**List of Technologies**

* HTML
* CSS
* Javascript
* JQuery
* Maven & Spring Framework
* Spring Boot
* Kafka

HTML: Hyper Text Markup Language

CSS: Cascading Style Sheet

Web development, there are 3 technologies are used

1. HTML: It is to display the content
2. CSS: It is to style the HTML
3. Javascript: It is to add effects dynamically and add HTML / CSS at runtime

Editors

1. VS Code
2. Eclipse

HTML

Heading tags: h1, h2, h3, h4, h5, h6

<h1>some content</h1>  
<h2>some content</h2>

Paragraphs: These contents are used to display the paragraphs, which gives a line break before & after the paragraph

bold & italic tags: <b> & <i>, these tags doesn’t have line breaks, these are used inline

<p>Some content <b> few more contents </b> again some more </p>

Self closing tags: These tags don’t have closing, they can be closed with the start tag itself

<hr />: Horizontal rule  
<br />: This is to give a line break  
<img />: This to display the image

Attributes: These are some extra information’s given to the tag, they appear in the start tag

ex: <img src=”url” width=”200” height=”200” />

Marquee: It makes content to move

Entities: These are some special characters which can be displayed using some entity names or numbers

&copy; this creates a copyright symbol  
&reg; this is to create registered mark  
t; for < symbol  
&gt; for > symbol

&#8377; creates rupee symbol

Links & Lists

anchor tag is used to create the link <a href=”url”>LinkName</a>  
<a href=”url” target=”\_blank”>LinkName</a>

Lists: There are 2 types

1. ordered <ol>
2. unordered <ul>

<li> it is the list index or list item that comes inside the ordered/unordered lists

Tables: <table> tag is used to create tables, it will have 2 child tags <thead> & <tbody> to create table headings & table contents respectively, both uses <tr> to create rows, to create columns we have <th> & <td>, <th> is used for heading purpose, <td> is used in the table body.

div: It is a container tag that can wrap other HTML tags, this is mainly to group the HTML tags & style them later

Forms: These allow to create forms that user can input, like text, checkbox, radio buttons, drop downs, buttons, password, text area, range bar, progress bar, file upload

Form comes with 2 attributes

1. action: Carries the server URL where the data needs to be submitted
2. method: This specifies the HTTP methods like GET / POST, which tells whether form data must be submitted through URL (GET) or through the body (POST)

Note: Default form submission is GET i.e., sends data through URL, hence its advisable to specify method=”post”

Usage:

<form action=”#” method=”post”>  
 <label>Enter Name</label>  
 <input type = “text” name = “username” />  
 <label>Enter Password</label>  
 <input type=”password” name=”secret” />  
 <!-- for drop down -->  
 <select name=”qualification”>   
 <option value=”btech”>BTech</option>  
 <option value=”mtech”>MTech</option>  
 </select>  
 </form>

CSS: Cascading Style Sheet, you can apply CSS in 3 ways

1. Inline CSS: applied on a particular element using style attribute
2. Internal CSS: applied to the whole HTML document, using <style> tag
3. External CSS: applied to multiple HTML documents by creating a CSS file, CSS file needs to be linked using <link> tag

Syntax: property : value; property : value;

CSS Selectors: These help in selecting the HTML elements, there are mainly 3 types of selectors

1. tag selector: to select the element using tag name

p { color : blue }  
h2 { color : red }

1. id selector: to uniquely select an element using the element id

#a { color : blue }  
#b { color : red }  
<p id = “a”> & <h2 id = “b” >

1. class selector: to select multiple elements using the class name

.i1 { color : blue }   
.i2 { color : red }  
<p class = “i1”> & <h2 class = “i1”>

3rd party CSS: These are predefined CSS provided by CSS communities that can be used anywhere in the world

ex: bootstrap, semantic-ui, angular-materials, primefaces, tailwind

Bootstrap common class-names

Button: btn btn-primary, btn-secondary, btn-success, btn-danger,

Forms: form-control, form-control-lg

div: container-fluid, container

Tables: table, table-striped

Activity: Align the Form to the center of the page

Javascript: It adds dynamic effects to your web page by accessing HTML & CSS at runtime, apart from modifying HTML & CSS it can also perform following operations

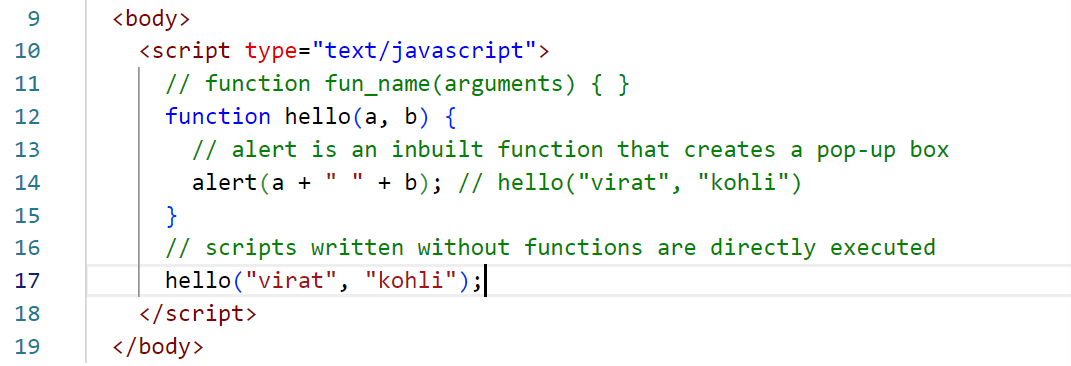
1. Validations
2. Event Handling

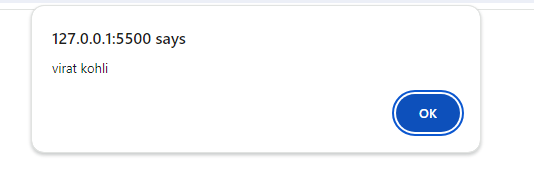
<script> tag is used to add javascript to your html

Fundamentals of Javascript

1. Variables
2. Functions
3. Operators
4. Conditional Statements
5. Arrays
6. Loops
7. Objects
8. Events

Our first JS program



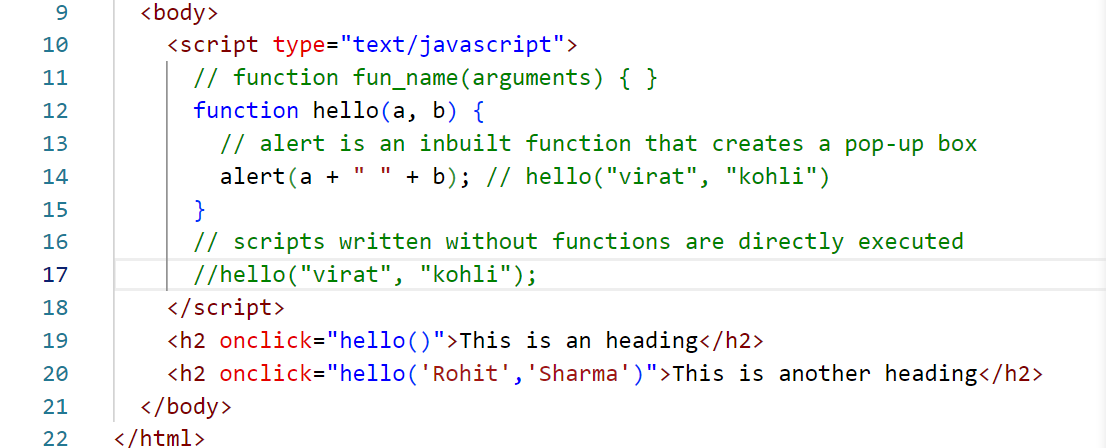
Output  


How to generate events & handle these events

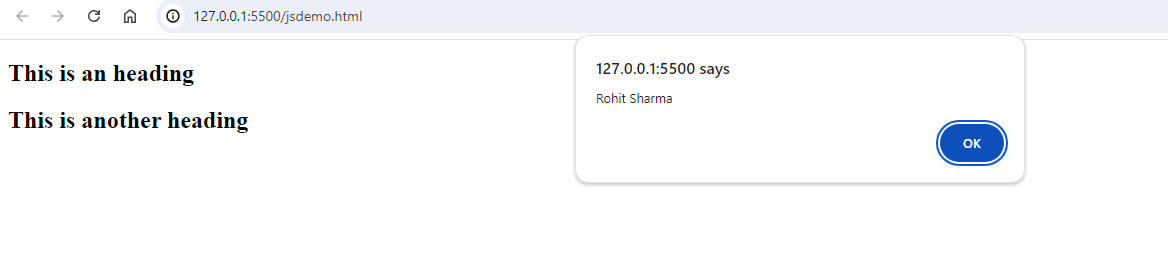
Events are anything that happens on browser, it could be click, mouse over, mouse out, form submit, input focus, input blur, all these events can occur on any HTML element

<tagName onclick=”fn()”>Some content</tag>

Program that generates click event on an HTML element



Output:



Arrays: Collection of elements, in Javascript arrays can have elements with different datatypes

var items = [ 20, 10, 30, 50, 40];

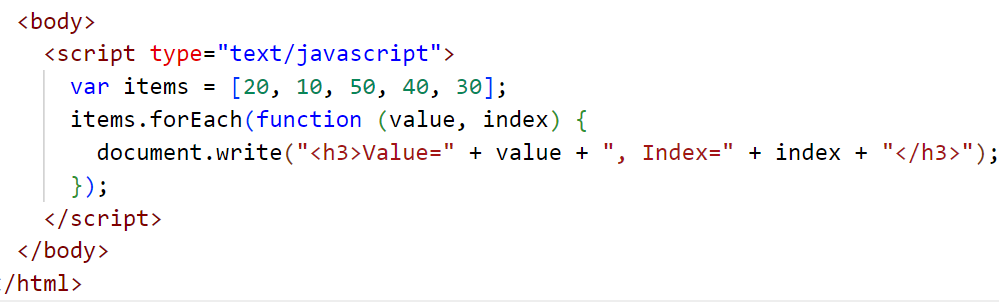
for(var index=0; index < items.length; index++) {   
 document.write(items[index]); // print the content in the browser  
}

Another way to iterate is using forEach() function of an array

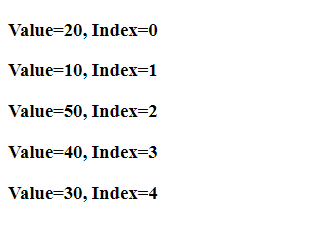
items.forEach( callbackFn );

items.forEach( function(value, index) {  
 document.write(value+ “ “ +index);  
 } );

Arrays iteration with forEach



Output:



Keywords used to declare variables in javascript

1. var (avoid using var): Because it doesn’t have any scope
2. let: It has scope & its value can be modified
3. const: It has scope & its value can’t be modified

Limitations of var

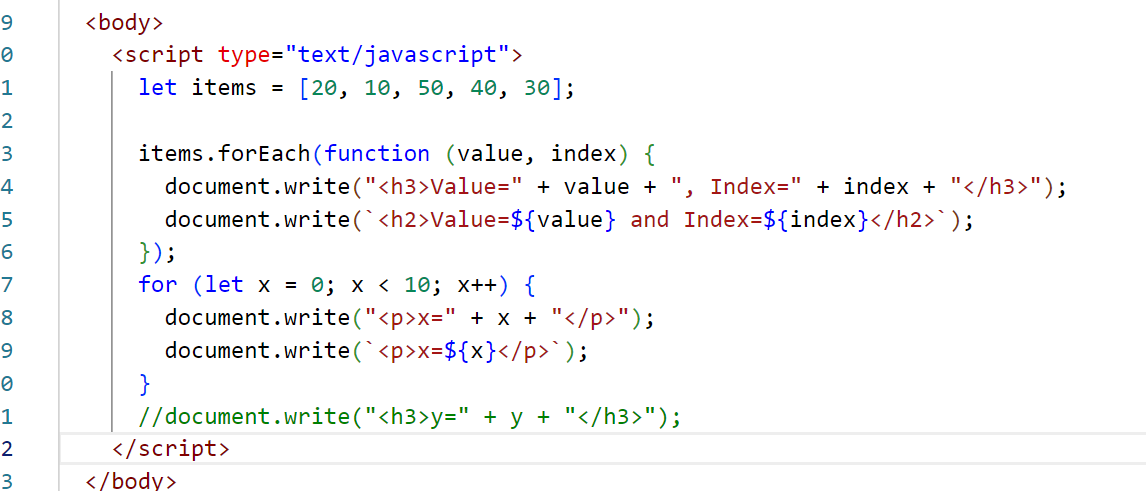
1. It doesn’t have a scope
2. It can be re-declared which is not correct

Javascript is derived from EcmaScript (ES), in ES6 they have made lot of changes that improved the Javascript syntax, those changes are:-

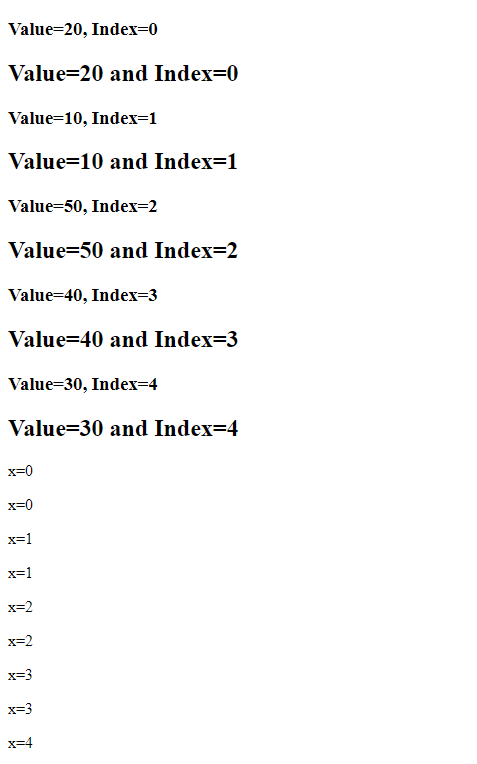
1. let, const keywords
2. class, super, extends keywords
3. template strings
4. arrow functions

Template String: This avoids lot of string concatenation you do with +, it uses back tick(`) character to include strings & javascript expressions without breaking the string.

let x = ‘<p>Hello ’+username+’</p>’;  
let y = `<p>Hello ${username}</p>`



Output:

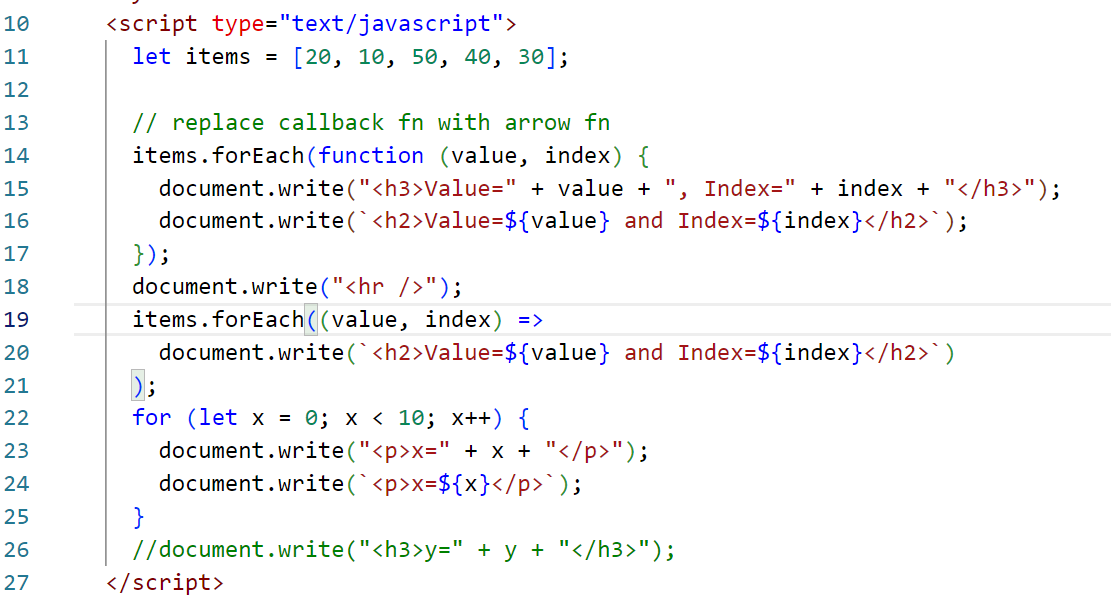


Arrow function:

