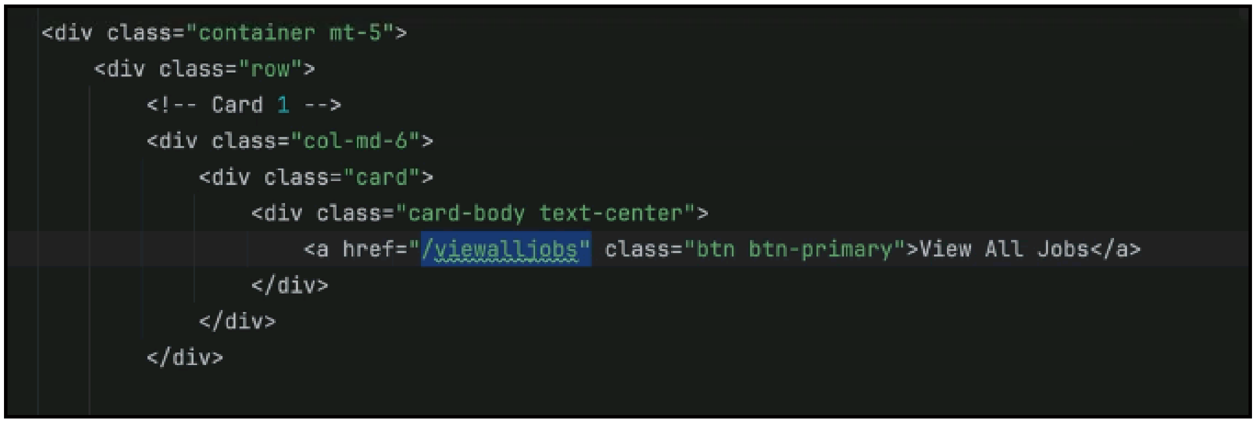
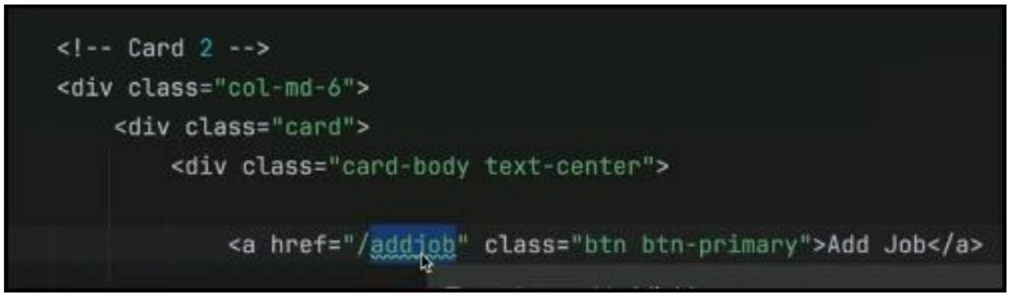


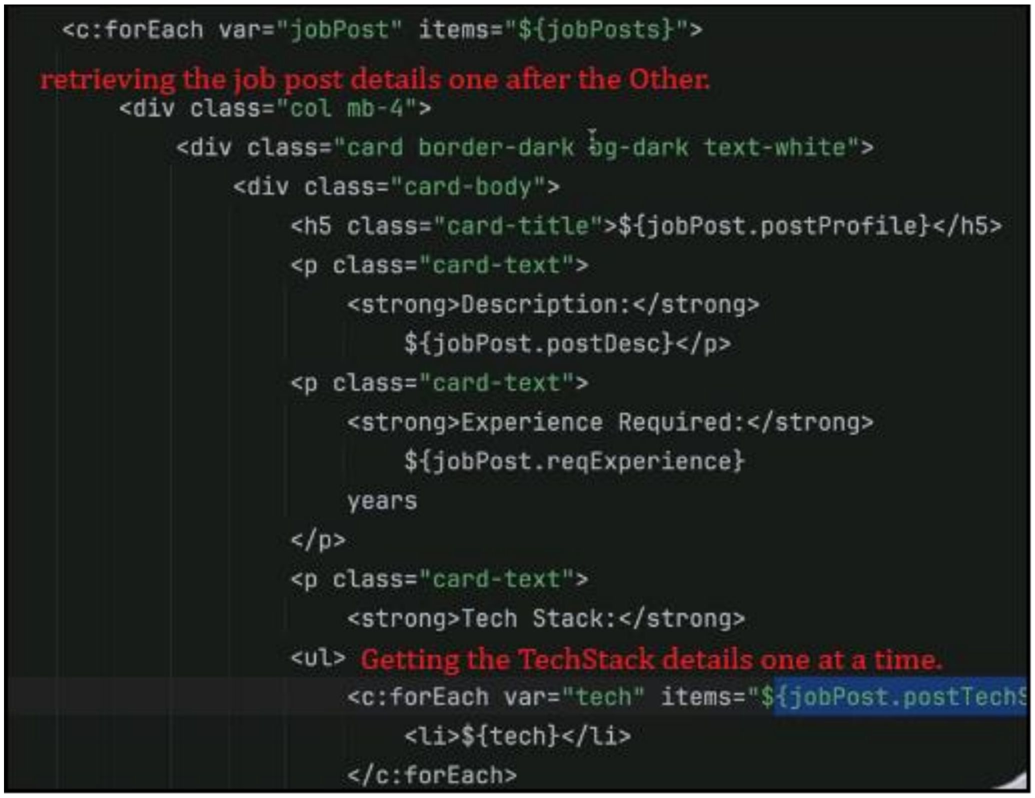
```
<div class="container mt-5">
    <div class="row">
        <!-- Card 1 -->
        <div class="col-md-6">
            <div class="card">
                <div class="card-body text-center">
                    <a href="/viewalljobs" class="btn btn-primary">View All Jobs</a>
                </div>
            </div>
        </div>
```

