Eng. Ayah Alrifai

PART 1

- Spring Inversion of Control - XML Configuration
- Spring Dependancy Injection - XML Configuration
- Spring bean scope and Life cycle - XML Configuration
- Spring Configuration with java annotation - Inversion of Control
- Spring Configuration with java annotation - Dependancy Injection
- Spring Configuration with java annotation - Bean scope and Life cycle
- Spring Configuration with java code - No XML

why spring

to solve coupling Problem

بحيث أنا مش محتاج أبني object لا أكتب الكود ، بينيص لا Run

* spring use MVC Design Pattern

Coupling Problem

Class ReadExcel implements Read
class ReadJSON implements Read
class ReadXML implements Read

Class Reading → has instance Variable where Type equal Read

→ هلا هاد ال Variable ممكن يكون أي نوع من الثلاث أنواع يلي فوق
بالتالي لو كان نوعها ReadJSON
وبالمستقبل قررت أغيره لـ ReadExcel
أنا محتاج أرجع للكود وأعدله

2. How spring solve coupling

- المفروض ال programmer ما يعمل new object داخل الكود ، ويعرف ال object ب Xml file عن طريق إستخدام bean tag

3. Bean attributes

<bean name="read" class="Reading"> </bean>

name → أي إسم لل bean

class → لازم ال Path كامل
ex: com.exampleTest

4. instance Variable inside class "Injection"

1. Prop injection
   by create setter & getter

2. Constructor Injection
   Pass Value by constructor

create Bean for any type of Read

<bean name="json" class="ReadJSON">
</bean>

① <bean name="read" class="Reading">
<Property name="read"
ref="json">
</Property>
</bean>

name → لازم يكون نفس إسم ال instance Variable يلي داخل ال class

② <bean nam="read" class="Reading">
<constructor-arg ref="json" />
</bean

Note: if Primitive dataType we will use Value
Value = "5"
Same note for ①

class has more than instance variable

① Private Read read;
Private Count int;
+ we have setter & getter

```
<bean name="read" class="Reading">
    <Property name="read" ref="json"></Property>
    <Property name="count" value="5"></Property>
</bean>
```

we can use 2 types in same bean

② Private Read read;
Private int count;
+ we have constructor (Read read, int x)
order!!

```
<bean name="read" class="Reading">
    <constructor-arg ref="json" />
    <constructor-arg value="5" />
</bean>
```

6. How to Inject List

Private List names + setter & getter

```
<Property name="names">
    <list>
        <value>java</value>
        <values>js</values>
    </list>
</Property>
```

7. How to inject set

Private Set names + setter & getter

```
<Property name="names">
    <set>
        <Value>java</Value>
        <values>js</Value>
    </set>
</Property>
```

How to inject Map

Private Map names; + setter & getter

```
<Property name="names">
    <map>
        <entry Key="1" Value="java"></entry>
        <entry Key="2" Value="js"></entry>
    </map>
</Property>
```

9. Method Hooks

```
<bean name="read" class="Reading" init-method="init" destroy-method="clean">
```

init-method="init" ← بعرف Reading class داخل method بحيث بتم إستدعائها أول ما تبدأ الـ bean بإسم الـ method

destroy-method="clean" ← بعرف Reading class داخل method بحيث بتم إستدعائها أول ما ينتهي شغل الـ bean "يعني أول ما ترجع ك garbage collection

b. Method Hooks by using spring

```
Public class Reading implement InitializingBean, DisposableBean {
    :
    @override
    Public void afterPropertiesSet() throws Exception {
    }

    @oVerride
    Public void distroy() throws Exception {
    }
}
```

no need to add init-method, destroy-method ~~tag~~ attribute in bean tag

Container

لا أعمل ال Run لل Project بروح بعمل creat ل obj ل bean ا في مكان خاص داخل ال spring
بالتالي أنا محتاجه أصل لهاد ال obj ، مثلاً بعمل لإله عن طريق إني أصل لل Container
ومنه بعمل لل bean

"Application Context"

```
ApplicationContext appcontext = new FileSystemXmlApplicationContext(" ");
```

← بعطيه ال Path كامل لل Xml

```
appContext = new ClassPathXmlApplicationContext(" ");
```

← بس بعطيه إسم ال Xml file

```
Reading read = (Reading) appContext.getBean("read");
```