It is a replacement for the callback functions, to simplify the syntax

|  |  |
| --- | --- |
| Callback Function | Arrow Function |
| function(x, y) {   console.log(x, y); } | (x, y) => console.log(x, y)  Note: If the body has only one line code, then { } is optional |
| function(x, y) {  return (x + y); } | (x, y) => x + y;  Note: By default arrow function returns the expression without return keyword |
| function(x, y) {   console.log(x, y);  return (x + y); } | Note: If arrow function uses { }, then it must use return keyword if necessary  (x, y) => {  console.log(x, y);  return x + y; } |
| items.forEach(function(x, y) { } ); | items.forEach((x, y) => { .. } ); |

Arrow function demo



Accessing & Manipulating the DOM

DOM stands for Document Object Model, it is a tree like structure of the HTML elements loaded in the browser, it can be accessed using document object which provides methods like

* getElementById(“id”): Returns the element having a matching id
* getElementsByTagName(“tag”): Returns an array of elements matching the name
* querySelector(“selector”): Returns the first element matching to the selector, here selector can be id, class or tag name
* querySelectorAll(“selector”)

<p id = “p1”>  
<p id = “p2”>  
<p class=”c1”>  
<p class=”c1”>

document.getElementById(“p1”): returns <p id=”p1”>  
document.querySelector(“#p1”): returns <p id = “p1”>  
document.querySelector(“.c1”): returns <p class=”c1”> first p.c1 is returned  
document.querySelectorAll(“.c1”): returns an array of [p.c1, p.c1]  
document.querySelectorAll(“p”): returns all the <p> elements  
document.getElementsByTagName(“p”): returns all the <p> elements

How to add Styles to the element in Javascript  
You need to add style attribute to the element at runtime

|  |  |
| --- | --- |
| HTML/CSS | Javascript |
| <p style=”color:red;background-color:yellow”> | let element = document.querySelector(“p”);  element.style.color=’red’; element.style.backgroundColor=’yellow’ |

Accessing the elements using event object

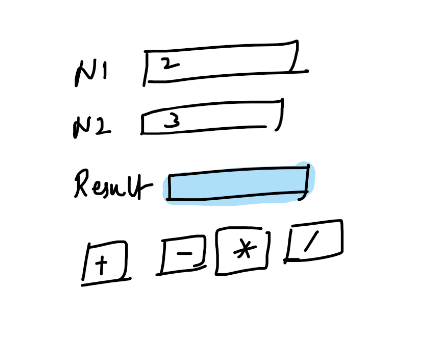
event: It is an object that is generated by an element, when an element generates the event, you can use event.target to know which element generated the event.

<p onclick=”hello(event)”>Hello World</p>  
hello( event ) { let ele = event.target }

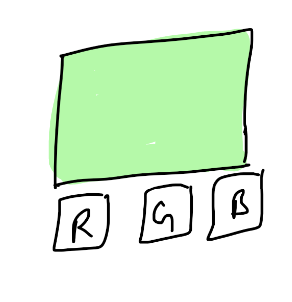
<div onclick=”hello(event)”>  
 <p>hello</p>  
 <h2>Welcome</h2>  
 <img src=”…”>  
</div>

Activity:

1. Create 3 textbox where you can enter numbers on 2 text box and the 3rd text box will display the result, create 4 buttons like add, subtract, multiply & division



1. Create 3 buttons with names like RED, GREEN, BLUE, on clicking these buttons add a background-color



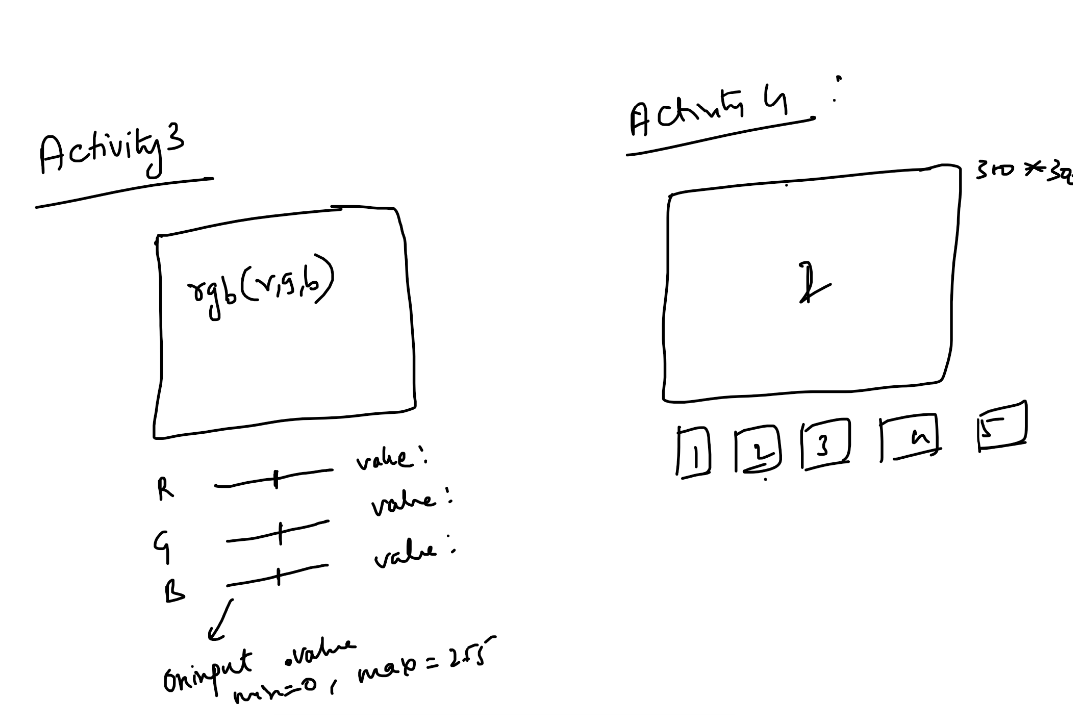
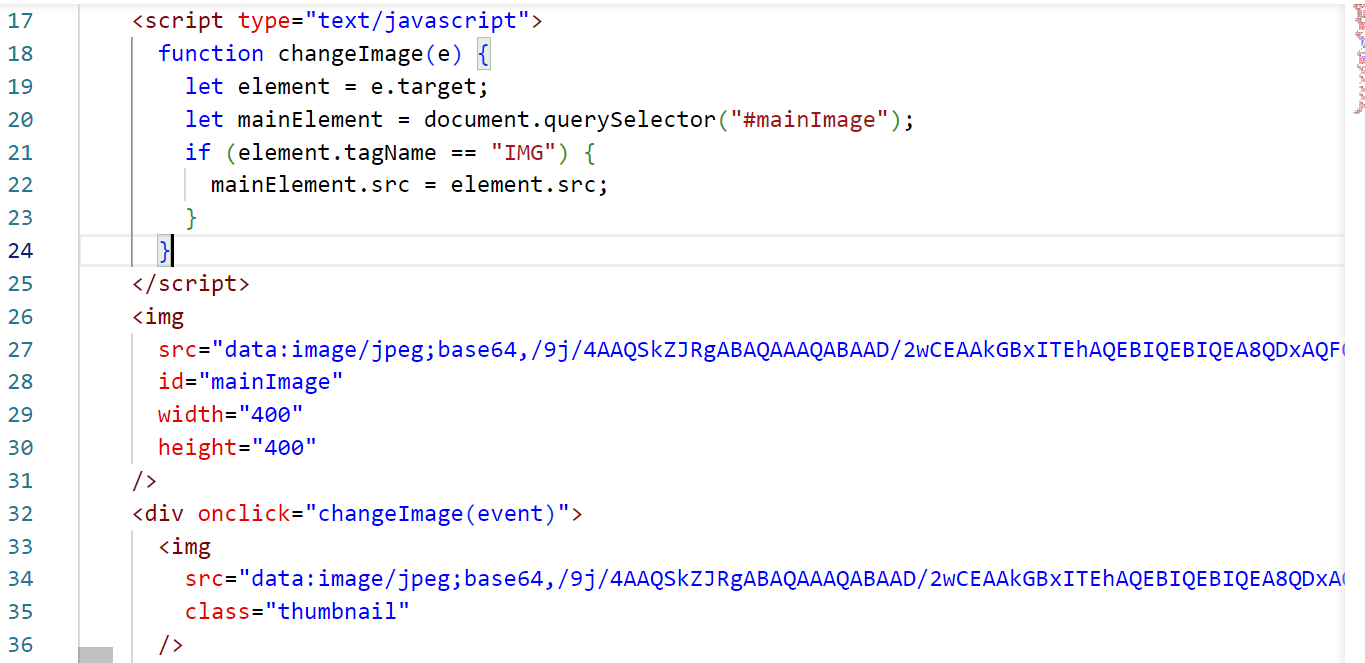


Image Changing Activity Solution





JQuery: It is a javascript library used to quickly manipulate DOM, it provides many inbuilt functions which are written in javascript

Javascript:

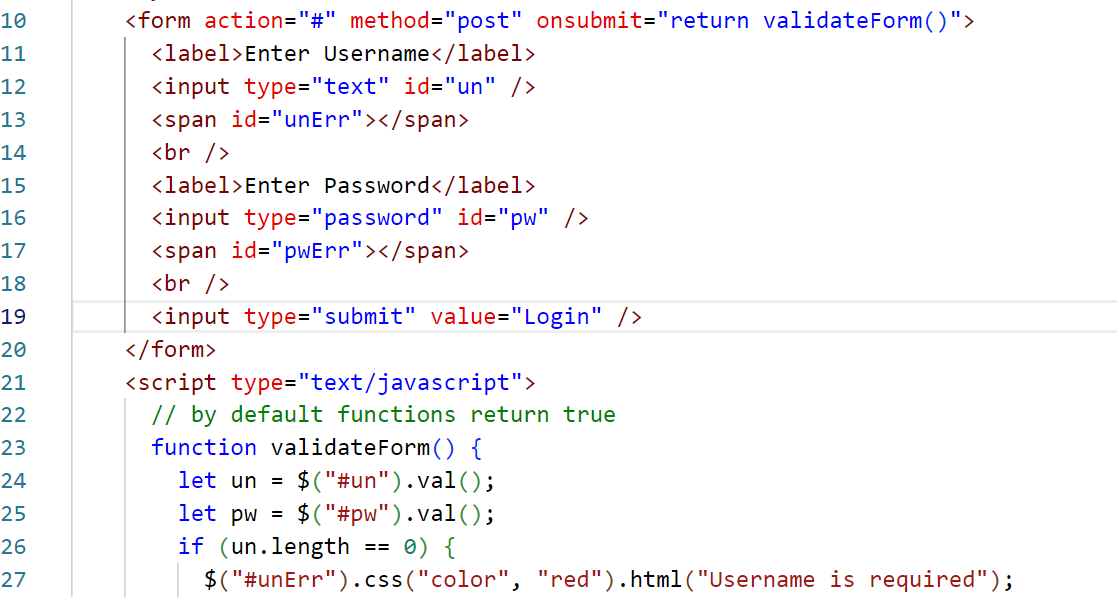
let element = document.querySelector(“#i1”);  
element.innerHTML = ‘<p>Some Content</p>’;  
element.style.color=’red’;  
element.style.backgroundColor=’yellow’;  
<p onclick=”test()”>…</p>  
<p onclick=”test()”>…</p>

JQuery:

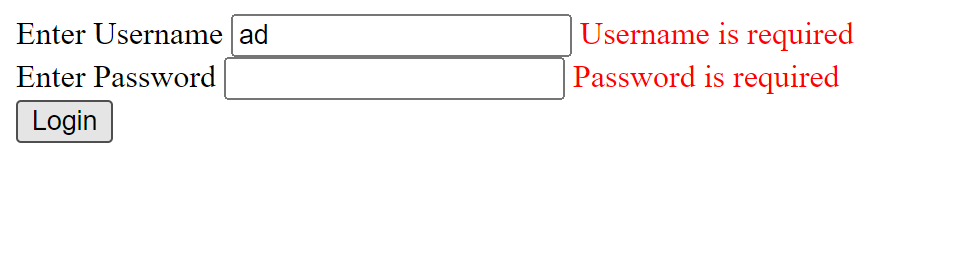
let element = $(“#i1”);  
element.html(‘<p>some content</p>’);  
element.css(‘color’, ‘red’);  
element.css(‘background-color’,’yellow’);  
$(“p”).click(function(){…});

Form Validations

Whenever user inputs invalid data we need to stop submitting the form, this can be done through validation.



Output:



Activity:

In the above form the error message is not disappearing when the username or password has value, that needs to be implemented.

Spring Framework

What is a Framework

* It helps you to develop complex applications in a simple way with fewer code
* 60% of the work will be done by framework - some common features framework only implements

Spring Framework is a java framework which helps to develop various kinds of applications

1. Desktop
2. Web
3. Enterprise
4. Cloud based applications

Things spring framework takes care

1. Design patterns
2. Object creation & initialization
3. Dependency Injection
4. Database connection

Design patterns:

Spring implements many design patterns required for your application like factory, singleton, proxy and so on.

Object Creation & Initialization

Spring takes care of creating objects & initializing reusable objects

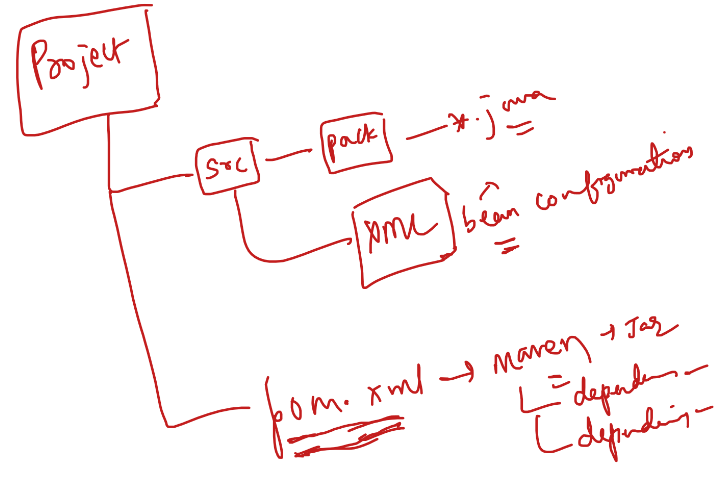
Dependency Injection

Spring uses this process to supply an object (injecting the object) to another object

Database Connection

Spring takes care of establishing the database connection by using the db credentials provided in some configuration files, developers don’t need to write DB connection logics (ex: DriverManager.getConnection(url, username, password);…)

Spring project structure



Libraries to add

1. spring-context: This library helps to implement all the spring core features like providing design patterns, dependency injection

Note: You must add this library in the pom.xml

<dependencies>  
 <dependency>..</dependency>  
 <dependency>..</dependency>  
</dependencies>

Visit: <http://github.com/Kishor-C/Framework-Configuration>

Goto Spring/Spring Core/, open beans.xml / Download that xml file using Download raw file option

Spring Bean Configuration file

* It is an xml file
* It can have any name
* It will have all the class configurations which spring framework needs to instantiate.