On Click of View All Jobs it should call /viewalljobs.



```
<!-- Card 2 -->
<div class="col-md-6">
    <div class="card">
        <div class="card-body text-center">

            <a href="/addjob" class="btn btn-primary">Add Job</a>
```

On click of /addjob it will re-direct to Add Job

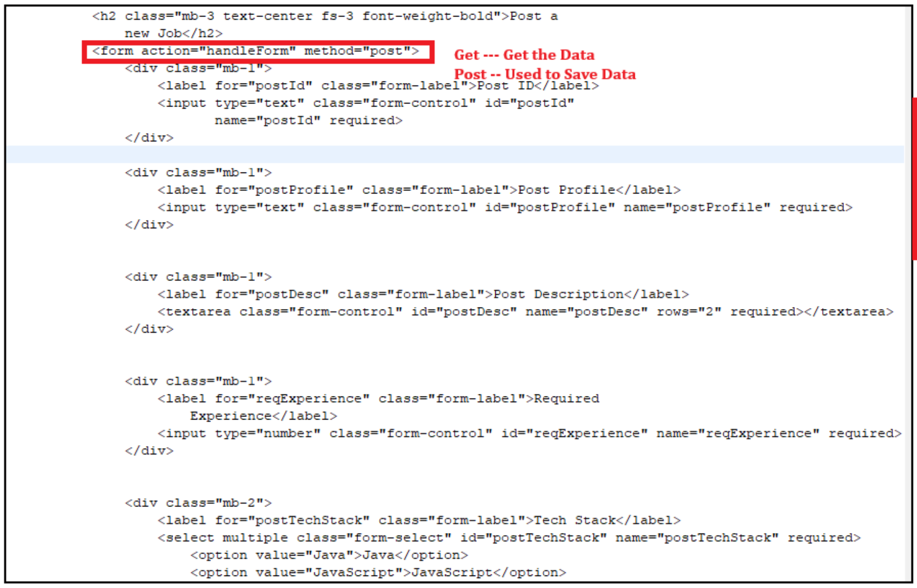


```
 <c:forEach var="jobPost" items="${jobPosts}">
retrieving the job post details one after the Other.
     <div class="col mb-4">
         <div class="card border-dark bg-dark text-white">
             <div class="card-body">
                 <h5 class="card-title">${jobPost.postProfile}</h5>
                 <p class="card-text">
                     <strong>Description:</strong>
                         ${jobPost.postDesc}</p>
                 <p class="card-text">
                     <strong>Experience Required:</strong>
                         ${jobPost.reqExperience}
                     years
                 </p>
                 <p class="card-text">
                     <strong>Tech Stack:</strong>
                 <ul> Getting the TechStack details one at a time.
                     <c:forEach var="tech" items="${jobPost.postTech
                         <li>${tech}</li>
                     </c:forEach>
```

Fetching each profile detail from the job Post

Get-get the details

Post: post the details

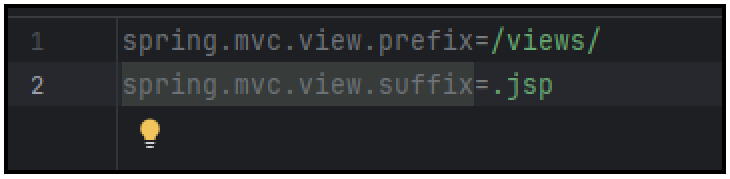


```
        <h2 class="mb-3 text-center fs-3 font-weight-bold">Post a
            new Job</h2>
        <form action="handleForm" method="post">
            <div class="mb-1">
                <label for="postId" class="form-label">Post ID</label>
                <input type="text" class="form-control" id="postId"
                       name="postId" required>
            </div>

            <div class="mb-1">
                <label for="postProfile" class="form-label">Post Profile</label>
                <input type="text" class="form-control" id="postProfile" name="postProfile" required>
            </div>


            <div class="mb-1">
                <label for="postDesc" class="form-label">Post Description</label>
                <textarea class="form-control" id="postDesc" name="postDesc" rows="2" required></textarea>
            </div>


            <div class="mb-1">
                <label for="reqExperience" class="form-label">Required
                    Experience</label>
                <input type="number" class="form-control" id="reqExperience" name="reqExperience" required>
            </div>


            <div class="mb-2">
                <label for="postTechStack" class="form-label">Tech Stack</label>
                <select multiple class="form-select" id="postTechStack" name="postTechStack" required>
                    <option value="Java">Java</option>
                    <option value="JavaScript">JavaScript</option>
```

Get --- Get the Data
Post -- Used to Save Data

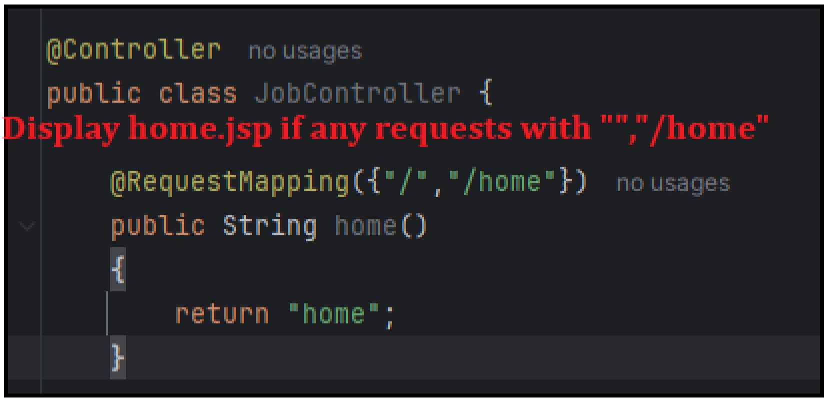
postId
postProfile
postDescription
Required Experience
TechStack

These all details will be provided in the Screen. when we all jobs we can view saved job details.



