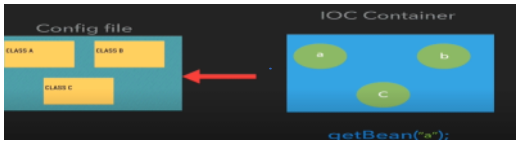
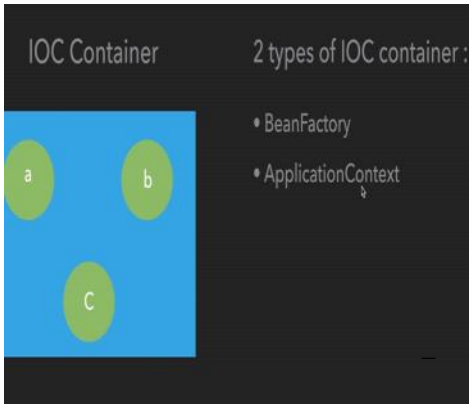
ControllerAdvice






```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">



<bean id = "airtel" class = "com.selniumexpress.ioc.Airtel"></bean>
```

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("config loaded");
        Vodaphone voda = context.getBean("vodaphone", Vodaphone.class);
        voda.calling();
        voda.data();

    }

}
```

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("config loaded");
        Airtel air = (Airtel)context.getBean("airtel");
        air.calling();
        air.data();

    }

}
```

Constructor loaded of airtel

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml")
        System.out.println("config loaded");

        Sim sim = context.getBean("sim",Sim.class);
        sim.calling();
        sim.data();

    }

}
```

Without touching vodafone and airtel

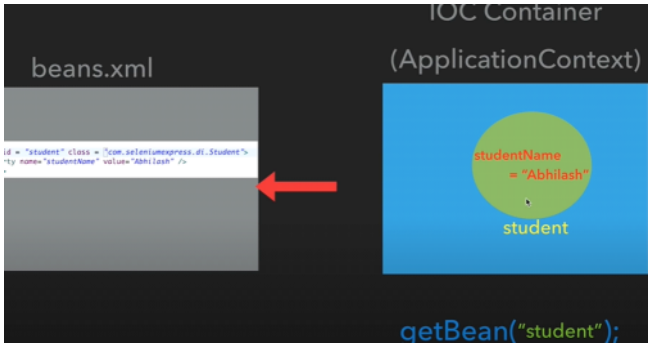
```xml
<?xml version="1.0" encoding="UTF-8"?>

<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">

<bean id = "sim" class = "com.selniumexpress.ioc.Jio"></bean>


</beans>
```

Control is taken by framework now they are creating object


```java
public class Exam {

    public static void main(String[] args) {

        Student student = new Student();
        student.setStudentName("Abhilash Panigrahi");

        student.displayStudentInfo();
    }
```

```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Abhilash Panigrahi" />
</bean>
```

Setter need to be added for student name




```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Abhilash Panigrahi" />
  <property name="id" value="1"></property>
</bean>

<bean id = "ashish" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Ashish B" />
  <property name="id" value="2"></property>
</bean>
```

```java
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
Student abhi = context.getBean("student", Student.class);
abhi.displayStudentInfo();

Student ashish = context.getBean("ashish" , Student.class);
ashish.displayStudentInfo();
```

Constructor Dependcy

```java
public class Student {

    private int id;
    private String studentName;

    public Student(int id, String studentName) {
        super();
        this.id = id;
        this.studentName = studentName;
    }
```

New Constructor added so we need t0 make new bean id
And spring will convert id to int automatically explicitly by adding type="id"

```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <constructor-arg name="studentName" value="Abhilash Panigrahi" />
  <constructor-arg name="id" value="1" />
</bean>
```

```
<bean id = "student" class="com.seleniumexpress.di.Student">
  <constructor-arg name="studentName" value="Abhilash Panigrahi" />
  <constructor-arg name="id" value="1" />
</bean>

<bean id = "dilip" class="com.seleniumexpress.di.Student">
  <constructor-arg name="id" value="1" />
</bean>
```

Object injecting

```
package com.seleniumexpress.di;

public class MathCheat {

    public void mathCheat()
    {
        System.out.println("math ceating started..");
    }

}
```

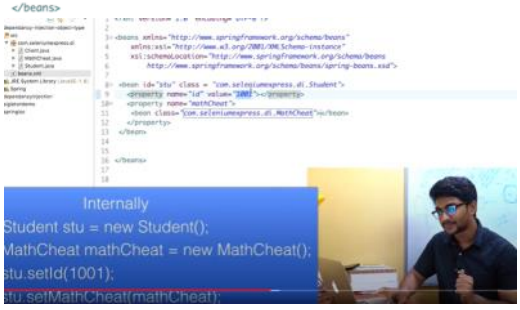
Not a good programming.

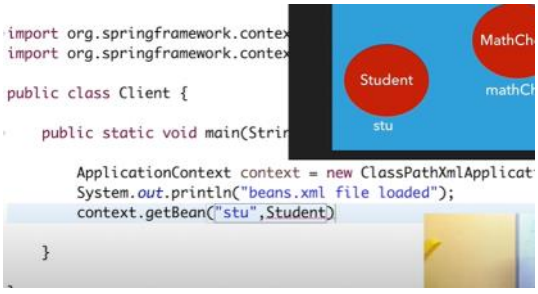
```
public class Student {

    MathCheat mathCheat = new MathCheat();

    public void cheating()
    {
        mathCheat.mathCheat();
    }
}
```

```
public class Student {

    MathCheat mathCheat;

    public void setMathCheat(MathCheat mathCheat) {
        this.mathCheat = mathCheat;
    }

    public void cheating()
    {
        mathCheat.mathCheat();
    }

}
```

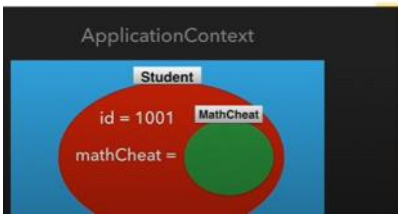
S

```
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat">
    <bean class="com.seleniumexpress.di.MathCheat"></bean>
  </property>
</bean>

</beans>
```

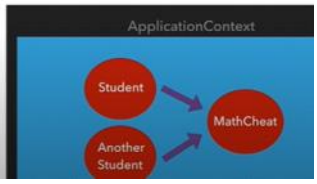




```xml
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat">
   <bean class="com.seleniumexpress.di.MathCheat"></bean>
  </property>
</bean>
```




```xml
<bean id = "mathCheatObjectValue" class="com.seleniumexpress.di.MathCheat"></bean>

<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat" ref="mathCheatObjectValue" />
</bean>

<bean id = "anotherStudent" class="com.seleniumexpress.di.AnotherStudent">
<property name="cheat" ref="mathCheatObjectValue" />
</bean>

</beans>
```



Loose coupling

```java
package com.seleniumexpress.di;

public class Student {

    private Cheat cheat;

    public void setCheat(Cheat cheat) {
        this.cheat = cheat;
    }

    public void cheating() {

        cheat.cheat();
    }

}
```

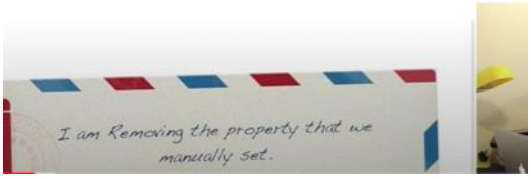
```java
public interface Cheat {

    public void cheat();

}
```

```xml
<bean id = "mathCheatObjectValue" class="com.seleniumexpress.di.MathCheat"></bean>
<bean id = "scienceCheatObjectValue" class="com.seleniumexpress.di.ScienceCheat"></bean>
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="cheat" ref="scienceCheatObjectValue" />
</bean>
</beans>
```

```java
public class Client {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("beans.xml file loaded");
        Student student = context.getBean("stu",Student.class);
        student.cheating();


    }
```

Autowired.....

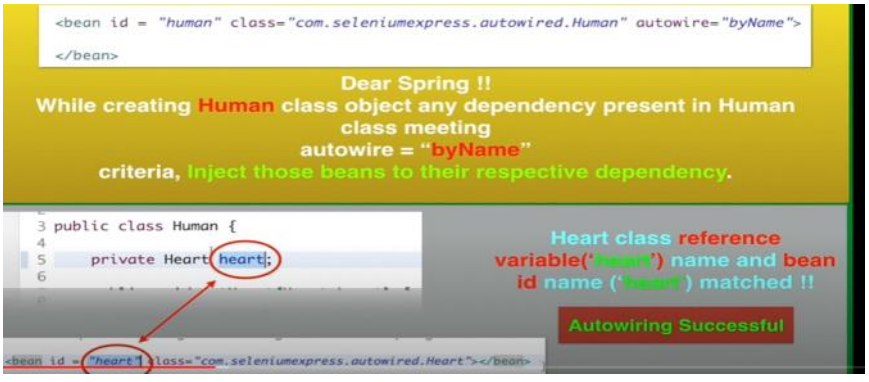

```
<bean id = "heartObject" class="com.seleniumexpress.autowired.Heart"></bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human">

<property name="heart" ref = "heartObject" />
</bean>

</beans>
```



By Name autowired

```
private Heart heart;

public void setHeart(Heart heart) {
    this.heart = heart;
}
```

```
<bean id = "heart" class="com.seleniumexpress.autowired.Heart"></bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human" autowire="byName">
```



By type autowired niche wala

Autowire constructor

```java
public class Human {

    private Heart heart;

    @Autowired
    public Human(Heart heart) {

        this.heart = heart;
    }

    public void setHeart(Heart heart) {
        this.heart = heart;
        System.out.println("setter method c
```

works same as autowire = "byConstructor"

```
<bean id = "human" class="com.seleniumexpress.autowired.Human" >

</bean>

</beans>
```

Human human = new Human();
nan() is a default constructor. Do we have it in our class?

```java
3 import org.springframework.beans.factory.annotat
4
5 public class Human {
6
7     private Heart heart;
8
9     public Human() {
0
1     }
2
3     @Autowired
4     public Human(Heart heart) {
5
6         this.heart = heart;
7     }
8
9     public void setHeart(Heart heart) {
0         this.heart = heart;
1         System.out.println("setter method called");
2     }
```

```
Sep 15, 2018 1:58:21 PM org.springframewo
INFO: Refreshing org.springframework.cont
Sep 15, 2018 1:58:22 PM org.springframewo
INFO: Loading XML bean definitions from c
you are dead
```

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

   <con

<bean id = "heartObjectValue" class="com.seleniumexpress.autowired.Heart"></bean>

 <bean id = "human" class="com.seleniumexpress.autowired.Human" >

 </bean>

/beans>
```

You need to turn on <context:annotaion-config/>
n order to use annotaions in spring. (eg: to use @Autowired)

```java
ublic class Human {

   private Heart heart;

   public Human() {

   }


   public Human(Heart heart) {

       this.heart = heart;
       System.out.println("human constr is getting called which has Heart as arg");
   }

   @Autowired
   public void setHeart(Heart heart) {
```

How @Autowired works?

1. first it try to resolves with “byType”.
2. If byType fails then it goes with “byName”

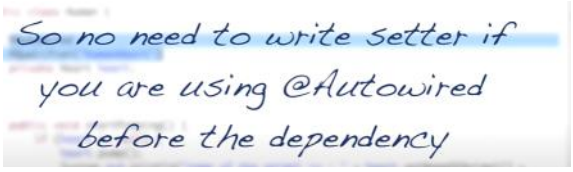
```
bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" />
bean id = "octpousHeart" class="com.seleniumexpress.autowired.Heart" />
```

Unique bean

Qualifier Two heart implmenetation

```
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <context:annotation-config />

 <bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" >
  <property name="nameOfAnimal" value="Human" />
  <property name="noOfHeart" value="1" />
 </bean>

 <bean id = "octpousHeart" class="com.seleniumexpress.autowired.Heart" >
   <property name="nameOfAnimal" value="Octpous" />
  <property name="noOfHeart" value="3" />
 </bean>

  <bean id = "human" class="com.seleniumexpress.autowired.Human" >
```

```
@Autowired
@Qualifier("humanHeart")
public void setHeart(Heart heart) {
    this.heart = heart;
    System.out.println("setter method called");
}
```



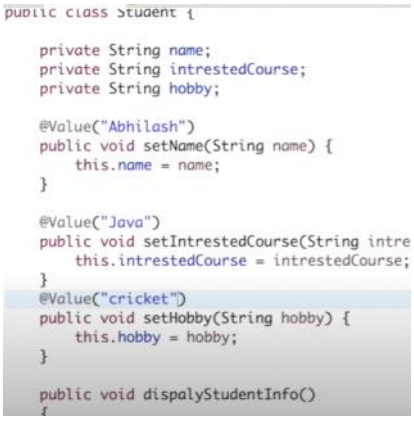
Bean property to read from property fiel
Student.name for student
Teacher.name for teacher

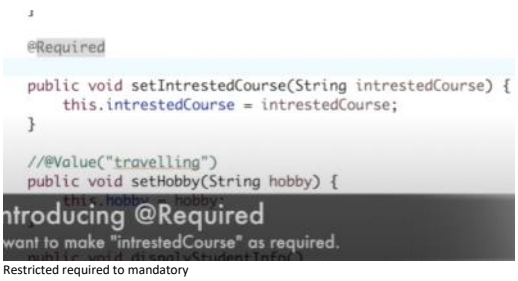
```
student.name = Abhilash
student.intrestedCourse = Java
shobby = cricket
```

```
<bean id = "student" class="com.seleniumexpress.loadingfrompropertiesfile.S
  <property name="name" value = "${student.name}"/>
  <property name="intrestedCourse" value = "${student.intrestedCourse}"/>
  <property name="hobby" value = "${student.hobby}"/>
</bean>

<context:property-placeholder/>
```

To know spring I am using from property file

@Value annotation remove properyy

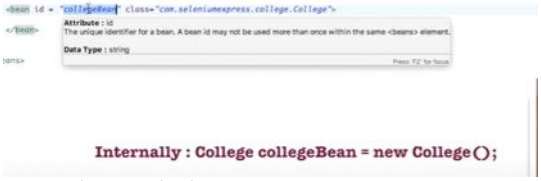




Restricted required to mandatory

We don't require setter method if we are putting @Value or autowired

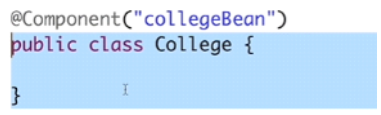
Beans file is configuration file



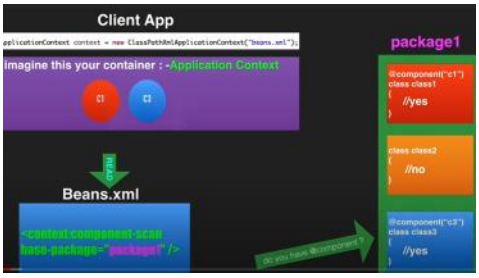
@Componet does same work as above

@Componet we neeed to write instead bean.xml to configure bean

@Componet does same thing 1 create bean
2 and register to IOC container




```
@Component("collegebean")
public class College {

    @Autowired   // for class type
    Student student;

    @Value("Murari")   // for property like string
    String name;

    void dis() {
        student.displa(name);

    }

}
```



@Configuration if we want to completely remove xml

Just below we need to write configuration scan to scan all the component

```
@Configuration
@ComponentScan(basePackages = "com.college")
public class CollegeConfig {

}
```

Now spring will read this file instead xml file

```
public class Application {
    public static void main(String[] args) {
        ApplicationContext ap= new AnnotationConfigApplicationContext(CollegeConfig.class);
        System.out.println("beans.loaded");
        University c=ap.getBean("collegebean", University.class);
        c.display();
    }

}
```

Instead @Componet we can remove @Component and use @Bean in configuration file

METHOD NAME of bean will beanId-> collegeBean

```
Configuration
ComponentScan(basePackages = "com.college")
ublic class CollegeConfig {

    @Bean
    public College collegeBean()
    {
        return new College();
    }
```

```
Configuration
ComponentScan(basePackages = "com.college")
ublic class CollegeConfig {

    @Bean
    public University university()
    {
        return new College(student());
    }

    @Bean
    public Student student()
    {
        return new Student();
    }

}
```

```
ublic class College implements University {

    Student student;

    public College(Student student) {
        super();
        this.student = student;
    }

    public College() {

    }

    @Value("Murari") // for property like string
    String name;

    public void display() {
        student.displa(name, "college");

    }

}
```

Constructor injection

Normal injection like string int using **@Value**
Class object using autowired or bean

For setter injection

```
@Configuration
public class CollegeConfig {

    @Bean
    public Principal principalBean()
    {
        return new Principal();
    }

    @Bean
    public College collegeBean() // collegeBean - bec
    {
        College college = new College();
        college.setPrincipal(principalBean());
        return college;
    }
```

```
public class College {

    private Principal principal;

    /*public College(Principal principal) {
        this.principal = principal;
    }*/

    public void setPrincipal(Principal principal) {
        this.principal = principal;
        System.out.println();
    }
```

From property injection down

```
@Component
public class College {

    @Value("${college.Name}")
    private String collegeName;

    @Autowired
    private Principal principal;

    @Autowired
    private Teacher teacher ;

    public void test()
    {
        principal.prinipalInfo();
```

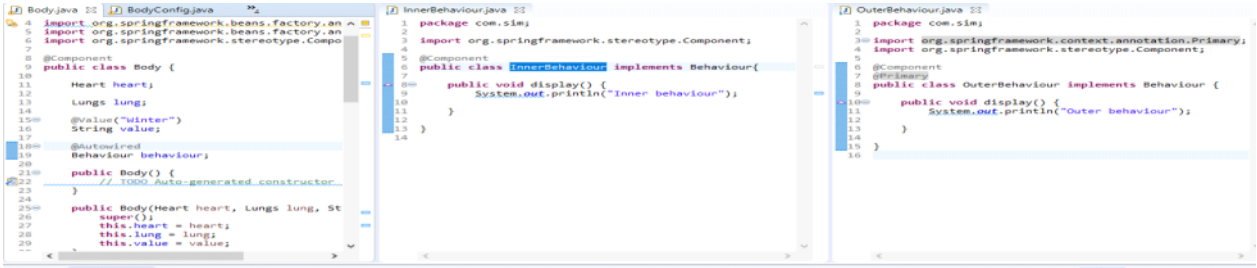
```
@Configuration
@ComponentScan(basePackages = "com.seleniumexpress.college")
@PropertySource("classpath:college-info.proerties")
public class CollegeConfig {
/*
    @Bean
    public Teacher mathTeacherBean()
    {
        return new MathTeacher();
    }
```

@Qualifier
100 of implementation if we want to use one then we use @Qualifier

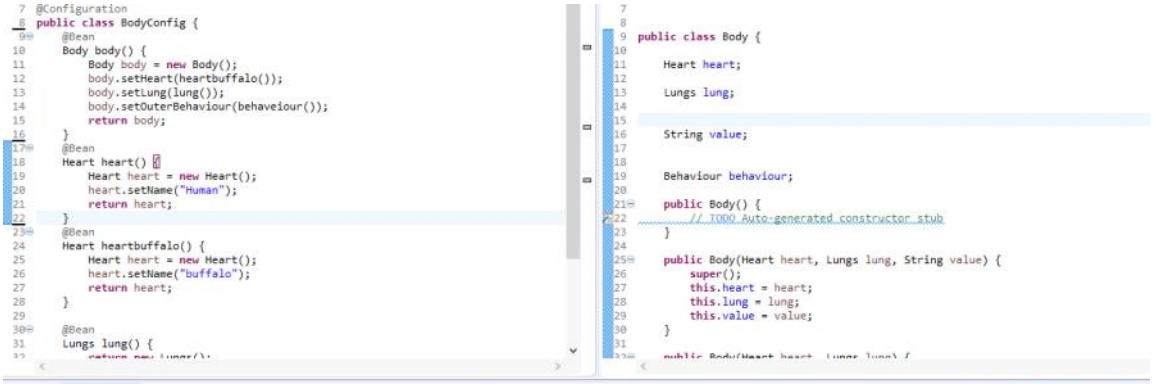
If I want an interface one implementation to be primary used

```
@Component
@Primary
public class MathTeacher implements Teacher {

    @Override
    public void teach() {

        System.out.println("Hi I am your math teacher");
        System.out.println("My name is Sourav");

    }

}
```

```
Heart is not functioning
Lung not functioning
Inner behaviour
```



```
Heart is not functioning
Lung not functioning
Outer behaviour
```



```
buffalo is functioning withnull
humanLungs is functioning withnull
Inner behaviour
```

```java
@Configuration
public class BodyConfig {
    @Bean
    Body body() {
        Body body = new Body();
        body.setHeart(heartbuffalo());
        body.setLung(lung());
        body.setOuterBehaviour(behaveiour());
        return body;
    }

    @Bean
    Behaviour behaveiour() {
        return new InnerBehaviour();
    }
    @Bean
    Heart heart() {
        Heart heart = new Heart();
        heart.setName("Human");
        return heart;
    }
    @Bean
    Heart heartbuffalo() {
        Heart heart = new Heart();
        heart.setName("buffalo");
        return heart;
    }

    @Bean
    Lungs lung() {
        return new Lungs();
```

```java
@Component
public class College {

    //@Value("${college.Name}")
    private String collegeName;
    @Required
    public void setCollegeName(String collegeName
        this.collegeName = collegeName;
    }
```

@Required if we don't want to be pass null in string only used on setter
Required bean

Most Important topic of spring……

Life cycleof bean

```java
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);

}

public void selectAllRows() throws ClassNotFoundException, SQLException {
    System.out.println("Retriving all students data..");

    // execute query
    Statement stmt = con.createStatement();

    ResultSet rs = stmt.executeQuery("SELECT * FROM ESNew.HostelStudentInfo");

    while (rs.next()) {
        int studentId = rs.getInt(1);
        String studentName = rs.getString(2);
        double hostelFees = rs.getDouble(3);
        String foodType = rs.getString(4);

        System.out.println(studentId + " " + studentName + " " + hostelFees + " " + foodType);
    }
```

For any utility method like selectallrows 0r delete or update we need to call explicitly this method **createStudentDBConnection** everywhere I have to call this method everywhere wherever db task to done

Once there is creation of bean then please execute this method automatically

---

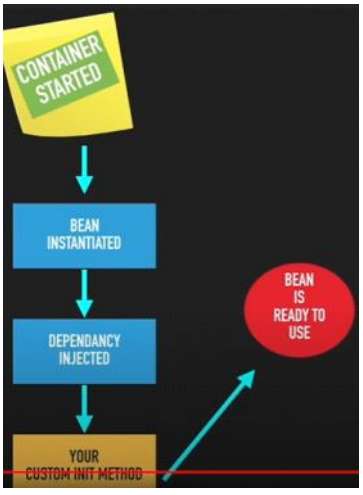
```java
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);

}
```

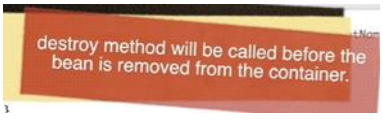
Init() ->Method

```
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);

}
```

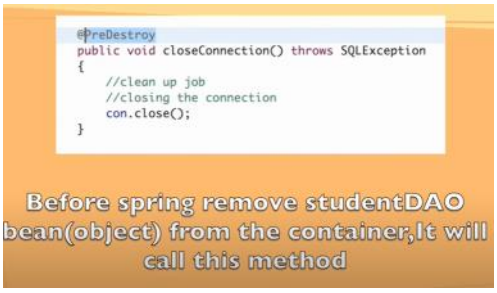
Here the createStudentDBConnection() is the init method for us. annotate a method with @PostConstruct to use it as a init method.

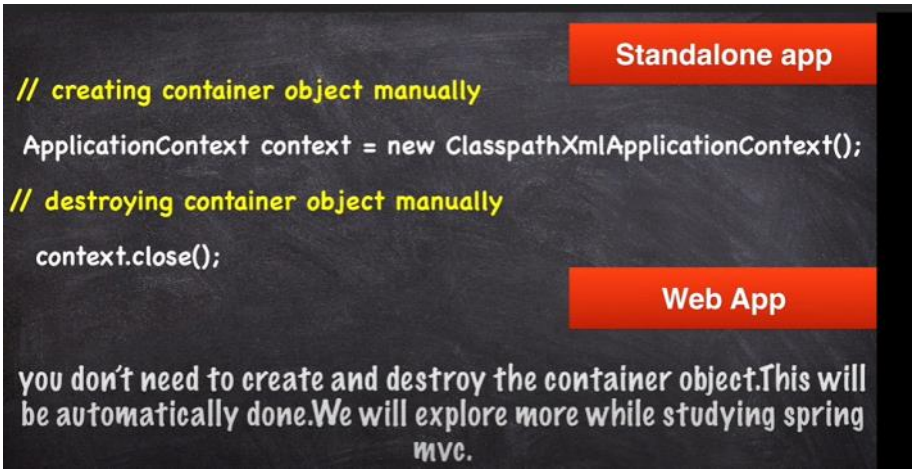
First create object in container
Then it will inject all dependency which is in form of setter

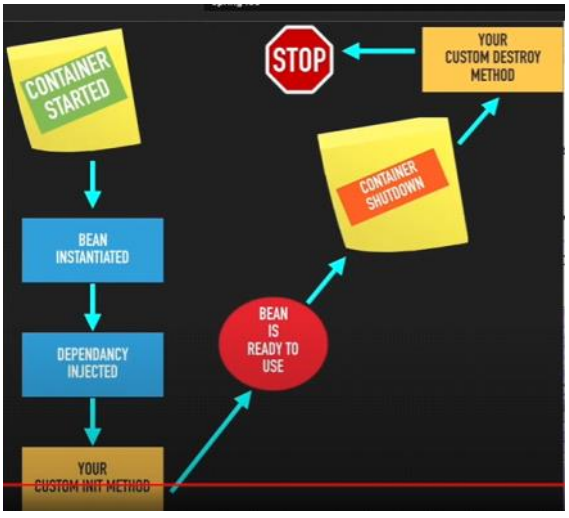


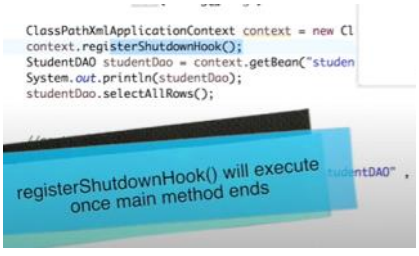


@PreDestory over the method which has to be called before the container is destroyed.



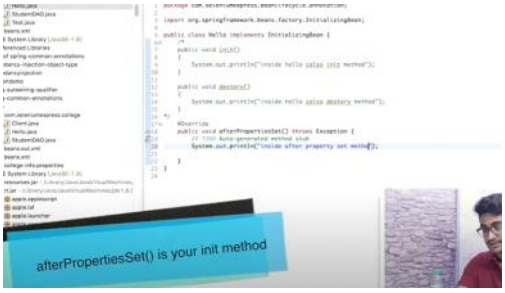






But it will not give error as context.close()

registerShutdownHook() will execute once the main thread ends. so once all your codes gets executed, It will be called and close your container. So It won't give us any exception irrespective of the line no we are calling it.

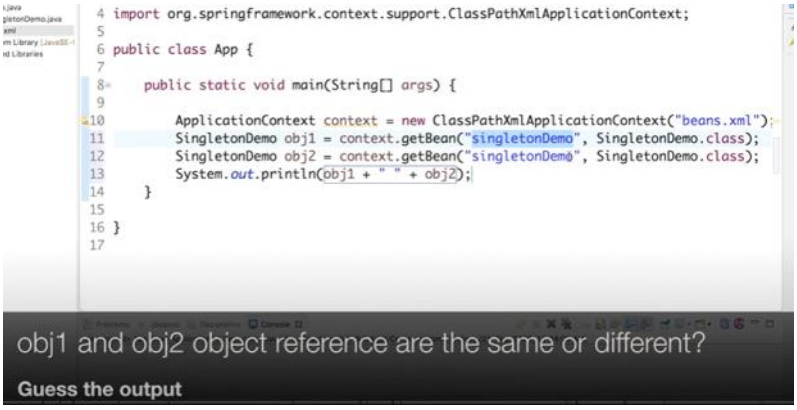


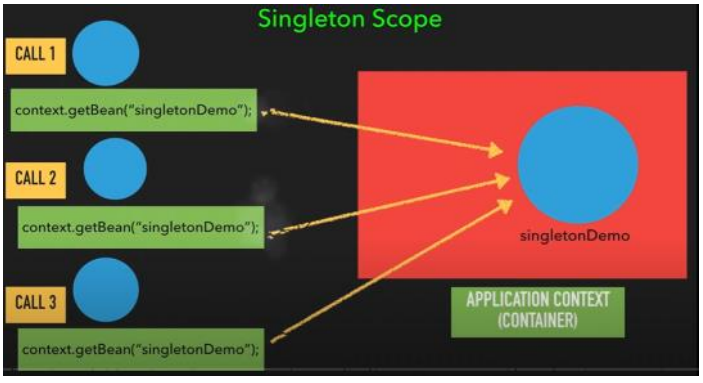

```
public class Application {
	public static void main(String[] args) {
		ApplicationContext ap = new AnnotationConfigApplicationContext(MobileConfig.class);
		System.out.println("beans.loaded");
		LandLine c = ap.getBean("landLine", LandLine.class);
		c.peep();
		Sim c = ap.getBean("sim", Sim .class);// for interface direct calling bean in config
	}

}


@Configuration
@ComponentScan(basePackages = "com.practice")
public class MobileConfig {

@Bean
Sim sim()
{
Return new vodafone();
}


@Component
public class LandLine {

	@Autowired
	@Qualifier("idea")
	Sim sim;

	void peep()
	{
		sim.calling();
	}


}


@Component
public class Idea implements Sim {
	@Value("IDEA")
	String value;
	@Autowired
	Network network;
	@Autowired
	Balance balance;

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

	public Idea(String value, Network network, Balance balance) {
		super();
		this.value = value;
		this.network = network;
		this.balance = balance;
	}

	public void calling() {
		network.display(value);

	}

	public void balance() {
		balance.display(value);

	}
}

@Component
public class Network {


	@Value("Network ")
	String name;

	 void display(String val) {
		System.out.println(val+" has a  "+name);

	}

}
```

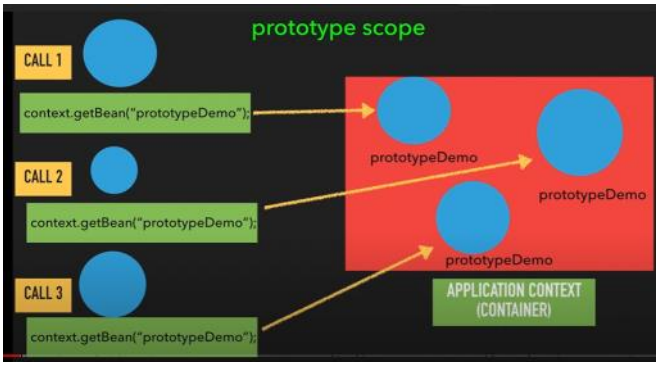
Spring scope->

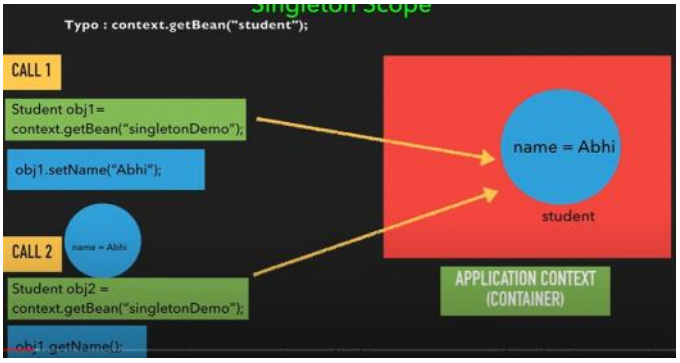
NO "GLOBAL SESSION" SCOPE

Spring removed this scope with the 5.x release.
No support for portlet.










```
@Component
@Scope(value="singleton")
public class School {
	public School() {
		System.out.println("Waoo school");
	}

}
@Component
@Scope(value="prototype")
public class Student {

	String name;

	public Student() {
		System.out.println("Waoo student");
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
```

So only singelton typebean will be loaded at run time eager loading

When protoype type bean is called then only bean is created when we asked for it

```
@Component
@Scope(value="singleton")
public class School {
    @Autowired
    Student student;
     public School() {
        System.out.println("Waoo school");
    }
```

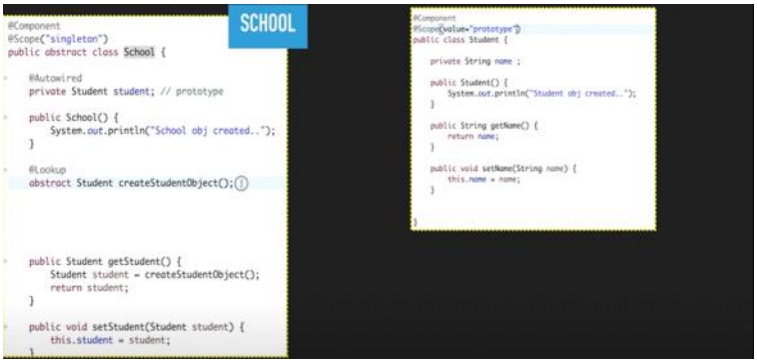
Then also it will give same object of student even if student is prototype

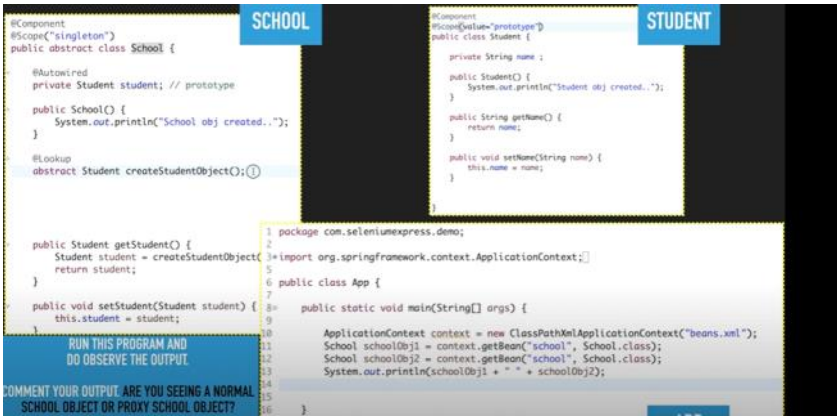
```
omponent
cope(value="prototype", proxyMode = ScopedProxyMode.TARGET_CLASS)
olic class Student {
```

This will give proxy to School class so each time we will get differne object

```
@Scope("singleton")
public class School {

    @Autowired
    private Student student; // prototype

    public School() {
        System.out.println("School obj created..");
    }

    @Lookup
    Student createStudentObject(){

        return null;
    }


    public Student getStudent() {
        Student student = createStudentObject();
        return student;
    }
```

Another method to return different object other than proxymode






```
public class App {

    public static void main(String[] args) {

        ApplicationContext appContext1 = new ClassPathXmlApplicationContext("beans.xml");
        ScopeTest scopeTest1 =appContext1.getBean("scopeTest",ScopeTest.class);
        ScopeTest scopeTest2 =appContext1.getBean("scopeTest",ScopeTest.class);

        System.out.println(scopeTest1 + " " + scopeTest2);

        ApplicationContext appcontext2 = new ClassPathXmlApplicationContext("beans2.xml");
        ScopeTest scopeTest3 = appcontext2.getBean("scopeTest",ScopeTest.class);

        System.out.println(scopeTest3);



    }
```


It's not same as singelton gof pattern
When we create two bean xml then again new object will be created

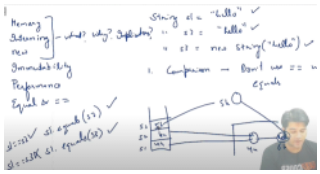
For request scope

```
@RequestMapping("/testing1")
public void test(HttpServletResponse response) throws IOException {

    MyBean myBean1 = context.getBean("myBean",MyBean.class);
    MyBean myBean2 = context.getBean("myBean",MyBean.class);

    System.out.println(myBean1 + " " + myBean2);


}
```

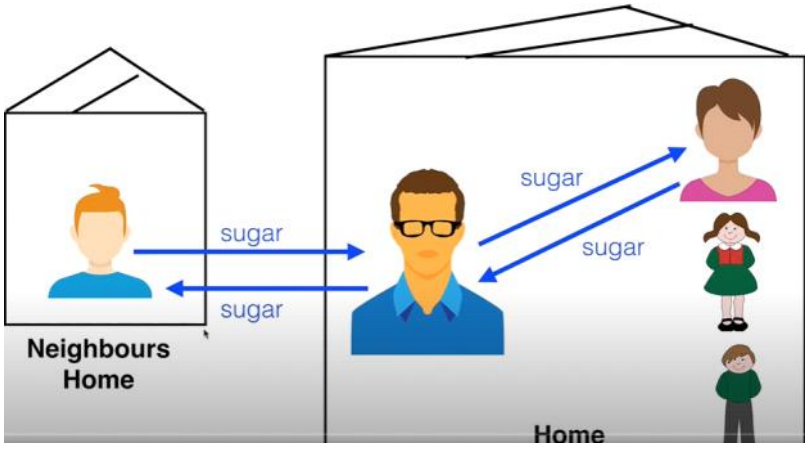
So in this only one my bean object created

Friday, December 18, 2020 8:04 PM

Spring MVC

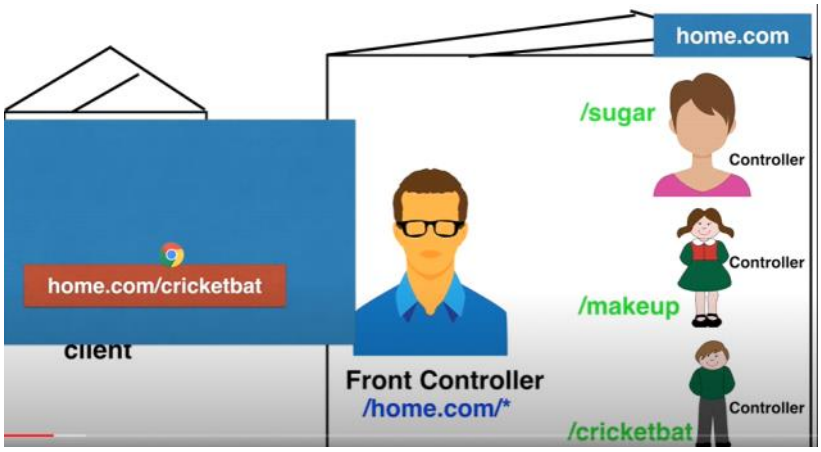


Front CONTROLLER



Any request come outside the home dad knows the avaliable resource so where to dispatch that incoming request

Every request come it is handle by the front controller. He will accept all the request which has url starting with home.com
Routing to other controller




```
<servlet>
    <servlet-name>dad-frontcontroller-dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>dad-frontcontroller-dispatcher</servlet-name>
    <url-pattern>/home.com/*</url-pattern>
</servlet-mapping>
```

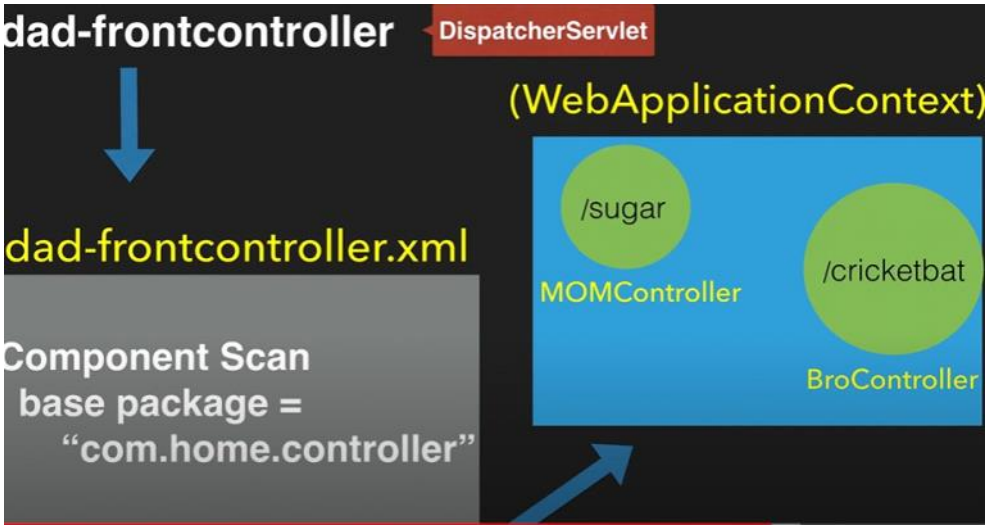
Dispatcher front controller server first point of hitting


```
@Controller
public class MomController {

    @ResponseBody
    @RequestMapping("/sugar")
    public String giveSugar()
    {
        return "Ok..Here is your sugar";
    }

}
```

@Response body map the value to HTTP

To print on webpage we need to use @Response Body

After intilization framework will look for servlet name dad-controller with appeneded with .xml

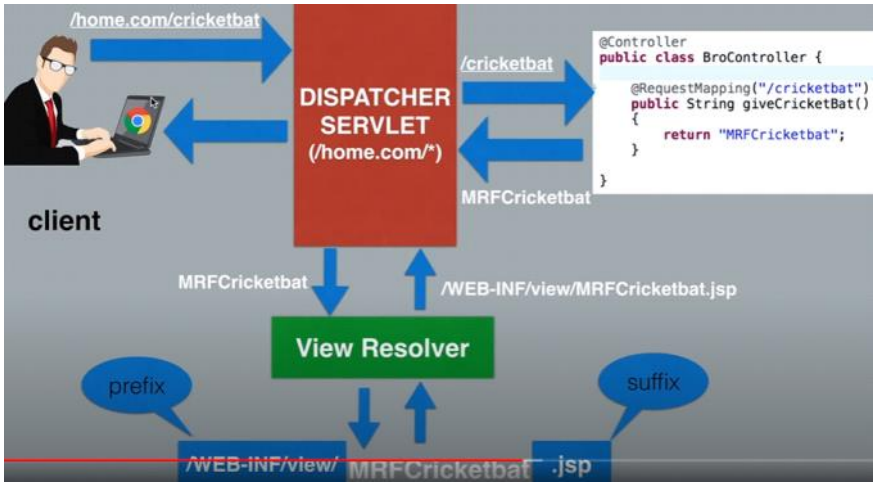



```java
@Controller
public class BroController {

    @ResponseBody
    @RequestMapping("/cricketbat")
    public String giveCricketBat()
    {
        return "MRFCricketbat";
    }

}
```

If we write responseBody annotation then that will write on the web page that string

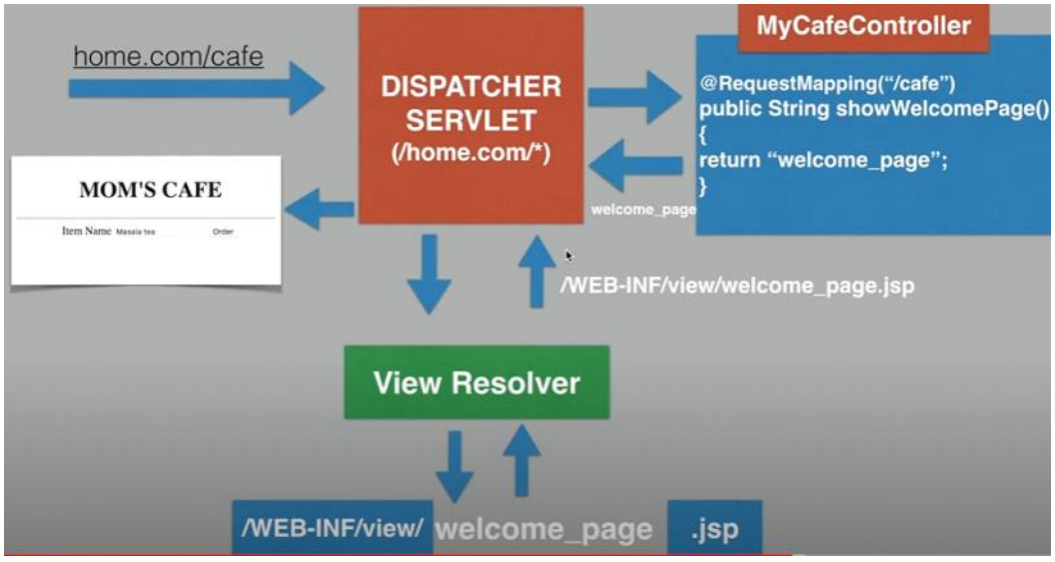
But if we remove the @Response body then contoller will expect a proper web page of name called MRFCricketBat name web page
It is called view name

View resolver--> webinf will change
.jsp will also change
But page name will not change



If we want to send data or string while returning to web page

```
@Controller
public class MyCafeControllers {

    @RequestMapping("/cafe")
    public String showWelcomePage(Model model)
    {
        //sending data to view(jsp page)
        String myName = "Abhilash";

        model.addAttribute("myNameValue", myName);

        return "welcom-page";
    }
```




```java
@RequestMapping("/processOrder")
public String processOrder(HttpServletRequest request,Model model)
{
    //handle the data received from the user
    String userEnteredValue = request.getParameter("foodType");

    //adding the captured value to the model

    model.addAttribute("userInput", userEnteredValue);

    //set the user data with the model object and send it to view
    return "process-order";
}
```