Spring Container/Spring IOC(Inversion of Control)

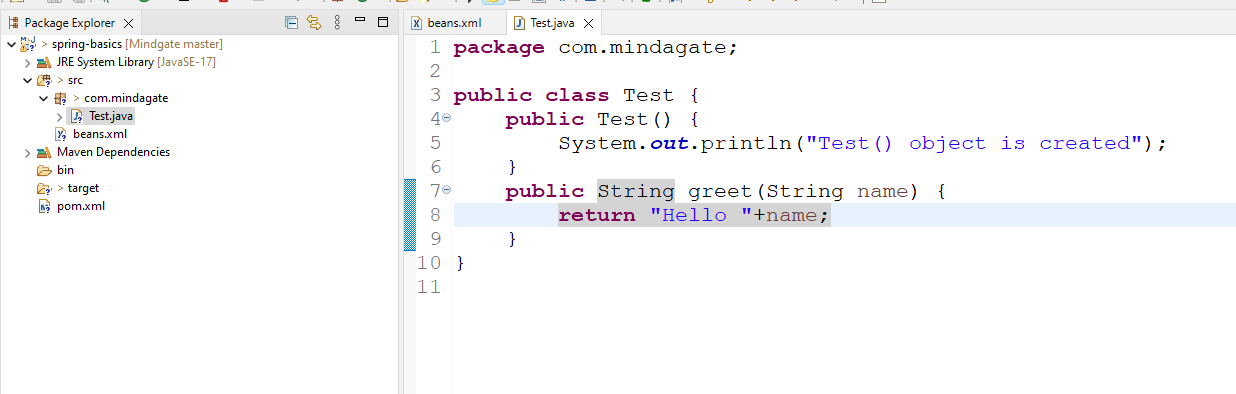
This takes care of creating the objects, maintains the objects & supplying the dependencies to other objects

First program

Using the factory pattern of spring to get the object

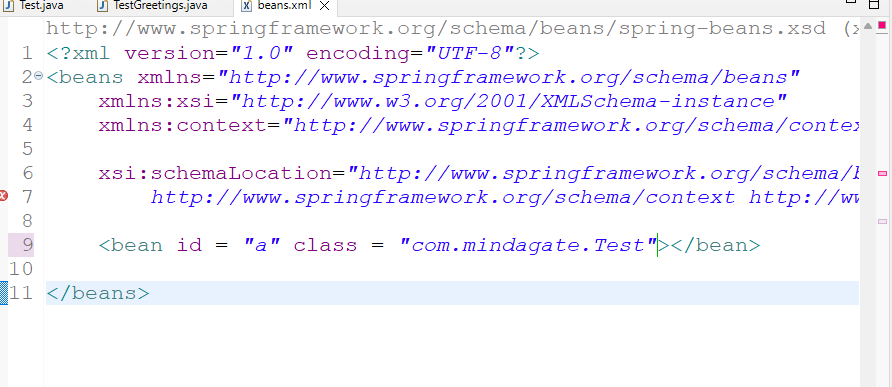
1. Test class with a greet() method
2. You will configure the beans.xml file to let spring to create the object
3. You will create main method which tries to get the object of `Test class` from spring container & calls the greet

src/com/mindgate/Test.java

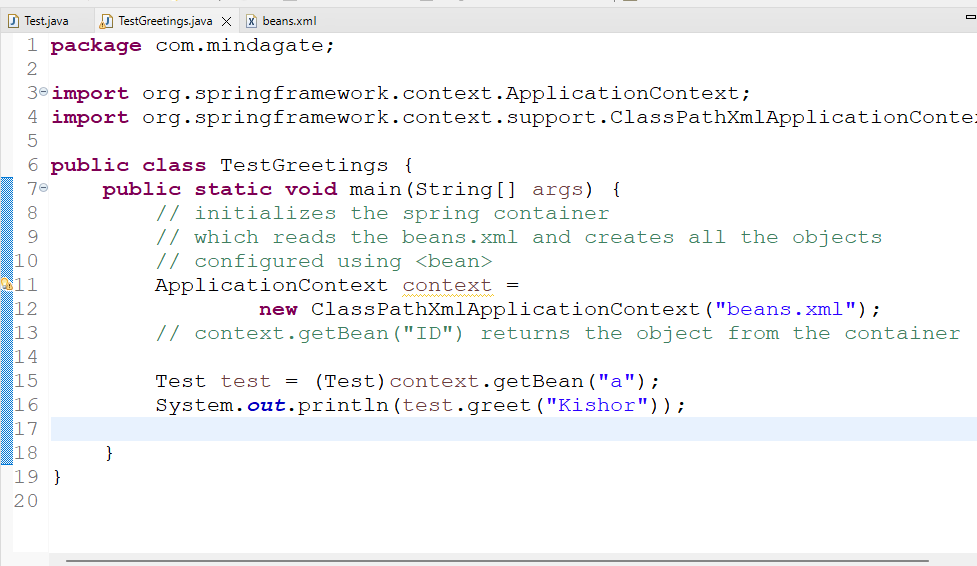


Spring provides an inbuilt class called ClassPathXmlApplicationContext that will initialize the spring container & create objects by reading the xml file  
ApplicationContext context = new ClassPathXmlApplicationContext(“beans.xml”);  
Test t = (Test)context.getBean(“a”);

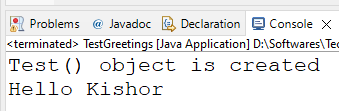
beans.xml



src/com/mindgate/TestGreetings.java



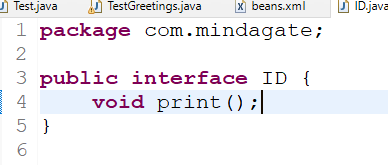
Output:



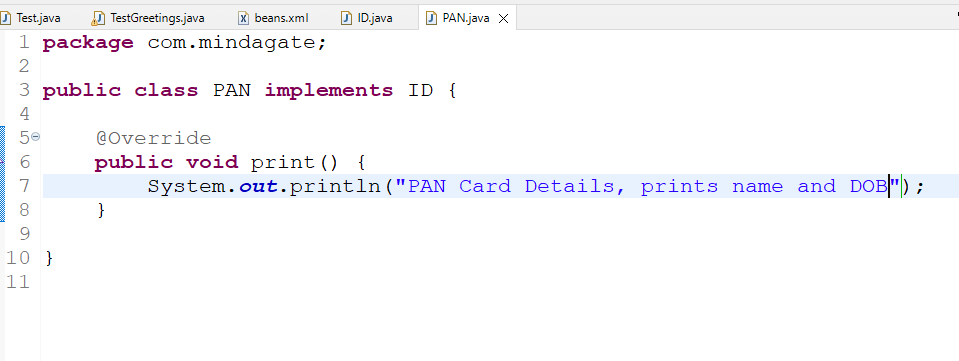
Using the interface to get multiple implementation object

1. interface ID - print()
2. class PAN, VoterID & Aadhar implements ID
3. XML configuration for any one of the implementation class
4. Main class to get the object & call print()

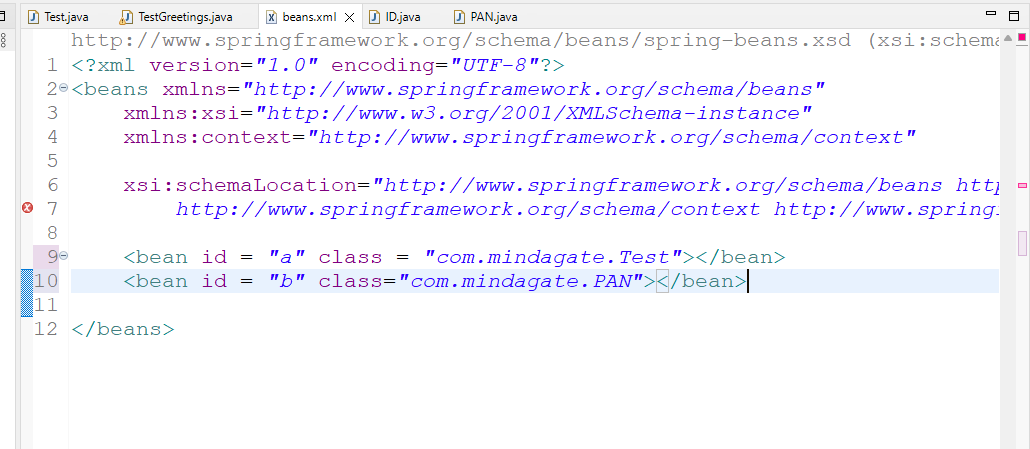
src/com/mindgate/ID.java



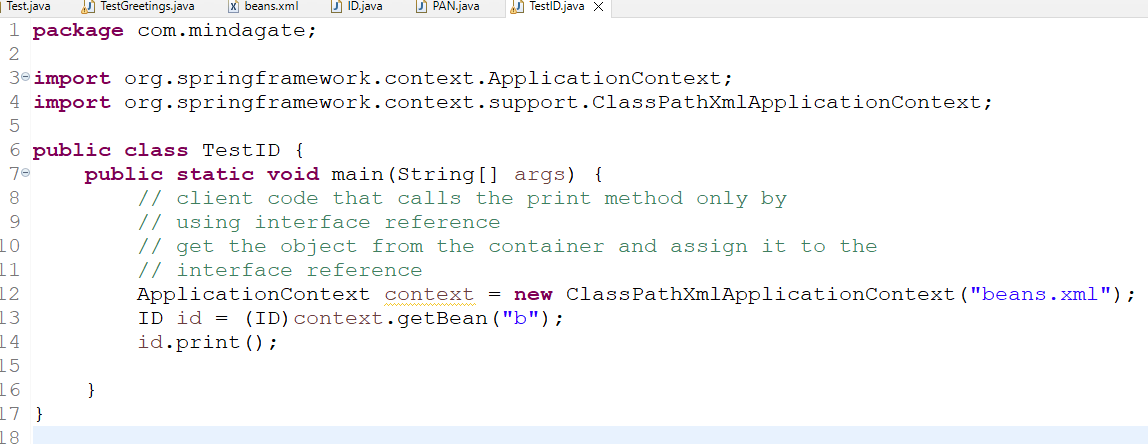
src/com/mindgate/PAN.java



beans.xml



src/com/mindgate/TestID.java



How to check this code is loosely coupled code

* Create another implementation for ID interface & implement print & change the beans.xml for the existing <bean> id = b, to instantiate the new implementation
* Run the main method you will see the print method of new implementation getting called, without making any changes in the main method (this is the client code)

Dependency Injection & Its types

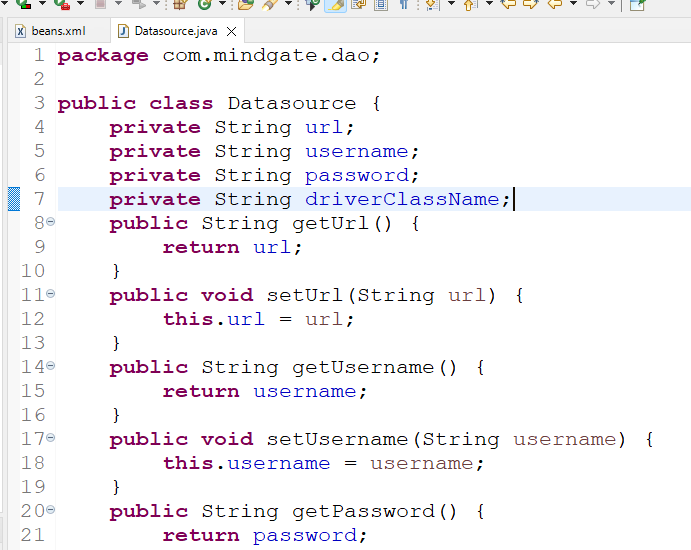
Dependency Injection is nothing but object creation & initialization & supplying these object to other objects, there are two types

1. setter injection: This initializes through setter method, in xml you will use <property> tag
2. constructor injection: This initializes through constructor, in xml you will use <constructor-arg>

Activity: Create a new project with the name `spring-di`, convert it to maven, add spring-context dependency in the pom.xml, then keep bean configuration file in src folder.

Generally database credentials & other application related properties are kept in XML, application needs to read those properties through the object that maintains these properties.

src/com/mindgate/Datasource.java



Datasource object maintains DB credentials, this object needs to be supplied to DBOperations object, before that we will test Datasource object properties in main by considering main as a DAO layer class

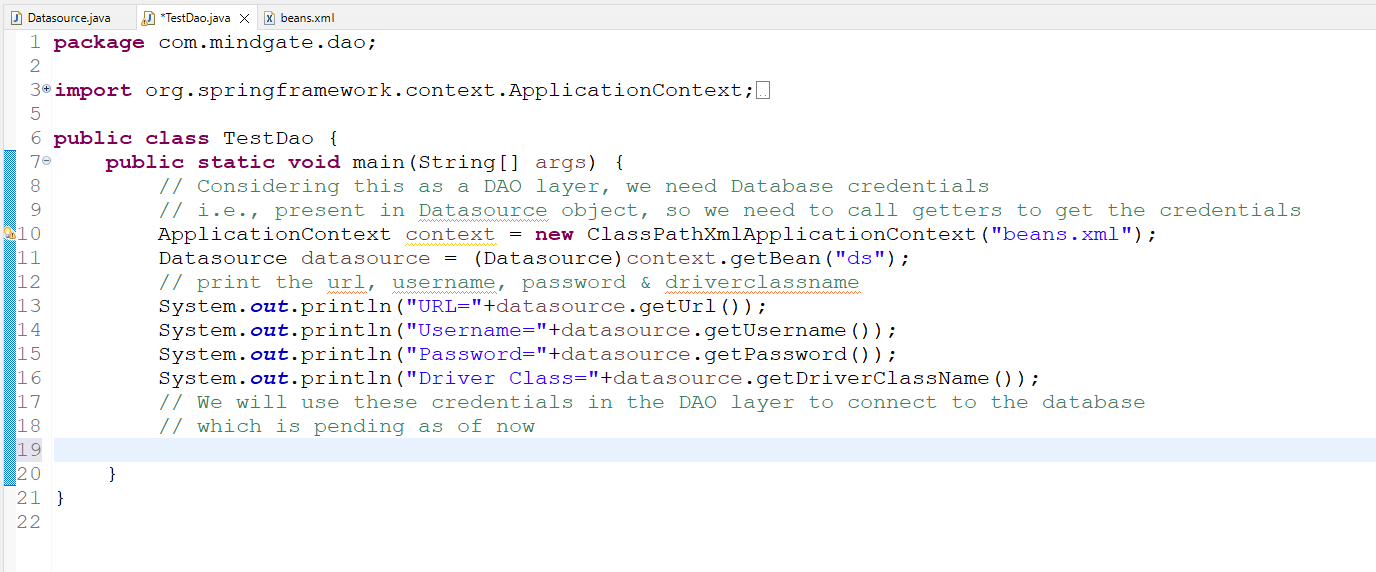
beans.xml



Dao layer will have CRUD operations

It needs to connect to perform CRUD operations, to connect it needs Datasource object

src/com/mindgate/TestDao.java



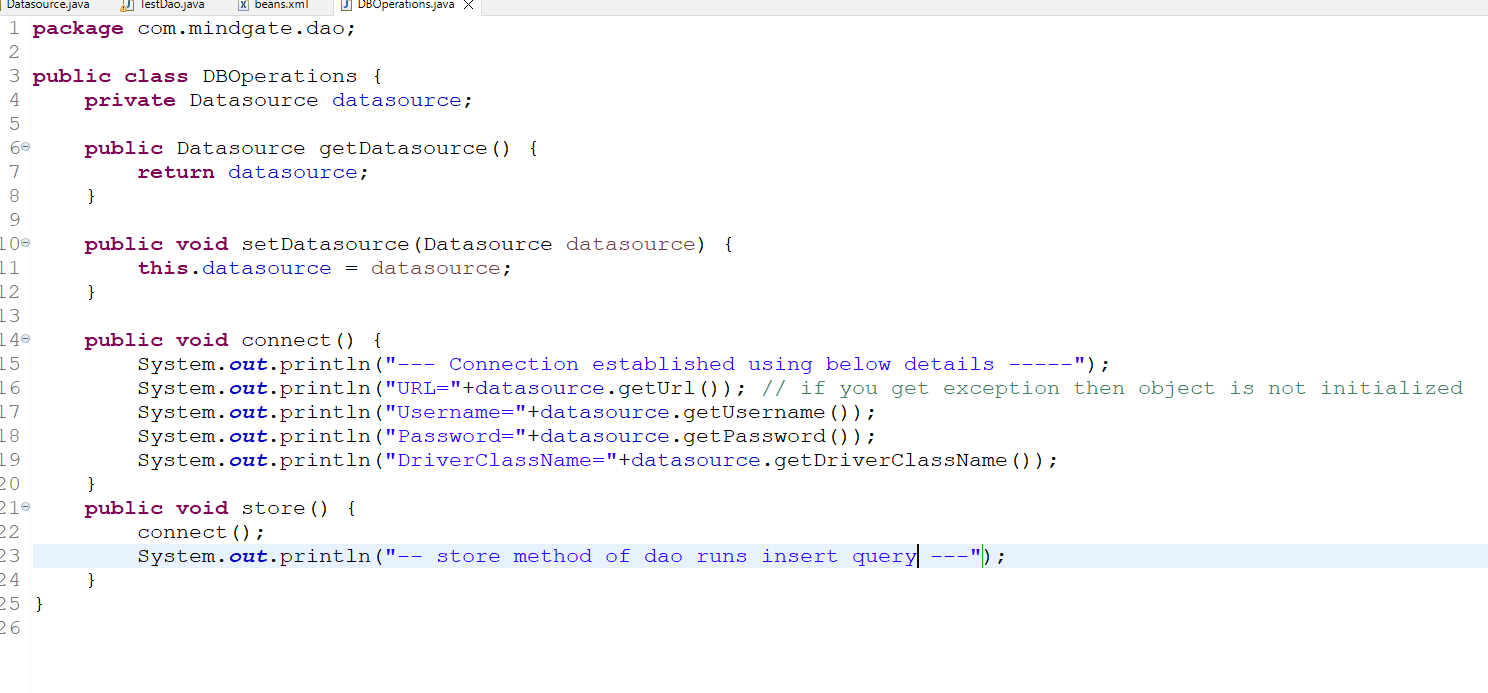
How to supply a complex object to another object