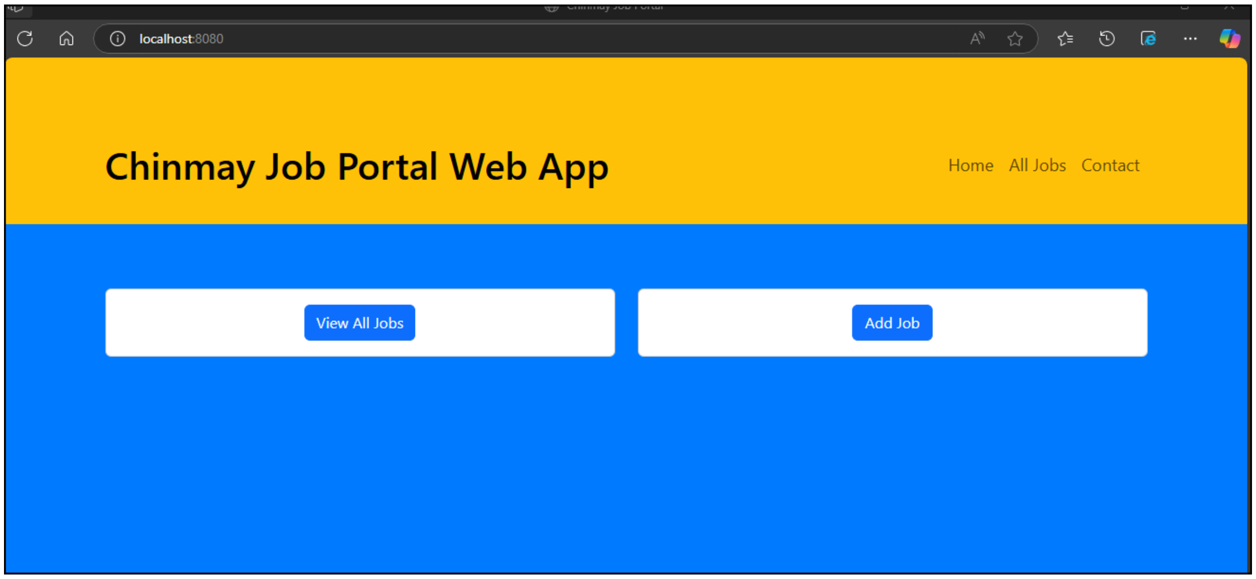
```
spring.mvc.view.prefix=/views/
spring.mvc.view.suffix=.jsp
```

Setting the properties in application.properties file for views

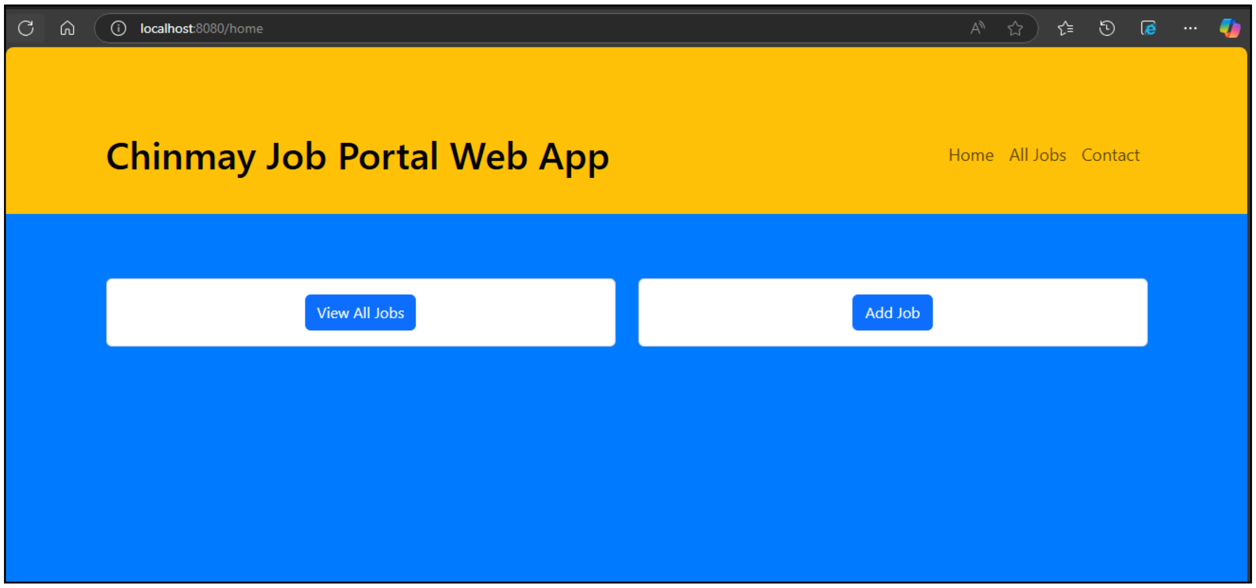


```
@Controller  no usages
public class JobController {

    @RequestMapping({"/","/home"})  no usages
    public String home()
    {
        return "home";
    }
```

Display home.jsp if any requests with "","/home"

