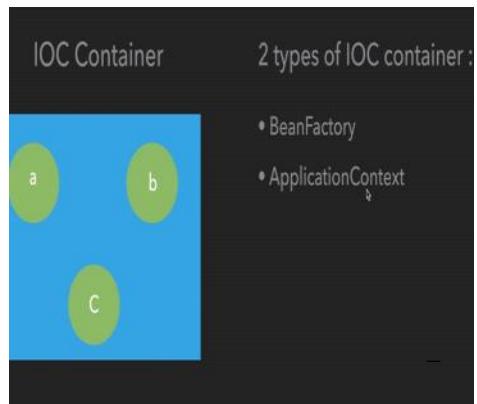
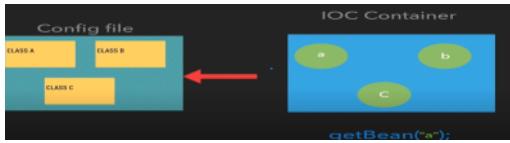


## ControllerAdvice



A screenshot of a Java IDE showing code related to Spring IOC. The code includes a beans.xml configuration file and Java classes for Vodaphone, Mobile, and Airtel. The beans.xml file defines a bean named 'airtel' of class 'com.seleniumexpress.ioc.Airtel'. The Mobile.java class contains a main method that prints 'config loaded', gets a bean for 'Vodaphone', and then calls methods on 'voda' (calling and data). The Airtel.java class has similar logic for 'air' (calling and data).

```
<?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.seleniumexpress.ioc.Airtel"></bean>

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();
    }
}
```

A screenshot of a Java IDE showing another version of the Java code. It uses the same beans.xml configuration but changes the main method in Mobile.java to use 'Airtel' instead of 'Vodaphone'. The output window shows the message 'Constructor loaded of airtel'.

```
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();
    }
}
```

```

6 public class Mobile {
7
8     public static void main(String[] args) {
9
10        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
11        System.out.println("config loaded");
12
13        Sim sim = context.getBean("sim", Sim.class);
14        sim.calling();
15        sim.data();
16    }
17
18 }

```

Without touching vodafone and airtel

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans
5   http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7 <bean id="sim" class="com.seleniumexpress.ioc.Jio"></bean>
8
9
10</beans>
11
12

```

Control is taken by framework now they are creating object





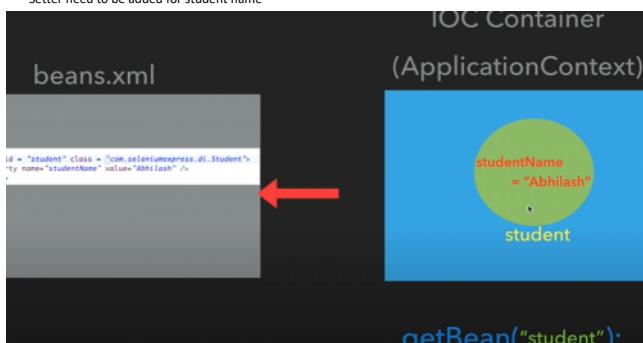
```

public class Exam {
    public static void main(String[] args) {
        Student student = new Student();
        student.setStudentName("Abhilash Panigrahi");
        student.displayStudentInfo();
    }
}
  
```

```

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

Setter need to be added for student name



```

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 Dependency

```

public class Student {
    private int id;
    private String studentName;

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

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

```

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

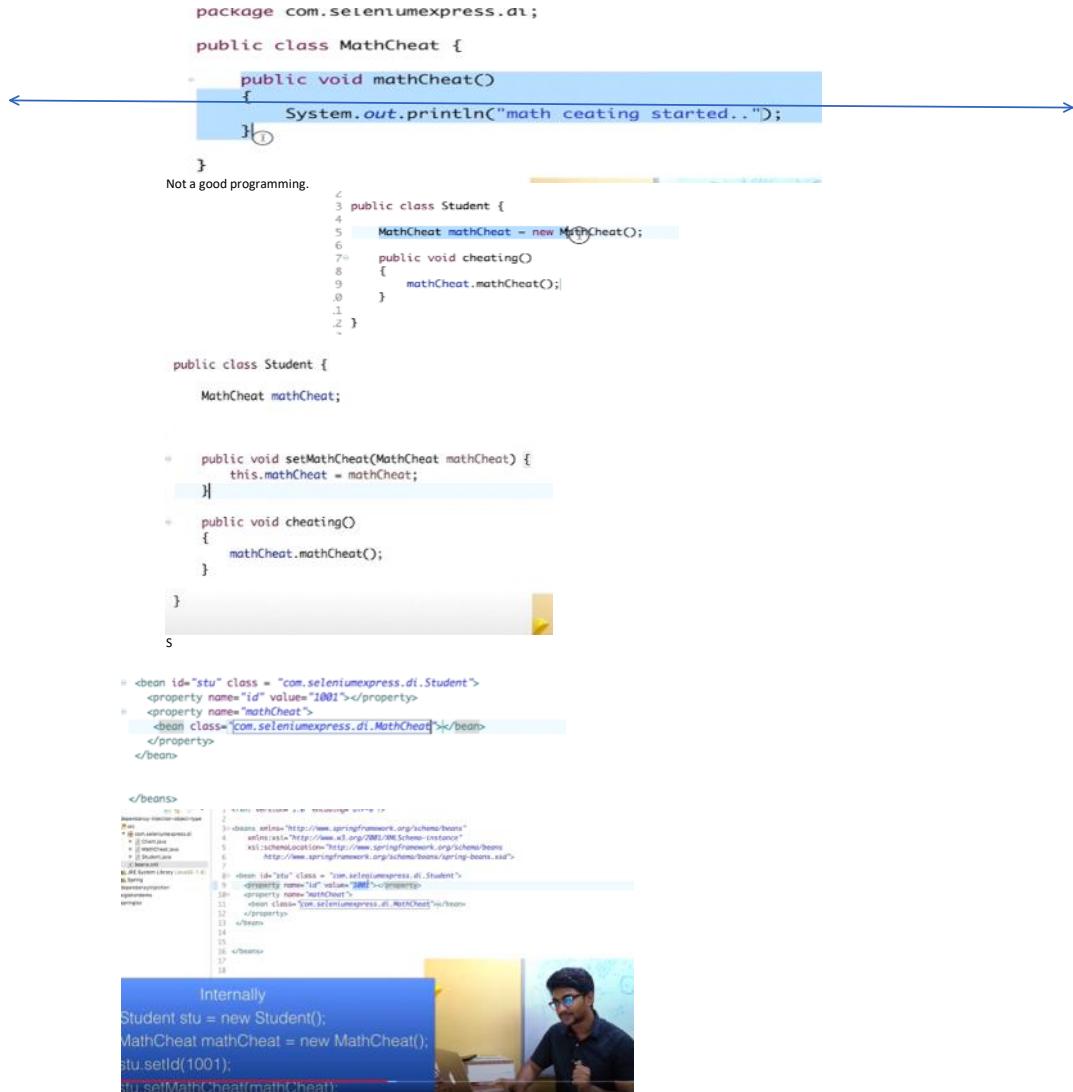
```

<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 cheating started..");
    }
}

Not a good programming.

public class Student {
    MathCheat mathCheat;

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

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

<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>

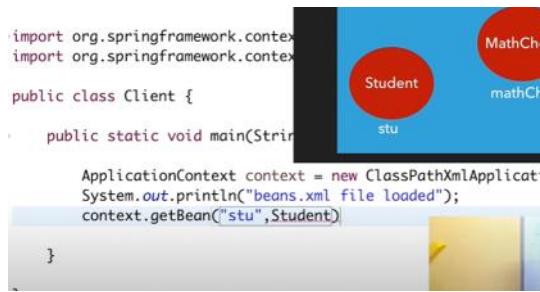
```

Internally

```

Student stu = new Student();
MathCheat mathCheat = new MathCheat();
stu.setId(1001);
stu.setMathCheat(mathCheat);

```

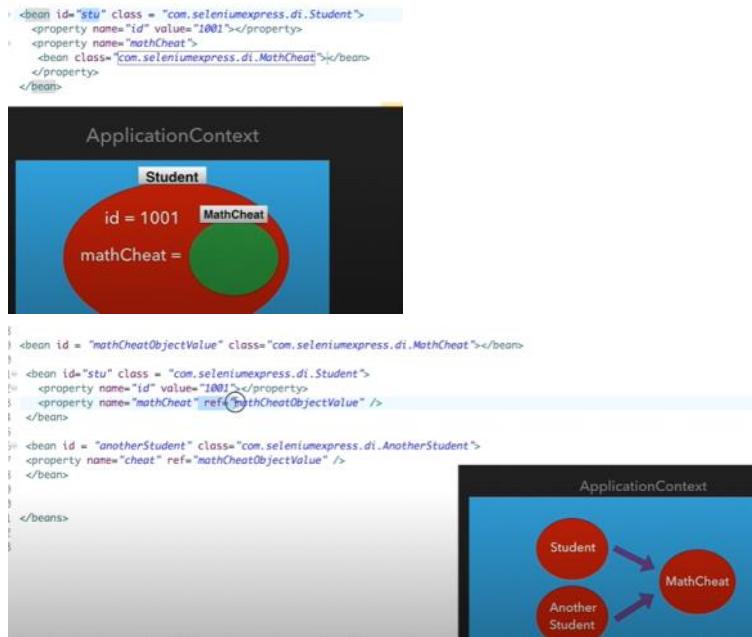


```

import org.springframework.context.*;
import org.springframework.context.*;

public class Client {
    public static void main(String[] args)
    {
        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("beans.xml file loaded");
        context.getBean("stu", Student);
    }
}

```



Loose coupling

```

1 package com.seleniumexpress.di;
2
3 public class Student {
4
5     private Cheat cheat;
6
7     public void setCheat(Cheat cheat) {
8         this.cheat = cheat;
9     }
10
11    public void cheating() {
12
13        cheat.cheat();
14    }
15
16 }
17

```

```

3 public interface Cheat {
4
5     public void cheat();
6
7 }
8

```

```

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

<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>

```

```

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.....

```

Human human = new Human();
Heart heartObject = new Heart();
human.setHeart(heartObject);

<bean id = "heartObject" class="com.seleniumexpress.autowired.Heart">
<property name="heart" ref = "heartObject"></property>
</bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human">
<property name="heart" ref = "heartObject" />
</bean>

```

I am Removing the property that we manually set.

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">
```

Dear Spring !!  
While creating **Human** class object any dependency present in Human class meeting **autowire = "byName"** criteria, Inject those beans to their respective dependency.

```

3 public class Human {
4
5     private Heart heart;
6

```

Heart class reference variable('heart') name and bean id name ('heart') matched !!

Autowiring Successful

By type autowired niche wala

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

Dear Spring !!  
While creating **Human** class object any dependency present in Human class meeting **autowire = "byType"** criteria(in beans.xml file), Inject those beans to their respective dependency.

```

3 public class Human {
4
5     private Heart heart;
6

```

The type of the Variable and the type of the Bean matched

Autowiring Successful

Autowire constructor

```

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 called");
    }
}

```

works same as autowire = "byConstructor"

```

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

```

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

```

3 import org.springframework.beans.factory.annotation.Autowired;
4
5 public class Human {
6     private Heart heart;
7     public Human() {
8
9     }
10
11     @Autowired
12     public Human(Heart heart) {
13         this.heart = heart;
14     }
15
16     public void setHeart(Heart heart) {
17         this.heart = heart;
18         System.out.println("setter method called");
19     }
20

```

[Sep 15, 2018 1:58:21 PM org.springframework.context.support.ClassPathXmlApplicationContext INFO: Loading XML bean definitions from c:\you are dead]

```

http://www.springframework.org/schema/beans/spring-beans.xsd"
<context:annotation-config>
<bean id = "heartObjectValue" class="com.seleniumexpress.autowired.Heart"></bean>
<bean id = "human" class="com.seleniumexpress.autowired.Human" >
</bean>

```

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

```

1 package com.seleniumexpress.autowired;
2
3 public class Human {
4     private Heart heart;
5     public Human() {
6
7     }
8     public Human(Heart heart) {
9         this.heart = heart;
10        System.out.println("human constr is getting called which has Heart as arg");
11    }
12
13    @Autowired
14    public void setHeart(Heart heart) {
15        this.heart = heart;
16    }
17
18    bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" />
19    bean id = "octopusHeart" class="com.seleniumexpress.autowired.Heart" />
20

```

**How @Autowired works?**

1. first it tries to resolve with "byType".
2. If byType fails then it goes with "byName"

```

5   xmlns:context="http://www.springframework.org/schema/context"
6   xsi:schemaLocation="http://www.springframework.org/schema/beans
7       http://www.springframework.org/schema/beans/spring-beans.xsd
8       http://www.springframework.org/schema/context
9       http://www.springframework.org/schema/context/spring-context.xsd"
10
11 <context:annotation-config />
12
13 <bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" >
14 <property name="nameOfAnimal" value="Human" />
15 <property name="noOfHeart" value="1" />
16 </bean>
17
18 <bean id = "octpusHeart" class="com.seleniumexpress.autowired.Heart" >
19 <property name="nameOfAnimal" value="Octpus" />
20 <property name="noOfHeart" value="3" />
21 </bean>
22
23 <bean id = "human" class="com.seleniumexpress.autowired.Human" >
24

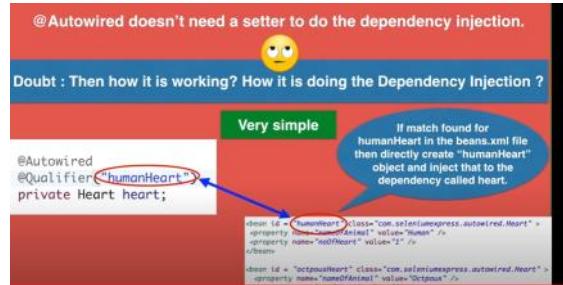
```

```

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

```

So no need to write setter if you are using @Autowired before the dependency



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

```

beans.xml Client.java *student-info.properties*
1 student.name = Abhilash
2 student.interestedCourse = Java
3 hobby = cricket

```

```

<bean id = "student" class="com.seleniumexpress.loadingFrompropertiesfile.Student" >
<property name="name" value = "${student.name}"/>
<property name="interestedCourse" value = "${student.interestedCourse}"/>
<property name="hobby" value = "${student.hobby}"/>
</bean>

```

<context:property-placeholder/>  
To know spring I am using from property file

@Value annotation remove property

```

12
13<bean id = "student" class="com.seleniumexpress.loadingfrompropertiesfile.Student">
14
15 </bean>
16
17 .
18
19

```

## use some Annotations

using @Value annotation

```

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()
}

```

**@Required**

```

public void setIntrestedCourse(String intrestedCourse) {
    this.intrestedCourse = intrestedCourse;
}

//@Value("travelling")
public void setHobby(String hobby) {
    this.hobby = hobby;
}

```

**Introducing @Required**  
want to make "intrestedCourse" as required.

Restricted required to mandatory

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

Beans file is configuration file

```

<bean id = "collegeBean" class="com.seleniumexpress.college.College">
</bean>

```

Attribute : id  
The object identifier for a bean. A bean id may not be used more than once within the same <beans> element.  
Data Type : string

**Internally : College collegeBean = new College();**

@Componet does same work as above

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

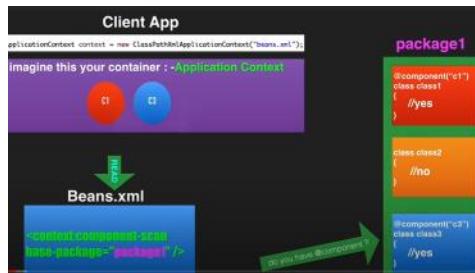
@Componet does same thing 1 create bean  
2 and register to IOC container

```

@Component("collegeBean")
public class College {
}

8
9 @Component("collegebean")
10 public class College {
11
12     @Autowired // for class type
13     Student student;
14
15     @Value("Murari") // for property like string
16     String name;
17
18     void dis() {
19         student.displa(name);
20     }
21 }
22
23

```



**@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")
public class CollegeConfig {
    @Bean
    public College collegeBean()
    {
        return new College();
    }
}
```

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

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

```

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

```
}
```

```
public 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 - becomes a bean
    {
        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();
        teacher.teach();
    }
}

@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 teachO {

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

```

Screenshot showing four Java code editors in a grid, each displaying a different class from the Spring framework. The classes are:

- `Body.java`: A component with a constructor annotated with `@Autowired` and `@Qualifier("innerBehaviour")`. It has fields `Heart heart`, `Lungs lung`, and `String value`.
- `InnerBehaviour.java`: A primary component that implements the `Behaviour` interface. Its `display()` method prints "Inner behaviour".
- `OuterBehaviour.java`: A primary component that implements the `Behaviour` interface. Its `display()` method prints "Outer behaviour".
- `BodyConfig.java`: A configuration class with bean definitions for `Body`, `Heart`, `HeartBuffalo`, and `Lungs`.

The console output shows the application terminating with the message "Human is not functioning".

```

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

```

```

@Component
public class College {

    //@Value("${college.Name}")
    private String collegeName;
    I @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 cycle of bean

```

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("Retrieving 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 or 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

```

@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {
    // load driver
    Class.forName(driver);

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

```

**Hey spring , Once you crate  
StudentDAO Object Please call  
`createStudentDBConnection()` by  
yourself. Don't wait for me to call it.**

Init() -> Method

```

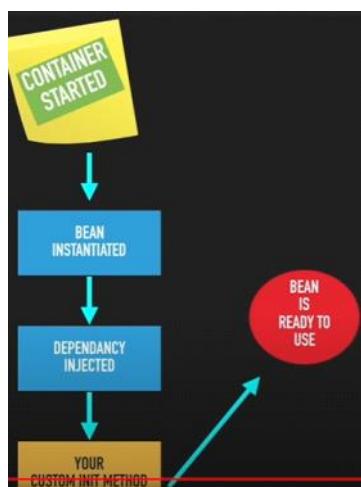
@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {
    // load driver
    Class.forName(driver);

    // 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



You can add custom code/logic during bean initialization

It can be used for setting up resources like db/socket/file etc

```

65:     public void createStudentDBConnection() throws ClassNotFoundException, SQ
66:     System.out.println("creating connection..");
67:     // load driver
68:     Class.forName(driver);
69:
70:     // get a connection
71:     con = DriverManager.getConnection(url, userName, password);
72:
73: }

```

why init()??

destroy method will be called before the bean is removed from the container.

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

```

@PreDestroy
public void closeConnection() throws SQLException
{
    //clean up job
    //closing the connection
    con.close();
}

```

Before spring remove studentDAO bean(object) from the container; It will call this method

**Standalone app**

```

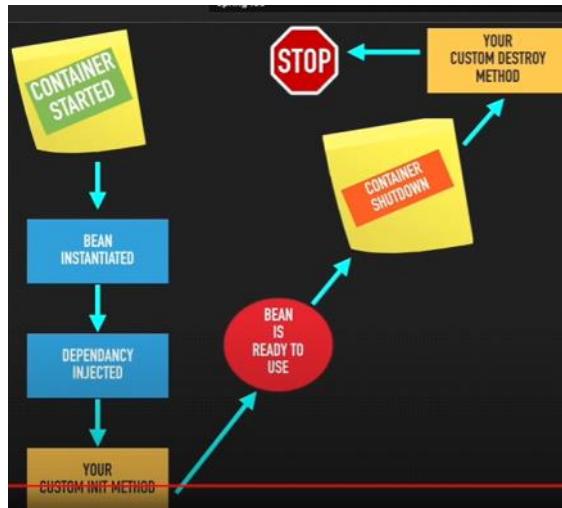
// creating container object manually
ApplicationContext context = new ClasspathXmlApplicationContext();

// destroying container object manually
context.close();

```

**Web App**

you don't need to create and destroy the container object. This will be automatically done. We will explore more while studying spring MVC.



```

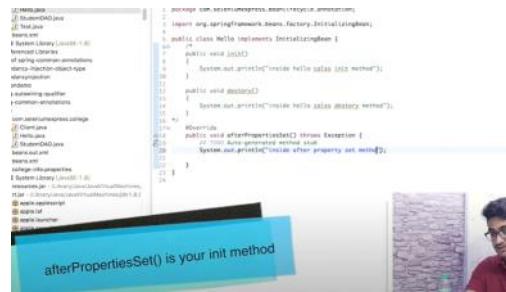
ClassPathXmlApplicationContext context = new Cl
context.registerShutdownHook();
StudentDAO studentDao = context.getBean("studen
System.out.println(studentDao);
studentDao.selectAllRows();

```

registerShutdownHook() will execute once main method ends

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.

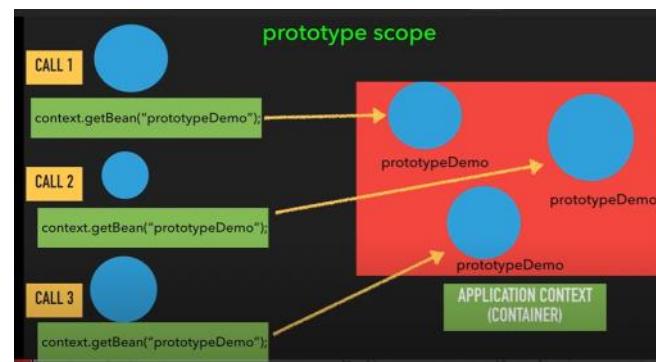
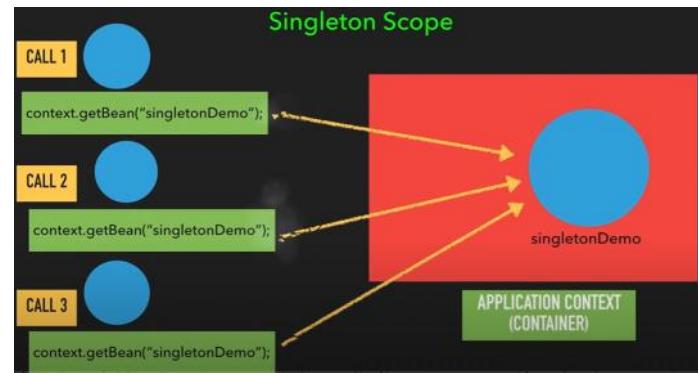
```
1 Java
2 package com.selenide;
3
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class App {
7
8     public static void main(String[] args) {
9
10         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
11         SingletonDemo obj1 = context.getBean("singletonDemo", SingletonDemo.class);
12         SingletonDemo obj2 = context.getBean("singletonDemo", SingletonDemo.class);
13         System.out.println(obj1 + " " + obj2);
14     }
15
16 }
17
```

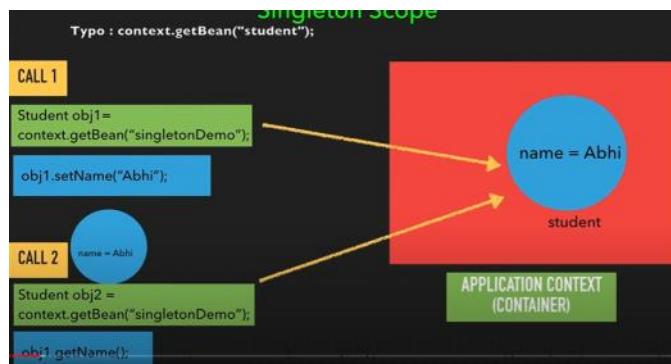
obj1 and obj2 object reference are the same or different?

