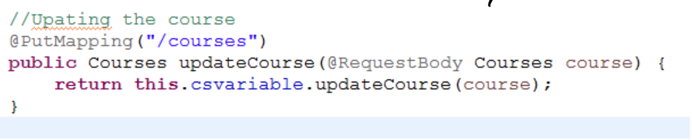
24 January 2023 14:12

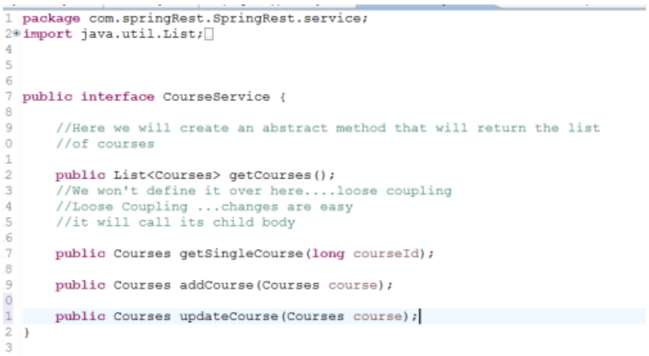
// Creating Controller $\boxed{\text{Put}}$

Put: we will update

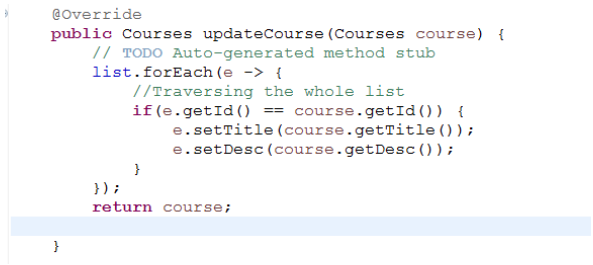
1. Add the controller
2. Adding the abstract method in service interface.
3. Creating Method in the implentation class



```
//Upating the course
@PutMapping("/courses")
public Courses updateCourse(@RequestBody Courses course) {
    return this.csvariable.updateCourse(course);
}
```



```
package com.springRest.SpringRest.service;
import java.util.List;



public interface CourseService {

    //Here we will create an abstract method that will return the list
    //of courses

    public List<Courses> getCourses();
    //We won't define it over here....loose coupling
    //Loose Coupling ...changes are easy
    //it will call its child body

    public Courses getSingleCourse(long courseId);

    public Courses addCourse(Courses course);

    public Courses updateCourse(Courses course);
}
```



```
    @Override
    public Courses updateCourse(Courses course) {
        // TODO Auto-generated method stub
        list.forEach(e -> {
            //Traversing the whole list
            if(e.getId() == course.getId()) {
                e.setTitle(course.getTitle());
                e.setDesc(course.getDesc());
            }
        });
        return course;

    }
```

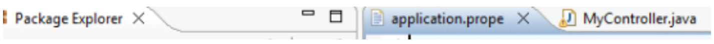
Whenever we are creating APIs, we need to return HTTP Status response like 200, 404 etc.

We can use: ResponseEntity<HttpStatus)

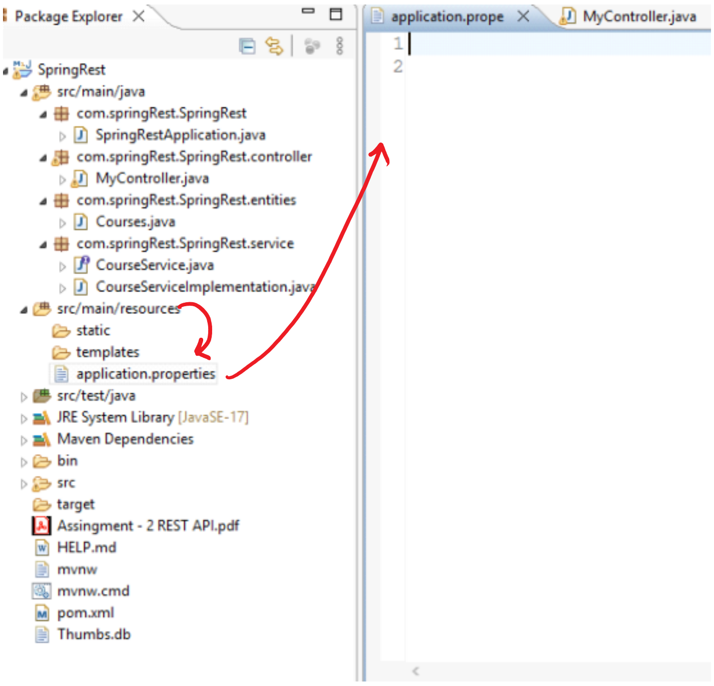
Now we need to connect with dbs.