← نفس الإسم يلي داخل ال Xml

12. JDBC Template

① create Bean for DB Connection

```
<bean id="dataSource" class="org.springframework.jdbc.datasource.Driver..">
```

← class موجود داخل spring
من خلالها بعمل ال Con DB

```
    <Property name="driverClassName" Value="com.mysql.jdbc.Driver"></Property>
    <Property name="url" Value="jdbc:mysql://[host]/[DB name]"></Property>
    <Property name="username" Value="[userName]"></Property>
    <Property name="Password" Value="[Password]"></Property>
</bean>
```

② create Bean for Read, Insert, delete from DB

```
<bean id="jdbcTemp" class="org.springframwork.jdbc.core.jdbcTemplate">
    <property name="dataSource" ref="dataSource"></Property>
</beans>
```

How to use jdbcTemplate

in class create instance variable [JdbcTemplate jdbcTemplate]

* jdbcTemplate.update(sql);
  sql statment as string ↵
  For Insert, update, delete

14. use jdbcTemplate for select

select stat

ال model يلي متوقعته ترجع

```
jdbcTemplate.query(sql, new RowMapper<User>() {

    @override
    Public User mapRow(Resultset set, int i) throws SQLException {

        User u = new User();
        u.setID(rs.getInt("ID"));
        u.setName(rs.getString("NAME"));

        return u;
    }
});
```

Row عبارة عن data يلي فيه من خلال إسم ال DB يلي column

الإسم يلي بال DB أو بال alias

Note: read about queryForObject

15. Prepared statement

```
jdbcTemplate.execute("insert into user values(?,?)", new PreparedStatementCallBack
        <Integer>() {
```

index 1 ↵
index 2 ↵

dataType ↵ ال value يلي بدي أرجعها

```
    @override
    Public Integer doInPreparedStatement(PreparedSt st) throws Exception {

        st.setString(1, "Ayah");
        st.setString(2, "1997Ayah");
        return st.executeUpdate(sql);
    }
});
```

nfiguring Spring Container

1. XML Configuration file (legacy)
2. Java annotation (modern)
3. Java source code (modern)

XML Configuration file

```
<bean  id= "alias"  class = " full Pkg + class Name >

</bean>
```

Create a spring Container (IOC)

spring container is generically known as Application Context

```
ClassPathXmlApplicationContext context = new
        ClassPathXmlApplicationContext("appContext.xml");
context.close();
```

Retrieve Beans from Container (IOC)

```
Coach coach = context.getBean("myCoach", Coach.class);
                  id in ↵
                  xml file
```

Dependancy Injection (DI)

① Contsructor Injection

```
<bean id="id" class="class" >
        <constructor-arg ref="id1"  />
</bean>
```

obj ليه ↵
لو primitive بكتب value

② setter Injection

```
<bean id="id" class="class" >
        <property name="same name in class" ref="id1" />
                                  or value ↵
                                  if primitive
</bean>
```

Inject Values by Prop file

① Create Prop file

sport.properties

foo.email = ayah97@gmail.com

foo.team = development

② import prop file in xml file

<context:property-placeholder Location = "classpath:sport.properties" />

③ Reference Value from Prop file

<property name="email" value = "${foo.email}" />

<property name="team" value = "${foo.team}" />

bean is a Singleton by default

Spring container creates only one instance of the bean, by default and it is cached in memory,

all request for the bean will return a shared referance to the same bean

bean Scope

<bean id="id" class="class" scope="singleton"> </bean>

Singleton, Prototype, request, session, global-session

↓

new instance for each request

bean life cycle

* init-method = "methodName"
* destroy-method = "methodName"

→ أو من خلال java بعمل class implement من spring

Spring Configuration with Annotations?

1. XML Configuration can be Verbose
2. Configure your spring beans with annotations.
3. annotation minimizes the XML configuration.

Spring will scan your java classes for special annotation and automatically register the beans in the spring container

Enable component scanning in spring config file

```
<beans>
    <context:component-scan base-package="Pkg name" />
</beans>
```

Add @Component Annotation (IOC)

```
@Component("theCoach")
Public class TennisCoach implement Coach {
    ...
}
```

Bean Name

default bean id

Class Name
StudentData → default bean id studentData

first letter ↩ small (lower-case)

just add @Component

Inject Dependancy using Annotation

1 Constructor Injection

Constructor ⇒

```
@Autowired
Public TennisCoach( FortuneService service) {
    this.service = service;
}
```

Component عبارة عن

2 setter Injection

ممكن يكون إسم مختلف
حسب شرط setter

```
@Autowired
Public void setService( FortuneService service) {
    this.service = service;
}
```

3 Field Injection

```
@Autowired
Private FortuneService service;
```

Qualifier

```
@Autowired
@Qualifier("happyFortuneService")
Private FortuneService service;
```

إذا كان عندي instance من نوع superclass variable

وعندي أكثر من class عامل extend لنفس

الـ super class، وكلهن beans، كيف ممكن

يحدد البرنامج أي bean بدي أستخدم؟!

