INTERNET TECHNOLOGIES AND WEB SERVICES

Final Project

Spring framework application

VinyLand   
Vinyl record store

Student: Nina Krstovic SDE 2018230089

Introduction

For the final project I decided to create a spring framework application for a vinyl record store called VinyLand. It functions by listing various information that is stored in different POJO classes, interconnected between themselves via the beans.xml file. The app is made to serve as an Introduction page of a website.

Technology

As previously stated in the introduction, for the development of this project I used the spring framework. Spring framework gives the application the infrastructure support that it needs in order to better its performance all while using plain old java objects also known as POJOs.

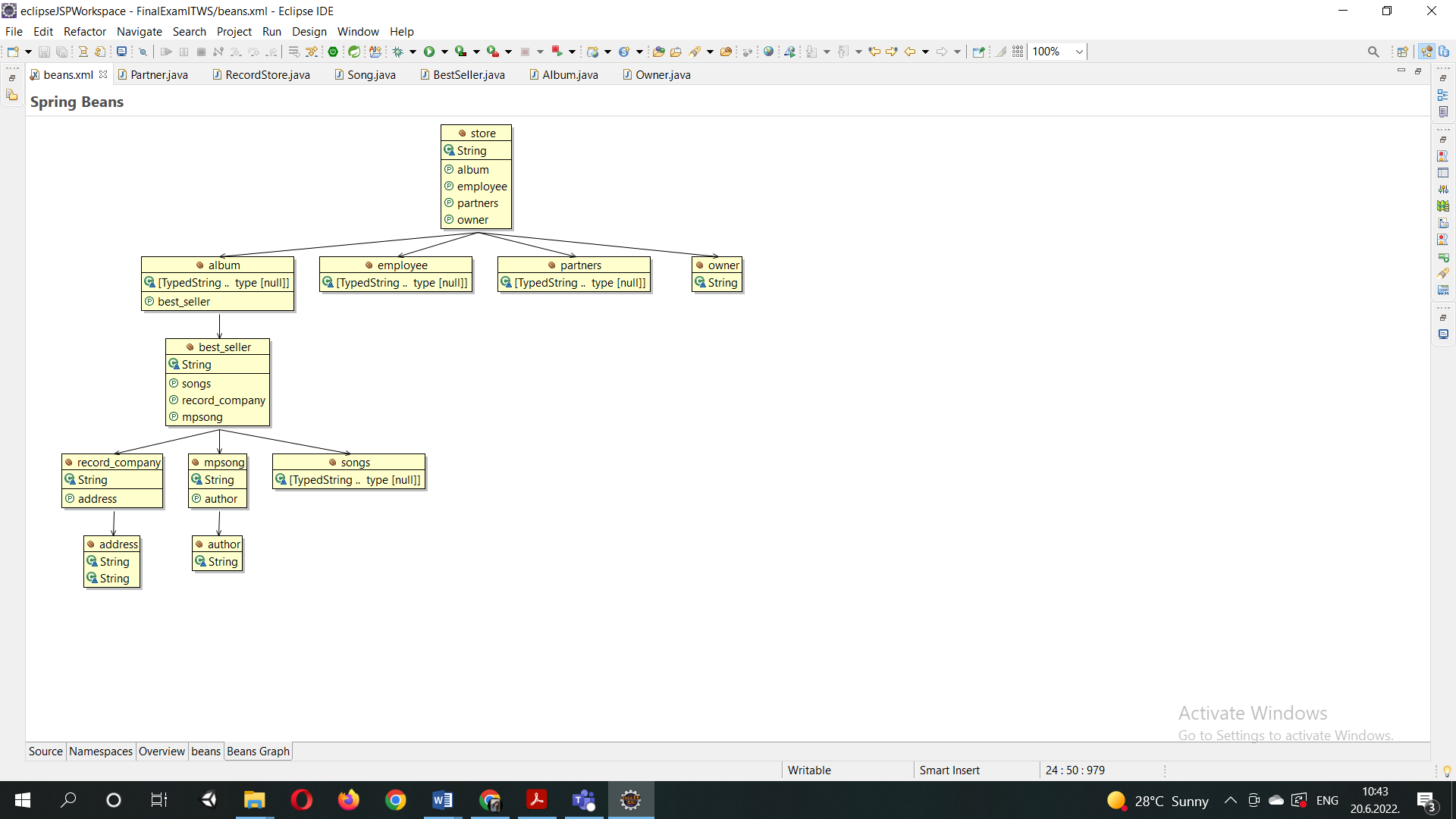
POJO classes are simple, not bound with restrictions and do now extend/implement any other class or interface. The only thing they contain are properties (i.e. id, name) and getters and setters for those properties.