We must have a property that needs an object

class DBOperations {   
 private Datasource datasource;   
// you must configure this property in xml such that spring container supplies the object  
}

<bean id = “ds” class = “com…..Datasource”>…</bean>  
<bean id = “dao” class = “com…DBOperations”>  
  
 DI happens here  
 <property name = “datasource” ref=”ds”></property>  
</bean>

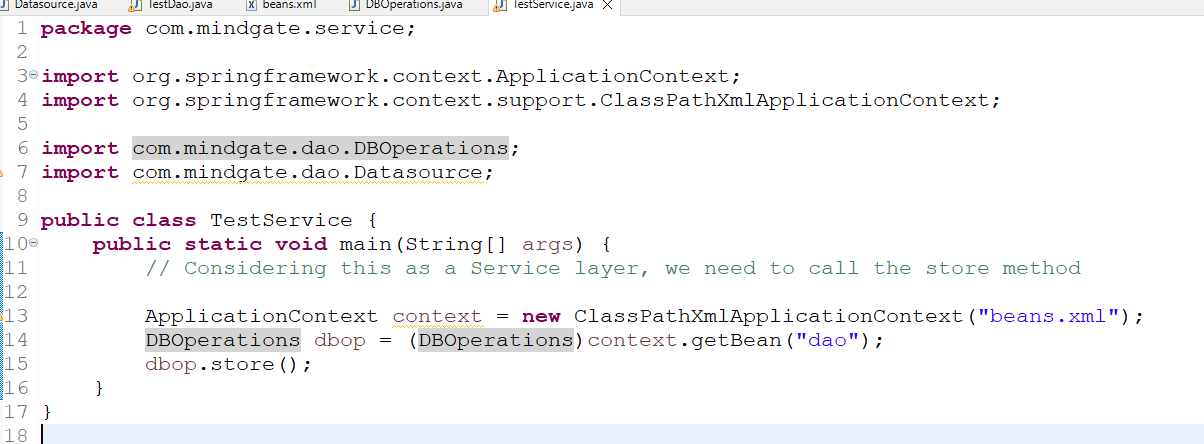
DBOperations.java



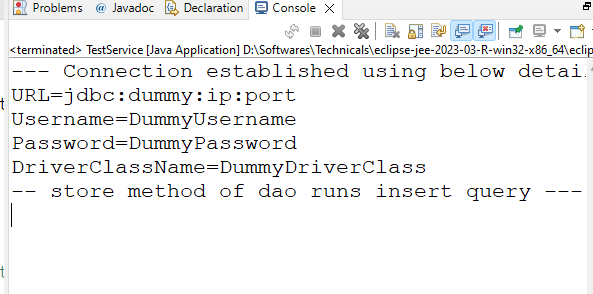
beans.xml



Service layer will call store() method, however we will test that through main method



Output:



There’s another type of DI which is constructor injection, where you initialize the object with constructor argument

1. You must have constructors with parameters
2. You must use <constructor-arg> tag with index & value attributes to pass the value to the properties, the 1st parameter uses 0th index

Activity: Create a Datasource2 class with url, username, password & driverClass properties & create 4 argument constructor to initialize these properties & only getters for the properties (no setters), create DBOperations2 class with a property of type Datasource2, then setters & getters for that Datasource2 property, then connect() method that prints Datasource2 properties & store() method that calls connect() method and prints some store() output, configure the beans.xml to create Datasource2 bean with <constructor-arg> & pass that bean to DBOperations2, from the main method call the store() method of DBOperations2 by getting its object from the container

Summary

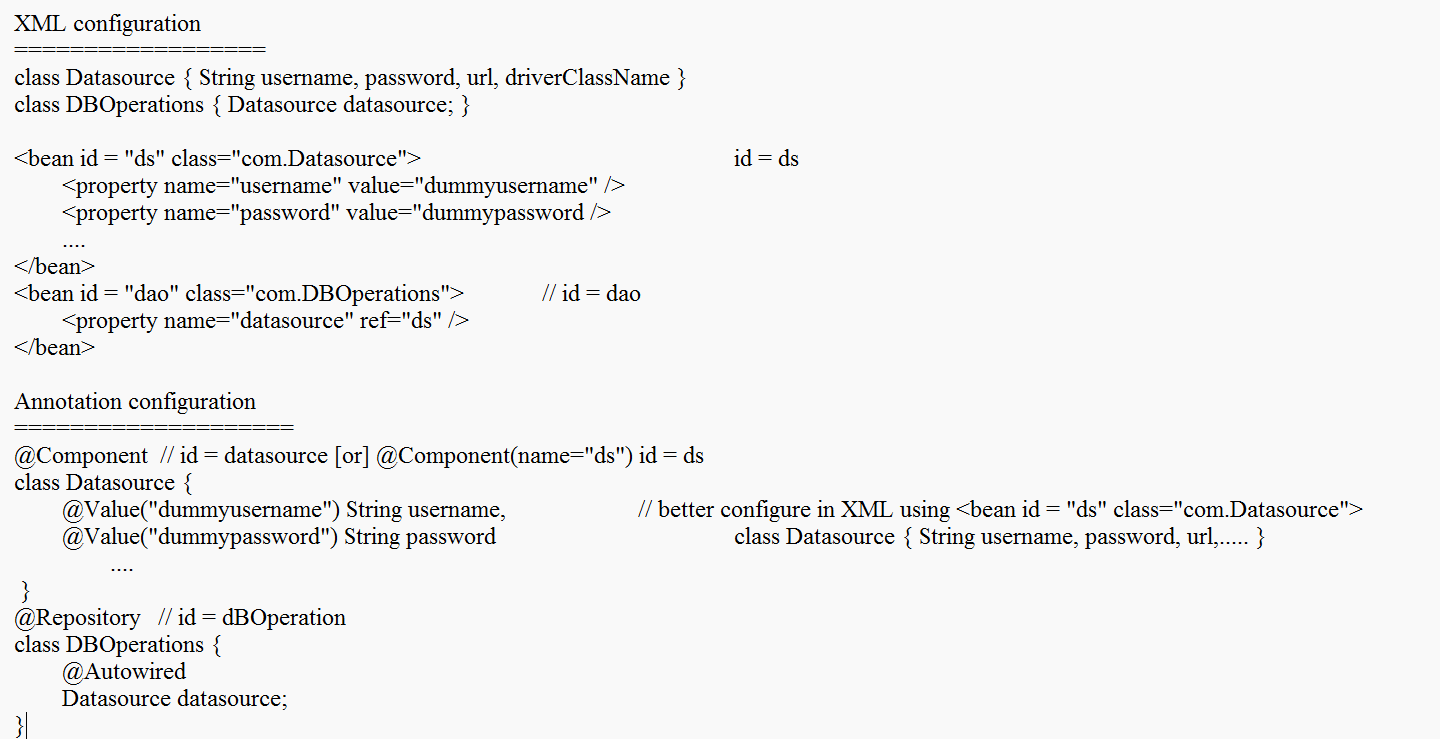
Spring Framework provides lot of common features every application needs so that you can develop the applications faster

1. Design patterns
2. Object initialization
3. Dependency Injection
4. Exception Handling
5. Database connection & closing

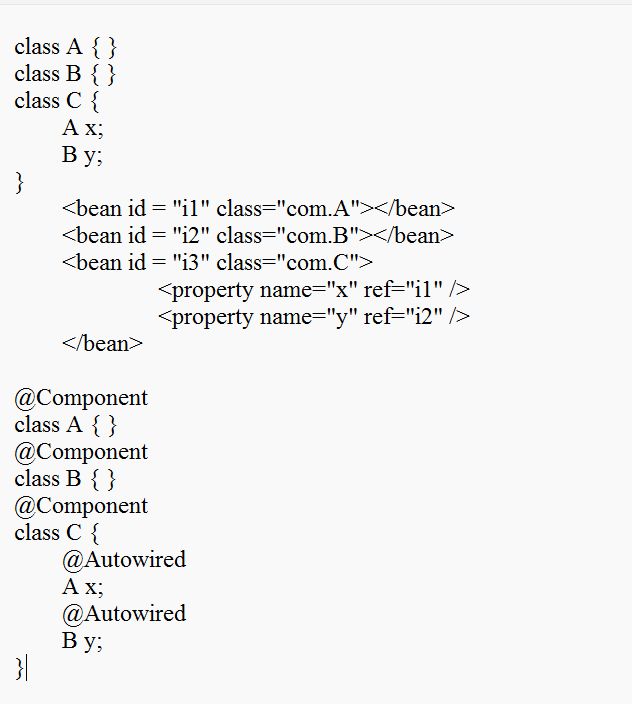
Annotation based configuration

Spring helps developers to configure the beans with annotations so that you can avoid using XML in some cases

|  |  |
| --- | --- |
| XML Configuration | Annotation Configuration |
| <bean id = “a” class=”com.Test”></bean> | @Component / @Service / @Repository / @Controller  class Test { } |
| <bean id = “b” class = “com.DBOperations”>  <property name=”datasource” ref=”ds/> </bean> <bean id = “ds” class=”com.Datasource”>  <property name=”username” value=”dummyusername”. /> </bean>  value attribute @Value ref attribute @Autowired | <bean id = “ds” class=”com.Datasource”>  <property …. /> </bean> [ or ] @Component  class Datasource {   @Value(“dummyusername”)  private String username;  @Value(“dummypassword”)  private String password; }  @Repository  class DBOperations {   @Autowired  Datasource datasource; } |
|  |  |

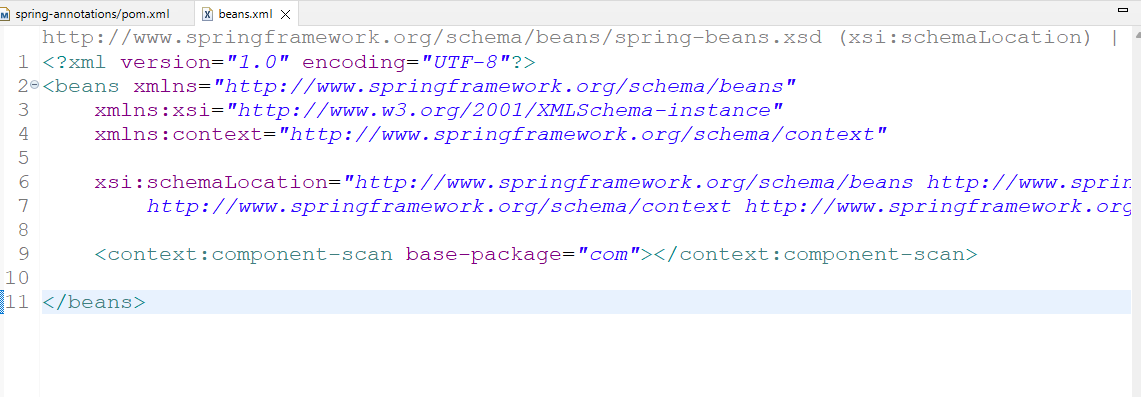


Scenario when annotation makes the configurations simple

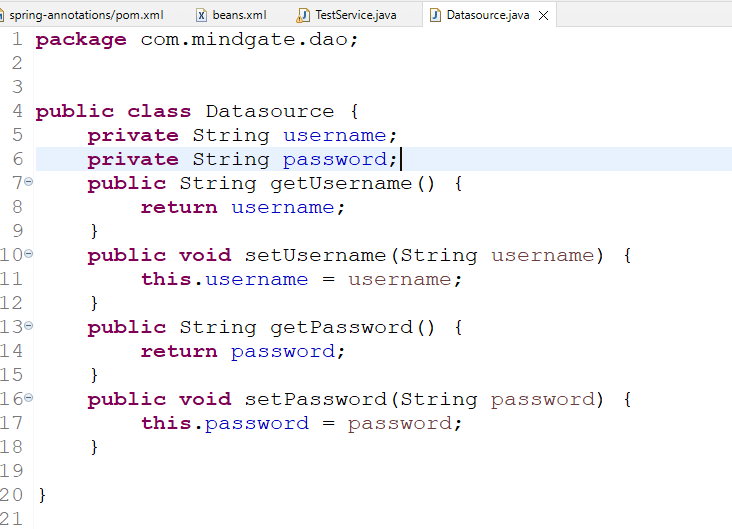


<component-scan base-package=”com”> tag: This tag must be used to specify the spring it has to scan for annotations