بيطلع NoUniqueBeanDefiExcep Exception

inject Prop file using Annotation

```
@Value("${foo.email}")
Private String email;

@Value("${foo.team}")
Private String team;
```

Scope with annotation

* default scope is singleton

```
@Component
@scope("singleton")
Public class TennisCoach implements coach {

}
```

life cycle Method with Annotation

```
@PostConstructor
Public Void init() {

}
```

```
@PreDestroy
Public Void destroy() {

}
```

3 ways to Configuring spring Container

1. full xml config ✓
2. XML Component scan ✓
3. java configuration class (No XML)

↓

```
@Configuration
@ComponentScan("Pkg name") ⇝ optional
Public class ConfigClass {

}
```

Read Spring java config class

```
AnnotationConfigApplicationContex context = new
        AnnotationConfigApplicationContext(ConfigClass.class);
```

Defining spring beans with java code

```
@Configuration
Public class sportConfig {

    @Bean                                  → bean id
    Public Coach swimCoach() {

        return new swimCoach();

    }
}
```

Inject bean dependancy

```
@Bean
Public FortuneService happyFortuneService() {
    return new HappyFortuneService();
}

@Bean
Public Coach swimCoach() {
    return new SwimCoach(happyFortuneService());
}
```

Retrieve bean from spring Container

```
context
Coach myCoach = context.getBean("swimCoach", Coach.class);
```

Inject Value from Prop file

1. load Prop file in java config

```
@Configuration
@PropertySource("classpath: sport.properties")
```

2. Reference values from prop file

in any class

```
@Value("${foo.email}")
```

Spring Configuration with XML

1. make file [applicationContext.xml] and add All beans (IoC)
2. in java ClassPathXmlApplicationContext → context (ID)

   context.getBean(BeanId)

Spring Configuration with Annotation

1. make file [applicationContext.xml] and just add (IoC)
   component-scan element

   for each bean add these Annotations @Component, @Autowired
   @Qualifier(beanName), ...

2. in java ClassPathXmlApplicationContext → context (ID)

   context.getBean(BeanId);

Spring Configuration with java Code (no xml)

1. make file [applicationContext.xml] and just add (IoC)
   component-scan element

   before Configuration class add these Annotations
   @Configuration, @PropertySource(path), ..

   Add method to get (create) bean and before method add
   this Annotation @Bean

2. in java AnnotationConfigApplicationContext → context (ID)

   context.getBean(BeanId);

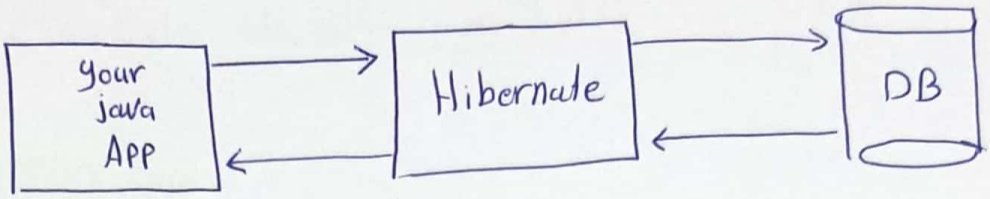
PART 3

Eng. Ayah Alrifai

* Hibernate Config with Annotation
* Hibernate CRUD Features
* Hibernate Advanced Mapping (@OneToOne, @OneToMany)
* Eager & lazy fetch Type
* Hibernate Advanced Mapping (@ManyToMany)