Since POJO classes do not extend or implement anything, in order to connect them to other classes we use beans contained within an .xml file of the same name. Beans are therefore the backbone of the application itself serving as a recipe for class instance creations. This is all done through the dependency injection, types of which are used within this application (property injection, object injection, constructor injection, setter injection).

Property injection as the name itself suggests refers to the injection of dependencies correlating with the properties defined within the class in question. Object injection on the other hand, refers to dependencies relating to another separate POJO class (i.e. album class in our case is dependent on the song class).

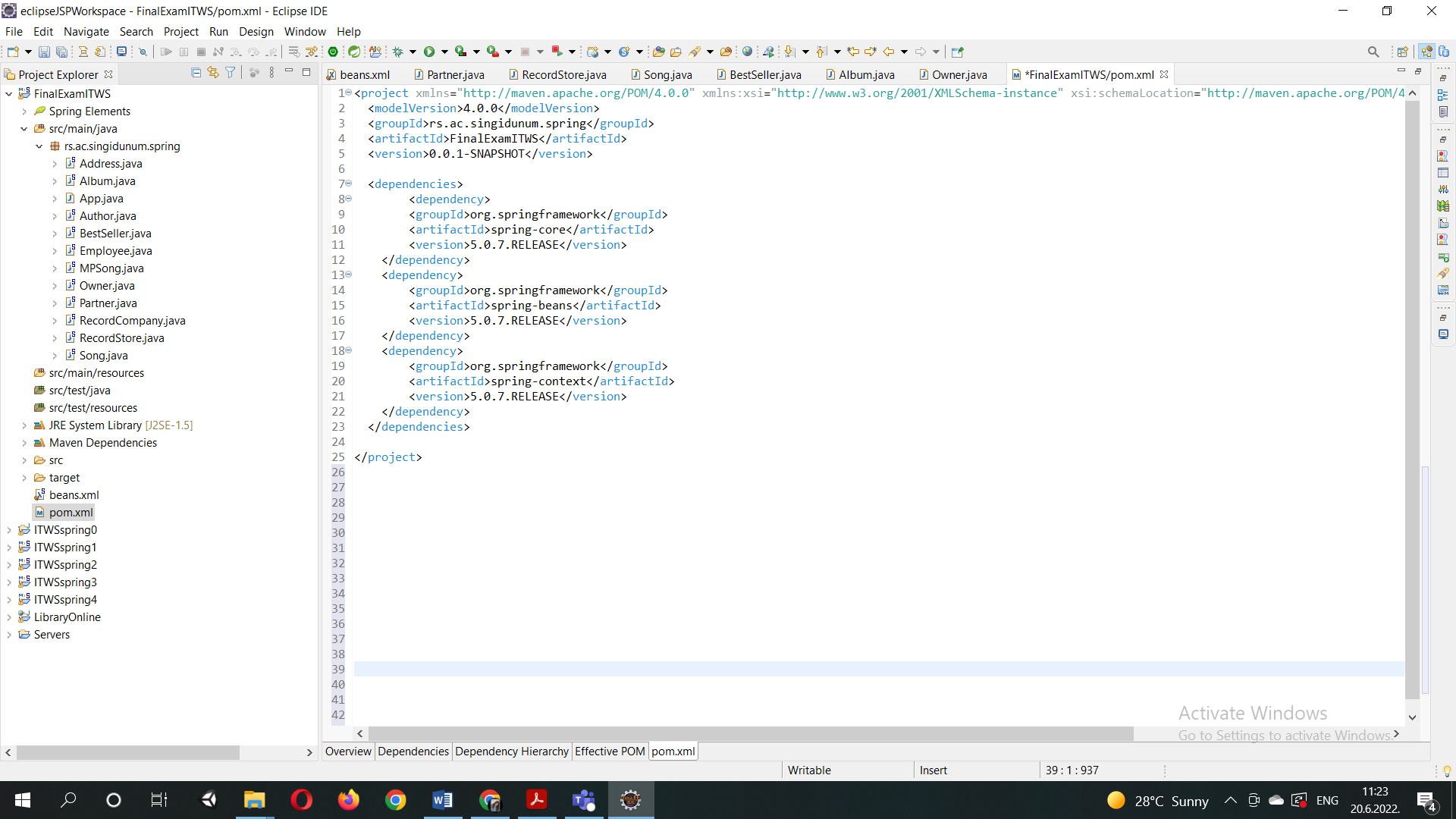
Constructor injection happens when the container invokes a class constructor with a number of arguments, each respectively representing a dependency.