Guess the output

```
14     if(obj1 == obj2) {
15         System.out.println("same object returned..");
16     }
17 }
```

terminated: App [Java Application] (LibraryJava/JavatutMachines/ide-9.0.4/jdk/Contents/Home/bin/java (21-Sep-2020, 2:28:10 PM)
com.seleniumexpress.demo.SingletonDemo@6f784a1a
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 singleton typebean will be loaded at run time eager loading

When prototype 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

```
Component
@Scope(value="prototype", proxyMode = ScopedProxyMode.TARGET_CLASS)
public class Student {
```

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

```

@component
@Scope("singleton")
public abstract class School {
    @Autowired
    private Student student; // prototype

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

    @Lookup
    abstract Student createStudentObject();①
}

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

public void setStudent(Student student) {
    this.student = student;
}

@component
@Scope("prototype")
public class Student {
    private String name;

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

    public String getName() {
        return name;
    }

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

```

```

@component
@Scope("singleton")
public abstract class School {
    @Autowired
    private Student student; // prototype

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

    @Lookup
    abstract Student createStudentObject();①
}

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

public void setStudent(Student student) {
    this.student = student;
}

RUN THIS PROGRAM AND
DO OBSERVE THE OUTPUT.

COMMENT YOUR OUTPUT. ARE YOU SEEING A NORMAL
SCHOOL OBJECT OR PROXY SCHOOL OBJECT?

```

```

1 package com.seleniumexpress.demo;
2 import org.springframework.context.ApplicationContext;
3
4 public class App {
5
6     public static void main(String[] args) {
7
8         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
9         School schoolObj1 = context.getBean("school", School.class);
10        School schoolObj2 = context.getBean("school", School.class);
11        System.out.println(schoolObj1 + " " + schoolObj2);
12
13    }
14
15 }

```

Output in Console:

```

Info: [App] [Java Application] (Library/Java/JavaVirtualMachines/jdk-0.4.jdk/Contents/Home/bin/java) [21-Sep-2020, 9:22:51 PM]
Info: [App] obj created.
Info: [App] com.seleniumexpress.demo.ScopeTest@60bf060 com.seleniumexpress.demo.ScopeTest@60bf060
Info: [App] obj created.

```

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

For request scope

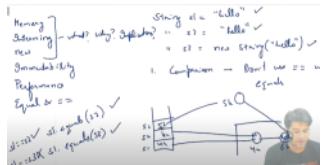
```

@RequestMapping("/testing1")
public void test(HttpServletRequest 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



## What Spring MVC does?

Spring MVC will help you to develop java J2EE (web) applications easily.

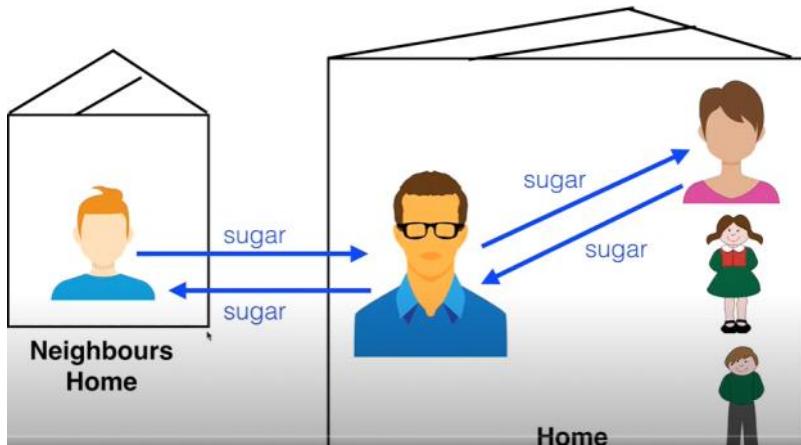
But the **problem** is setting up your project (or working environment) manually by doing a lot of configurations by using **XML** etc.

## What Spring boot does?

Spring boot helps us to wrap all spring components in a convenient way where no external XML configuration is needed.

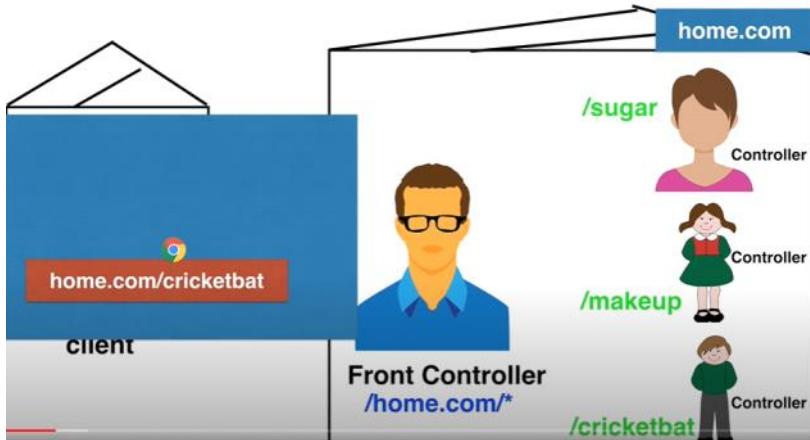
so the spring boot wraps spring MVC itself.

### Front CONTROLLER



Any request come outside the home dad knows the available 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

**Spring will automatically  
initialize the class having a  
@controller  
annotation and register that class  
with the spring container.**

```

@Controller
public class MomController {

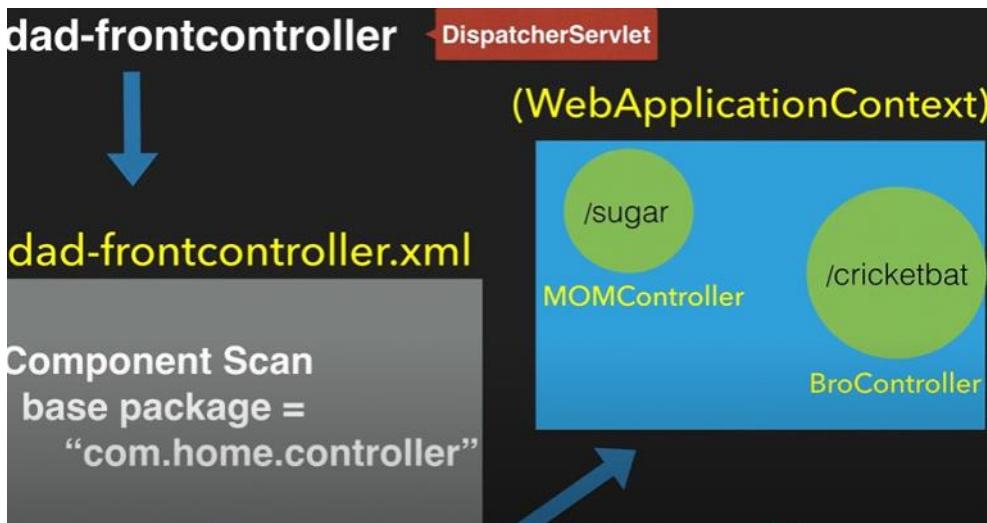
    @ResponseBody
    @RequestMapping("/sugar")
    public String giveSugar()
    {
        return "Ok..Here is your sugar";
    }
}

```

@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 appended with .xml



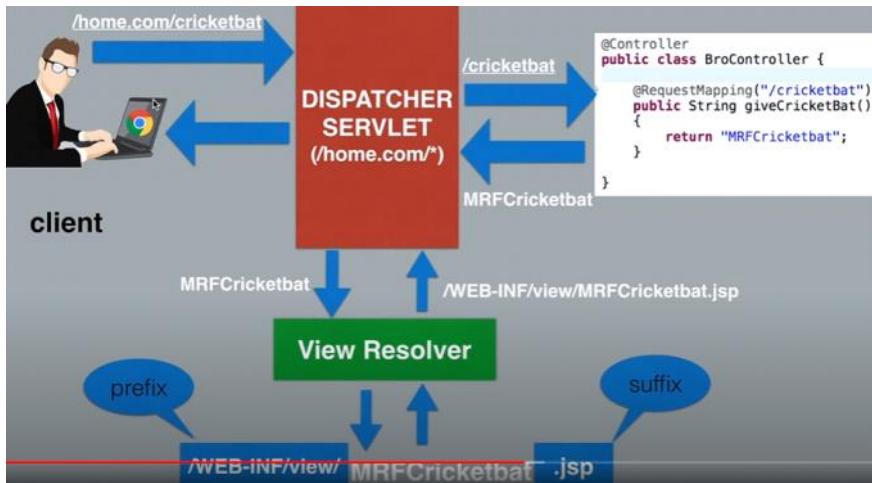
```
@Controller
public class BroController {
    @ResponseBody
    @RequestMapping("/cricketbat")
    public String giveCricketBat()
    {
        return "MRFCricketbat";
    }
}
```

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

But if we remove the @Response body then controller 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



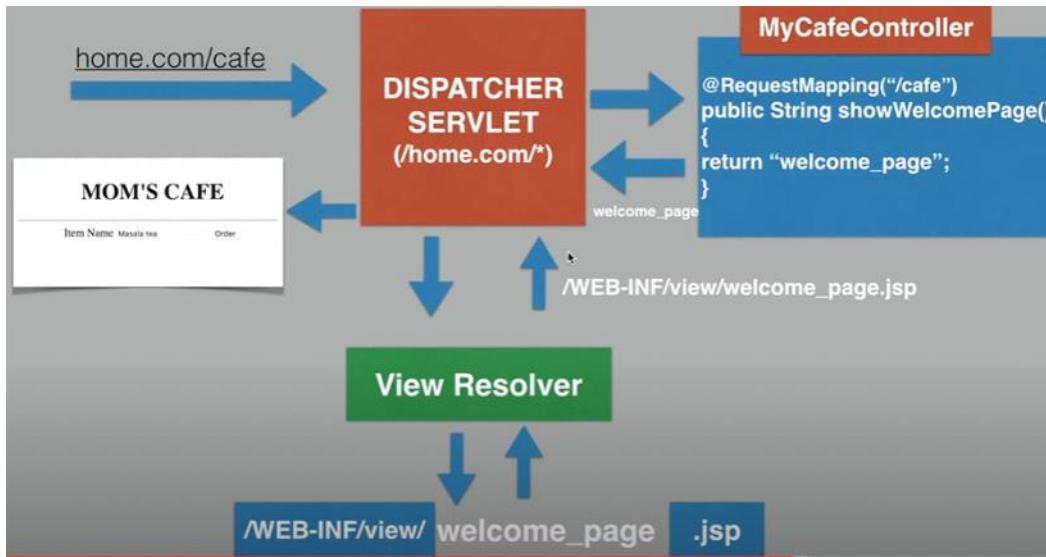
```
class InternalResourceViewResolver
{
    private String prefix = "";
    private String suffix = "";

    public void setPrefix(String prefix)
    {
        this.prefix = prefix;
    }
    public void setSuffix(String suffix)
    {
        this.suffix = suffix;
    }
}

InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
viewResolver.setPrefix("/WEB-INF/view/");
viewResolver.setSuffix(".jsp");
```

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

```
7 @Controller
8 public class MyCafeControllers {
9
.0= 10     @RequestMapping("/cafe")
11     public String showWelcomePage(Model model)
12     {
13         //sending data to view(jsp page)
14         String myName = "Abhilash";
15
16         model.addAttribute("myNameValue", myName);17
17
18     return "welcom-page";
19 }
```



```

@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";
}

```

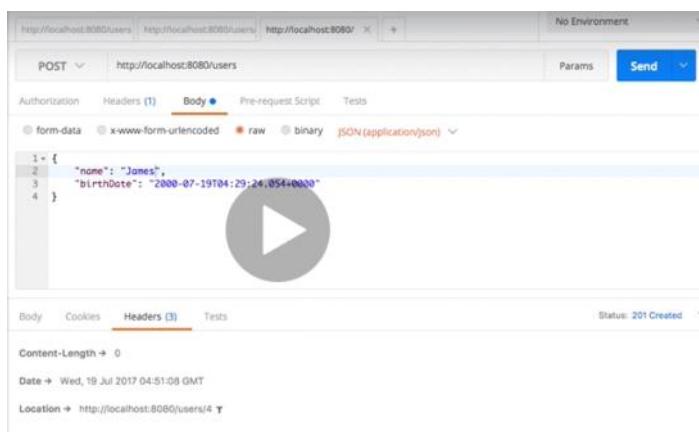
Reprsentational state transfer  
Maximum use of rest

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
    }
}

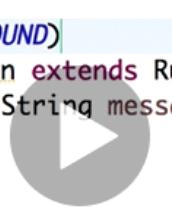
@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);
    if(user==null)
        throw new UserNotFoundException("id-" + id);

    return user;
}
```

```

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class UserNotFoundException extends RuntimeException {
        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);

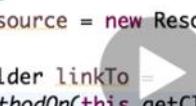
//"/all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

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

//HATEOAS

return user;

```



RestfulWebServicesApplication.java

2017-07-19 14:52:26,937 [http-nio-8080-exec-1] DEBUG o.s.w.s.r.Resource - Resource{id=1, name=Adam, birthDate=2017-07-19T09:26:18.337+0000, links=[{rel=/all-users, href=http://localhost:8080/users}]} registered.

2017-07-19 14:52:26,937 [http-nio-8080-exec-1] DEBUG o.s.w.s.r.Resource - Resource{id=1, name=Adam, birthDate=2017-07-19T09:26:18.337+0000, links=[{rel=/all-users, href=http://localhost:8080/users}]} registered.

2017-07-19 14:52:26,937 [http-nio-8080-exec-1] DEBUG o.s.w.s.r.Resource - Resource{id=1, name=Adam, birthDate=2017-07-19T09:26:18.337+0000, links=[{rel=/all-users, href=http://localhost:8080/users}]} registered.

2017-07-19 14:52:26,937 [http-nio-8080-exec-1] DEBUG o.s.w.s.r.Resource - Resource{id=1, name=Adam, birthDate=2017-07-19T09:26:18.337+0000, links=[{rel=/all-users, href=http://localhost:8080/users}]} registered.

```

//"/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

```

@Autowire
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;
}

```

Key	Value	Description
Content-Type	application/json	
Accept-Language	us	

```

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

    localeResolver.setDefaultLocale(Locale.US);

    return localeResolver;
}

```

Swagger->

```

@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: {...}
}

```

```

@ApiModelProperty(description="All details about the user. ")
public class User {
    private Integer id;
    @Size(min=2, message="Name should have atleast 2 characters")
    @ApiModelProperty(notes="Name should have atleast 2 characters")
    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

```

public class SomeBean {

    private String field1;

    private String field2;

    @JsonIgnore
    private String field3;

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

```

Dynamic filter

```

@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1

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

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

        FilterProvider filters = new SimpleFilterProvider().addFilter("SomeBeanFil
        MappingJacksonValue mapping = new MappingJacksonValue(someBean);

        mapping.setFilters(filters);

        return mapping;
    }
}

```

```

@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

```

@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"));
}

@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"));
}

```

GET ▼ http://localhost:8083/person/header

Headers (7)

Key	Value
User-Agent	PostmanRuntime/7.26.8
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
X-API-VERSION	2
Body	
Cookies	
Headers (5)	
Test Results	

Pretty Raw Preview Visualize JSON

```

1 [
2   "name": {
3     "firstname": "Murari",
4     "lastname": "Kumar"
5   }

```

### 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"));
}

```

GET ▼ http://localhost:8080/person/produces

Headers (2)

Key	Value	Description
X-API-VERSION	2	**
Accept	application/vnd.company.app-v1+json	

Body

Pretty Raw Preview JSON

```

1 [
2   "name": "Bob Charlie"
3 ]

```

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
    UserRepository user;

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

    @GetMapping("/findById/{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 UserRepository extends JpaRepository<User, Integer>{
}

```

Many to one relationship

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

```
@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

```
@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
    }
}
```

```
@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();
}
```

```
@Entity
public class Post {
    @Id
    @GeneratedValue
    int id;
    String description;
    @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

1 GET /users/{PathVariable}?QueryKey=QueryValue HTTP/1.1  
2 Host: localhost:8080  
3 header-key: headerValue

## HTTP Request To Spring Annotations

- Path Variable = `@PathVariable`
- Query (Request) Parameter = `@RequestParam`
- Header Parameter = `@RequestHeader`

```

1  @RestController
2  @RequestMapping("/requestParam")
3  public class RequestParamController {
4
5      @GetMapping("/required")
6      public void requiredRequestParam(@RequestParam String name){
7          System.out.println(name);
8      }
9
10     //optional
11
12     @GetMapping("/notRequired")
13     public void notRequiredReqeustParam(@RequestParam String name){
14         System.out.println(name);
15     }
16
17     @GetMapping("/notRequiredDef")
18     public void notRequiredDefaultRequestParam(@RequestParam String name){
19         System.out.println(name);
20     }
21
22     @GetMapping("/notRequiredOpt")
23     public void notRequiredOptionalRequestParam(@RequestParam String name){
24         System.out.println(name);
25     }
26
27     //allParams
28     @GetMapping("/allParams")
29     public void notRequiredOptionalAllParams(@RequestParam String name){
30         System.out.println(name);
31     }
32
33     //allParams
34     @GetMapping("/allParams")
35     public void notRequiredOptionalAllParams(@RequestParam String name){
36         System.out.println(name);
37     }
38
39

```

Postman

My Workspace

testapi

test

GET http://localhost:8080/requestParam/required?name=QueryValue

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	QueryValue	

Query parameter 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));
}

```

Postman screenshot showing a GET request to `http://localhost:8080/requestParam/allParams?name=QueryValue&param2=darklord&param3=3232ads`. The 'Params' tab displays the following query parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	QueryValue	
<input checked="" type="checkbox"/> param2	darklord	
<input checked="" type="checkbox"/> param3	3232ads	

Postman screenshot showing a GET request to `http://localhost:8080/requestParam/listParams?id=1,2,3`. The 'Params' tab displays the following query parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	1,2,3	
param2	darklord	
param3	3232ads	

### 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 repository.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>

```

```

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

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

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

@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 ehache exceed from its configured size ehcache removed expired elements  
If that doesn't clear enough space it clear object from ehcache  
By default it remove LRU elements

Second level caching is we put on entity or whole object

```

@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/>>

Strange  
high to memory  
DBE & DBO  
operations  
on  
stack  
high  
depth

$$(1111)' = (111) + 0001 \\ = \begin{array}{r} 0000 \\ 0001 \\ \hline 0001 \end{array}$$

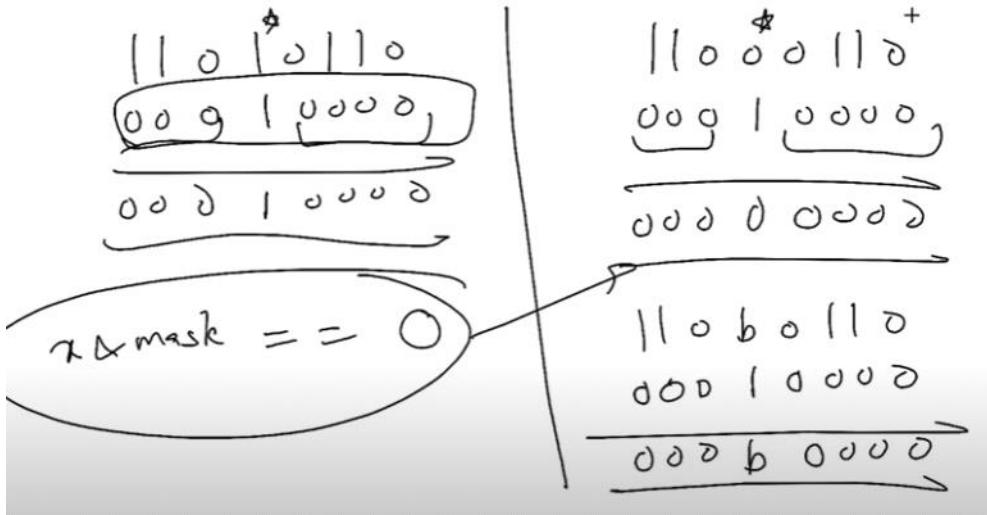
$0\ 000$	$\rightarrow 0$	$A1 -$
$0\ 001$	$\rightarrow 1$	
$0\ 010$	$\rightarrow 2$	$A2 -$
$0\ 011$	$\rightarrow 3$	
$0\ 100$	$\rightarrow 4$	
$0\ 101$	$\rightarrow 5$	$A3$
$0\ 110$	$\rightarrow 6$	
$0\ 111$	$\rightarrow 7$	
$1\ 000$	$\rightarrow 8$	$-0 \quad -8$
$1\ 001$	$\rightarrow 9$	$-1 \quad -7$
$1\ 010$	$\rightarrow 10$	$-2 \quad -6$
$1\ 011$	$\rightarrow 11$	$-3 \quad -5$
$1\ 100$	$\rightarrow 12$	$-4 \quad -4$
$1\ 101$	$\rightarrow 13$	$-5 \quad -3$
$1\ 110$	$\rightarrow 14$	$-6 \quad -2$
$1\ 111$	$\rightarrow 15$	$-7 \quad -1$

$$>> y = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}}$$

$$y >> 3 = \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$$

$$y >>> 3 = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}}$$

On	for	off	& and	toggle	nxor	check
$x = \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$		$x = \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$				$\underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$
$mask = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}}$		$mask = \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}}$				
$(mask \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}})$		$(x \& mask) \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$		$\underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}}$		
$0 \rightarrow X$		$0 \rightarrow \checkmark$				
$1 \rightarrow \checkmark$		$1 \rightarrow X$				



```
//write your code here
int rsbm = n & -n;
System.out.println(Integer.toBinaryString(rsbm))
```

Most significant one  $\rightarrow$  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

J10 # J21

	✓	✓	✓	✓					
-1	1 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	0 <sub>3</sub>	1 <sub>4</sub>	0 <sub>5</sub>	1 <sub>6</sub>	2 <sub>7</sub>	1 <sub>8</sub>
0 - 6	0	0	0	1					
1 - 0	1	2	2	2					
2 - 0	0	0	1		1				
key	0#0	1#-1	2#-2	2#-1	1#-1				
-1	0	1	2						

ashmaps #datastructure #algorithms

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

12 views • Oct 25, 2020

0	101	5
0	110	6
0	111	7
1	000	8
1	001	9
1	010	10
1	011	11
1	100	12
1	101	13

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

Sum of value

```
Scannersc=newScanner(System.in);
intsum=0;
for(inti=0;i<4;i++){
System.out.println("Enter value");
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 remainder  
15|4  
11| 11

If divide by 8 last 3 will be remainder  
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

$$n << x$$

$$n \neq 2^x$$

$$n \gg x$$

$$\frac{n}{2^x} +$$

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 communication-> 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;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Counter implements Runnable{
    static int id=0;
    int count;

    public Counter(){
        this.count=0;
    }

    public synchronized int getCount() throws InterruptedException{
        wait();
        this.count=count+1;
        System.out.println(count);
        return count;
    }