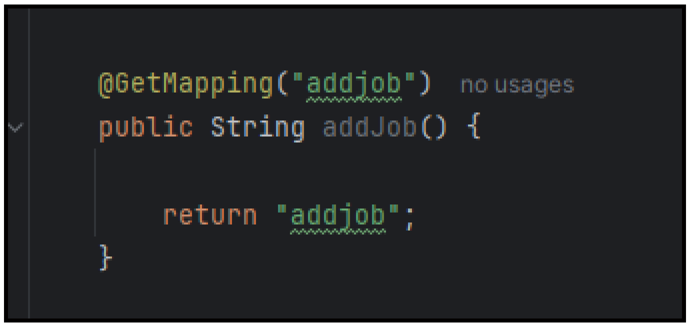


On click on localhost:8080/



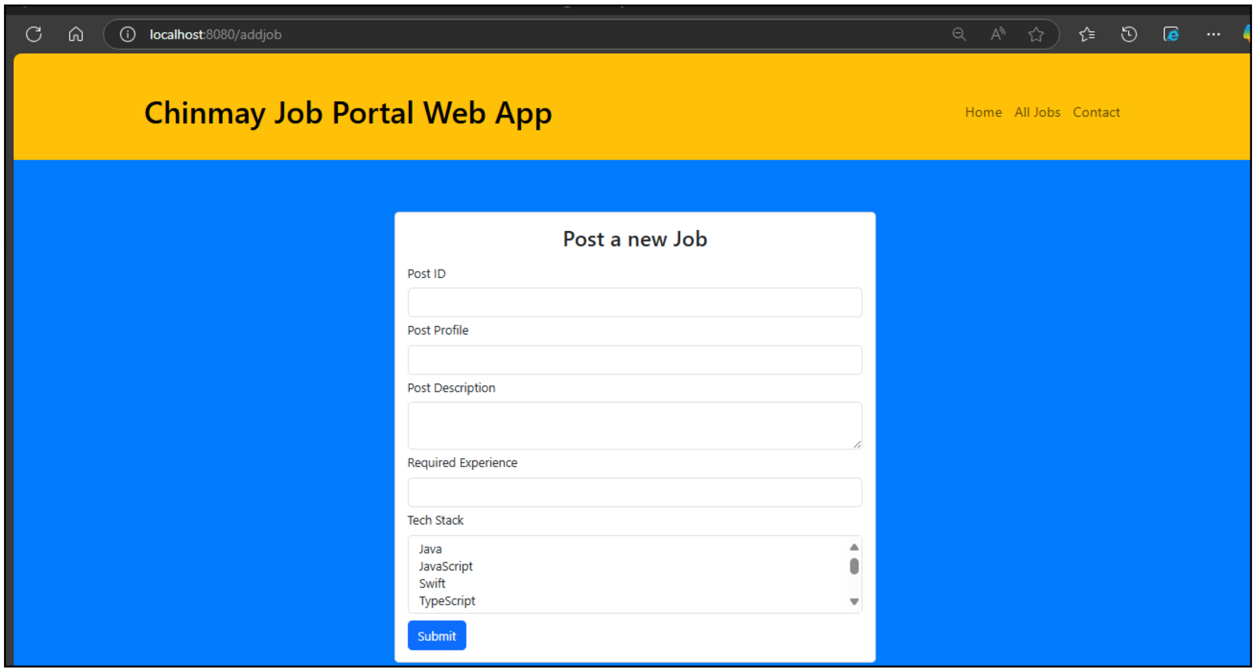
http://localhost:8080/home

For addjob



```
@GetMapping("addjob")  no usages
public String addJob() {

    return "addjob";
}
```

On click of addjob re-direct to addjob.jsp



localhost:8080/addjob

Chinmay Job Portal Web App

Home All Jobs Contact

Post a new Job

Post ID

Post Profile

Post Description

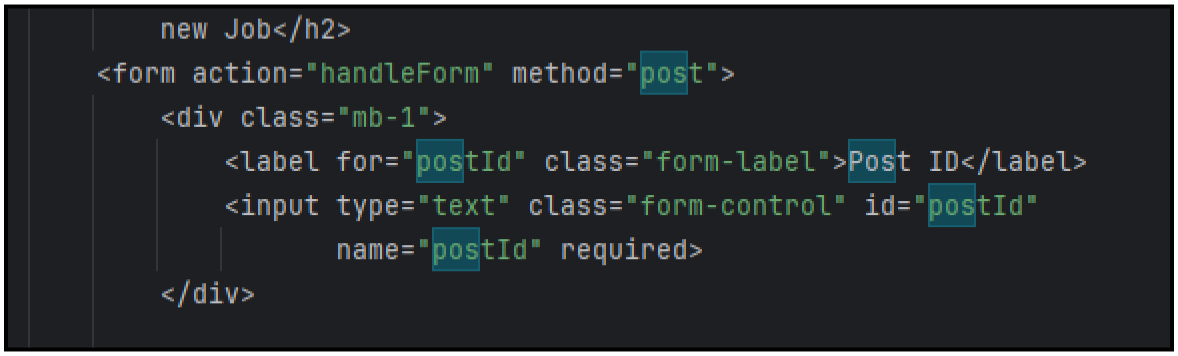
Required Experience

Tech Stack

Java
JavaScript
Swift
TypeScript

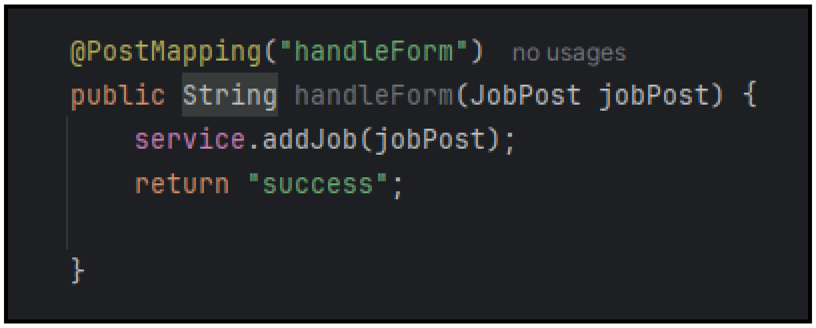
Submit

When we click on submit we are calling handleForm and now we need to mapping for handleForm with post Request.



