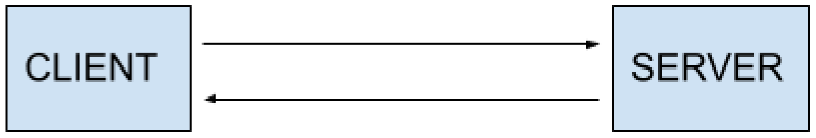
Spring MVC Architecture and applications

It is a spring module used to develop web applications using spring. MVC stands for Model View Controller design pattern. It is a way to organize the code of our application. It needs servlet api to perform operations.

- ❖ Model – data
- ❖ View – present data
- ❖ Controller – control data

It is a client-server architecture where the client sends over the request to the server and the server responds back with the response in the form of web pages.



- ➔ Dispatcher Servlet-- Front controller accepts request from client and transfers that request to controller.
- ➔ View resolver- jsp file which assigns view to a particular file which controller wants the model(data) to be viewed.
- ➔ Controller- class which handles the request assigned by Dispatcher Servlet.
- ➔ View- response which will be displayed to the client.

Add configurations to path: WEB-INF/web.xml

- Configure Spring MVC Dispatcher Servlet
- Set URL mappings to Spring MVC Dispatcher Servlet

Add configurations to path: WEB-INF/Spring MVC Dispatcher Servlet.xml

- Add support for Spring component scanning
- Add support for conversion, formatting and validation
- Configure Spring MVC View Resolver

View Resolver Configs

```
        <bean
                class="org.springframework.web.servlet.view.InternalResourceViewResolver">
                <property name="Prefix" value="/WEB-INF/view/" />
                <property name="Suffix" value=".jsp" />
        </bean>
```

Spring MVC Architecture and applications

Spring MVC Form Handling application

Development Process
Create a Dynamic web project or import a demo project from git.
Add MVC Starter Files in src/main/webapp/WEB-INF path.
Create a package for controller classes and one for entity classes.
Create controller class in package define methods and return views
Create views and do required coding there.
run the project

Add spring jar files to get access to spring annotations imports and properties.
Create a controller method to show initial HTML form, map it with Http using @Requestmapping annotation and return a .jsp view page. Then create a controller method to process initial HTML form, map it with Http using @Requestmapping and return .jsp view page.

Adding data to model
Model attribute is a bean which holds form data for the data binding.
Model is used to pass the data between controllers and views

In the controller, create a method and in parameter add HttpServletRequest request, Model theModel. Read request parameters from HTML form, do a particular operation, create the message, add a message to model, return "jsp file name".

Binding Request
Annotation used is @RequestParam. In this concept, spring will read data from @RequestParam and bind the data to the variable.
Remove HttpServletRequest request from parameter and replace it with @RequestParam(" pass http server request object"). In this we don't need to read request parameters from the HTML form in the method body as we have declared in the parameters.

Ambiguous Mapping
When there is the same HTTP mapping on different controllers then this exception occurs.
To resolve this issue setup a parent mapping at the top of the controller class i.e, @RequestMapping("mapping name") such that the whole controller will follow the path we have given in request mapping.

Form tags in spring
Spring MVC form tags can make use of data binding by automatically setting/retrieving data from a Java object/bean. Here are some of the form tags in MVC: -

- form:form
- form:input
- form:textarea
- form:checkbox
- form:radiobutton
- form:select

Spring MVC Architecture and applications

Use at top of the jsp page: -
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>

We have to add a model attribute before adding the form tag. In spring we use a model object to add attributes where we give attribute name and the value. The attribute name will be the same name that form will use to reference the model attribute.
eg**- theModel.addAttribute("visitor", new Visitor())** where theModel is a model object, "visitor" is the entity class object we are creating in the same property.
@ModelAttribute binds form data to object.
eg**- (@ModelAttribute("visitor") Visitor theVisitor)**

Form tag- text fields

Create a visitor POJO class with getter, setter, toString, null constructor, constructor using fields
Create a controller with @Controller and RequestMapping("visitor")
Create a method with @Requestmapping("/showForm")
Get the object of customer and ad customer object to the model.
Add Model to the controller method with adding attributes of customer.
Return .jsp view page

Create a .jsp view page and create a form there using mvc form tags, give form action and path of modelAttribute as visitor.

Note: Path to be given in form path will be the model class field name we want to process.

Create a controller method of the same mapping that we gave in form action.
in the method parameter add (@ModelAttribute("visitor") Visitor theVisitor).