    @Override
    public void run(){
        try{
            getCount();
        }catch(Exception e){
        }
    }
}

class Counter1 implements Runnable{
    static int id=0;
    int count;

    public Counter1(){
        this.count=0;
    }
}
```

```
}

public synchronized void getCount() throws InterruptedException{

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

notify();
}
}

@Override
public void run(){
try{
getCount();

} catch(Exception e){

}
}
}

public class Application{
public static void main(String[] args) throws InterruptedException{
ExecutorService executorService= Executors.newFixedThreadPool(4);
executorService.execute(new Thread(new Counter()));
executorService.execute(new Thread(new Counter1()));
}
}
```

## Microservice

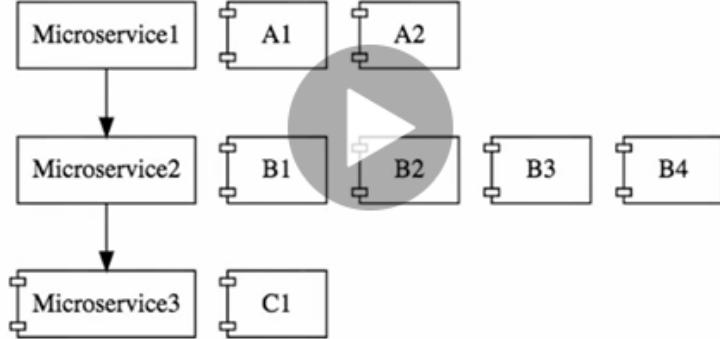
### 1.Bounded Context

Earlier we were building one monolithic app now 10-20 microservices

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

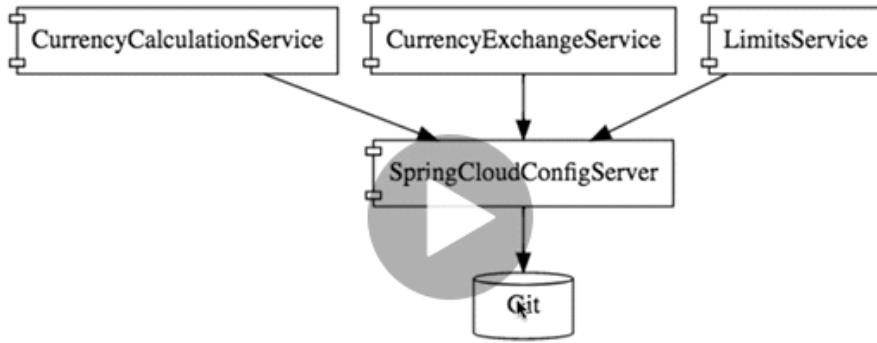
### 4. Visibility

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

5 Pack of cards

If one microservice fail then all will fail

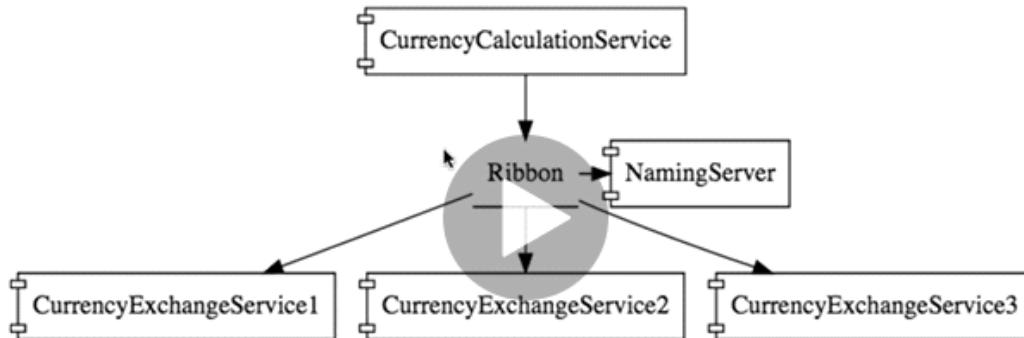
## Spring cloud



## *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*

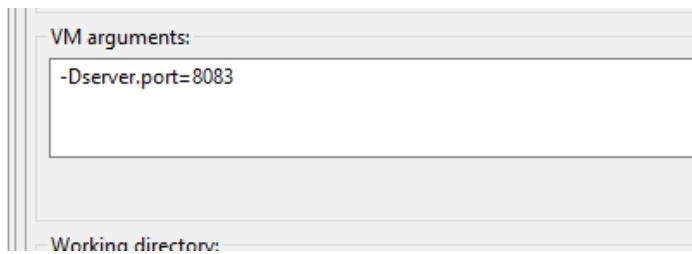
Setting property from application properties

```

@.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

```

@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>

---

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

---

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

---

```
@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

```

@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

```
@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);
    }
}
```

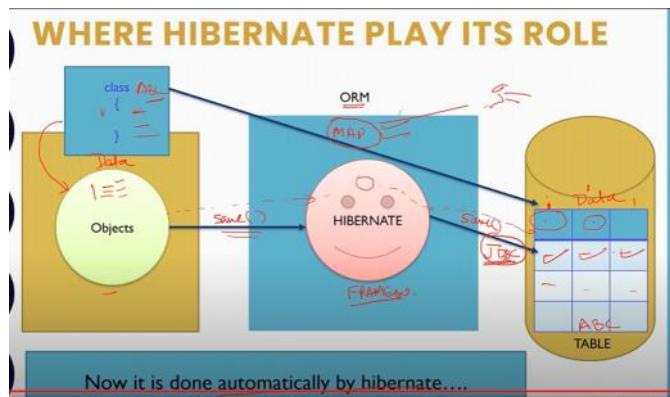
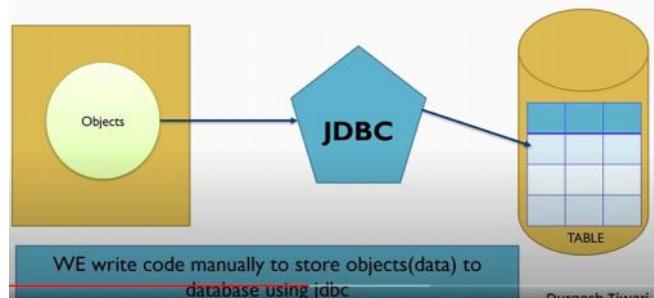
## Hibernate

### #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 force 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**.

## TRADITIONAL WAY TO SAVE DATA(JDBC)



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 hibernate

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

III. Configuration  
IV. test class

```

package beans;
public class Student {
    private int id;
    " String name;
    " " email;
    " int marks;
    || public setters & getters
}

```

Student.hbm.xml

```

DTD
<hibernate-mapping>
    <class name="beans.Student" table="Student" schema="System">
        PK <id name="id" column="sid"/>
        <property name="name" column="sname"/>
        <property name="email" column="email"/>
        <property name="marks" column="marks"/>
    </class>
</hibernate-mapping>

```

```

package beans;
public class Student {
    private int id;
    " String name;
    " " email;
    " int marks;
    || public setters & getters
}

```

pojo class

```

St=new Student();
St.setId(111);
St.setName("ABC");
St.setEmail("abc@gmail.com");
St.setMarks(500);

```

Student.hbm.xml

```

<class name="beans.Student" schema="System">
    PK <id name="id" column="sid"/>
    <property name="name" column="sname"/>
    <property name="email" column="email"/>
    <property name="marks" column="marks"/>
</class>
</hibernate-mapping>

```

Configuration cfg = new Configuration();
cfg.configure();
SessionFactory sf = cfg.buildSessionFactory();
Session s = sf.openSession();
s.save(st);

**DURGASOFT**  
[www.durgasoft.com](http://www.durgasoft.com)

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

```

<property name="dialect"> org.hibernate.dialect.OracleDialect </property>
<mapping resource="Student.hbm.xml"/>

```

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

```

Student st=new Student();
st.setId(111);
st.setName("abc");
st.setEmail("abc@gmail.com");
st.setMarks(500);
//student object state is transient
Configuration cfg=new Configuration();
cfg.configure("resources/hibernate.cfg.xml");
SessionFactory sf=cfg.buildSessionFactory();
Session s=sf.openSession();
s.save(st);
//student object state is persistant
s.beginTransaction().commit();
//student object will move database
s.evict(st);
//student object will be remove from database
//then gc can collect ur student object

```

**DURGASOFT**  
[www.durgasoft.com](http://www.durgasoft.com)

a hibernate command Hbm2ddl.auto we can do DDL operation

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

```
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-mapping-3.

  <class name="beans.Student" table="student" schema
    <id name="id" column="sid"/>
    <property name="name" column="sname"/>
    <property name="email" column="semail"/>
    <property name="marks" column="smarks"/>
  </class>

</hibernate-mapping>
```

```
<property name="connection.driver_class">oracle.jd-
<property name="connection.url">jdbc:oracle:thin:@
<property name="connection.username">system</prope
<property name="connection.password">manager</prop
<property name="connection.pool_size">10</property>

<property name="dialect">org.hibernate.dialect.Ora
<property name="hbm2ddl.auto">create</property>

<mapping resource="resources/student.hbm.xml"/>
<mapping resource="resources/course.hbm.xml"/>
<mapping resource="resources/employee.hbm.xml"/>
<mapping resource="resources/department.hbm.xml

/session-factory>
```

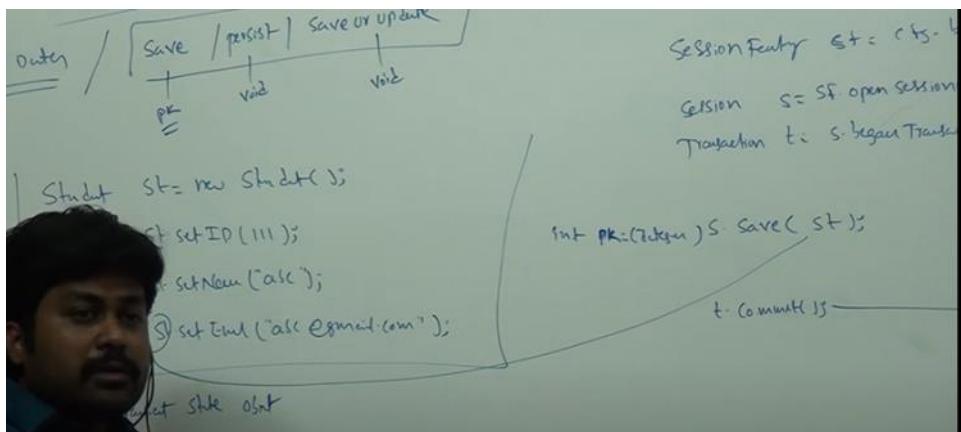
DURGASOFT

```
ty name="hbm2ddl.auto">create</property>
ty name="show_sql">true</property>
```

Internal table creation

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

Validate  
Must have that 5 columns



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

Serializable

```
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 fil= new FileInputStream(f);
    ObjectInputStream dli= new ObjectInputStream(fil);
    Save sl=(Save) dli.readObject();
    System.out.println(sl.id);

}

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

If we execute saveOrUpdate(st) operation

First it will run select query on that primary id;

Then it will compare

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

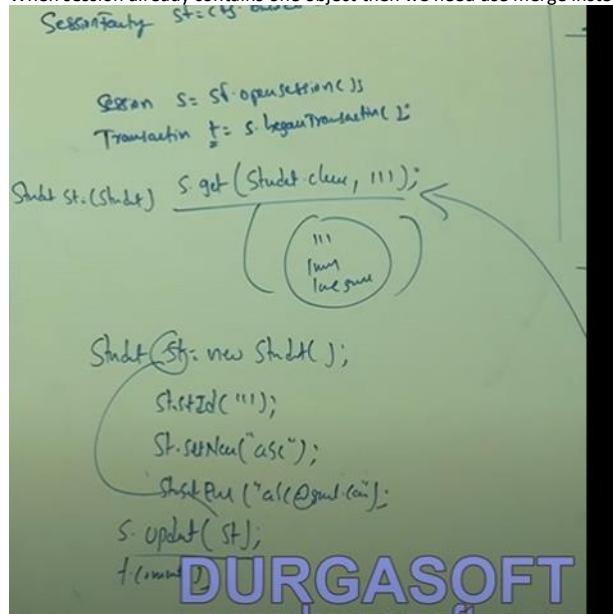
Limitation

We can only update non primary keys

It is not possible to update one column

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 generated

1.

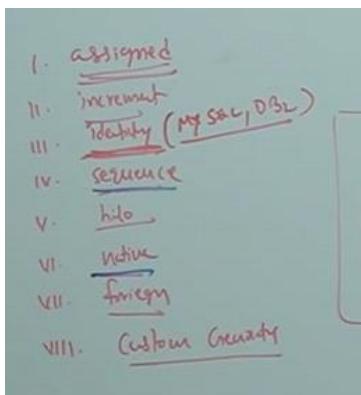
Assigned (default)-> It is handle by user itself

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

```

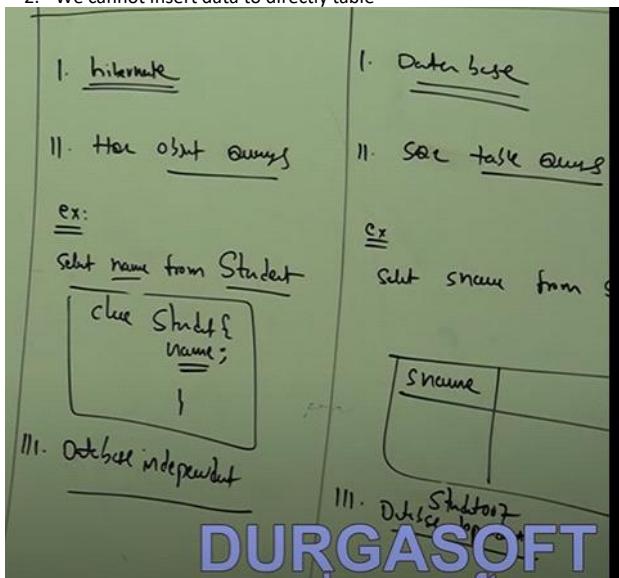
<hibernate-mapping>
    <class name="beans.BookMovie" table="tbc">
        <id name="id">
            <generator class="assigned"/>
        </id>
        <property name=""></property>
    </class>

```



#### HQL-> Hiberante Query Language

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



**DURGASOFT**

```

Transaction t=s.beginTransaction();

//insert

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

Query q=s.createQuery("insert into Student9(id,name,email);
int i=q.executeUpdate();
t.commit();
sf.close();
System.out.println("insert success");

}
}

```

**DURGASOFT**

```
}
```

```
nto Student9(id,name,email,address) select s.id,s.name,
l,s.address from Student8 s");
```

Int i=q.executeUpdate();  
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 column and cannot update primary key

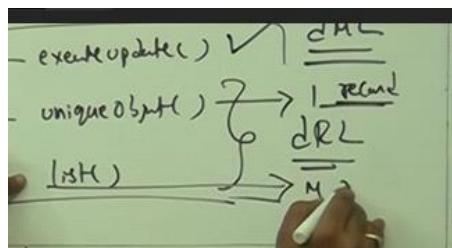
```
Session session=sf.openSession();
Transaction t=session.beginTransaction();
String hql="update Student set name='ABC',email='ABC@y
Query query=session.createQuery(hql);
int i=query.executeUpdate();
System.out.println(i);
session.close();
```

```
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 unique record

① 1 Row

```
String hql = "from Student where id=1";
Query q = s.createQuery(hql);
Object o = q.uniqueResult();
Student st = (Student)o;
```

```
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

② 1 column

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

```
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();
st.close();
```

If we want to get 1 whole column above one

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

① 2 columns

```

String hql = "select name, email from Student";
        name
        sum
        sum
Query q = s.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]);
}

```

Object o[] = new Object[2];

o[0] = "a";

o[1] = "alpha.wi";

DURGASOFT

```

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();

```

DURGASOFT  
www.durgasoft.

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();
sf.close();

```

DURGASOFT  
www.durgasoft.

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

(i) Select All

```
Session s = sf.openSession();
Criteria cr = s.createCriteria(Employee.class);
[Select * from Employee]
List<Employee> lst=cr.list();
for(Employee e:lst){
    System.out.println(e.getId());
    ;
}
```

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

Crieterian means condition

```
Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
[Select * from Employee]
Criteria cr = Restrictions.eq("id", 1);
c.add(cr);
Employee e=(Employee)c.uniqueResult();
```

Restriction means where Id=1

Restrictions.  
eq(=)  
gt(>)  
lt(<)  
between (5000, 10000 salary)  
lo  
hi  
distinct()

```
Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
[Select * from Employee]
Where salary > 8000
Criteria cr = Restrictions.gt("salary", 8000);
c.add(cr);
List<Employee> lst = (List)c.list();
for(Employee emp:lst){
    System.out.println(emp.getId());
}
```

**DURG**  
www.durgsoft.com

```

cfg.configure("resources/oracle.cfg.xml");
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()); DURG

```

Projections->

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

Select name from employee only one column ->Projection.property("name");

Partial record->

```

Session s = sf.openSession();
Criteria c= s.createCriteria(Employee.class);

projection p= projection.property("name");
c.setProjection(p);      select name from employee;

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

```

For one criteria we can apply multiple criteria

But at a time only one projection

```

Session s = sf.openSession();
Criteria c= s.createCriteria(Employee.class);

projection p1= projection.property("email");
projection p= projection.property("name");
c.setProjection(p);
c.setProjection(p1);

```

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

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

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Project p1 = projects.property("email");
Projection p = projections.property("name");
printList pl= projects.printList();
pl.add(p);
pl.add(p1);
c.setProjection(pl);
List<Object> list = c.list();

```

If we want to aggregate or do sum

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Projection p = projects.avg("Salary");
c.setProjection(p);
double avgSal=(Double)c.uniqueResult();

```

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Projection p = projects.max("id");
c.setProjection(p);
int id=(Integer)c.uniqueResult();

```

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

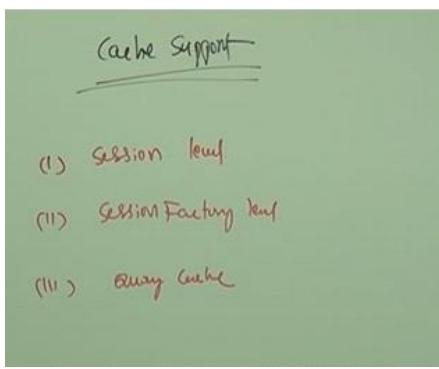
Projection for aggregate function and particular column select operation

```

Configuration cfg=configure("resources/oracle.cfg.xml");
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();
Criteria c = session.createCriteria(Employee.class);
/*Projection p=Projections.avg("salary");
c.setProjection(p);
double avg_salary=(Double)c.uniqueResult();
System.out.println(avg_salary);*/
Projection p=Projections.max("salary");
c.setProjection(p);
double avg_salary=(Double)c.uniqueResult();
System.out.println(avg_salary);

```

Cache support



One user  
All user  
One instance

- (I) Session level
- (II) SessionFactory level
- (III) Query level

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 1 lakh connection and closed.. How much processing will waste

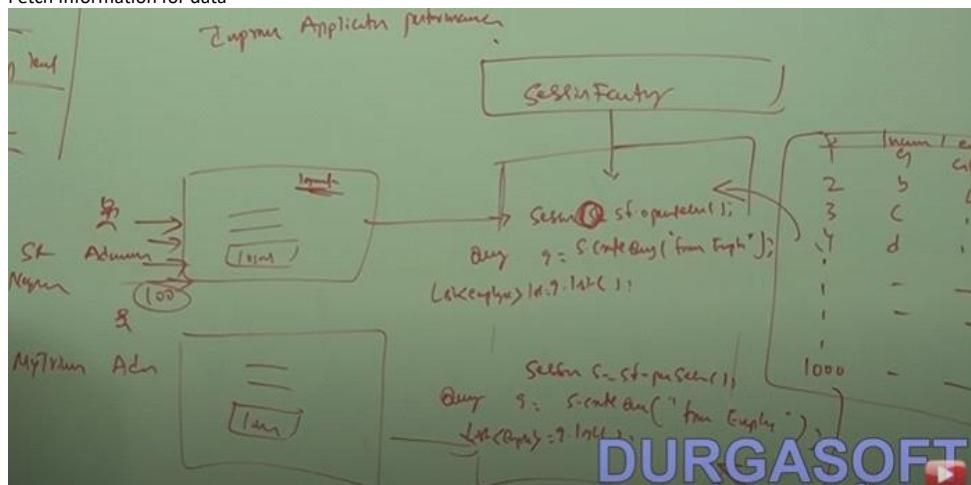
Cache will take temporary storage and it will return 1 lakh time..