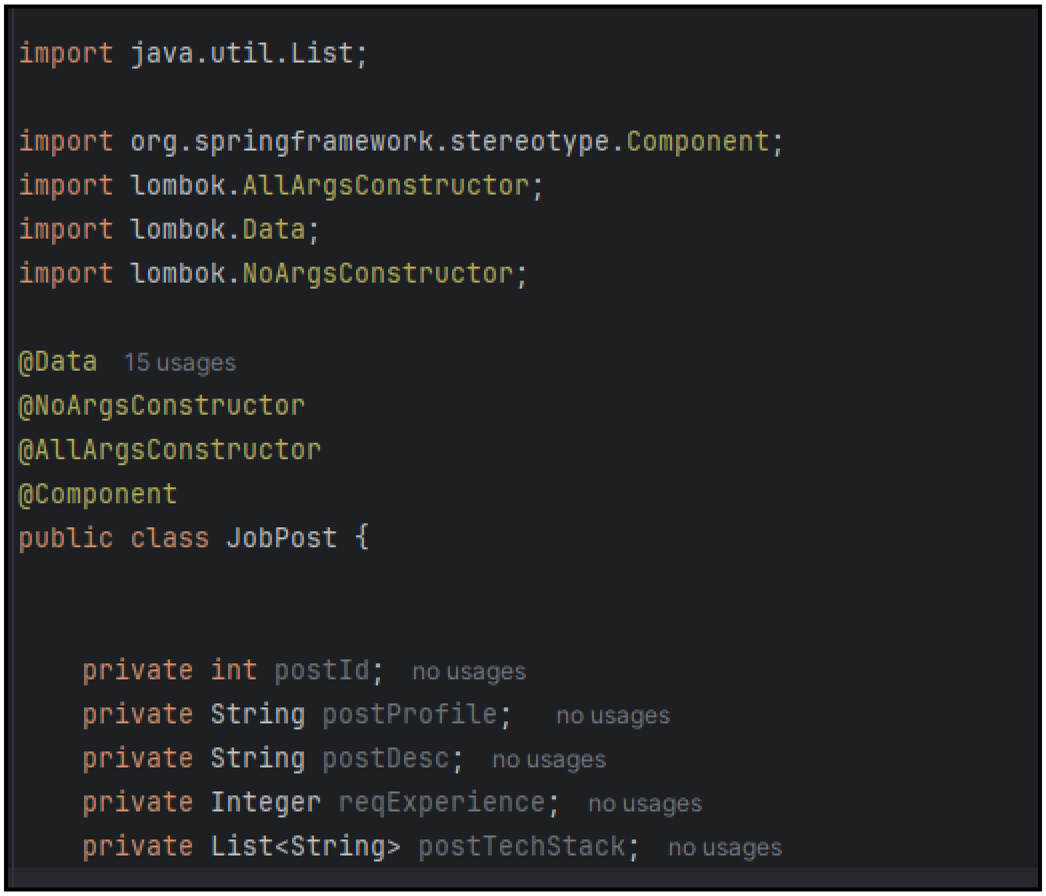
```
    new Job</h2>
<form action="handleForm" method="post">
    <div class="mb-1">
        <label for="postId" class="form-label">Post ID</label>
        <input type="text" class="form-control" id="postId"
               name="postId" required>
    </div>
```

225:Handling Form



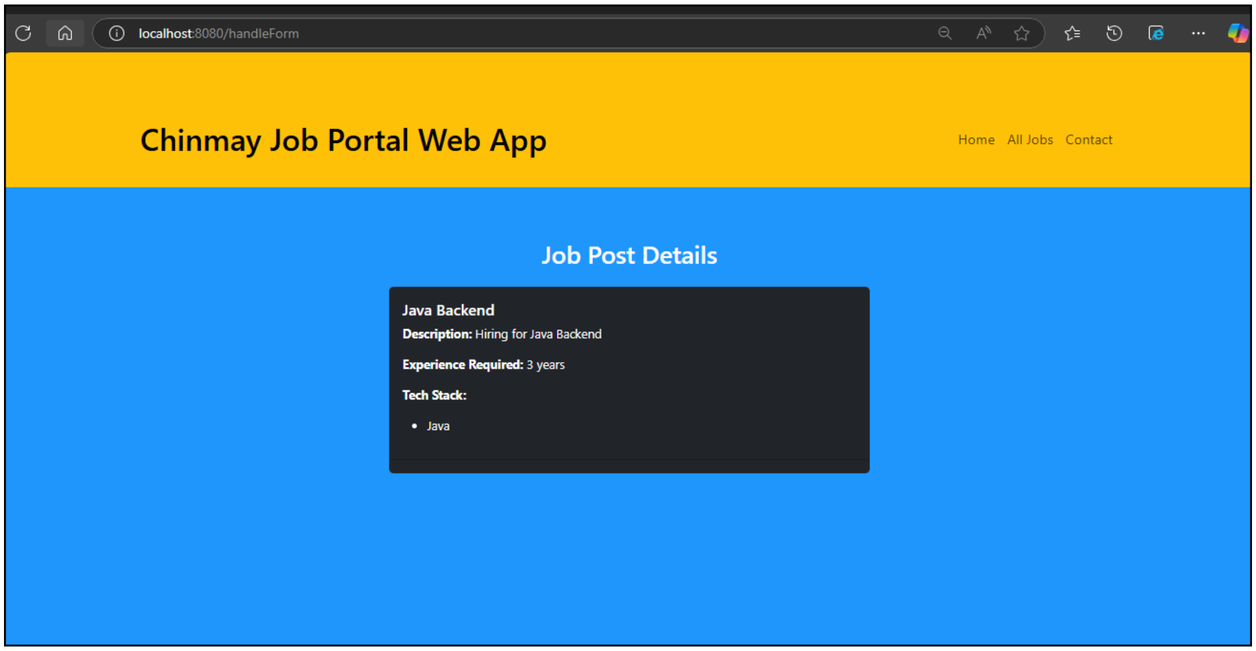
```
@PostMapping("handleForm")  no usages
public String handleForm(JobPost jobPost) {
    service.addJob(jobPost);
    return "success";

}
```