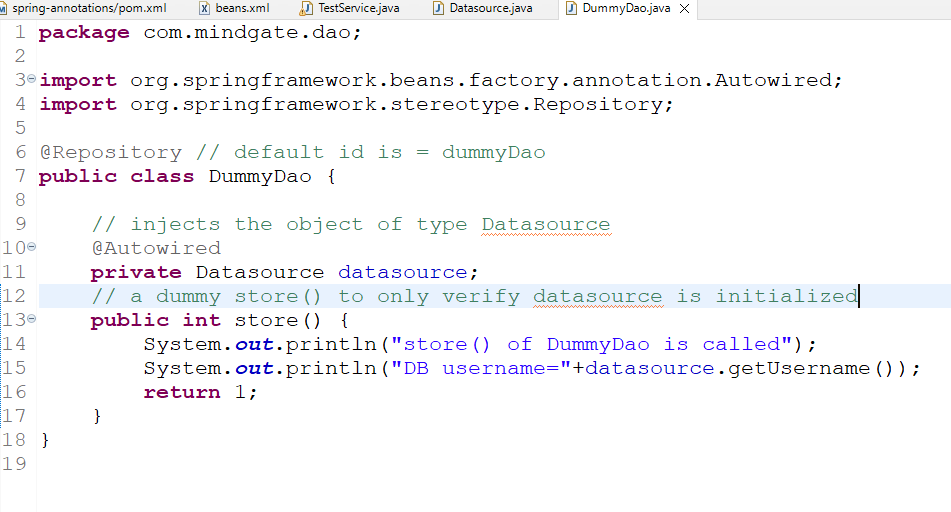
beans.xml



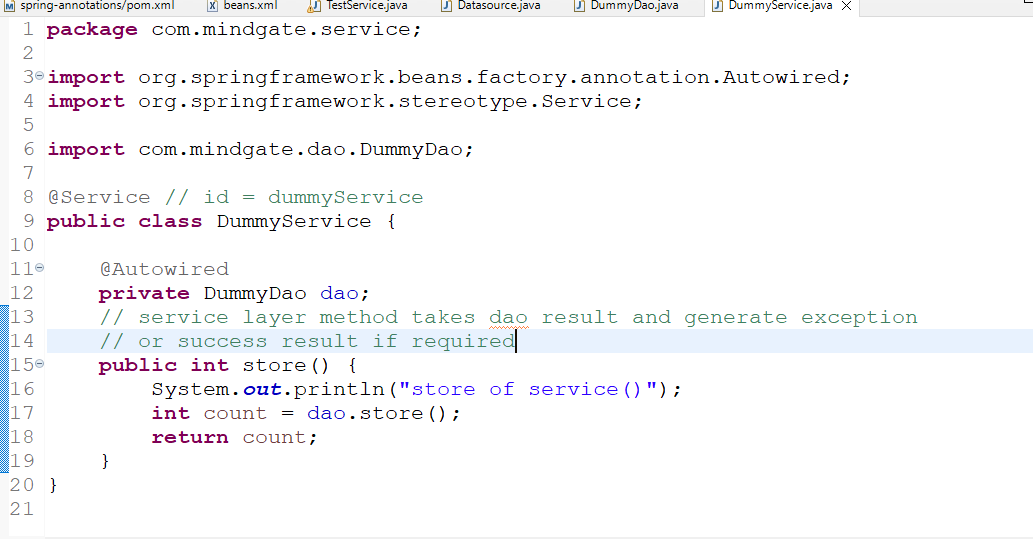
src/com/mindgate/dao/Datasource.java



src/com/mindgate/dao/DummyDao.java

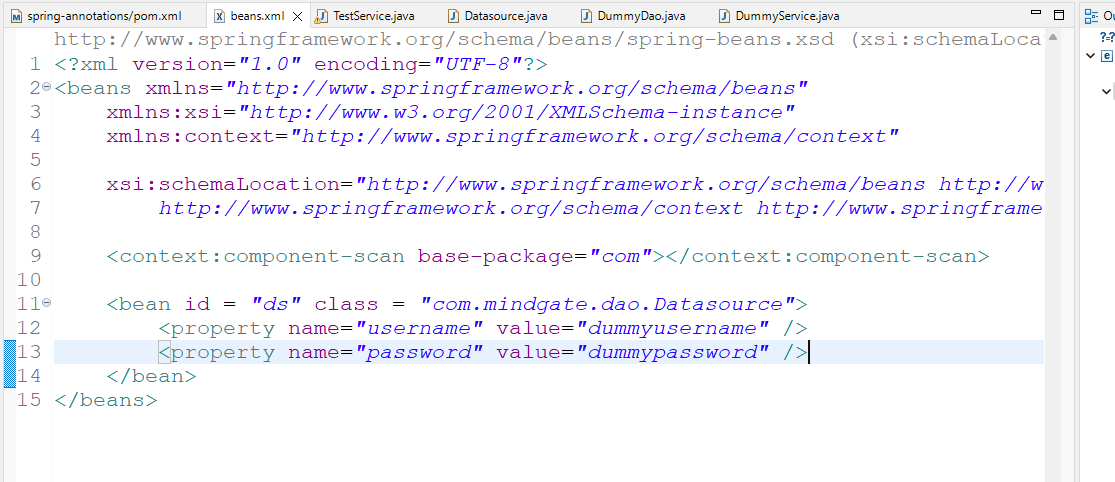


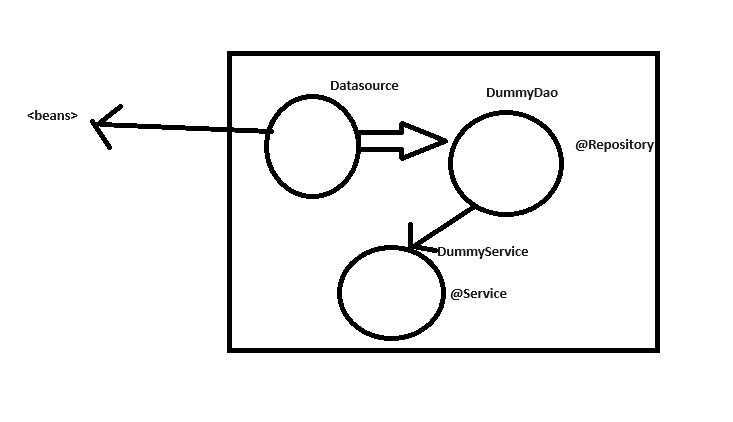
src/com/mindgate/service/DummyService.java



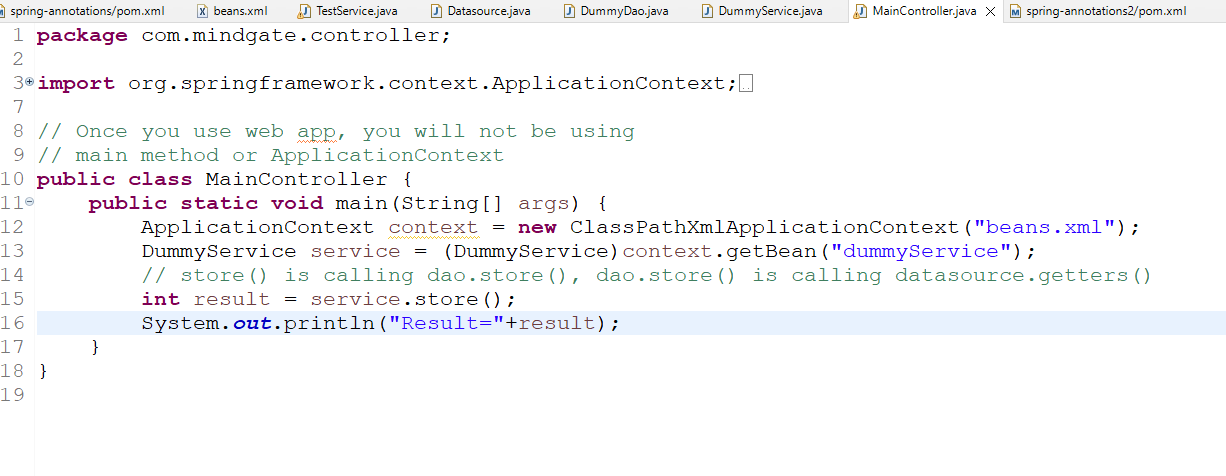
Note: Dao layer will have sql queries  
Note: Service layer will have business logics, like calling debit & credit of DAO & deciding the status of the transaction

beans.xml

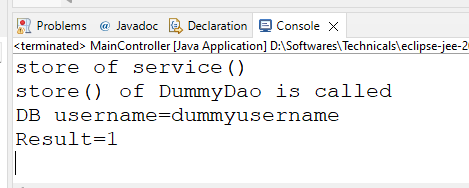
Spring container creates 3 objects in the spring container



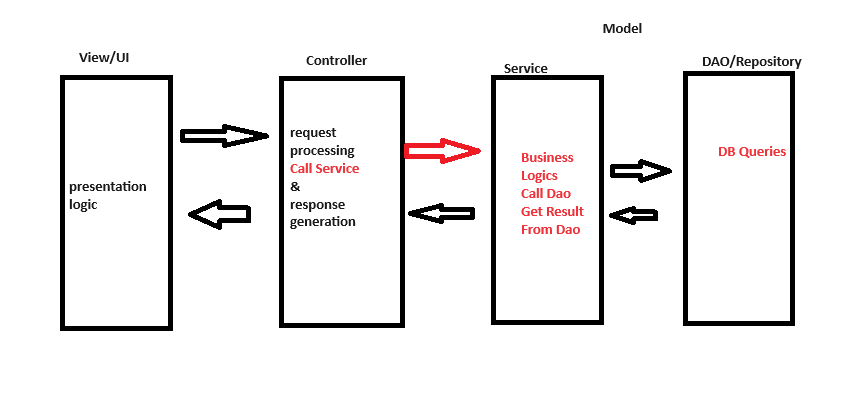
src/com/mindgate/controller/MainController.java



Output:



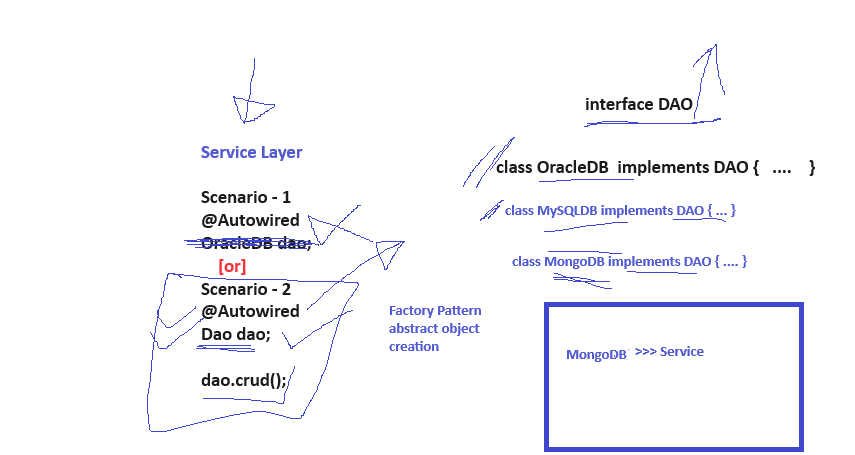
MVC architecture



How to create application loosely coupled

Flexible to the changes

1. If DAO layer is modified, then other layers must not be changed
2. Same thing is applied even in Service/controller/view layer
3. You must use interface based approach, the client code uses interface to call the methods



Spring has many modules to ease the development process

1. spring context: Takes care of DI, Container initialization, type conversion
2. spring jdbc: Simplifies DB interactions by taking care of exception handling, creating connections, setting values to the ?, closing connections

ex: in JDBC you will write  
prepareStatement.setInt(1, 200);  
prepareStatement.setString(2, “Raj”);  
prepareStatement.setDouble(3, 45000);  
In Spring JDBC: You don’t use any JDBC Api’s like Connection, PreparedStatement, ResultSet  
You have only a few Spring JDBC Api’s like JdbcTemplate which connects to the DB & has CRUD methods

i.e, jdbcTemplate.update(“insert into table values(?,?,?), 200, “Raj”,45000);

List of Spring JDBC classes

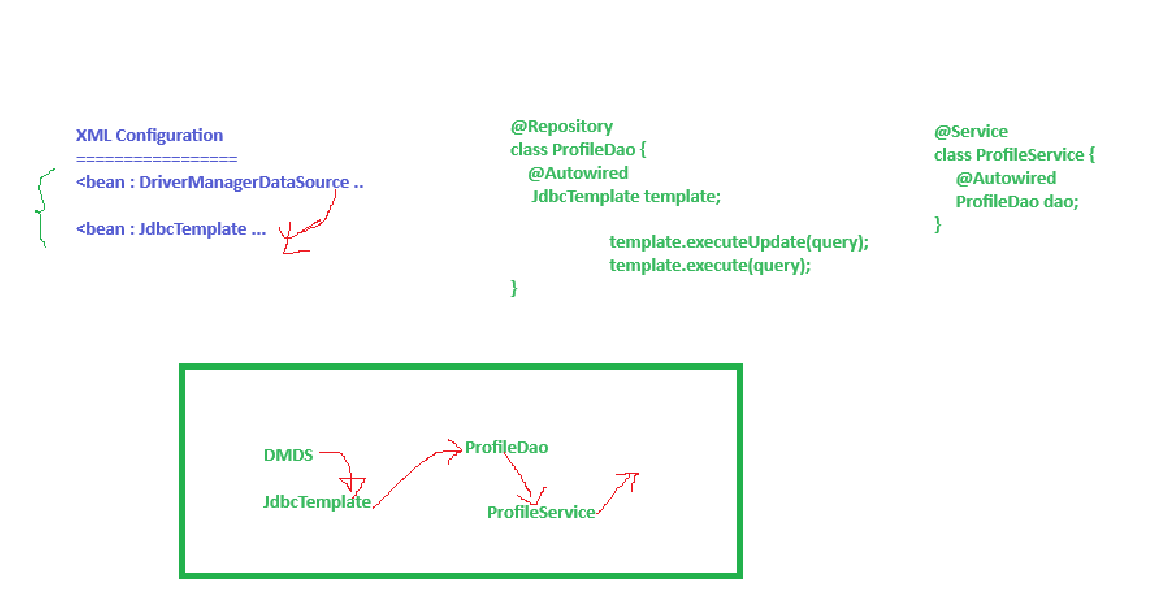
1. DriverManagerDataSource: it has properties like username, password, url, driverClassName, this class object connects to the DB & also disconnects from the DB automatically

<bean id = “ds” class = “com.springframework.jdbc.DriverManagerDatasource”>  
 <property name = “username” value = “realDBUsername” />  
 ….  
</bean>

1. JdbcTemplate: It has a depends on DriverManagerDataSource, it provides methods to perform CRUD operations like execute(sql), executeUpdate(sql) and so on

<bean id = “jdbc” class = “com.springframework.jdbc.JdbcTemplate”>  
 <property name = “datasource” ref = “ds” />  
</bean>

Dependency Flow of DMDS, JdbcTemplate, DAO & Service



Things to implement

1. Table : Employee with id, name & salary
2. Employee class (java bean/pojo class): with id, name and salary properties

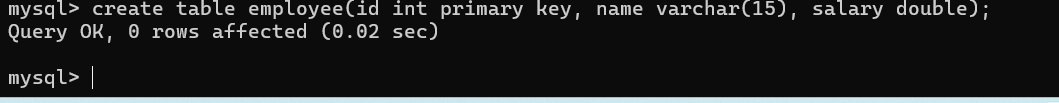
Employee e= new Employee(1, “Raj”, 35000); // fundamentals  
Employee e= new Employee(rs.getInt(1), rs.getString(2), rs.getDouble(3)); // jdbc

1. EmployeeDao interface: save(Employee), deleteById(int id), findById(int id)
2. EmployeeDaoV1 implements EmployeeDao { jdbc template code }
3. EmployeeService: Service layer that calls the DAO layer
4. Main class: This acts like controller to call the Service layer
5. beans.xml: configure DriverManagerDataSource & JdbcTemplate
6. spring-jdbc library & oracle.jar/mysql.jar needs to be added

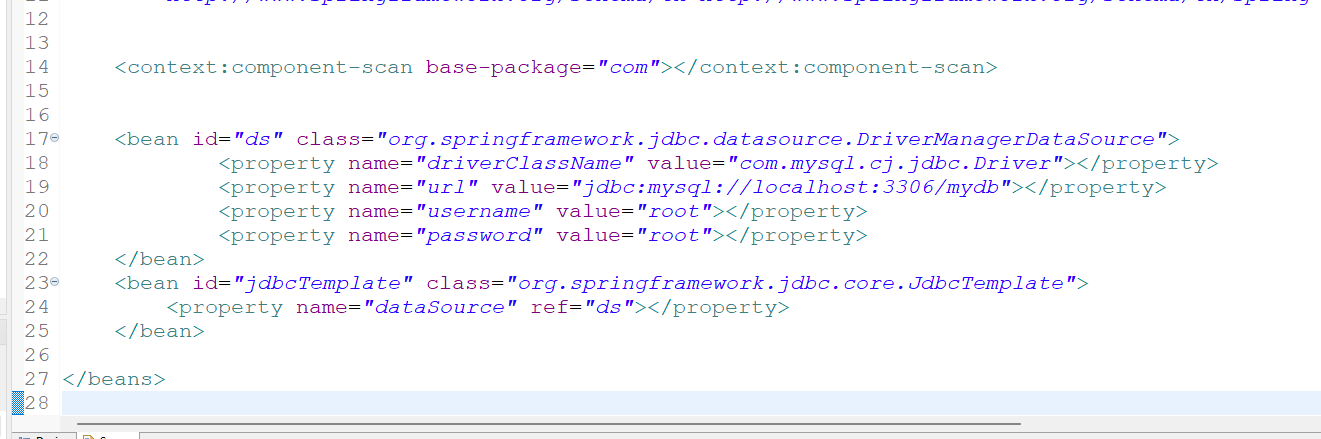
Project setup

1. New Project >> Add spring-context & spring-jdbc dependency >> Add oracle jar
2. Create a table employee with id(primary key), name & salary
3. Create beans.xml this time you will take a different xml from the GIT