Cache reduce database 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 retrieve 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 database

Session factory->

It is also called second level caching

If there are two user then one will hit database then it will available for second user also

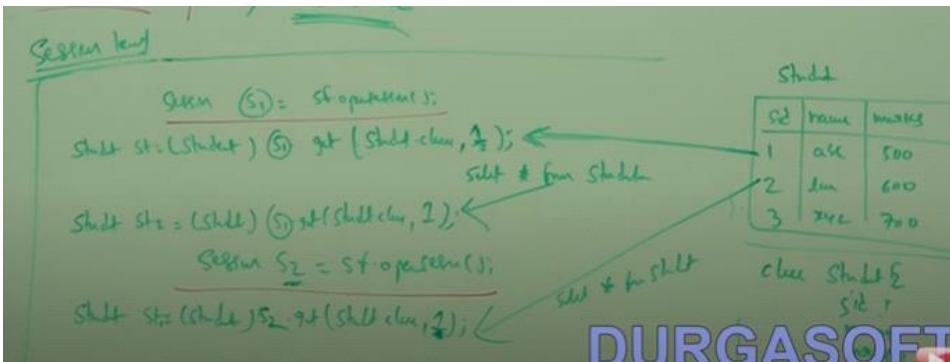
As it is for session factory level

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

```

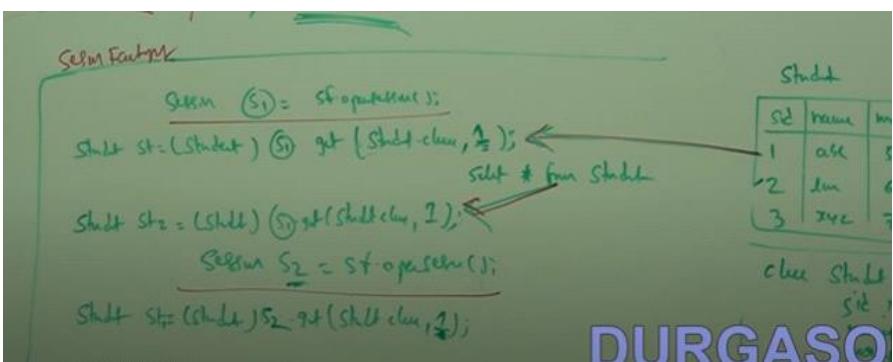
Session s1 = sf.openSession();
Shuttle st=(Shuttle) s1.get(Shuttle.class, 1);

```



Two call for session cache for two session factory

But session factory cache then same data available for both session factory



```

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    updateCheck="false">
    <defaultCache
        maxElementsInMemory="1000"
        timeToLiveSeconds="60"
        eternal="false"
        overflowToDisk="false">
    </defaultCache>
</ehcache>

```

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

```

ehcache.xml oracle.cfg.xml
<property name="connection.pool_size">15</property>

<property name="dialect">org.hibernate.dialect.OracleDialect</property>
<property name="hbm2ddl.auto">update</property>
<property name="show_sql">true</property>
second level -->
<property name="cache.use_second_Level_cache">true</property>
<property name="cache.region.factory_class">org.hibernate.cache
<property name="net.sf.ehcache.configurationResourceName">resources/
ng resource="resources/employee.hbm.xml"/>
Factory>

```

DURGASOFT

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

```
hibernate.cfg.xml   oracle.cfg.xml   student.hbm.xml  21
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.d
<hibernate-mapping>
  <class name="beans.Student" table="student007" schema
    <cache usage="read-write"/>
    <id name="sid"/>
    <property name="sname"/>
    <property name="semail"/>
    <property name="smarks"/>

  </class>

</hibernate-mapping>
```

DURGASOFT

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

System.out.println("using first session");
Student st1 = (Student) session.get(Student.class);
System.out.println(st1.getSname());
System.out.println(st1.getSemail());

Student st2 = (Student) session.get(Student.class);
System.out.println(st2.getSname());
System.out.println(st2.getSemail());

System.out.println("using second session");
Student st3 = (Student) session.get(Student.class);
```

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

But in session factory only one select operation

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

System.out.println("For first query...");

Query q=s.createQuery("select ename from Employee");
q.setCacheable(true);
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);
```

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

```

Client.java Employee.java employee.hbm.xml InsertClient.java oracle.cfg.xml SelectClient.java
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();

```

DURGASOFT  
www.durasoft.co.in

```

<property name="net.sf.ehcache.configurationResourceName" value="ehcache.xml"/>
<property name="cache.use_query_cache">true</property>
<mapping resource="resources/employee.hbm.xml"/>

</session-factory>

```

```

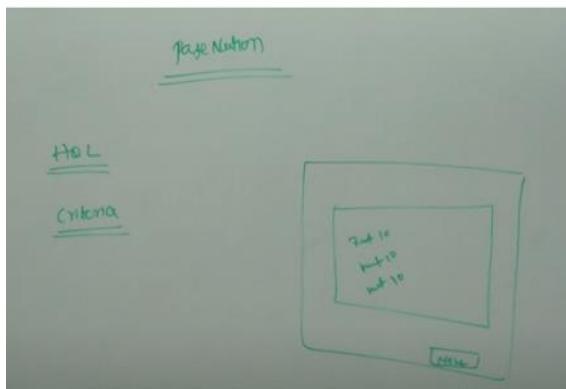
e1.setEid(111);
e1.setEname("abc");
e1.setEmail("abc@gmail.com");
e1.setSalary(500000);

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 retrieve limited value

We want to retrieve only 5 records

```
Session s = sf.openSession();
Query q = s.createQuery("from Student");
q.setFirstResult(0);
q.setMaxResults(5);

int i=1;
int j=5;
for(int k=0; k< q.list().size(); k++)
{
    q.setFirstResult(i);
    q.setMaxResults(j);
}

List<Student> st=q.list();
i=i+5;
j=j+5;
```

```
throws ServletException, IOException {
PrintWriter out=resp.getWriter();
Session s = sf.openSession();
int fr = Integer.parseInt(req.getParameter("fr"));
int mr = Integer.parseInt(req.getParameter("mr"));

Query q = s.createQuery("from Student");

q.setFirstResult(fr);
q.setMaxResults(mr);

List<Student> list=q.list();
for(Student st:list)
{
    out.println("ID="+st.getId()+"\t NAME");
}
```

Criteria pagination

```
PrintWriter out = resp.getWriter();
Session s = sf.openSession();
int fr = Integer.parseInt(req.getParameter("fr"));
int mr = Integer.parseInt(req.getParameter("mr"));

Criteria cr = s.createCriteria(Student.class);
cr.setFirstResult(fr);
cr.setMaxResults(mr);

List<Student> list = cr.list();
for (Student st : list) {
    out.println("ID=" + st.getId() + "\t NAME=" +
    + "\t email=" + st.getEmail() + "\t m");
}

s.close();
```

@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 ehcache exceed from its configured size ehcache removed expired elements

If that doesn't clear enough space it clear object from ehcache

By default it remove LRU elements

Second level caching is we put on entity or whole object

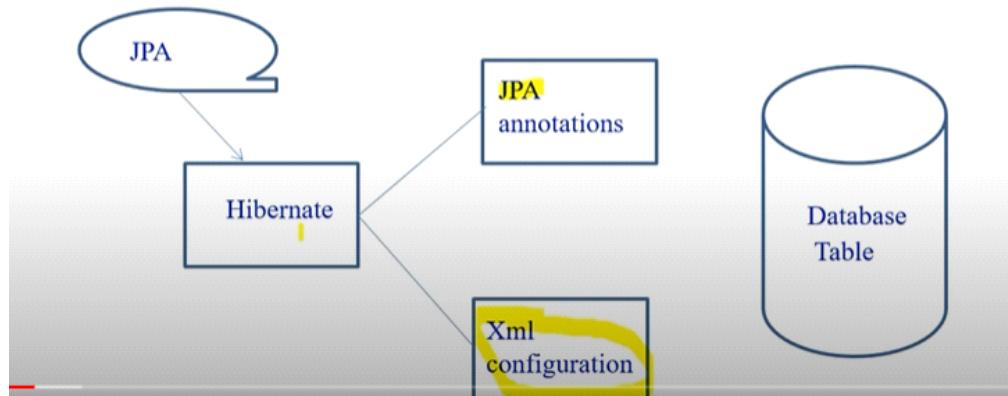
Default cache is made

```
@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.



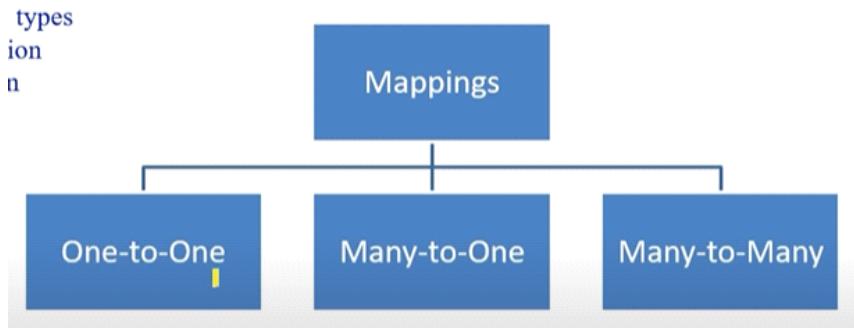
- **Why Hibernate ?**
- Hibernate **eliminates all the boiler-plate code** that comes with JDBC.
- It supports HQL with is more Object oriented.
- It provides transaction management implicitly.
- Hibernate throws **JDBCException** or **HibernateException** which are the unchecked exceptions, so we don't 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(persistentObj);
t.commit();
factory.close();
session.close();
```



# 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.

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;
    }

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

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

public class BuilderPatternExample {

    public 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.

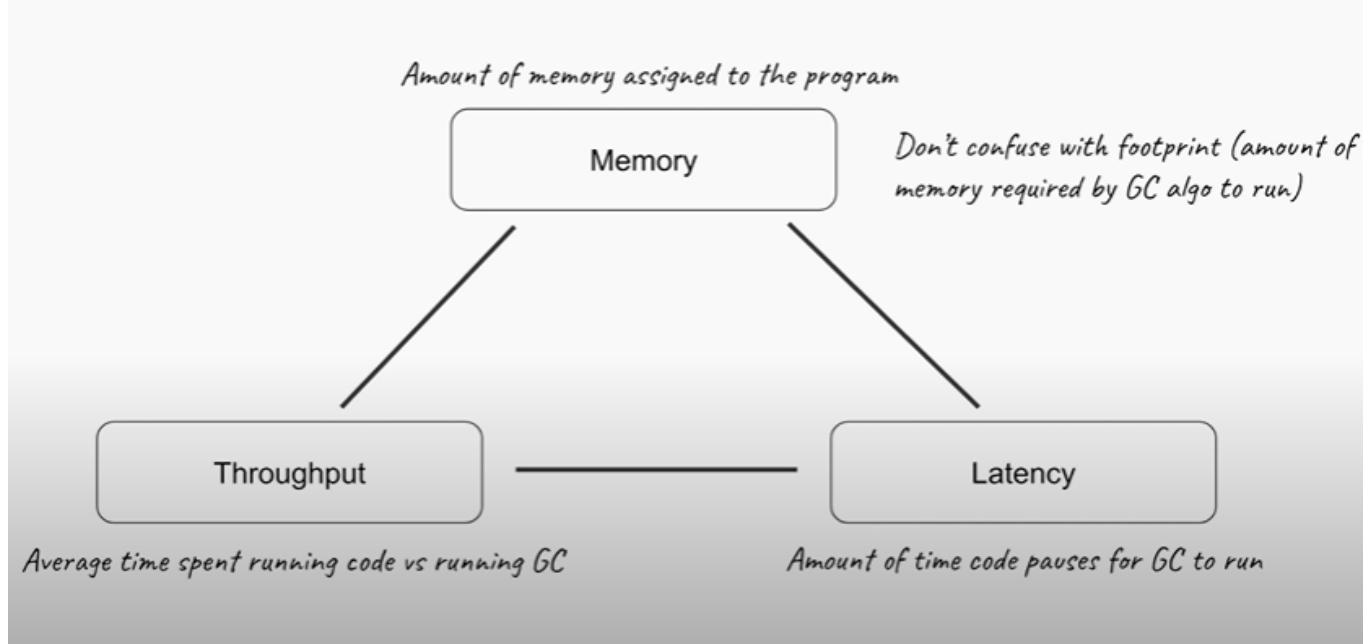
- It is thread safe without synchronizing the whole map.
- Reads can happen very fast while write is done with a lock.
- There is no locking at the object level.

>

## 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

##### 1) HTML (HyperText Markup Language)

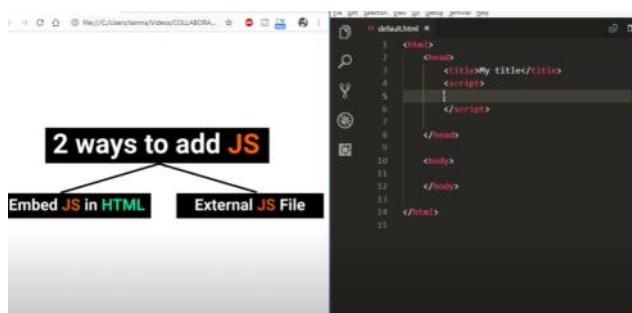
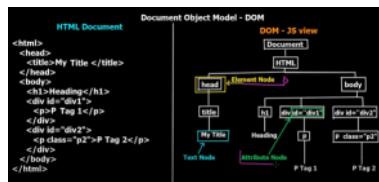
- >> Adds structure to our web pages.
- >> Tags used for e.g. <div>, <section>, <p>

##### 2) CSS (Cascading Style Sheets)

- >> Adds **styles** to our webpages.
- >> E.g. colors, border, margin, image, positions etc.
- >> Can use ids, classes or direct tags to reference HTML tags

##### 3) JS (JavaScript)

- >> Adds programming to our web pages.
- >> Adds functionality, e.g. client side validation, effects & events etc.



```

default.html
1 <html>
2   <head>
3     <title>My Title</title>
4     <script>
5       |
6       |
7       |
8       |
9       |
10    </script>
11
12  </head>
13
14  <body>
15
16    <h1>Hello World</h1>
17
18  </body>
19
20</html>

```

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

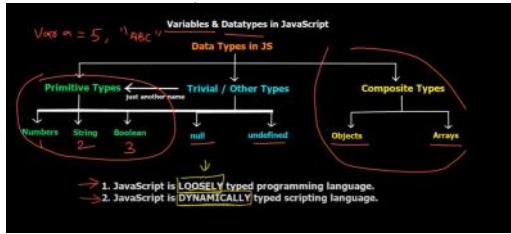
```

demo.js
1 > Users > kumamur > Desktop > JavaScript > demo.js
1 document.write("<h1>External Hello World</h1>");

default.html
1 <html>
2   <head>
3     <title>Muari Kumar</title>
4     <script src="demo.js" type="text/javascript">
5     </script>
6
7
8   </head>
9
10 <body>
11
12   <h1>External Hello World</h1>
13
14 </body>
15
16 </html>

```

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 I

```
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 more 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 method**

```
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;
```

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>

```

```

<Click me>
Murari Learning
<head>
<title>Murari Kumar</title>
<script type="text/javascript">
    function buttonclick()
    {
        var the_document=document.getElementById("first");
        the_document.innerHTML="Murari Learning";
        tb="second";
    }
</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

```

This page says
Value inside the text box Murari
<input id="text1" placeholder="Enter Anything">
<button onclick="fn1()" id="html1">Click me</button>

```

Username & password

```

<head>
    <title>Murari Kumar</title>
    <script type="text/javascript">
        function fn1()
        {
            var user=document.getElementById("user").value;
            pass=document.getElementById("password").value;
            if(user=="Murari" && pass=="9102")
            {
                alert("Hl successful login "+ user);
            }
            else{
                alert("Wrong credential");
            }
        }
    </script>
</head>
<body>
    <input id="user" placeholder="Enter Username">
    <br>
    <input type="password" id="password" placeholder="Password">
</body>

```

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

```

<script type="text/javascript">
//200 manipulations & getelementbyid method/
function fn1()
{
    var rd1 = document.getElementById("rd1");
    var rd2 = document.getElementById("rd2");

    if(rd1.checked==true)
        alert("The channel selected is "+rd1.value);
    else if(rd2.checked==true)
        alert("The channel selected is "+rd2.value);
    else
        alert("No channel selected");
}
</script>
</head>
<body>
    <input id="rd1" name="grp1" type="radio" value="Simple Snippets">
    <br>
    <input id="rd2" name="grp1" type="radio" value="Murari Learning">
    Murari Learning</input>
</body>

```

```

1   <html>
2
3   <head>
4       <title>Murari Kumar</title>
5       <script type="text/javascript">
6           function fn1()
7           {
8               var r1=document.getElementById("select");
9               alert("Pop up "+r1.value);
10          }
11
12      </script>
13  </head>
14
15  <body>
16      <select id="select">
17          <option value="Murari">Murari</option>
18          <option value="Kumar">Kumar</option>
19          <option value="Rahul">Rahul</option>
20      </select>
21      <button onclick="fn1()" id="btn1">On click</button>
22
23
24  </body>
25
26
27 </html>

```

Get element by tag name

```

<html>
  <head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
      /*DOM manipulations & getelementById method*/
      function fn1()
      {
        var select = document.getElementById("selectbox");
        alert(select.options[select.selectedIndex].value)
      }
    </script>
  </head>
  <body>
    <select id="selectbox">
      <option value="Simple Snippets">Simple Snippets</option>
      <option value="Telusko Learnings">Telusko Learnings</option>
      <option value="MKBHD">MKBHD</option>
    </select>
    <br>
    <button onclick="fn1()">Click Me</button>
  </body>
</html>

```

Simple Snippets

Click Me

```

<html>
  <head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
      /*DOM manipulations & getelementByTagName method*/
      function changeStyling()
      {
        var para = document.getElementsByTagName("");
      }
    </script>
  </head>
  <body>
    <p>This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p>This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p>This is paragraph 5</p>
    <br>
    <a href="#">This is anchor tag</a>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <br>
    <button onclick="changeStyling()">Click Me</button>
  </body>
</html>

```

This is paragraph 1

This is paragraph 2

This is paragraph 3

This is paragraph 4

This is paragraph 5

This is anchor tag

Click Me

```

<html>
  <head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
      /*DOM manipulations & getelementsByTagName method*/
      function changeStyling()
      {
        var para = document.getElementsByTagName("p");
        para[0].style.fontSize = "25px";
        para[1].style.color = "#ee0000";
        para[2].style.fontWeight = "bold";
        para[3].style.fontStyle = "italic";
      }
    </script>
  </head>
  <body>
    <p>This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p>This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p>This is paragraph 5</p>
    <br>
    <button onclick="changeStyling()">Click Me</button>
  </body>
</html>

```

This is paragraph 1

This is paragraph 2

This is paragraph 3

This is paragraph 4

This is paragraph 5

Click Me

```

<html>
  <head>
    <title>DOM manipulations & getelementById method</title>
    <script type="text/javascript">
      /*DOM manipulations & getelementById method*/
      function changeStyling()
      {
        var tag = document.getElementsByTagName("p");
        var element = document.getElementsById("mypara");
        for(var x=0;x<element.length;x++)
        {
          element[x].style.color="red";
        }
      }
    </script>
  </head>
  <body>
    <p id="p1" class="mypara">This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p class="mypara">This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p class="mypara">This is paragraph 5</p>
    <a class="mypara" href="#">This is Anchor tag</a>
    <br>
    <button onclick="changeStyling()">Click Me</button>
  </body>
</html>

```

This is paragraph 1

This is paragraph 2

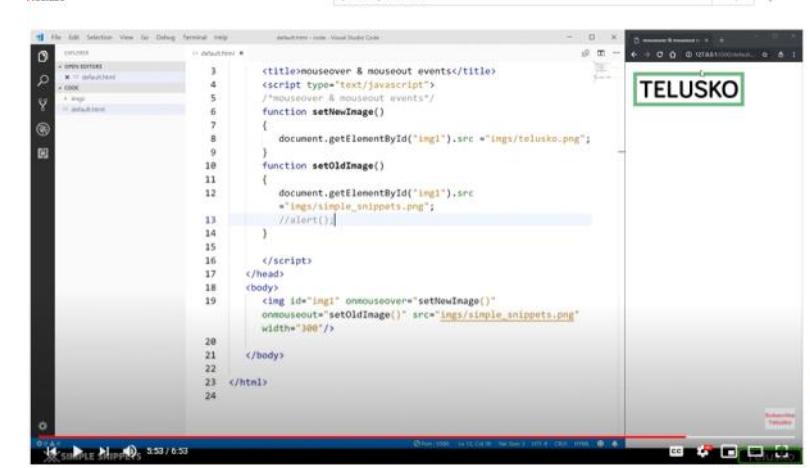
This is paragraph 3

This is paragraph 4

This is paragraph 5

This is Anchor tag

Click Me



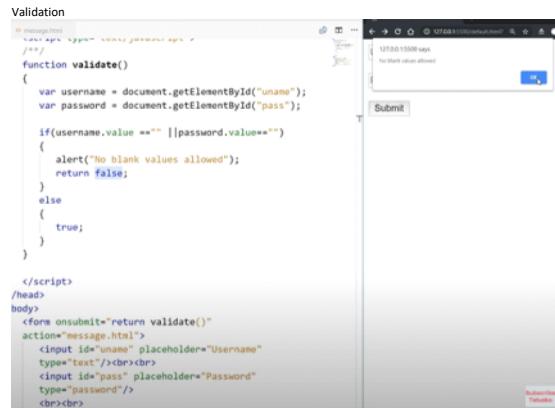
```

<html>
  <head>
    <title>mouseover & mouseout events</title>
    <script type="text/javascript">
      /*mouseover & mouseout events*/
      function setNewImage()
      {
        document.getElementById("img1").src ="imgs/telusko.png";
      }
      function setOldImage()
      {
        document.getElementById("img1").src
        ="imgs/simple_snippets.png";
        //alert();
      }
    </script>
  </head>
  <body>
    
  </body>
</html>

```

TELUSKO

#25 Change Image onmouseover and onmouseout events in JavaScript



```

<html>
  <head>
    <title>Validation</title>
    </head>
    <body>
      <script type="text/javascript">
        function validate()
        {
          var username = document.getElementById("uname");
          var password = document.getElementById("pass");

          if(username.value == "" || password.value == "")
          {
            alert("No blank values allowed");
            return false;
          }
          else
          {
            true;
          }
        }
      </script>
    </body>
    <form onsubmit="return validate()" action="message.html">
      <input id="uname" placeholder="Username" type="text"><br><br>
      <input id="pass" placeholder="Password" type="password">
      <br><br>
    </form>
  </body>
</html>

```

On submit extra feature

The screenshot shows a browser window with a login form. The form has two fields: 'Username' and 'Password', and a 'Submit' button. Below the form, there is a red error message: 'Invalid Username'.

```

<html>
<head>
    <title>Regular Expressions in JS </title>
    <script type="text/javascript">
        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;
            }
        }
    </script>
</head>
<body>
    <form action="message.html">
        <input id="uname" placeholder="Username" type="text"/><br>
        <label id="blouser" style="color: red; visibility: hidden">Invalid Username</label>
        <br>
        <input id="pass" type="password"/>
        <br>
        <input type="submit" value="Submit" />
    </form>
</body>

```

On the right side of the browser, there is a developer tools panel showing the 'REGULAR EXPRESSION' tab. It displays the regex pattern `/^\w{3,10}/` and a test string `7656565656`. The status bar at the bottom of the browser says 'Invalid Username'.

# 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);
    }
});
```

Lambdas 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 concise. We have to keep multiple context in mind, what `e` is and what `i` is and what we are comparing the two. We lost the general benefits of lambdas—concise and expressive—this code fails both.

## 2. It's noisy

Writing a single line lambda was nice. The minute we wanted to do more than one line, we have to bring the `{}`, the `,`, and possibly the `return` if we had something to return. That's Java's (and C#'s too) way of telling us—it's telling us not to do that.

## 3. Leads to Duplication

Whatever that multiline code does, it's not reusable. If we want to

Method reference-> 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 reference

```

// 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)));
}

```

```

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 -- FEMALE -- 20, Sara -- FEMALE -- 22].

```

```

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
    //name

    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 completablefuture

```

| Account.java ✘
1 package threadingexample;
2
3 public class Account {
4     private int balance=20000;
5     public int getbalance()
6     {
7         return balance;
8     }
9     public void withdraw(int val)
10    {
11        this.balance=this.balance-val;
12    }
13
14 }

| Client.java ✘
1 package threadingexample;
2
3 public class Client {
4     public static void main(String[] args) {
5         Account account= new Account();
6         AccountHolder holder= new AccountHolder(account);
7         Thread t1= new Thread(holder);
8         Thread t2= new Thread(holder);
9         t1.setName("Murari");
10        t2.setName("kumar");
11        t1.start();
12        t2.start();
13    }
14
15
16 }

| *AccountHolder.java ✘
10
11     @Override
12     public void run() {
13         for (int i = 1; i < 5; i++) {
14             makewithdrawable(2000);
15             if (account.getbalance() < 0) {
16                 System.out.println("Already taken....." + Thread.currentThread().getName());
17             }
18         }
19
20
21     private synchronized void makewithdrawable(int i) { // protect mu
22         if (account.getbalance() >= i) {
23             System.out.println(
24                 Thread.currentThread().getName() + " " + "Current
25             try {

```

If not synchronized 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);
    }
});
```

Lambdas 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 concise, and we have to keep multiple context in mind, what `e` is and what `i` is and what we are comparing the two. We lost the general benefits of lambdas—concise and expressive—this code fails both.

## 2. It's noisy

Writing a single line lambda was nice. The minute we wanted to do more than one line, we have to bring the `{}`, the `,`, and possibly the `return` keyword if we had something to return. That's Java's (and C#'s too) way of telling us—it's telling us not to do that.

## 3. Leads to Duplication

Whatever that multiline code does, it's not reusable. If we want to

Method reference-> 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 reference

```

// 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)));
}

```

```

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 -- FEMALE -- 20], [Sara -- FEMALE -- 22].

```

```

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
    //name

    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 completablefuture

```

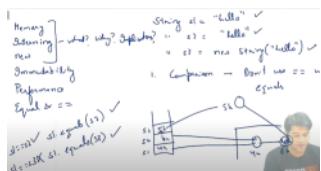
Account.java
1 package threadingexample;
2
3 public class Account {
4     private int balance=20000;
5     public int getbalance()
6     {
7         return balance;
8     }
9     public void withdraw(int val)
10    {
11        this.balance=this.balance-val;
12    }
13 }
14

Client.java
1 package threadingexample;
2
3 public class Client {
4     public static void main(String[] args) {
5         Account account= new Account();
6         AccountHolder holder= new AccountHolder(account);
7         Thread t1= new Thread(holder);
8         Thread t2= new Thread(holder);
9         t1.setName("Murari");
10        t2.setName("kumar");
11        t1.start();
12        t2.start();
13    }
14
15
16 }

AccountHolder.java
10
11     @Override
12     public void run() {
13         for (int i = 1; i < 5; i++) {
14             makewithdrawable(2000);
15             if (account.getbalance() < 0) {
16                 System.out.println("Already taken....." + Thread.currentThread().getName());
17             }
18         }
19     }
20
21     private synchronized void makewithdrawable(int i) { // protect mu
22         if (account.getbalance() >= i) {
23             System.out.println(
24                 Thread.currentThread().getName() + " " + "Current
25             try {

```

If not synchronized all thread access at same time. If synchronized it will lock thread  
<https://www.mockaroo.com/> To mock data



## What Spring MVC does?

Spring MVC will help you to develop java J2EE (web) applications easily.