Work over: DOA layer (ref: J2EE Architecture)

To configure anything on spring boot →
You need to open application. properties file.



In properties file we



In properties file we write key:value

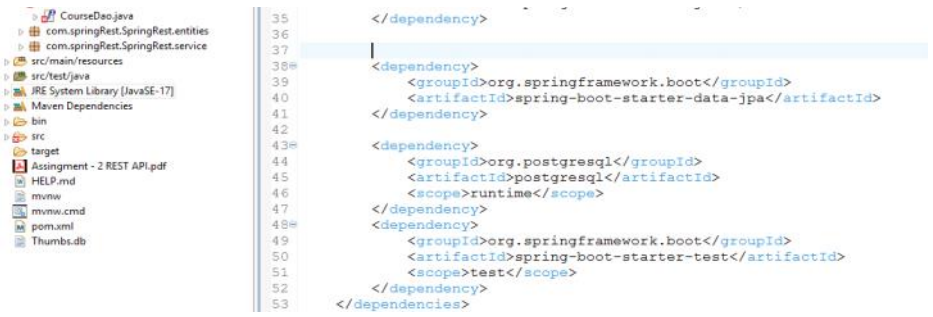
we can change our port here as well.

server.port = 9090 (eg)

To add comment -> #

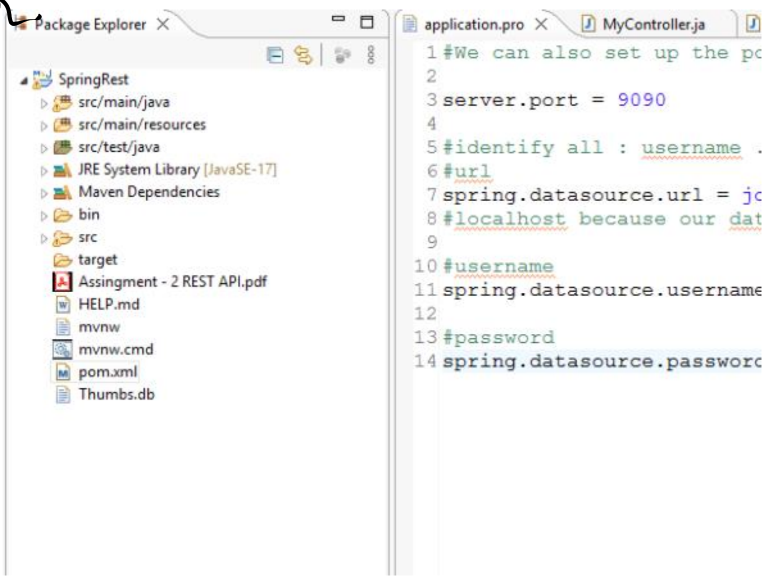
(or use ctrl + /)

check whether you have correct dependancies.


CourseDao.java
com.springRest.SpringRest.entities
com.springRest.SpringRest.service
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
bin
src
target
Assingment - 2 REST API.pdf
HELP.md
mvnw
mvnw.cmd
pom.xml
Thumbs.db

```xml
        </dependency>


        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>

        <dependency>
            <groupId>org.postgresql</groupId>
            <artifactId>postgresql</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

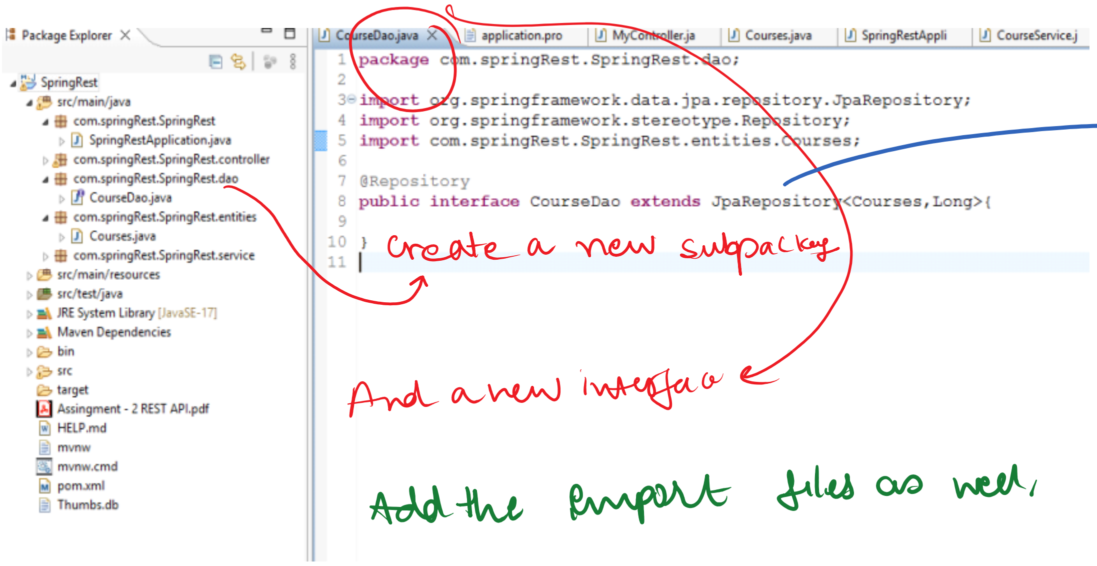
Now after setting the application propelties we will create dao layer.