What ever the data that we are giving in the form we are accepting into JobPost Model Object



```
import java.util.List;

import org.springframework.stereotype.Component;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data  15 usages
@NoArgsConstructor
@AllArgsConstructor
@Component
public class JobPost {


    private int postId;  no usages
    private String postProfile;  no usages
    private String postDesc;  no usages
    private Integer reqExperience;  no usages
    private List<String> postTechStack;  no usages
```

Once the data is sent, we will store it in the JobPost and same will be displayed onto the success.jsp page.



Job Post Details

Java Backend

Description: Hiring for Java Backend

Experience Required: 3 years

Tech Stack:

- Java

Once we click on submit the above page will be displayed.

As of now we are now not storing this data anywhere now we will try to store this data so that when we click on AllJobs we will get the saved Job details .

226: Working with Layers

Controller Layer will not be responsible for Processing and storing the data.so it will try to interact with the service Layer.

JobServcie will have two methods

```
-- addJob
// method to add a jobPost
public void addJob(JobPost jobPost) {
    repo.addJob(jobPost);


}
```

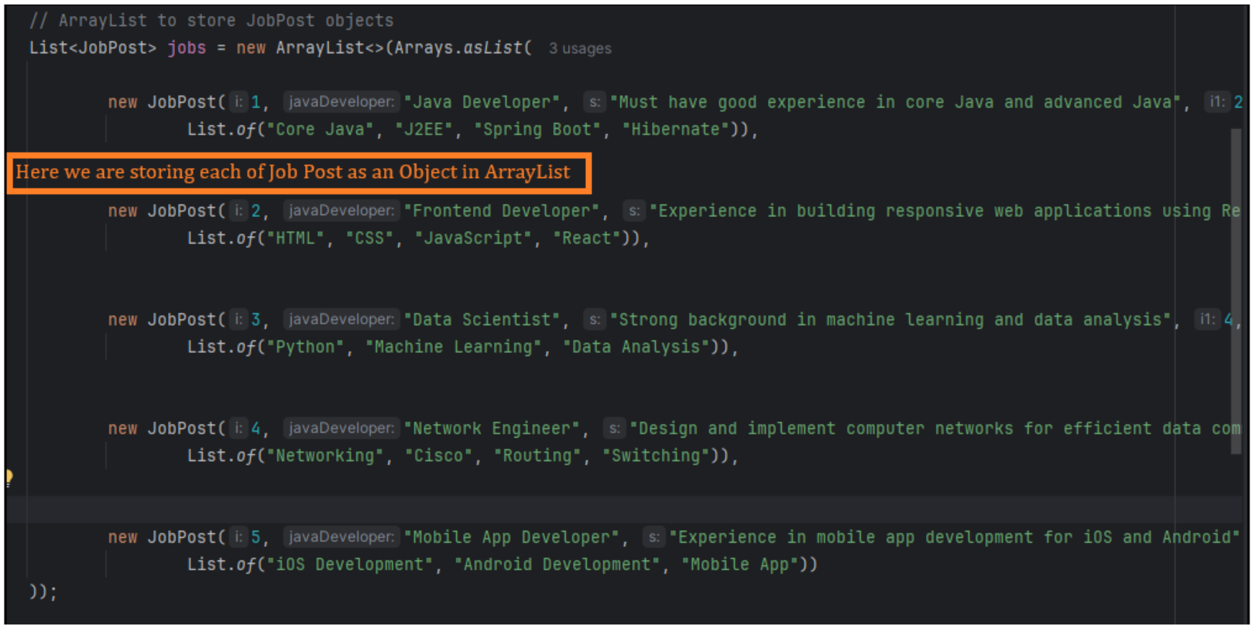
```
-- getAllJobs
//method to return all JobPosts
public List<JobPost> getAllJobs() {
```

```java
    return repo.getAllJobs();
}
```

From the Servcie it will be re-directed to repository layer to addJob and get the Job details

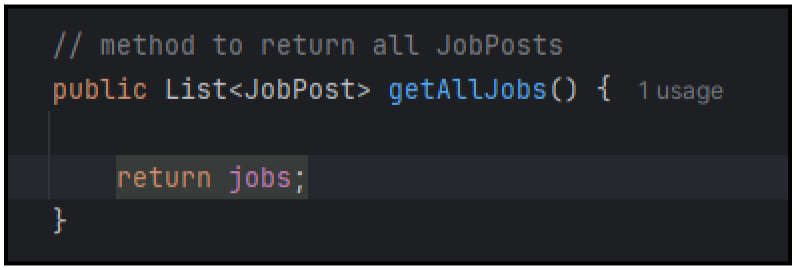
```java
// method to save a job post object into arrayList
public void addJob(JobPost job) {
    jobs.add(job);
    System.out.println(jobs);
}
```

What ever the job details that which we add will be stored in ArrayList as an Object.