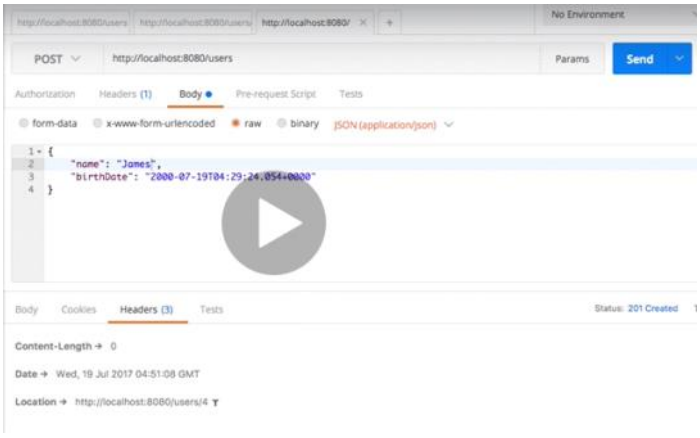
Monday, December 21, 2020 2:33 AM

Reprsentaional state transfer
Maximum use of rest

HTTP ->if we request anything on web browser then it will get the response from server and display in the form of html with all output. It conatins body and header
Swagger->service defination

Post

```
@PostMapping("/users")
ResponseEntity<Object> addvalue(@RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri(); // add /users/id for new id and
    //return 201 created status while posting

    return ResponseEntity.created(uri).build();
}
```



New location url also given in header when we are posting with above code

Exception

```
public class UserNotFoundException extends RuntimeException {

    public UserNotFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
```

```
@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);
    if(user==null)
        throw new UserNotFoundException("id-"+ id);

    return user;
}
```

```java
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeException {
    public UserNotFoundException(String message) {
        super(message);
    }
}
```

```java
ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
        ex.getBindingResult().toString());
```

It will add the exception with proper message

@Valid to request body so that proper format has been added or posted

```java
@PostMapping("/users")
ResponseEntity<Object> addvalue(@Valid @RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri

    // return 201 created status while posting
    return ResponseEntity.created(uri).build();
}
```

```java
@Component
public class User {

    Integer id;
    @Size(min =2)
    String name;
    String address;
```

Hateos

To provide link in the response

```java
if(user==null)
    throw new UserNotFoundException("id-"+ id);


//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());


//HATEOAS

return user;
```

```
2017-07-19 14
2017-07-19 14
2017-07-19 14
2017-07-19 14
```

```
{
    "id": 1,
    "name": "Adam",
    "birthDate": "2017-07-19T09:26:18.337+0000",
    "_links": {
        "all-users": {
            "href": "http://localhost:8080/users"
        }
    }
}
```

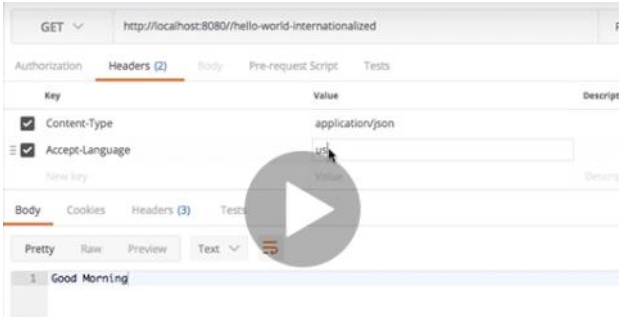
```
//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());

resource.add(linkTo.withRel("all-users"));

//HATEOAS
```

```
#### Internationalization

##### Configuration
- LocaleResolver
   - Default Locale - Locale.US
- ResourceBundleMessageSource

##### Usage
- Autowire MessageSource
- @RequestHeader(value = "Accept-Language", required = false) Locale locale
- messageSource.getMessage("helloWorld.message", null, locale)
```

// default local=us

We can pass as @Request Header accept-language

```
@Autowired
MessageSource messagesource;

@GetMapping("/hellovalue")
String message(@RequestHeader(name="Accept-Language", required = false) Locale locale) {  //
    // as locale needs to be passed using header on based accept-language we need to pass locale in header
    return messagesource.getMessage("good.morning.message", null, locale);
}

@Bean
ResourceBundleMessageSource source() {
    ResourceBundleMessageSource source = new ResourceBundleMessageSource();
    source.setBasename("messages");
    return source;
}
```




```
@Bean
public LocaleResolver localeResolver() {
    AcceptHeaderLocaleResolver localeResolver = new AcceptHeaderLocaleResolver();

    localeResolver.setDefaultLocale(Locale.US);

    return localeResolver;
}
```

Swagger->

```java
@Configuration
@EnableSwagger2
public class Swagger {

    @Bean
    Docket docket()
    {
        return new Docket(DocumentationType.SWAGGER_12);
    }

}
```

```
{
    swagger: "2.0",
  + info: {…},
    host: "localhost:8080",
    basePath: "/",
  + tags: […],
  - paths: {
      + /error: {…},
      + /hello-world: {…},
      + /hello-world-bean: {…},
      + /hello-world-internationalized: {…},
      + /hello-world/path-variable/{name}: {…},
      + /users: {…},
      + /users/{id}: {…}
    },
  + definitions: {…}
}
```

```java
@ApiModel(description="All details about the user. ")
public class User {

    private Integer id;

    @Size(min=2, message="Name should have atleast 2 charac
    @ApiModelProperty(notes="Name should have atleast 2 cha
    private String name;

    @Past
    @ApiModelProperty(notes="Birth date should be in the past")
    private Date birthDate;

    protected User() {

    }
```

Filter out some of the contents and don't want to show much to end user

```java
public class SomeBean {

    private String field1;

    private String field2;

    @JsonIgnore
    private String field3;

@JsonIgnoreProperties(value={"field1","field2"})
public class SomeBean {
```

Dynamic filter

```java
@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1
```

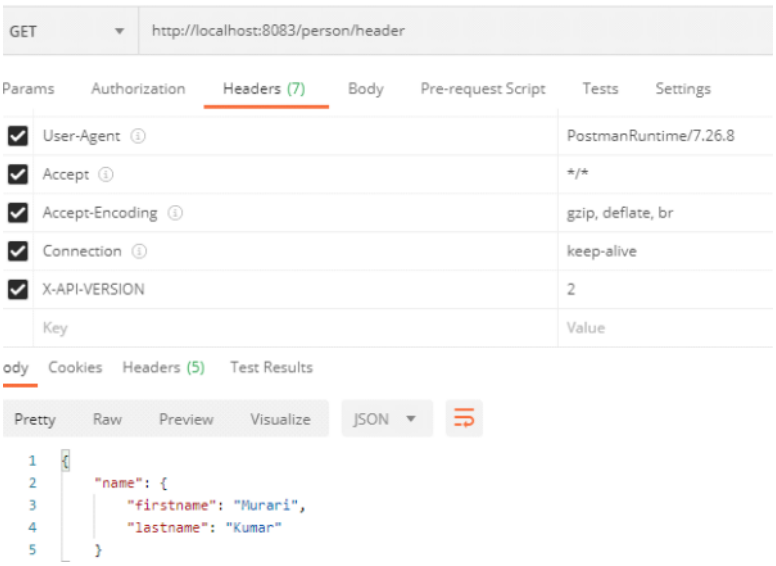
```java
@GetMapping("/filtering")
public MappingJacksonValue retrieveSomeBean(){
    SomeBean someBean = new SomeBean("value1","value2","value3");

    SimpleBeanPropertyFilter filter = SimpleBeanPropertyFilter.
            filterOutAllExcept("field1","field2");

    FilterProvider filters = new SimpleFilterProvider().addFilter("SomeBeanFilt

    MappingJacksonValue mapping = new MappingJacksonValue(someBean);

    mapping.setFilters(filters);

    return mapping;
```

```java
@RestController
public class VersioningController {

    @GetMapping("v1/person")  // url versioning
    PersonV1 personv1()
    {
        return new PersonV1("Murari Kumar");
    }

    @GetMapping("v2/person")
    PersonV2 personv2()
    {
        return new PersonV2(new Name("Murari", "Kumar"));
    }

}
```

Versioning using request param

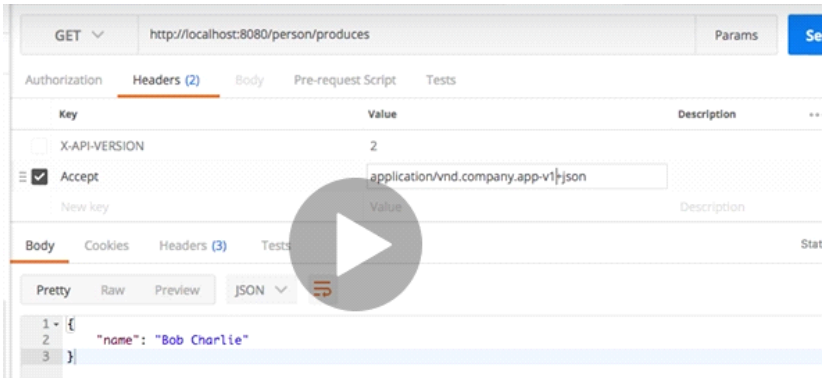
```java
@GetMapping(value = "v1/person", params = "version=1") // versioning using request param
PersonV1 paramv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "v2/person", params = "version=2")
PersonV2 paramv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```

```java
@GetMapping(value = "/person/header", headers = "X-API-VERSION=1") // versioning using request header
PersonV1 headerv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/header", headers = "X-API-VERSION=2")
PersonV2 headerv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Versioning using produces

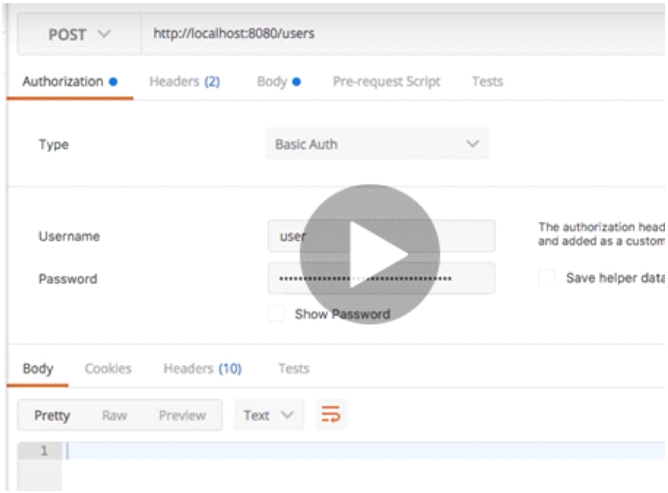
```
@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v1+json") // versioning using produces
PersonV1 producesv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v2+json")
PersonV2 producesv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Header request problem in caching difficult // cannot use from browser// swagger and documentation difficulty

Still cache using uri

Security password sending in authorization



H2 DATABASES FECTHING DATA

```
@Entity
public class UserDTO {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    Long phone;
```

```
server.port=8083
spring.jpa.show-sql=true
 #this start logging jpa
spring.h2.console.enabled=true
```

```
@RestController
public class UserJPAResource {

    @Autowired
    UserRepositroy user;

    @GetMapping("/findall")
    List<User> findAll() {
        return user.findAll();
    }

    @GetMapping("/findId/{id}")
    Optional<User> findById(@PathVariable int id
        return user.findById(id);
    }

    @PostMapping("/users")
    void addUser(@RequestBody User u) {
        user.save(u);
    }

    @DeleteMapping("/users/{id}")
    void deleteuser(@PathVariable int id) {
        user.deleteById(id);
```

```
@Repository
public interface UserRepositroy  extends JpaRepository<User, Integer>{

}
```

Many to one relationship

User can have many post
Post will have same user id for all post

```java
@Entity
public class Post {

    @Id
    @GeneratedValue
    private Integer id;
    private String description;

    @ManyToOne(fetch=FetchType.LAZY)
    @JsonIgnore
    private User user;

    public Integer getId() {
        return id;
    }
```

To get all his post

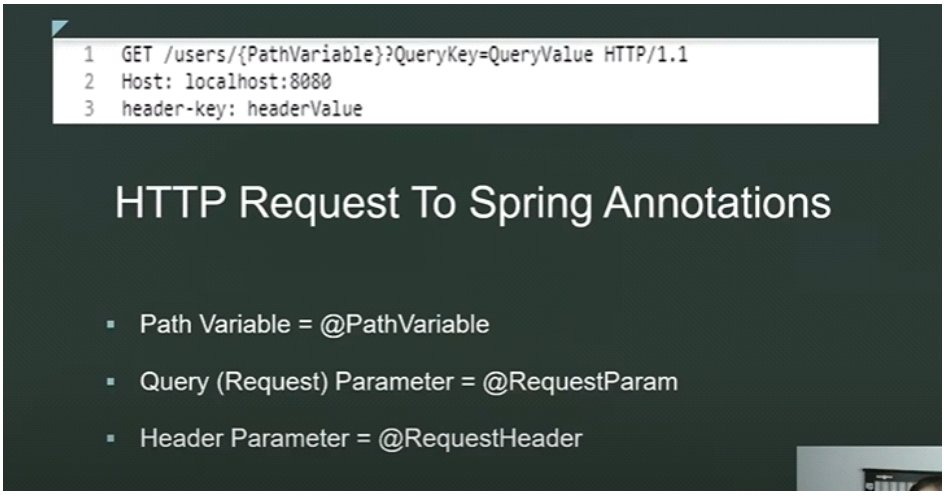
```java
@Entity
public class User {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    int phone;
    @OneToMany(mappedBy = "user")
    List<Post> post;

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

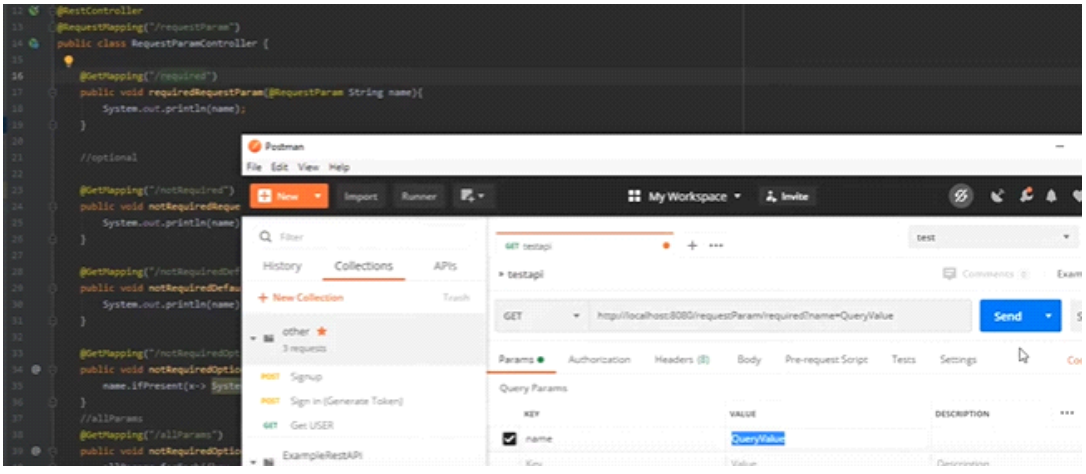
```java
@GetMapping("/jpa/users/{id}/posts")
public List<Post> retrieveAllUsers(@PathVariable int id) {
    Optional<User> userOptional = userRepository.findById(id);

    if(!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    return userOptional.get().getPosts();
}
```

```java
@Entity
public class Post {
    @Id
    @GeneratedValue
    int id;
    String descrription;
    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnore  // infinite loop as both will start calling each other so we put ignore
    User user;

    public int getId() {
```

```java
@PostMapping("/jpa/users/{id}/posts")
public ResponseEntity<Object> createUser(@PathVariable int id, @RequestBody Post post)
{
Optional<User> userOptional= userRepository.findById(id);
if(!userOptional.isPresent())
{
throw new UserNotFoundException("id: "+ id);
}
User user=userOptional.get();
//map the user to the post
post.setUser(user);
//save post to the database
postRepository.save(post);
//getting the path of the post and append id of the post to the URI
URI location=ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(post.getId()).toUri();
//returns the location of the created post
return ResponseEntity.created(location).build();
}
```

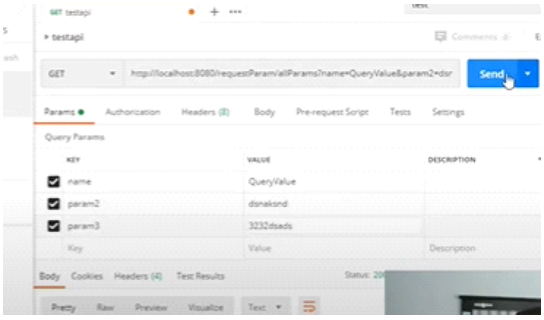
When posting post from postman while in list post from user

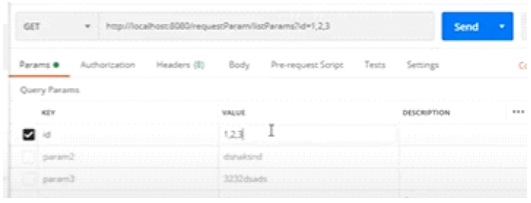




Query parammater using map @RequestParam

```
//allParams
@GetMapping("/allParams")
public void notRequiredOptional(@RequestParam Map<String,String> allParams ){
    allParams.forEach((key, value) -> {
        System.out.println(String.format("Request Param '%s' = %s", key, value));
    });
}

@GetMapping("/listParams")
public void paramList(@RequestParam List<Integer> id ){
    id.forEach(eachId-> System.out.println(eachId));
}
```





List param

When we want parse api json

```
@GetMapping("/restore")
List<BitCoin> storevalue() {
    RestTemplate rest = new RestTemplate();
    ResponseEntity<List<BitCoin>> rateResponse =
            rest.exchange("https://bitpay.com/api/rates",
                    HttpMethod.GET, null, new ParameterizedTypeReference<List<BitCoin>>() {
                });

    return reposiotry.save(rateResponse.getBody());
```

```
@SpringBootApplication
//@EnableDiscoveryClient
@ComponentScan(basePackages = "com.*")
@EntityScan("com.*")
public class CurrencyExchangeServiceApplication {
```

https://www.foreach.be/blog/spring-cache-annotations-some-tips-tricks

Spring cache to search using ehcache
Spring Cache Example using EhCache in Spring Boot | Tech Primers

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="ehcache.xsd"
         updateCheck="true"
         monitoring="autodetect"
         dynamicConfig="true">

    <diskStore path="java.io.tmpdir" />

    <cache name="usersCache"
           maxEntriesLocalHeap="10000"
           maxEntriesLocalDisk="1000"
           eternal="false"
           diskSpoolBufferSizeMB="20"
           timeToIdleSeconds="300" timeToLiveSeconds="600"
           memoryStoreEvictionPolicy="LFU"
           transactionalMode="off">
        <persistence strategy="localTempSwap" />
    </cache>
```

```java
@EnableJpaRepositories(basePackages = { "com.techprimers.cache.repository", "com.techprimers.cache" })
@EnableCaching
@Configuration
public class EhCacheConfiguration {

    @Bean
    public CacheManager cacheManager() {
        return new EhCacheCacheManager(cacheMangerFactory().getObject());
    }

    @Bean
    public EhCacheManagerFactoryBean cacheMangerFactory() {
        EhCacheManagerFactoryBean bean = new EhCacheManagerFactoryBean();
        bean.setConfigLocation(new ClassPathResource("ehcache.xml"));
        bean.setShared(true);
        return bean;
    }
}
```

```java
@Component
public class UsersCache {

    @Autowired
    UsersRepository usersRepository;

    @Cacheable(value = "usersCache", key = "#name")
    public Users getUser(String name) {
        System.out.println("Retrieving from Database for name: " + name);
        return usersRepository.findByName(name);
    }
}
```

```
timeToIdleSeconds="300"
timeToLiveSeconds="600">
```

If it has not been used for 300 seconds
Or after stays in cache for 600 seconds
When echache exceed from its configured size ehcache removed expired elements
If that doesn't clear enough space it cleasr object from ehcache
By default it remove LRU elements

Second level caching is we put on entity or whole object

```java
@Configuration
@EnableCaching
public class CacheConfig {

    @Bean
    public CacheManager cacheManager() {
        SimpleCacheManager cacheManager = new SimpleCacheManager();
        List<Cache> cacheList = new ArrayList<Cache>();
        cacheList.add(new ConcurrentMapCache("userCache"));
        cacheList.add(new ConcurrentMapCache("addressCache"));
        cacheManager.setCaches(cacheList);
        return cacheManager;
    }
}
```

If we don’t want to configure in ehache.xml

Evict cache -> If we now want to clear cache just evict it

```
@RequestMapping(value = "/greetings")
@Cacheable("greetings")
public Collection<Greeting> getGreetings() throws InterruptedException {
    Thread.sleep(5000);
    return greetingMap.values();
}

@RequestMapping(value = "/greetings", method=RequestMethod.POST)
public String createGreeting(@RequestBody Greeting g) {
    greetingMap.put(g.getId(), g);
    return "Greeting is saved successfully";
}

@RequestMapping(value="/evictcache")
public String evictCache() {
    return "Cache is cleared successfully";
}
```

```
@RestController
 public class CachingController {
@Autowired
 CachingService cachingService;
@GetMapping("clearAllCaches")
public void clearAllCaches() {
 cachingService.evictAllCaches(); } }
```

From <https://www.baeldung.com/spring-boot-evict-cache>

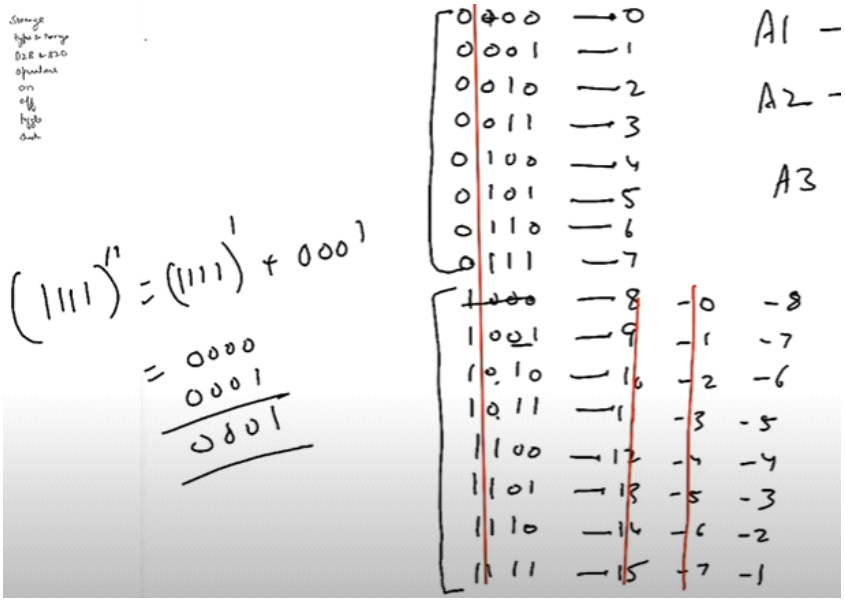
3.4. Sending HTTP Headers using RestTemplate

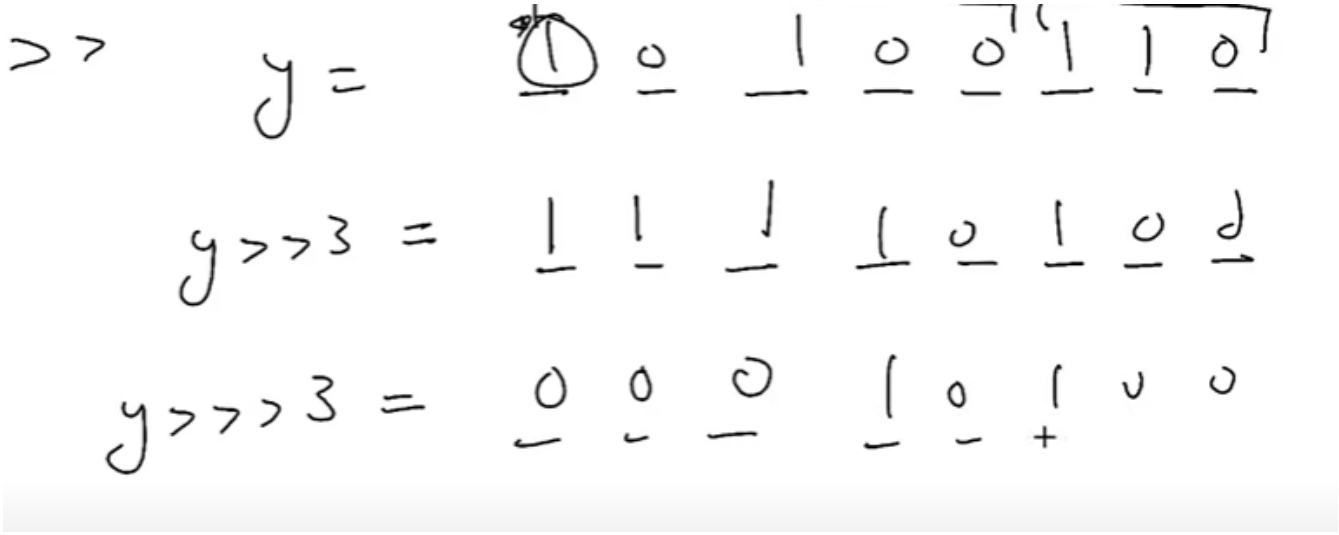
```
private static void getEmployees()
{
    final String uri = "http://localhost:8080/springrestexample/employees";
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
    headers.set("X-COM-PERSIST", "NO");
    headers.set("X-COM-LOCATION", "USA");

    HttpEntity<String> entity = new HttpEntity<String>(headers);

    ResponseEntity<String> response = restTemplate.exchange(uri, HttpMethod.GET, entity, String.class);

    //Use the response.getBody()
}
```

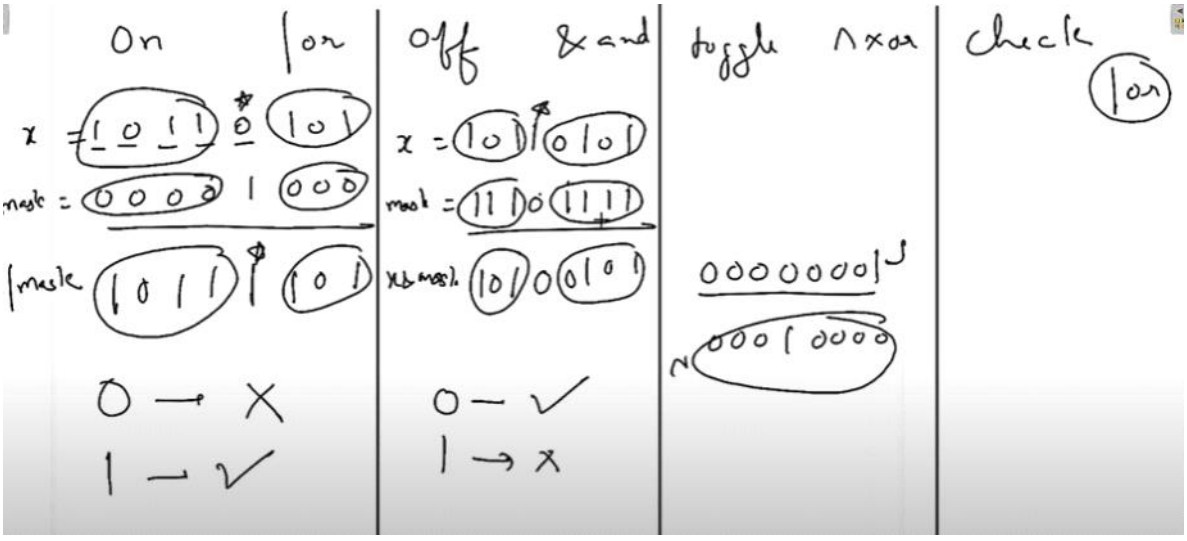
3.5. Sending URL Parameters using RestTemplate

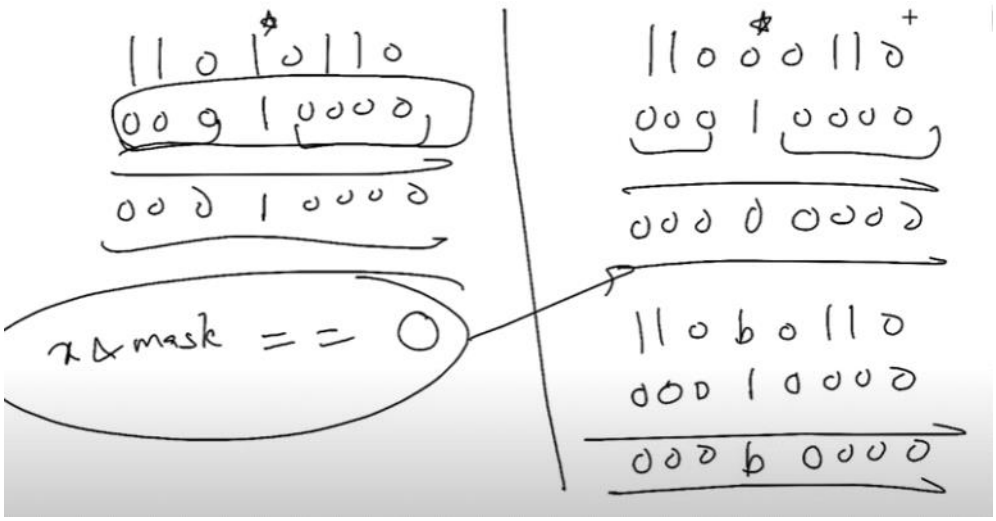
```
private static void getEmployeeById()
{
    final String uri = "http://localhost:8080/springrestexample/employees/{id}";
    RestTemplate restTemplate = new RestTemplate();

    Map<String, String> params = new HashMap<String, String>();
    params.put("id", "1");

    EmployeeVO result = restTemplate.getForObject(uri, EmployeeVO.class, params);

    //Use the result
}
```

From <https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/>


A1 -
A2 -
A3


>>
y =
y>>3 =
y>>>3 =


On
|or
Off
&and
toggle
^xor
check
|or
mask
xb mask

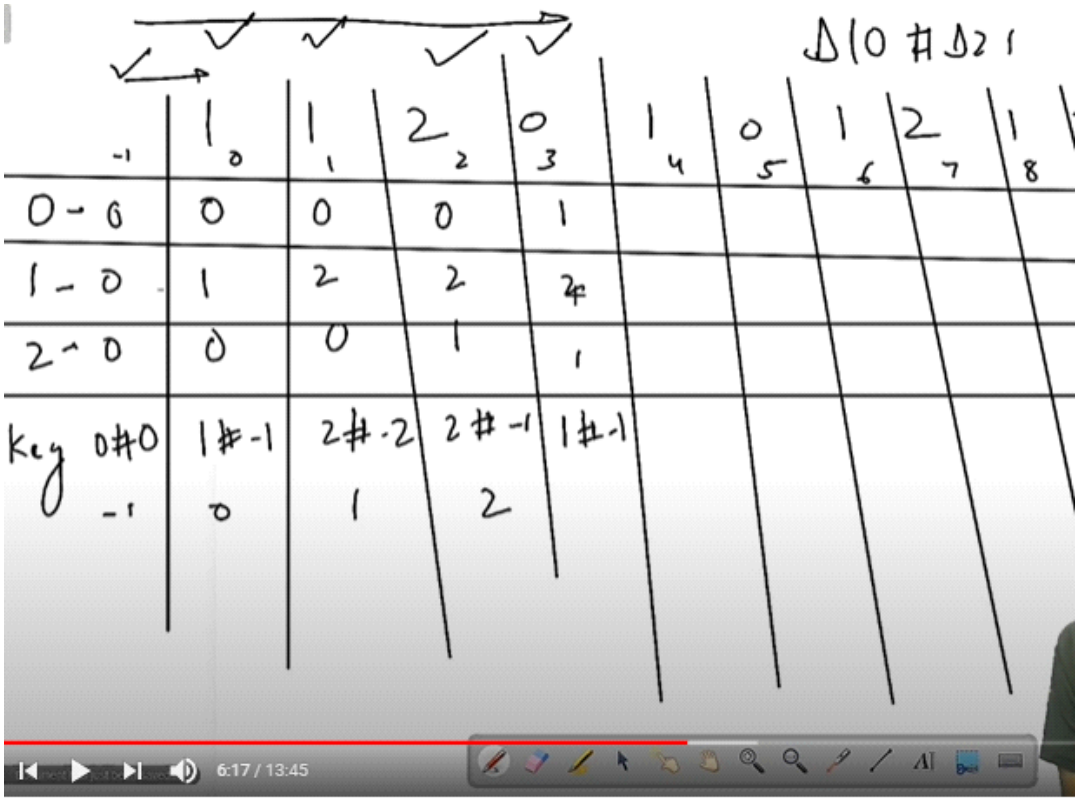



```
//write your code here
int rsbm = n & -n;

System.out.println(Integer.toBinaryString(
```

Most significtnt one -> value & 2's compliment value

```
boolean flag = false;
int rev = 0;
int j = 0;

for(int i = 31; i >= 0; i--){
    int mask = (1 << i);

    if(flag){
        if((n & mask) != 0){
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
        } else {
            System.out.print(0);
        }

        j++;
    } else {
        if((n & mask) != 0){
            flag = true;
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
            j++;
        } else {
        }
    }
}
```

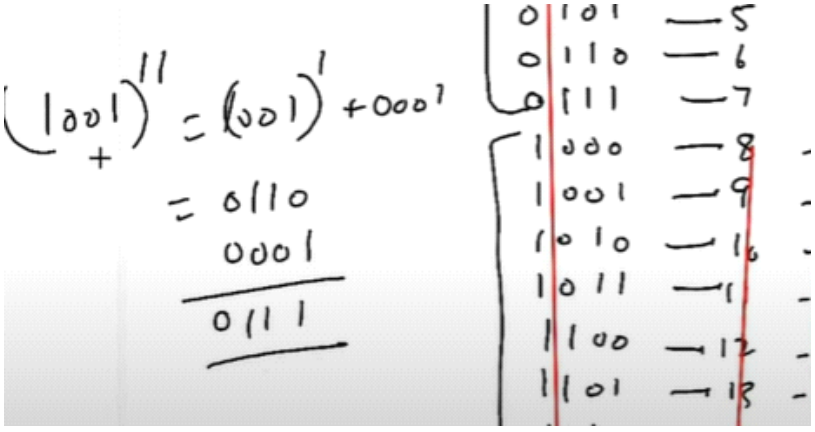
Reverse bit



ashmaps #datastructure #algorithms

ongest Subarray with Equal 0s 1s and 2s | Hashmap Interview Questions Playlist

i2 views • Oct 25, 2020

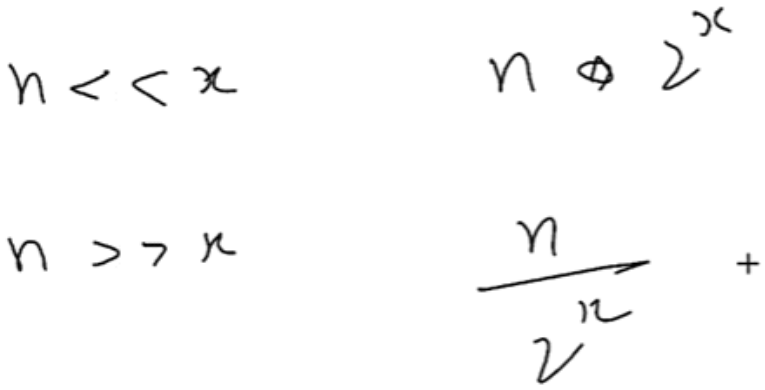


One compliment toggle whole value
Add+1

```
intv=13;
intw=1<<2;   ~0010=  1101
System.out.println(v&(~w));
Bits off formula at particular position

Int v=13
Int w =1<<2
System.out.println(v|w) bits on

System.out.println(Integer.toBinaryString(v));
intw=(~v);
System.out.println(Integer.toBinaryString(v&(w+1)));
intvd=(24&-24);
System.out.println(Integer.toBinaryString(vd));
```

```
Sum of value

Scannersc=newScanner(System.in);
intsum=0;
for(inti=0;i<4;i++){
System.out.println("Entervalue");
sum=sum|1<<sc.nextInt();
}
System.out.println(sum);


Value=32
Value=31===(1<<5)-1

25|12
sum|people[i]


System.out.println(((n<<3)-n)>>3);  =>>8n-n/8

System.out.println(n<<1); n=8->16
System.out.println(n<<2); n=8->32
System.out.println(n<<3);n=8->64
System.out.println(n>>3);n=64->8
```

```
Quotient remainder
If divide by 4  last 2 will be reminader
15|4
11| 11

If divide by 8  last 3 will be reminader
15|8
1|111


If we shift by any number then it will become twice

111<<1-> 111=7
1110->  1*2+1*4+1*8

If we shift by two then it will multiplied by 4
```



Threads->

Join-> When two thread running then main thread will wait to execute the bot the thread. Because both the thread become after join

Inter thread communictaion-> All should be synchronized
Notify()-> Whent the thread is in waiting state then notify the waiting state to start the resuming the thread executiom
Wait()->When the thread is waiting it is waiting for notify to start execution

Executorservice provide thread pool to execute the task which will automatically work when one thread done the work

Creating the thread is expensive task
Fixed number of thread

```
packagepractice;

importjava.util.concurrent.ExecutorService;
importjava.util.concurrent.Executors;

classCounterimplementsRunnable{
staticintid=0;
intcount;

publicCounter(){
this.count=0;
}

publicsynchronizedintgetCount()throwsInterruptedException{
wait();
this.count=count+1;
System.out.println(count);
returncount;
}


@Override
publicvoidrun(){
try{
getCount();
}catch(Exceptione){

}

}
}

classCounter1implementsRunnable{
staticintid=0;
intcount;

publicCounter1(){
this.count=0;
```

```
}

publicsynchronizedvoidgetCount()throwsInterruptedException{

while(true){
this.count=count+1;
System.out.println(count);

notify();
}
}

@Override
publicvoidrun(){
try{
getCount();

}catch(Exceptione){

}
}
}

publicclassApplictaion{
publicstaticvoidmain(String[]args)throwsInterruptedException{
ExecutorServiceexecutorService=Executors.newFixedThreadPool(4);
executorService.execute(newThread(newCounter()));
executorService.execute(newThread(newCounter1()));
}
}
```

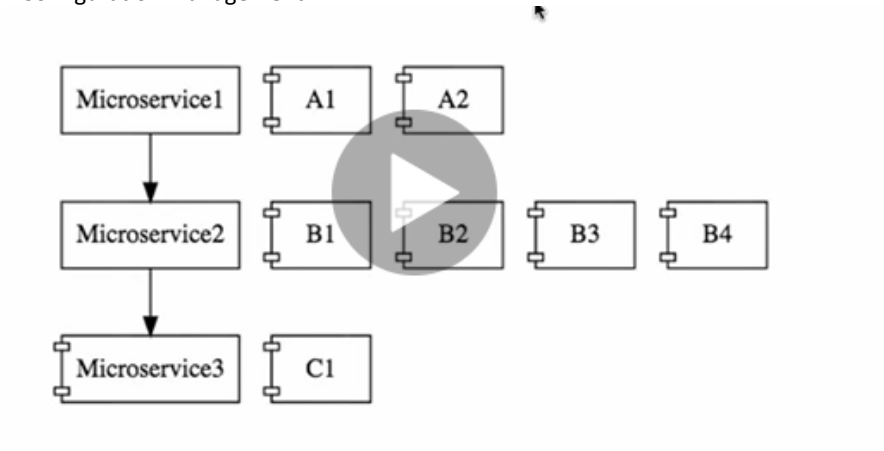
Saturday, December 26, 2020 2:55 AM

Microservice

1.Bounded Context

Earlier we were building one monolthic app now 10-20 microservicce
How to identify what to do in each of microservices
What to do or not?

2.Configuration Management



5 microservice with different number of instances

3 Dynamic scale up and scale down

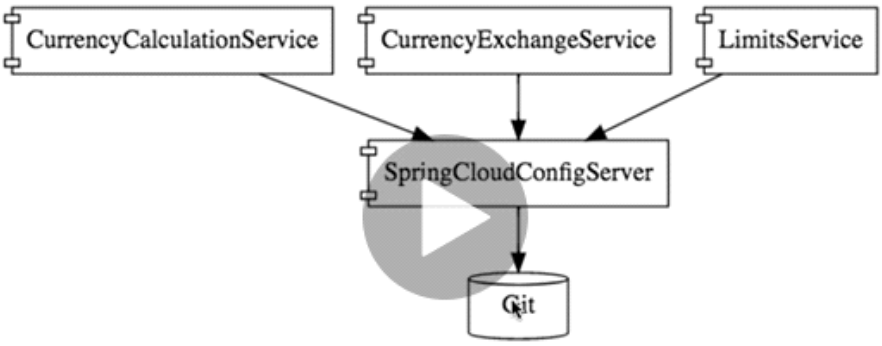
Load different instance at different time
At particular time I want two instance and after that I don't. Bring down when we don't want

4. Visibility

How to identify where is bug. We want centralized and monitor tha logs and server which is up and down

5 Pack of cards
If one microservice fail then all will fail

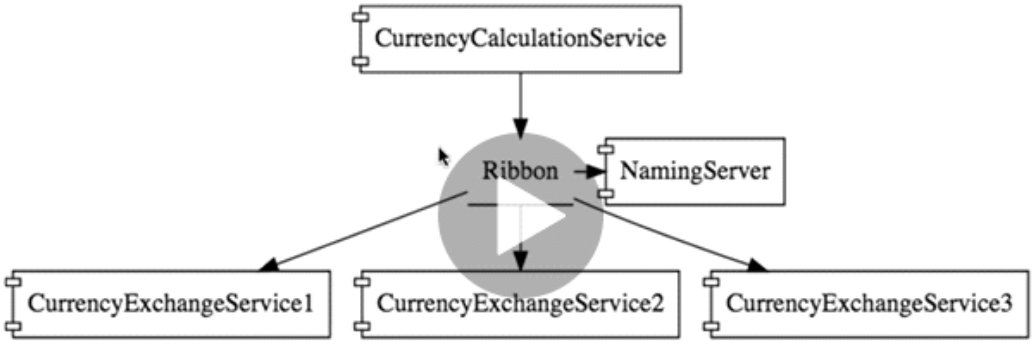
Spring cloud



Spring Cloud Config Server

We can store all the configuration different server and instance to one place…And easy to maintain

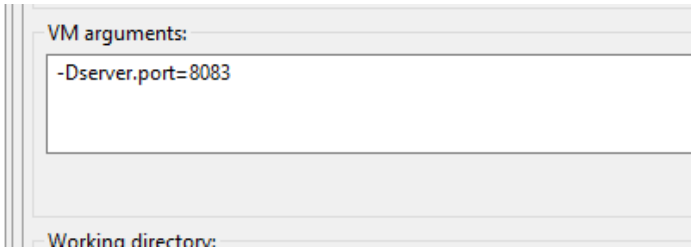
Dynamic scale up and down



Ribbon Load Balancing

Setting property from application properties

```
@Autowired
org.springframework.core.env.Environment env;

@GetMapping("/exchange/from/{from}/to/{to}")
ExchangeValue exchangevalue(@PathVariable String from, @PathVariable String to) {
    ExchangeValue e = new ExchangeValue(1L, from, to, BigDecimal.valueOf(65));
    e.setPort(Integer.parseInt(env.getProperty("server.port")));
    return e;
}
```




```
public interface CurrencyExchangeRepository extends JpaRepository<ExchangeValue, Long> {
    ExchangeValue findByFromAndTo(String from, String to);

}
```

WHEN WE WANT TO FIND BY FROM AND TO String then jpa repo query

Response mapping from other localhost

```
Map<String, String> urivariables = new HashMap<String, String>();
        urivariables.put("from", from);
        urivariables.put("to", to);
        ResponseEntity<ConversionValue> responseEntity =  restTemplate().getForEntity(
                "http://localhost:8081/exchange/from/{from}/to/{to}", ConversionValue.class,
                urivariables);
        ConversionValue body = responseEntity.getBody();
```

Rest template alternative feign

```
@FeignClient(name = "currency-exchange-service", url = "localhost:8081")
public interface CurrencyExchangeServiceProxy {
    @GetMapping("/exchange/from/{from}/to/{to}")
    ConversionValue exchangevalue(@PathVariable("from") String from, @PathVariable("to") String to);
}
```

Post for object
Post for entity
Exchange method of Spring RestTemplate - Part 1 || Calling REST API using RestTemplate

@Get from other api