Setter injection uses the setter method on an object in order to create the dependencies.   
  
 To clear up the dependencies whose implementation will be shown later, here is the schema of all the object classes within the application and how they are connected:

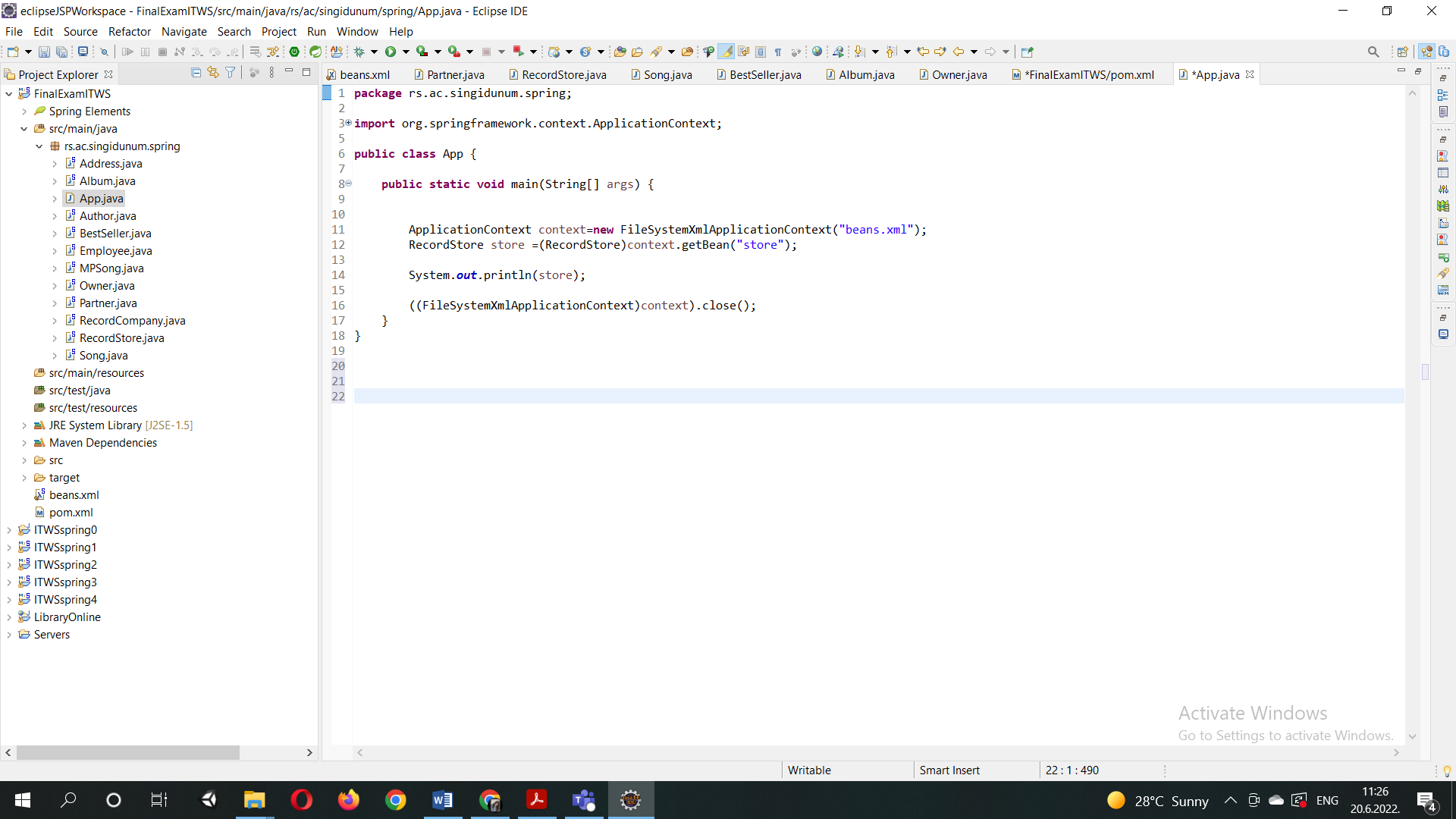


IMPLEMENTATION

Before starting any code we must first define the spring dependencies for the whole project. This is done within the pom.xml file:

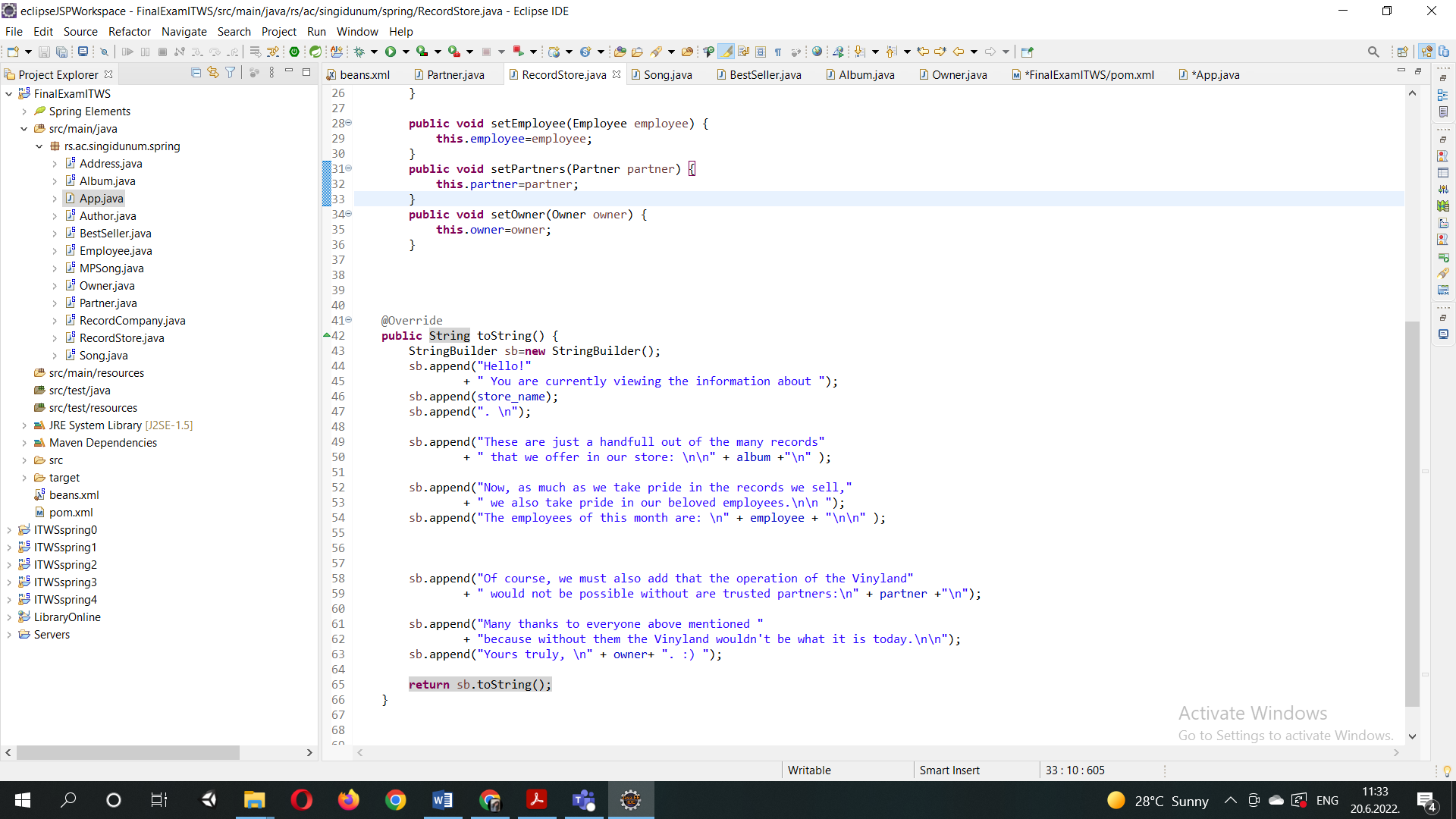
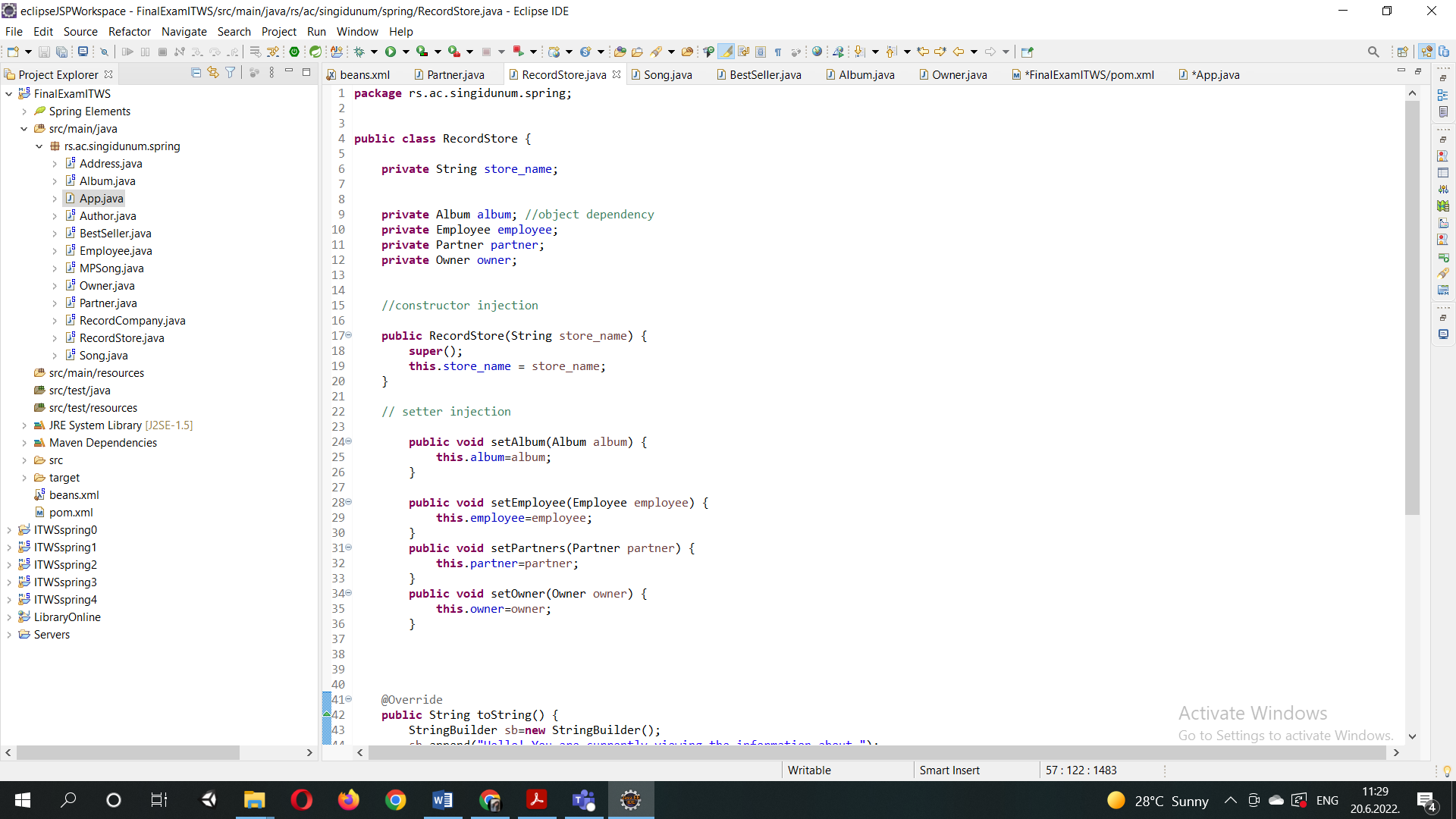


After that has been done we can begin creating are classes. First we will create a runnable application class that will fetch the context of the other classes (in our case the main class called RecordStore).