Creating table

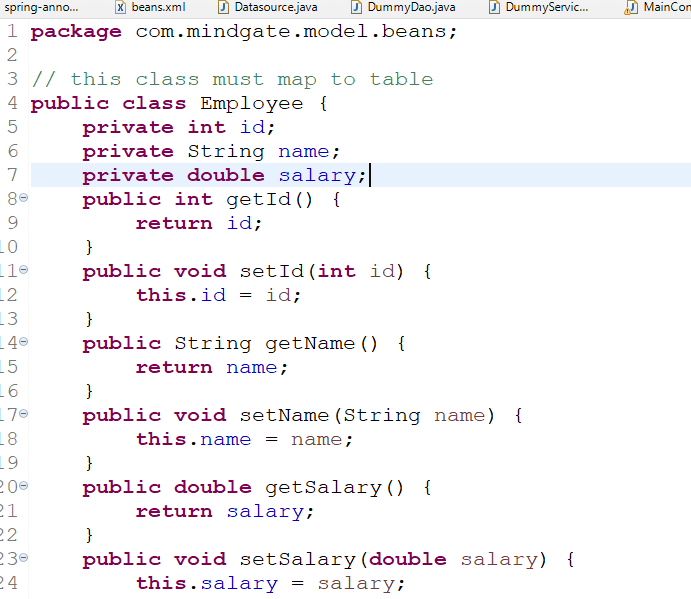


beans.xml



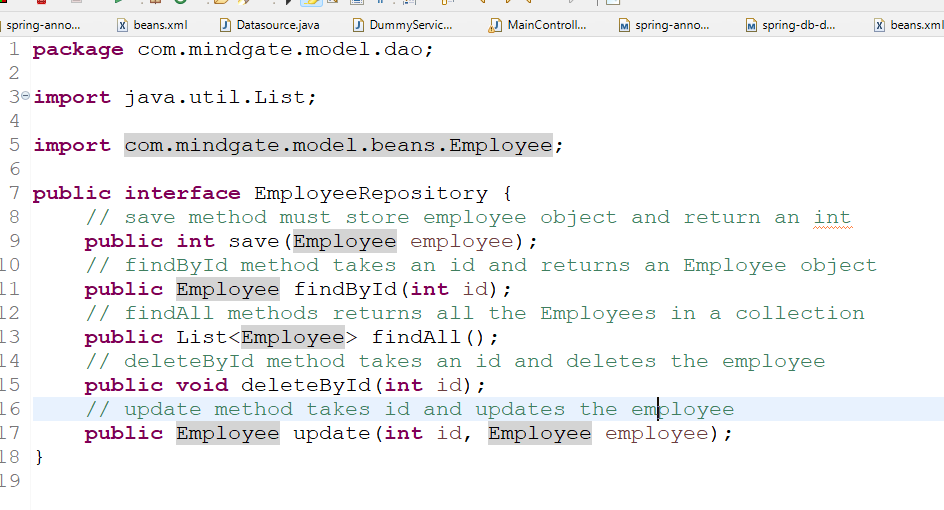
Creating a POJO/Java bean class

src/com/mindgate/model/beans/Employee.java



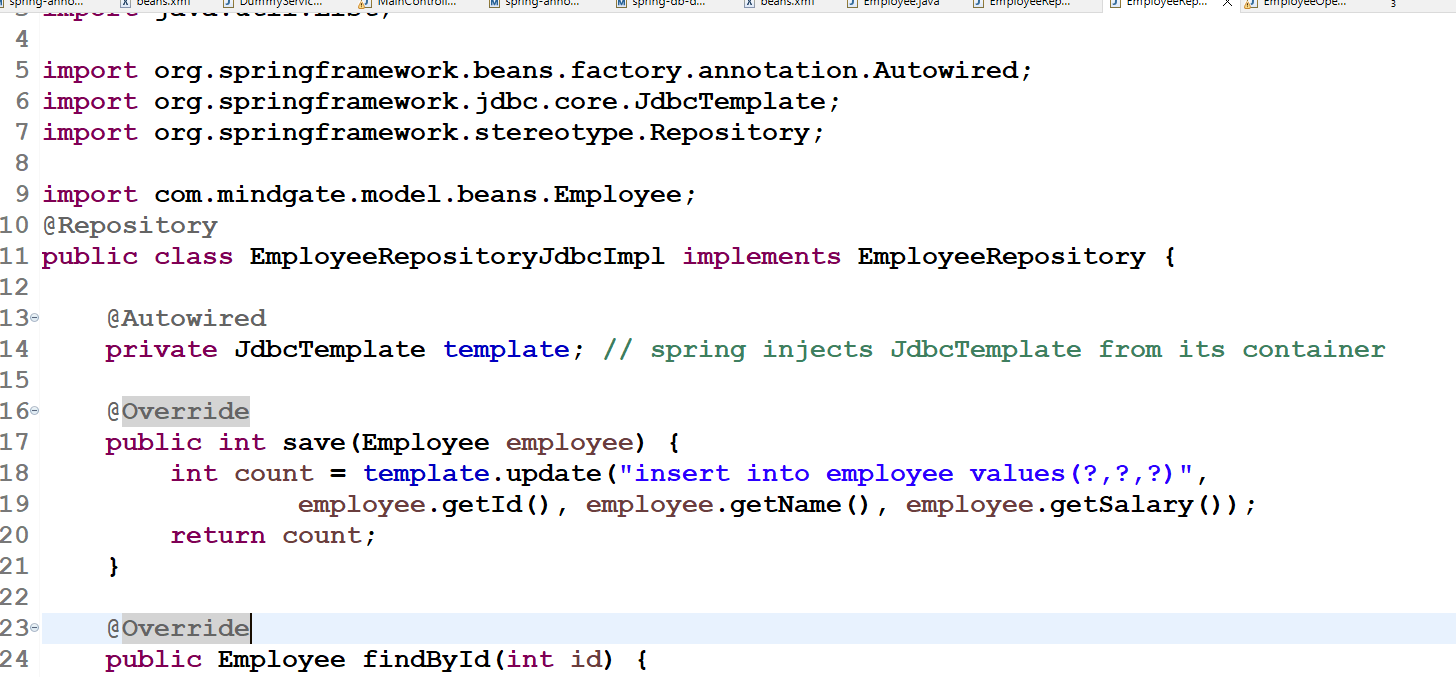
Creating an interface in the DAO layer & Implementing it

src/com/mindgate/model/dao/EmployeeRepository.java

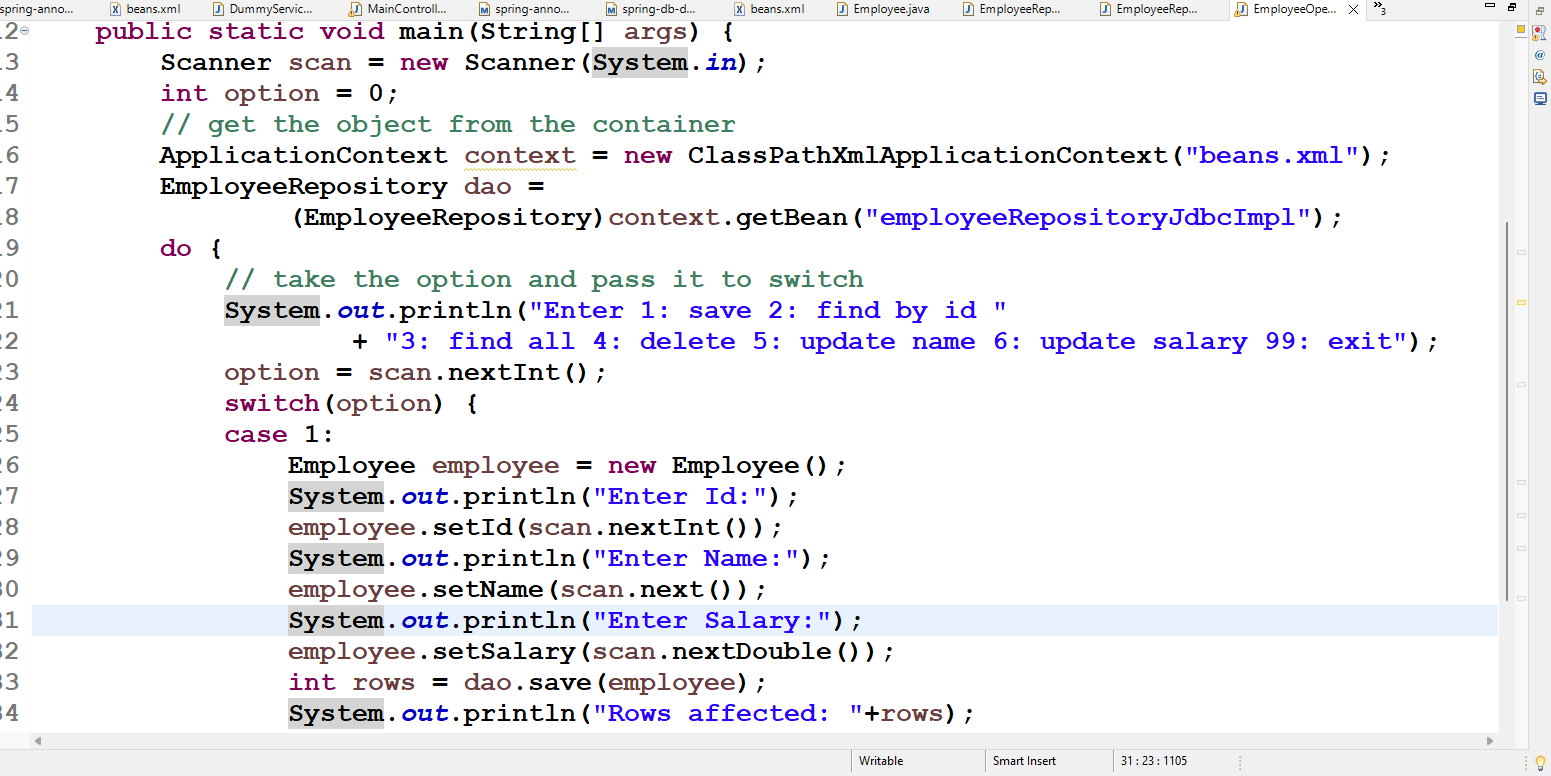


Implement EmployeeRepository in EmployeeRepositoryJdbcImpl

src/com/mindgate/model/dao/EmployeeRepositoryJdbcImpl.java



Main class that mimics like Service layer



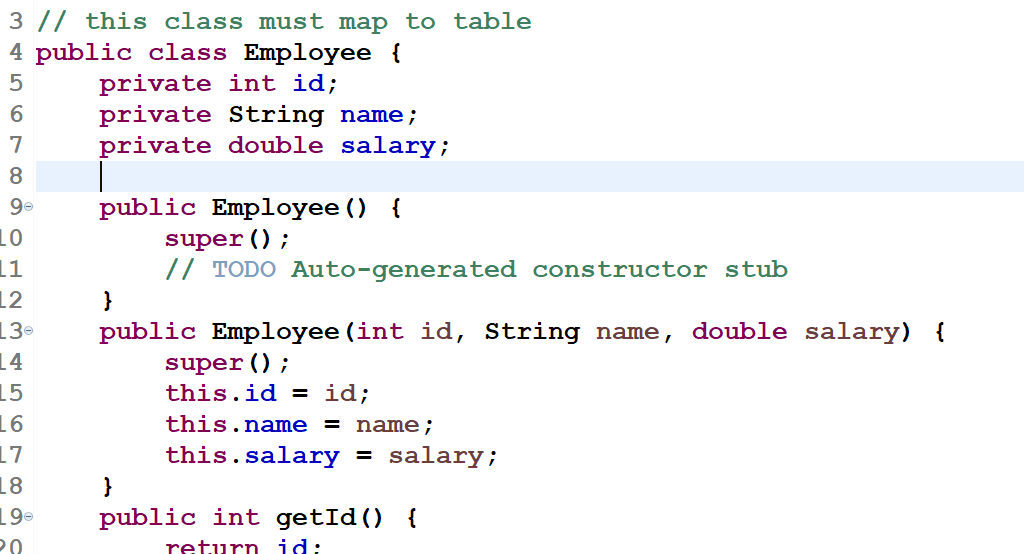
How to read records from the JDBC template

In JdbcTemplate you get a RowMapper interface that takes care of converting resultset to java object based on the select query, it can return either an object or a collection based on your query

RowMapper is having only one abstract method called mapRow(int rowNum, ResultSet rs), you can implement using Java8 lambda expression as below

(rowNum, result) -> { return new Employee(…) } [or]  
(rowNum, result) -> new Employee(…);

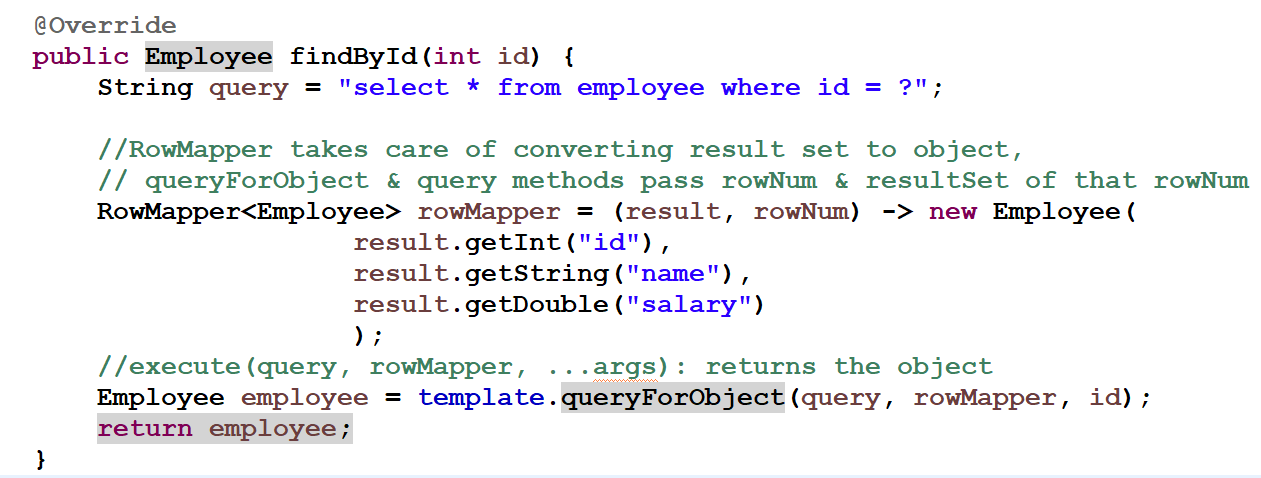
Note: First generate 2 constructors in Employee class