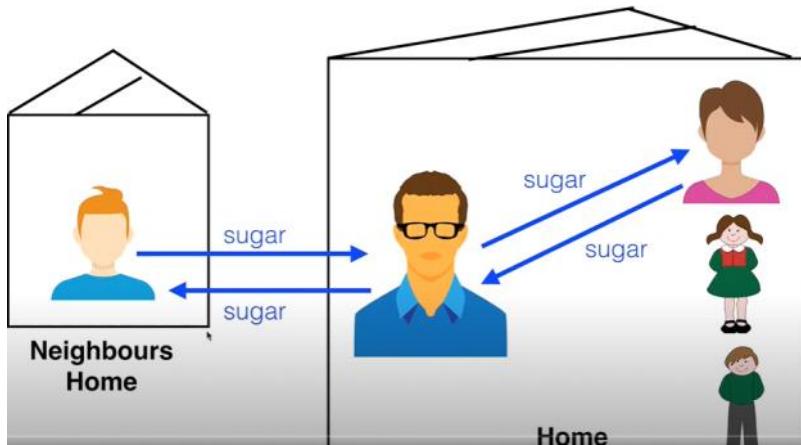
But the **problem** is setting up your project (or working environment) manually by doing a lot of configurations by using **XML** etc.

## What Spring boot does?

Spring boot helps us to wrap all spring components in a convenient way where no external XML configuration is needed.

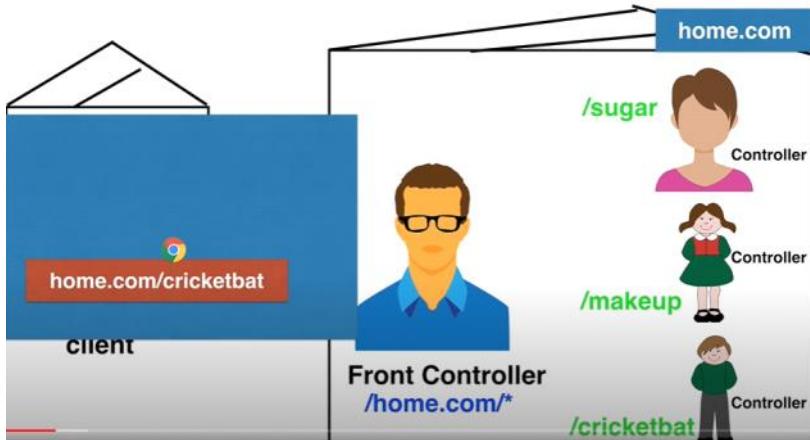
so the spring boot wraps spring MVC itself.

### Front CONTROLLER



Any request come outside the home dad knows the available 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

**Spring will automatically  
initialize the class having a  
@controller  
annotation and register that class  
with the spring container.**

```

@Controller
public class MomController {

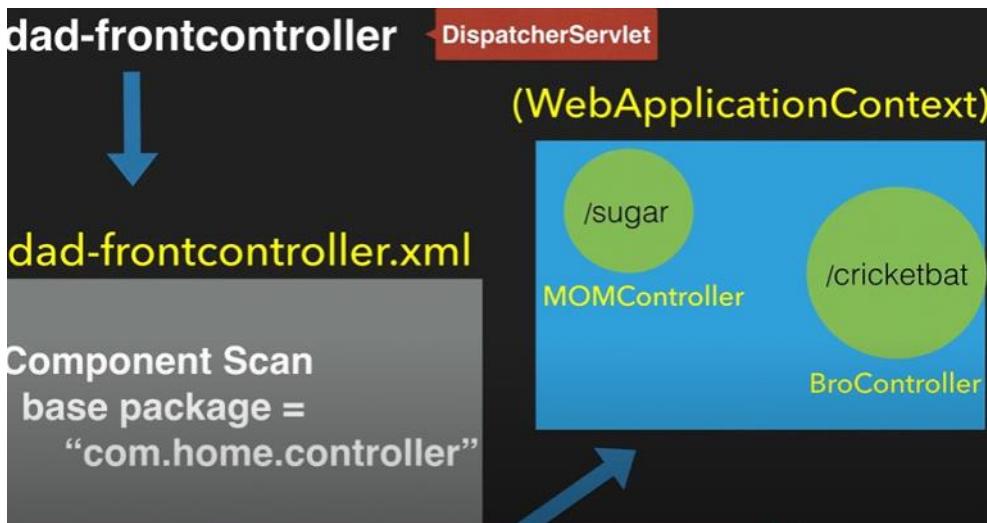
    @ResponseBody
    @RequestMapping("/sugar")
    public String giveSugar()
    {
        return "Ok..Here is your sugar";
    }
}

```

@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 appended with .xml



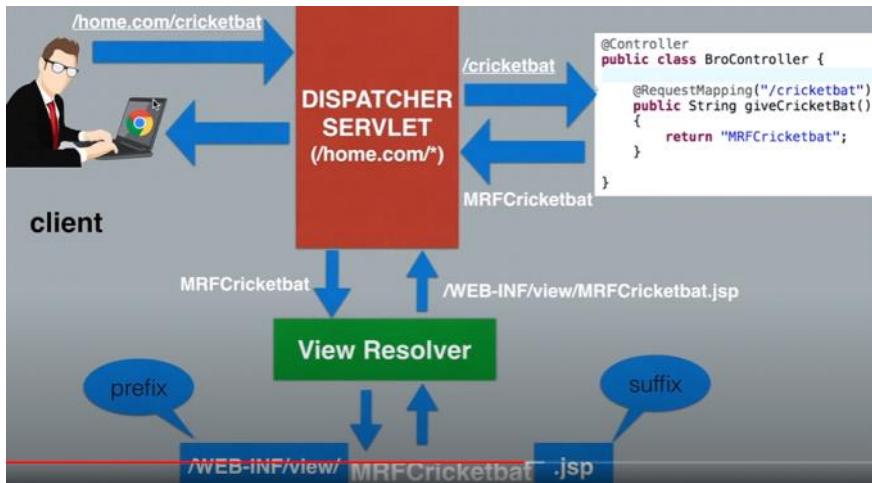
```
@Controller
public class BroController {
    @ResponseBody
    @RequestMapping("/cricketbat")
    public String giveCricketBat()
    {
        return "MRFCricketbat";
    }
}
```

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

But if we remove the @Response body then controller 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



```

class InternalResourceViewResolver
{
    private String prefix = "";
    private String suffix = "";

    public void setPrefix(String prefix)
    {
        this.prefix = prefix;
    }
    public void setSuffix(String suffix)
    {
        this.suffix = suffix;
    }
}

InternalResourceViewResolver viewResolver = new InternalResourceViewResolver();
viewResolver.setPrefix("/WEB-INF/view/");
viewResolver.setSuffix(".jsp");

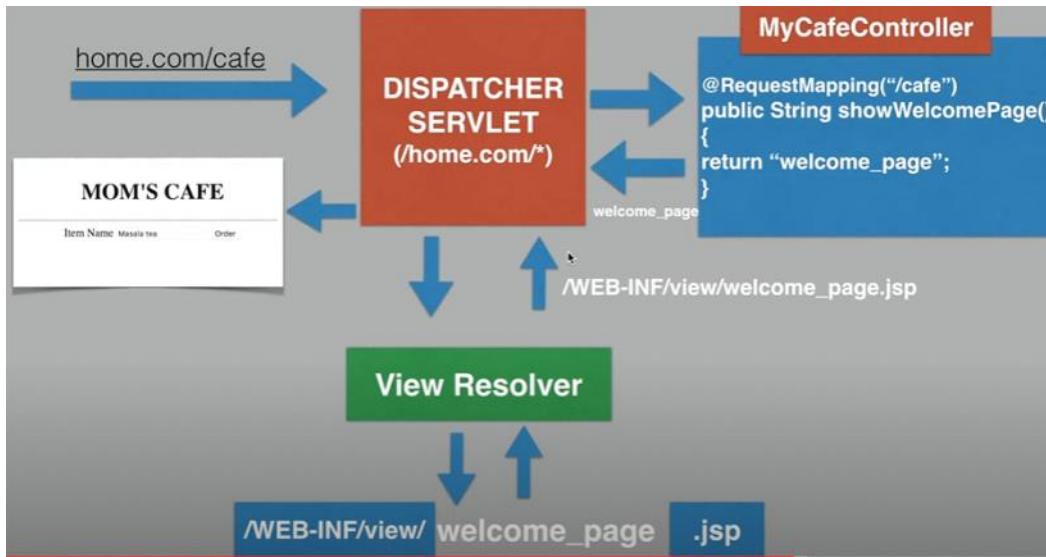
```

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

```

7 @Controller
8 public class MyCafeControllers {
9
.0= 10     @RequestMapping("/cafe")
11     public String showWelcomePage(Model model)
12     {
13         //sending data to view(jsp page)
14         String myName = "Abhilash";
15
16         model.addAttribute("myNameValue", myName);
17
18     return "welcom-page";
19 }

```



```

@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";
}

```

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;
    }

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

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

public class BuilderPatternExample {

    public 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.

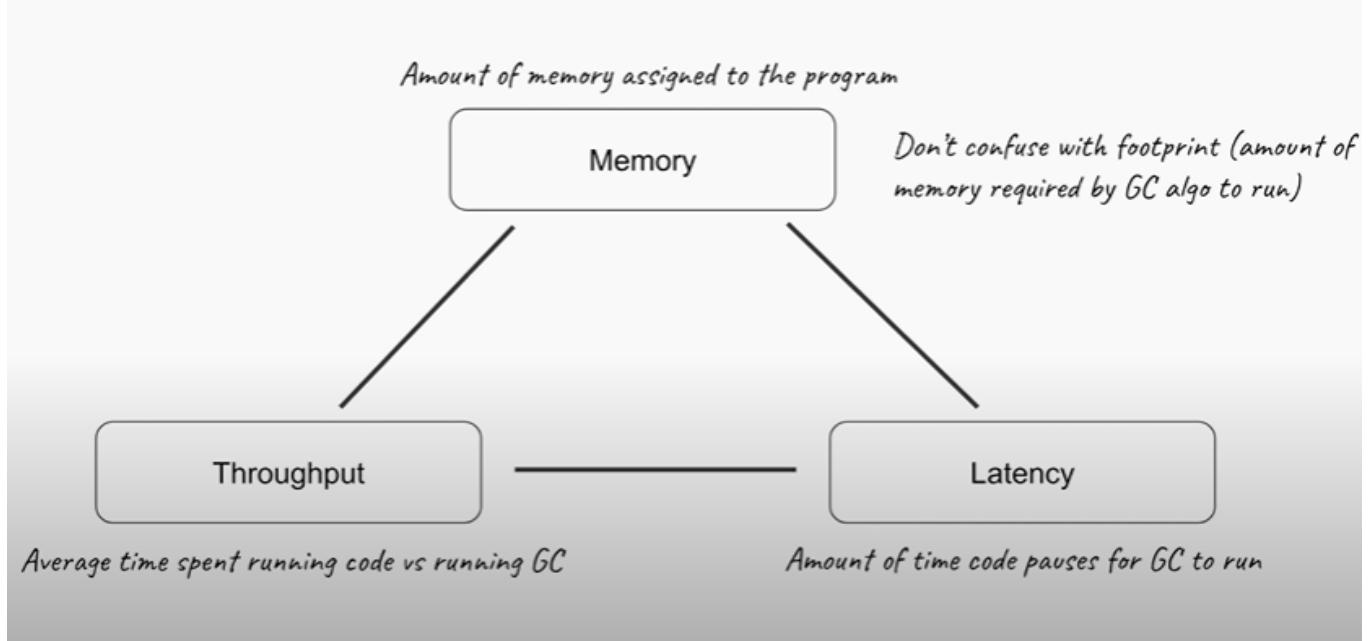
- It is thread safe without synchronizing the whole map.
- Reads can happen very fast while write is done with a lock.
- There is no locking at the object level.

>

## 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.

Strange  
type to memory  
DBE & DSU  
operations  
on  
diff  
types  
done

$$(1111)' = (111) + 0001 \\ = \begin{array}{c} 0000 \\ 0001 \\ \hline 0001 \end{array}$$

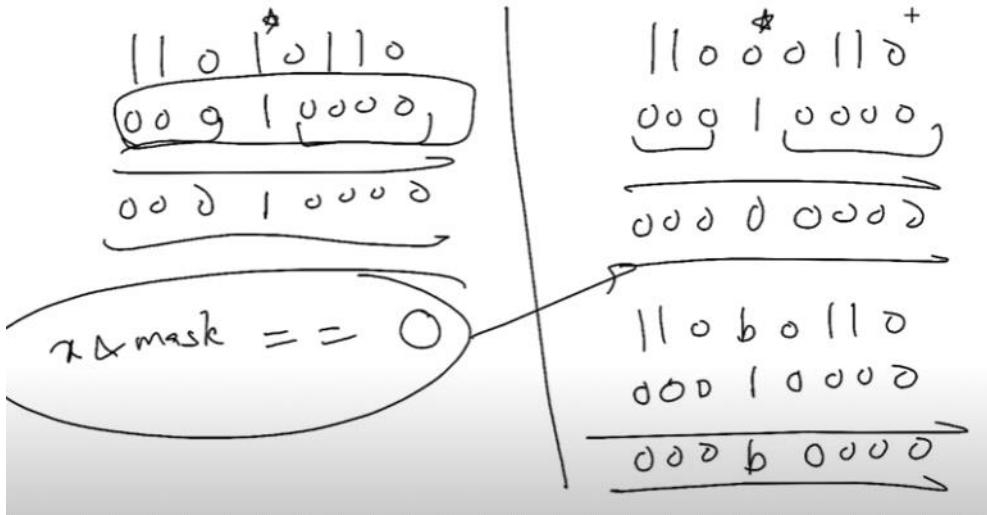
$0\ 000$	$\rightarrow 0$	$A1 -$
$0\ 001$	$\rightarrow 1$	
$0\ 010$	$\rightarrow 2$	$A2 -$
$0\ 011$	$\rightarrow 3$	
$0\ 100$	$\rightarrow 4$	
$0\ 101$	$\rightarrow 5$	$A3$
$0\ 110$	$\rightarrow 6$	
$0\ 111$	$\rightarrow 7$	
$1\ 000$	$\rightarrow 8$	$-0 \quad -8$
$1\ 001$	$\rightarrow 9$	$-1 \quad -7$
$1\ 010$	$\rightarrow 10$	$-2 \quad -6$
$1\ 011$	$\rightarrow 11$	$-3 \quad -5$
$1\ 100$	$\rightarrow 12$	$-4 \quad -4$
$1\ 101$	$\rightarrow 13$	$-5 \quad -3$
$1\ 110$	$\rightarrow 14$	$-6 \quad -2$
$1\ 111$	$\rightarrow 15$	$-7 \quad -1$

$$>> y = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}}$$

$$y >> 3 = \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$$

$$y >>> 3 = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}}$$

On	for	off	& and	toggle	nxor	check
$x = \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$		$x = \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$				$\underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$
$mask = \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}}$		$mask = \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{1}}$				
$(mask \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}})$		$(x \& mask) \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}} \underline{\underline{0}} \underline{\underline{1}}$		$\underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{0}} \underline{\underline{1}}$		
$0 \rightarrow X$		$0 \rightarrow \checkmark$				
$1 \rightarrow \checkmark$		$1 \rightarrow X$				



```
//write your code here
int rsbm = n & -n;
System.out.println(Integer.toBinaryString(rsbm))
```

Most significant one  $\rightarrow$  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

Handwritten note above the grid:  $J10 \# J21$

Key below the grid:

key	0#0	1#-1	2#-2	2#-1	1#-1					
	-1	0	1	2						

#ashmaps #datastructure #algorithms

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

12 views • Oct 25, 2020

0	101	—5
0	110	—6
0	111	—7
1	000	—8
1	001	—9
1	010	—10
1	011	—11
1	100	—12
1	101	—13

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

Sum of value

```
Scannersc=newScanner(System.in);
intsum=0;
for(inti=0;i<4;i++){
System.out.println("Enter value");
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 remainder  
15|4  
11| 11

If divide by 8 last 3 will be remainder  
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

$$n << x$$

$$n \neq 2^x$$

$$n \gg x$$

$$\frac{n}{2^x} +$$

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 communication-> 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;

import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Counter implements Runnable{
    static int id=0;
    int count;

    public Counter(){
        this.count=0;
    }

    public synchronized int getCount() throws InterruptedException{
        wait();
        this.count=count+1;
        System.out.println(count);
        return count;
    }

    @Override
    public void run(){
        try{
            getCount();
        }catch(Exception e){
        }
    }
}

class Counter1 implements Runnable{
    static int id=0;
    int count;

    public Counter1(){
        this.count=0;
    }
}
```

```
}

public synchronized void getCount() throws InterruptedException{

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

notify();
}
}

@Override
public void run(){
try{
getCount();

} catch(Exception e){

}
}
}

public class Application{
public static void main(String[] args) throws InterruptedException{
ExecutorService executorService=Executors.newFixedThreadPool(4);
executorService.execute(new Thread(new Counter()));
executorService.execute(new Thread(new Counter1()));
}
}
```

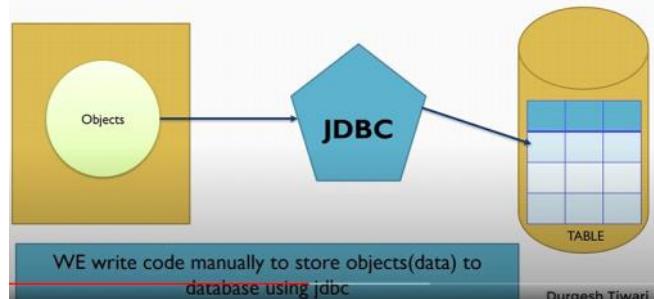
## Hibernate

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

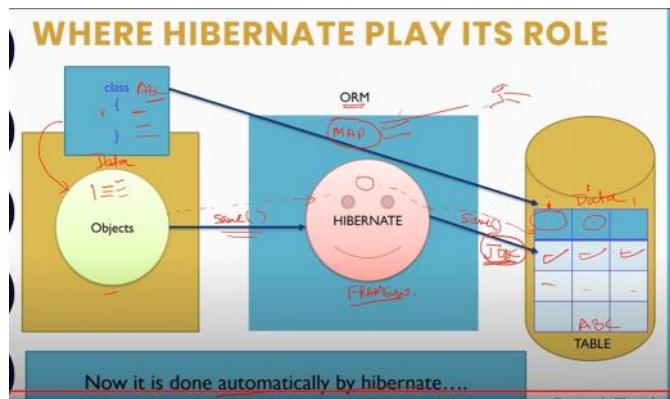
We don't need to worry about sql query it will be performed by hibernate

- 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 force 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**.