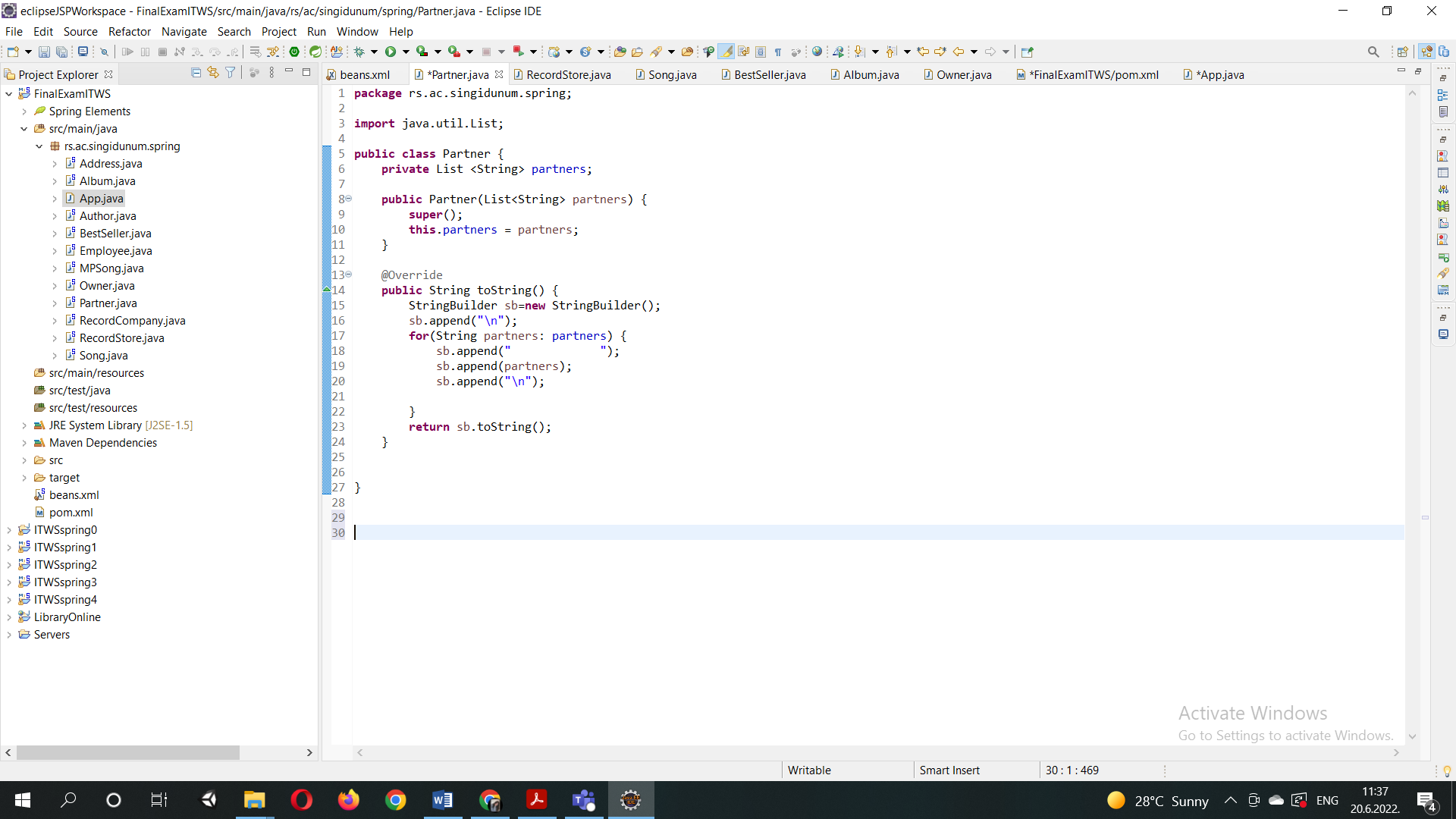
