

based configuration metadata configures these beans as `<bean/>` elements inside a top-level `<beans/>` element. Java configuration typically uses `@Bean` -annotated methods within a `@Configuration` class.

These bean definitions correspond to the actual objects that make up your application. Typically, you define service lay objects, data access objects (DAOs), presentation objects such as Struts `Action` instances, infrastructure objects such Hibernate `SessionFactories` , JMS `Queues` , and so forth. Typically, one does not configure fine-grained domain objects container, because it is usually the responsibility of DAOs and business logic to create and load domain objects. Howev can use Spring's integration with AspectJ to configure objects that have been created outside the control of an IoC cont See Using AspectJ to dependency-inject domain objects with Spring.

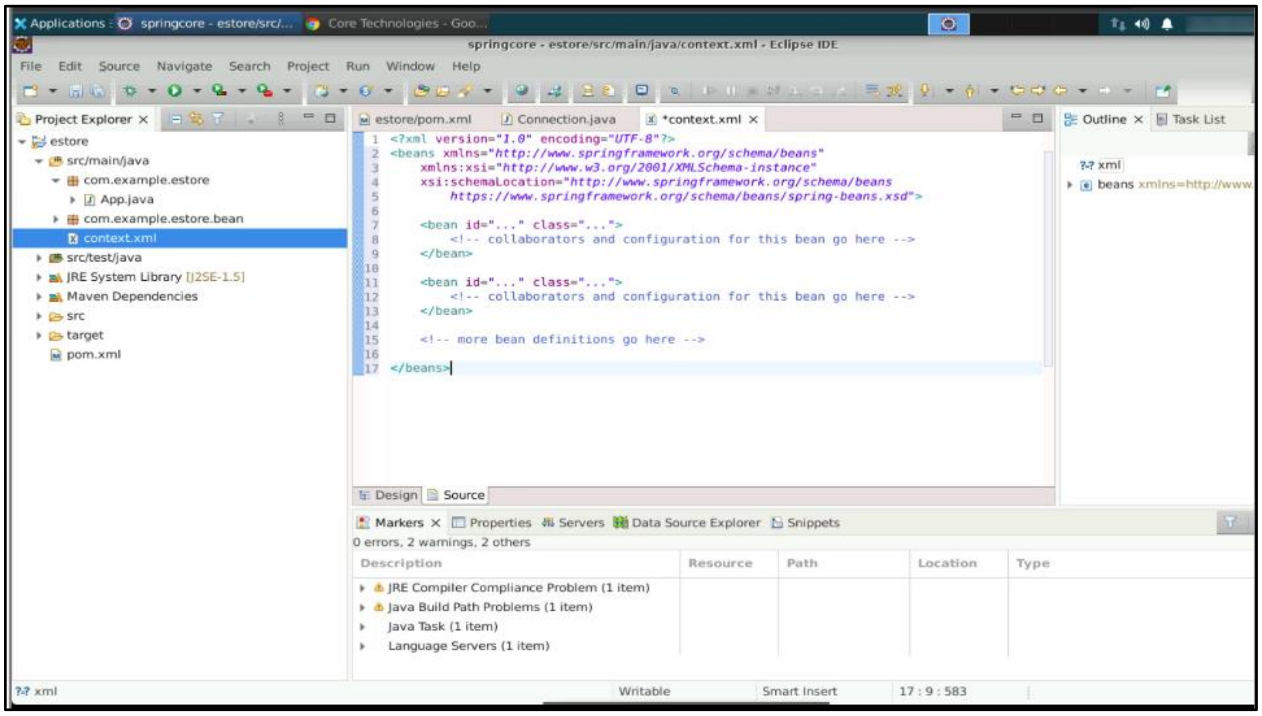
The following example shows the basic structure of XML-based configuration metadata:

```
<?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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="..." class="..."> (1) (2)
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <bean id="..." class="...">
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

1 The `id` attribute is a string that identifies the individual bean definition.

2 The `class` attribute defines the type of the bean and uses the fully qualified

4.7 Paste the XML configuration into the newly created XML file in Eclipse.



```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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="..." class="...">
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <bean id="..." class="...">
        <!-- collaborators and configuration for this bean go here -->
    </bean>

    <!-- more bean definitions go here -->

</beans>
```

```
Markers  Properties  Servers  Data Source Explorer  Snippets
0 errors, 2 warnings, 2 others
Description                                    Resource   Path   Location   Type
▸ JRE Compiler Compliance Problem (1 item)
▸ Java Build Path Problems (1 item)
▸ Java Task (1 item)
▸ Language Servers (1 item)
```

You have now successfully configured the Spring Core in your Java project.