```
@GetMapping("/conversion-object/from/{from}/to/{to}/quantity/{quantity}")
ConversionValue getvaluemapping(@PathVariable String from, @PathVariable String to) {
    Map<String, String> urivariables = new HashMap<String, String>();
    urivariables.put("from", from);
    urivariables.put("to", to);
    MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
    headers.add("Content-Type", "application/json");
    headers.add("Authorization", "tokenxxx");
    ResponseEntity<ConversionValue> entity = new RestTemplate().exchange(
            "http://localhost:8081/exchange/from/{from}/to/{to}", HttpMethod.GET, new
            HttpEntity<Object>(headers),
```

```
                ConversionValue.class,urivariables);

        return entity.getBody();

}
  @GetMapping("/conversion-object/from/{from}/to/{to}/quantity/{quantity}")
  ConversionValue getvaluemapping(@PathVariable String from, @PathVariable String to) {
      Map<String, String> urivariables = new HashMap<String, String>();
      urivariables.put("from", from);
      urivariables.put("to", to);
      MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
      headers.add("Content-Type", "application/json");
      headers.add("Authorization", "tokenxxx");
      ResponseEntity<ConversionValue> entity = new RestTemplate().exchange(
              "http://localhost:8081/exchange/from/{from}/to/{to}", HttpMethod.GET, new HttpEntity<Object>(headers),
              ConversionValue.class,urivariables);

      return entity.getBody();

  }
```

Post Mapping

```
   @PostMapping("/conversion-object")
   ResponseEntity<ProxyConversion> getvaluemapping(@RequestHeader(value = "Username") String username,
           @RequestHeader(value = "Password") String password, @RequestBody ProxyConversion conversion) {
       MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
       headers.add("Username", username);
       headers.add("Password", password);
       ProxyConversion proxy = new ProxyConversion(11L, "ABC", "DEB", BigDecimal.valueOf(23));
       HttpEntity<Object> httpEntity = new HttpEntity<Object>(proxy, headers);
       adduserexchangeforpost(httpEntity);

       return adduserexchangeforpost(httpEntity);
   }

   private ResponseEntity<ProxyConversion> adduserexchangeforpost(HttpEntity<Object> httpEntity) {
       ResponseEntity<ProxyConversion> exchange = new RestTemplate().exchange("http://localhost:8081/exchange",
               HttpMethod.POST, httpEntity, ProxyConversion.class);
       return exchange;

   }


  ResponseEntity<String> getvaluemappingpost() {
      MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
      headers.set("Content-Type", "application/json");
      ProxyConversion proxy = new ProxyConversion(11L, "ABC", "DEB", BigDecimal.valueOf(23));
      HttpEntity<Object> httpEntity = new HttpEntity<Object>(proxy, headers);
      adduserexchangeforpost(httpEntity);

      return adduserexchangeforpost(httpEntity);
  }

  private ResponseEntity<String> adduserexchangeforpost(HttpEntity<Object> httpEntity) {
      ResponseEntity<String> exchange = new RestTemplate().exchange("http://localhost:8081/exchange",
              HttpMethod.POST, httpEntity, String.class);
      return exchange;

  }
```

Pagebale concept

```java
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public List<UserDto> getUsers(int page, int limit) {
        List<UserDto> returnValue = new ArrayList<>();

        Pageable pageableRequest = PageRequest.of(page, limit);
        Page<UserEntity> users = userRepository.findAll(pageableRequest);
        List<UserEntity> userEntities = users.getContent();

        for (UserEntity userEntity : userEntities) {
            UserDto userDto = new UserDto();
            BeanUtils.copyProperties(userEntity, userDto);
            returnValue.add(userDto);
        }

        return returnValue;
    }

}
```

https://www.javainuse.com/spring/SpringBootUsingPagination

```java
@RequestParam(value = "page", defaultValue = "0") int page
```

ıd the limit request parameter our method above has the **limit** method argument:

```java
@RequestParam(value = "limit", defaultValue = "30") int limit
```

21

Apart from these mentioned differences in framework, one major difference is `@RequestParam` will always expect a value to bind. Hence, if value is not passed, it will give error. This is not the case in `@QueryParam`

`Query param if we have already the value then we search using query param`

From <https://stackoverflow.com/questions/26709560/what-is-the-difference-b-w-requestparam-and-queryparam-anotation>

Redis server

```java
@SpringBootApplication
public class CurrencyConversionApplication {

    @Bean
    JedisConnectionFactory factory() {
        return new JedisConnectionFactory();
    }

    @Bean
    RedisTemplate<String, User> getredis() {
        RedisTemplate<String, User> redis = new RedisTemplate<String, User>();
        redis.setConnectionFactory(factory());
        return redis;
    }

    public static void main(String[] args) {
        SpringApplication.run(CurrencyConversionApplication.class, args);
    }

}
```

Hashoperation we cannot use reddis server directly we need to use via hashoperation

```java
@Repository
public class UserRepositoryImpl implements UserRepository {

    private RedisTemplate<String, User> redisTemplate;

    private HashOperations hashOperations;


    public UserRepositoryImpl(RedisTemplate<String, User> redisTemplate) {
        this.redisTemplate = redisTemplate;

        hashOperations = redisTemplate.opsForHash();
    }

    @Override
    public void save(User user) {
        hashOperations.put("USER", user.getId(), user);
    }

    @Override
    public Map<String, User> findAll() {
        return hashOperations.entries("USER");
    }

    @Override
    public User findById(String id) {
        return (User)hashOperations.get("USER", id);
    }

    @Override
    public void update(User user) {
        save(user);
    }
```

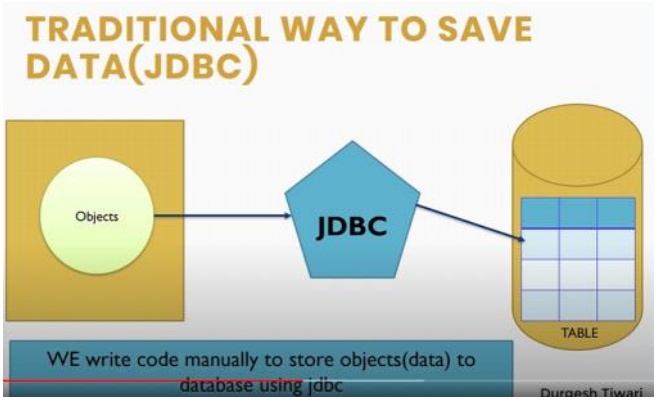
Saturday, December 26, 2020 3:50 AM

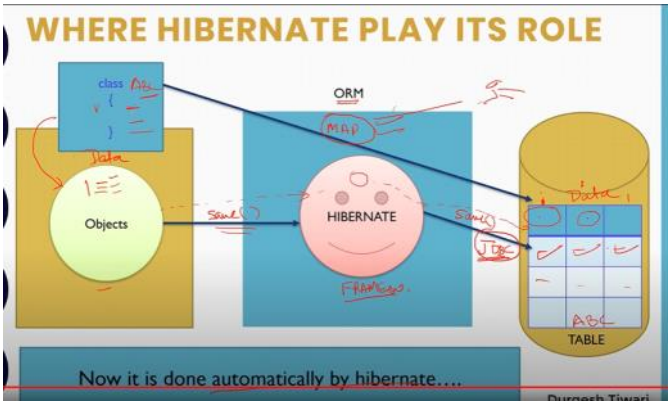
Hibernate

#0. Hibernate Tutorial | Course Overview | Prerequisite of hibernate course | hindi

We don't need to worry about sql query it will be performed by hibernate

- Hibernate is a Java framework that simplifies the development of Java application to interact with the database.
- Hibernate **is ORM (Object Relational Mapping)** tool.
- Hibernate is an Open source, lightweight.
- Hibernate is **a non-invasive** framework, means it won't forces the programmers to extend/implement any class/interface.
- It is invented by **Gavin King in 2001.**
- **Any type of application can build with Hibernate Framework.**





There are 3 objects in hibernate
1.Transient-> Normal object
2.Persistent- object to hibernate object
3.Detached-> hibernate to normal object

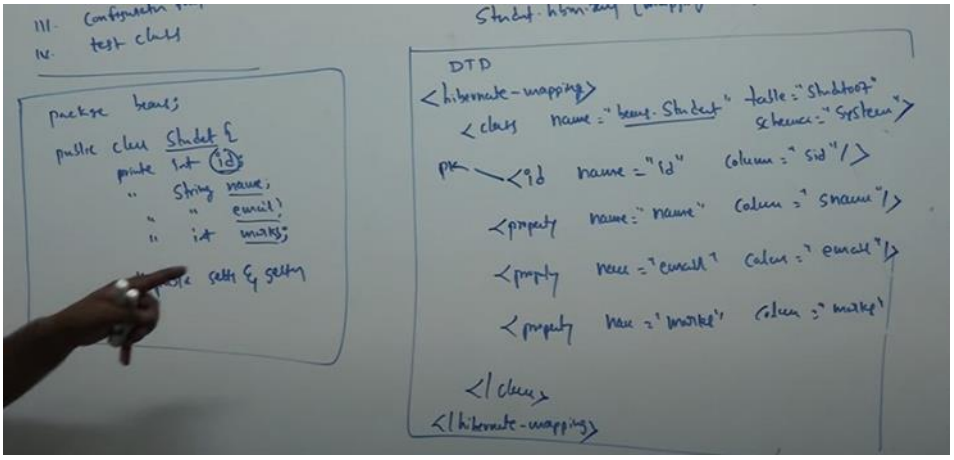
Here we have to make crud operation using object

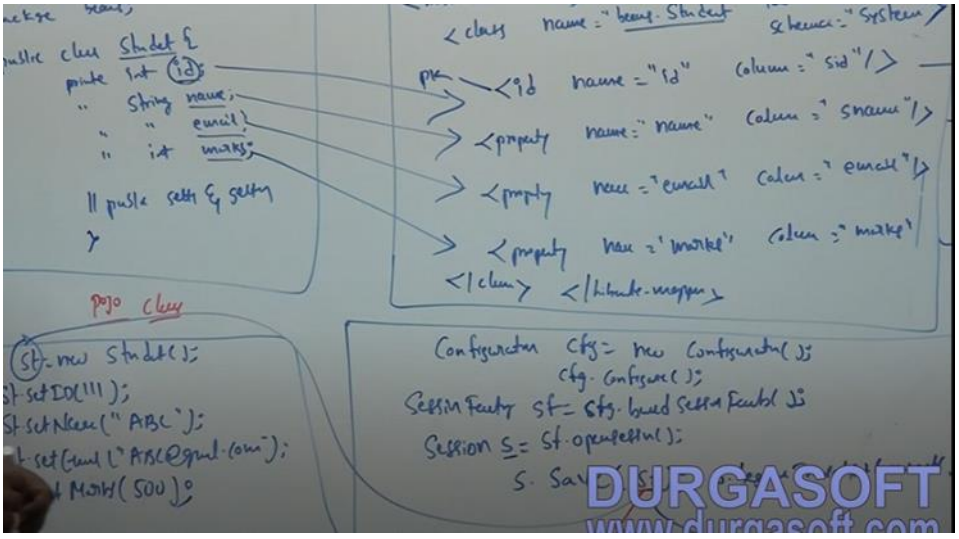
Transient object is normal object

Once we attached to hibernate.. This attached is called persistent object

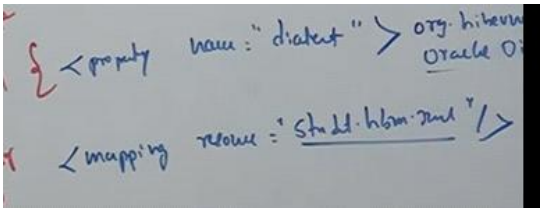
If we remove object from hibernate then it is called detached hiberante

If we do transaction.commit() then it will move database state or permanent state

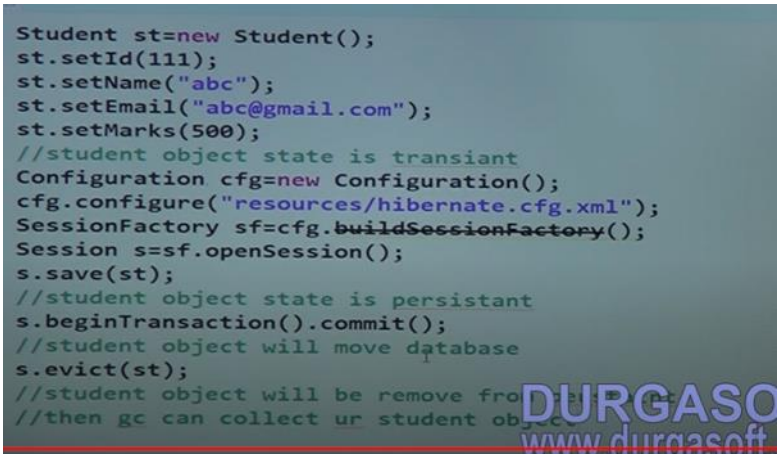




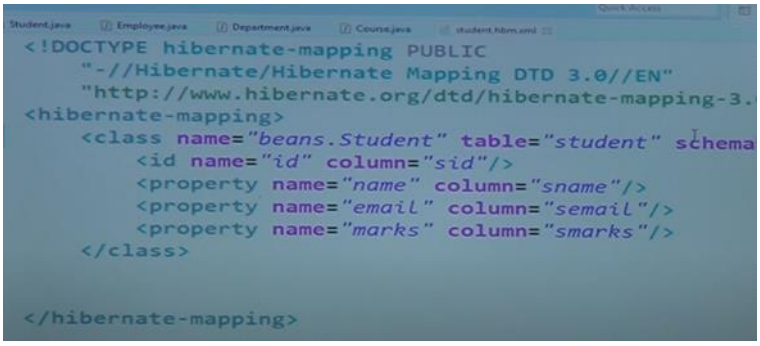
This pojo class is temporary state it can anytime collected by garbage collector
So to avoid that we need to add that in s.save

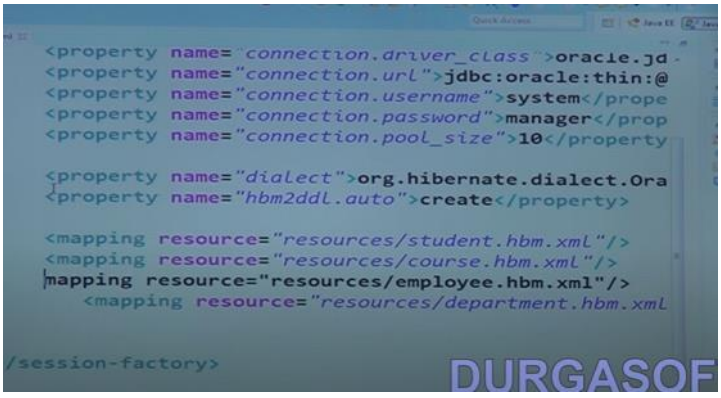


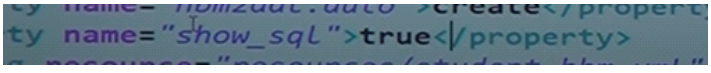
We need to attach mapping file in hibernate.config.file



Hbm2ddl.auot> create<
Then it will automatically create table with dropping earlier schema



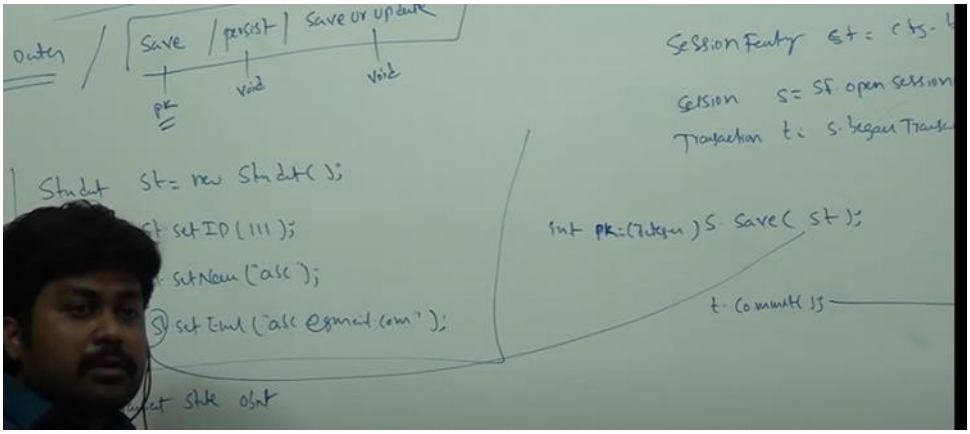




Internal table creation

Hbm2ddl.auot> Update<
Then it will automatically create table with dropping earlier schema

Validate
Must have that 5 coloumns



s.save(st)will return primary key
s.persist(st) will return void

For update records we have update and merge

Update will not work in cache method so then we need to do with merge method

Serialzible

```
public static void main(String[] args) throws IOException, Exception {
    Save s= new Save();
    s.id=10;
    File f= new File("C:\\Users\\kumarmur\\Desktop\\output.txt");
    FileOutputStream fl= new FileOutputStream(f);
    ObjectOutputStream dl= new ObjectOutputStream(fl);
    dl.writeObject(s); //state of object means s.id=10 will be stored


    FileInputStream fl1= new FileInputStream(f);
    ObjectInputStream dl1= new ObjectInputStream(fl1);
    Save s1=(Save) dl1.readObject();
    System.out.println(s1.id);

}

static class Save implements Serializable { // we cannot allow to store object bydefault
    // it can be used for malicious purpsoe so we use serialize interface
    int id;
```

If we excute saveorupdate(st) opeartion

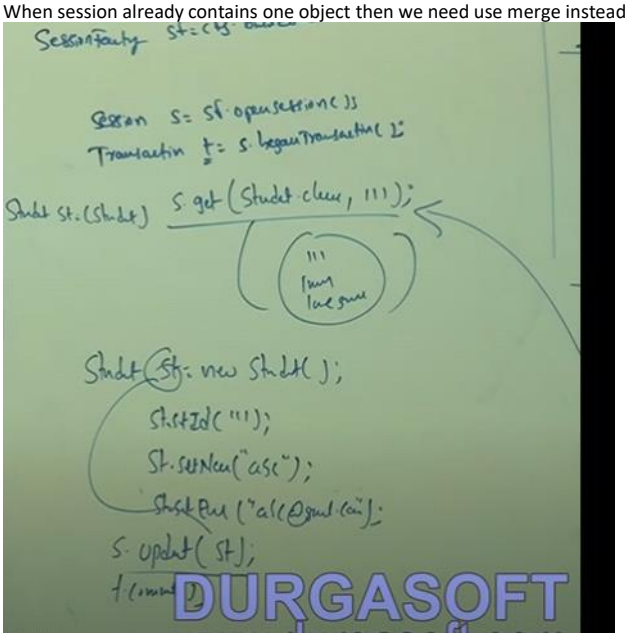
First it will run select query on that primary id;
Then it will compare

Using s.update(St);
s.merge(st);

Limitation
We can only update non primary keys
It is not possible to update one coloumn

When to use merge

When session already contains one object then we need use merge instead update otherwise it will throw exception of duplicate



```
Transaction t = session.beginTransaction();

session.get(Student.class, 111);

Student st=new Student();
st.setId(111);
st.setName("ABC");
st.setEmail("ABC@GMAIL.COM");
st.setAddress("HYD");

session.update(st);
t.commit();
session.close();
sf.close();
System.out.println("update success");
```

```
public static void main(String[] args) {

    Configuration cfg = new Configuration();
    cfg.configure("resources/oracle.cfg.xml");
    SessionFactory sf = cfg.buildSessionFactory();
    Session session = sf.openSession();
    Transaction t = session.beginTransaction();

    Student st=new Student();
    Student st=new Student();
```

```
Student st=new Student();

st.setId(222);

session.delete(st);
t.commit();
session.close();
sf.close();
System.out.println("");
```

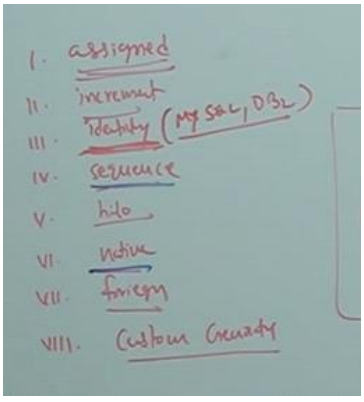
```
Object o=session.get(Student.class, 111);

Student st=(Student)o;
System.out.println(st.getId());
System.out.println(st.getName());
System.out.println(st.getEmail());
System.out.println(st.getAddress());


session.close();
sf.close();
System.out.println("select success");
```

It will do select operation only if when we do get of non primary key in session.load(Student.class, 11);

Primary key auto genrated
1.
Assigned (default)-> It is handle by user itself

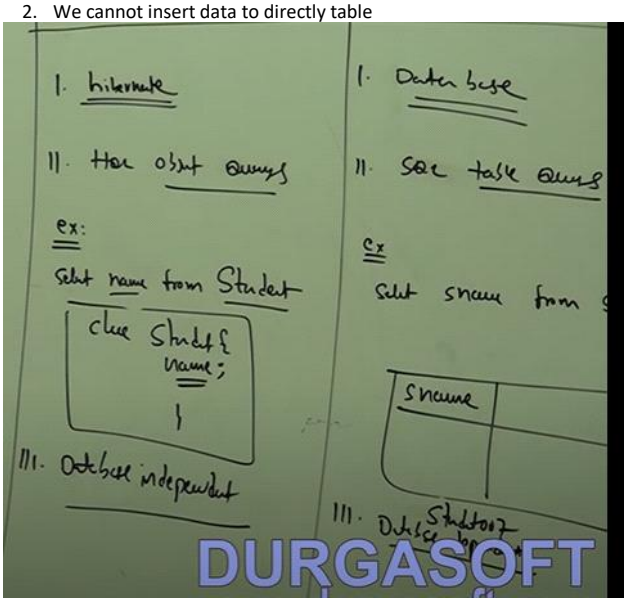
2.Increment-> It will do select max(id) from record and then update by incrmeneting by 1

```
<hibernate-mapping>
    <class name="beans.BookMovie" table="tbc
    <id name="id">
        <generator class="assigned"/>
    </id>
    <property name=""></property>
    </class>
```

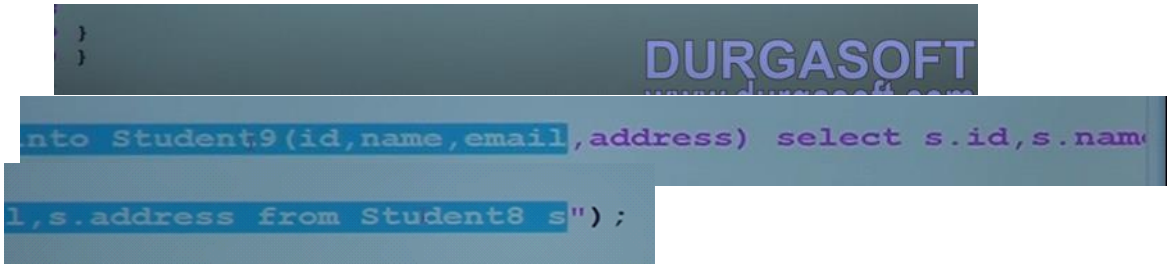


HQL-> Hiberante Query Language

1. We can shift one table data to other table
2. We cannot insert data to directly table

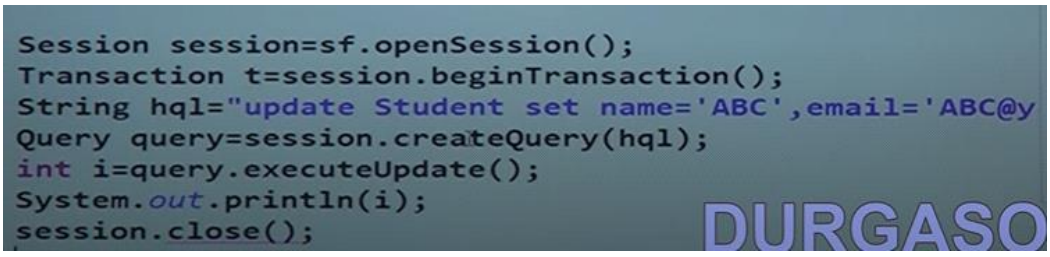


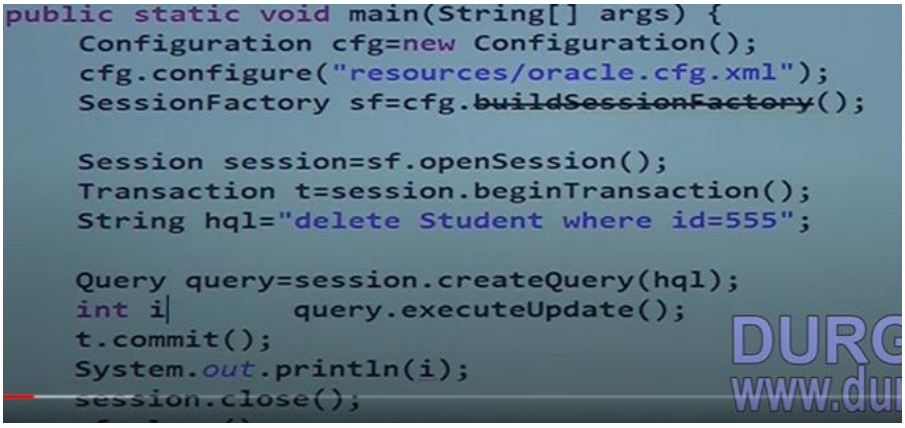

```
    Transaction t=s.beginTransaction();

    //insert

    //one table data we have to insert into another tab

Query q=s.createQuery("insert into Students(id,name,ema
int i=q.executeUpdate();
t.commit();
sf.close();
System.out.println("insert success");
}
}
```



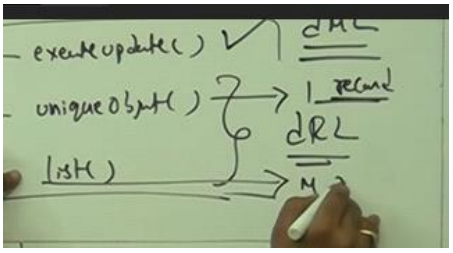
Int i=q.excuteUpdate();
I is how many rows affected by this query

Update Hql -> Resolve merge and update problem-> That earlier we have to update all value of coloumn and cannot upadate primary key

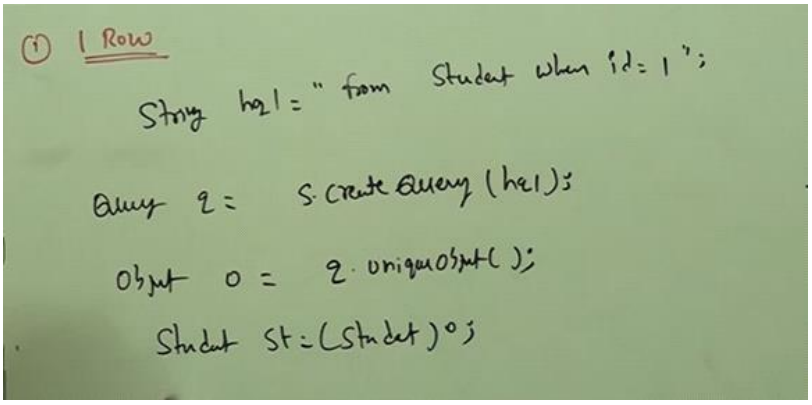




Select single row operation->

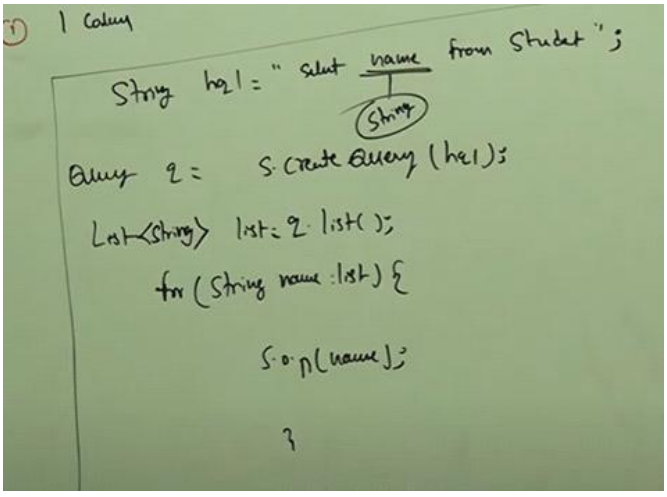


Multiple record= using list method
If only one record then we can go with unqiue record




```
Configuration cfg = new Configuration();
cfg.configure("resources/oracle.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="from Employee where id=111";

Query q=session.createQuery(hql);
Employee emp=(Employee) q.uniqueResult();
        System.out.println(emp.getId());
        System.out.println(emp.getName());
        System.out.println(emp.getEmail());
        System.out.println(emp.getSalary());
```

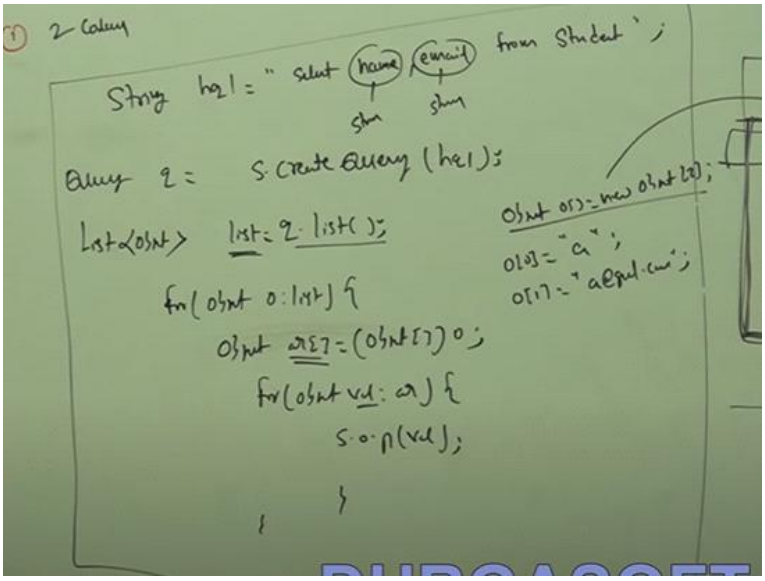
One single object of student then we will use unique object




```
Configuration cfg = new Configuration();
cfg.configure("resources/oracle.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="select name from Employee";
Query q=session.createQuery(hql);
List<String> list=  q.list();
for(String name:list)
{
    System.out.println(name);
}
session.close();
sf.close();
```

If we want to get 1 whole coloumn above one

If we want to get two column then it will convert to object of arrays




```
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="select name,email from Employee";
Query q=session.createQuery(hql);
List<Object> list=  q.list();
for(Object o:list)
{
    Object ar[]=(Object[])o;
    System.out.println(ar[0]);
    System.out.println(ar[1]);

}
session.close();
sf.close();
```

Whole object

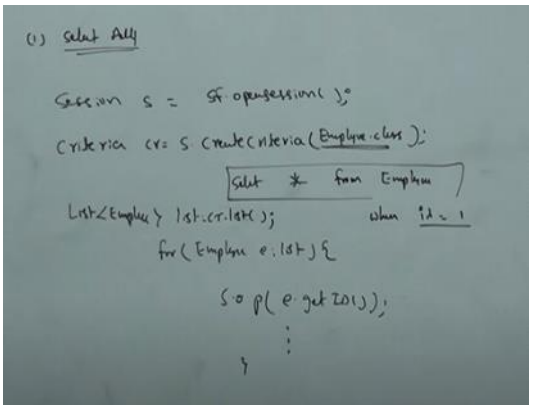
```
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="from Employee";
Query q=session.createQuery(hql);
List<Employee> list=    q.list();
for(Employee emp:list)
{
        System.out.println("ID:"+emp.getId());
        System.out.println("NAME:"+emp.getName());
        System.out.println("EMAIL:"+emp.getEmail());
        System.out.println("SALARY:"+
}
session.close();
```

Aggregate or sum

```
String hql="select avg(salary) from Employee";
Query q=session.createQuery(hql);
    double avg=(Double)q.uniqueResult();
System.out.println("salary avg="+avg);
session.close();
```

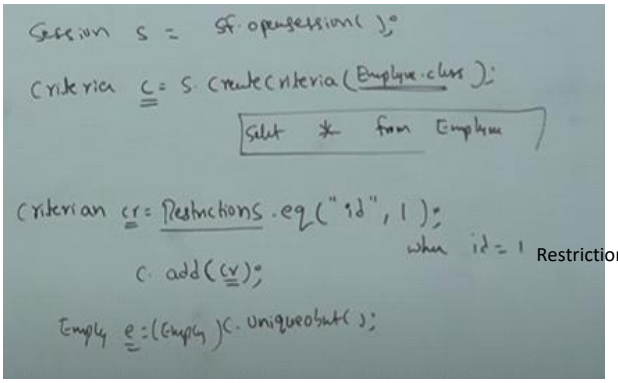
HQL we can use any CRUD operation

Criterias->**we can use only select operation with condition**

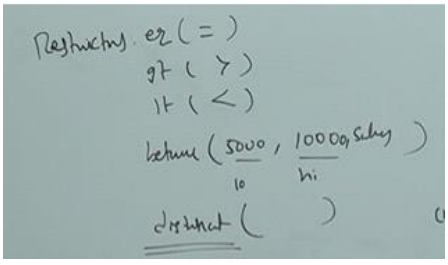


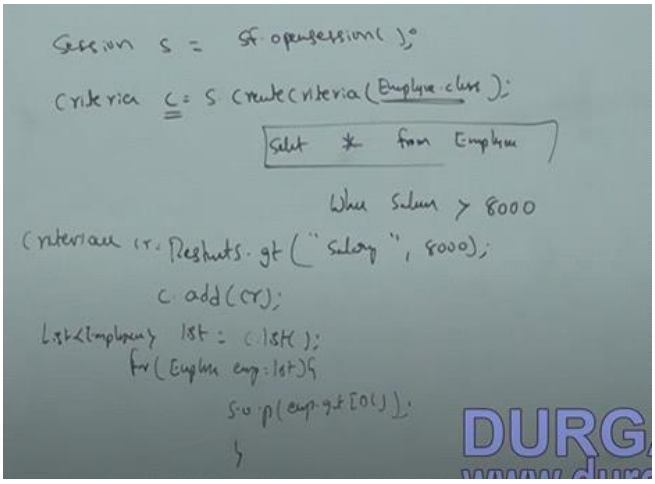
When we want to apply select * from employee where id=1; where condition then we will apply restriction

Crieterian means condition



Restriction means where Id=1



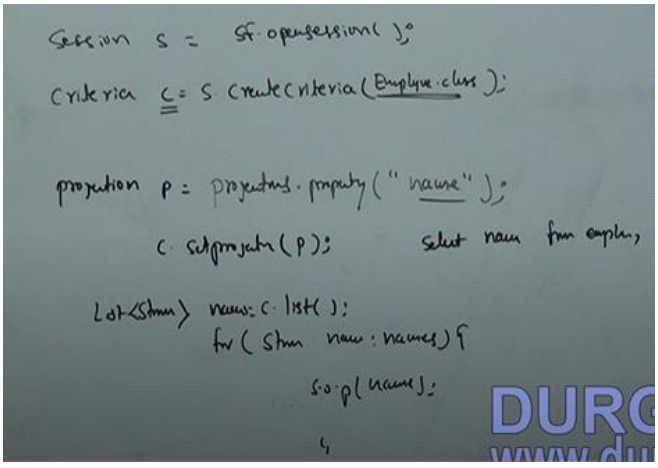


```
SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Criteria c= session.createCriteria(Employee.class);

        //where =
Criterion cr=Restrictions.eq("id", 111);
c.add(cr);
Employee emp=(Employee) c.uniqueResult();
System.out.println(emp.getName());
```

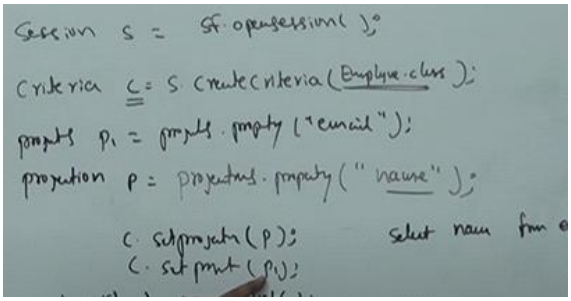
```
        //where =
//Criterion cr=Re
//where >
Criterion cr=Restrictions.gt("salary",70000);
c.add(cr);
    List<Employee>  emplist=c.list();
    for(Employee emp:emplist)
System.out.println(emp.getName()+";
```

Projections->
If we want only one column or max salary or aggregate one value we use projections api

Select name from employee only one coloumn ->Projection.porperty("name");
Partial record->

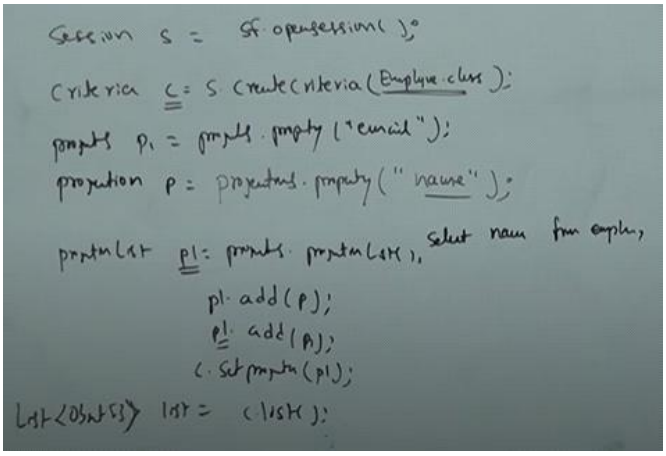


For one crietria we can apply multple criteria
But at a time only one projection

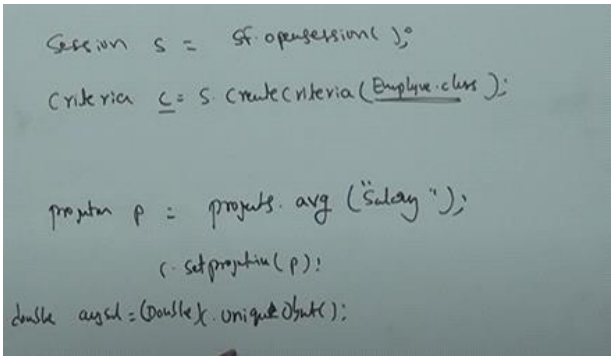


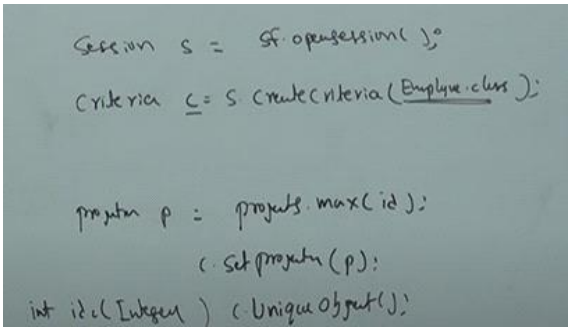
Above image is wrong as first projection will get overide by second one

If we want multiple column then we need to use projection list class



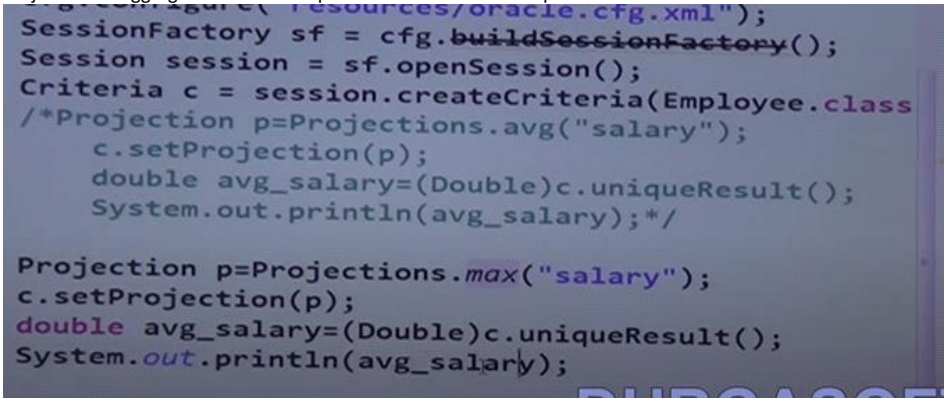
If we want to aggregate or do sum





Restriction are for condtion (=,>,<, between, like)

Projection for aggregate function and particular coloumn select operation



Cache support

One user
All user
One instance

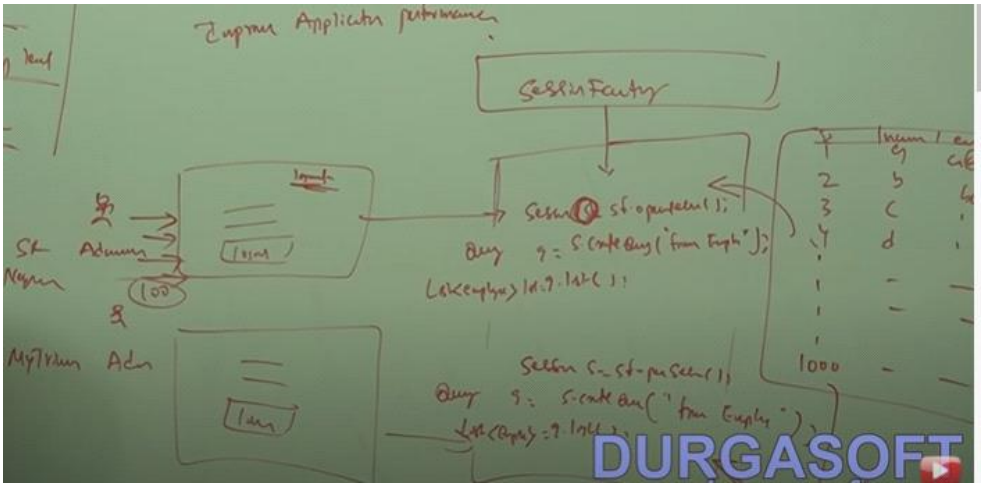
If Constant table if we want playerlist everytime

If constant it will everytime hit database..

View-> Application-> Database
If we want to see
Playerlist
It will hit everytime
To database
10000 hit call from application to database. Then we require I lakh connection and closed.. How much processing will waste

Cache will take temporary storage and it will return 1lakh time..
Cache reduce databse call. Improve application performance

Session cache->
It is useful for one user login to application.
Fetch information for data



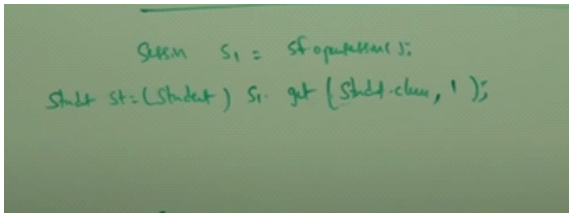
It will fetch only one time per session for one user
For remain 99 call they call from cache only from same session object

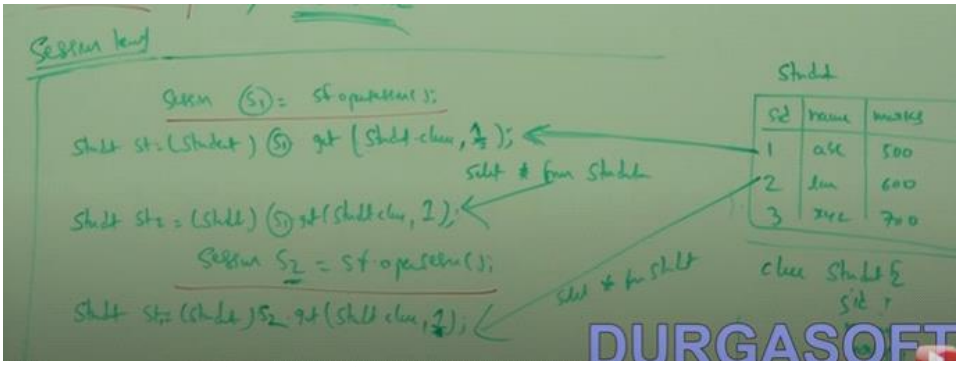
Different user has different session that it will cache
Example- If we login to gmail

1. First time it will retrive from database
2. But after that when we refresh it will return from session cache
3. Until logout only single call to database.
4. In session one time call datatabse

Session factory->
It is also called second level caching
If there are two user then one will hit database then it will avaible for second user also
As it is for session factory level

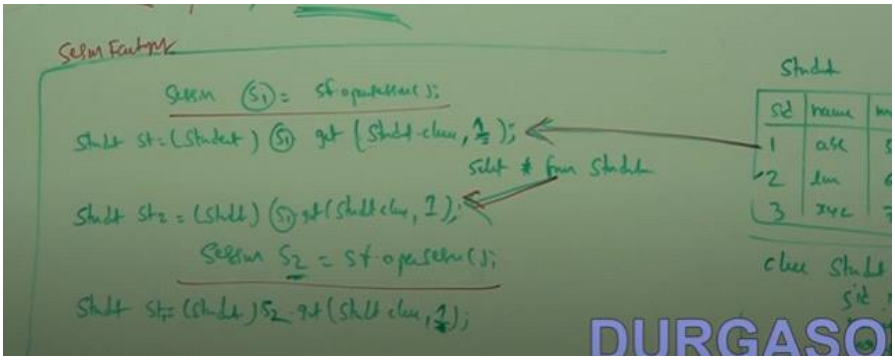
Query level-> Same type query same query like maximum salary.

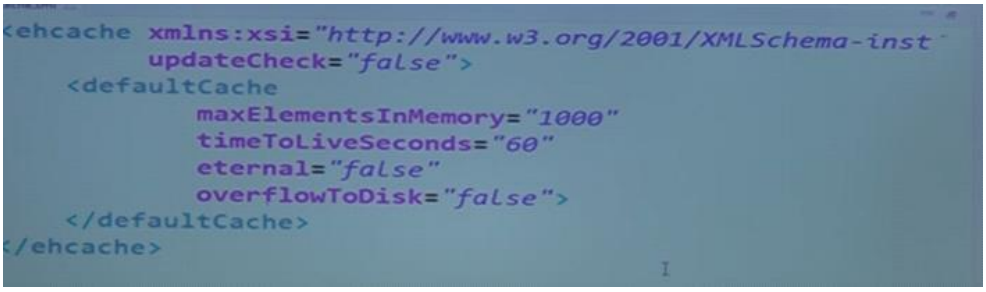




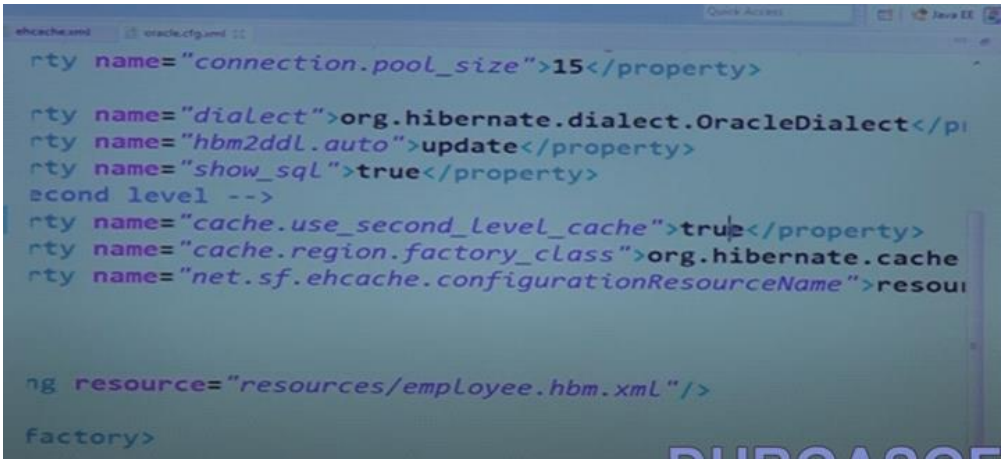
Two call for session cache for two session factory

But sesion factory cahce then same data availbale for both session factory



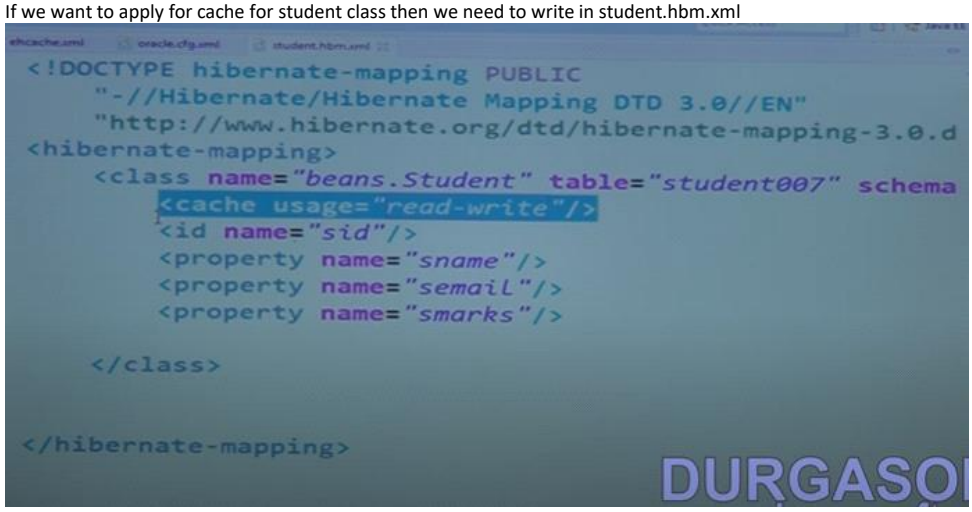


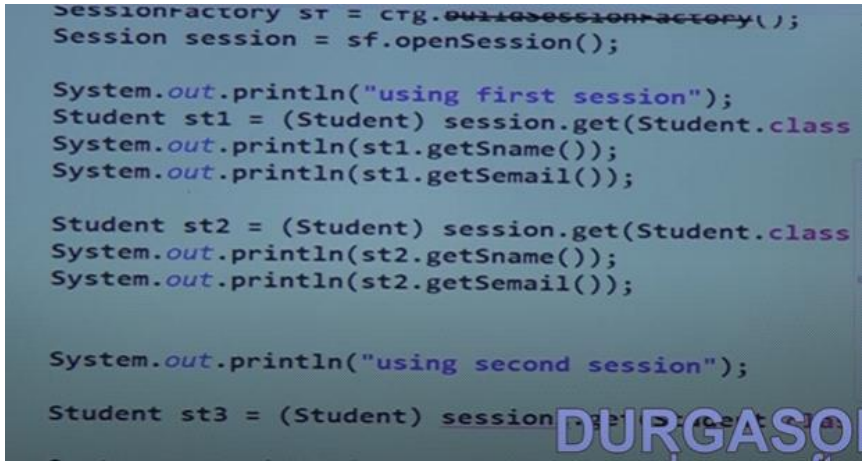
To add session level factory we need to add this
Maxelementinmemory-> Number of objects



We need to add this in configuration file

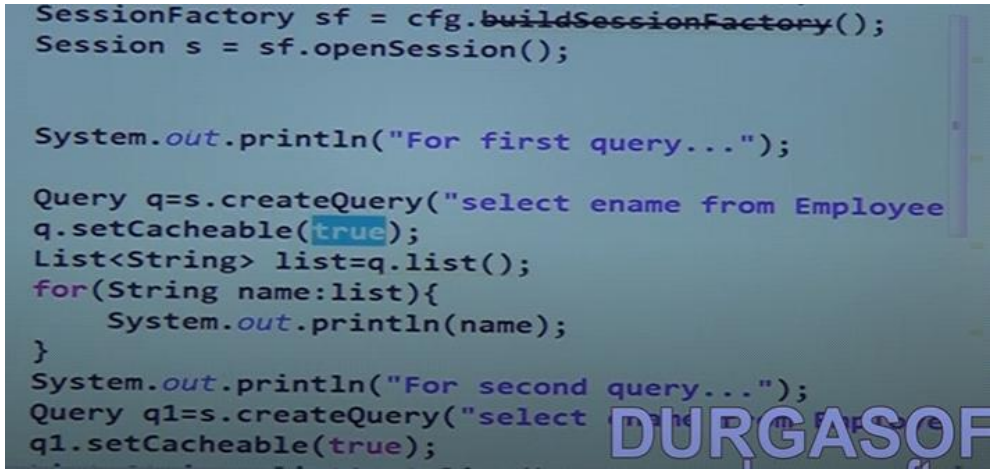
If we want to apply for cache for student class then we need to write in student.hbm.xml





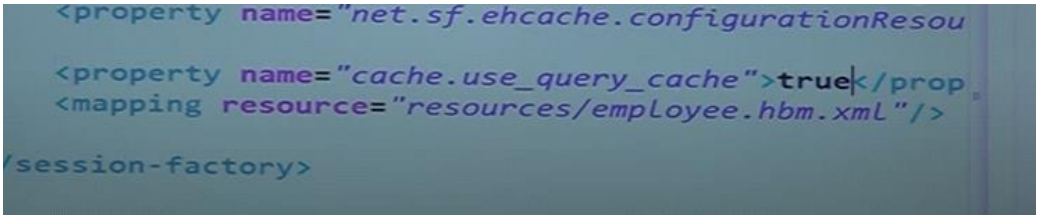
For this st1 and st2 it will fire single query as same session in session cache

But in session factory only one sleect operation



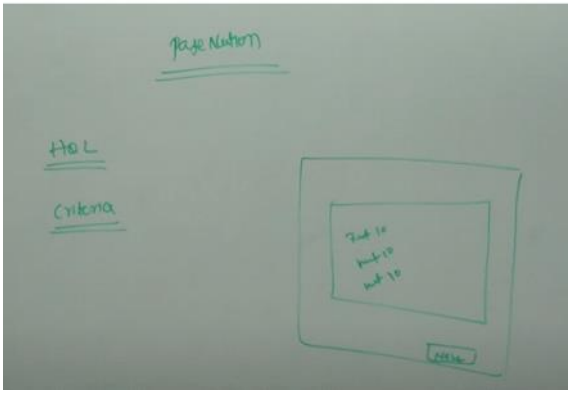
Query cache-> q.setcacheable(True)
It will store this query in cache