## TRADITIONAL WAY TO SAVE DATA(JDBC)



Dinesh Tiwari



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 hibernate

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

III. Configuration  
IV. test class

```

package beans;
public class Student {
    private int id;
    " String name;
    " " email;
    " int marks;
    || public setters & getters
}

```

Student.hbm.xml

```

DTD
<hibernate-mapping>
    <class name="beans.Student" table="Student" schema="System">
        PK <id name="id" column="sid"/>
        <property name="name" column="sname"/>
        <property name="email" column="email"/>
        <property name="marks" column="marks"/>
    </class>
</hibernate-mapping>

```

```

package beans;
public class Student {
    private int id;
    " String name;
    " " email;
    " int marks;
    || public setters & getters
}

```

pojo class

```

St=new Student();
St.setId(111);
St.setName("ABC");
St.setEmail("abc@gmail.com");
St.setMarks(500);

```

Student.hbm.xml

```

<class name="beans.Student" schema="System">
    PK <id name="id" column="sid"/>
    <property name="name" column="sname"/>
    <property name="email" column="email"/>
    <property name="marks" column="marks"/>
</class>
</hibernate-mapping>

```

Configuration cfg = new Configuration();
cfg.configure();
SessionFactory sf = cfg.buildSessionFactory();
Session s = sf.openSession();
s.save(st);

**DURGASOFT**  
[www.durgasoft.com](http://www.durgasoft.com)

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

```

<property name="dialect"> org.hibernate.dialect.OracleDialect </property>
<mapping resource="Student.hbm.xml"/>

```

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

```

Student st=new Student();
st.setId(111);
st.setName("abc");
st.setEmail("abc@gmail.com");
st.setMarks(500);
//student object state is transient
Configuration cfg=new Configuration();
cfg.configure("resources/hibernate.cfg.xml");
SessionFactory sf=cfg.buildSessionFactory();
Session s=sf.openSession();
s.save(st);
//student object state is persistant
s.beginTransaction().commit();
//student object will move database
s.evict(st);
//student object will be remove from database
//then gc can collect ur student object

```

**DURGASOFT**  
[www.durgasoft.com](http://www.durgasoft.com)

a hibernate command Hbm2ddl.auto we can do DDL operation

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

```
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-mapping-3.

  <class name="beans.Student" table="student" schema
    <id name="id" column="sid"/>
    <property name="name" column="sname"/>
    <property name="email" column="semail"/>
    <property name="marks" column="smarks"/>
  </class>

</hibernate-mapping>
```

```
<property name="connection.driver_class">oracle.jd-
<property name="connection.url">jdbc:oracle:thin:@
<property name="connection.username">system</prope
<property name="connection.password">manager</prop
<property name="connection.pool_size">10</property>

<property name="dialect">org.hibernate.dialect.Ora
<property name="hbm2ddl.auto">create</property>

<mapping resource="resources/student.hbm.xml"/>
<mapping resource="resources/course.hbm.xml"/>
<mapping resource="resources/employee.hbm.xml"/>
<mapping resource="resources/department.hbm.xml

/session-factory>
```

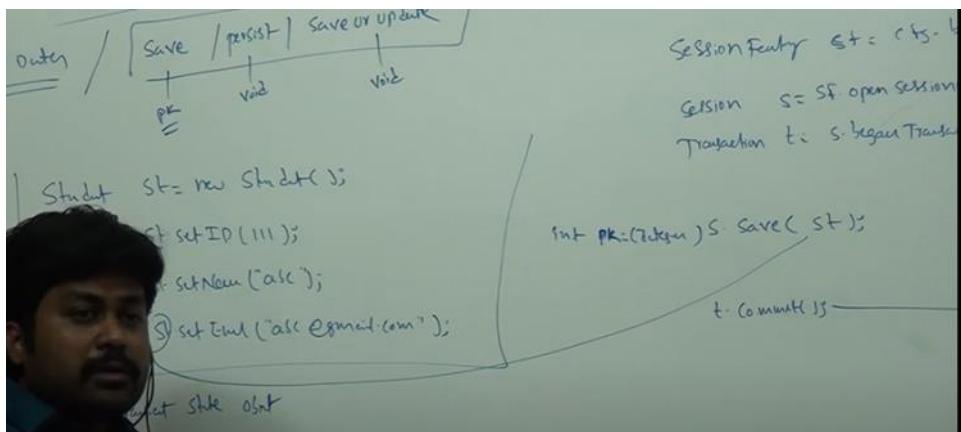
DURGASOFT

```
ty name="hbm2ddl.auto" >create</property>
ty name="show_sql" >true</property>
```

Internal table creation

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

Validate  
Must have that 5 columns



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

Serializable

```
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 fil= new FileInputStream(f);
    ObjectInputStream dli= new ObjectInputStream(fil);
    Save sl=(Save) dli.readObject();
    System.out.println(sl.id);

}

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

If we execute saveOrUpdate(st) operation

First it will run select query on that primary id;

Then it will compare

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

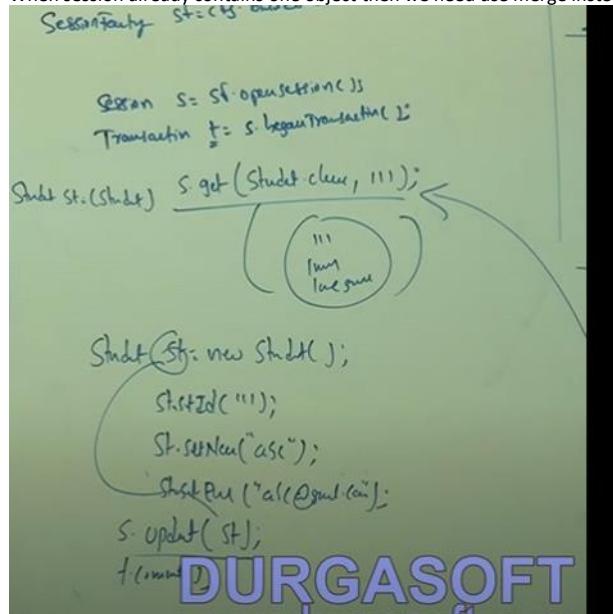
Limitation

We can only update non primary keys

It is not possible to update one column

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 generated

1.

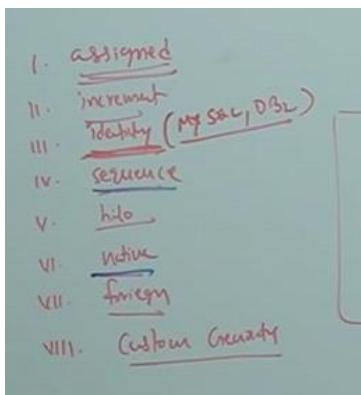
Assigned (default)-> It is handle by user itself

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

```

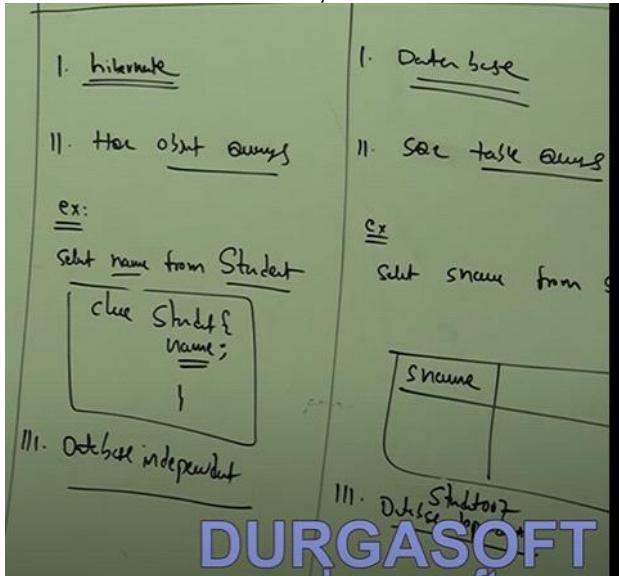
<hibernate-mapping>
    <class name="beans.BookMovie" table="tbc">
        <id name="id">
            <generator class="assigned"/>
        </id>
        <property name=""></property>
    </class>

```



#### 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 Student9(id,name,email);
int i=q.executeUpdate();
t.commit();
sf.close();
System.out.println("insert success");

}
}

```

**DURGASOFT**

```
}
```

```
nto Student9(id,name,email,address) select s.id,s.name,
l,s.address from Student8 s");
```

Int i=q.executeUpdate();  
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 column and cannot update primary key

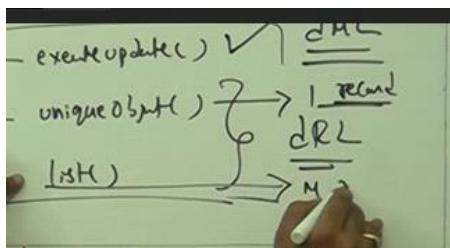
```
Session session=sf.openSession();
Transaction t=session.beginTransaction();
String hql="update Student set name='ABC',email='ABC@y
Query query=session.createQuery(hql);
int i=query.executeUpdate();
System.out.println(i);
session.close();
```

```
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 unique record

① 1 Row

```
String hql = "from Student where id=1";
Query q = s.createQuery(hql);
Object o = q.uniqueResult();
Student st = (Student)o;
```

```
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

② 1 column

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

```
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();
st.close();
```

If we want to get 1 whole column above one

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

① 2 columns

```

String hql = "select name, email from Student";
        name
        sum
        sum
Query q = s.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]);
}

```

Object o[] = new Object[2];

o[0] = "a";

o[1] = "alpha.wi";

DURGASOFT

```

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();

```

DURGASOFT  
www.durgasoft.

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();
sf.close();

```

DURGASOFT  
www.durgasoft.

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

(i) Select All

```
Session s = sf.openSession();
Criteria cr = s.createCriteria(Employee.class);
List<Employee> lst = cr.list();
for(Employee e : lst) {
    System.out.println(e.getId());
}
```

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

Crieterian means condition

```
Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
c.add(Select * from Employee);

Criteria cr = Restrictions.eq("id", 1);
c.add(cr);
Employee e = (Employee)c.uniqueResult();
```

Restriction means where Id=1

Restrictions.eq(=)  
gt(>)  
lt(<)  
between (5000, 10000 salary)  
lo hi  
distinct()

```
Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
c.add(Select * from Employee);

Where salary > 8000
Criteria cr = Restrictions.gt("salary", 8000);
c.add(cr);
List<Employee> lst = (List)c.list();
for(Employee emp : lst) {
    System.out.println(emp.getId());
}
```

DURG  
www.durgsoft.com

```

SessionFactory cfg=configure("resources/oracle.cfg.xml");
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()); DURG

```

Projections->

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

Select name from employee only one column ->Projection.property("name");

Partial record->

```

Session s = sf.openSession();
Criteria c= s.createCriteria(Employee.class);

projection p= projection.property("name");
c.setProjection(p);      select name from employee;

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

```

For one criteria we can apply multiple criteria

But at a time only one projection

```

Session s = sf.openSession();
Criteria c= s.createCriteria(Employee.class);

projection p1= projection.property("email");
projection p= projection.property("name");
c.setProjection(p);
c.setProjection(p1);

```

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

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

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Project p1 = projects.property("email");
Projection p = projections.property("name");
printList pl= projects.printList();
pl.add(p);
pl.add(p1);
c.setProjection(pl);
List<Object> list = c.list();

```

If we want to aggregate or do sum

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Projection p = projects.avg("Salary");
c.setProjection(p);
double avgSal=(Double)c.uniqueResult();

```

```

Session s = sf.openSession();
Criteria c = s.createCriteria(Employee.class);
Projection p = projects.max("id");
c.setProjection(p);
int id=(Integer)c.uniqueResult();

```

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

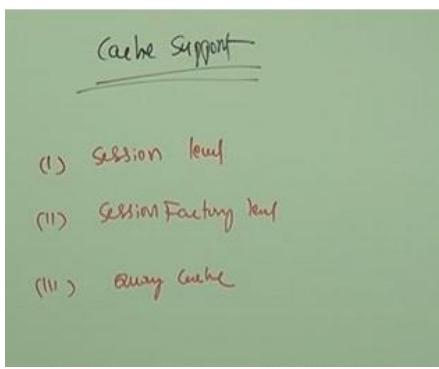
Projection for aggregate function and particular column select operation

```

Configuration cfg=configure("resources/oracle.cfg.xml");
SessionFactory sf = cfg.buildSessionFactory();
Session session = sf.openSession();
Criteria c = session.createCriteria(Employee.class);
/*Projection p=Projections.avg("salary");
c.setProjection(p);
double avg_salary=(Double)c.uniqueResult();
System.out.println(avg_salary);*/
Projection p=Projections.max("salary");
c.setProjection(p);
double avg_salary=(Double)c.uniqueResult();
System.out.println(avg_salary);

```

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 1 lakh connection and closed.. How much processing will waste

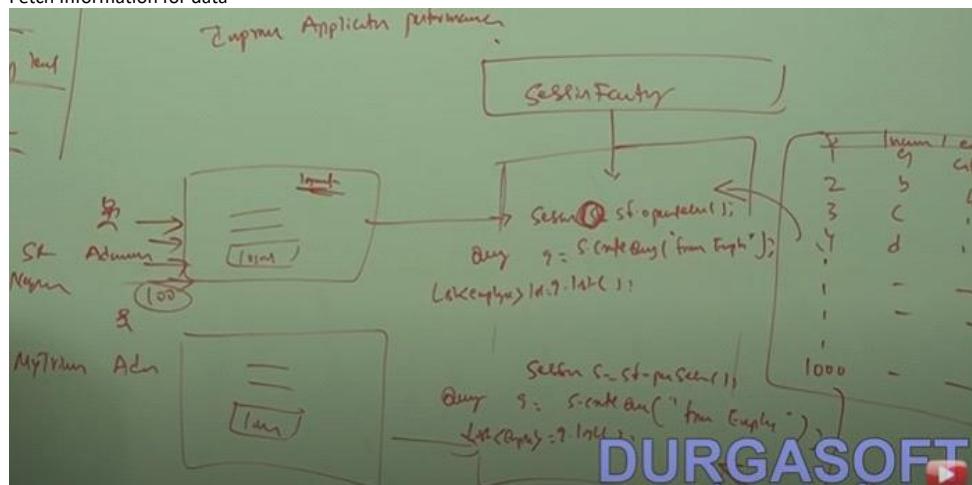
Cache will take temporary storage and it will return 1 lakh time..

Cache reduce database 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 retrieve 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 database

Session factory->

It is also called second level caching

If there are two user then one will hit database then it will available for second user also

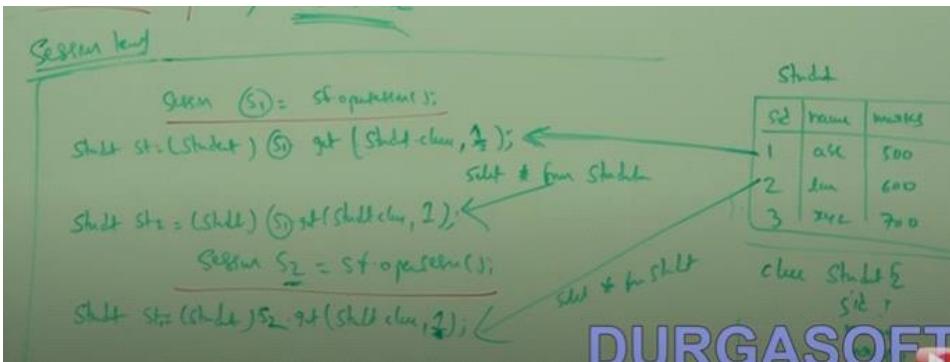
As it is for session factory level

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

```

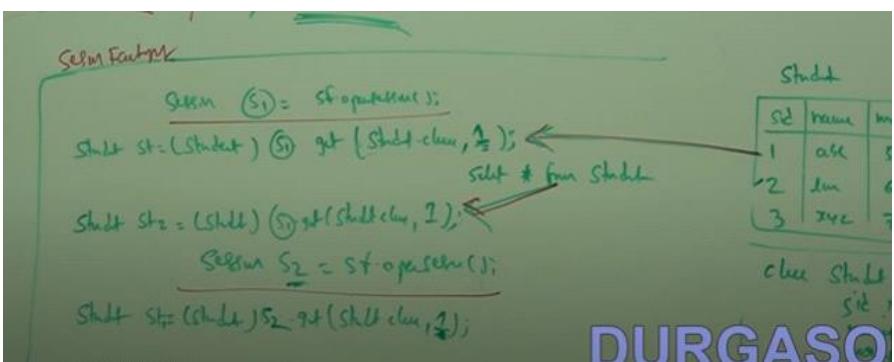
Session s1 = sf.openSession();
Shuttle st=(Shuttle) s1.get(Shuttle.class, 1);

```



Two call for session cache for two session factory

But session factory cache then same data available for both session factory



```

<ehcache xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    updateCheck="false">
    <defaultCache
        maxElementsInMemory="1000"
        timeToLiveSeconds="60"
        eternal="false"
        overflowToDisk="false">
    </defaultCache>
</ehcache>

```

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

```

ehcache.xml oracle.cfg.xml
<property name="connection.pool_size">15</property>

<property name="dialect">org.hibernate.dialect.OracleDialect</property>
<property name="hbm2ddl.auto">update</property>
<property name="show_sql">true</property>
second level -->
<property name="cache.use_second_Level_cache">true</property>
<property name="cache.region.factory_class">org.hibernate.cache
<property name="net.sf.ehcache.configurationResourceName">resources/
ng resource="resources/employee.hbm.xml"/>
Factory>

```

DURGASOFT

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

```
hibernate.cfg.xml   oracle.cfg.xml   student.hbm.xml  22
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-mapping-3.0.d
<hibernate-mapping>
  <class name="beans.Student" table="student007" schema
    <cache usage="read-write"/>
    <id name="sid"/>
    <property name="sname"/>
    <property name="semail"/>
    <property name="smarks"/>

  </class>

</hibernate-mapping>
```

DURGASOFT

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

System.out.println("using first session");
Student st1 = (Student) session.get(Student.class);
System.out.println(st1.getSname());
System.out.println(st1.getSemail());

Student st2 = (Student) session.get(Student.class);
System.out.println(st2.getSname());
System.out.println(st2.getSemail());

System.out.println("using second session");
Student st3 = (Student) session.get(Student.class);
```

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

But in session factory only one select operation

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

System.out.println("For first query...");

Query q=s.createQuery("select ename from Employee");
q.setCacheable(true);
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);
```

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

```

Client.java Employee.java employee.hbm.xml InsertClient.java oracle.cfg.xml SelectClient.java
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();

```

DURGASOFT  
www.durasoft.co.in

```

<property name="net.sf.ehcache.configurationResourceName" value="ehcache.xml"/>
<property name="cache.use_query_cache">true</property>
<mapping resource="resources/employee.hbm.xml"/>

</session-factory>

```

```

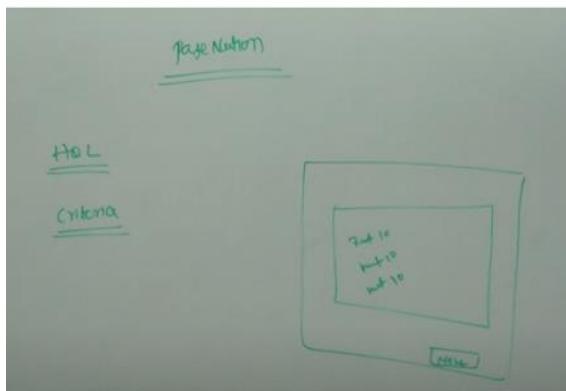
e1.setEid(111);
e1.setEname("abc");
e1.setEmail("abc@gmail.com");
e1.setSalary(500000);

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 retrieve limited value

We want to retrieve only 5 records

```
Session s = sf.openSession();
Query q = s.createQuery("from Student");
q.setFirstResult(0);
q.setMaxResults(5);

for(int k=0; k<q.list().size(); k++)
{
    q.setFirstResult(k);
    q.setMaxResults(1);

    List<Student> st=q.list();
    i=k+1;
    j=j+1;
}
```

```
throws ServletException, IOException {
PrintWriter out=resp.getWriter();
Session s = sf.openSession();
int fr = Integer.parseInt(req.getParameter("fr"));
int mr = Integer.parseInt(req.getParameter("mr"))

Query q = s.createQuery("from Student");

    q.setFirstResult(fr);
    q.setMaxResults(mr);

    List<Student> list=q.list();
    for(Student st:list)
    {
        out.println("ID="+st.getId()+"\t NAME"
    }
}
```

Criteria pagination

```
PrintWriter out = resp.getWriter();
Session s = sf.openSession();
int fr = Integer.parseInt(req.getParameter("fr"));
int mr = Integer.parseInt(req.getParameter("mr"))

Criteria cr = s.createCriteria(Student.class);
cr.setFirstResult(fr);
cr.setMaxResults(mr);

List<Student> list = cr.list();
for (Student st : list) {
    out.println("ID=" + st.getId() + "\t NAME=" +
        "\t email=" + st.getEmail() + "\t m
}

s.close();
```

@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 ehcache exceed from its configured size ehcache removed expired elements

If that doesn't clear enough space it clear object from ehcache

By default it remove LRU elements

Second level caching is we put on entity or whole object

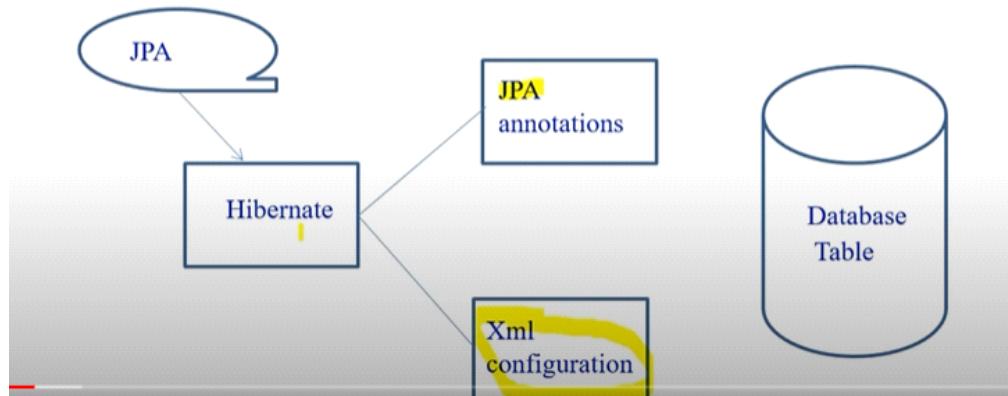
Default cache is made

```
@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.



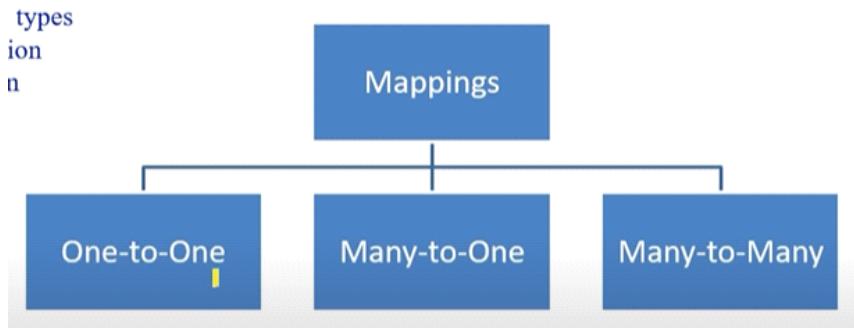
- **Why Hibernate ?**
- Hibernate **eliminates all the boiler-plate code** that comes with JDBC.
- It supports HQL with is more Object oriented.
- It provides transaction management implicitly.
- Hibernate throws **JDBCException** or **HibernateException** which are the unchecked exceptions, so we don't 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(persistentObj);
t.commit();
factory.close();
session.close();
```



# 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

##### 1) HTML (HyperText Markup Language)

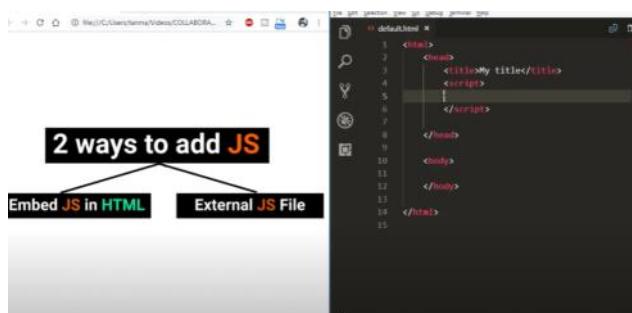
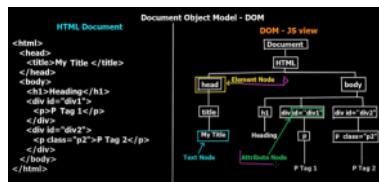
- >> Adds structure to our web pages.
- >> Tags used for e.g. <div>, <section>, <p>

##### 2) CSS (Cascading Style Sheets)

- >> Adds **styles** to our webpages.
- >> E.g. colors, border, margin, image, positions etc.
- >> Can use ids, classes or direct tags to reference HTML tags

##### 3) JS (JavaScript)

- >> Adds programming to our web pages.
- >> Adds functionality, e.g. client side validation, effects & events etc.