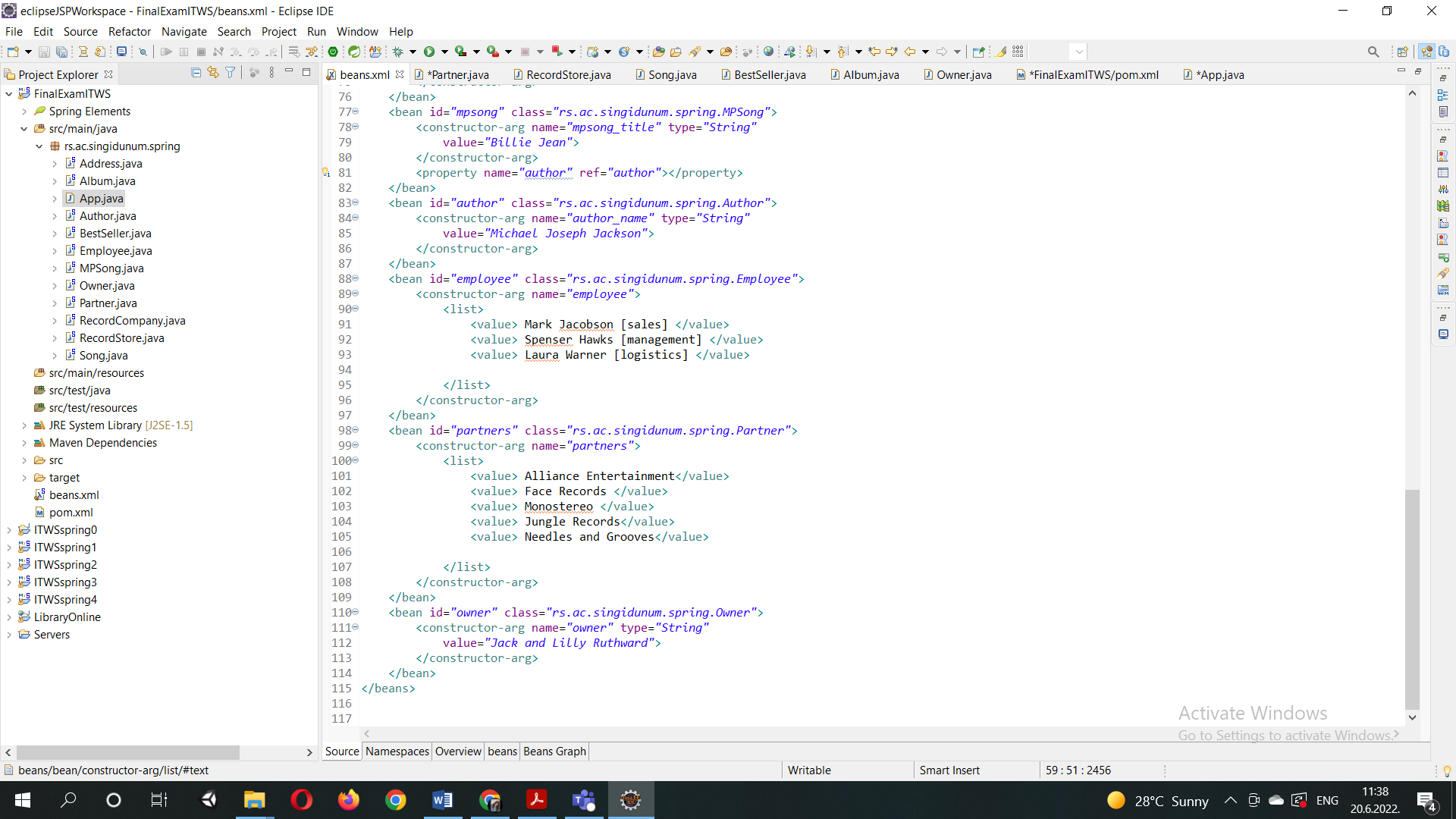