```
List<String> list=q.list();
for(String name:list){
    System.out.println(name);
}

System.out.println("For second query...");
Query q1=s.createQuery("select ename from Employee");
q1.setCacheable(true);

List<String> list1=q1.list();
for(String name:list1){
    System.out.println(name);
}
System.out.println("for third querys..");

Query q2=s.createQuery("select ename from Employee");
q2.setCacheable(true);
List<String> list2=q2.list();
```



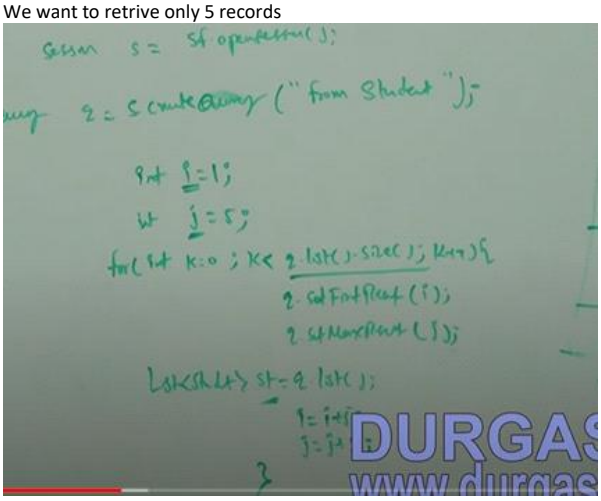

```
e1.setEid(111);
e1.setEname("abc");
e1.setEmail("abc@gmail.com");
e1.setSalary(500000);

Employee e2=new Employee();
e2.setEid(222);
e2.setEname("lmn");
e2.setEmail("lmn@gmail.com");
e2.setSalary(600000);

s.save(e1);
s.save(e2);
s.beginTransaction().commit();
System.out.println("success");
```

Pagination

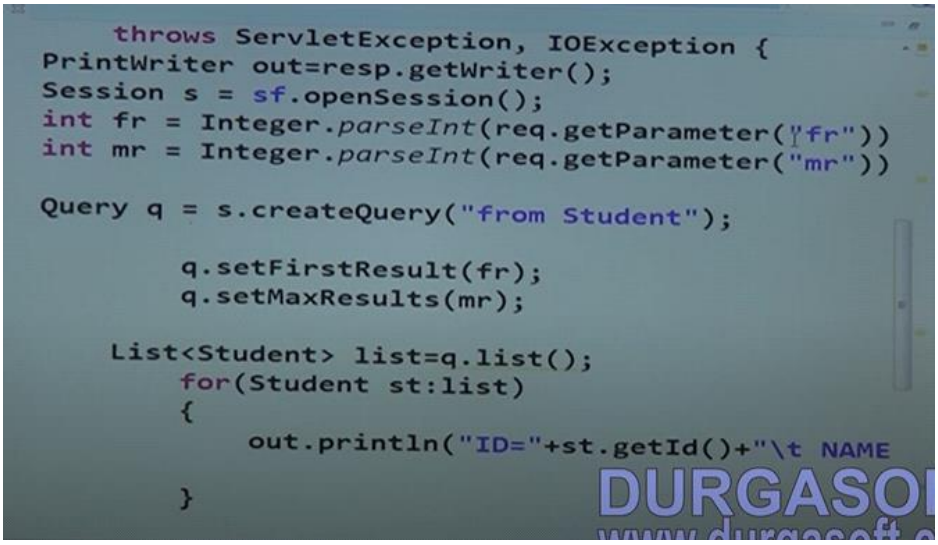


If we want to retrive limited value

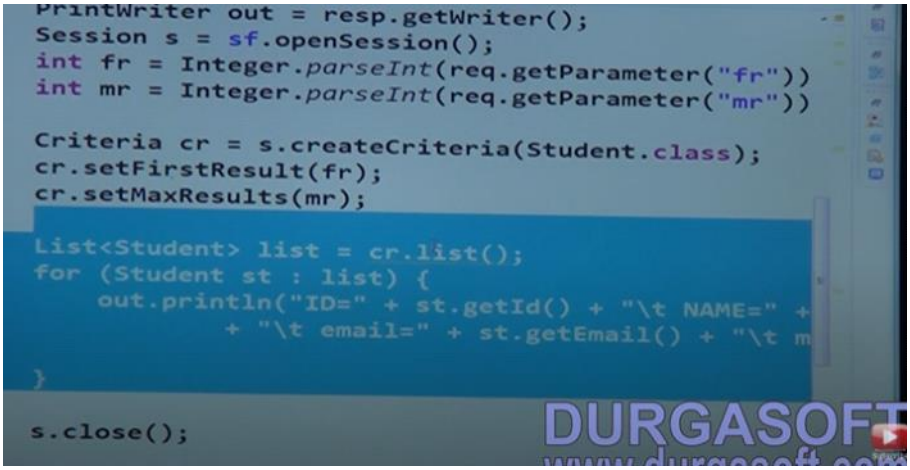
We want to retrive only 5 records



q.setfirstsize(0)
q.setmaxresult(5)



Criteria pagination



@Transaction is session management

```
timeToIdleSeconds="300"
timeToLiveSeconds="600">
```

If it has not been used for 300 seconds
Or after stays in cache for 600 seconds
When echache exceed from its configured size ehcache removed expired elements
If that doesn't clear enough space it cleasr object from ehcache
By default it remove LRU elements

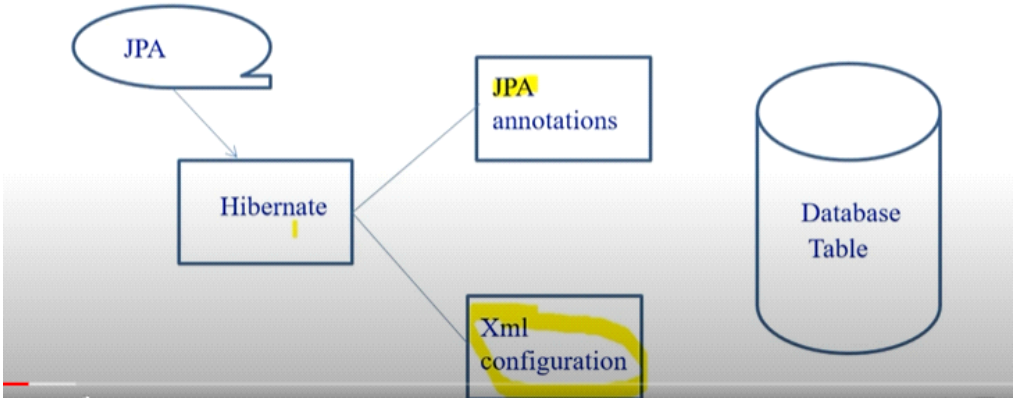
Second level caching is we put on entity or whole object

Default cache is made

```
@Configuration
@EnableCaching
public class CacheConfig {

    @Bean
    public CacheManager cacheManager() {
        SimpleCacheManager cacheManager = new SimpleCacheManager();
        List<Cache> cacheList = new ArrayList<Cache>();
        cacheList.add(new ConcurrentMapCache("userCache"));
        cacheList.add(new ConcurrentMapCache("addressCache"));
        cacheManager.setCaches(cacheList);
        return cacheManager;
    }
}
```

Nosql->
We call collection to table

- **How** ?
- It provides JPA implementation hence we can use JPA annotations as well as xml configurations to achieve this mapping.



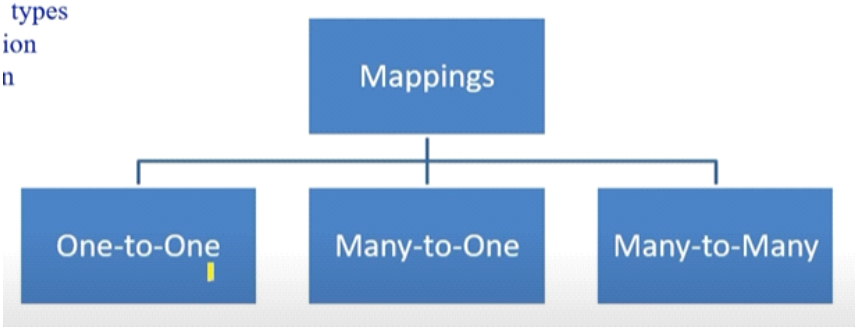
- **Why Hibernate ?**
- Hibernate eliminates all the boiler-plate code that comes with JDBC.
- It supports HQL with is more Object oriented.
- It provides transaction management implicitly.
- Hibernate throws JDBCException or HibernateException which are the unchecked exceptions, so we don't need to worry about handling using try and catch.
- Hibernate supports caching for better performance.

Q) Important Interfaces used in Hibernate

- **SessionFactory (org.hibernate.SessionFactory)** – Instance of this is used to retrieve Session objects for database operations. We need to initialize that once and can cache it to reuse it again and again. Its like one SessionFactory object per database connection. Like 1 for mysql, 1 for oracle.
- **Session (org.hibernate.Session)** – Its factory for transaction, it's used for connecting application with persistant store like hibernate framework / DB. It is used to get a physical connection with the database. It also provides methods for CRUD operations.
- **Transaction (org.hibernate.Transaction).** – This specifies single / atomic units of work

```
SessionFactory factory = metadata.getSessionFactoryBuilder().build();
Session session = factory.openSession();
Transaction t = session.beginTransaction();

session.save(persistantObj);
    t.commit();
factory.close();
    session.close();
```

types
ion
n



Many to Many

```
class Student{

@ManyToMany(targetEntity = Degree.class, cascade = { CascadeType.ALL })
@JoinTable(name = "DegreeStudentThirdTable",
           joinColumns = { @JoinColumn(name = "StudentId") },
           inverseJoinColumns = { @JoinColumn(name = "CertificateId") })
private List<Degree> degrees;

}
```

Q) What are hibernate configuration file

contains database specific configurations and used to initialize SessionFactory.

Conventionally, its name should be hibernate.cfg.xml

If u need to connect to SQL then create another one here.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">codedecode</property>
        <property name="connection.password">codedecode</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <mapping resource="employee.hbm.xml"/>
    </session-factory>

</hibernate-configuration>
```

Q) What are hibernate mapping file

The mapping file name conventionally, should be class_name.hbm.xml.

hibernate-mapping : root element
class : specifies the Persistent class.
id : specifies the primary key attribute in the class.
generator : used to generate the primary key.
property : specifies the property name of the Persistent class.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd

 <hibernate-mapping>
  <class name="com.codedecode.Employee" table="empTable">
    <id name="id">
     <generator class="assigned"></generator>
    </id>

    <property name="firstName"></property>
    <property name="lastName"></property>

  </class>

</hibernate-mapping>
```

Q) difference between openSession and getCurrentSession

getCurrentSession() method returns the session bound to the context.
Since this session object belongs to the context of Hibernate, it is okay if you don't close it. Once the SessionFactory is closed, this session object gets closed.

While

openSession() method helps in opening a new session.
`You should close this session object once you are done with all the database operations. And also, you should open a new session for each request in a multi-threaded environment.

- get() loads the data as soon as it's called whereas load() returns a proxy object and loads data only when it's actually required, **so load() is better because it support lazy loading.**
- **Since load() throws exception** when data is not found, we should use it only when we know data exists.
- We should use **get() when we want to make sure data exists** in the database.

Q) hibernate caching – Second level cache

- Hibernate Second Level cache is disabled by default but we can enable it through configuration.
- Currently EHCache and Infinispan provides implementation for Hibernate Second level cache and we can use them.

Sunday, December 27, 2020 3:31 AM

Single design pattern

Singleton Design pattern - Properties

- Creational design pattern
- Only one instance of the class should exist
- Other classes should be able to get Instance of Singleton class
- Used in Logging, Cache, Session, Drivers

Singleton Design pattern - Implementation

- Constructor should be private
- Public method for returning instance
- Instance type – private static

Initialisation Type:

- Eager Initialisation
- Lazy Initialisation
- Thread safe Method Initialisation
- Thread safe block Initialisation

```
    private Singlepatt() {

    }

    static Singlepatt getinsta() {

        if (single == null) {
            single = new Singlepatt();
            return single;
        } else {
            return single;
        }
    }
}

class Single {
    static Single single;

    private Single() {

    }

    static synchronized Single getinstance() {
        if (single == null) {
            single = new Single();
            return single;
        } else {
            return single;
        }
    }
}
```

Builder Design pattern - Properties

- Creational design pattern
- Used when we have too many arguments to send in Constructor & it's hard to maintain the order.
- When we don't want to send all parameters in Object initialisation

(Generally we send optional parameters as Null)

Used when we have too many arguments to send in constructor & it's hard to maintain order

```
package builder;

class Vehicle {
  //required parameter
  private String engine;
  private int wheel;

  //optional parameter
  private int airbags;

  public String getEngine() {
    return this.engine;
  }

  public int getWheel() {
    return this.wheel;
  }

  public int getAirbags() {
    return this.airbags;
  }

  private Vehicle(VehicleBuilder builder) {
    this.engine = builder.engine;
    this.wheel = builder.wheel;
    this.airbags = builder.airbags;
  }

  public static class VehicleBuilder {
    private String engine;
    private int wheel;

    private int airbags;

    public VehicleBuilder(String engine, int wheel) {
      this.engine = engine;
      this.wheel = wheel;
    }
```

```
ublic static class VehicleBuilder {
  private String engine;
  private int wheel;

  private int airbags;

  public VehicleBuilder(String engine, int wheel) {
    this.engine = engine;
    this.wheel = wheel;
  }

  public VehicleBuilder setAirbags(int airbags) {
    this.airbags = airbags;
    return this;
  }

  public Vehicle build() {
    return new Vehicle(this);
  }


lic class BuilderPatternExample {

ublic static void main(String[] args) {
  Vehicle car = new Vehicle.VehicleBuilder("1500cc", 4).setAirbags(4).build();
  Vehicle bike = new Vehicle.VehicleBuilder("250cc", 2).build();

  System.out.println(car.getEngine());
  System.out.println(car.getWheel());
  System.out.println(car.getAirbags());

  System.out.println(bike.getEngine());
  System.out.println(bike.getWheel());
  System.out.println(bike.getAirbags());
```

ConcurrentHashMap->

- Reads can happen very fast while write is done with a lock.
- You should use ConcurrentHashMap when you need very high concurrency in your project.

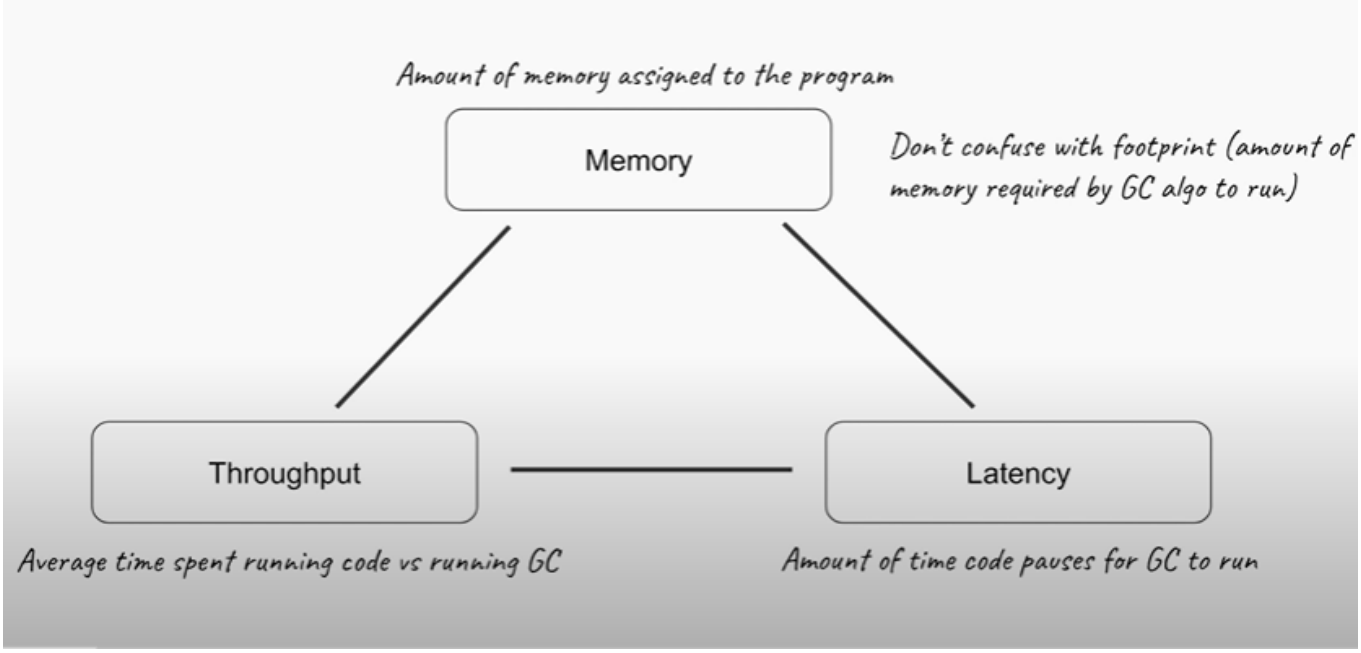
- It is thread safe without synchronizing the whole map.
- Reads can happen very fast while write is done with a lock.
- There is no locking at the object level.

>

SynchronizedHashMap->

- Every read/write operation needs to acquire lock.
- Locking the entire collection is a performance overhead.
- This essentially gives access to only one thread to the entire map & blocks all the other threads.

From <https://crunchify.com/hashmap-vs-concurrenthashmap-vs-synchronizedmap-how-a-hashmap-can-be-synchronized-in-java/>



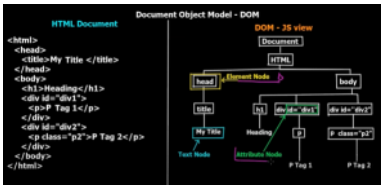
Garbage collection in java is an automatic process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.

An in use object, or a referenced object, means that some part of your program still maintains a pointer to that object.

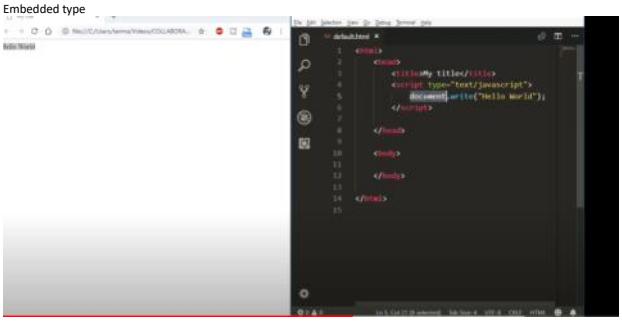
An unused object, or unreferenced object, is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed.

Main **Advantage** of automatic garbage collection in java is that it removes the burden of manual memory allocation/deallocation from us so that we can focus on problem solving.

Java script



Embedded type



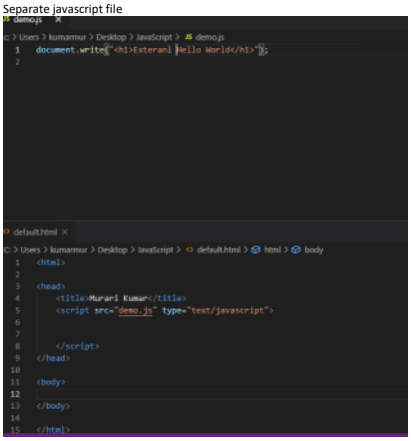
Document.write("Hello world")

```
document.write("<h1>Hello World</h1>");
```

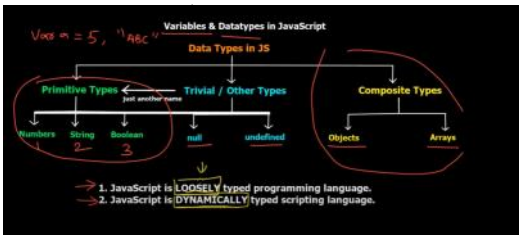
It represent entire document module in dom

Write is method which is directly writes on browser

Separate javascript file



This is statement and instruction to print hello world on screen




```
<script type="text/javascript">

   // int x = 5;

   // JS is LOOSELY typed.
   // JS is DYNAMICALLY TYPED

   var num = 16;  // Number
   var Name = "Tanmay Sakpal";   // String
   var flag = true; // boolean


   document.write(num); // printing on the browser
</script>
</head>
```

```
// JS dynamically typed

var num = 16;  // Number
var Name = "Tanmay Sakpal";   // String
var flag = false; // boolean
num = "Tanmay";
```

```
   Booleans can be objects (if defined with the new
   keyword)
   Numbers can be objects (if defined with the new
   keyword)
   Strings can be objects (if defined with the new
   keyword)
   Dates are always objects
   Maths are always objects
   Regular expressions are always objects
   */
   var str1 = new String();

   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   document.write(typeof(str1));
   //Car.teslaAutoPilot();
```

```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";
```

If we want to add mor property of car

```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";

   Car.newFn = function()
   {
      document.write("<h2>Added function</h2>");
   }
```

Adding metho

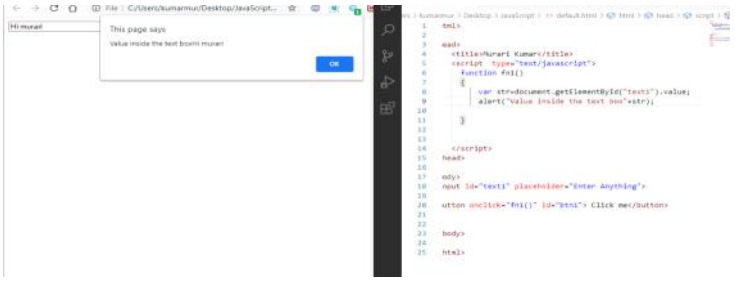
```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";
   delete Car.price;

   document.write("<h2>"+Car.price+"</h2>");
   //Car.teslaAutoPilot();
```

Delete property

Button clicked action

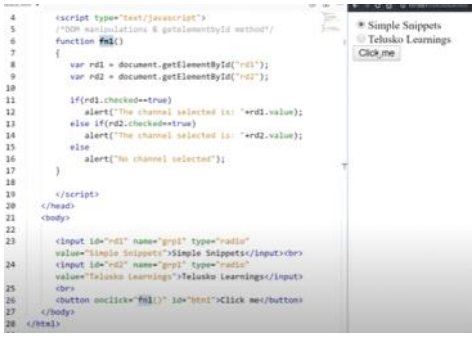
```
<head>
    <title>Murari Kumar</title>
    <script  type="text/javascript">

    function buttonclick()
    {
        alert("Button Clicked");
    }

    </script>
</head>

<body>
    <button onclick="buttonclick()">Click me</button>
    <h1 id="first">Can you please click on button</h1>
</body>

</html>
```

Input and give alter box



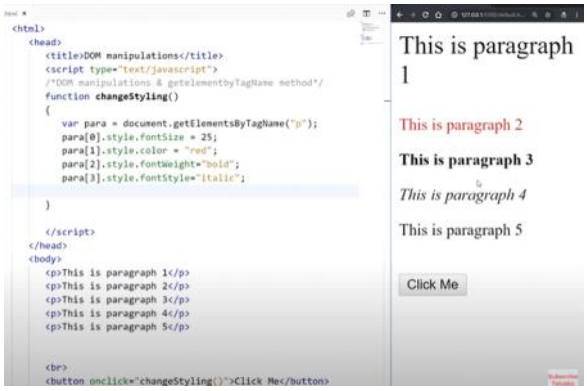
Username & password

Radio group name for both then we can select any one
Other wise we both will be slected if we give different name

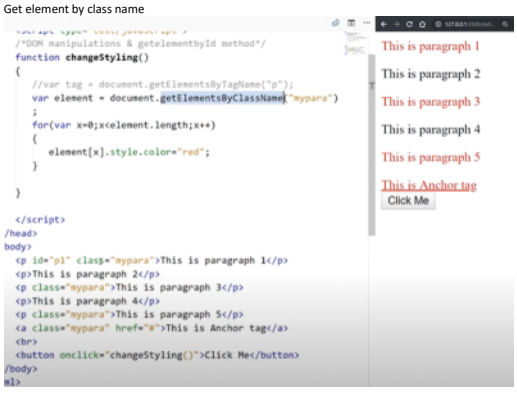


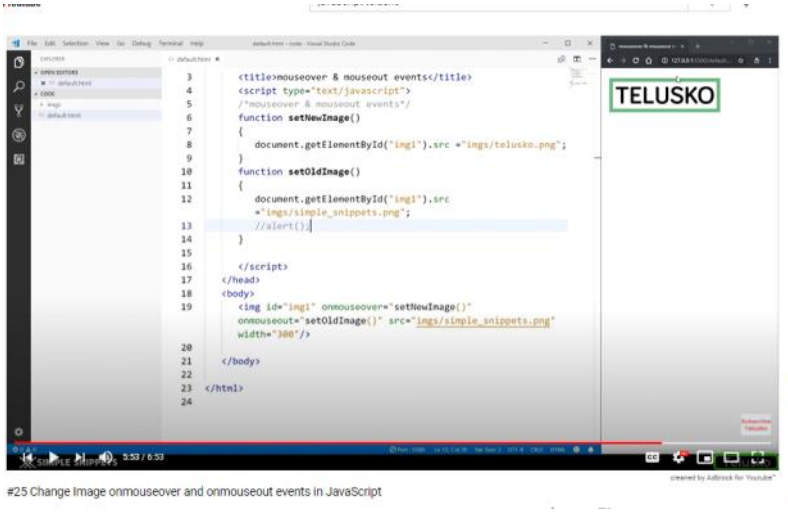

```
<html>

<head>
    <title>Murari Kumar</title>
    <script  type="text/javascript">
    function fn1()
    {
        var r1=document.getElementById("select");
        alert("Pop up  "+r1.value);
    }


    </script>
</head>

<body>

    <select id="select">
<option value="Murari">Murari</option>
<option value="Kumar">Kumar</option>
<option value="Rahul">Rahul</option>
    </select>
<button onclick="fn1()" id="btn1">On click</button>


</body>

</html>
```

Get element by tag name



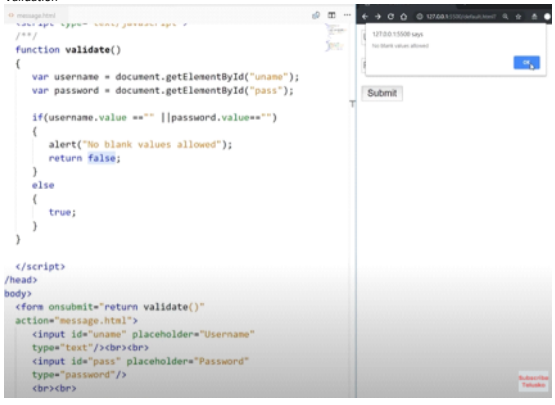
Get element by class name



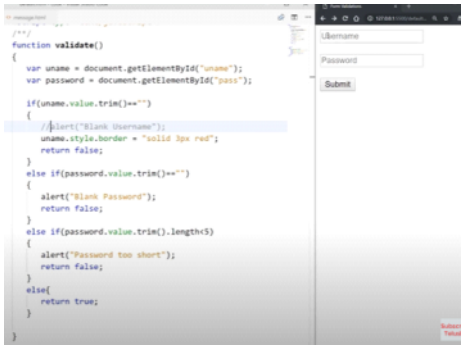


#25 Change Image onmouseover and onmouseout events in JavaScript

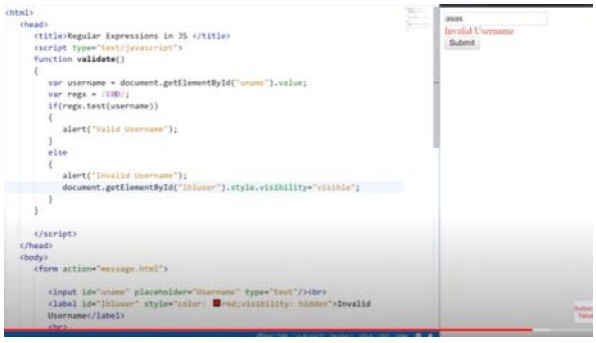
Validation



On submit extra fetaure



```
/**/
function validate()
{
   var uname = document.getElementById("uname");
   var password = document.getElementById("pass");

   if(uname.value.trim()=="")
   {
      //alert("Blank Username");
      uname.style.border = "solid 3px red";
      return false;
   }
   else if(password.value.trim()=="")
   {
      alert("Blank Password");
      return false;
   }
   else if(password.value.trim().length<5)
   {
      alert("Password too short");
      return false;
   }
   else{
      return true;
   }
}
```



```
<html>
   <head>
      <title>Regular Expressions in JS </title>
      <script type="text/javascript">
      function validate()
      {
         var username = document.getElementById("uname").value;
         var regx = /[0-9]/;
         if(regx.test(username))
         {
            alert("Valid Username");
         }
         else
         {
            alert("Invalid Username");
            document.getElementById("lbluser").style.visibility="visible";
         }
      }

      </script>
   </head>
   <body>
      <form action="message.html">

         <input id="uname" placeholder="Username" type="text"/><br>
         <label id="lbluser" style="color: red;visibility: hidden">Invalid
         Username</label>
         <br>
```


JAVA 8

Thursday, January 7, 2021 2:14 AM

```
public class Sample {
  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    //external iterators
    // for(int i = 0; i < numbers.size(); i++) {
    //    System.out.println(numbers.get(i));
    // }

    //external iterators also
    // for(int e : numbers) {
    //    System.out.println(e);
    // }
```

```
  //internal iterators
  // numbers.forEach(new Consumer<Integer>() {
  //    public void accept(Integer value) {
  //       System.out.println(value);
  //    }
  // });


  numbers.forEach(new Consumer<Integer>() {
    public void accept(Integer value) {
      System.out.println(value);
    }
  });
}
```

Lambada be only one line

here are a few smells in this code. Let's discuss each one of them:

1. **The code is hard to read**
   We can't quickly grasp what's going on in this code, it's not conc
   have to keep multiple context in mind, what `e` is and what `i` is a
   we are comparing the two. We lost the general benefits of lamb
   concise and expressive—this code fails both.
2. **It's noisy**
   Writing a single line lambda was nice. The minute we wanted to
   than one line, we have to bring the `{}`, the `;`, and possibly the r
   we had something to return. That's Java's (and C#'s too) way of
   us—it's telling us not to do that.
3. **Leads to Duplication**
   Whatever that multiline code does, it's not reusable. If we wante

Method refrnce-> If we simply get the value and don't want to alter then we can use this

```
// numbers.forEach(e -> System.out.println(e));
// numbers.forEach(System.out::println);

numbers.stream()
       // .map(e -> String.valueOf(e))
       .map(String::valueOf)
       .forEach(System.out::println);
```

```
numbers.stream()
       .map(e -> String.valueOf(e))
       // .map(e -> e.toString())
       .map(String::toString)
       .forEach(System.out::println);
```

```
// numbers.stream()
//         .map(e -> String.valueOf(e))
//         // .map(e -> e.toString())
//         .map(String::toString)
//         .forEach(System.out::println);

System.out.println(
  numbers.stream()
         // .reduce(0, (total, e) -> Integer.sum(total, e)));
         .reduce(0, Integer::sum));
```

If we are not altering just pass method refrnce

```
// System.out.println(
//   numbers.stream()
//          // .reduce(0, (total, e) -> Integer.sum(total, e)));
//          .reduce(0, Integer::sum));

System.out.println(
  numbers.stream()
         .map(String::valueOf)
         // .reduce("", (carry, str) -> carry.concat(str)));
         .reduce("", String::concat));
```

Mobile phone finding big room. More person find resources accessed more

```
    numbers.stream()
           .filter(e -> e % 2 == 0)
           .map(e -> e * 2)
           .forEach(e -> doubleOfEven.add(e));
    //mutability is OK, sharing is nice, shared mutability is devils work

    //friends don't let friends do shared mutation.

    System.out.println(doubleOfEven); //Don't do this.

    List<Integer> doubleOfEven2 =
      numbers.stream()
             .filter(e -> e % 2 == 0)
             .map(e -> e * 2)
             .collect(toList());
    System.out.println();
  }
```

Map->

```
    new Person("Jack", Gender.MALE, 72),
    new Person("Jill", Gender.FEMALE, 12)
  );
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
          )));
}
```

First object passed toMap is key value
Second object passed to get object or value

```
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
            person -> person)));
}
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the people with that name.

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName)));
```

```
-[Bob -- MALE -- 20], Sara=[Sara -- FEMALE -- 20, Sara -- FEMALE -- 22],
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the ages of people with that
ame

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName,
              mapping(Person::getAge, toList()))));
}
```

Get a Taste of Lambdas and Get Addicted to Streams by Venkat Subramaniam

what is completeblfuture

Account.java

```java
package threadingexample;

public class Account {
    private int balance=20000;
    public int getbalance()
    {
        return balance;
    }
    public void withdraw(int val)
    {
        this.balance=this.balance-val;
    }
}
```

*AccountHolder.java

```java
    @Override
    public void run() {
        for (int i = 1; i < 5; i++) {
            makewithdrawble(2000);
            if (account.getbalance() < 0) {
                System.out.println("Already taken........." + Thread.
            }
        }

    }

    private synchronized  void makewithdrawble(int i) { // protect mu
        if (account.getbalance() >= i) {
            System.out.println(
                    Thread.currentThread().getName() + "    " + "Curre
            try {
```

Client.java

```java
package threadingexample;

public class Client {
    public static void main(String[] args) {
        Account account= new Account();
        AccountHolder holder= new AccountHolder(account);
        Thread t1= new Thread(holder);
        Thread t2= new Thread(holder);
        t1.setName("Murari");
        t2.setName("kumar");
        t1.start();
        t2.start();
    }


}
```

If not synhchronized all thread access at same time. If synchronized it will lock thread

https://www.mockaroo.com/ To mock data

JAVA 8

Thursday, January 7, 2021 2:14 AM

```
public class Sample {
  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    //external iterators
    // for(int i = 0; i < numbers.size(); i++) {
    //   System.out.println(numbers.get(i));
    // }

    //external iterators also
    // for(int e : numbers) {
    //   System.out.println(e);
    // }
```

```
  //internal iterators
  // numbers.forEach(new Consumer<Integer>() {
  //   public void accept(Integer value) {
  //     System.out.println(value);
  //   }
  // });


  numbers.forEach(new Consumer<Integer>() {
    public void accept(Integer value) {
      System.out.println(value);
    }
  });
}
```

Lambada be only one line

here are a few smells in this code. Let's discuss each one of them:

1. **The code is hard to read**
   We can't quickly grasp what's going on in this code, it's not conc
   have to keep multiple context in mind, what `e` is and what `i` is a
   we are comparing the two. We lost the general benefits of lamb
   concise and expressive—this code fails both.
2. **It's noisy**
   Writing a single line lambda was nice. The minute we wanted to
   than one line, we have to bring the `{}`, the `;`, and possibly the `r`
   we had something to return. That's Java's (and C#'s too) way of
   us—it's telling us not to do that.
3. **Leads to Duplication**
   Whatever that multiline code does, it's not reusable. If we wante

Method refrnce-> If we simply get the value and don't want to alter then we can use this

```
// numbers.forEach(e -> System.out.println(e));
// numbers.forEach(System.out::println);

numbers.stream()
       // .map(e -> String.valueOf(e))
       .map(String::valueOf)
       .forEach(System.out::println);
```

```
numbers.stream()
       .map(e -> String.valueOf(e))
       // .map(e -> e.toString())
       .map(String::toString)
       .forEach(System.out::println);
```

```
// numbers.stream()
//         .map(e -> String.valueOf(e))
//         // .map(e -> e.toString())
//         .map(String::toString)
//         .forEach(System.out::println);

System.out.println(
  numbers.stream()
         // .reduce(0, (total, e) -> Integer.sum(total, e)));
         .reduce(0, Integer::sum));
```

If we are not altering just pass method refrnce

```
// System.out.println(
//   numbers.stream()
//          // .reduce(0, (total, e) -> Integer.sum(total, e)));
//          .reduce(0, Integer::sum));

System.out.println(
  numbers.stream()
         .map(String::valueOf)
         // .reduce("", (carry, str) -> carry.concat(str)));
         .reduce("", String::concat));
```

Mobile phone finding big room. More person find resources accessed more

```
    numbers.stream()
           .filter(e -> e % 2 == 0)
           .map(e -> e * 2)
           .forEach(e -> doubleOfEven.add(e));
    //mutability is OK, sharing is nice, shared mutability is devils work

    //friends don't let friends do shared mutation.