what is Hibernate

A framework to persisting / saving java objects in a database

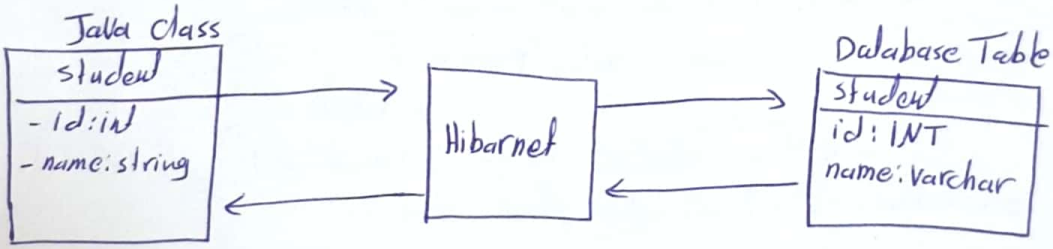


Benefits of Hibernate

1. handels all the low-level SQL.
2. Minimizes the amount of JDBC code you have to develop.
3. Provides the object-to-Relational Mapping (ORM)

Object-to-Relational Mapping (ORM)

The developer defines mapping between java class and database table



Retrive / save data from database

① int id = (Integer) session.save(studentEntity);

Hibernat نوع من object

② Student student = session.get(Student.class, id);

model PK

Querying for java objects

```
Query query = session.createQuery("from student ");
List <Student> students = query.list();
```

↳ hql

Hibernat CRUD Apps

1. Create object
2. Read object
3. Update Object
4. Delete object.

Setting up Hibernate in Eclipse

in [src] create file [hibernate.cfg.xml]

```
<hibernate-configuration>
    <session-factory>
        <Property nam="connection.driver_class">
            Com.mysql.jdbc.Driver
        </Property>
        <Property name="connection.url">
            jdbc:mysql://localhost:3306/db-name
        </Property>
        <Property name="connection.username">
            root
        </Property>
        <Property name="connection.password">
            123456
        </Property>
        ⋮
    </session-factory>
</hibernate-configuration>
```

Annotate Java class

```
@Entity
@Table(name="student")          ~> name in DB
Public class Student {

    @Id  ~> PK
    @Column(name="id")          ~> name in DB
    Private int id;

    @Column(name="first_name")  ~> name in DB
    Private String firstName;
    ⋮
}
```

إذا كان الإسم في الـ DB مختلف وحدد الـ name
نفس الإسم في الـ Java

Develop java code to perform DB operations

```
SessionFactory sessionFactory = new Configuration()
                                .configure("hibernate.cfg.xml")
                                .addAnnotatedClass(Student.class)
                                .buildSessionFactory();

Session session = sessionFactory.getCurrentSession();

Student s = new Student("Ayah", "Alrefai", "a@a.com");

session.beginTransaction();

session.save(s);

session.commit(); getTransaction();
```

Hibernate Identity - Primary Key

```
@Id
@GeneratedValue(strategy=GenerationTypeIDENTITY)
@Column(name="id")
```

auto-increment

Primary Key - changing the starting Value

MYSQL : Alter table

Reading object with Hibernate

```
Student student = session.get(Student.class, 5);
```

[PK @Id]

Querying object with Hibernate

```
List<Student> students = session.createQuery("from Student")
                                .getResultList();
```

java obj

Examples

```
from Student s where s.lastName = 'Done'
```

field Name in java

```
from Student s where s.firstName = 'ayah'
    OR s.lastName = 'alrifai'
```

update object using Hibernate

```
int student_id = 1;
Student student = session.get(Student.class, student_id);
student.setFirstName("Ayah");
session.getTransaction().commit();
```

update data in DB

Query object - update

```
Session.createQuery("update Student set email = " +
    "'foo@gmail.com').executeUpdate();
```

Deleting object with Hibernate

```
Student s = session.get(Student.class, 1);
Session.delete(s);
```

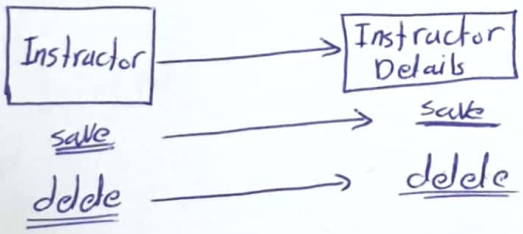
Query object - Delete

```
Session.createQuery("delete from Student where id = 1")
    .executeUpdate();
```