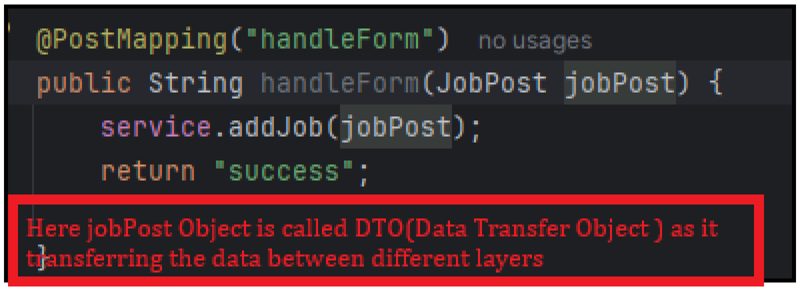


```java
// ArrayList to store JobPost objects
List<JobPost> jobs = new ArrayList<>(Arrays.asList(  3 usages

        new JobPost( i: 1,  javaDeveloper: "Java Developer",  s: "Must have good experience in core Java and advanced Java",  i1: 2
                List.of("Core Java", "J2EE", "Spring Boot", "Hibernate")),
Here we are storing each of Job Post as an Object in ArrayList
        new JobPost( i: 2,  javaDeveloper: "Frontend Developer",  s: "Experience in building responsive web applications using Re
                List.of("HTML", "CSS", "JavaScript", "React")),


        new JobPost( i: 3,  javaDeveloper: "Data Scientist",  s: "Strong background in machine learning and data analysis",  i1: 4,
                List.of("Python", "Machine Learning", "Data Analysis")),


        new JobPost( i: 4,  javaDeveloper: "Network Engineer",  s: "Design and implement computer networks for efficient data com
                List.of("Networking", "Cisco", "Routing", "Switching")),


        new JobPost( i: 5,  javaDeveloper: "Mobile App Developer",  s: "Experience in mobile app development for iOS and Android"
                List.of("iOS Development", "Android Development", "Mobile App"))
));
```

Once we click on allJobs we are getting the Jobdetails stored in the ArrayList

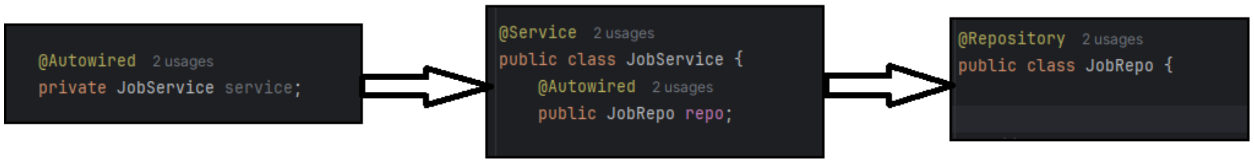


```java
// method to return all JobPosts
public List<JobPost> getAllJobs() {  1 usage

    return jobs;
}
```



```java
@PostMapping("handleForm")  no usages
public String handleForm(JobPost jobPost) {
    service.addJob(jobPost);
    return "success";
}
```

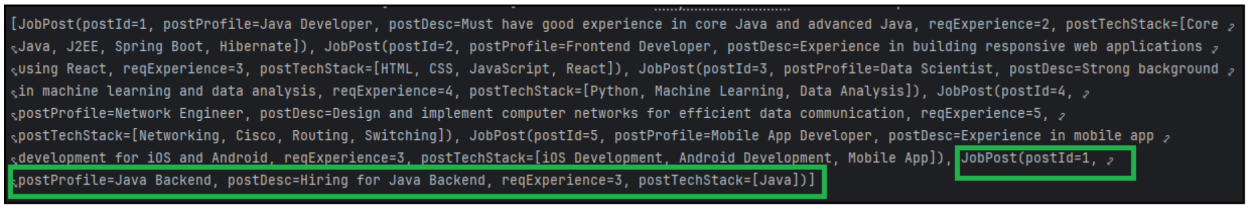
Here jobPost Object is called DTO(Data Transfer Object) as it transferring the data between different layers



```java
@Autowired  2 usages
private JobService service;
```

```java
@Service  2 usages
public class JobService {
    @Autowired  2 usages
    public JobRepo repo;
```

```java
@Repository  2 usages
public class JobRepo {
```

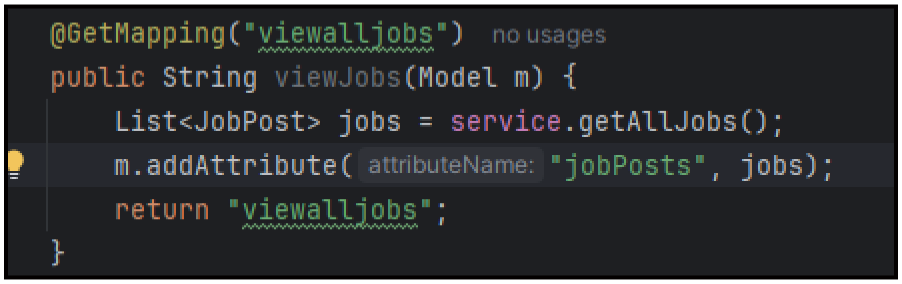


```
[JobPost(postId=1, postProfile=Java Developer, postDesc=Must have good experience in core Java and advanced Java, reqExperience=2, postTechStack=[Core
Java, J2EE, Spring Boot, Hibernate]), JobPost(postId=2, postProfile=Frontend Developer, postDesc=Experience in building responsive web applications
using React, reqExperience=3, postTechStack=[HTML, CSS, JavaScript, React]), JobPost(postId=3, postProfile=Data Scientist, postDesc=Strong background
in machine learning and data analysis, reqExperience=4, postTechStack=[Python, Machine Learning, Data Analysis]), JobPost(postId=4,
postProfile=Network Engineer, postDesc=Design and implement computer networks for efficient data communication, reqExperience=5,
postTechStack=[Networking, Cisco, Routing, Switching]), JobPost(postId=5, postProfile=Mobile App Developer, postDesc=Experience in mobile app
development for iOS and Android, reqExperience=3, postTechStack=[iOS Development, Android Development, Mobile App]), JobPost(postId=1,
postProfile=Java Backend, postDesc=Hiring for Java Backend, reqExperience=3, postTechStack=[Java])]
```

Above we could see we were able to view the newly added Job Details.