#### Embedded type

```

<html>
<head>
<title>My title</title>
</head>
<body>
<script>
<!-- Hello World -->
<!-- Hello World -->
</script>
</body>
</html>
  
```

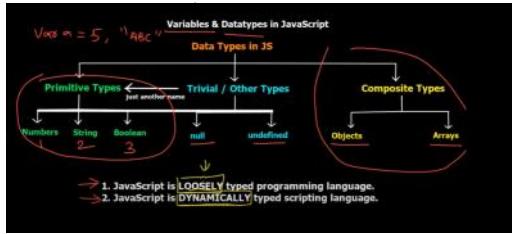
Document.write("Hello world")  
document.write("<h1>Hello World</h1>");  
It represent entire document module in dom  
Write is method which is directly writes on browser

#### Separate javascript file

```

<html>
<head>
<title>Muari Kumar</title>
<script src="demo.js" type="text/javascript">
</script>
</head>
<body>
</body>
</html>
  
```

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 I

```
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 more 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 method**

```
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>

```

```

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

        function buttonclick()
        {
            var the_document=document.getElementById("first");
            the_document.innerHTML="Murari Learning";
            the_document.innerHTML="Second";
        }

    </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

```

<head>
    <title>Murari</title>
    This page says
    Value inside the text box Murari
</head>
<body>
    <script type="text/javascript">
        var str=document.getElementById("text").value;
        alert("Value inside the text box"+str);
    </script>
</body>

```

Username & password

```

<head>
    <title>Murari Kumar</title>
    <script type="text/javascript">
        function fn1()
        {
            var user=document.getElementById("user").value;
            pass=document.getElementById("password").value;
            if(user=="Murali" && pass=="9102")
            {
                alert("Hl successful login "+ user);
            }
            else{
                alert("Wrong credential");
            }
        }
    </script>
</head>
<body>
    <input id="user" placeholder="Enter Username">
    <br>
    <input type="password" id="password" placeholder="Password">
</body>

```

Radio group name for both then we can select any one  
Otherwise we both will be selected if we give different name

```

<head>
    <title>Murari Kumar</title>
    <script type="text/javascript">
        //200 manipulations & getelementbyid method/
        function fn1()
        {
            var r1=document.getElementById("r1").value;
            var r2=document.getElementById("r2").value;
            if(r1.checked==true)
                alert("The channel selected is "+r1.value);
            else if(r2.checked==true)
                alert("The channel selected is "+r2.value);
            else
                alert("No channel selected");
        }
    </script>
</head>
<body>
    <input id="r1" name="grp1" type="radio" value="Simple Snippets">
    Simple Snippets</input>
    <br>
    <input id="r2" name="grp1" type="radio" value="Murari Learning">
    Murari Learning</input>
</body>

```

```

1   <html>
2
3   <head>
4       <title>Murari Kumar</title>
5       <script type="text/javascript">
6           function fn1()
7           {
8               var r1=document.getElementById("select");
9               alert("Pop up "+r1.value);
10          }
11
12      </script>
13  </head>
14
15  <body>
16      <select id="select">
17          <option value="Murali">Murari</option>
18          <option value="Kumar">Kumar</option>
19          <option value="Rahul">Rahul</option>
20      </select>
21      <button onclick="fn1()" id="btn1">On click</button>
22
23
24  </body>
25
26
27 </html>

```

Get element by tag name

```

<html>
<head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
        /*DOM manipulations & getelementById method*/
        function fn1()
        {
            var select = document.getElementById("selectbox");
            alert(select.options[select.selectedIndex].value);
        }
    </script>
</head>
<body>
    <select id="selectbox">
        <option value="Simple Snippets">Simple Snippets</option>
        <option value="Telusko Learnings">Telusko Learnings</option>
        <option value="MKBHD">MKBHD</option>
    </select>
    <br>
    <button onclick="fn1()">Click Me</button>
</body>
</html>

```

```

<html>
<head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
        /*DOM manipulations & getelementByTagName method*/
        function changeStyling()
        {
            var para = document.getElementsByTagName("");
        }
    </script>
</head>
<body>
    <p>This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p>This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p>This is paragraph 5</p>
    <br>
    <a href="#">This is anchor tag</a>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <div></div>
    <br>
    <button onclick="changeStyling()">Click Me</button>
</body>
</html>

```

```

<html>
<head>
    <title>DOM manipulations</title>
    <script type="text/javascript">
        /*DOM manipulations & getelementsByTagName method*/
        function changeStyling()
        {
            var para = document.getElementsByTagName("p");
            para[0].style.fontSize = "25px";
            para[1].style.color = "#ee0000";
            para[2].style.fontWeight = "bold";
            para[3].style.fontStyle = "italic";
        }
    </script>
</head>
<body>
    <p>This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p>This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p>This is paragraph 5</p>
    <br>
    <button onclick="changeStyling()">Click Me</button>
</body>
</html>

```

This is paragraph 1

This is paragraph 2

This is paragraph 3

This is paragraph 4

This is paragraph 5

Click Me

```

<html>
<head>
    <title>DOM manipulations & getelementById method</title>
    <script type="text/javascript">
        /*DOM manipulations & getelementById method*/
        function changeStyling()
        {
            var tag = document.getElementsByName("p");
            var element = document.getElementsById("mypara");
            for(var x=0;x<element.length;x++)
            {
                element[x].style.color="red";
            }
        }
    </script>
</head>
<body>
    <p id="p1" class="mypara">This is paragraph 1</p>
    <p>This is paragraph 2</p>
    <p class="mypara">This is paragraph 3</p>
    <p>This is paragraph 4</p>
    <p class="mypara">This is paragraph 5</p>
    <a class="mypara" href="#">This is Anchor tag</a>
    <br>
    <button onclick="changeStyling()">Click Me</button>
</body>
</html>

```

This is paragraph 1

This is paragraph 2

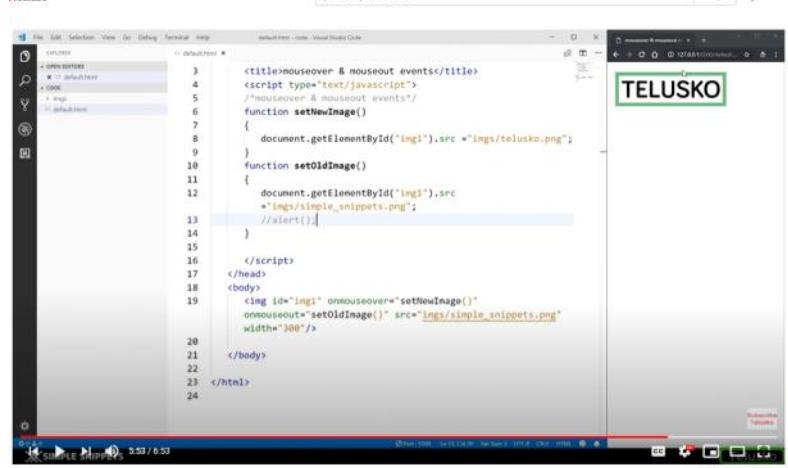
This is paragraph 3

This is paragraph 4

This is paragraph 5

This is Anchor tag

Click Me

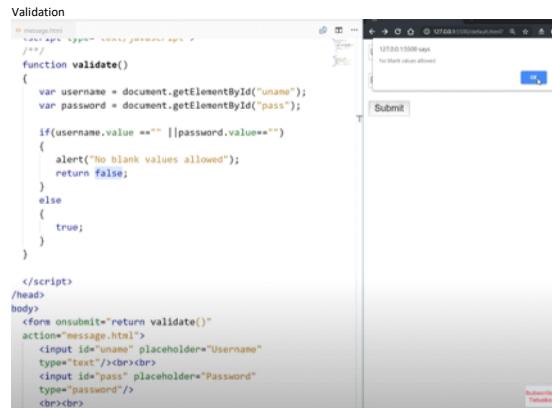


```

<html>
<head>
    <title>mouseover & mouseout events</title>
    <script type="text/javascript">
        /*mouseover & mouseout events*/
        function setNewImage()
        {
            document.getElementById("img1").src ="imgs/telusko.png";
        }
        function setOldImage()
        {
            document.getElementById("img1").src
            ="imgs/simple_snippets.png";
            //alert();
        }
    </script>
</head>
<body>
    
</body>
</html>

```

#25 Change Image onmouseover and onmouseout events in JavaScript



```

<html>
<head>
    <title>Validation</title>
    <script type="text/javascript">
        function validate()
        {
            var username = document.getElementById("uname");
            var password = document.getElementById("pass");

            if(username.value == "" || password.value == "")
            {
                alert("No blank values allowed");
                return false;
            }
            else
            {
                true;
            }
        }
    </script>
</head>
<body>
    <form onsubmit="return validate()" action="message.html">
        <input id="uname" placeholder="Username" type="text"><br><br>
        <input id="pass" placeholder="Password" type="password">
        <br><br>
    </form>
</body>

```

On submit extra feature

JavaScript Page 127

```

// Generated by jsbeautify.org - http://jsbeautify.org/
// (c) 2011 Douglas Crockford
// MIT License
// http://jsbeautify.org/LICENSE.txt

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;
    }
}

```

Username:   
Password:   
Submit

Value: Value

```

<html>
<head>
    <title>Regular Expressions in JS </title>
    <script type="text/javascript">
        function validate()
        {
            var username = document.getElementById("uname").value;
            var regex = /\w+/;
            if(regex.test(username))
            {
                alert("Valid Username");
            }
            else
            {
                alert("Invalid Username");
                document.getElementById("blouser").style.visibility="visible";
            }
        }
    </script>
</head>
<body>
    <form action="message.html">
        <input id="uname" placeholder="Username" type="text"/><br>
        <label id="blouser" style="color: red; visibility: hidden">Invalid Username</label>
        <br/>
    </form>
</body>

```

3035 Invalid Username Submit

REGULAR EXPRESSION TEST STRING

7656565656

# 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 concise. We have to keep multiple context in mind, what `e` is and what `i` is and what we are comparing the two. We lost the general benefits of lambdas—concise and expressive—this code fails both.

## 2. It's noisy

Writing a single line lambda was nice. The minute we wanted to do more than one line, we have to bring the `{}`, the `,`, and possibly the `return` if we had something to return. That's Java's (and C#'s too) way of telling us—it's telling us not to do that.

## 3. Leads to Duplication

Whatever that multiline code does, it's not reusable. If we want to

Method refrence-> 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 reference

```

// 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)));
}

```

```

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 -- FEMALE -- 20], [Sara -- FEMALE -- 22].

```

```

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
    //name

    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 completablefuture

```

| Account.java ✘
1 package threadingexample;
2
3 public class Account {
4     private int balance=20000;
5     public int getbalance()
6     {
7         return balance;
8     }
9     public void withdraw(int val)
10    {
11        this.balance=this.balance-val;
12    }
13 }
14

| Client.java ✘
1 package threadingexample;
2
3 public class Client {
4     public static void main(String[] args) {
5         Account account= new Account();
6         AccountHolder holder= new AccountHolder(account);
7         Thread t1= new Thread(holder);
8         Thread t2= new Thread(holder);
9         t1.setName("Murari");
10        t2.setName("kumar");
11        t1.start();
12        t2.start();
13    }
14
15
16 }

| AccountHolder.java ✘
10@ Override
11 public void run() {
12     for (int i = 1; i < 5; i++) {
13         makewithdrawable(2000);
14         if (account.getbalance() < 0) {
15             System.out.println("Already taken....." + Thread..);
16         }
17     }
18 }
19
20
21 private synchronized void makewithdrawable(int i) { // protect mu
22     if (account.getbalance() >= i) {
23         System.out.println(
24             Thread.currentThread().getName() + " " + "Current
25         try {

```

If not synchronized all thread access at same time. If synchronized it will lock thread  
<https://www.mockaroo.com/> To mock data

## Microservice

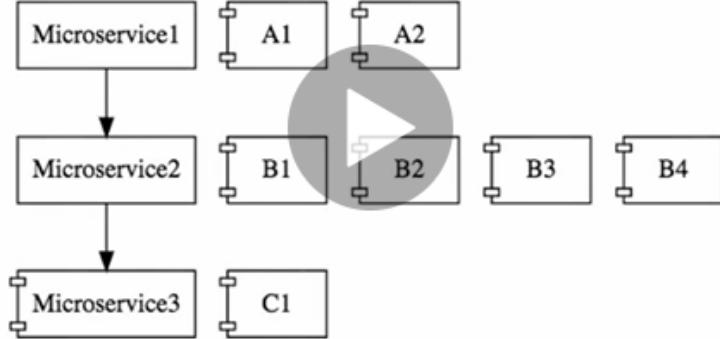
### 1.Bounded Context

Earlier we were building one monolithic app now 10-20 microservices

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

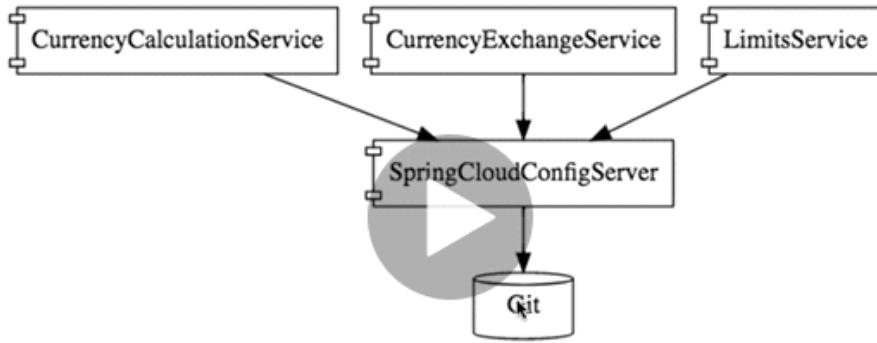
### 4. Visibility

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

### 5 Pack of cards

If one microservice fail then all will fail

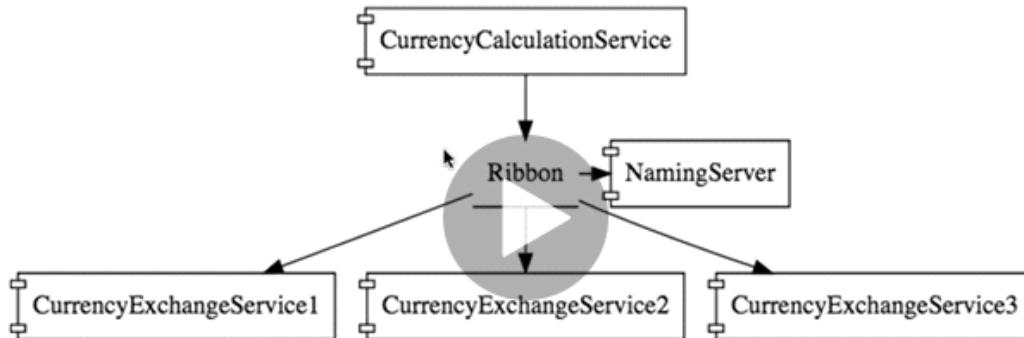
## Spring cloud



## *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*

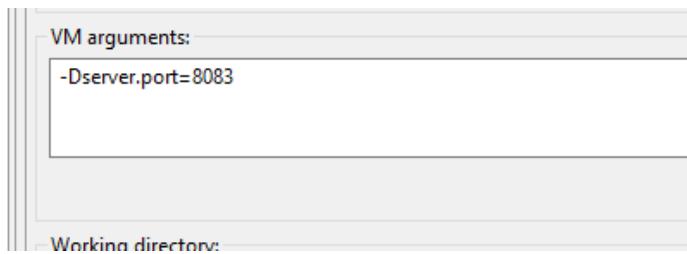
Setting property from application properties

```

@.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

```

@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>

---

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

---

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

---

```
@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

```

@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

```
@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);
    }
}
```

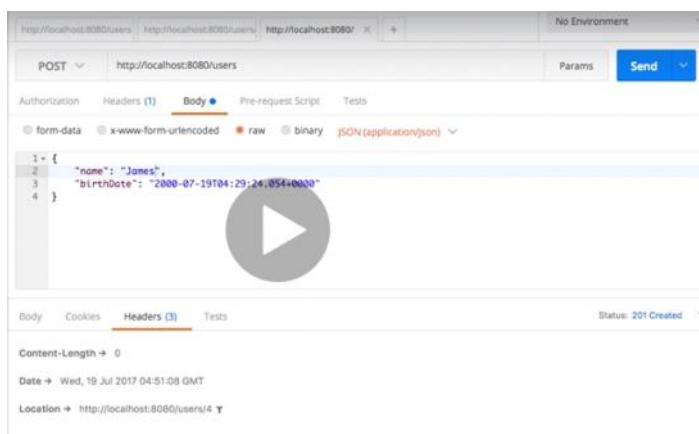
Reprsentational state transfer  
Maximum use of rest

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
    }
}

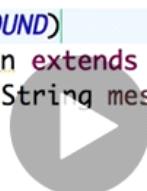
@GetMapping("/users/{id}")
public User retrieveUser(@PathVariable int id) {
    User user = service.findOne(id);
    if(user==null)
        throw new UserNotFoundException("id-" + id);

    return user;
}
```

```

    @ResponseStatus(HttpStatus.NOT_FOUND)
    public class UserNotFoundException extends RuntimeException {
        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")
    @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);

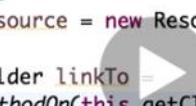
//"/all-users", SERVER_PATH + "/users"
//retrieveAllUsers
Resource<User> resource = new Resource<User>(user);

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

//HATEOAS

return user;

```



RestfulWebServicesApplication.java  
2017-07-19 14:  
2017-07-19 14:  
2017-07-19 14:  
2017-07-19 14:



```

//"/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

```

@Autowire
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;
}

```

Key	Value	Description
Content-Type	application/json	
Accept-Language	us	

```

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

    localeResolver.setDefaultLocale(Locale.US);

    return localeResolver;
}

```

Swagger->

```

@Configuration
@EnableSwagger2
public class Swagger {

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

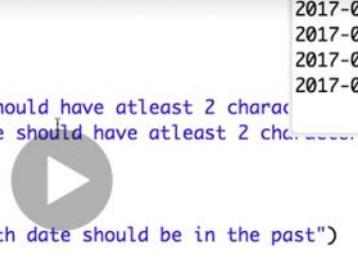
{
    swagger: "2.0",
    info: {...},
    host: "localhost:8080",
    basePath: "/",
    tags: [...],
    paths: {
        /error: {...},
        /hello-world: {...},
        /hello-world-bean: {...},
        /hello-world-internationalized: {...},
        /hello-world/path-variable/{name}: {...},
 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

```

```

@ApiModelProperty(description="All details about the user. ")
public class User {
    private Integer id;
    @Size(min=2, message="Name should have atleast 2 characters")
    @ApiModelProperty(notes="Name should have atleast 2 characters")
    private String name;
    @Past
    @ApiModelProperty(notes="Birth date should be in the past")
    private Date birthDate;
    protected User() {
    }
}

```



The screenshot shows the Swagger UI interface for a 'User' resource. It lists several endpoints: /error, /hello-world, /hello-world-bean, /hello-world-internationalized, /hello-world/path-variable/{name}, /users, and /users/{id}. Each endpoint has a detailed description and parameters listed below it.

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

```

public class SomeBean {

    private String field1;

    private String field2;

    @JsonIgnore
    private String field3;

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

```

Dynamic filter

```

@JsonFilter("SomeBeanFilter")
public class SomeBean {

    private String field1;

    private String field2;

    private String field3;

    public SomeBean(String field1

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

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

        FilterProvider filters = new SimpleFilterProvider().addFilter("SomeBeanFil
        MappingJacksonValue mapping = new MappingJacksonValue(someBean);

        mapping.setFilters(filters);

        return mapping;
    }
}

```

```

@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

```

@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"));
}

@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"));
}

```

GET ▼ http://localhost:8083/person/header

Headers (7)

Key	Value
User-Agent	PostmanRuntime/7.26.8
Accept	*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
X-API-VERSION	2
Body	
Cookies	
Headers (5)	
Test Results	

Pretty Raw Preview Visualize JSON

```

1 [
2   "name": {
3     "firstname": "Murari",
4     "lastname": "Kumar"
5   }

```

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"));
}

```

GET ▼ http://localhost:8080/person/produces

Headers (2)

Key	Value	Description
X-API-VERSION	2	..
Accept	application/vnd.company.app-v1+json	

Body

Pretty Raw Preview JSON

```

1 {
2   "name": "Bob Charlie"
3 }

```

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
    UserRepository user;

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

    @GetMapping("/findById/{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 UserRepository extends JpaRepository<User, Integer>{
}

```

Many to one relationship

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

```
@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

```
@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
    }

    @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();
    }
}

@Entity
public class Post {
    @Id
    @GeneratedValue
    int id;
    String description;
    @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

1 GET /users/{PathVariable}?QueryKey=QueryValue HTTP/1.1  
2 Host: localhost:8080  
3 header-key: headerValue

## HTTP Request To Spring Annotations

- Path Variable = `@PathVariable`
- Query (Request) Parameter = `@RequestParam`
- Header Parameter = `@RequestHeader`

```

12  * @RestController
13  * @RequestMapping("/requestParam")
14  public class RequestParamController {
15
16     @GetMapping("/required")
17     public void requiredRequestParam(@RequestParam String name){
18         System.out.println(name);
19     }
20
21     //optional
22
23     @GetMapping("/notRequired")
24     public void notRequiredReqeustParam(@RequestParam String name){
25         System.out.println(name);
26     }
27
28     @GetMapping("/notRequiredDef")
29     public void notRequiredDefault(@RequestParam String name){
30         System.out.println(name);
31     }
32
33     @GetMapping("/notRequiredOpt")
34     public void notRequiredOptional(@RequestParam String name){
35         if(name.isPresent()){
36             System.out.println(name.get());
37         }
38     }
39     //allParams
40     @GetMapping("/allParams")
41     public void notRequiredOptional(@RequestParam Map<String, String> map){
42         System.out.println(map);
43     }

```

Query parameter 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));
}

```

Postman screenshot showing a GET request to `http://localhost:8080/requestParam/allParams?name=QueryValue&param2=darklord&param3=3232ads`. The 'Params' tab displays the following query parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	QueryValue	
<input checked="" type="checkbox"/> param2	darklord	
<input checked="" type="checkbox"/> param3	3232ads	

Postman screenshot showing a GET request to `http://localhost:8080/requestParam/listParams?id=1,2,3`. The 'Params' tab displays the following query parameters:

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> id	1,2,3	
param2	darklord	
param3	3232ads	

### 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 repository.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>

```

```

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

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

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

@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 ehache exceed from its configured size ehcache removed expired elements  
If that doesn't clear enough space it clear object from ehcache  
By default it remove LRU elements

Second level caching is we put on entity or whole object

```

@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");

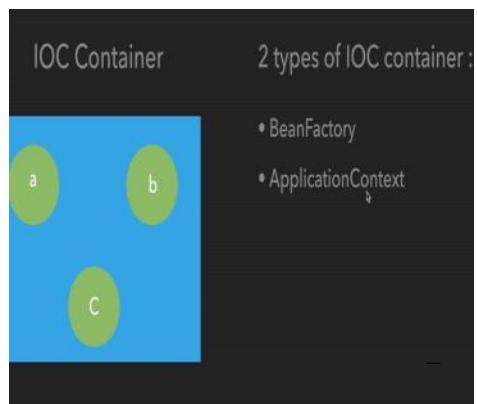
    EmployeeVO result = restTemplate.getForObject(uri, EmployeeVO.class, params);

    //Use the result
}

```

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

## ControllerAdvice



A screenshot of an IDE showing Java code and XML configuration. The Java code in 'Mobile.java' includes imports for 'Sim.java', 'Airtel.java', 'Vodaphone.java', and 'Mobile.java', along with a main method that prints 'config loaded', gets a bean named 'vodafone', and calls its 'calling()' and 'data()' methods. The XML configuration in 'beans.xml' defines beans for 'vodafone', 'airtel', and 'mobile'.