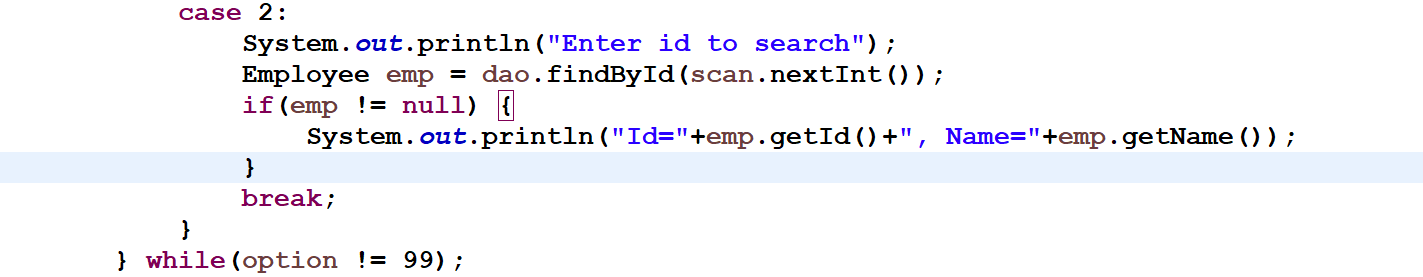
JdbcTemplate has methods that takes select queries.

queryForObject(query, rowMapperImpl, …arguments): This is used when you want a query with conditions, ideally for one object

query(query, rowMapperImpl): This is used to get all the rows



Main class to call findById(id)



Activity:

In DAO there are few methods which are not implemented, you need to implement them and call them through the main class, use parameters & return type of those methods as hint to pass the input & display the output in the main class

Handling the EmptyResultDataAccessException in the DAO layer

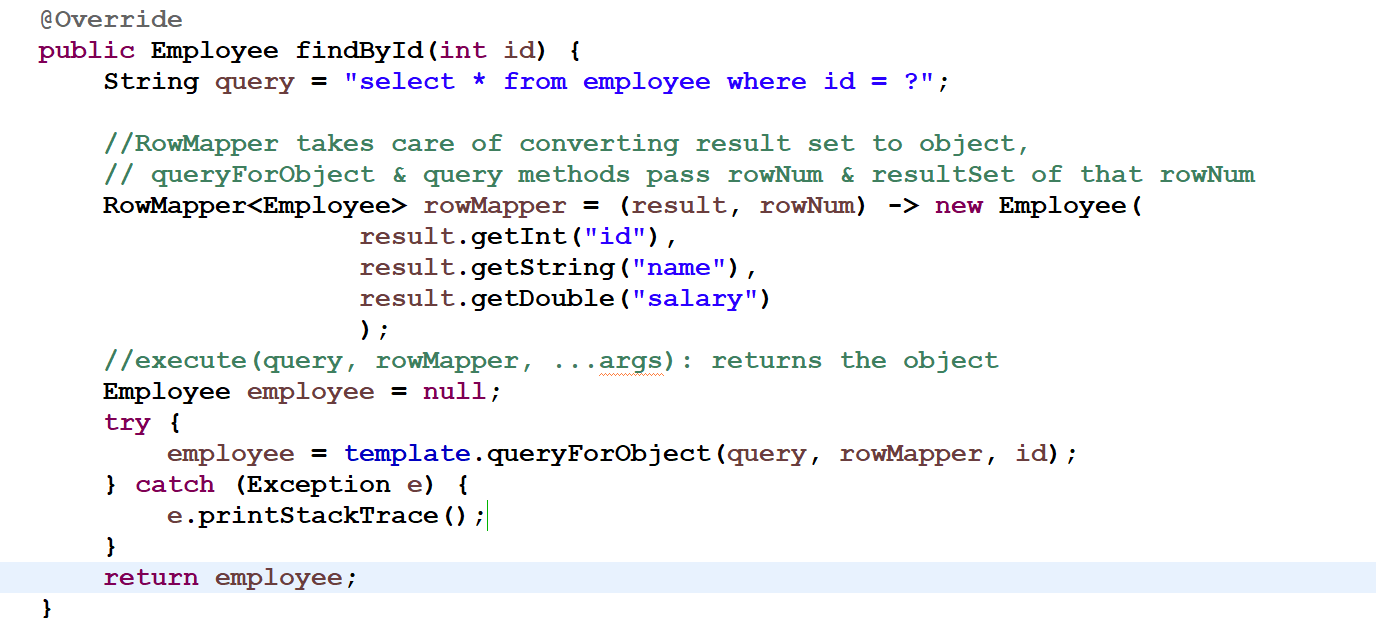
DAO layer

findById(int id) {   
Employee employee = null;  
try {  
 employee = template.queryForObject(query, RowMapper, arguments);  
} catch(Exception e) {   
 e.printStackTrace();   
}  
return employee;  
Service layer  
employee = dao.findById(value);  
if(employee != null) return employee;  
throw new EmployeeNotFoundException(“Id “+value+” not found”);

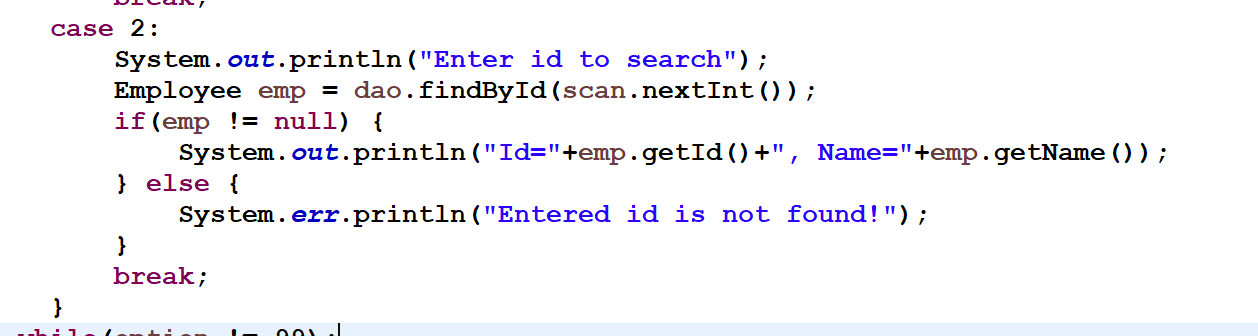
Controller Layer

for success : wraps employee in the response  
for exception : wraps error message in the response

findById() in the DAO layer



Main method calling findById()



How to generate sequence of numbers in oracle-db/mysql

In oracle we have a statement called create sequence seq\_name statement

In mysql we have an auto\_increment keyword

RowMapper: It is used to convert SQL Result to Java object, i.e., row mapping to java object

Server: These are programs which can host your applications, these servers runs with some port number, so that clients can use the host address & port number to access the applications running in the server

Where to keep HTML, CSS & JS files in dynamic web project

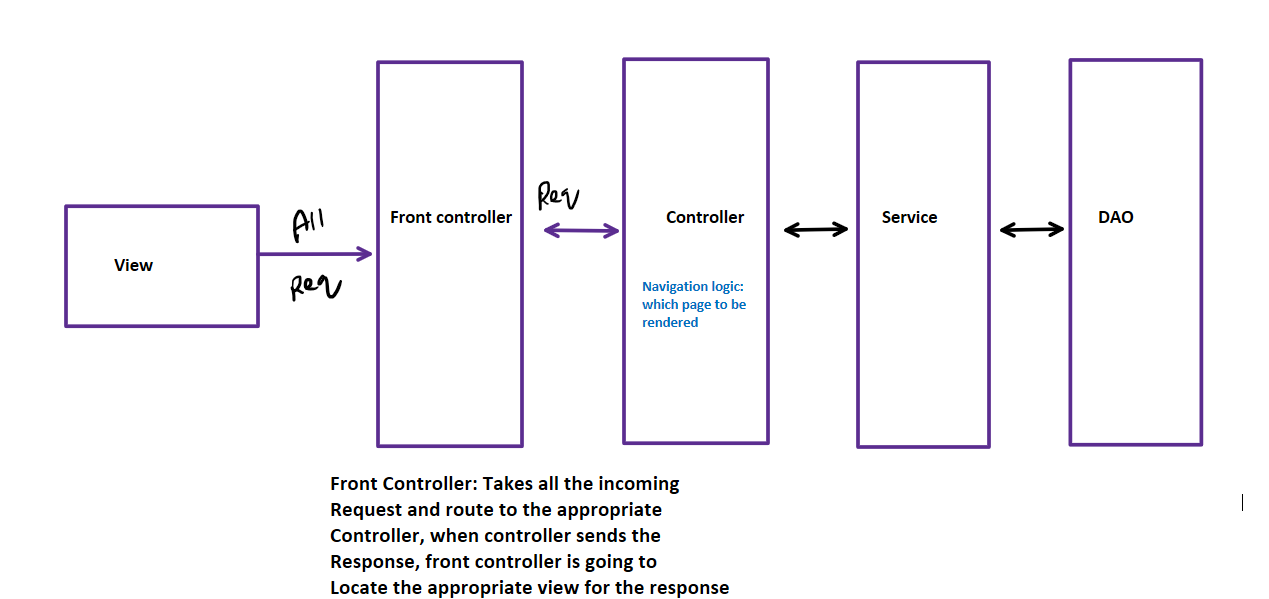
It must be inside WebContent folder in older eclipse, in newer eclipse it must be in src/main/webapp

web.xml: This is a deployment descriptor file, where you can have configurations related to welcome page, front controller, session timeouts and so on.

Spring Web MVC

This is a module in spring which allows you to develop web applications

Spring Web MVC architecture



What is View in Spring MVC: View can be HTML/JSP

JSP: Java Server Pages, which can display the dynamic data

Controller: It is not a single file, it is a method mapped with some URL

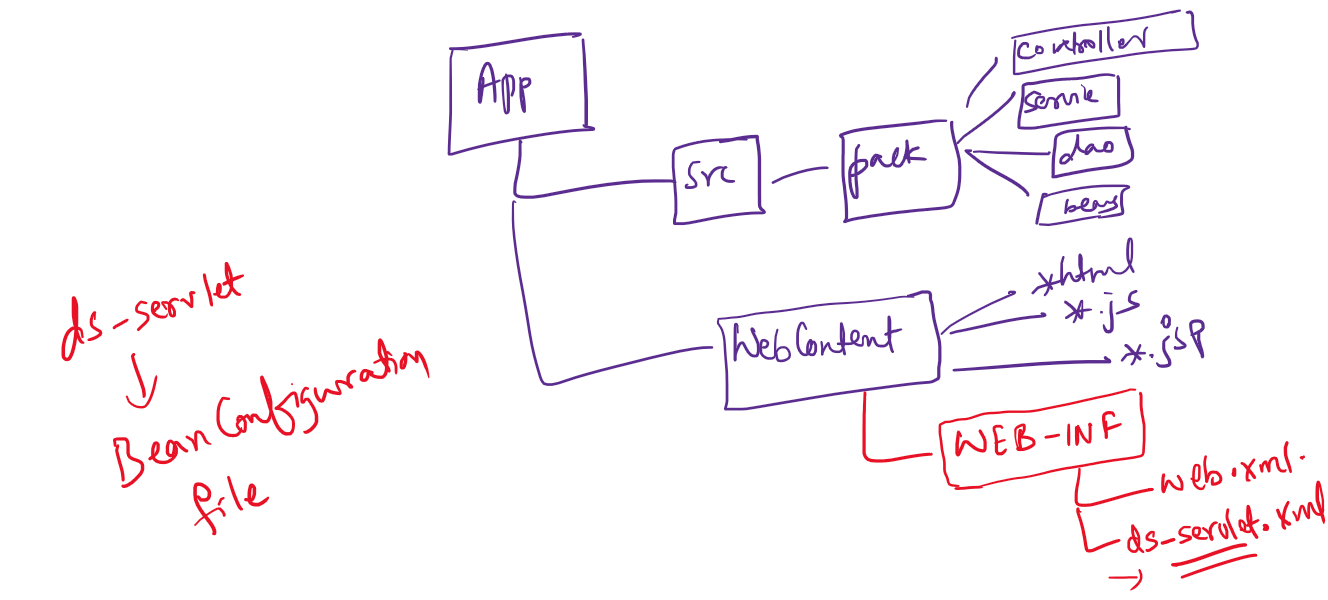
ex: /login, /register, /find, /update are all different requests, we need different controllers to handle these requests, hence we will have multiple methods mapped with these URL’s, so that they can handle those requests  
  
@Controller  
class AppController {   
 @Autowired  
 EmployeeService service;   
 @RequestMapping(“/login”)  
 login() { … }   
 @RequestMapping(“/register)  
 register() { .. }   
}

Front Controller:

We don’t’ have to write front controller code, spring gives us Front Controller(DispatcherServlet.class) which is inbuilt, it does following job:

1. Taking all the incoming requests from the user & routing to the appropriate controller
2. Front Controller takes care of initializing the spring container, by using the ApplicationContext ctx = new ClassPathXmlApplicationContext(getsFileNameFromWeb.xml);
3. web.xml provides the spring bean configuration file name, which front-controller uses to load

Spring Web Application Structure



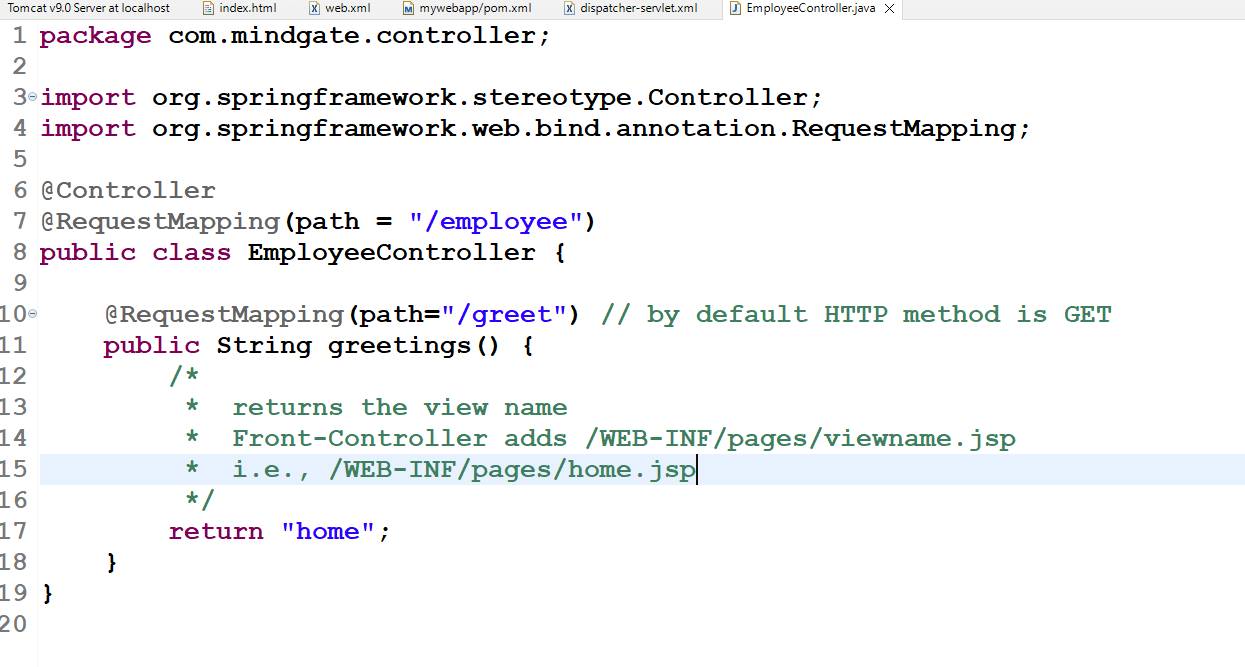
web.xml: This file will have spring beans configuration file name as well as the front controller class (DispatcherServlet) configuration