227:View Data:

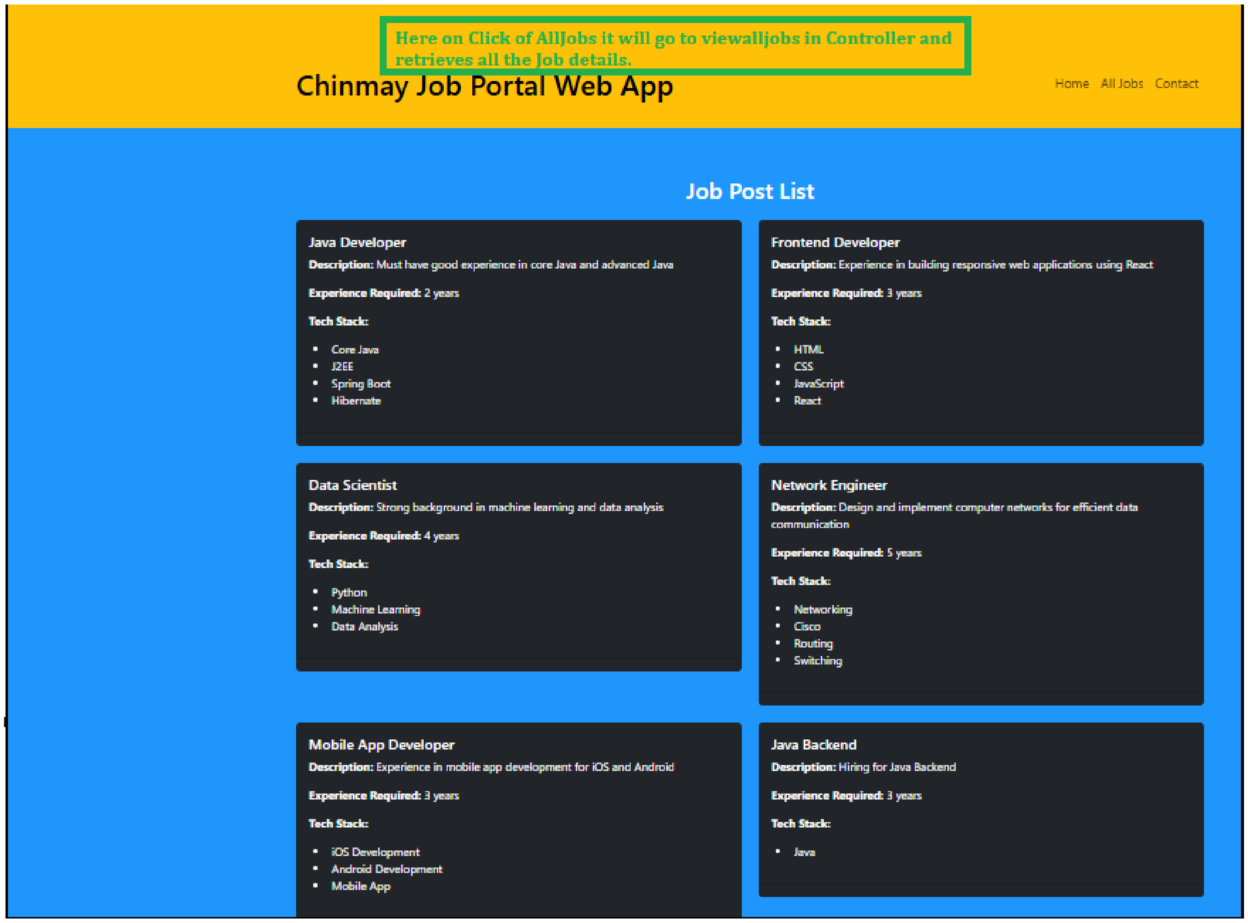
On click of viewalljobs we will be able to view the AllJob details on the Screen



```java
@GetMapping("viewalljobs")  no usages
public String viewJobs(Model m) {
    List<JobPost> jobs = service.getAllJobs();
    m.addAttribute( attributeName: "jobPosts", jobs);
    return "viewalljobs";
}
```

When we click on viewalljobs we are retrieving the data from the Repo and adding it to the model Attribute

We were adding details to Model interface addAttribute method



Here on Click of AllJobs it will go to viewalljobs in Controller and retrieves all the Job details.

Chinmay Job Portal Web App

Job Post List

Java Developer

Description: Must have good experience in core Java and advanced Java

Experience Required: 2 years

Tech Stack:

- Core Java
- J2EE
- Spring Boot
- Hibernate

Frontend Developer

Description: Experience in building responsive web applications using React

Experience Required: 3 years

Tech Stack:

- HTML
- CSS
- JavaScript
- React

Data Scientist

Description: Strong background in machine learning and data analysis

Experience Required: 4 years

Tech Stack:

- Python
- Machine Learning
- Data Analysis

Network Engineer

Description: Design and implement computer networks for efficient data communication

Experience Required: 5 years

Tech Stack:

- Networking
- Cisco
- Routing
- Switching

Mobile App Developer

Description: Experience in mobile app development for iOS and Android

Experience Required: 3 years

Tech Stack:

- iOS Development
- Android Development
- Mobile App

Java Backend

Description: Hiring for Java Backend

Experience Required: 3 years

Tech Stack:

- Java

7 JobApp-Project.zip