    System.out.println(doubleOfEven); //Don't do this.

    List<Integer> doubleOfEven2 =
      numbers.stream()
             .filter(e -> e % 2 == 0)
             .map(e -> e * 2)
             .collect(toList());
    System.out.println();
  }
```

Map->

```
    new Person("Jack", Gender.MALE, 72),
    new Person("Jill", Gender.FEMALE, 12)
  );
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
          )));
}
```

First object passed toMap is key value
Second object passed to get object or value

```
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
            person -> person)));
}
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the people with that name.

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName)));
```

```
-[Bob -- MALE -- 20], Sara=[Sara -- FEMALE -- 20, Sara -- FEMALE -- 22],
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the ages of people with that
ame

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName,
              mapping(Person::getAge, toList()))));
}
```

Get a Taste of Lambdas and Get Addicted to Streams by Venkat Subramaniam

what is completeblfuture

Account.java

```java
package threadingexample;

public class Account {
    private int balance=20000;
    public int getbalance()
    {
        return balance;
    }
    public void withdraw(int val)
    {
        this.balance=this.balance-val;
    }
}
```

*AccountHolder.java

```java
    @Override
    public void run() {
        for (int i = 1; i < 5; i++) {
            makewithdrawble(2000);
            if (account.getbalance() < 0) {
                System.out.println("Already taken........." + Thread.
            }
        }

    }

    private synchronized  void makewithdrawble(int i) { // protect mu
        if (account.getbalance() >= i) {
            System.out.println(
                    Thread.currentThread().getName() + "    " + "Curre
            try {
```

Client.java

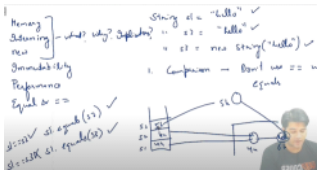
```java
package threadingexample;

public class Client {
    public static void main(String[] args) {
        Account account= new Account();
        AccountHolder holder= new AccountHolder(account);
        Thread t1= new Thread(holder);
        Thread t2= new Thread(holder);
        t1.setName("Murari");
        t2.setName("kumar");
        t1.start();
        t2.start();
    }


}
```

If not synhchronized all thread access at same time. If synchronized it will lock thread

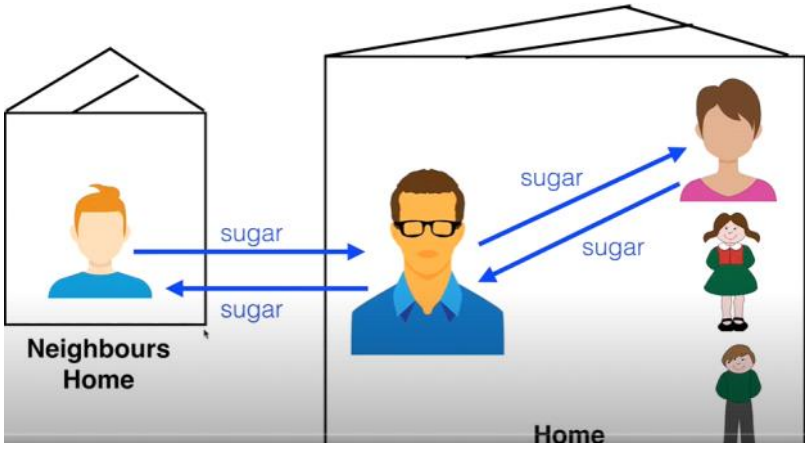
https://www.mockaroo.com/ To mock data

Friday, December 18, 2020 8:04 PM

Spring MVC

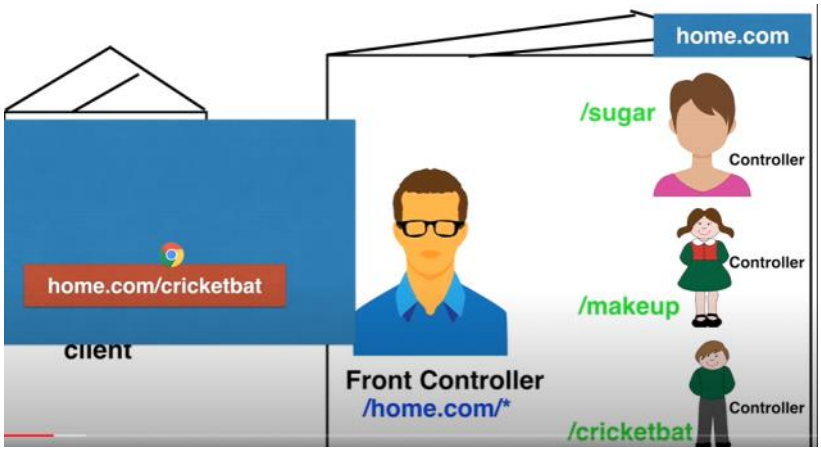


Front CONTROLLER



Any request come outside the home dad knows the avaliable resource so where to dispatch that incoming request

Every request come it is handle by the front controller. He will accept all the request which has url starting with home.com
Routing to other controller




```
<servlet>
    <servlet-name>dad-frontcontroller-dispatcher</servlet-name>
    <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>dad-frontcontroller-dispatcher</servlet-name>
    <url-pattern>/home.com/*</url-pattern>
</servlet-mapping>
```

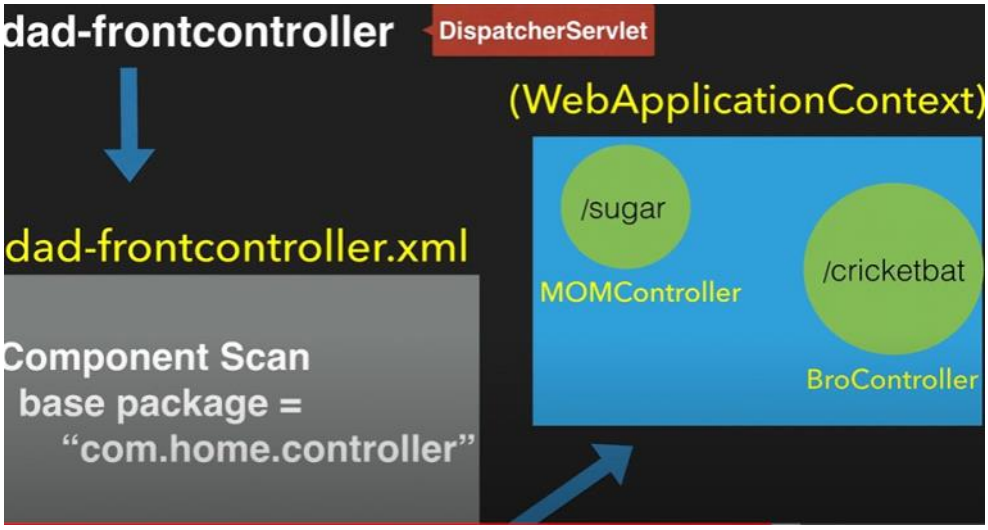
Dispatcher front controller server first point of hitting


```
@Controller
public class MomController {

    @ResponseBody
    @RequestMapping("/sugar")
    public String giveSugar()
    {
        return "Ok..Here is your sugar";
    }

}
```

@Response body map the value to HTTP

To print on webpage we need to use @Response Body

After intilization framework will look for servlet name dad-controller with appeneded with .xml

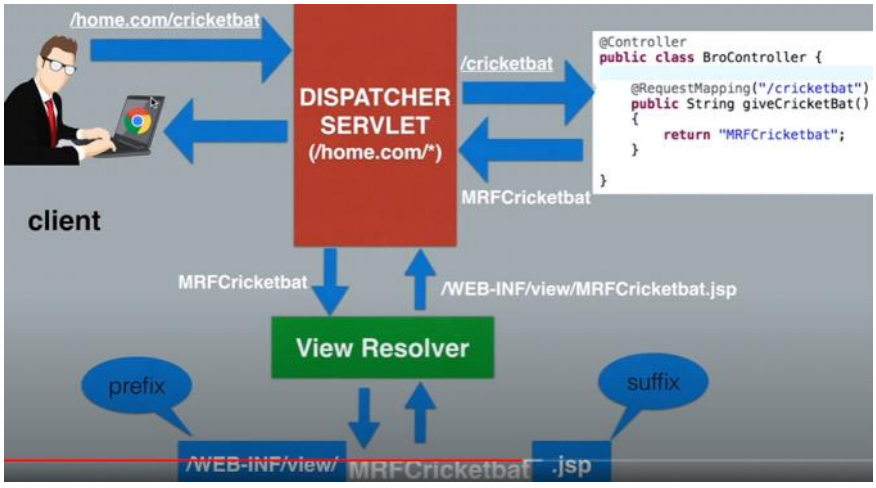



```java
@Controller
public class BroController {

    @ResponseBody
    @RequestMapping("/cricketbat")
    public String giveCricketBat()
    {
        return "MRFCricketbat";
    }

}
```

If we write responseBody annotation then that will write on the web page that string

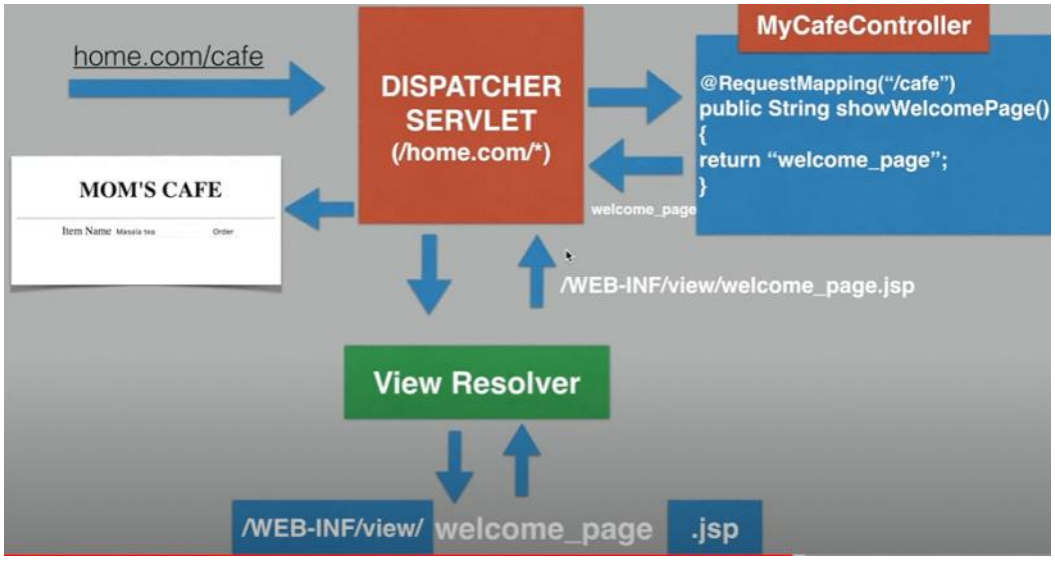
But if we remove the @Response body then contoller will expect a proper web page of name called MRFCricketBat name web page
It is called view name

View resolver--> webinf will change
.jsp will also change
But page name will not change



If we want to send data or string while returning to web page

```
@Controller
public class MyCafeControllers {

    @RequestMapping("/cafe")
    public String showWelcomePage(Model model)
    {
        //sending data to view(jsp page)
        String myName = "Abhilash";

        model.addAttribute("myNameValue", myName);

        return "welcom-page";
    }
```




```java
@RequestMapping("/processOrder")
public String processOrder(HttpServletRequest request,Model model)
{
    //handle the data received from the user
    String userEnteredValue = request.getParameter("foodType");

    //adding the captured value to the model

    model.addAttribute("userInput", userEnteredValue);

    //set the user data with the model object and send it to view
    return "process-order";
}
```

Sunday, December 27, 2020 3:31 AM

Single design pattern

Singleton Design pattern - Properties

- Creational design pattern
- Only one instance of the class should exist
- Other classes should be able to get Instance of Singleton class
- Used in Logging, Cache, Session, Drivers

Singleton Design pattern - Implementation

- Constructor should be private
- Public method for returning instance
- Instance type – private static

Initialisation Type:

- Eager Initialisation
- Lazy Initialisation
- Thread safe Method Initialisation
- Thread safe block Initialisation

```
    private Singlepatt() {

    }

    static Singlepatt getinsta() {

        if (single == null) {
            single = new Singlepatt();
            return single;
        } else {
            return single;
        }
    }
}

class Single {
    static Single single;

    private Single() {

    }

    static synchronized Single getinstance() {
        if (single == null) {
            single = new Single();
            return single;
        } else {
            return single;
        }
    }
}
```

Builder Design pattern - Properties

- Creational design pattern
- Used when we have too many arguments to send in Constructor & it's hard to maintain the order.
- When we don't want to send all parameters in Object initialisation

(Generally we send optional parameters as Null)

Used when we have too many arguments to send in constructor & it's hard to maintain order

```
package builder;

class Vehicle {
  //required parameter
  private String engine;
  private int wheel;

  //optional parameter
  private int airbags;

  public String getEngine() {
    return this.engine;
  }

  public int getWheel() {
    return this.wheel;
  }

  public int getAirbags() {
    return this.airbags;
  }

  private Vehicle(VehicleBuilder builder) {
    this.engine = builder.engine;
    this.wheel = builder.wheel;
    this.airbags = builder.airbags;
  }

  public static class VehicleBuilder {
    private String engine;
    private int wheel;

    private int airbags;

    public VehicleBuilder(String engine, int wheel) {
      this.engine = engine;
      this.wheel = wheel;
    }
```

```
ublic static class VehicleBuilder {
  private String engine;
  private int wheel;

  private int airbags;

  public VehicleBuilder(String engine, int wheel) {
    this.engine = engine;
    this.wheel = wheel;
  }

  public VehicleBuilder setAirbags(int airbags) {
    this.airbags = airbags;
    return this;
  }

  public Vehicle build() {
    return new Vehicle(this);
  }


lic class BuilderPatternExample {

ublic static void main(String[] args) {
  Vehicle car = new Vehicle.VehicleBuilder("1500cc", 4).setAirbags(4).build();
  Vehicle bike = new Vehicle.VehicleBuilder("250cc", 2).build();

  System.out.println(car.getEngine());
  System.out.println(car.getWheel());
  System.out.println(car.getAirbags());

  System.out.println(bike.getEngine());
  System.out.println(bike.getWheel());
  System.out.println(bike.getAirbags());
```

ConcurrentHashMap->

- Reads can happen very fast while write is done with a lock.
- You should use ConcurrentHashMap when you need very high concurrency in your project.

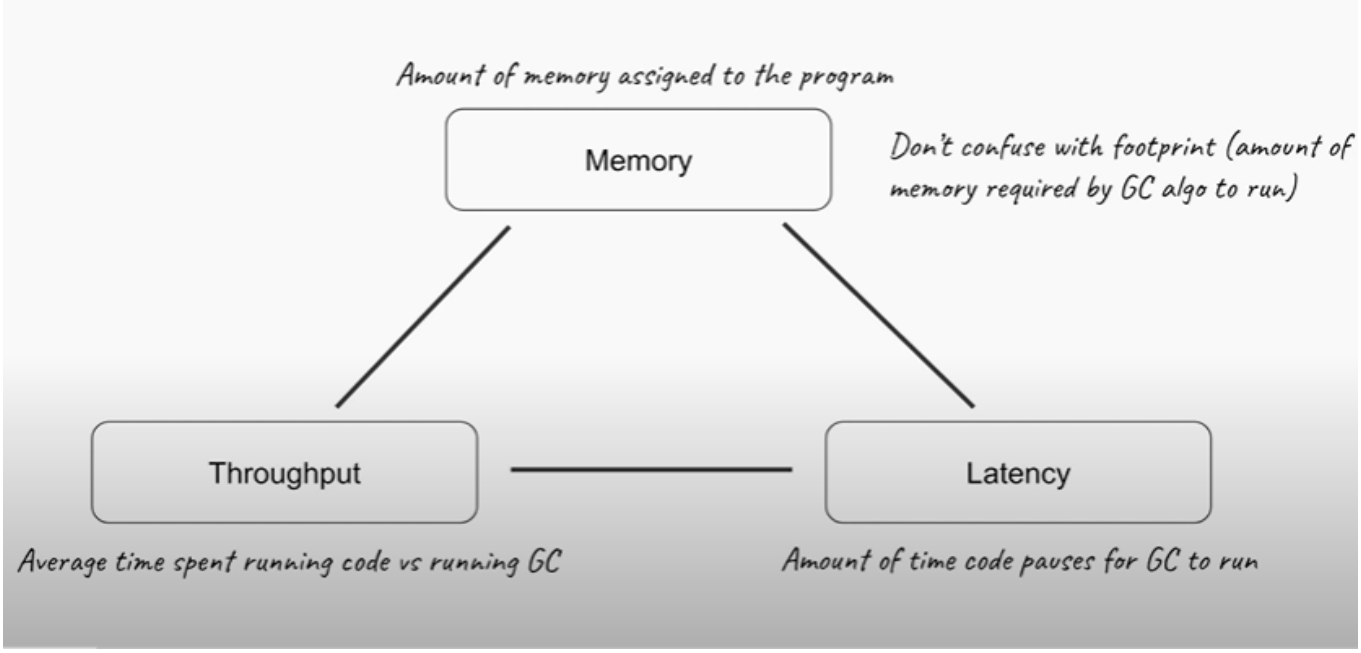
- It is thread safe without synchronizing the whole map.
- Reads can happen very fast while write is done with a lock.
- There is no locking at the object level.

>

SynchronizedHashMap->

- Every read/write operation needs to acquire lock.
- Locking the entire collection is a performance overhead.
- This essentially gives access to only one thread to the entire map & blocks all the other threads.

From <https://crunchify.com/hashmap-vs-concurrenthashmap-vs-synchronizedmap-how-a-hashmap-can-be-synchronized-in-java/>

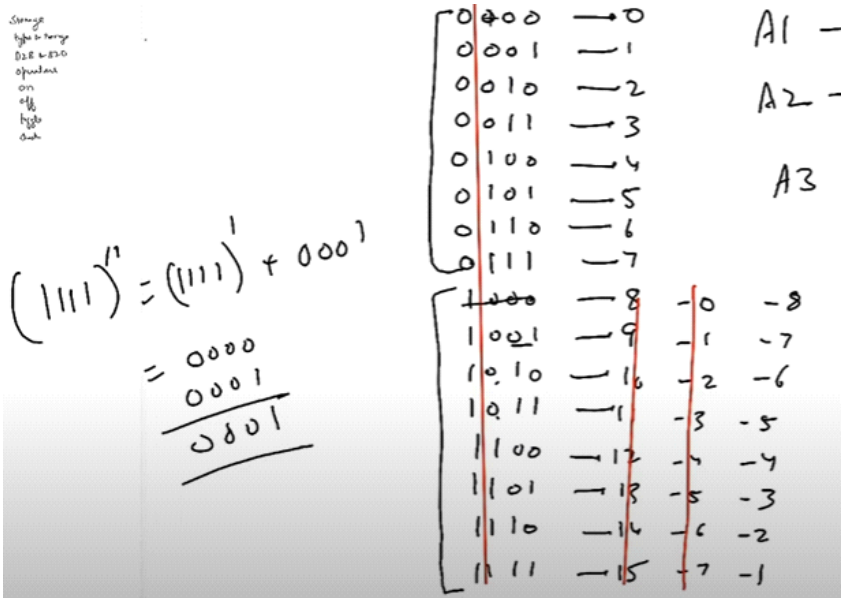


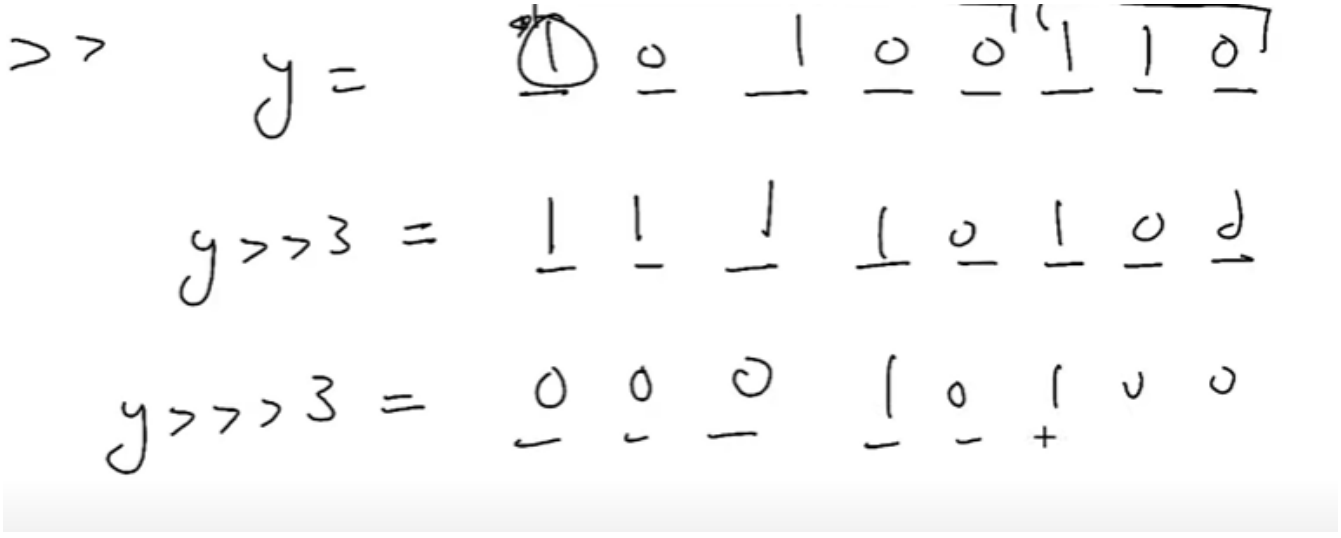
Garbage collection in java is an automatic process of looking at heap memory, identifying which objects are in use and which are not, and deleting the unused objects.

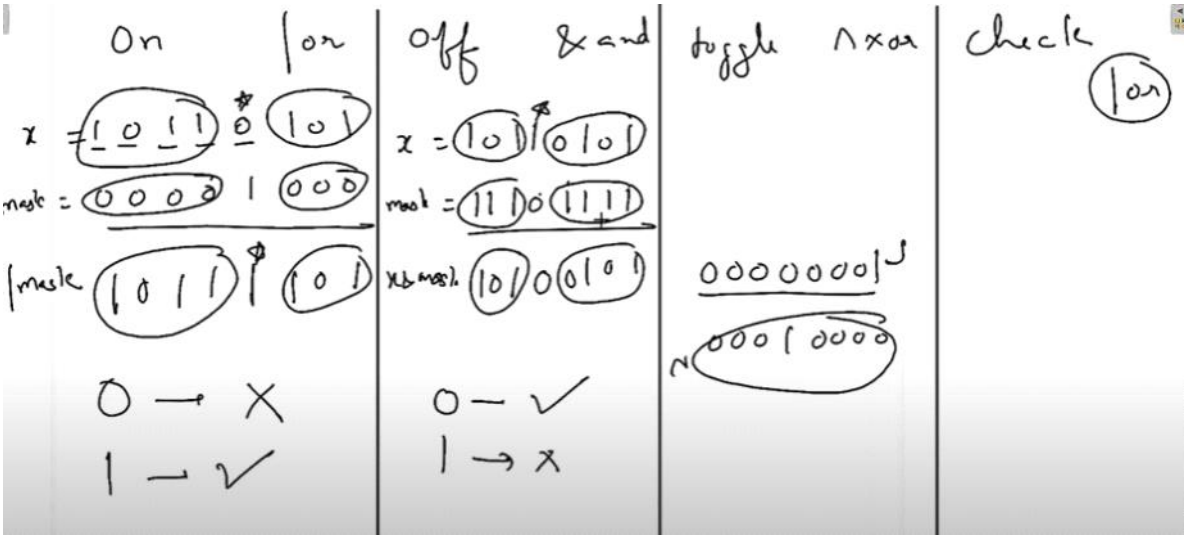
An in use object, or a referenced object, means that some part of your program still maintains a pointer to that object.

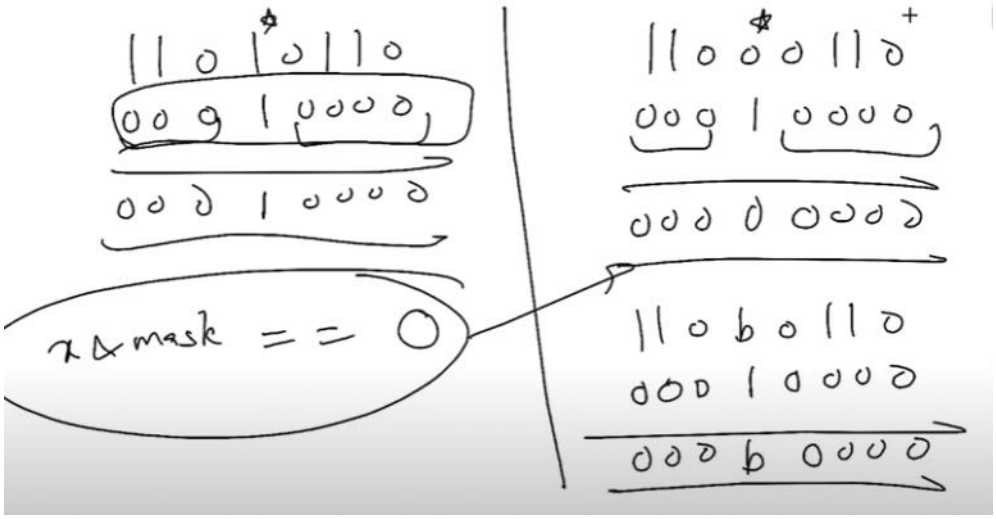
An unused object, or unreferenced object, is no longer referenced by any part of your program. So the memory used by an unreferenced object can be reclaimed.

Main **Advantage** of automatic garbage collection in java is that it removes the burden of manual memory allocation/deallocation from us so that we can focus on problem solving.


A1 -
A2 -
A3


>>
y =
y>>3 =
y>>>3 =


On
|or
Off
&and
toggle
^xor
check
|or
mask
x &mask
0 → X
1 → ✓
0 → ✓
1 → X

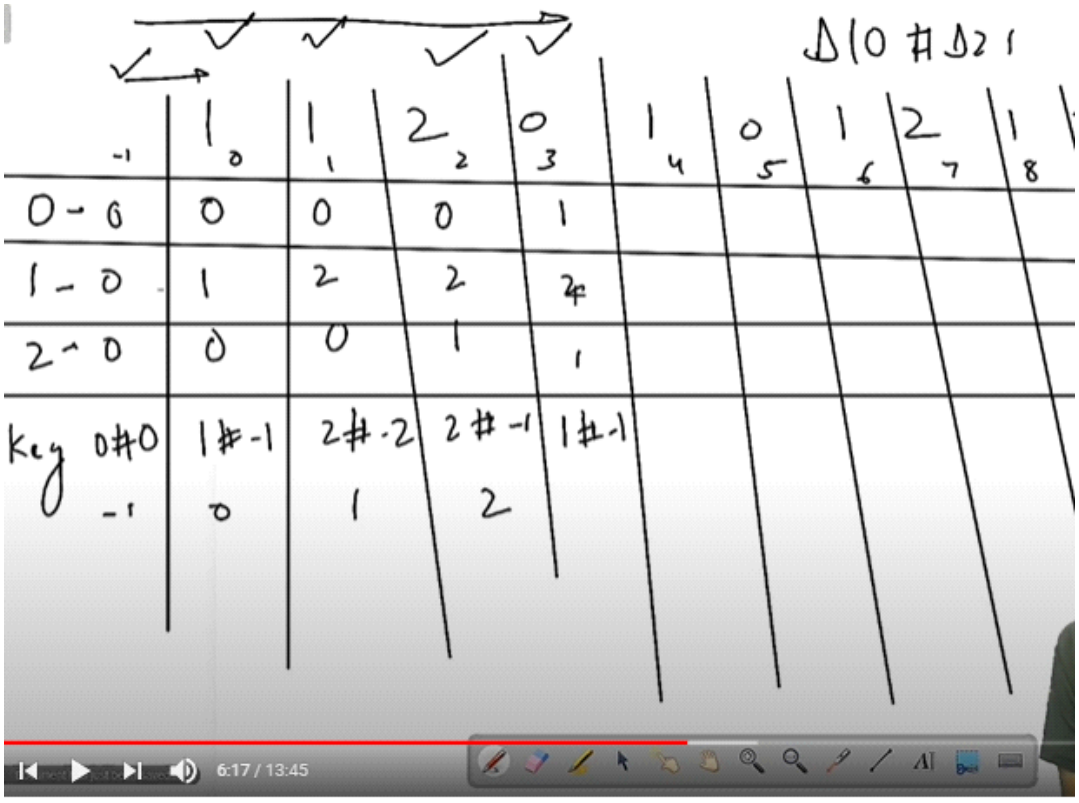



```
//write your code here
int rsbm = n & -n;

System.out.println(Integer.toBinaryString()
```

Most significtnt one -> value & 2's compliment value

```
boolean flag = false;
int rev = 0;
int j = 0;

for(int i = 31; i >= 0; i--){
    int mask = (1 << i);

    if(flag){
        if((n & mask) != 0){
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
        } else {
            System.out.print(0);
        }

        j++;
    } else {
        if((n & mask) != 0){
            flag = true;
            System.out.print(1);

            int smask = (1 << j);
            rev |= smask;
            j++;
        } else {
        }
    }
}
```

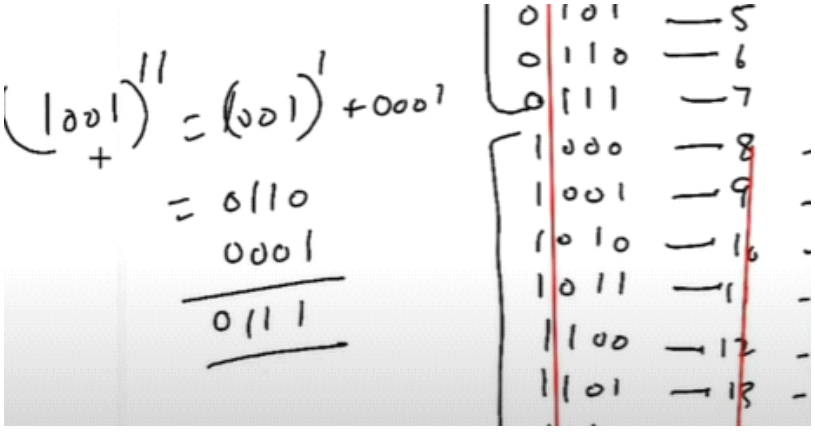
Reverse bit



ashmaps #datastructure #algorithms

ongest Subarray with Equal 0s 1s and 2s | Hashmap Interview Questions Playlist

i2 views • Oct 25, 2020



One compliment toggle whole value
Add+1

```
intv=13;
intw=1<<2;   ~0010=  1101
System.out.println(v&(~w));
Bits off formula at particular position

Int v=13
Int w =1<<2
System.out.println(v|w) bits on

System.out.println(Integer.toBinaryString(v));
intw=(~v);
System.out.println(Integer.toBinaryString(v&(w+1)));
intvd=(24&-24);
System.out.println(Integer.toBinaryString(vd));
```

Sum of value

```
Scannersc=newScanner(System.in);
intsum=0;
for(inti=0;i<4;i++){
System.out.println("Entervalue");
sum=sum|1<<sc.nextInt();
}
System.out.println(sum);
```

```
Value=32
Value=31===(1<<5)-1
```

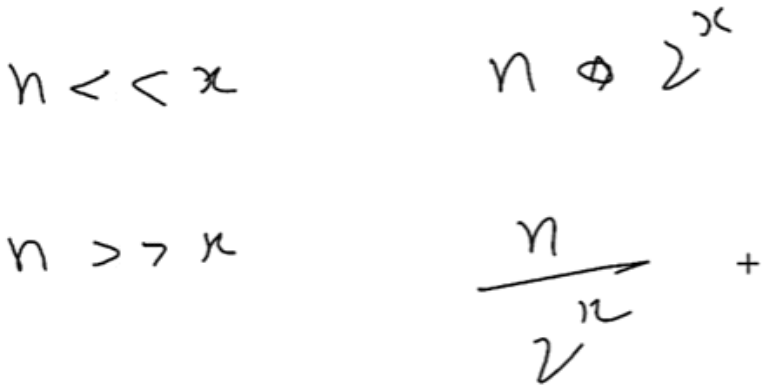
```
25|12
sum|people[i]
```

```
System.out.println(((n<<3)-n)>>3);  =>>8n-n/8
```

```
System.out.println(n<<1); n=8->16
System.out.println(n<<2); n=8->32
System.out.println(n<<3);n=8->64
System.out.println(n>>3);n=64->8
```

```
Quotient remainder
If divide by 4  last 2 will be reminader
15|4
11| 11

If divide by 8  last 3 will be reminader
15|8
1|111
```

```
If we shift by any number then it will become twice

111<<1-> 111=7
1110->  1*2+1*4+1*8

If we shift by two then it will multiplied by 4
```



Threads->

Join-> When two thread running then main thread will wait to execute the bot the thread. Because both the thread become after join

Inter thread communictaion-> All should be synchronized
Notify()-> Whent the thread is in waiting state then notify the waiting state to start the resuming the thread executiom
Wait()->When the thread is waiting it is waiting for notify to start execution

Executorservice provide thread pool to execute the task which will automatically work when one thread done the work

Creating the thread is expensive task
Fixed number of thread

```
packagepractice;

importjava.util.concurrent.ExecutorService;
importjava.util.concurrent.Executors;

classCounterimplementsRunnable{
staticintid=0;
intcount;

publicCounter(){
this.count=0;
}

publicsynchronizedintgetCount()throwsInterruptedException{
wait();
this.count=count+1;
System.out.println(count);
returncount;
}


@Override
publicvoidrun(){
try{
getCount();
}catch(Exceptione){

}

}
}

classCounter1implementsRunnable{
staticintid=0;
intcount;

publicCounter1(){
this.count=0;
```

```
}

publicsynchronizedvoidgetCount()throwsInterruptedException{

while(true){
this.count=count+1;
System.out.println(count);

notify();
}
}

@Override
publicvoidrun(){
try{
getCount();

}catch(Exceptione){

}
}
}

publicclassApplictaion{
publicstaticvoidmain(String[]args)throwsInterruptedException{
ExecutorServiceexecutorService=Executors.newFixedThreadPool(4);
executorService.execute(newThread(newCounter()));
executorService.execute(newThread(newCounter1()));
}
}
```

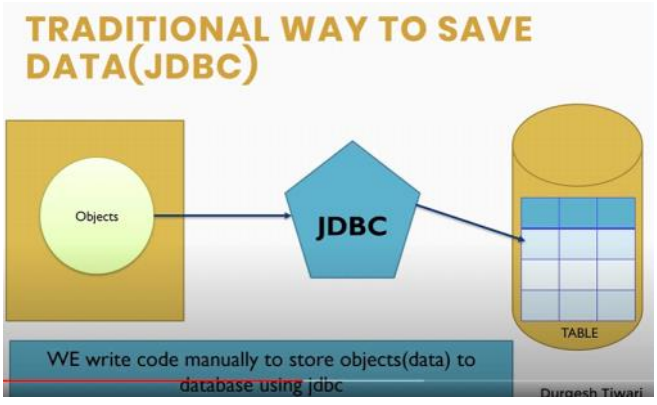
Saturday, December 26, 2020 3:50 AM

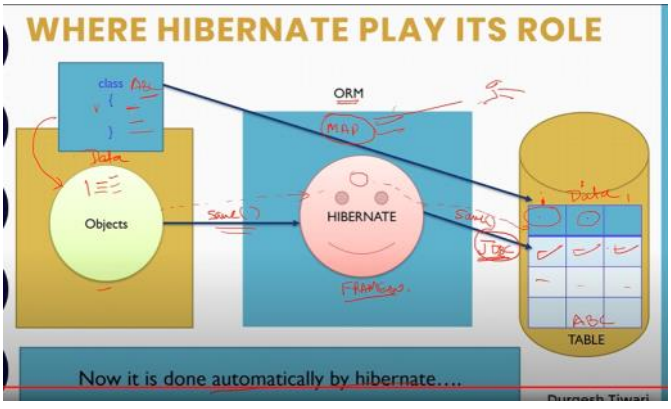
Hibernate

#0. Hibernate Tutorial | Course Overview | Prerequisite of hibernate course | hindi

We don't need to worry about sql query it will be performed by hibernate

- Hibernate is a Java framework that simplifies the development of Java application to interact with the database.
- Hibernate **is ORM (Object Relational Mapping)** tool.
- Hibernate is an Open source, lightweight.
- Hibernate is **a non-invasive** framework, means it won't forces the programmers to extend/implement any class/interface.
- It is invented by **Gavin King in 2001.**
- **Any type of application can build with Hibernate Framework.**





There are 3 objects in hibernate
1.Transient-> Normal object
2.Persistent- object to hibernate object
3.Detached-> hibernate to normal object

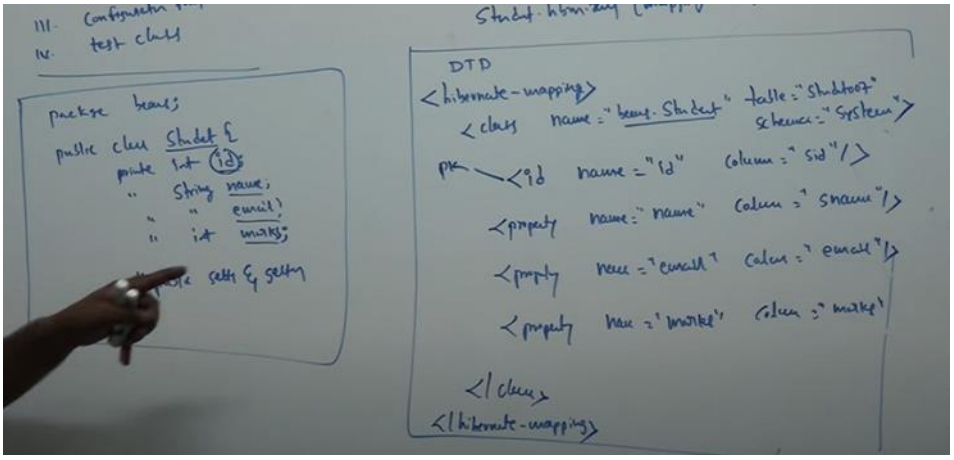
Here we have to make crud operation using object

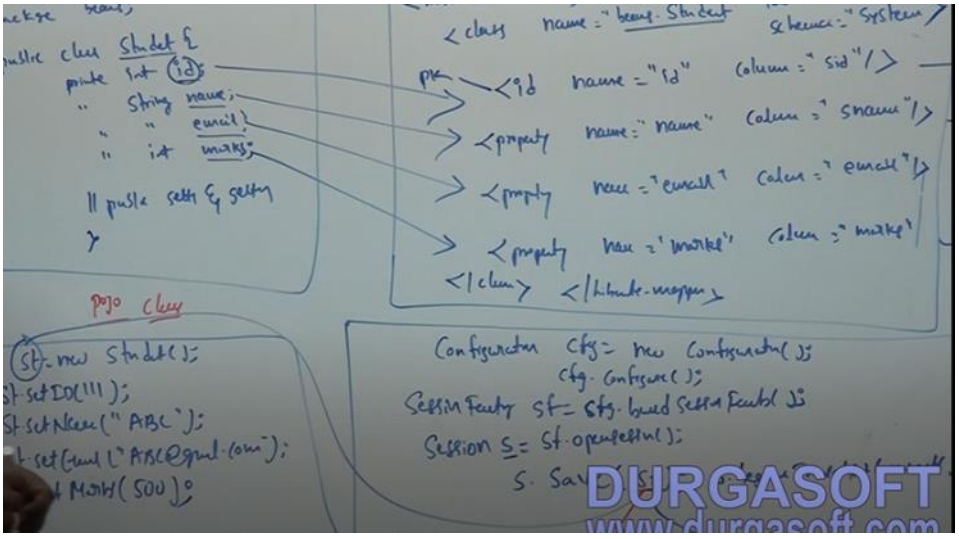
Transient object is normal object

Once we attached to hibernate.. This attached is called persistent object

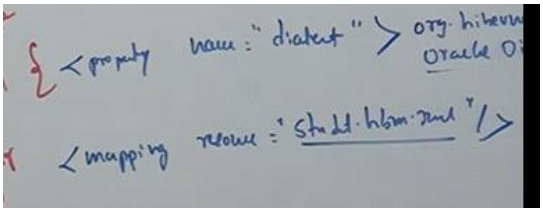
If we remove object from hibernate then it is called detached hiberante

If we do transaction.commit() then it will move database state or permanent state

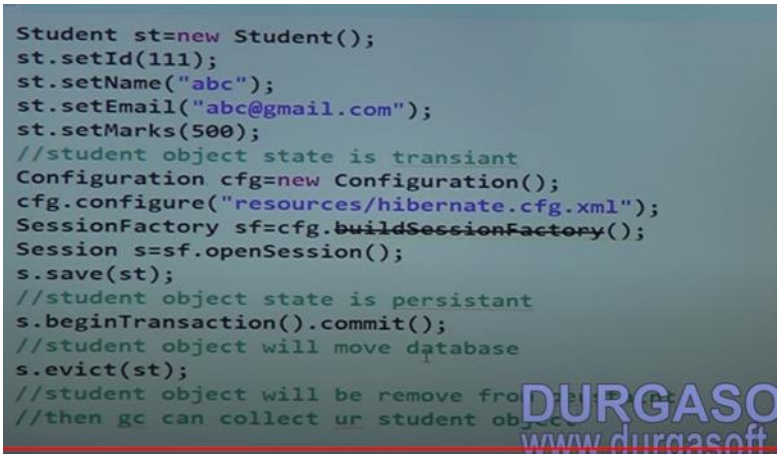




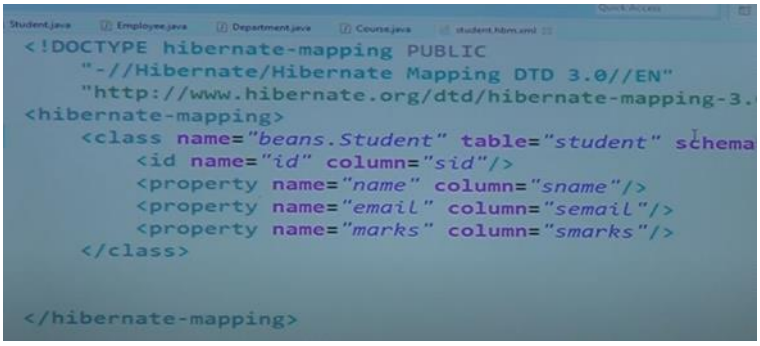
This pojo class is temporary state it can anytime collected by garbage collector
So to avoid that we need to add that in s.save

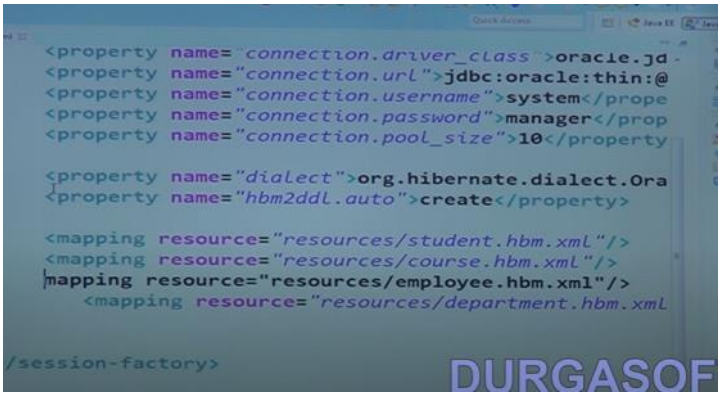


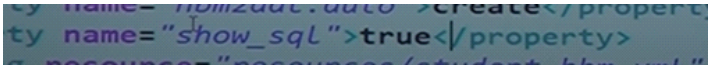
We need to attach mapping file in hibernate.config.file



Hbm2ddl.auot> create<
Then it will automatically create table with dropping earlier schema



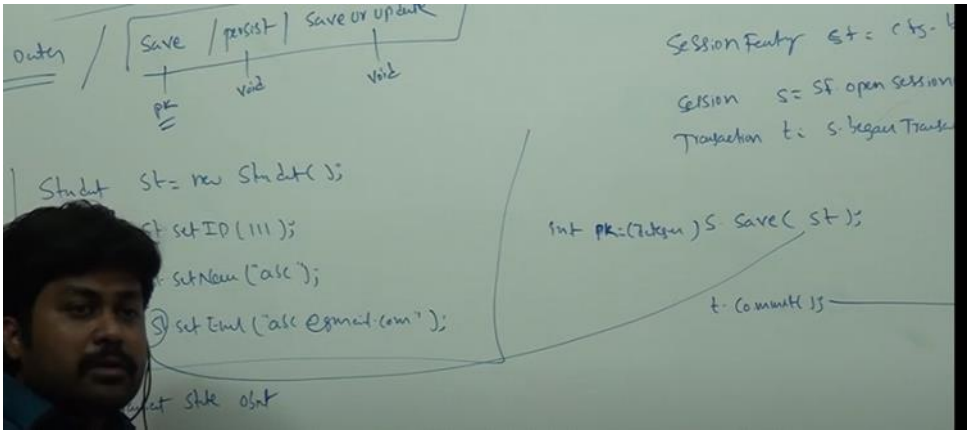




Internal table creation

Hbm2ddl.auot> Update<
Then it will automatically create table with dropping earlier schema

Validate
Must have that 5 coloumns



s.save(st)will return primary key
s.persist(st) will return void

For update records we have update and merge

Update will not work in cache method so then we need to do with merge method

Serialzible