Steps

Libraries: spring-context, spring-webmvc, spring-jdbc

How to create a controller:   
You will create a class with @Controller  
@Controller  
@RequestMapping(path=”/employee”) : it can handle /spring/employee  
public class EmployeeController {   
 // spring/employee/greet  
 @RequestMapping(path=”/greet”)  
 public String greetings() { return “home”; } // WEB-INF/pages/home.jsp  
}

src/com/mindgate/controller/EmployeeController.java



What to do when tomcat gives error even when everything’s is good

Follow these steps

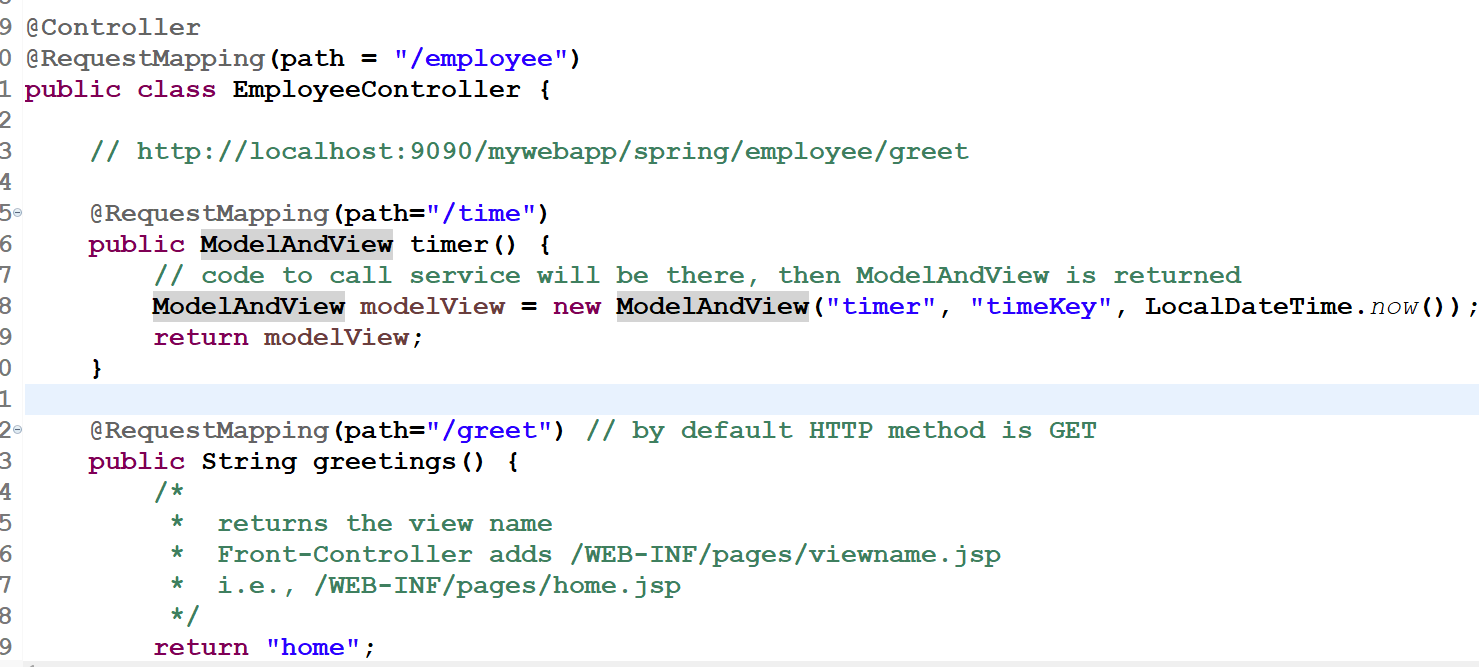
1. Stop the server
2. In Eclipse ensure the build path is referring to JDK else configure the project to refer to JDK folder instead of JRE folder in the build path
3. In Eclipse Go to Project & Select Clean
4. Right Click on Tomcat Server and perform below steps
   1. Clean & again right click and perform below option
   2. Clean Tomcat Work Directory
5. Right Click on Project >> Run As >> build… >> in the Goals type

clean package & select run

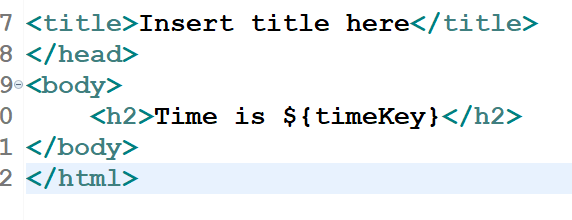
1. Right Click on Project >> Maven >> Update Project >> Select Force Update checkbox
2. Right click on project & run on server

ModelAndView object: It is a return type of the controller that can store view & model (data from the service) both  
Note: controller can return String, which will be only view, however ModelAndView object can maintain view & model

Controller code



timer.jsp



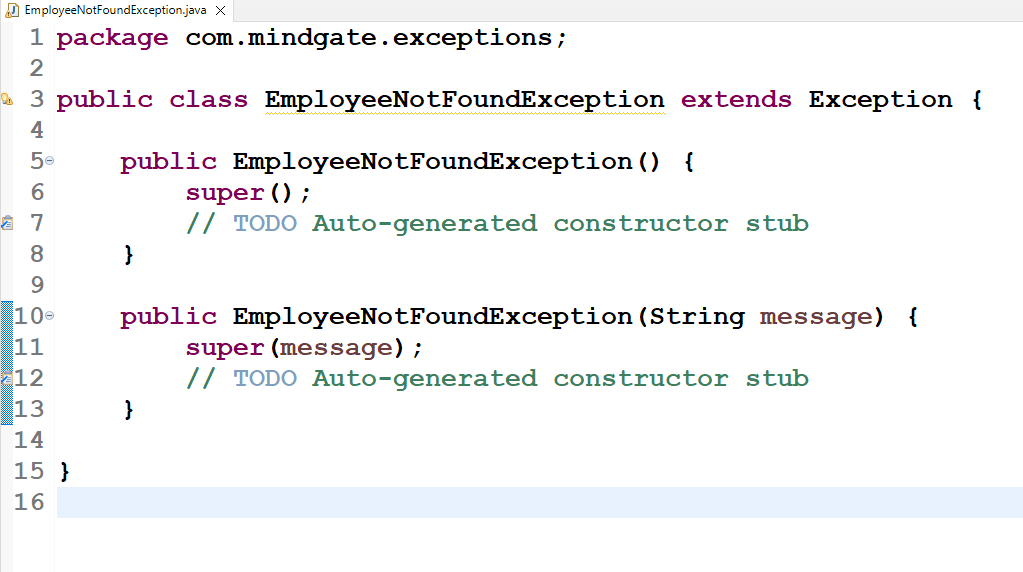
Form that can send data



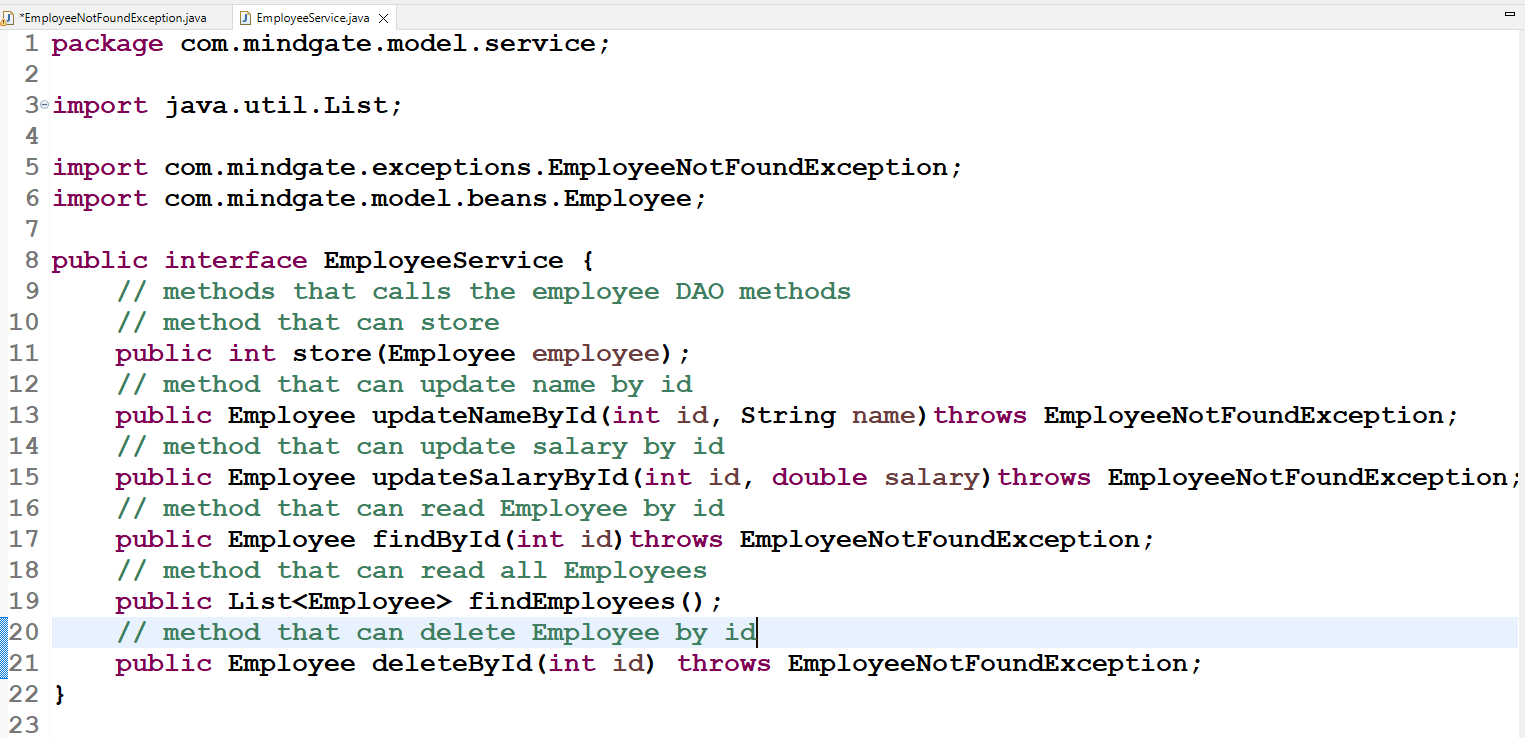
Things to create

1. dispatcher-servlet.xml >> beans for DMDS & JdbcTemplate
2. Employee.java
3. EmployeeRepository
4. EmployeeRepositoryJdbcImpl
5. EmployeeNotFoundException extends Exception
6. EmployeeService: interface with methods that
7. EmployeeServiceImplV1: can access EmployeeRepository methods
8. EmployeeController

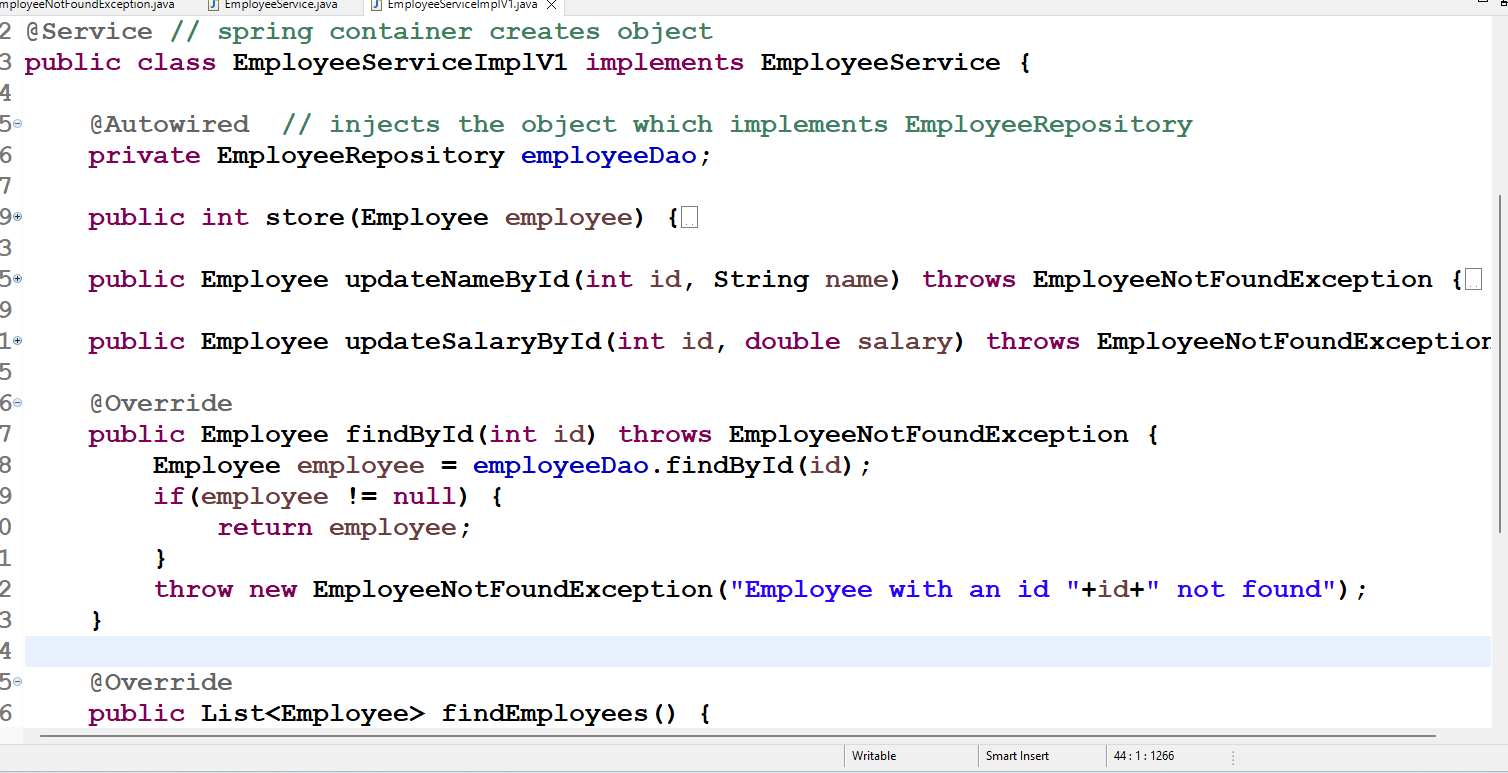
EmployeeNotFoundException



EmployeeService

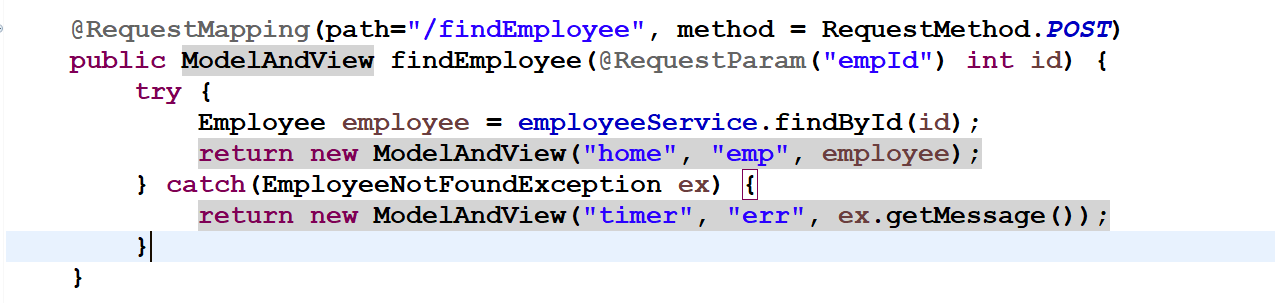


Implementing Service layer, using @Service on top of the class & @Autowired on EmployeeRepository

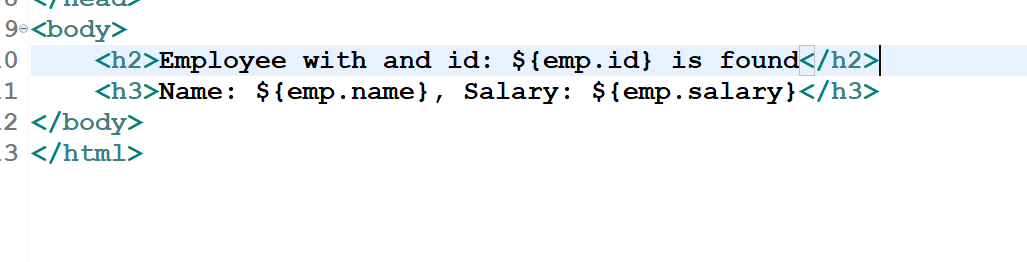


Controller will use service layer using @Autowired

EmployeeController



home.jsp



timer.jsp