Database Concepts

1. Cascade

Apply the same operation to related entities



إذا كانت العلاقة مثلاً many to many

ما بينفع نستخدم cascade delete لأنه رح يحذف كل إشي

2. Fetch Type

Eager: will retrieve every thing

Lazy: will retrieve on request

3. uni-Directional

Instructor → Instructor Details

يعني لما أجيب الـ Instructor بجيب معه Instructor Details بس العكس لأ

4. Bi-Directional

Instructor ⟷ Instructor Details

ne-to-one @OneToOne

```
@Entity
@Table(name="InstructorDetails")
Public class InstructorDetails {

        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        @Column(name="id")
        Private int id;

        @Column(name="youtube_channel")
        String youtubeChannel;

        @Column(name="hobby")
        Private String hobby;

        ⋮
}
```

```
@Entity
@Table(name="Instructor")
Public class Instructor {
      @Id
      @Column(name="id")
      Private int id;

      ⋮

      @OneToOne
      @JoinColumn(name="instructor_detail_id")
      Private InstructorDetails instructorDetails;
```

Name of column in data base

Cascade

```
@OneToOne(cascade = CascadeType.ALL)
```

Bydefault no cascade Type

- PERSIST → if persisted / save
- REMOVE → if removed / delete
- REFRESH → if refreshed
- DETACH
- MERGE

ultiple CascadeType

```
@OneToOne( cascade={ CascadeType.REFRESH,
                     CascadeType.REMOVE,
                     CascadeType.MERGE})
```

Build main app

```
SessionFactory factory = new Configuration()
                          .configure()
                          .addAnnotationClass(Instructor.class)
                          .addAnnotation Class(InstructorDetails.class)
                          .buildSessionFactory();

Session session = factory.getCurrentSession();
:
Session.save(tempInstructor);
```

↳ will save InstructorDetails (cascade)

@OneToOne Delete Entity

```
Instructor instructor = session.get(Instructor.class, 1);
```

→ هل بترجع الـ InstructorDetails ؟!

```
session.delete(Instructor);
```

↳ will delete InstructorDetails
لأنه عملنا الـ CascadeType.ALL

ممكن أستخدم الـ createQuery مباشرة

@OneToOne Bi-Directional

بالطريقة السابقة كنت أعمل الـ InstructorDetails مباشرة من الـ Instructor بس العكس لأ

* اهعل أعدل على InstructorDetails , أضيف field جديد من نوع Instructor و بعلق @OneToOne

```
@OneToOne(mappedBy = "instructorDetails")
Private Instructor instructor;
+ getter / setter
```

إسم الـ field يلي داخل Instructor Details

, Cascade = CascadeType.ALL

OneToOne Bi-dirictional Delete

session.delete(InstructorDetails);
↳ will delete Instructor because cascadeType is ALL

@OneToOne Bi-Dirictional Delete only InstructorDetails

updata InstructorDetails class → Instructor field
set cascadeType all types exept [ALL, REMOVE]

instructorDetails.getInstructor().setInstructorDetails(null);
session.delete(instructorDetails);

لازم أكسر العلاقة قبل ما أعمل delete

@OneToMany & @ManyToOne

Instructor —1:m→ Course

كل إشي بينطبق ع oneToone بينطبق عليهن بس الفرق إنه لما تكون العلاقة oneToMany بكون عندي List

* Course has FK (Instructor_id)

Eager & Lazy Fetch Types

Eager: دائماً بعمل load لل الداتا بالتالي ممكن يأثر ع Performancy

Lazy: بعمل load لل data لما أنا أحتاجها

@OneToMany (fetch = FetchType.LAZY, mappedBy = "instructor")

Default Fetch Type

@OneToOne → EAGER
@OneToMany → LAZY
@ManyToOne → EAGER
@ManyToMany → LAZY

...ception when Fetch Type lazy

* if session open and fetch type is lazy
  when load Instructor will not load courses
  but when use instructor.getCourses()
  will load Courses.

* there are a problem if close session before
  get Courses

* to solve problem always use getter before close session

* or can use HQL

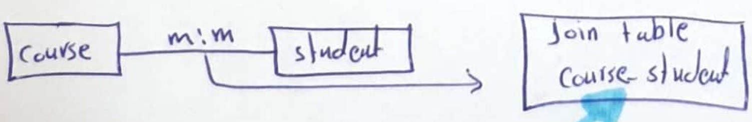
```
Query <Instructor> query = session.createQuery("select i from Instructor i
        Join FETCH i.courses where i.id = :theId", Instructor.class);
query.setParameter("theId", 1);

Instructor instructor=query.getSingleResult();
```