```
public static void main(String[] args) throws IOException, Exception {
    Save s= new Save();
    s.id=10;
    File f= new File("C:\\Users\\kumarmur\\Desktop\\output.txt");
    FileOutputStream fl= new FileOutputStream(f);
    ObjectOutputStream dl= new ObjectOutputStream(fl);
    dl.writeObject(s); //state of object means s.id=10 will be stored


    FileInputStream fl1= new FileInputStream(f);
    ObjectInputStream dl1= new ObjectInputStream(fl1);
    Save s1=(Save) dl1.readObject();
    System.out.println(s1.id);

}

static class Save implements Serializable { // we cannot allow to store object bydefault
    // it can be used for malicious purpsoe so we use serialize interface
    int id;
```

If we excute saveorupdate(st) opeartion

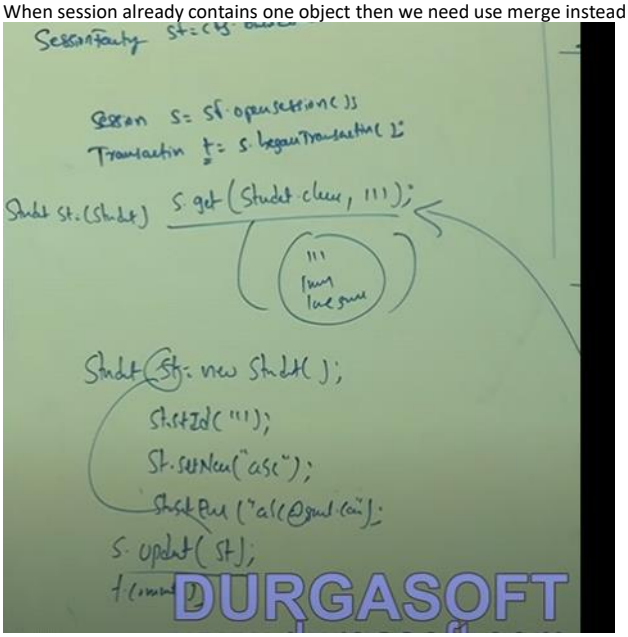
First it will run select query on that primary id;
Then it will compare

Using s.update(St);
s.merge(st);

Limitation
We can only update non primary keys
It is not possible to update one coloumn

When to use merge

When session already contains one object then we need use merge instead update otherwise it will throw exception of duplicate



```
Transaction t = session.beginTransaction();

session.get(Student.class, 111);

Student st=new Student();
st.setId(111);
st.setName("ABC");
st.setEmail("ABC@GMAIL.COM");
st.setAddress("HYD");

session.update(st);
t.commit();
session.close();
sf.close();
System.out.println("update success");
```

```
public static void main(String[] args) {

    Configuration cfg = new Configuration();
    cfg.configure("resources/oracle.cfg.xml");
    SessionFactory sf = cfg.buildSessionFactory();
    Session session = sf.openSession();
    Transaction t = session.beginTransaction();

    Student st=new Student();
    Student st=new Student();
```

```
Student st=new Student();

st.setId(222);

session.delete(st);
t.commit();
session.close();
sf.close();
System.out.println("");
```

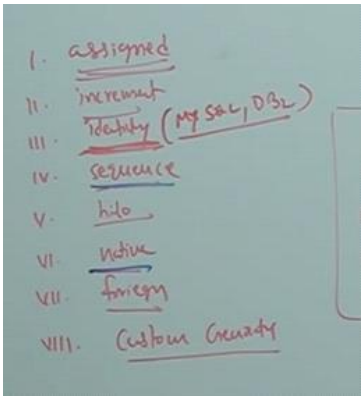
```
Object o=session.get(Student.class, 111);

Student st=(Student)o;
System.out.println(st.getId());
System.out.println(st.getName());
System.out.println(st.getEmail());
System.out.println(st.getAddress());


session.close();
sf.close();
System.out.println("select success");
```

It will do select operation only if when we do get of non primary key in session.load(Student.class, 11);

Primary key auto genrated
1.
Assigned (default)-> It is handle by user itself

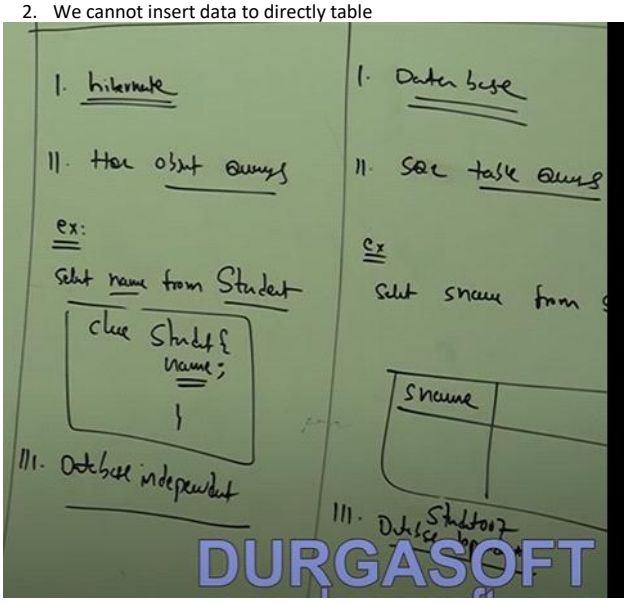
2.Increment-> It will do select max(id) from record and then update by incrmeneting by 1

```
<hibernate-mapping>
    <class name="beans.BookMovie" table="tbc

    <id name="id">
        <generator class="assigned"/>
    </id>
    <property name=""></property>


    </class>
```

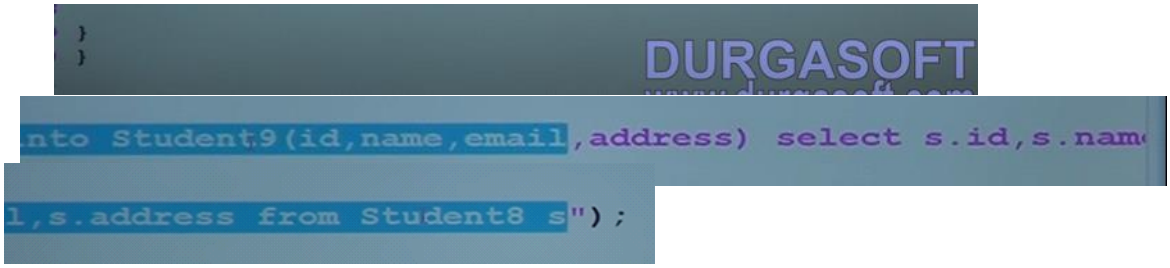


HQL-> Hiberante Query Language

1. We can shift one table data to other table
2. We cannot insert data to directly table




```
        Transaction t=s.beginTransaction();

        //insert

        //one table data we have to insert into another tab

Query q=s.createQuery("insert into Students(id,name,ema
int i=q.executeUpdate();
t.commit();
sf.close();
System.out.println("insert success");


}
}
```



Int i=q.excuteUpdate();
I is how many rows affected by this query

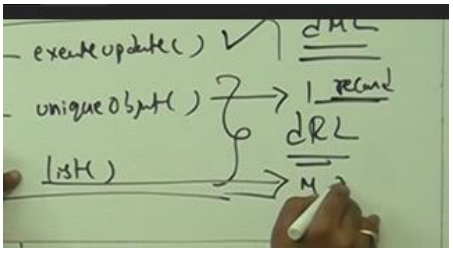
Update Hql -> Resolve merge and update problem-> That earlier we have to update all value of coloumn and cannot upadate primary key

```
Session session=sf.openSession();
Transaction t=session.beginTransaction();
String hql="update Student set name='ABC',email='ABC@y
Query query=session.createQuery(hql);
int i=query.executeUpdate();
System.out.println(i);
session.close();
```

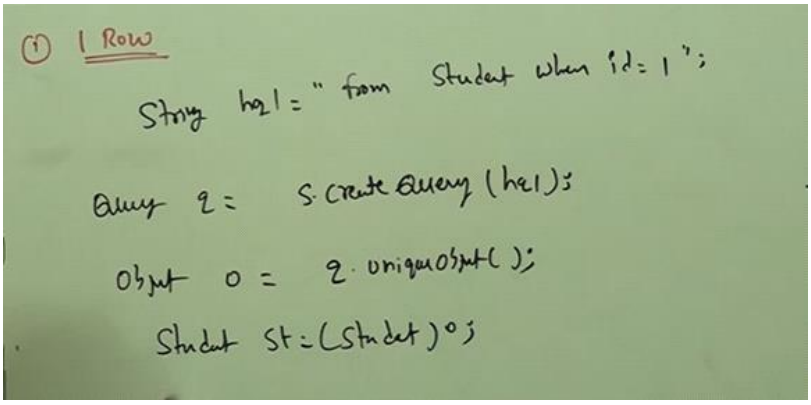


```
public static void main(String[] args) {
    Configuration cfg=new Configuration();
    cfg.configure("resources/oracle.cfg.xml");
    SessionFactory sf=cfg.buildSessionFactory();

    Session session=sf.openSession();
    Transaction t=session.beginTransaction();
    String hql="delete Student where id=555";

    Query query=session.createQuery(hql);
    int i        query.executeUpdate();
    t.commit();
    System.out.println(i);
    session.close();
```


Select single row operation->

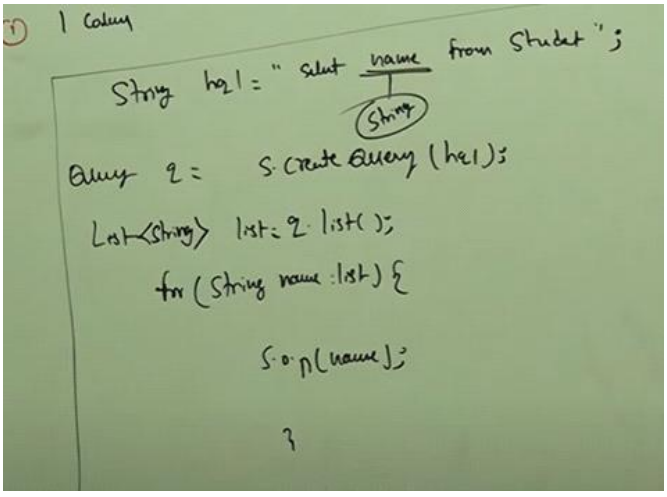


Multiple record= using list method
If only one record then we can go with unqiue record




```
Configuration cfg = new Configuration();
cfg.configure("resources/oracle.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="from Employee where id=111";

Query q=session.createQuery(hql);
Employee emp=(Employee) q.uniqueResult();
        System.out.println(emp.getId());
        System.out.println(emp.getName());
        System.out.println(emp.getEmail());
        System.out.println(emp.getSalary());
```

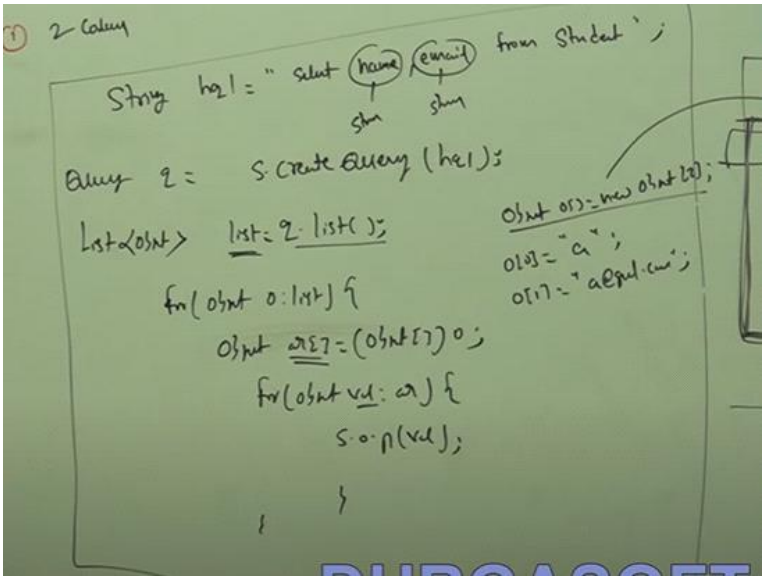
One single object of student then we will use unique object




```
Configuration cfg = new Configuration();
cfg.configure("resources/oracle.cfg.xml");

SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="select name from Employee";
Query q=session.createQuery(hql);
List<String> list=  q.list();
for(String name:list)
{
    System.out.println(name);
}
session.close();
sf.close();
```

If we want to get 1 whole coloumn above one

If we want to get two column then it will convert to object of arrays




```
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="select name,email from Employee";
Query q=session.createQuery(hql);
List<Object> list=  q.list();
for(Object o:list)
{
    Object ar[]=(Object[])o;
    System.out.println(ar[0]);
    System.out.println(ar[1]);

}
session.close();
sf.close();
```

Whole object

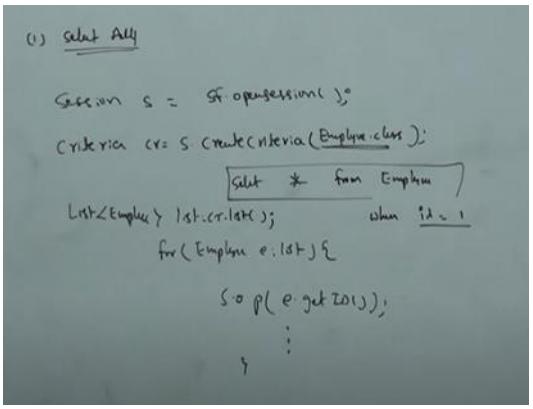
```
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();

String hql="from Employee";
Query q=session.createQuery(hql);
List<Employee> list=    q.list();
for(Employee emp:list)
{
        System.out.println("ID:"+emp.getId());
        System.out.println("NAME:"+emp.getName());
        System.out.println("EMAIL:"+emp.getEmail());
        System.out.println("SALARY:"+emp.getSalary());
}
session.close();
```

Aggregate or sum

```
String hql="select avg(salary) from Employee";
Query q=session.createQuery(hql);
    double avg=(Double)q.uniqueResult();
System.out.println("salary avg="+avg);
session.close();
```

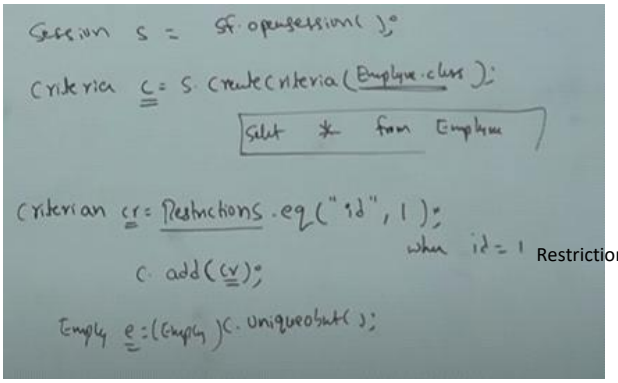
HQL we can use any CRUD operation

Criterias->**we can use only select operation with condition**

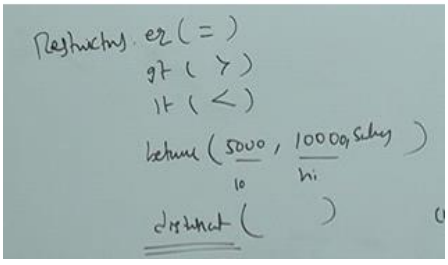


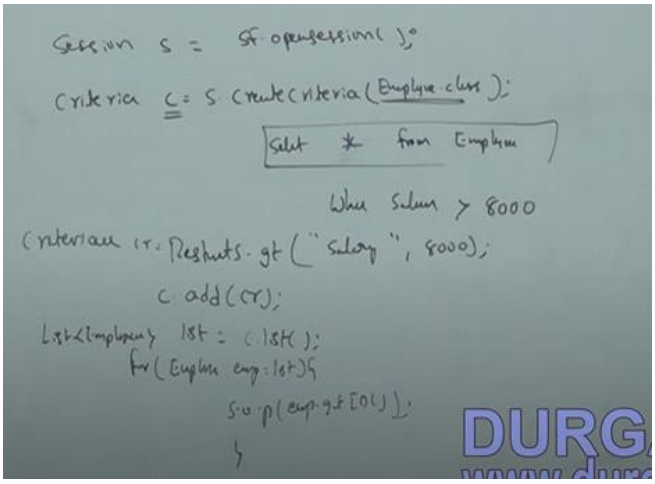
When we want to apply select * from employee where id=1; where condition then we will apply restriction

Crieterian means condition



Restriction means where Id=1



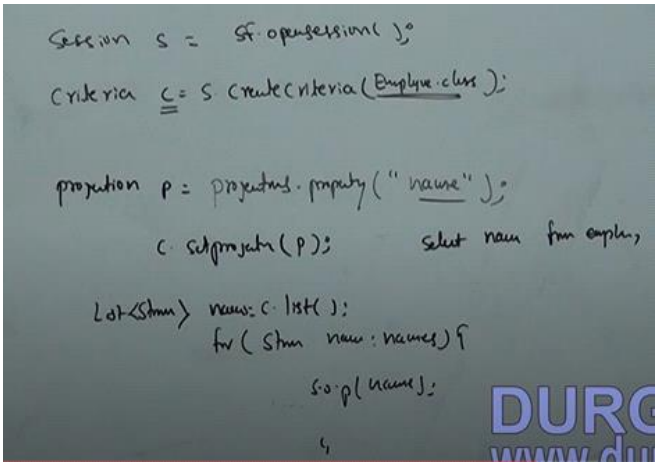


```
SessionFactory sf=cfg.buildSessionFactory();

Session session=sf.openSession();

Criteria c= session.createCriteria(Employee.class);

        //where =
Criterion cr=Restrictions.eq("id", 111);
c.add(cr);
Employee emp=(Employee) c.uniqueResult();
System.out.println(emp.getName());
```

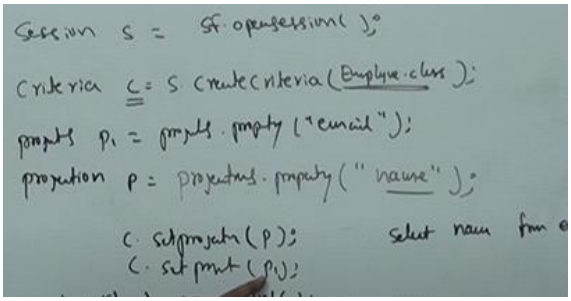
```
        //where =
//Criterion cr=Re
//where >
Criterion cr=Restrictions.gt("salary",70000);
c.add(cr);
    List<Employee>  emplist=c.list();
    for(Employee emp:emplist)
System.out.println(emp.getName()+";
```

Projections->
If we want only one column or max salary or aggregate one value we use projections api

Select name from employee only one coloumn ->Projection.porperty("name");
Partial record->

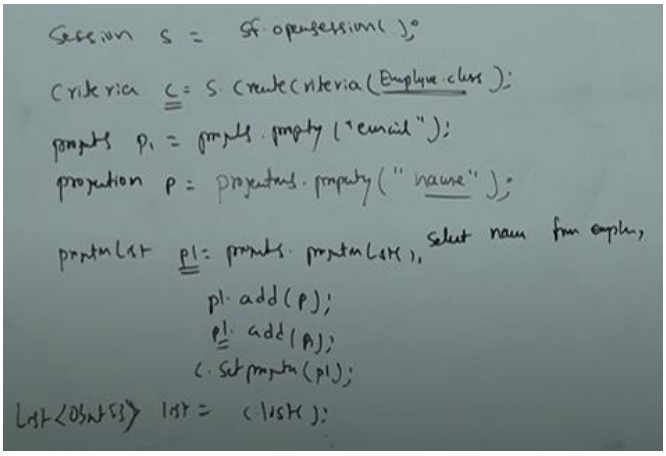


For one crietria we can apply multple criteria
But at a time only one projection

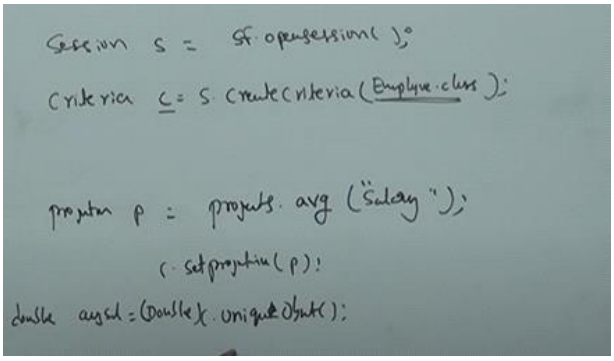


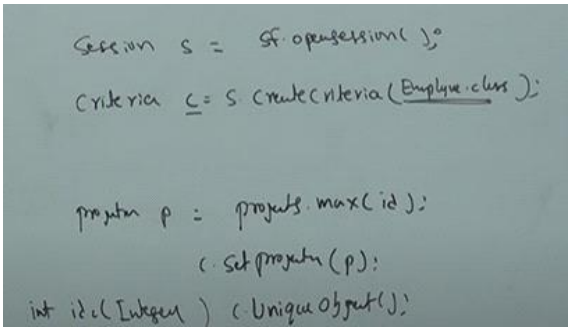
Above image is wrong as first projection will get overide by second one

If we want multiple column then we need to use projection list class



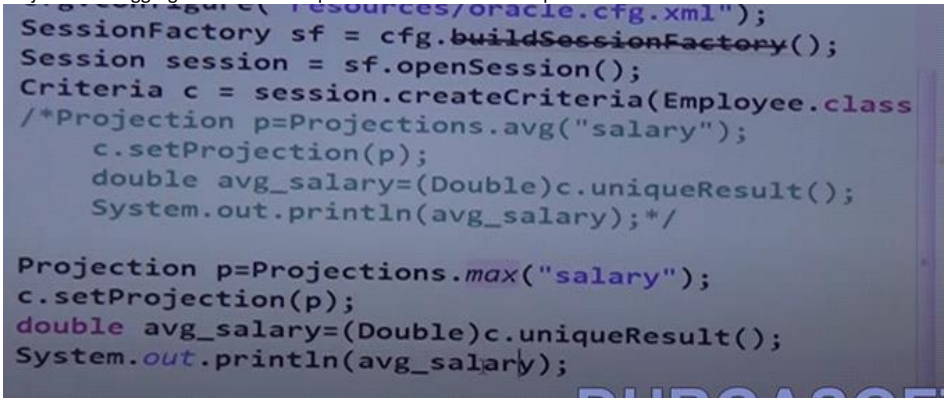
If we want to aggregate or do sum





Restriction are for condtion (=,>,<, between, like)

Projection for aggregate function and particular coloumn select operation



Cache support

One user
All user
One instance

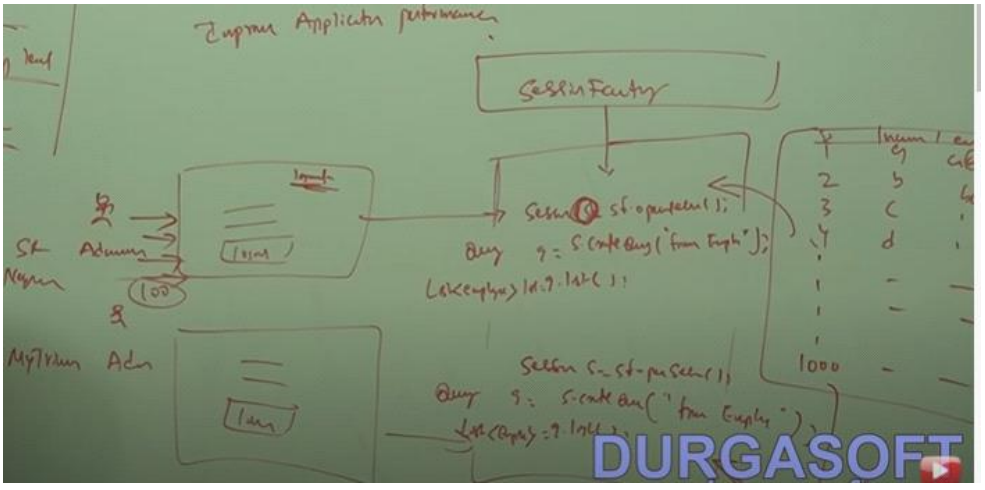
If Constant table if we want playerlist everytime

If constant it will everytime hit database..

View-> Application-> Database
If we want to see
Playerlist
It will hit everytime
To database
10000 hit call from application to database. Then we require I lakh connection and closed.. How much processing will waste

Cache will take temporary storage and it will return 1lakh time..
Cache reduce databse call. Improve application performance

Session cache->
It is useful for one user login to application.
Fetch information for data



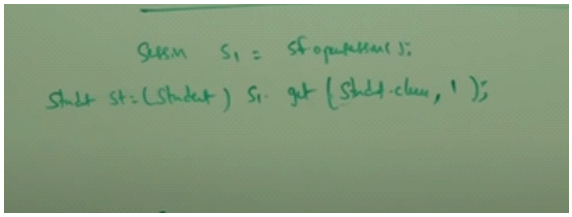
It will fetch only one time per session for one user
For remain 99 call they call from cache only from same session object

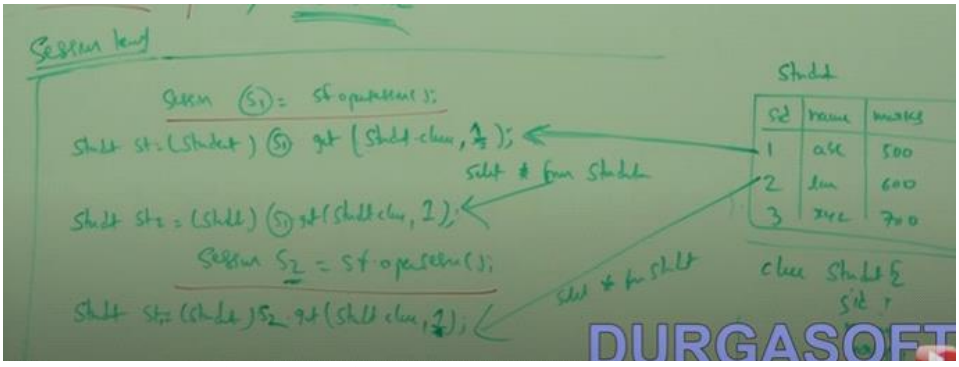
Different user has different session that it will cache
Example- If we login to gmail

1. First time it will retrive from database
2. But after that when we refresh it will return from session cache
3. Until logout only single call to database.
4. In session one time call datatabse

Session factory->
It is also called second level caching
If there are two user then one will hit database then it will avaible for second user also
As it is for session factory level

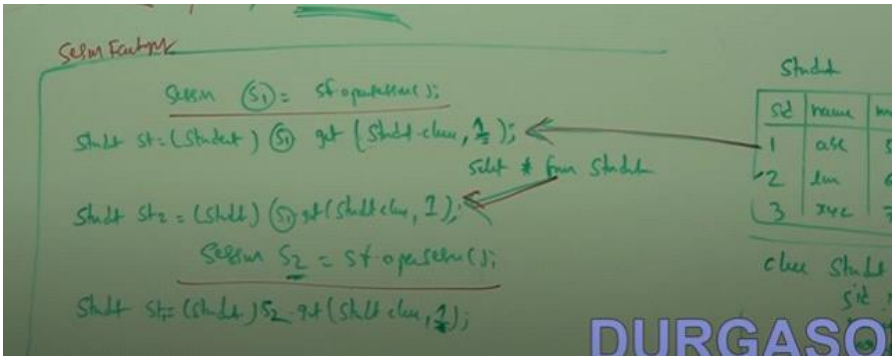
Query level-> Same type query same query like maximum salary.

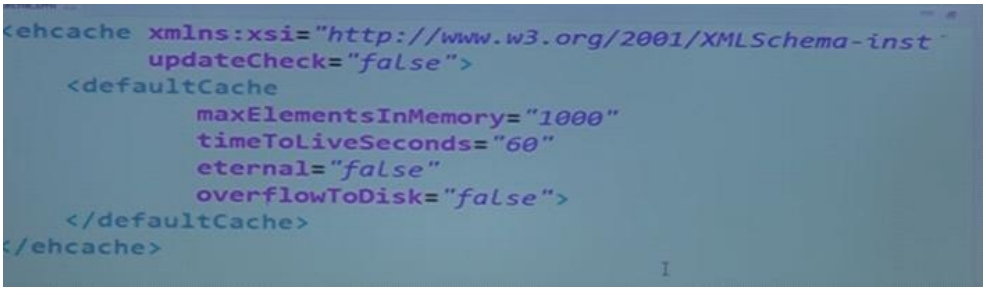




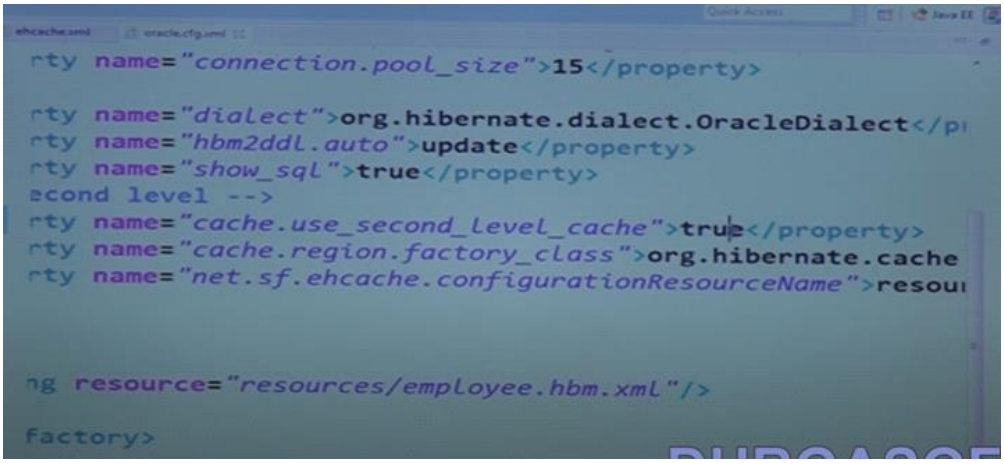
Two call for session cache for two session factory

But sesion factory cahce then same data availbale for both session factory



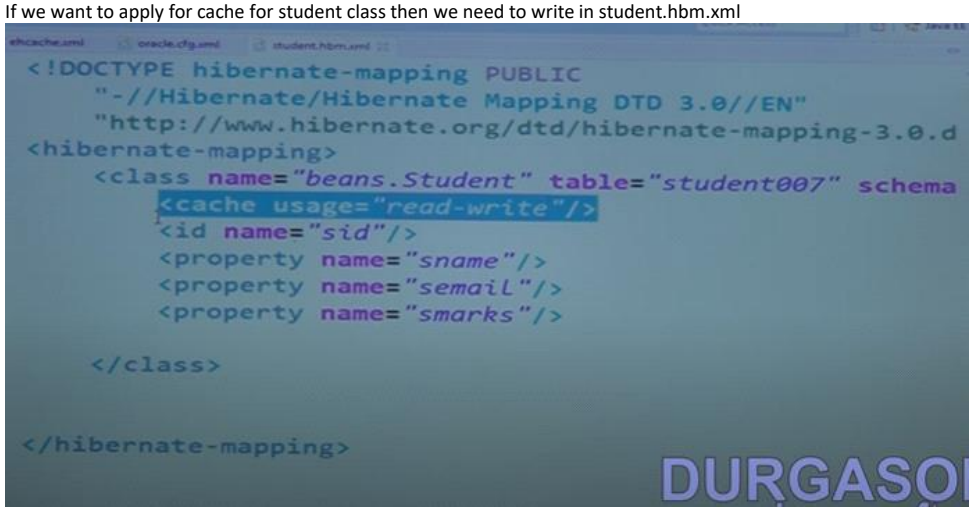


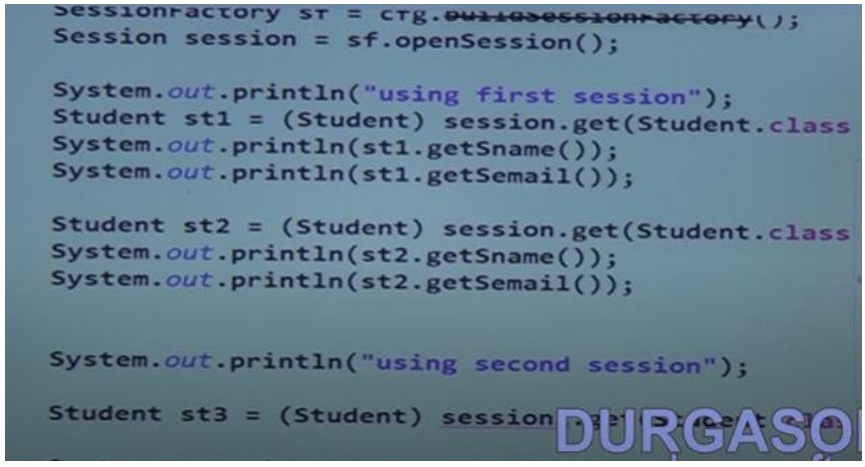
To add session level factory we need to add this
Maxelementinmemory-> Number of objects



We need to add this in configuration file

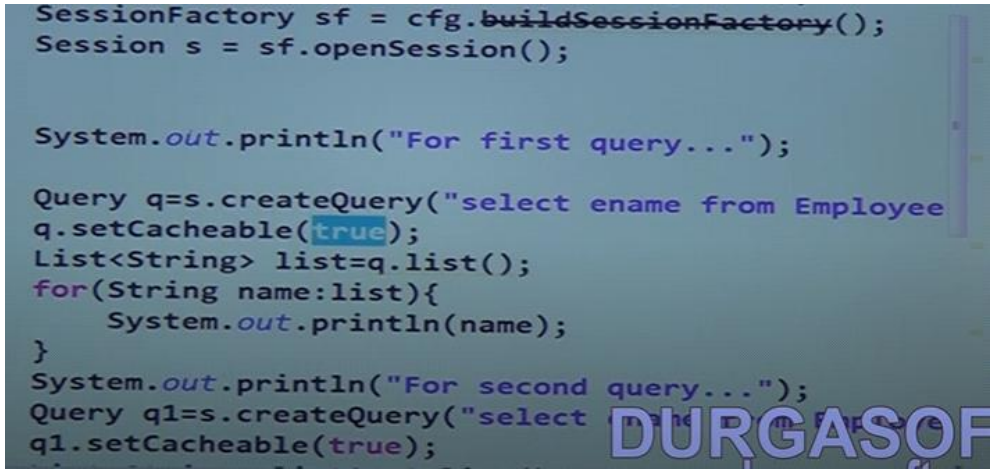
If we want to apply for cache for student class then we need to write in student.hbm.xml





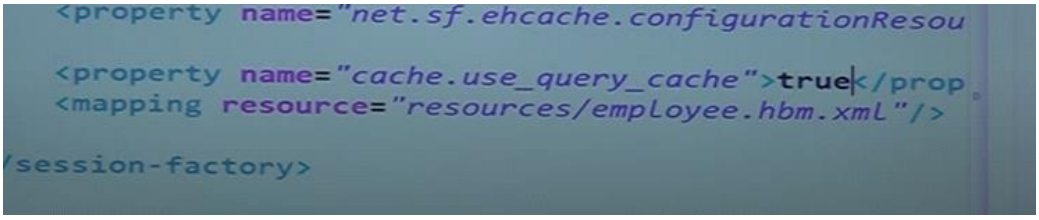
For this st1 and st2 it will fire single query as same session in session cache

But in session factory only one sleect operation



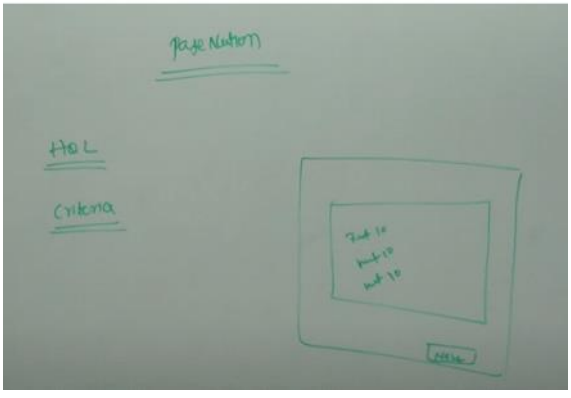
Query cache-> q.setcacheable(True)
It will store this query in cache

```
List<String> list=q.list();
for(String name:list){
    System.out.println(name);
}

System.out.println("For second query...");
Query q1=s.createQuery("select ename from Employee");
q1.setCacheable(true);

List<String> list1=q1.list();
for(String name:list1){
    System.out.println(name);
}
System.out.println("for third querys..");

Query q2=s.createQuery("select ename from Employee");
q2.setCacheable(true);
List<String> list2=q2.list();
```



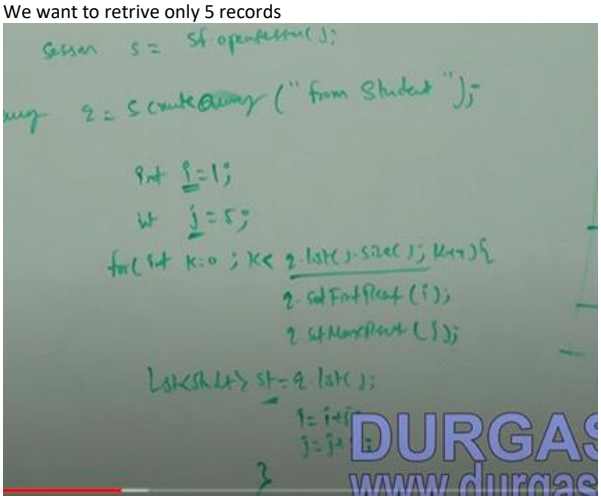

```
e1.setEid(111);
e1.setEname("abc");
e1.setEmail("abc@gmail.com");
e1.setSalary(500000);

Employee e2=new Employee();
e2.setEid(222);
e2.setEname("lmn");
e2.setEmail("lmn@gmail.com");
e2.setSalary(600000);

s.save(e1);
s.save(e2);
s.beginTransaction().commit();
System.out.println("success");
```

Pagination

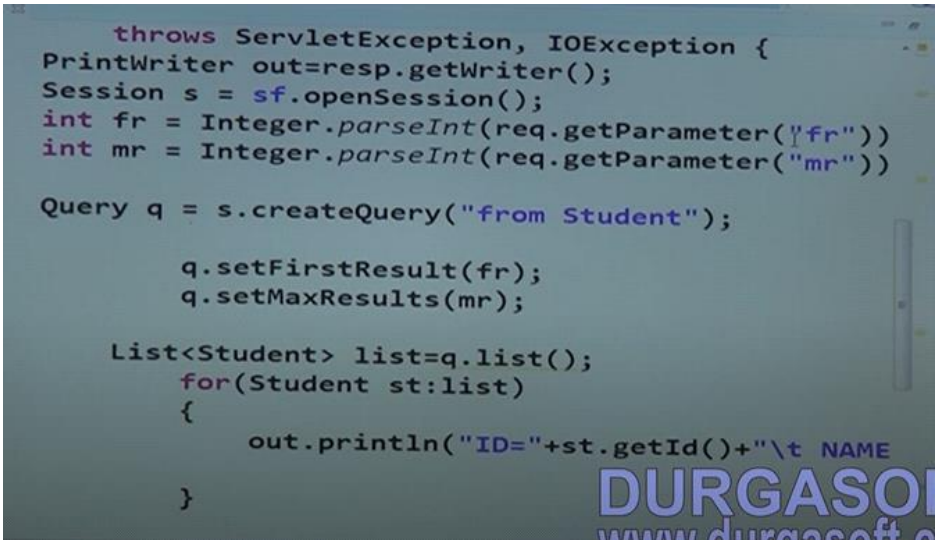


If we want to retrive limited value

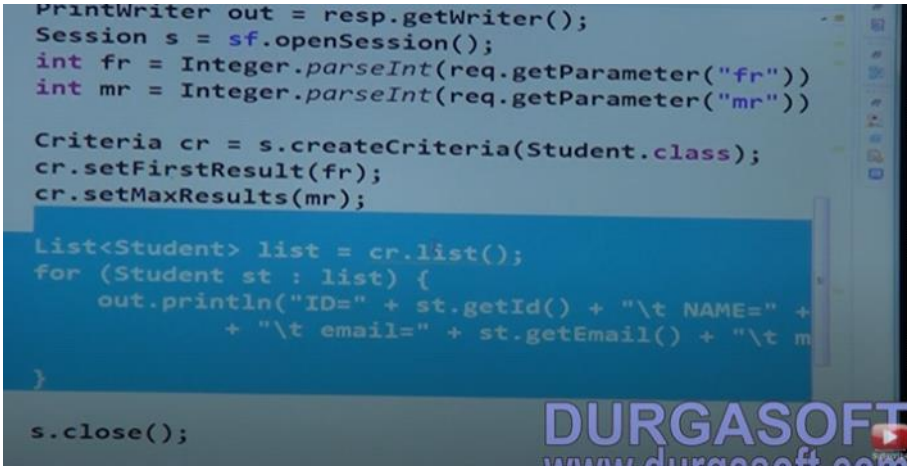
We want to retrive only 5 records



q.setfirstsize(0)
q.setmaxresult(5)



Criteria pagination



@Transaction is session management

```
timeToIdleSeconds="300"
timeToLiveSeconds="600">
```

If it has not been used for 300 seconds
Or after stays in cache for 600 seconds
When echache exceed from its configured size ehcache removed expired elements
If that doesn't clear enough space it cleasr object from ehcache
By default it remove LRU elements

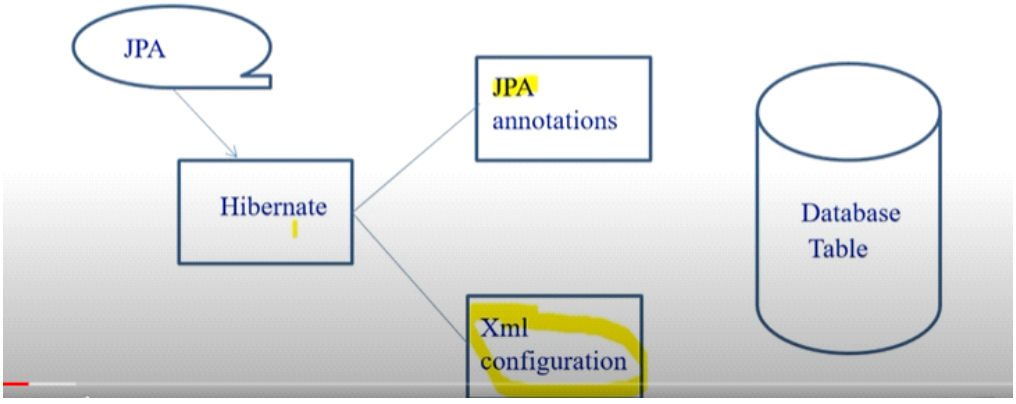
Second level caching is we put on entity or whole object

Default cache is made

```
@Configuration
@EnableCaching
public class CacheConfig {

    @Bean
    public CacheManager cacheManager() {
        SimpleCacheManager cacheManager = new SimpleCacheManager();
        List<Cache> cacheList = new ArrayList<Cache>();
        cacheList.add(new ConcurrentMapCache("userCache"));
        cacheList.add(new ConcurrentMapCache("addressCache"));
        cacheManager.setCaches(cacheList);
        return cacheManager;
    }
}
```

Nosql->
We call collection to table

- **How** ?
- It provides JPA implementation hence we can use JPA annotations as well as xml configurations to achieve this mapping.



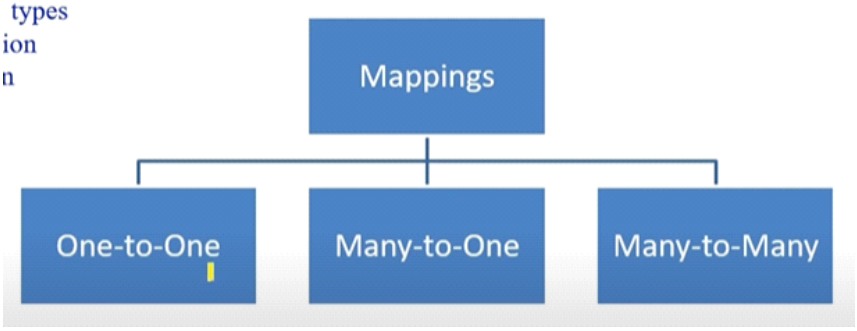
- **Why Hibernate ?**
- Hibernate eliminates all the boiler-plate code that comes with JDBC.
- It supports HQL with is more Object oriented.
- It provides transaction management implicitly.
- Hibernate throws JDBCException or HibernateException which are the unchecked exceptions, so we don't need to worry about handling using try and catch.
- Hibernate supports caching for better performance.

Q) Important Interfaces used in Hibernate

- **SessionFactory (org.hibernate.SessionFactory)** – Instance of this is used to retrieve Session objects for database operations. We need to initialize that once and can cache it to reuse it again and again. Its like one SessionFactory object per database connection. Like 1 for mysql, 1 for oracle.
- **Session (org.hibernate.Session)** – Its factory for transaction, it's used for connecting application with persistant store like hibernate framework / DB. It is used to get a physical connection with the database. It also provides methods for CRUD operations.
- **Transaction (org.hibernate.Transaction).** – This specifies single / atomic units of work

```
SessionFactory factory = metadata.getSessionFactoryBuilder().build();
Session session = factory.openSession();
Transaction t = session.beginTransaction();

session.save(persistantObj);
    t.commit();
factory.close();
    session.close();
```

types
ion
n



Many to Many

```
class Student{

@ManyToMany(targetEntity = Degree.class, cascade = { CascadeType.ALL })
@JoinTable(name = "DegreeStudentThirdTable",
           joinColumns = { @JoinColumn(name = "StudentId") },
           inverseJoinColumns = { @JoinColumn(name = "CertificateId") })
private List<Degree> degrees;

}
```

Q) What are hibernate configuration file

contains database specific configurations and used to initialize SessionFactory.

Conventionally, its name should be hibernate.cfg.xml

If u need to connect to SQL then create another one here.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">codedecode</property>
        <property name="connection.password">codedecode</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</property>
    <mapping resource="employee.hbm.xml"/>
    </session-factory>

</hibernate-configuration>
```

Q) What are hibernate mapping file

The mapping file name conventionally, should be class_name.hbm.xml.

hibernate-mapping : root element
class : specifies the Persistent class.
id : specifies the primary key attribute in the class.
generator : used to generate the primary key.
property : specifies the property name of the Persistent class.