```properties
#We can also set up the pc

server.port = 9090

#identify all : username .
#url
spring.datasource.url = jc
#localhost because our dat

#username
spring.datasource.username

#password
spring.datasource.passworc
```

Create a subpackage in your src/main/java and

Create new interface in it. For DAO layer



```java
package com.springRest.SpringRest.dao;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.springRest.SpringRest.entities.Courses;

@Repository
public interface CourseDao extends JpaRepository<Courses,Long>{

}
```

Create a new subpackage

And a new interface

Add the import files as well,

Sometimes this doesn't works -- so force update in that case.

If we don't use JPA , we have to do a lot of things …..we have to do all things manually like creating interfaces and them completing them

It takes two things :

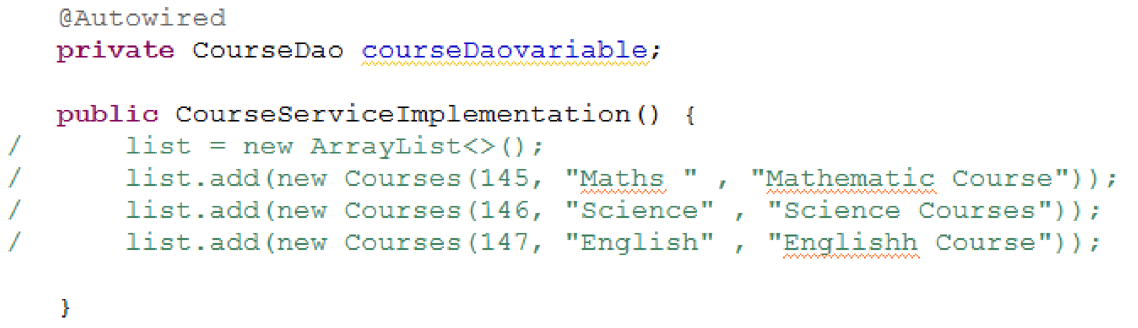
1. Entity you are dealing with.
2. Type of the primary key of that entity

Now it has all the inbuilt methods to do the work.

@Repository
public interface CourseDao extends JpaRepository<Courses,Long>{

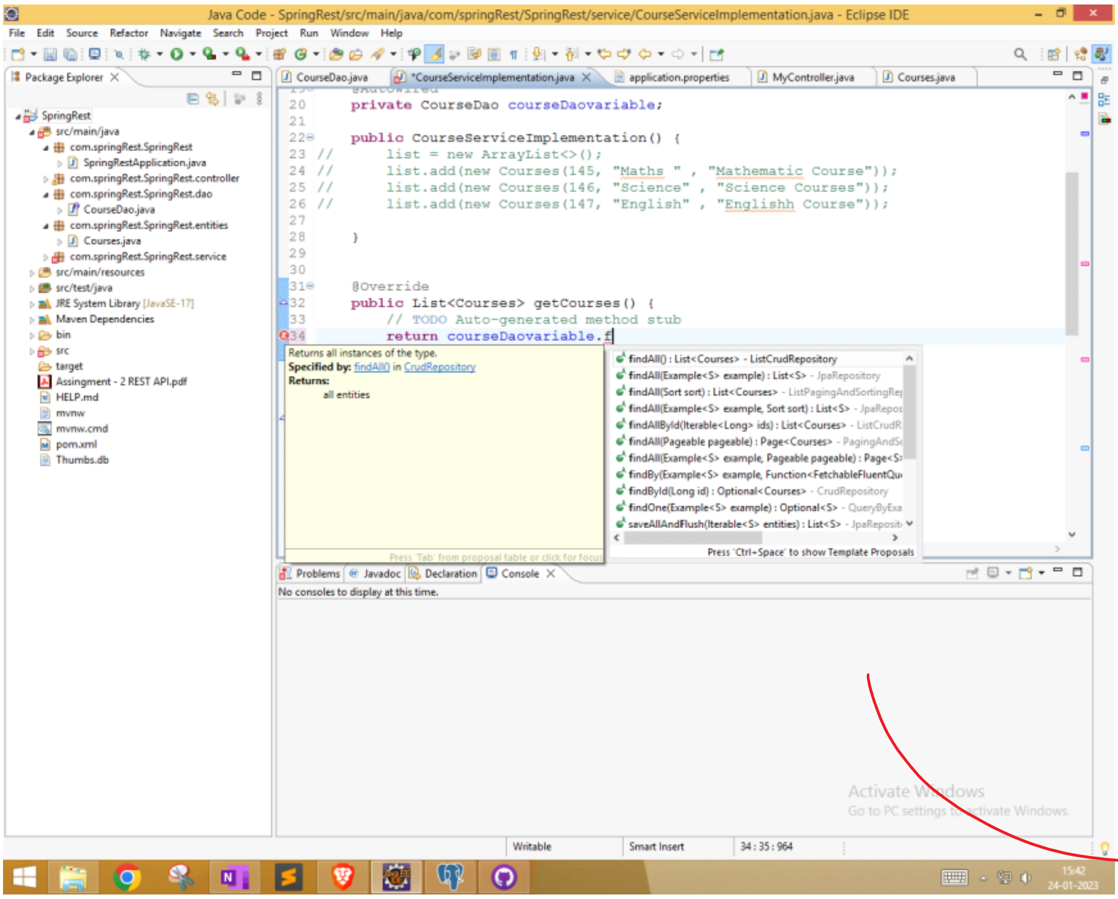
}

Now we will go to the course implementation file and in that we will comment out all the earlier Methods --. because they were dealing with the data manipulation at RAM Level.