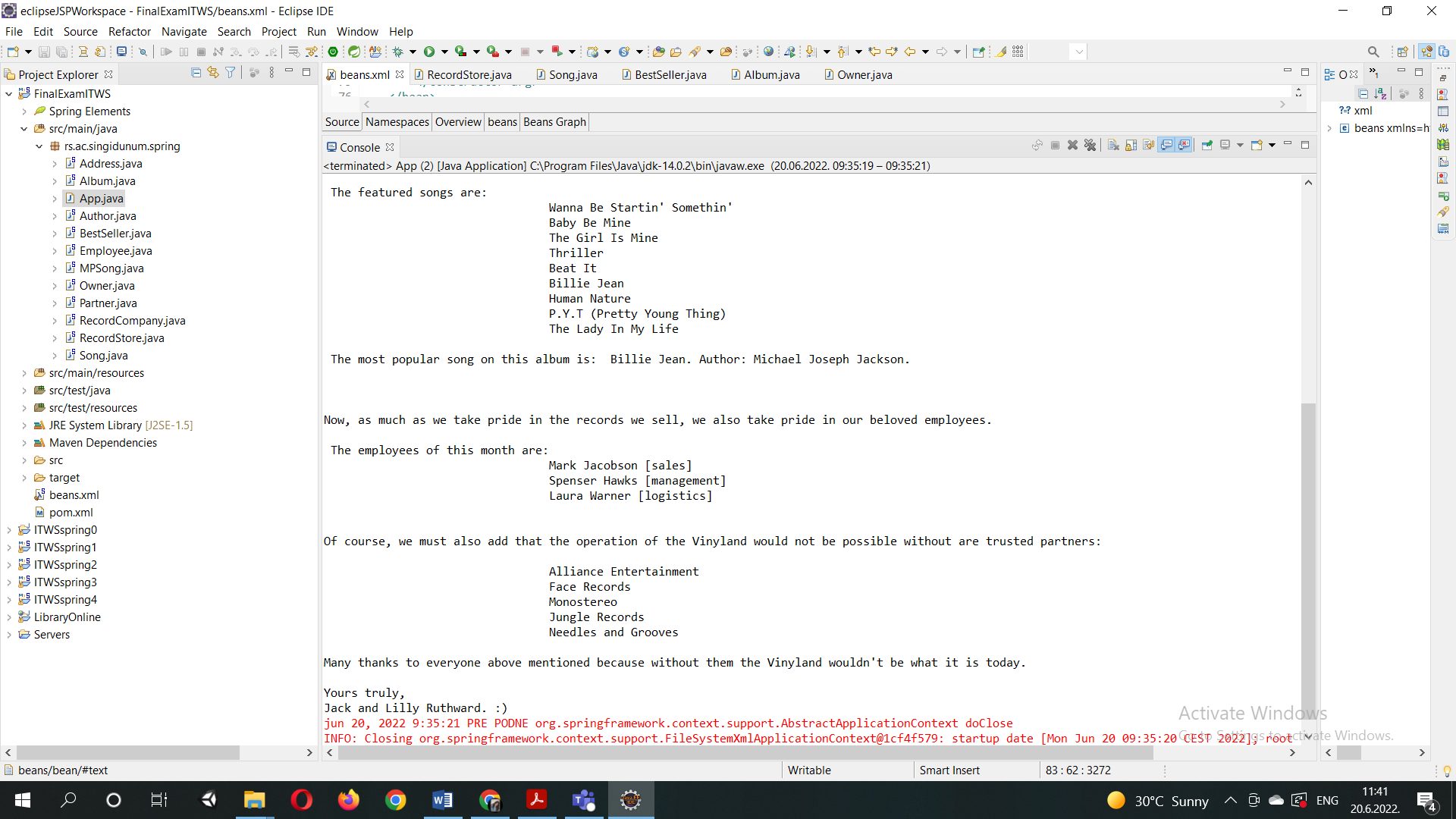
Then the creation of the POJO classes can begin and beans can begin. For the sake of an explanation but also brevity, I will only provide one example of a POJO class in this document.

Here is the RecordStore POJO class. We can see all the dependancy injections previosly mentioned, as well as how the dependecies are called within the toString method to pull data from other classes, such as album, employee, owner and partner classes in this case.



Now if we take a look inside one of those classes, partners for example, and its corresponding bean configuration:



 It is perfectly clear why and how we get the following output:

This is the manner in which all the POJO classes shown in the diagram above are connected.

Conclusion and future upgrades

To conclude, although the application is fairly simple and does not contain any frontend configuration, with spring we can still create something that makes sense and can serve a purpose.

Of course, if the application were to be developed further, .html views would need to be defined, the implementation of other classes could be done as well, such as different types of employees, rental for the store, payments etc. Also we could use spring boot to connect our application to the database from which we could read certain information.