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd

 <hibernate-mapping>
  <class name="com.codedecode.Employee" table="empTable">
    <id name="id">
     <generator class="assigned"></generator>
    </id>

    <property name="firstName"></property>
    <property name="lastName"></property>

  </class>

</hibernate-mapping>
```

Q) difference between openSession and getCurrentSession

getCurrentSession() method returns the session bound to the context.
Since this session object belongs to the context of Hibernate, it is okay if you don't close it. Once the SessionFactory is closed, this session object gets closed.

While

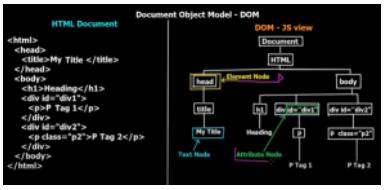
openSession() method helps in opening a new session.
`You should close this session object once you are done with all the database operations. And also, you should open a new session for each request in a multi-threaded environment.

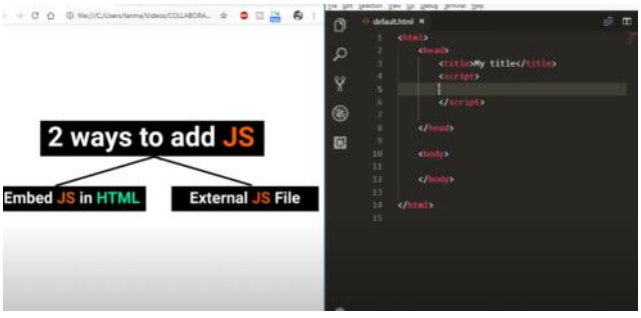
- get() loads the data as soon as it's called whereas load() returns a proxy object and loads data only when it's actually required, **so load() is better because it support lazy loading.**
- **Since load() throws exception** when data is not found, we should use it only when we know data exists.
- We should use **get() when we want to make sure data exists** in the database.

Q) hibernate caching – Second level cache

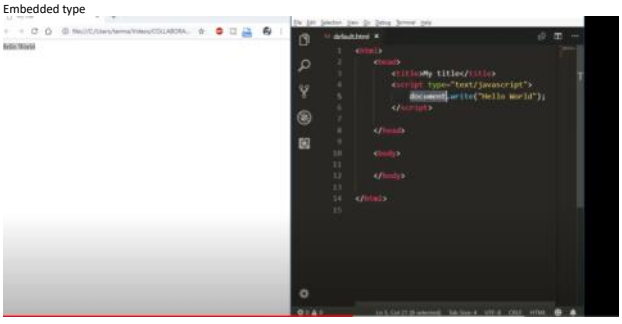
- Hibernate Second Level cache is disabled by default but we can enable it through configuration.
- Currently EHCache and Infinispan provides implementation for Hibernate Second level cache and we can use them.

Java script





Embedded type



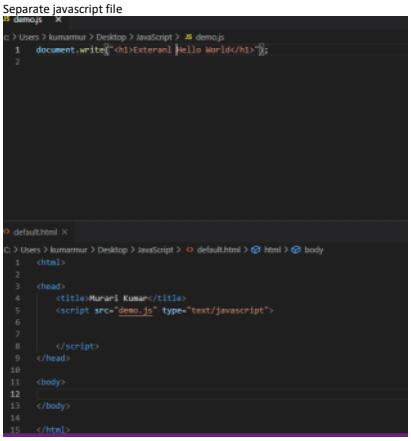
Document.write("Hello world")

```
document.write("<h1>Hello World</h1>");
```

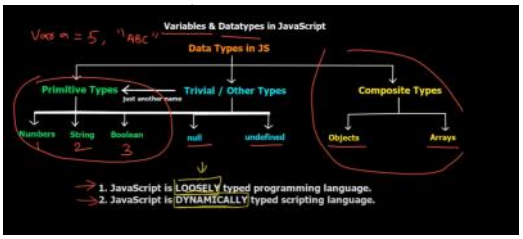
It represent entire document module in dom

Write is method which is directly writes on browser

Separate javascript file



This is statement and instruction to print hello world on screen




```
<script type="text/javascript">

   // int x = 5;

   // JS is LOOSELY typed.
   // JS is DYNAMICALLY TYPED

   var num = 16;  // Number
   var Name = "Tanmay Sakpal";   // String
   var flag = true; // boolean


   document.write(num); // printing on the browser
</script>
</head>
```

```
// JS dynamically typed

var num = 16;  // Number
var Name = "Tanmay Sakpal";   // String
var flag = false; // boolean
num = "Tanmay";
```

```
   Booleans can be objects (if defined with the new
   keyword)
   Numbers can be objects (if defined with the new
   keyword)
   Strings can be objects (if defined with the new
   keyword)
   Dates are always objects
   Maths are always objects
   Regular expressions are always objects
   */
   var str1 = new String();

   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   document.write(typeof(str1));
   //Car.teslaAutoPilot();
```

```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";
```

If we want to add mor property of car

```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";

   Car.newFn = function()
   {
      document.write("<h2>Added function</h2>");
   }
```

Adding metho

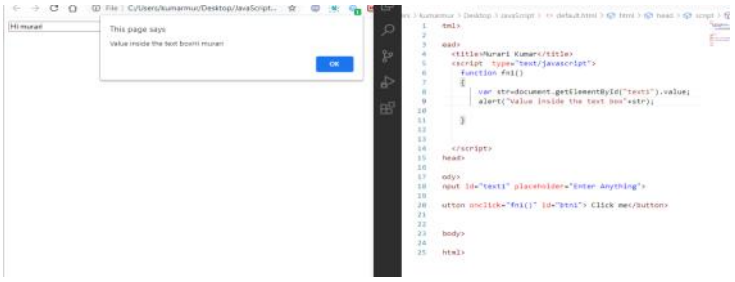
```
   var Car = {
      car_brand :"Tesla",
      car_model : "Model 3",
      price : 35000,
      teslaAutoPilot : function()
      {
         document.write("<h2>This car has Auto pilot</h2>");
      }
   }
   // 100 lines of code

   Car.fuelType ="Electric";
   delete Car.price;

   document.write("<h2>"+Car.price+"</h2>");
   //Car.teslaAutoPilot();
```

Delete property

Button clicked action

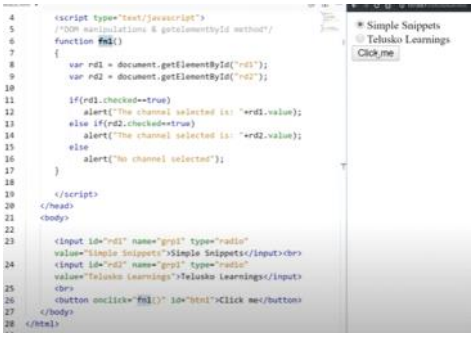
```
<head>
    <title>Murari Kumar</title>
    <script  type="text/javascript">

    function buttonclick()
    {
        alert("Button Clicked");
    }

    </script>
</head>

<body>
    <button onclick="buttonclick()">Click me</button>
    <h1 id="first">Can you please click on button</h1>
</body>

</html>
```

Input and give alter box



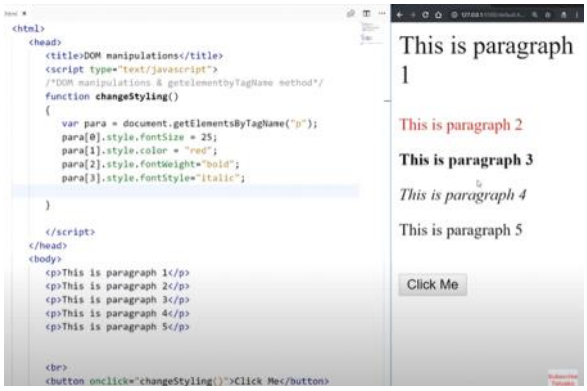
Username & password

Radio group name for both then we can select any one
Other wise we both will be slected if we give different name

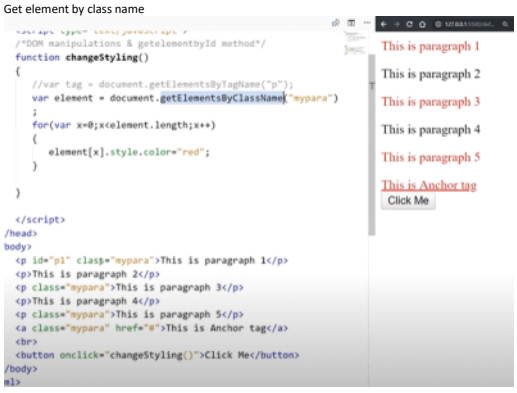


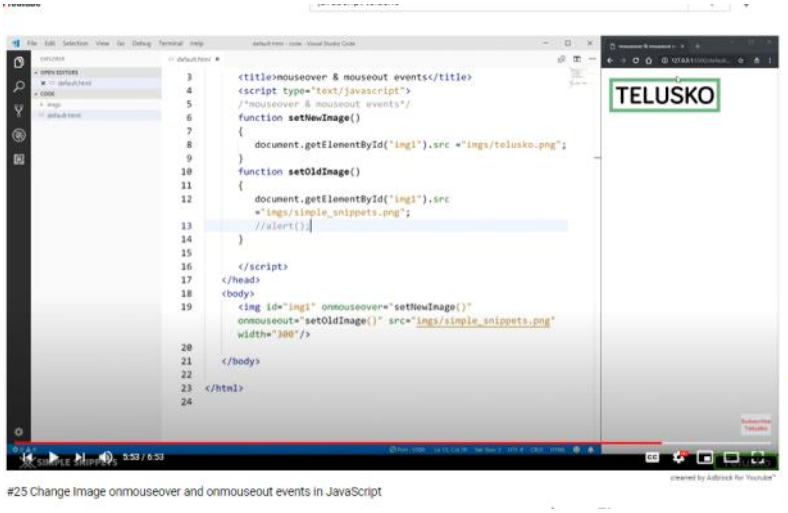

```
<html>

<head>
    <title>Murari Kumar</title>
    <script  type="text/javascript">
    function fn1()
    {
        var r1=document.getElementById("select");
        alert("Pop up  "+r1.value);
    }


    </script>
</head>

<body>

    <select id="select">
<option value="Murari">Murari</option>
<option value="Kumar">Kumar</option>
<option value="Rahul">Rahul</option>
    </select>
<button onclick="fn1()" id="btn1">On click</button>


</body>

</html>
```

Get element by tag name



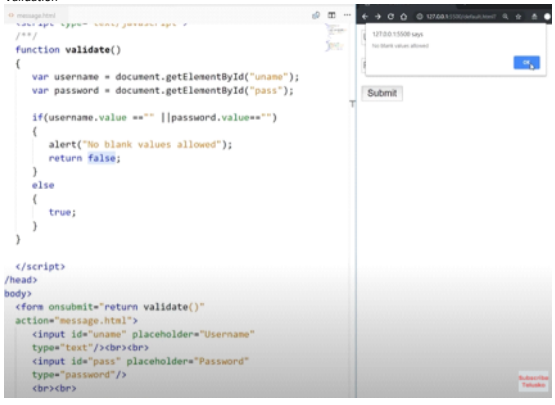
Get element by class name



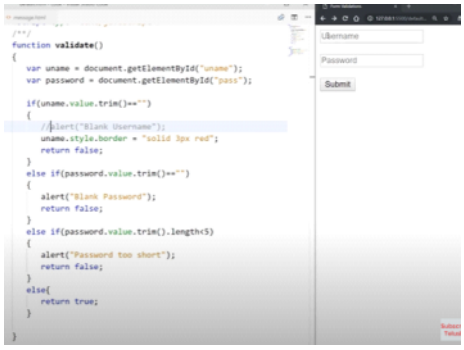


#25 Change Image onmouseover and onmouseout events in JavaScript

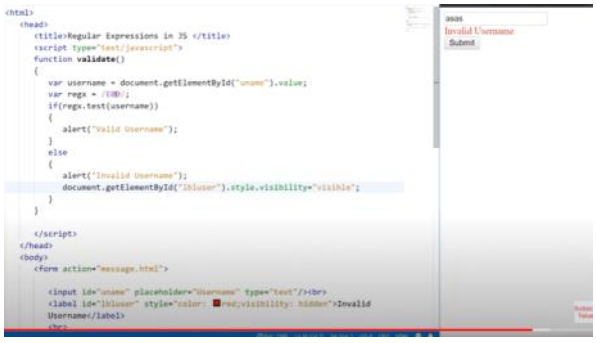
Validation



On submit extra fetaure



```
/**/
function validate()
{
   var uname = document.getElementById("uname");
   var password = document.getElementById("pass");

   if(uname.value.trim()=="")
   {
      //alert("Blank Username");
      uname.style.border = "solid 3px red";
      return false;
   }
   else if(password.value.trim()=="")
   {
      alert("Blank Password");
      return false;
   }
   else if(password.value.trim().length<5)
   {
      alert("Password too short");
      return false;
   }
   else{
      return true;
   }
}
```



```
<html>
   <head>
      <title>Regular Expressions in JS </title>
      <script type="text/javascript">
      function validate()
      {
         var username = document.getElementById("uname").value;
         var regx = /[0-9]/;
         if(regx.test(username))
         {
            alert("Valid Username");
         }
         else
         {
            alert("Invalid Username");
            document.getElementById("lbluser").style.visibility="visible";
         }
      }

      </script>
   </head>
   <body>
      <form action="message.html">

         <input id="uname" placeholder="Username" type="text"/><br>
         <label id="lbluser" style="color: red;visibility: hidden">Invalid
         Username</label>
         <br>
```

JAVA 8

Thursday, January 7, 2021 2:14 AM

```
public class Sample {
  public static void main(String[] args) {
    List<Integer> numbers = Arrays.asList(1, 2, 3, 4, 5, 6, 7, 8, 9, 10);

    //external iterators
    // for(int i = 0; i < numbers.size(); i++) {
    //    System.out.println(numbers.get(i));
    // }

    //external iterators also
    // for(int e : numbers) {
    //    System.out.println(e);
    // }
```

```
  //internal iterators
  // numbers.forEach(new Consumer<Integer>() {
  //    public void accept(Integer value) {
  //       System.out.println(value);
  //    }
  // });


  numbers.forEach(new Consumer<Integer>() {
    public void accept(Integer value) {
      System.out.println(value);
    }
  });
}
```

Lambada be only one line

here are a few smells in this code. Let's discuss each one of them:

1. **The code is hard to read**
   We can't quickly grasp what's going on in this code, it's not conc
   have to keep multiple context in mind, what `e` is and what `i` is a
   we are comparing the two. We lost the general benefits of lamb
   concise and expressive—this code fails both.
2. **It's noisy**
   Writing a single line lambda was nice. The minute we wanted to
   than one line, we have to bring the `{}`, the `;`, and possibly the r
   we had something to return. That's Java's (and C#'s too) way of
   us—it's telling us not to do that.
3. **Leads to Duplication**
   Whatever that multiline code does, it's not reusable. If we wante

Method refrnce-> If we simply get the value and don't want to alter then we can use this

```
// numbers.forEach(e -> System.out.println(e));
// numbers.forEach(System.out::println);

numbers.stream()
       // .map(e -> String.valueOf(e))
       .map(String::valueOf)
       .forEach(System.out::println);
```

```
numbers.stream()
       .map(e -> String.valueOf(e))
       // .map(e -> e.toString())
       .map(String::toString)
       .forEach(System.out::println);
```

```
// numbers.stream()
//         .map(e -> String.valueOf(e))
//         // .map(e -> e.toString())
//         .map(String::toString)
//         .forEach(System.out::println);

System.out.println(
  numbers.stream()
         // .reduce(0, (total, e) -> Integer.sum(total, e)));
         .reduce(0, Integer::sum));
```

If we are not altering just pass method refrnce

```
// System.out.println(
//   numbers.stream()
//          // .reduce(0, (total, e) -> Integer.sum(total, e)));
//          .reduce(0, Integer::sum));

System.out.println(
  numbers.stream()
         .map(String::valueOf)
         // .reduce("", (carry, str) -> carry.concat(str)));
         .reduce("", String::concat));
```

Mobile phone finding big room. More person find resources accessed more

```
    numbers.stream()
           .filter(e -> e % 2 == 0)
           .map(e -> e * 2)
           .forEach(e -> doubleOfEven.add(e));
    //mutability is OK, sharing is nice, shared mutability is devils work

    //friends don't let friends do shared mutation.

    System.out.println(doubleOfEven); //Don't do this.

    List<Integer> doubleOfEven2 =
      numbers.stream()
             .filter(e -> e % 2 == 0)
             .map(e -> e * 2)
             .collect(toList());
    System.out.println();
  }
```

Map->

```
    new Person("Jack", Gender.MALE, 72),
    new Person("Jill", Gender.FEMALE, 12)
  );
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
          )));
}
```

First object passed toMap is key value
Second object passed to get object or value

```
}

public static void main(String[] args) {
  List<Person> people = createPeople();

  //create a Map with name and age as key, and the person as value.

  System.out.println(
    people.stream()
          .collect(toMap(
            person -> person.getName() + "-" + person.getAge(),
            person -> person)));
}
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the people with that name.

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName)));
```

```
-[Bob -- MALE -- 20], Sara=[Sara -- FEMALE -- 20, Sara -- FEMALE -- 22],
```

```java
public static void main(String[] args) {
  List<Person> people = createPeople();

  //given a list of people, create a map where
  //their name is the key and value is all the ages of people with that
ame

  System.out.println(
    people.stream()
          .collect(groupingBy(Person::getName,
              mapping(Person::getAge, toList()))));
}
```

Get a Taste of Lambdas and Get Addicted to Streams by Venkat Subramaniam

what is completeblfuture

Account.java

```java
package threadingexample;

public class Account {
    private int balance=20000;
    public int getbalance()
    {
        return balance;
    }
    public void withdraw(int val)
    {
        this.balance=this.balance-val;
    }
}
```

*AccountHolder.java

```java
    @Override
    public void run() {
        for (int i = 1; i < 5; i++) {
            makewithdrawble(2000);
            if (account.getbalance() < 0) {
                System.out.println("Already taken........." + Thread.
            }
        }

    }

    private synchronized  void makewithdrawble(int i) { // protect mu
        if (account.getbalance() >= i) {
            System.out.println(
                    Thread.currentThread().getName() + "    " + "Curre
            try {
```

Client.java

```java
package threadingexample;

public class Client {
    public static void main(String[] args) {
        Account account= new Account();
        AccountHolder holder= new AccountHolder(account);
        Thread t1= new Thread(holder);
        Thread t2= new Thread(holder);
        t1.setName("Murari");
        t2.setName("kumar");
        t1.start();
        t2.start();
    }


}
```

If not synhchronized all thread access at same time. If synchronized it will lock thread

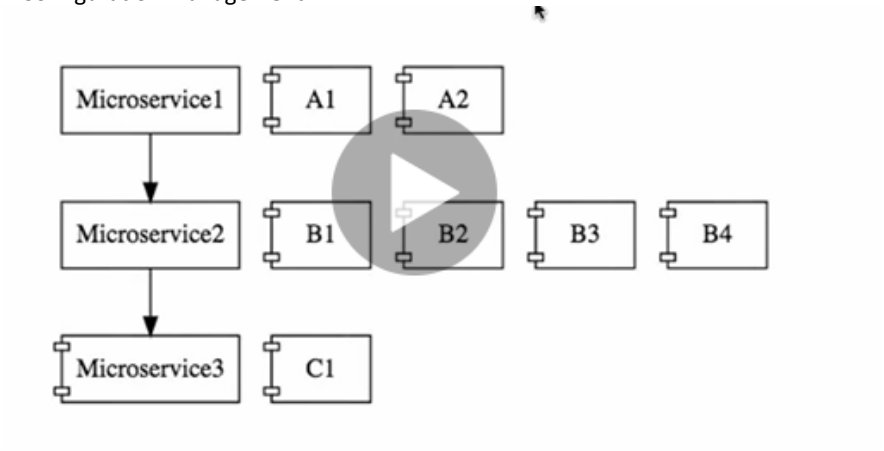
https://www.mockaroo.com/ To mock data

Microservice

1.Bounded Context

Earlier we were building one monolthic app now 10-20 microservicce
How to identify what to do in each of microservices
What to do or not?

2.Configuration Management



5 microservice with different number of instances

3 Dynamic scale up and scale down

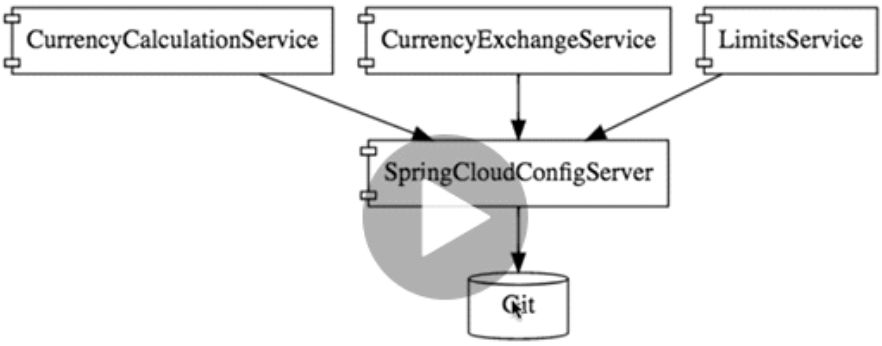
Load different instance at different time
At particular time I want two instance and after that I don't. Bring down when we don't want

4. Visibility

How to identify where is bug. We want centralized and monitor tha logs and server which is up and down

5 Pack of cards
If one microservice fail then all will fail

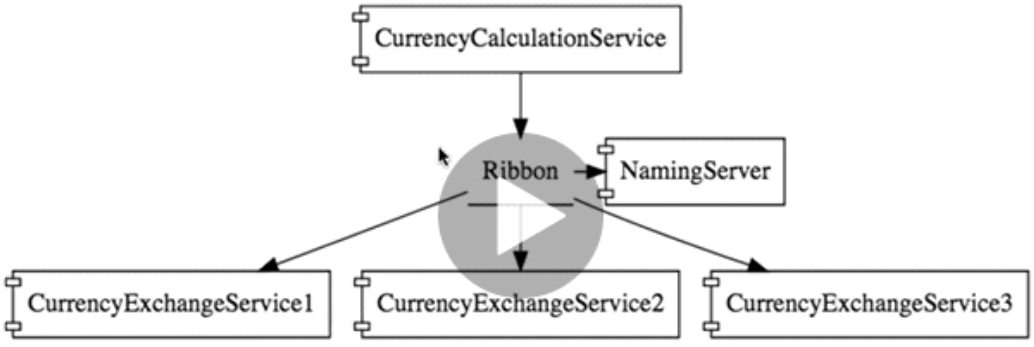
Spring cloud



Spring Cloud Config Server

We can store all the configuration different server and instance to one place…And easy to maintain

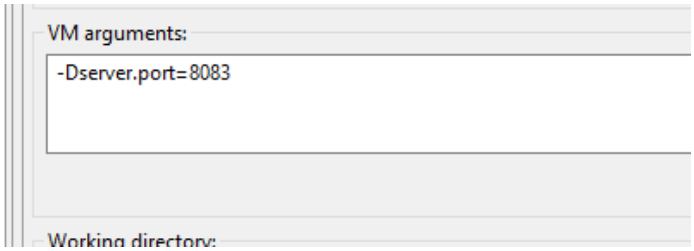
Dynamic scale up and down



Ribbon Load Balancing

Setting property from application properties

```
@Autowired
org.springframework.core.env.Environment env;

@GetMapping("/exchange/from/{from}/to/{to}")
ExchangeValue exchangevalue(@PathVariable String from, @PathVariable String to) {
    ExchangeValue e = new ExchangeValue(1L, from, to, BigDecimal.valueOf(65));
    e.setPort(Integer.parseInt(env.getProperty("server.port")));
    return e;
}
```




```
public interface CurrencyExchangeRepository extends JpaRepository<ExchangeValue, Long> {
    ExchangeValue findByFromAndTo(String from, String to);

}
```

WHEN WE WANT TO FIND BY FROM AND TO String then jpa repo query

Response mapping from other localhost

```
Map<String, String> urivariables = new HashMap<String, String>();
        urivariables.put("from", from);
        urivariables.put("to", to);
        ResponseEntity<ConversionValue> responseEntity =  restTemplate().getForEntity(
                "http://localhost:8081/exchange/from/{from}/to/{to}", ConversionValue.class,
                urivariables);
        ConversionValue body = responseEntity.getBody();
```

Rest template alternative feign

```
@FeignClient(name = "currency-exchange-service", url = "localhost:8081")
public interface CurrencyExchangeServiceProxy {
    @GetMapping("/exchange/from/{from}/to/{to}")
    ConversionValue exchangevalue(@PathVariable("from") String from, @PathVariable("to") String to);
}
```

Post for object
Post for entity
Exchange method of Spring RestTemplate - Part 1 || Calling REST API using RestTemplate

@Get from other api

```
@GetMapping("/conversion-object/from/{from}/to/{to}/quantity/{quantity}")
ConversionValue getvaluemapping(@PathVariable String from, @PathVariable String to) {
    Map<String, String> urivariables = new HashMap<String, String>();
    urivariables.put("from", from);
    urivariables.put("to", to);
    MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
    headers.add("Content-Type", "application/json");
    headers.add("Authorization", "tokenxxx");
    ResponseEntity<ConversionValue> entity = new RestTemplate().exchange(
            "http://localhost:8081/exchange/from/{from}/to/{to}", HttpMethod.GET, new
            HttpEntity<Object>(headers),
```

```
                ConversionValue.class,urivariables);

        return entity.getBody();

}
  @GetMapping("/conversion-object/from/{from}/to/{to}/quantity/{quantity}")
  ConversionValue getvaluemapping(@PathVariable String from, @PathVariable String to) {
      Map<String, String> urivariables = new HashMap<String, String>();
      urivariables.put("from", from);
      urivariables.put("to", to);
      MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
      headers.add("Content-Type", "application/json");
      headers.add("Authorization", "tokenxxx");
      ResponseEntity<ConversionValue> entity = new RestTemplate().exchange(
              "http://localhost:8081/exchange/from/{from}/to/{to}", HttpMethod.GET, new HttpEntity<Object>(headers),
              ConversionValue.class,urivariables);

      return entity.getBody();

  }
```

Post Mapping

```
   @PostMapping("/conversion-object")
   ResponseEntity<ProxyConversion> getvaluemapping(@RequestHeader(value = "Username") String username,
           @RequestHeader(value = "Password") String password, @RequestBody ProxyConversion conversion) {
       MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
       headers.add("Username", username);
       headers.add("Password", password);
       ProxyConversion proxy = new ProxyConversion(11L, "ABC", "DEB", BigDecimal.valueOf(23));
       HttpEntity<Object> httpEntity = new HttpEntity<Object>(proxy, headers);
       adduserexchangeforpost(httpEntity);

       return adduserexchangeforpost(httpEntity);
   }

   private ResponseEntity<ProxyConversion> adduserexchangeforpost(HttpEntity<Object> httpEntity) {
       ResponseEntity<ProxyConversion> exchange = new RestTemplate().exchange("http://localhost:8081/exchange",
               HttpMethod.POST, httpEntity, ProxyConversion.class);
       return exchange;

   }


  ResponseEntity<String> getvaluemappingpost() {
      MultiValueMap<String, String> headers = new LinkedMultiValueMap<>();
      headers.set("Content-Type", "application/json");
      ProxyConversion proxy = new ProxyConversion(11L, "ABC", "DEB", BigDecimal.valueOf(23));
      HttpEntity<Object> httpEntity = new HttpEntity<Object>(proxy, headers);
      adduserexchangeforpost(httpEntity);

      return adduserexchangeforpost(httpEntity);
  }

  private ResponseEntity<String> adduserexchangeforpost(HttpEntity<Object> httpEntity) {
      ResponseEntity<String> exchange = new RestTemplate().exchange("http://localhost:8081/exchange",
              HttpMethod.POST, httpEntity, String.class);
      return exchange;

  }
```

Pagebale concept

```java
@Service
public class UserServiceImpl implements UserService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public List<UserDto> getUsers(int page, int limit) {
        List<UserDto> returnValue = new ArrayList<>();

        Pageable pageableRequest = PageRequest.of(page, limit);
        Page<UserEntity> users = userRepository.findAll(pageableRequest);
        List<UserEntity> userEntities = users.getContent();

        for (UserEntity userEntity : userEntities) {
            UserDto userDto = new UserDto();
            BeanUtils.copyProperties(userEntity, userDto);
            returnValue.add(userDto);
        }

        return returnValue;
    }

}
```

https://www.javainuse.com/spring/SpringBootUsingPagination

```java
@RequestParam(value = "page", defaultValue = "0") int page
```

ad the limit request parameter our method above has the **limit** method argument:

```java
@RequestParam(value = "limit", defaultValue = "30") int limit
```

21

Apart from these mentioned differences in framework, one major difference is `@RequestParam` will always expect a value to bind. Hence, if value is not passed, it will give error. This is not the case in `@QueryParam`

`Query param if we have already the value then we search using query param`

From <https://stackoverflow.com/questions/26709560/what-is-the-difference-b-w-requestparam-and-queryparam-anotation>

Redis server

```java
@SpringBootApplication
public class CurrencyConversionApplication {

    @Bean
    JedisConnectionFactory factory() {
        return new JedisConnectionFactory();
    }

    @Bean
    RedisTemplate<String, User> getredis() {
        RedisTemplate<String, User> redis = new RedisTemplate<String, User>();
        redis.setConnectionFactory(factory());
        return redis;
    }

    public static void main(String[] args) {
        SpringApplication.run(CurrencyConversionApplication.class, args);
    }

}
```

Hashoperation we cannot use reddis server directly we need to use via hashoperation

```java
@Repository
public class UserRepositoryImpl implements UserRepository {

    private RedisTemplate<String, User> redisTemplate;

    private HashOperations hashOperations;


    public UserRepositoryImpl(RedisTemplate<String, User> redisTemplate) {
        this.redisTemplate = redisTemplate;

        hashOperations = redisTemplate.opsForHash();
    }

    @Override
    public void save(User user) {
        hashOperations.put("USER", user.getId(), user);
    }

    @Override
    public Map<String, User> findAll() {
        return hashOperations.entries("USER");
    }

    @Override
    public User findById(String id) {
        return (User)hashOperations.get("USER", id);
    }

    @Override
    public void update(User user) {
        save(user);
    }
```

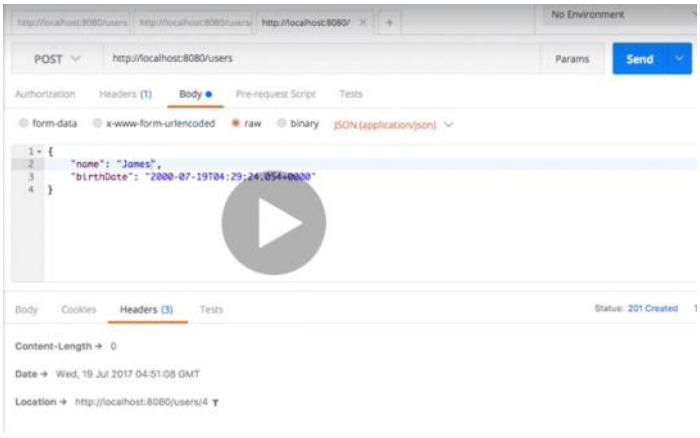
Monday, December 21, 2020 2:33 AM

Reprsentaional state transfer
Maximum use of rest

HTTP ->if we request anything on web browser then it will get the response from server and display in the form of html with all output. It conatins body and header
Swagger->service defination

Post

```
@PostMapping("/users")
ResponseEntity<Object> addvalue(@RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri(); // add /users/id for new id and
    //return 201 created status while posting

    return ResponseEntity.created(uri).build();
}
```



New location url also given in header when we are posting with above code

Exception

```
public class UserNotFoundException extends RuntimeException {

    public UserNotFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }

}
```

```
@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);
    if(user==null)
        throw new UserNotFoundException("id-"+ id);

    return user;
}
```

```
@ResponseStatus(HttpStatus.NOT_FOUND)
public class UserNotFoundException extends RuntimeExceptior.
    public UserNotFoundException(String message) {
        super(message);
    }
}
```

```
ExceptionResponse exceptionResponse = new ExceptionResponse(new Date(),
        ex.getBindingResult().toString());
```

It will add the exception with proper message

@Valid to request body so that proper format has been added or posted

```
@PostMapping("/users")
ResponseEntity<Object> addvalue(@Valid @RequestBody User user) {
    User u = userconfig.addvalue(user);
    URI uri = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(u.getId()).toUri

    // return 201 created status while posting
    return ResponseEntity.created(uri).build();
}
```

```
@Component
public class User {

    Integer id;
    @Size(min =2)
    String name;
    String address;
```

Hateos

To provide link in the response

```
if(user==null)
    throw new UserNotFoundException("id-"+ id);


//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());


//HATEOAS

return user;
```

```
{
    "id": 1,
    "name": "Adam",
    "birthDate": "2017-07-19T09:26:18.337+0000",
    "_links": {
        "all-users": {
            "href": "http://localhost:8080/users"
        }
    }
}
```

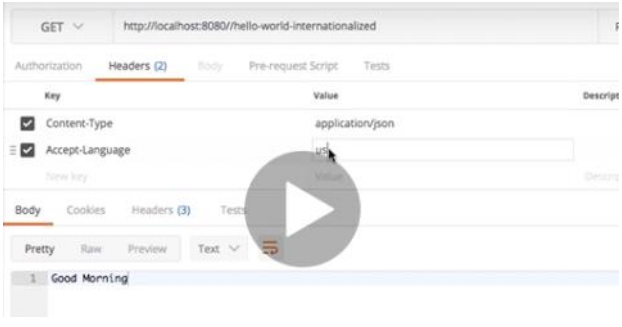
```
//"all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

ControllerLinkBuilder linkTo =
        linkTo(methodOn(this.getClass()).retrieveAllUsers());

resource.add(linkTo.withRel("all-users"));

//HATEOAS
```

```
#### Internationalization

##### Configuration
- LocaleResolver
   - Default Locale - Locale.US
- ResourceBundleMessageSource

##### Usage
- Autowire MessageSource
- @RequestHeader(value = "Accept-Language", required = false) Locale locale
- messageSource.getMessage("helloWorld.message", null, locale)
```

// default local=us

We can pass as @Request Header accept-language

```
@Autowired
MessageSource messagesource;

@GetMapping("/hellovalue")
String message(@RequestHeader(name="Accept-Language", required = false) Locale locale) {  //
    // as locale needs to be passed using header on based accept-language we need to pass locale in header
    return messagesource.getMessage("good.morning.message", null, locale);
}

@Bean
ResourceBundleMessageSource source() {
    ResourceBundleMessageSource source = new ResourceBundleMessageSource();
    source.setBasename("messages");
    return source;
}
```




```
@Bean
public LocaleResolver localeResolver() {
    AcceptHeaderLocaleResolver localeResolver = new AcceptHeaderLocaleResolver();

    localeResolver.setDefaultLocale(Locale.US);

    return localeResolver;
}
```

Swagger->

```java
@Configuration
@EnableSwagger2
public class Swagger {

    @Bean
    Docket docket()
    {
        return new Docket(DocumentationType.SWAGGER_12);
    }

}
```

```
{
    swagger: "2.0",
  + info: {…},
    host: "localhost:8080",
    basePath: "/",
  + tags: […],
  - paths: {
      + /error: {…},
      + /hello-world: {…},
      + /hello-world-bean: {…},
      + /hello-world-internationalized: {…},
      + /hello-world/path-variable/{name}: {…},
      + /users: {…},
      + /users/{id}: {…}
    },
  + definitions: {…}
}
```

```java
@ApiModel(description="All details about the user. ")
public class User {

    private Integer id;

    @Size(min=2, message="Name should have atleast 2 charac
    @ApiModelProperty(notes="Name should have atleast 2 cha
    private String name;

    @Past
    @ApiModelProperty(notes="Birth date should be in the past")
    private Date birthDate;

    protected User() {

    }
```

Filter out some of the contents and don't want to show much to end user

```java
public class SomeBean {

    private String field1;

    private String field2;

    @JsonIgnore
    private String field3;

@JsonIgnoreProperties(value={"field1","field2"})
public class SomeBean {
```

Dynamic filter

```java
@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1
```

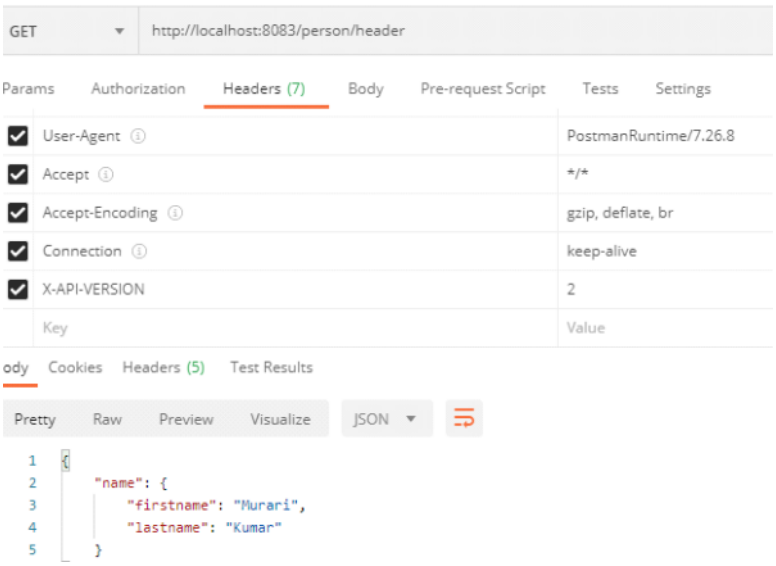
```java
@GetMapping("/filtering")
public MappingJacksonValue retrieveSomeBean(){
    SomeBean someBean = new SomeBean("value1","value2","value3");

    SimpleBeanPropertyFilter filter = SimpleBeanPropertyFilter.
            filterOutAllExcept("field1","field2");

    FilterProvider filters = new SimpleFilterProvider().addFilter("SomeBeanFilt

    MappingJacksonValue mapping = new MappingJacksonValue(someBean);

    mapping.setFilters(filters);

    return mapping;
```

```java
@RestController
public class VersioningController {

    @GetMapping("v1/person")  // url versioning
    PersonV1 personev1()
    {
        return new PersonV1("Murari Kumar");
    }

    @GetMapping("v2/person")
    PersonV2 personev2()
    {
        return new PersonV2(new Name("Murari", "Kumar"));
    }

}
```

Versioning using request param

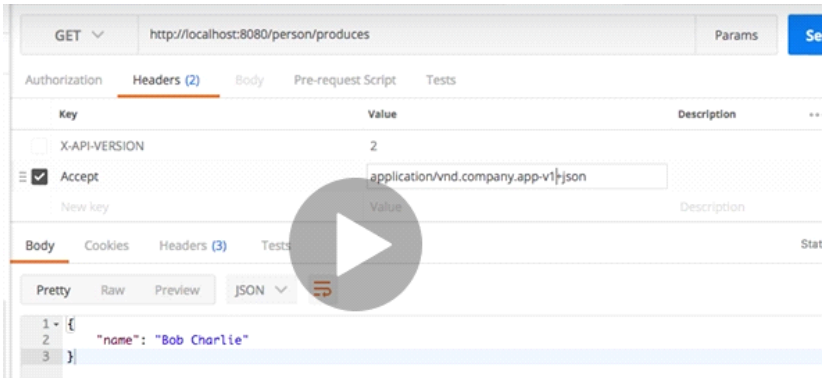
```java
@GetMapping(value = "v1/person", params = "version=1") // versioning using request param
PersonV1 paramv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "v2/person", params = "version=2")
PersonV2 paramv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```

```java
@GetMapping(value = "/person/header", headers = "X-API-VERSION=1") // versioning using request header
PersonV1 headerv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/header", headers = "X-API-VERSION=2")
PersonV2 headerv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Versioning using produces

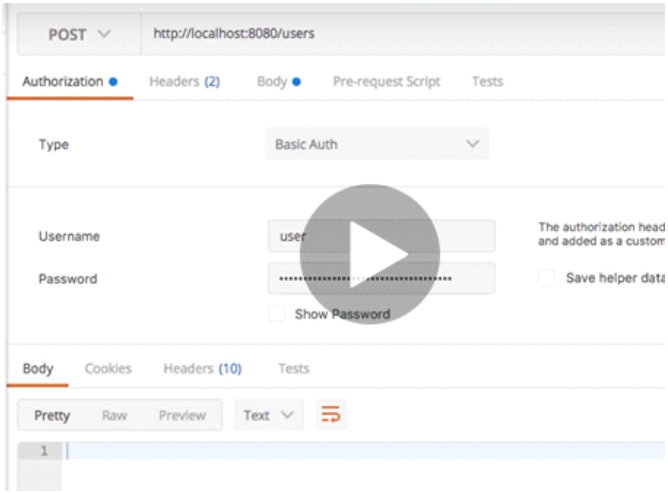
```
@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v1+json") // versioning using produces
PersonV1 producesv1() {
    return new PersonV1("Murari Kumar");
}

@GetMapping(value = "/person/produces", produces = "application/vnd.company.app-v2+json")
PersonV2 producesv2() {
    return new PersonV2(new Name("Murari", "Kumar"));
}
```



Header request problem in caching difficult // cannot use from browser// swagger and documentation difficulty

Still cache using uri

Security password sending in authorization



H2 DATABASES FECTHING DATA

```
@Entity
public class UserDTO {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    Long phone;
```

```
server.port=8083
spring.jpa.show-sql=true
 #this start logging jpa
spring.h2.console.enabled=true
```

```
@RestController
public class UserJPAResource {

    @Autowired
    UserRepositroy user;

    @GetMapping("/findall")
    List<User> findAll() {
        return user.findAll();
    }

    @GetMapping("/findId/{id}")
    Optional<User> findById(@PathVariable int id
        return user.findById(id);
    }

    @PostMapping("/users")
    void addUser(@RequestBody User u) {
        user.save(u);
    }

    @DeleteMapping("/users/{id}")
    void deleteuser(@PathVariable int id) {
        user.deleteById(id);
```

```
@Repository
public interface UserRepositroy  extends JpaRepository<User, Integer>{

}
```

Many to one relationship

User can have many post
Post will have same user id for all post

```java
@Entity
public class Post {

    @Id
    @GeneratedValue
    private Integer id;
    private String description;