```
<?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="vodafone" class="com.seleniumexpress.ioc.Vodaphone"/>
<bean id="airtel" class="com.seleniumexpress.ioc.Airtel"/>
<bean id="mobile" class="com.seleniumexpress.ioc.Mobile"/>

```

```
public class Mobile {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("config loaded");
        Vodaphone voda = context.getBean("vodafone", Vodaphone.class);
        voda.calling();
        voda.data();
    }
}
```

A screenshot of an IDE showing Java code and XML configuration. The Java code in 'Mobile.java' includes imports for 'Sim.java', 'Airtel.java', 'Vodaphone.java', and 'Mobile.java', along with a main method that prints 'config loaded', gets a bean named 'airtel', and calls its 'calling()' and 'data()' methods. The XML configuration in 'beans.xml' defines beans for 'vodafone', 'airtel', and 'mobile'.

```
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

```

6 public class Mobile {
7
8     public static void main(String[] args) {
9
10        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
11        System.out.println("config loaded");
12
13        Sim sim = context.getBean("sim", Sim.class);
14        sim.calling();
15        sim.data();
16    }
17
18 }

```

Without touching vodafone and airtel

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4   xsi:schemaLocation="http://www.springframework.org/schema/beans
5   http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7 <bean id="sim" class="com.seleniumexpress.ioc.Jio"></bean>
8
9
10</beans>
11
12
13

```

Control is taken by framework now they are creating object



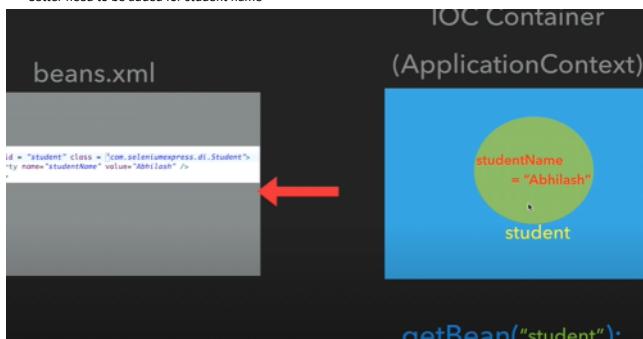
USING SETTER INJECTION  
USING CONSTRUCTOR INJECTION



```
public class Exam {
    public static void main(String[] args) {
        Student student = new Student();
        student.setStudentName("Abhilash Panigrahi");
        student.displayStudentInfo();
    }
}
```

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

Setter need to be added for student name



```
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 Dependency

```
public class Student {
    private int id;
    private String studentName;

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

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

```
9<bean id = "student" class="com.seleniumexpress.di.Student">
10   <constructor-arg name="studentName" value="Abhilash Panigrahi" />
11   <constructor-arg name="id" value="1" />
12 </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 cheating started..");
    }
}

Not a good programming.

public class Student {
    MathCheat mathCheat;

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

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

<beans>
<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>

```

**Internally**

```

Student stu = new Student();
MathCheat mathCheat = new MathCheat();
stu.setId(1001);
stu.setMathCheat(mathCheat);

```

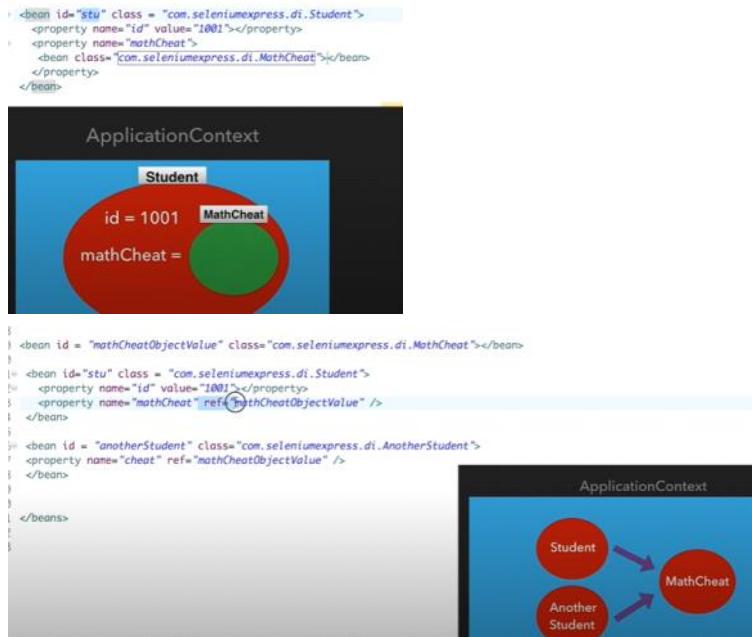
```

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

public class Client {

    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
        System.out.println("beans.xml file loaded");
        context.getBean("stu", Student);
    }
}

```



Loose coupling

```

1 package com.seleniumexpress.di;
2
3 public class Student {
4
5     private Cheat cheat;
6
7     public void setCheat(Cheat cheat) {
8         this.cheat = cheat;
9     }
10
11    public void cheating() {
12
13        cheat.cheat();
14    }
15
16 }
17

```

```

3 public interface Cheat {
4
5     public void cheat();
6
7 }
8

```

```

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.....

```

Human human = new Human();
Heart heartObject = new Heart();
human.setHeart(heartObject);

<bean id = "heartObject" class="com.seleniumexpress.autowired.Heart">
<property name="heart" ref = "heartObject"></property>
</bean>

<bean id = "human" class="com.seleniumexpress.autowired.Human">
<property name="heart" ref = "heartObject" />
</bean>

```

I am Removing the property that we manually set.

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">
```

Dear Spring !!  
While creating **Human** class object any dependency present in Human class meeting **autowire = "byName"** criteria, Inject those beans to their respective dependency.

```

3 public class Human {
4
5     private Heart heart;
6

```

Heart class reference variable('heart') name and bean id name ('heart') matched !!

Autowiring Successful

By type autowired niche wala

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

Dear Spring !!  
While creating **Human** class object any dependency present in Human class meeting **autowire = "byType"** criteria(in beans.xml file), Inject those beans to their respective dependency.

```

3 public class Human {
4
5     private Heart heart;
6

```

The type of the Variable and the type of the Bean matched

Autowiring Successful

Autowire constructor

```

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 called");
    }
}

```

works same as autowire = "byConstructor"

```

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

```

/beans>



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

```

3 import org.springframework.beans.factory.annotation.Autowired;
4
5 public class Human {
6     private Heart heart;
7
8     public Human() {
9
10 }
11
12     @Autowired
13     public Human(Heart heart) {
14         this.heart = heart;
15     }
16
17     public void setHeart(Heart heart) {
18         this.heart = heart;
19         System.out.println("setter method called");
20     }

```

[Sep 15, 2018 1:58:21 PM org.springframework.context.support.ClassPathXmlApplicationContext publishEvent  
Sep 15, 2018 1:58:22 PM org.springframework.context.support.ClassPathXmlApplicationContext publishEvent  
INFO: Loading XML bean definitions from class path resource [beans.xml]  
you are dead



```

http://www.springframework.org/schema/beans/spring-beans.xsd"
<context:annotation-config>
<!--
<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:annotation-config/>  
in order to use annotations in spring. (eg: to use @Autowired )

```

1 package com.seleniumexpress.autowired;
2
3 public class Human {
4     private Heart heart;
5
6     public Human() {
7
8     }
9
10    public Human(Heart heart) {
11        this.heart = heart;
12        System.out.println("human constr is getting called which has Heart as arg");
13    }
14
15    @Autowired
16    public void setHeart(Heart heart) {
17        this.heart = heart;
18    }
19
20}

```

**How @Autowired works?**

1. first it tries to resolve with "byType".
2. If byType fails then it goes with "byName"



```

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

```

Unique bean

```

5   xmlns:context="http://www.springframework.org/schema/context"
6   xsi:schemaLocation="http://www.springframework.org/schema/beans
7       http://www.springframework.org/schema/beans/spring-beans.xsd
8       http://www.springframework.org/schema/context
9       http://www.springframework.org/schema/context/spring-context.xsd"
10
11 <context:annotation-config />
12
13 <bean id = "humanHeart" class="com.seleniumexpress.autowired.Heart" >
14 <property name="nameOfAnimal" value="Human" />
15 <property name="noOfHeart" value="1" />
16 </bean>
17
18 <bean id = "octpusHeart" class="com.seleniumexpress.autowired.Heart" >
19 <property name="nameOfAnimal" value="Octpus" />
20 <property name="noOfHeart" value="3" />
21 </bean>
22
23 <bean id = "human" class="com.seleniumexpress.autowired.Human" >
24

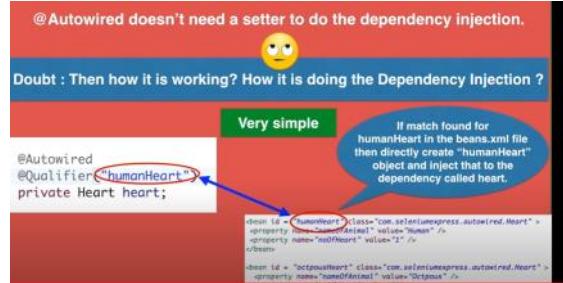
```

```

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

```

So no need to write setter if you are using @Autowired before the dependency



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

```

beans.xml  Student.java  Client.java  *student-info.properties*
1 student.name = Abhilash
2 student.interestedCourse = Java
3 hobby = cricket

```

```

<bean id = "student" class="com.seleniumexpress.loadingFrompropertiesfile.Student">
<property name="name" value = "${student.name}"/>
<property name="interestedCourse" value = "${student.interestedCourse}"/>
<property name="hobby" value = "${student.hobby}"/>
</bean>

```

<context:property-placeholder/>  
To know spring I am using from property file

@Value annotation remove property

```

12
13<bean id = "student" class="com.seleniumexpress.loadingfrompropertiesfile.Student">
14
15</bean>
16
17 .
18
19

```

## use some Annotations

using @Value annotation

```

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()
}

```

**@Required**

```

public void setIntrestedCourse(String intrestedCourse) {
    this.intrestedCourse = intrestedCourse;
}

//@Value("travelling")
public void setHobby(String hobby) {
    this.hobby = hobby;
}

```

**Introducing @Required**  
want to make "intrestedCourse" as required.

Restricted required to mandatory

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

Beans file is configuration file

```

<bean id = "collegeBean" class="com.seleniumexpress.college.College">
</bean>

```

Attribute : id  
The object identifier for a bean. A bean id may not be used more than once within the same <beans> element.  
Data Type : string

**Internally : College collegeBean = new College();**

@Componet does same work as above

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

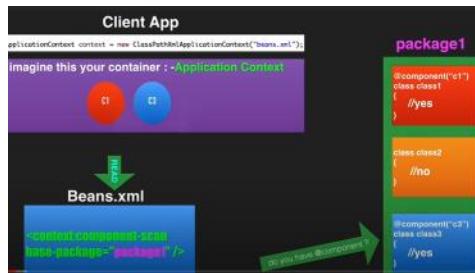
@Componet does same thing 1 create bean  
2 and register to IOC container

```

@Component("collegeBean")
public class College {
}

8
9 @Component("collegebean")
10 public class College {
11
12     @Autowired // for class type
13     Student student;
14
15     @Value("Murari") // for property like string
16     String name;
17
18     void dis() {
19         student.displa(name);
20     }
21 }
22
23

```



**@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")
public class CollegeConfig {
    @Bean
    public College collegeBean()
    {
        return new College();
    }
}
```

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

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

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

```
    public 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 - becomes a bean
    {
        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();
        teacher.teach();
    }
}

@Configuration
@ComponentScan(basePackages = "com.seleniumexpress.college")
@PropertySource("classpath:college-info.properties")
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");
    }
}

```

```

7 @Component
8 public class Body {
9     ...
10    @Value("Winter")
11    String value;
12
13    @Autowired
14    @Qualifier("innerBehaviour")
15    Behaviour behaviour;
16
17    public Body() {
18        // TODO Auto-generated constructor
19    }
20
21    public Body(Heart heart, Lungs lung, St
22    super();
23    this.heart = heart;
24    this.lung = lung;
25}

```

```

4 import org.springframework.stereotype.Component;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Component;
7
8 @Component
9 public class InnerBehaviour implements Behaviour{
10
11    public void display() {
12        System.out.println("Inner behaviour");
13    }
14}

```

```

4 import org.springframework.stereotype.Component;
5 import org.springframework.stereotype.Primary;
6 import org.springframework.stereotype.Component;
7
8 @Primary
9 public class OuterBehaviour implements Behaviour {
10
11    public void display() {
12        System.out.println("Outer behaviour");
13    }
14}

```

```

4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Component;
7
8 @Component
9 public class Body {
10
11    Heart heart;
12    Lungs lung;
13
14    @Value("Winter")
15    String value;
16
17    @Autowired
18    Behaviour behaviour;
19
20    public Body() {
21        // TODO Auto-generated constructor
22    }
23
24    public Body(Heart heart, Lungs lung, St
25    super();
26    this.heart = heart;
27    this.lung = lung;
28    this.value = value;
29}

```

```

1 package com.sim;
2
3 import org.springframework.context.annotation.Primary;
4 import org.springframework.stereotype.Component;
5
6 @Component
7 public class InnerBehaviour implements Behaviour{
8
9    public void display() {
10        System.out.println("Inner behaviour");
11    }
12}

```

```

1 package com.sim;
2
3 import org.springframework.context.annotation.Primary;
4 import org.springframework.stereotype.Component;
5
6 @Primary
7 public class OuterBehaviour implements Behaviour {
8
9    public void display() {
10        System.out.println("Outer behaviour");
11    }
12}

```

```

7 @Configuration
8 public class BodyConfig {
9
10    @Bean
11    Body body() {
12        Body body = new Body();
13        body.setHeart(heartbuffalo());
14        body.setLung(lung());
15        body.setOuterBehaviour(behaveiour());
16        return body;
17    }
18
19    @Bean
20    Heart heart() {
21        Heart heart = new Heart();
22        heart.setName("Human");
23        return heart;
24    }
25
26    @Bean
27    Heart heartbuffalo() {
28        Heart heart = new Heart();
29        heart.setName("buffalo");
30        return heart;
31    }
32
33    @Bean
34    Lungs lung() {
35        return new Lungs();
36    }

```

```

7
8 public class Body {
9
10    Heart heart;
11
12    Lungs lung;
13
14
15    String value;
16
17
18    Behaviour behaviour;
19
20    public Body() {
21        // TODO Auto-generated constructor stub
22    }
23
24    public Body(Heart heart, Lungs lung, String value) {
25        super();
26        this.heart = heart;
27        this.lung = lung;
28        this.value = value;
29    }
30}

```

```

@Component
public class College {

    //@Value("${college.Name}")
    private String collegeName;
    I @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 cycle of bean

```

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("Retrieving 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 or 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

```

@PostConstruct
public void createStudentDBConnection() throws ClassNotFoundException, SQLException {
    // load driver
    Class.forName(driver);

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

```

**Hey spring , Once you crate  
StudentDAO Object Please call  
`createStudentDBConnection()` by  
yourself. Don't wait for me to call it.**

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



You can add custom code/logic during bean initialization

It can be used for setting up resources like db/socket/file etc

```

65:     public void createStudentDBConnection() throws ClassNotFoundException, SQ
66:     System.out.println("creating connection..");
67:     // load driver
68:     Class.forName(driver);
69:
70:     // get a connection
71:     con = DriverManager.getConnection(url, userName, password);
72:
73: }

```

why init()??

destroy method will be called before the bean is removed from the container.

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

```

@PreDestroy
public void closeConnection() throws SQLException
{
    //clean up job
    //closing the connection
    con.close();
}

```

Before spring remove studentDAO bean(object) from the container; It will call this method

**Standalone app**

```

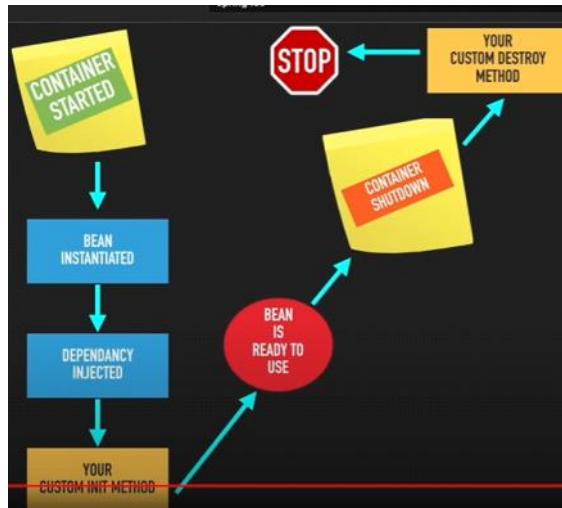
// creating container object manually
ApplicationContext context = new ClasspathXmlApplicationContext();

// destroying container object manually
context.close();

```

**Web App**

you don't need to create and destroy the container object. This will be automatically done. We will explore more while studying spring MVC.



```

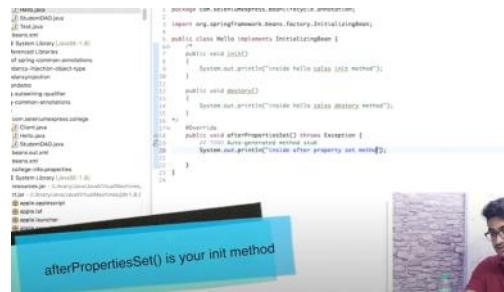
ClassPathXmlApplicationContext context = new Cl
context.registerShutdownHook();
StudentDAO studentDao = context.getBean("studen
System.out.println(studentDao);
studentDao.selectAllRows();

```

registerShutdownHook() will execute once main method ends

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.

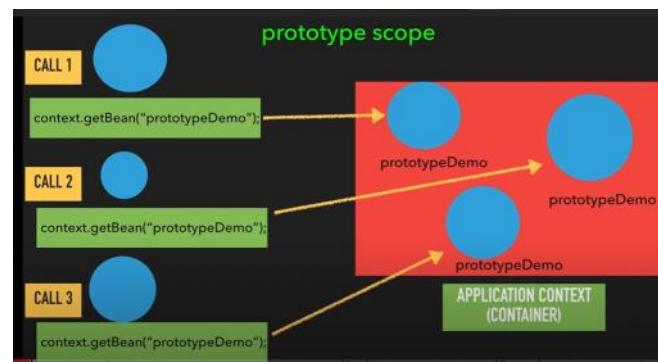
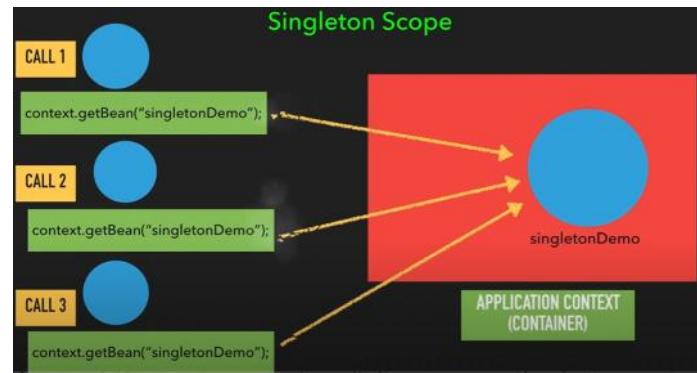
```
1 Java
2 package com.selenide;
3
4 import org.springframework.context.support.ClassPathXmlApplicationContext;
5
6 public class App {
7
8     public static void main(String[] args) {
9
10         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
11         SingletonDemo obj1 = context.getBean("singletonDemo", SingletonDemo.class);
12         SingletonDemo obj2 = context.getBean("singletonDemo", SingletonDemo.class);
13         System.out.println(obj1 + " " + obj2);
14     }
15
16 }
17
```

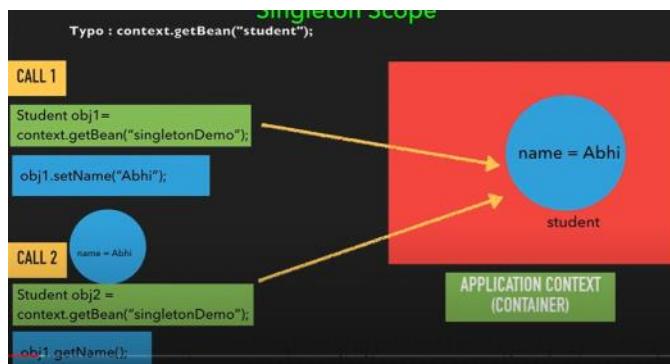
obj1 and obj2 object reference are the same or different?

Guess the output

```
14     if(obj1 == obj2) {
15         System.out.println("same object returned..");
16     }
17 }
18
19 }
20
```

terminated: App [Java Application] (LibraryJava/JavatutMachines/ide-9.0.4/jdk/Contents/Home/bin/java (21-Sep-2020, 2:28:10 PM)
com.seleniumexpress.demo.SingletonDemo@6f784a1a
same object returned...





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

    String name;

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

    public String getName() {
        return name;
    }

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

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

When prototype 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("Wooo school");
    }
}
```

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

```
Component
@Scope(value="prototype", proxyMode = ScopedProxyMode.TARGET_CLASS)
public class Student {
```

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

```

@component
@Scope("singleton")
public abstract class School {
    @Autowired
    private Student student; // prototype

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

    @Lookup
    abstract Student createStudentObject();①
}

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

public void setStudent(Student student) {
    this.student = student;
}

@component
@Scope("prototype")
public class Student {
    private String name;

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

    public String getName() {
        return name;
    }

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

```

```

@component
@Scope("singleton")
public abstract class School {
    @Autowired
    private Student student; // prototype

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

    @Lookup
    abstract Student createStudentObject();①
}

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

public void setStudent(Student student) {
    this.student = student;
}

RUN THIS PROGRAM AND
DO OBSERVE THE OUTPUT.

COMMENT YOUR OUTPUT. ARE YOU SEEING A NORMAL
SCHOOL OBJECT OR PROXY SCHOOL OBJECT?

```

```

1 package com.seleniumexpress.demo;
2 import org.springframework.context.ApplicationContext;
3
4 public class App {
5
6     public static void main(String[] args) {
7
8         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
9         School schoolObj1 = context.getBean("school", School.class);
10        School schoolObj2 = context.getBean("school", School.class);
11        System.out.println(schoolObj1 + " " + schoolObj2);
12
13    }
14
15 }

```

Output in Console:

```

Info: [App] [Java Application] (Library/Java/JavaVirtualMachines/jdk-0.4.jdk/Contents/Home/bin/java) [21-Sep-2020, 9:22:51 PM]
0: obj created.
com.seleniumexpress.demo.ScopeTest@60bf060
0: obj created.
com.seleniumexpress.demo.ScopeTest@60bf060

```

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

For request scope

```

@RequestMapping("/testing1")
public void test(HttpServletRequest 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