Many to Many

* بال DB لا تكون عندي علاقة m:m لازم يكون فيها Join Table بحيث يكون فيها Ref لكل Table، أما بال java

ما في داعي أعملها



[I] Course class

```
@ManyToMany
@JoinTable( name = "Course_student"
  joinColumns =@joinColumn ("Course_id")
  , inverseJoinColumn = @joinColumn("student_id")
)
Private List <student> students;
```

نفس الشي بال student
بس بعكس joinColumn
و inverseJoinColumn

Spring Boot

Eng. Ayah Alrifai

- Start Project
- Controller
- Service
- Connect to DB
- Model
- Repository
- Handel Exceptions
- Test
- Security

Start Project

1. create maven project
2. add this dependancy (Pom.Xml)

```
<Parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.0.4.RELEASE</version>
</Parent>
```

+dependancies → copy and past

main class in springboot

add @SpringBootApplication

+

in main method add

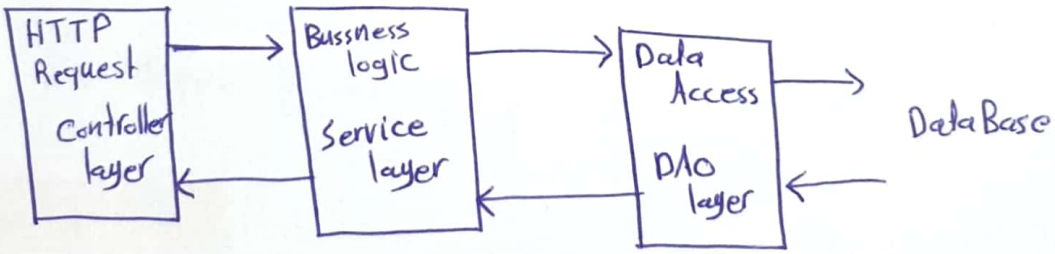
```
ClassName.run(ClassName.class, args);
```

in resources make file

application.properties

```
server.port=8080
```

N Tier Architecture



Controller

add these annotations

@RestController

@RequestMapping("/"); ← string اي

تكون فيه معلومات عن الرابط
يلي المفروض أرد عليها أو يلي
المفروض انفذها أول ما أفتح ال link

@Controller Vs @RestController

إذا كنت بتشتغل على web site ومحتاج أرجع view بتكون مبنية بال jsp مثلاً

في حال بدي أشتغل بس web service بدون ما ابعت view

@RequesParam String (id)

http://localhost:8080/spring?(id=5)

Public String fun(@RequestParam String(

@PathVariable

@GetMapping("/spring/{id}")
Public String fun(@PathVarieble String(id))

localhost:8080/spring/5

@Requestbody

Public String fun(@Requestbody User user)

بيبعت بال body يلي هو JSON بس لازم أكون عامله mapping بشكل صحيح

mapping for many Paths

@GetMapping({ "", "/"})

auto build in maven

add this dependancy

```
<dependancy>
    <groupId> org.springFramework.boot </groupId>
    <artifactId> spring-boot-devtools </artifcatId>
    <scope> runtime </scope>
</dependancy>
```

Service layer

add annotation @service

أيه ال service بال spring هي عبارة عن Singelton → Private constructor
Public static method return instance

Autowired

لهتي ما أعلى كل الحكي يلي فوق , بال Controller
ابني (أستدعي) ال method يلي بترجع instance variable
من service بعد هيك بال Controller → @AutoWird
Private NewService service
← بستخدمها مباشرة

Connection to database

1. add dependancy to pom.xml
2. add database config to application.proprties
   spring.datasource.url =
   spring.dataSource.username=
   spring.dataSource.Password=
3. add new ~~class~~ interface Repository with annotation @Repository
4. interface extend Curd Repository → MySQL لانه بنستخدم
   لو بنستخدم مثلاً Mango بدي extend interface مختلف
   JPaRepository
   MongoRepository
5. extend CurdRepository <Model, string>

Model [MySQL]

@Entity → بعرف هاي ال class
على إنها model

@Table(name = "T_table") → بحدد إسم ال Table
على الداتابيس انا نفسي
إسم ال class ما في داعي

@JSONIgnorProperties(ignorUnkawn="true")
← إذا رجلي بال JSON ← prop
ومش معرفيتها عندي بال model
بعملها ignor وبيكمل mapping