    @ManyToOne(fetch=FetchType.LAZY)
    @JsonIgnore
    private User user;

    public Integer getId() {
        return id;
    }
```

To get all his post

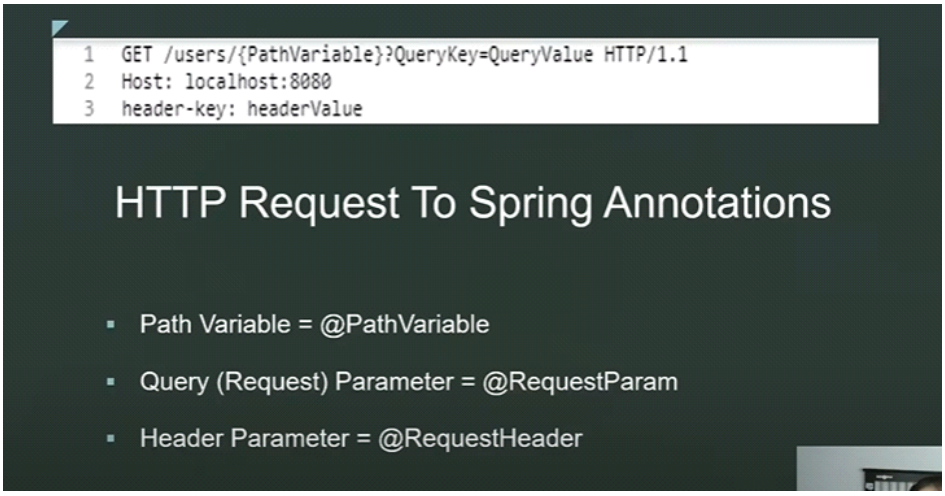
```java
@Entity
public class User {

    @Id
    @GeneratedValue
    int id;
    String name;
    String address;
    int phone;
    @OneToMany(mappedBy = "user")
    List<Post> post;

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

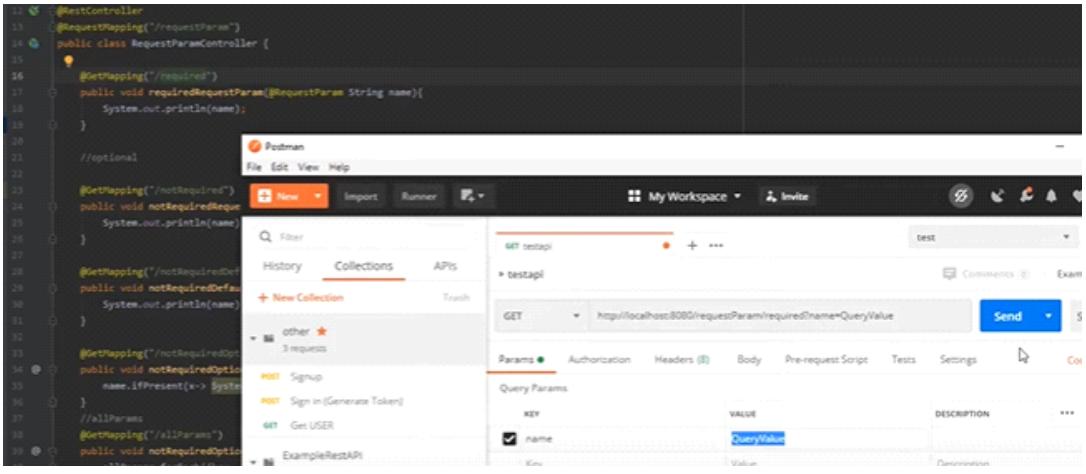
```java
@GetMapping("/jpa/users/{id}/posts")
public List<Post> retrieveAllUsers(@PathVariable int id) {
    Optional<User> userOptional = userRepository.findById(id);

    if(!userOptional.isPresent()) {
        throw new UserNotFoundException("id-" + id);
    }

    return userOptional.get().getPosts();
}
```

```java
@Entity
public class Post {
    @Id
    @GeneratedValue
    int id;
    String descrription;
    @ManyToOne(fetch = FetchType.LAZY)
    @JsonIgnore  // infinite loop as both will start calling each other so we put ignore
    User user;

    public int getId() {
```

```
@PostMapping("/jpa/users/{id}/posts")
public ResponseEntity<Object> createUser(@PathVariable int id, @RequestBody Post post)
{
Optional<User> userOptional= userRepository.findById(id);
if(!userOptional.isPresent())
{
throw new UserNotFoundException("id: "+ id);
}
User user=userOptional.get();
//map the user to the post
post.setUser(user);
//save post to the database
postRepository.save(post);
//getting the path of the post and append id of the post to the URI
URI location=ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(post.getId()).toUri();
//returns the location of the created post
return ResponseEntity.created(location).build();
}
```

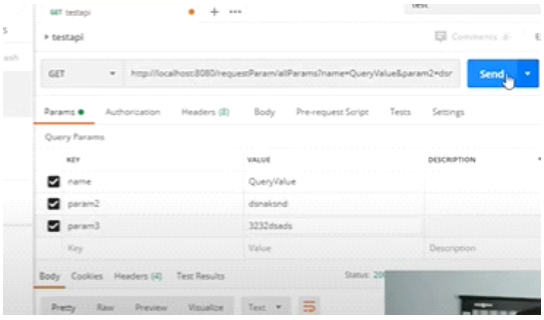
When posting post from postman while in list post from user

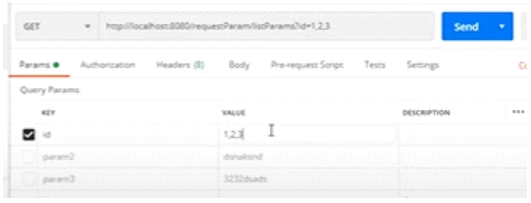




Query parammater using map @RequestParam

```
//allParams
@GetMapping("/allParams")
public void notRequiredOptional(@RequestParam Map<String,String> allParams ){
    allParams.forEach((key, value) -> {
        System.out.println(String.format("Request Param '%s' = %s", key, value));
    });
}

@GetMapping("/listParams")
public void paramList(@RequestParam List<Integer> id ){
    id.forEach(eachId-> System.out.println(eachId));
}
```





List param

When we want parse api json

```
@GetMapping("/restore")
List<BitCoin> storevalue() {
    RestTemplate rest = new RestTemplate();
    ResponseEntity<List<BitCoin>> rateResponse =
            rest.exchange("https://bitpay.com/api/rates",
                    HttpMethod.GET, null, new ParameterizedTypeReference<List<BitCoin>>() {
                });

    return reposiotry.save(rateResponse.getBody());
```

```
@SpringBootApplication
//@EnableDiscoveryClient
@ComponentScan(basePackages = "com.*")
@EntityScan("com.*")
public class CurrencyExchangeServiceApplication {
```

https://www.foreach.be/blog/spring-cache-annotations-some-tips-tricks

Spring cache to search using ehcache
Spring Cache Example using EhCache in Spring Boot | Tech Primers

```
<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:noNamespaceSchemaLocation="ehcache.xsd"
         updateCheck="true"
         monitoring="autodetect"
         dynamicConfig="true">

    <diskStore path="java.io.tmpdir" />

    <cache name="usersCache"
           maxEntriesLocalHeap="10000"
           maxEntriesLocalDisk="1000"
           eternal="false"
           diskSpoolBufferSizeMB="20"
           timeToIdleSeconds="300" timeToLiveSeconds="600"
           memoryStoreEvictionPolicy="LFU"
           transactionalMode="off">
        <persistence strategy="localTempSwap" />
    </cache>
```

```java
@EnableJpaRepositories(basePackages = { "com.techprimers.cache.repository", "com.techprimers.cache" })
@EnableCaching
@Configuration
public class EhCacheConfiguration {

    @Bean
    public CacheManager cacheManager() {
        return new EhCacheCacheManager(cacheMangerFactory().getObject());
    }

    @Bean
    public EhCacheManagerFactoryBean cacheMangerFactory() {
        EhCacheManagerFactoryBean bean = new EhCacheManagerFactoryBean();
        bean.setConfigLocation(new ClassPathResource("ehcache.xml"));
        bean.setShared(true);
        return bean;
    }
}
```

```java
@Component
public class UsersCache {

    @Autowired
    UsersRepository usersRepository;

    @Cacheable(value = "usersCache", key = "#name")
    public Users getUser(String name) {
        System.out.println("Retrieving from Database for name: " + name);
        return usersRepository.findByName(name);
    }
}
```

```
timeToIdleSeconds="300"
timeToLiveSeconds="600">
```

If it has not been used for 300 seconds
Or after stays in cache for 600 seconds
When echache exceed from its configured size ehcache removed expired elements
If that doesn't clear enough space it cleasr object from ehcache
By default it remove LRU elements

Second level caching is we put on entity or whole object

```java
@Configuration
@EnableCaching
public class CacheConfig {

    @Bean
    public CacheManager cacheManager() {
        SimpleCacheManager cacheManager = new SimpleCacheManager();
        List<Cache> cacheList = new ArrayList<Cache>();
        cacheList.add(new ConcurrentMapCache("userCache"));
        cacheList.add(new ConcurrentMapCache("addressCache"));
        cacheManager.setCaches(cacheList);
        return cacheManager;
    }
}
```

If we don’t want to configure in ehache.xml

Evict cache -> If we now want to clear cache just evict it

```
@RequestMapping(value = "/greetings")
@Cacheable("greetings")
public Collection<Greeting> getGreetings() throws InterruptedException {
    Thread.sleep(5000);
    return greetingMap.values();
}

@RequestMapping(value = "/greetings", method=RequestMethod.POST)
public String createGreeting(@RequestBody Greeting g) {
    greetingMap.put(g.getId(), g);
    return "Greeting is saved successfully";
}

@RequestMapping(value="/evictcache")
public String evictCache() {
    return "Cache is cleared successfully";
}
```

```
@RestController
 public class CachingController {
@Autowired
 CachingService cachingService;
@GetMapping("clearAllCaches")
public void clearAllCaches() {
 cachingService.evictAllCaches(); } }
```

From <https://www.baeldung.com/spring-boot-evict-cache>

3.4. Sending HTTP Headers using RestTemplate

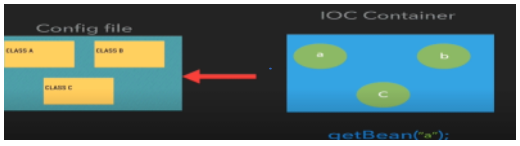
```
private static void getEmployees()
{
    final String uri = "http://localhost:8080/springrestexample/employees";
    RestTemplate restTemplate = new RestTemplate();

    HttpHeaders headers = new HttpHeaders();
    headers.setAccept(Arrays.asList(MediaType.APPLICATION_JSON));
    headers.set("X-COM-PERSIST", "NO");
    headers.set("X-COM-LOCATION", "USA");

    HttpEntity<String> entity = new HttpEntity<String>(headers);

    ResponseEntity<String> response = restTemplate.exchange(uri, HttpMethod.GET, entity, String.class);

    //Use the response.getBody()
}
```

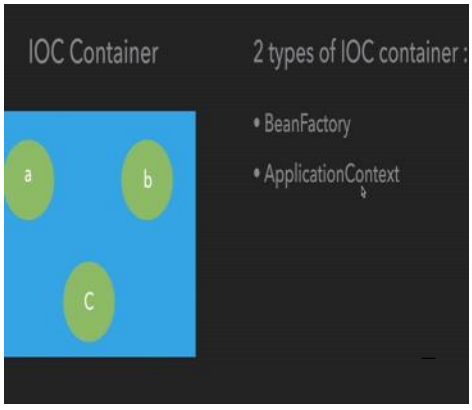
3.5. Sending URL Parameters using RestTemplate

```
private static void getEmployeeById()
{
    final String uri = "http://localhost:8080/springrestexample/employees/{id}";
    RestTemplate restTemplate = new RestTemplate();

    Map<String, String> params = new HashMap<String, String>();
    params.put("id", "1");

    EmployeeVO result = restTemplate.getForObject(uri, EmployeeVO.class, params);

    //Use the result
}
```

From <https://howtodoinjava.com/spring-boot2/resttemplate/spring-restful-client-resttemplate-example/>

ControllerAdvice






```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">



<bean id = "airtel" class = "com.selniumexpress.ioc.Airtel"></bean>
```

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("config loaded");
        Vodaphone voda = context.getBean("vodaphone", Vodaphone.class);
        voda.calling();
        voda.data();

    }

}
```

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("config loaded");
        Airtel air = (Airtel)context.getBean("airtel");
        air.calling();
        air.data();

    }

}
```

Constructor loaded of airtel

```java
public class Mobile {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml")
        System.out.println("config loaded");

        Sim sim = context.getBean("sim",Sim.class);
        sim.calling();
        sim.data();

    }

}
```

Without touching vodafone and airtel

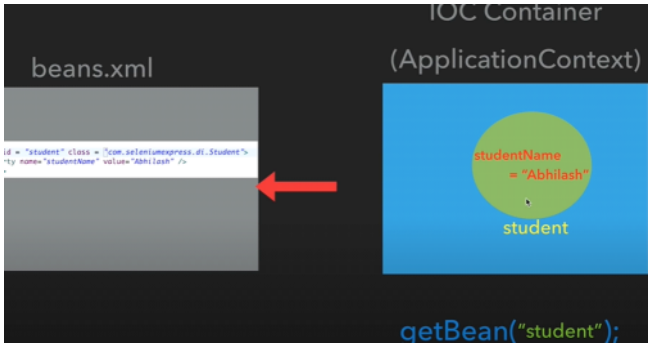
```xml
<?xml version="1.0" encoding="UTF-8"?>

<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">

<bean id = "sim" class = "com.selniumexpress.ioc.Jio"></bean>


</beans>
```

Control is taken by framework now they are creating object


```java
public class Exam {

    public static void main(String[] args) {

        Student student = new Student();
        student.setStudentName("Abhilash Panigrahi");

        student.displayStudentInfo();
    }
```

```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Abhilash Panigrahi" />
</bean>
```

Setter need to be added for student name




```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Abhilash Panigrahi" />
  <property name="id" value="1"></property>
</bean>

<bean id = "ashish" class="com.seleniumexpress.di.Student">
  <property name="studentName" value="Ashish B" />
  <property name="id" value="2"></property>
</bean>
```

```java
ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
Student abhi = context.getBean("student", Student.class);
abhi.displayStudentInfo();

Student ashish = context.getBean("ashish" , Student.class);
ashish.displayStudentInfo();
```

Constructor Dependcy

```java
public class Student {

    private int id;
    private String studentName;

    public Student(int id, String studentName) {
        super();
        this.id = id;
        this.studentName = studentName;
    }
```

New Constructor added so we need t0 make new bean id
And spring will convert id to int automatically explicitly by adding type="id"

```xml
<bean id = "student" class="com.seleniumexpress.di.Student">
  <constructor-arg name="studentName" value="Abhilash Panigrahi" />
  <constructor-arg name="id" value="1" />
</bean>
```

```
<bean id = "student" class="com.seleniumexpress.di.Student">
  <constructor-arg name="studentName" value="Abhilash Panigrahi" />
  <constructor-arg name="id" value="1" />
</bean>

<bean id = "dilip" class="com.seleniumexpress.di.Student">
  <constructor-arg name="id" value="1" />
</bean>
```

Object injecting

```
package com.seleniumexpress.di;

public class MathCheat {

    public void mathCheat()
    {
        System.out.println("math ceating started..");
    }

}
```

Not a good programming.

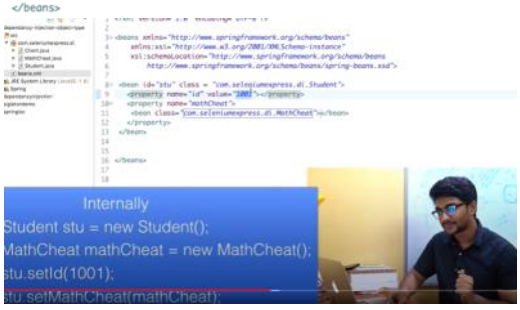
```
public class Student {

    MathCheat mathCheat = new MathCheat();

    public void cheating()
    {
        mathCheat.mathCheat();
    }
}
```

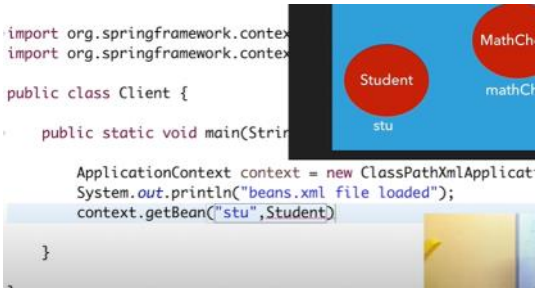
```
public class Student {

    MathCheat mathCheat;


    public void setMathCheat(MathCheat mathCheat) {
        this.mathCheat = mathCheat;
    }

    public void cheating()
    {
        mathCheat.mathCheat();
    }

}
```

S

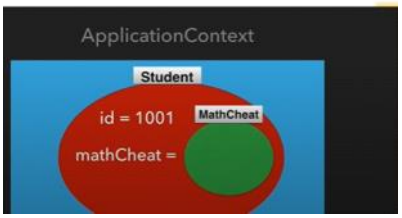
```
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat">
    <bean class="com.seleniumexpress.di.MathCheat"></bean>
  </property>
</bean>
```

```
</beans>
```

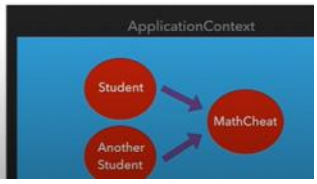




```xml
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat">
   <bean class="com.seleniumexpress.di.MathCheat"></bean>
  </property>
</bean>
```




```xml
<bean id = "mathCheatObjectValue" class="com.seleniumexpress.di.MathCheat"></bean>

<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="id" value="1001"></property>
  <property name="mathCheat" ref="mathCheatObjectValue" />
</bean>

<bean id = "anotherStudent" class="com.seleniumexpress.di.AnotherStudent">
<property name="cheat" ref="mathCheatObjectValue" />
</bean>

</beans>
```



Loose coupling

```java
package com.seleniumexpress.di;

public class Student {

    private Cheat cheat;

    public void setCheat(Cheat cheat) {
        this.cheat = cheat;
    }

    public void cheating() {

        cheat.cheat();
    }

}
```

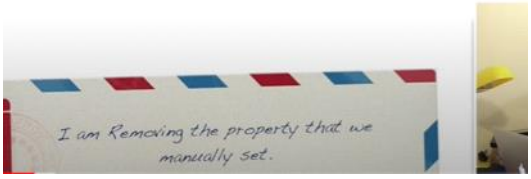
```java
public interface Cheat {

    public void cheat();

}
```

```xml
<bean id = "mathCheatObjectValue" class="com.seleniumexpress.di.MathCheat"></bean>
<bean id = "scienceCheatObjectValue" class="com.seleniumexpress.di.ScienceCheat"></bean>
<bean id="stu" class = "com.seleniumexpress.di.Student">
  <property name="cheat" ref="scienceCheatObjectValue" />
</bean>
</beans>
```

```java
public class Client {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("beans.xml file loaded");
        Student student = context.getBean("stu",Student.class);
        student.cheating();

    }
```

Autowired.....

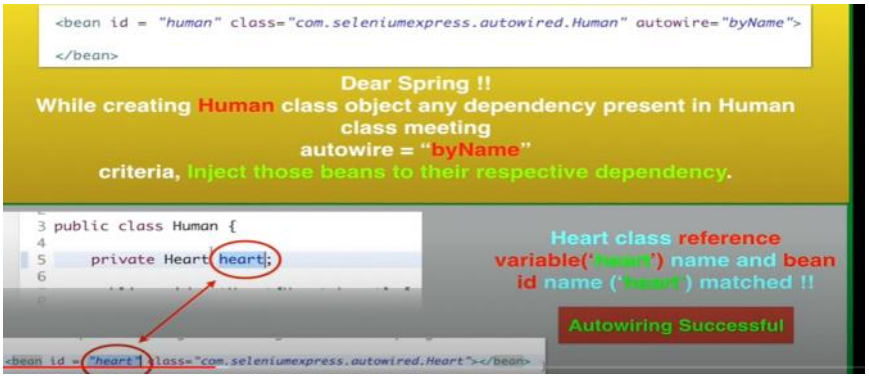

```
<bean id = "heartObject" class="com.seleniumexpress.autowired.Heart"></bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human">

<property name="heart" ref = "heartObject" />
</bean>

</beans>
```



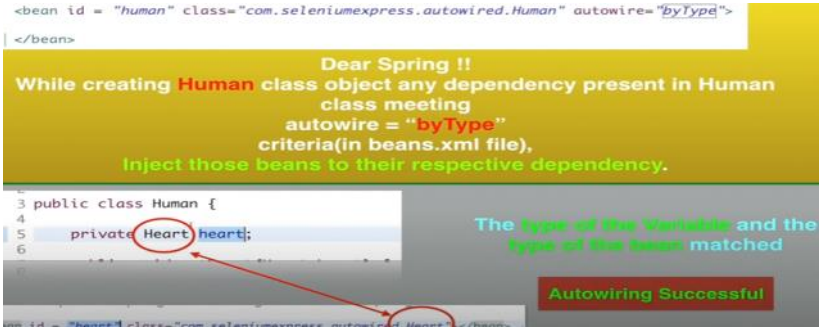
By Name autowired

```
private Heart heart;

public void setHeart(Heart heart) {
    this.heart = heart;
}
```

```
<bean id = "heart" class="com.seleniumexpress.autowired.Heart"></bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human" autowire="byName">
```



By type autowired niche wala



Autowire constructor

```java
public class Human {

    private Heart heart;

    @Autowired
    public Human(Heart heart) {

        this.heart = heart;
    }

    public void setHeart(Heart heart) {
        this.heart = heart;
        System.out.println("setter method
```

works same as autowire = "byConstructor"

```
<bean id = "human" class="com.seleniumexpress.autowired.Human" >

</bean>

</beans>
```

Human human = new Human();
nan() is a default constructor. Do we have it in our class?

```java
import org.springframework.beans.factory.annotat

public class Human {

    private Heart heart;

    public Human() {

    }

    @Autowired
    public Human(Heart heart) {

        this.heart = heart;
    }

    public void setHeart(Heart heart) {
        this.heart = heart;
        System.out.println("setter method called");
    }
```

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

  <con

<bean id = "heartObjectValue" class="com.seleniumexpress.autowired.Heart"></bean>

 <bean id = "human" class="com.seleniumexpress.autowired.Human" >

 </bean>

/beans>
```

You need to turn on <context:annotaion-config/>
n order to use annotaions in spring. (eg: to use @Autowired)

```java
ublic class Human {

   private Heart heart;

   public Human() {

   }


   public Human(Heart heart) {

       this.heart = heart;
       System.out.println("human constr is getting called which has Heart as arg");
   }

   @Autowired
   public void setHeart(Heart heart) {
```

How @Autowired works?

1. first it try to resolves with “byType”.
2. If byType fails then it goes with “byName”

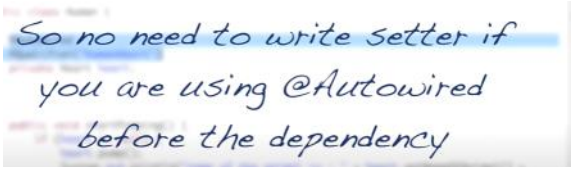
```
bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" />
bean id = "octpousHeart" class="com.seleniumexpress.autowired.Heart" />
```

Unique bean

Qualifier Two heart implmenetation

```
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <context:annotation-config />

 <bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" >
  <property name="nameOfAnimal" value="Human" />
  <property name="noOfHeart" value="1" />
 </bean>

 <bean id = "octpousHeart" class="com.seleniumexpress.autowired.Heart" >
   <property name="nameOfAnimal" value="Octpous" />
  <property name="noOfHeart" value="3" />
 </bean>

  <bean id = "human" class="com.seleniumexpress.autowired.Human" >
```

```
@Autowired
@Qualifier("humanHeart")
public void setHeart(Heart heart) {
    this.heart = heart;
    System.out.println("setter method called");
}
```



Bean property to read from property fiel
Student.name for student
Teacher.name for teacher

```
student.name = Abhilash
student.intrestedCourse = Java
shobby = cricket
```

```
<bean id = "student" class="com.seleniumexpress.loadingfrompropertiesfile.S
  <property name="name" value = "${student.name}"/>
  <property name="intrestedCourse" value = "${student.intrestedCourse}"/>
  <property name="hobby" value = "${student.hobby}"/>
</bean>
```

```
<context:property-placeholder/>
```

To know spring I am using from property file

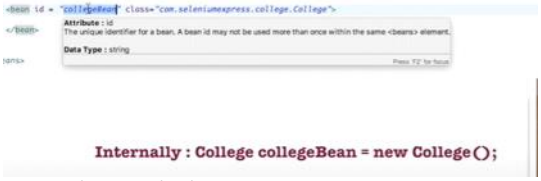
@Value annotation remove properyy


```
public class Student {

    private String name;
    private String intrestedCourse;
    private String hobby;

    @Value("Abhilash")
    public void setName(String name) {
        this.name = name;
    }

    @Value("Java")
    public void setIntrestedCourse(String intre
        this.intrestedCourse = intrestedCourse;
    }
    @Value("cricket")
    public void setHobby(String hobby) {
        this.hobby = hobby;
    }

    public void dispalyStudentInfo()
```

Restricted required to mandatory

We don't require setter method if we are putting @Value or autowired

Beans file is configuration file



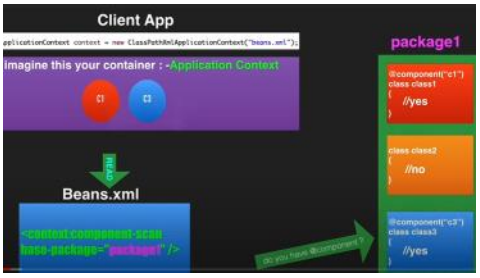
@Componet does same work as above

@Componet we neeed to write instead bean.xml to configure bean

@Componet does same thing 1 create bean
2 and register to IOC container

```
@Component("collegeBean")
public class College {

}
```

```
@Component("collegebean")
public class College {

    @Autowired   // for class type
    Student student;

    @Value("Murari")   // for property like string
    String name;

    void dis() {
        student.displa(name);

    }

}
```



@Configuration if we want to completely remove xml

Just below we need to write configuration scan to scan all the component

```
@Configuration
@ComponentScan(basePackages = "com.college")
public class CollegeConfig {

}
```

Now spring will read this file instead xml file

```
public class Application {
    public static void main(String[] args) {
        ApplicationContext ap= new AnnotationConfigApplicationContext(CollegeConfig.class);
        System.out.println("beans.loaded");
        University c=ap.getBean("collegebean", University.class);
        c.display();
    }

}
```

Instead @Componet we can remove @Component and use @Bean in configuration file

METHOD NAME of bean will beanId-> collegeBean

```
Configuration
ComponentScan(basePackages = "com.college")
ublic class CollegeConfig {

    @Bean
    public College collegeBean()
    {
        return new College();
    }
```

```
Configuration
ComponentScan(basePackages = "com.college")
ublic class CollegeConfig {

    @Bean
    public University university()
    {
        return new College(student());
    }

    @Bean
    public Student student()
    {
        return new Student();
    }

}
```

```
ublic class College implements University {

    Student student;

    public College(Student student) {
        super();
        this.student = student;
    }

    public College() {

    }

    @Value("Murari") // for property like string
    String name;

    public void display() {
        student.displa(name, "college");

    }

}
```

Constructor injection

Normal injection like string int using **@Value**
Class object using autowired or bean

For setter injection

```
@Configuration
public class CollegeConfig {

    @Bean
    public Principal principalBean()
    {
        return new Principal();
    }

    @Bean
    public College collegeBean() // collegeBean - bea
    {
        College college = new College();
        college.setPrincipal(principalBean());
        return college;
    }
```

```
public class College {

    private Principal principal;

    /*public College(Principal principal) {
        this.principal = principal;
    }*/

    public void setPrincipal(Principal principal) {
        this.principal = principal;
        System.out.println();
    }
```

From property injection down

```
@Component
public class College {

    @Value("${college.Name}")
    private String collegeName;

    @Autowired
    private Principal principal;

    @Autowired
    private Teacher teacher ;

    public void test()
    {
        principal.prinipalInfo();
```

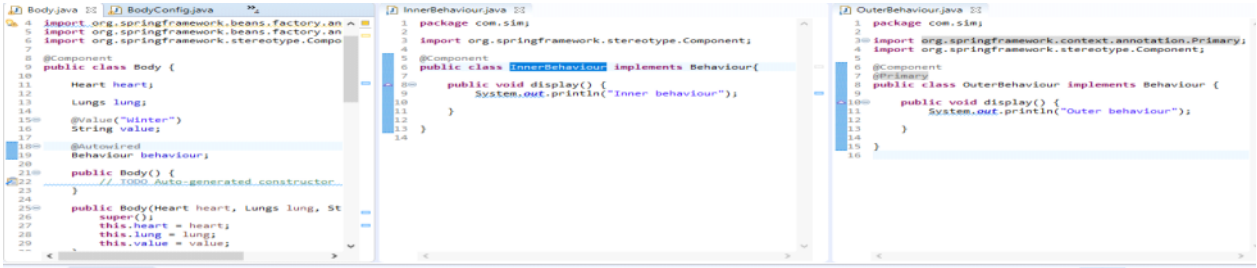
```
@Configuration
@ComponentScan(basePackages = "com.seleniumexpress.college")
@PropertySource("classpath:college-info.proerties")
public class CollegeConfig {
/*
    @Bean
    public Teacher mathTeacherBean()
    {
        return new MathTeacher();
    }
```

@Qualifier
100 of implementation if we want to use one then we use @Qualifier

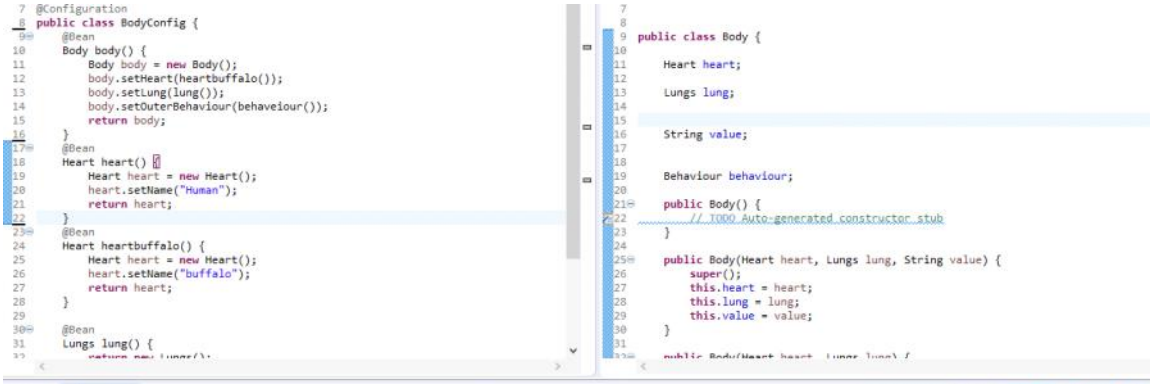
If I want an interface one implementation to be primary used

```
@Component
@Primary
public class MathTeacher implements Teacher {

    @Override
    public void teach() {

        System.out.println("Hi I am your math teacher");
        System.out.println("My name is Sourav");

    }

}
```

```
Heart is not functioning
Lung not functioning
Inner behaviour
```



```
Heart is not functioning
Lung not functioning
Outer behaviour
```



```
buffalo is functioning withnull
humanLungs is functioning withnull
Inner behaviour
```

```java
@Configuration
public class BodyConfig {
    @Bean
    Body body() {
        Body body = new Body();
        body.setHeart(heartbuffalo());
        body.setLung(lung());
        body.setOuterBehaviour(behaveiour());
        return body;
    }

    @Bean
    Behaviour behaveiour() {
        return new InnerBehaviour();
    }
    @Bean
    Heart heart() {
        Heart heart = new Heart();
        heart.setName("Human");
        return heart;
    }
    @Bean
    Heart heartbuffalo() {
        Heart heart = new Heart();
        heart.setName("buffalo");
        return heart;
    }

    @Bean
    Lungs lung() {
        return new Lungs();
```

```java
@Component
public class College {

    //@Value("${college.Name}")
    private String collegeName;
    @Required
    public void setCollegeName(String collegeName
        this.collegeName = collegeName;
    }
```

@Required if we don't want to be pass null in string only used on setter
Required bean

Most Important topic of spring……

Life cycleof bean

```java
    public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

        // load driver
        Class.forName(driver);

        // get a connection
        con = DriverManager.getConnection(url, userName, password);

    }

    public void selectAllRows() throws ClassNotFoundException, SQLException {
        System.out.println("Retriving all students data..");

        // execute query
        Statement stmt = con.createStatement();

        ResultSet rs = stmt.executeQuery("SELECT * FROM ESNew.HostelStudentInfo");

        while (rs.next()) {
            int studentId = rs.getInt(1);
            String studentName = rs.getString(2);
            double hostelFees = rs.getDouble(3);
            String foodType = rs.getString(4);

            System.out.println(studentId + " " + studentName + " " + hostelFees + " " + foodType);
        }
```

For any utility method like selectallrows 0r delete or update we need to call explicitly this method **createStudentDBConnection** everywhere I have to call this method everywhere wherever db task to done

Once there is creation of bean then please execute this method automatically

---

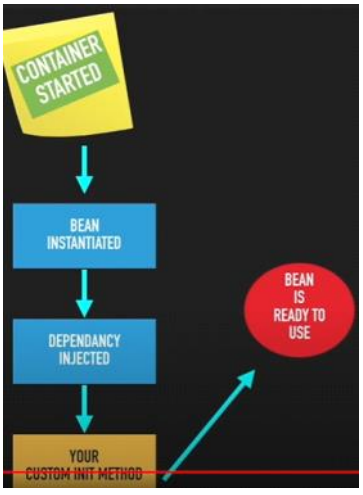
```java
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);

}
```

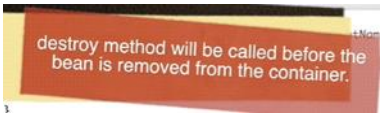
Init() ->Method

```
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {

    // load driver
    Class.forName(driver);

    // get a connection
    con = DriverManager.getConnection(url, userName, password);

}
```

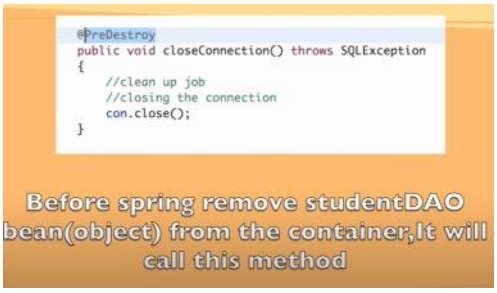
Here the createStudentDBConnection() is the init method for us. annotate a method with @PostConstruct to use it as a init method.

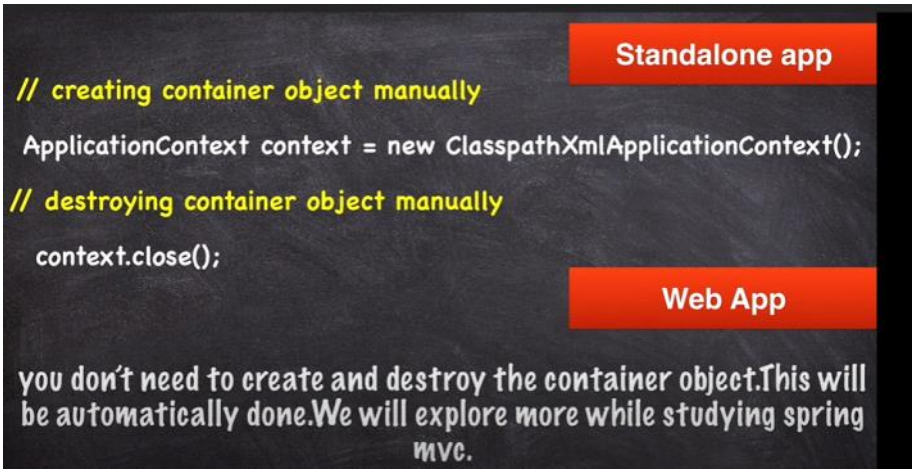
First create object in container
Then it will inject all dependency which is in form of setter

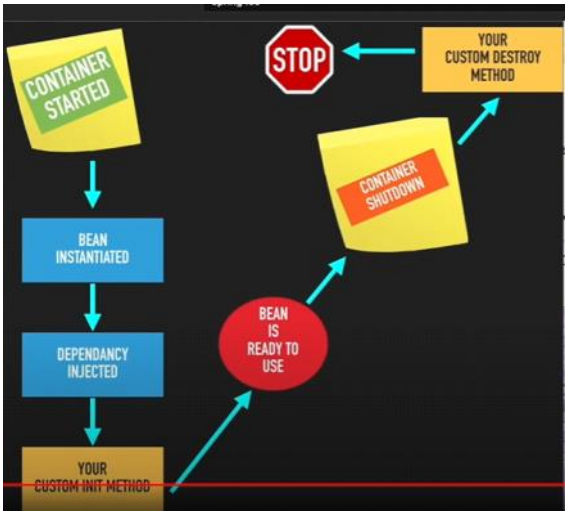


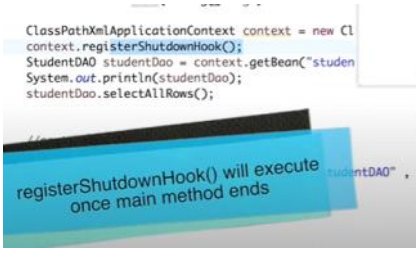


@PreDestory over the method which has to be called before the container is destroyed.



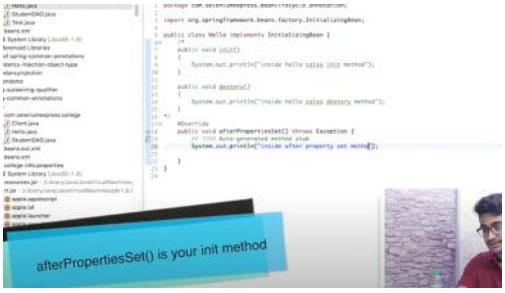






But it will not give error as context.close()

registerShutdownHook() will execute once the main thread ends. so once all your codes gets executed, It will be called and close your container. So It won't give us any exception irrespective of the line no we are calling it.




```
public class Application {
	public static void main(String[] args) {
		ApplicationContext ap = new AnnotationConfigApplicationContext(MobileConfig.class);
		System.out.println("beans.loaded");
		LandLine c = ap.getBean("landLine", LandLine.class);
		c.peep();
		Sim c = ap.getBean("sim", Sim .class);// for interface direct calling bean in config
	}

}


@Configuration
@ComponentScan(basePackages = "com.practice")
public class MobileConfig {

@Bean
Sim sim()
{
Return new vodafone();
}


@Component
public class LandLine {

	@Autowired
	@Qualifier("idea")
	Sim sim;

	void peep()
	{
		sim.calling();
	}


}


@Component
public class Idea implements Sim {
	@Value("IDEA")
	String value;
	@Autowired
	Network network;
	@Autowired
	Balance balance;

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

	public Idea(String value, Network network, Balance balance) {
		super();
		this.value = value;
		this.network = network;
		this.balance = balance;
	}

	public void calling() {
		network.display(value);

	}

	public void balance() {
		balance.display(value);

	}
}

@Component
public class Network {


	@Value("Network ")
	String name;

	 void display(String val) {
		System.out.println(val+" has a  "+name);

	}

}
```

Spring scope->

NO "GLOBAL SESSION" SCOPE

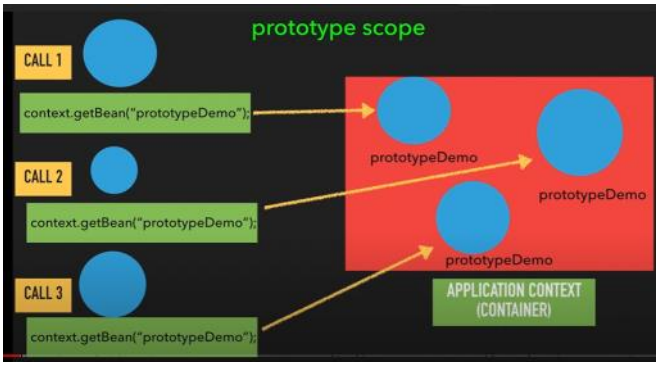
Spring removed this scope with the 5.x release.
No support for portlet.

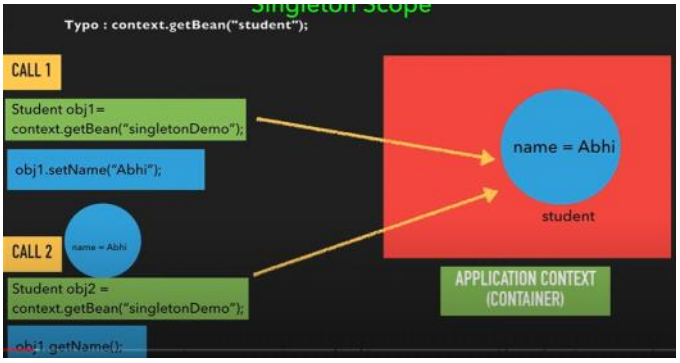
```java
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

    public static void main(String[] args) {

        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        SingletonDemo obj1 = context.getBean("singletonDemo", SingletonDemo.class);
        SingletonDemo obj2 = context.getBean("singletonDemo", SingletonDemo.class);
        System.out.println(obj1 + " " + obj2);
    }

}
```

obj1 and obj2 object reference are the same or different?

Guess the output

```java
        if(obj1 == obj2) {
            System.out.println("same object returned..");
        }
    }

}
```






```
@Component
@Scope(value="singleton")
public class School {
	public School() {
		System.out.println("Waoo school");
	}

}
@Component
@Scope(value="prototype")
public class Student {

	String name;

	public Student() {
		System.out.println("Waoo student");
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

}
```

So only singelton typebean will be loaded at run time eager loading

When protoype type bean is called then only bean is created when we asked for it

```
@Component
@Scope(value="singleton")
public class School {
    @Autowired
    Student student;
     public School() {
        System.out.println("Waoo school");
    }
```

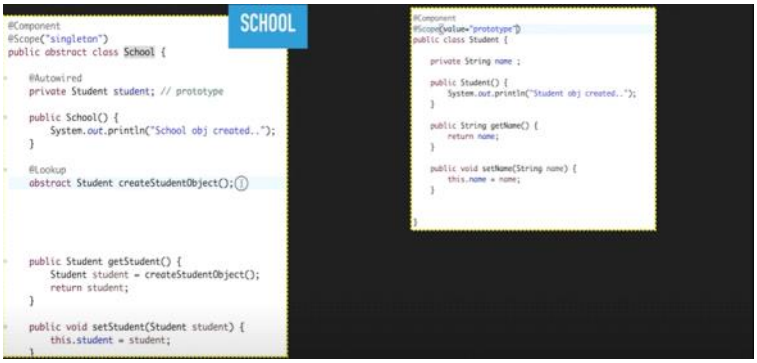
Then also it will give same object of student even if student is prototype

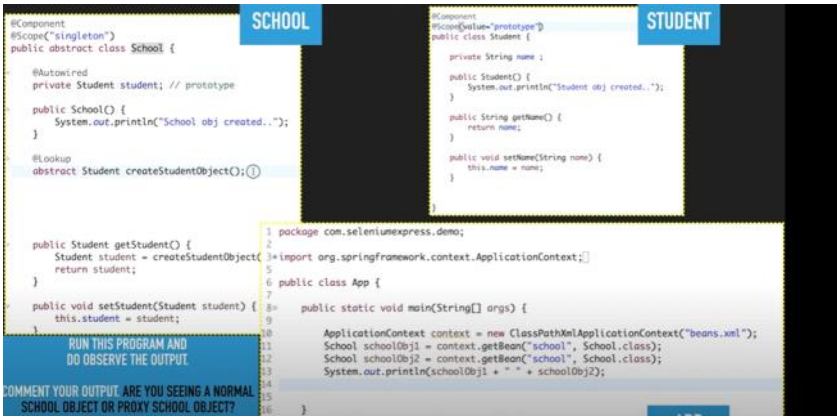
```
omponent
cope(value="prototype", proxyMode = ScopedProxyMode.TARGET_CLASS)
olic class Student {
```

This will give proxy to School class so each time we will get differne object

```
@Scope("singleton")
public class School {

    @Autowired
    private Student student; // prototype

    public School() {
        System.out.println("School obj created..");
    }

    @Lookup
    Student createStudentObject(){

        return null;
    }


    public Student getStudent() {
        Student student = createStudentObject();
        return student;
    }
```

Another method to return different object other than proxymode






```
public class App {

    public static void main(String[] args) {

        ApplicationContext appContext1 = new ClassPathXmlApplicationContext("beans.xml");
        ScopeTest scopeTest1 =appContext1.getBean("scopeTest",ScopeTest.class);
        ScopeTest scopeTest2 =appContext1.getBean("scopeTest",ScopeTest.class);

        System.out.println(scopeTest1 + " " + scopeTest2);

        ApplicationContext appcontext2 = new ClassPathXmlApplicationContext("beans2.xml");
        ScopeTest scopeTest3 = appcontext2.getBean("scopeTest",ScopeTest.class);

        System.out.println(scopeTest3);


    }
```

It's not same as singelton gof pattern
When we create two bean xml then again new object will be created

For request scope

```
@RequestMapping("/testing1")
public void test(HttpServletResponse response) throws IOException {

    MyBean myBean1 = context.getBean("myBean",MyBean.class);
    MyBean myBean2 = context.getBean("myBean",MyBean.class);

    System.out.println(myBean1 + " " + myBean2);


}
```

So in this only one my bean object created