```java
@Autowired
private CourseDao courseDaovariable;

public CourseServiceImplementation() {
/       list = new ArrayList<>();
/       list.add(new Courses(145, "Maths " , "Mathematic Course"));
/       list.add(new Courses(146, "Science" , "Science Courses"));
/       list.add(new Courses(147, "English" , "Englishh Course"));

}
```

Since coursedao is an interface and we can't form object of that but we need an object for same, So we will use autowired,



Java Code - SpringRest/src/main/java/com/springRest/SpringRest/service/CourseServiceImplementation.java - Eclipse IDE

```
    @Autowired
    private CourseDao courseDaovariable;

    public CourseServiceImplementation() {
//      list = new ArrayList<>();
//      list.add(new Courses(145, "Maths " , "Mathematic Course"));
//      list.add(new Courses(146, "Science" , "Science Courses"));
//      list.add(new Courses(147, "English" , "Englishh Course"));

    }


    @Override
    public List<Courses> getCourses() {
        // TODO Auto-generated method stub
        return courseDaovariable.f
```

Returns all instances of the type.
Specified by: findAll() in CrudRepository
Returns:
all entities

findAll() : List<Courses> - ListCrudRepository
findAll(Example<S> example) : List<S> - JpaRepository
findAll(Sort sort) : List<Courses> - ListPagingAndSortingRe
findAll(Example<S> example, Sort sort) : List<S> - JpaRepos
findAllById(Iterable<Long> ids) : List<Courses> - ListCrudR
findAll(Pageable pageable) : Page<Courses> - PagingAndS
findAll(Example<S> example, Pageable pageable) : Page<S
findBy(Example<S> example, Function<FetchableFluentQu
findById(Long id) : Optional<Courses> - CrudRepository
findOne(Example<S> example) : Optional<S> - QueryByExa
saveAllAndFlush(Iterable<S> entities) : List<S> - JpaReposit
Press 'Ctrl+Space' to show Template Proposals
Press 'Tab' from proposal table or click for focus
Package Explorer
SpringRest
src/main/java
com.springRest.SpringRest
SpringRestApplication.java
com.springRest.SpringRest.controller
com.springRest.SpringRest.dao
CourseDao.java
com.springRest.SpringRest.entities
Courses.java
com.springRest.SpringRest.service
src/main/resources
src/test/java
JRE System Library [JavaSE-17]
Maven Dependencies
bin
src
target
Assingment - 2 REST API.pdf
HELP.md
mvnw
mvnw.cmd
pom.xml
Thumbs.db
CourseDao.java
*CourseServiceImplementation.java
application.properties
MyController.java
Courses.java
Problems
Javadoc
Declaration
Console
No consoles to display at this time.
Writable
Smart Insert
34 : 35 : 964
15:42
24-01-2023

Now to complete our functions we have inbuilt fxns. Just type and you will get suggestions.

For ex: for getting courses .findAll()

Similarly to get single course : return courseDaovariable.getReferenceById(courseId);
In update and add we use : courseDaovariable.save(course);

Why ? : Because if its not available JPA will add and if its available JPA will update it