Repository

should use it inside service and it is Autowired

How to return Response and Entity

Public List<Model> fun() ⟺ Public Response<List<Model>> fun()

return new Response <Model> (result, HttpStatus.created); ← Enum

Handel Exceptions

1. Creat new class " ApiExceptionHandler "
2. add annotation @ControllerAdvice → بتكون shard بين كل ال Controllers
3. extend Response EntityHandler → في داخلها كثير Exep ويمكن اضيف انا أشياء جديدة
4. create new class ErrorDetails → هو ال Json يلي رح يرجع لل user
5. create class extend Exception (Runtime) for all possible Exception
6. inside ApiExceptionHandler add new method

```
@ExceptionHandler(NotFoundExcP.class)
Public ResponseEntity <ErrorDetails> handelApiException(ex, request) {
```

(ex → one of the exception that created on point 5 and other Param WebRequest)

```
    ErrorDetails e = new ErrorDetails(ex.getMessage, request.getDescription(fa
    return new ResponseEntity <>(e, ex.getStatusCode);
}
```

Handel Exceptions

في طريقتين لحتى أعمل handel الـExceptions إما أن يكون custom Exp أو يكون defined Exep

هلا بالبداية بنبني class بتعمل extend لـExceptions وبعمل custom EXP بعمل Extent إلها

بنبني class بتعمل extend ResponseEntityExceptionHandl وبحط فيها annotation @ControllerAdvice

هاي الـclass فيها handel لكثير Exceptions، جواتها بعمل method تعمل handel لكل الـ custom Exep لأنه كلهن حالات extend لـclass 1 [Polymorphsim]

بس الـExp الثانية يلي الـclass حاططلهن Emp بقدر اعمل أعملهن override وأكتب الـemp بالطريقة يلي بدي اياها

Dependancy for test

artifact Id → spring-boot-starter-test
scope → test

→ بنعتمد على
* Junit
* AssertJ
* mockito

How to test

* create new class [NameTest]
* all methods in test class are [Void]

```
@Test
Public void sumTest() {
        Calc cal = new Calc();    // as instance variable
        Assertions.assertThat(cal.sum(1,2)).isEqualTo(4);
}
```

ممكن اعملها as instance variable ودخلها annotation ← @Befor

Start test RestFul API

* Create new class for test with name PollServiceTest
* add annotation @Runwith(SpringRunner.class)
* @Test
  Public void whenFindAll() {

مش فاهم إشي

```
@autowired          → spring
private PollService Pollservice;
or
@TestConfiguration
static class PollServiceContextConfigura
{
    Public (struck through)
    @Bean
    Public PollService Pollservice() {
        return new PollService();
    }
}
```

Springboot Security

* add this dependancy

  spring-boot-starter-security

* when run app will generate password should you pass it with user nam such as [user] to authentication

[Post man] → auth → Basic Auth → set password, userName

"طبعاً كل هاد الكي غير منطقي ولازم نعدله"

* Create new Pkg for Security and create new class SecurityConfig
* add annotation @EnableWebSecurity and extend webSecurityConfigurationAdapte
* override → Configure Param HttpSecurity htps~~ecurity~~
* http.cors().and().csrf().disable().sessionManagment()
  - sessionCreationPolicy(SessionCreationPolicy.STATELESS)
  - and().authorizeRequests().andMatchers("apiPath").PermitAll()

عكن يكون Pattern

نعدله ! كي api بدي تكون مفاتحة auth

.anyRequest().authenticated();

أي اشي غير هيك فوق فيه مفاتيح auth لا

عكن يستقبل array of Strings with Pattern

Run code before any thing

```
@Component
Public class FirstTimeIniti implements CommandLineRunner {

    @override
    Public Void run(String ...args) throws Exception {
        //your code
    }
}
```

Get Vs Post

* Get : data send in header (limited amount of data)
  Post : data send in body
* Get : not secured
  Post : secured
* Get : can be bookmarked
  Post : can't be bookmarked
* Get : idempotent ⟹ second request will be ignored until response of first request is delivered.
  Post : non-idempotent
* Get : more efficient
  Post : less efficient

Session Vs Cookies

Session

* sorted on server side on temporary directory file
* session end when use logout or close his web browser
* Can store unlimited of data 128 MB
* more secured, save data in a encrypted form

Cookies

* sorted on client sid as text file
* Cookies end on the life time set by the user
* can store limited of data 4KB
* not secure, data sorted as txt file