Note: Name inside @ModelAttribute should be the same that we gave to ModelAttribute while creating the form.
Return the .jsp view page
Create .jsp view page and in that use ${visitor.firstName}${visitor.lastName}
Note: We have "visitor" as a model attribute object and the "fields" of our model class we want to process.

Drop Down List
We use <form:select> and <form:option> for drop down. It calls getter methods from POJO.
Add drop down code to the form page.
Create a field for the country and generate getters and setters.
Update form page with ${visitor.city}.

Spring MVC Architecture and applications

If we don't want to hardcode the values in the form and want the values from the java then create a field of countryOptions as LinkedHashMap<String, String>. Populate cityOptions field with key and values like <"MUM", "Mumbai"> then go to form and use <form:options items="${visitor.city} and remove <form:option> and its values and add country in processing .jsp page

Radio button
Add this in the form jsp <form:radiobutton path="" value="">. It calls the setter method of POJO class and add modelAttribute name accessing the setter method of a particular field of POJO class in the ${} in process view page.

Checkbox
Add this in the .jsp form file<form:checkbox path="" value="">. Create a field of String Array in pojo class as it can take multiple values. To access jstl tags use
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %> in the view page of process data.

Use <c:forEach car="temp" items="${model object.pojo field}"> to loop the string values of the checkbox.

Form validation
use annotations like @NotNull, @Min, @Max, @Size, @Pattern (to must match regular expression pattern), @Future/@Past (to display past or future dates from present date).

Add hibernate validation jar files to the lib folder.

Development Process

- Add validation rule to Model class
  add annotations from above at top of fields
  @NotNull(message="is required")
  @Size(message="is required", min=1)
- Display error messages on HTML form
  Add <form:errors path="" cssClass="error" />
  error is css style so declare that in <style> tag
- Perform validation in controller class
  Create a controller method with RequestMapping, Model, addAttribute and return .jsp page.
- Update process view .jsp page

Spring MVC Architecture and applications

Use **@Valid** and **BindingResult** to the controller method parameter
tell spring to validate the object which is passed in and bind that object to store the values.
@Valid - performs validation rules on entity class object
BindingResult - results of validation are placed in BindingResult object

To restrict white spaces to your form use **@InitBinder**. This annotation works as a preprocessor means it will process before any other annotation.

Add below method to our controller to perform whitespace restrict-

```
    @InitBinder
    public void initBinder(WebDataBinder dataBinder) {

        StringTrimmerEditor stringTrimmerEditor = new StringTrimmerEditor(true);
        dataBinder.registerCustomEditor(String.class, stringTrimmerEditor);
    }
```

To restrict **Number Range** use below annotations-

```
@NotNull("is required")
@Min(value=0, message="must be greater than or equal to zero")
@Max(value=10, message="must be less than or equal to ten")
```

Custom error message
Create a resources folder in that create a message.properties file and write below code **typeMismatch.customer.freePasses=Invalid number** where typeMismatch is Error type, customer is spring model attribute name, freePasses is field name and Invalid number is our custom error message.

Custom Validation
We can also create our own custom java annotation. Let's create @ConCode for example.

```
@ConCode(value="IN", message="must start with IN")
private String ConCode;
```

Create ConCodeConstraintValidator as a helper class which contains our custom business logic for validation.

Spring MVC Architecture and applications

Create an annotation for instance ConCode and write code similar to below

```
@Constraint(validatedBy = ConCodeConstraintValidator.class)
@Target({ElementType.METHOD,ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface ConCode {

    //define default country code
    public String value() default "IN";

    //define default error message
    public String message() default "must start with IN";

    //define default groups
    public Class<?>[] groups() default{};

    //define default payloads
    public Class<? extends Payload>[] payload() default{};
}
```

Create a helper class ConCodeConstraintValidator implements ConstraintValidator<>

```
public class ConCodeConstraintValidator
        implements ConstraintValidator<ConCode, String>{
    String conPrefix;

    @Override
    public void initialize(ConCode theConCode) {

    //define prefix
    conPrefix = theConCode.value();

    }

@Override
public boolean isValid(String
theCode,ConstraintValidatorContexttheConstraintValidatorContext) {

    boolean result = theCode.startsWith(conPrefix);
    return result;
}
```

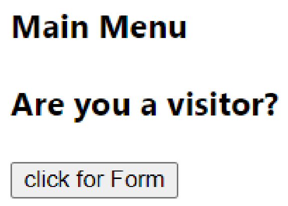
Add a field in entity class and write @ConCode before it, constructor will be untouched, add property in .jsp form file, add property in .jsp file to process data.

Spring MVC Architecture and applications

Download Project from github:
https://github.com/Leaf-Co-Kb/Job-Finder-SpringMVC.git

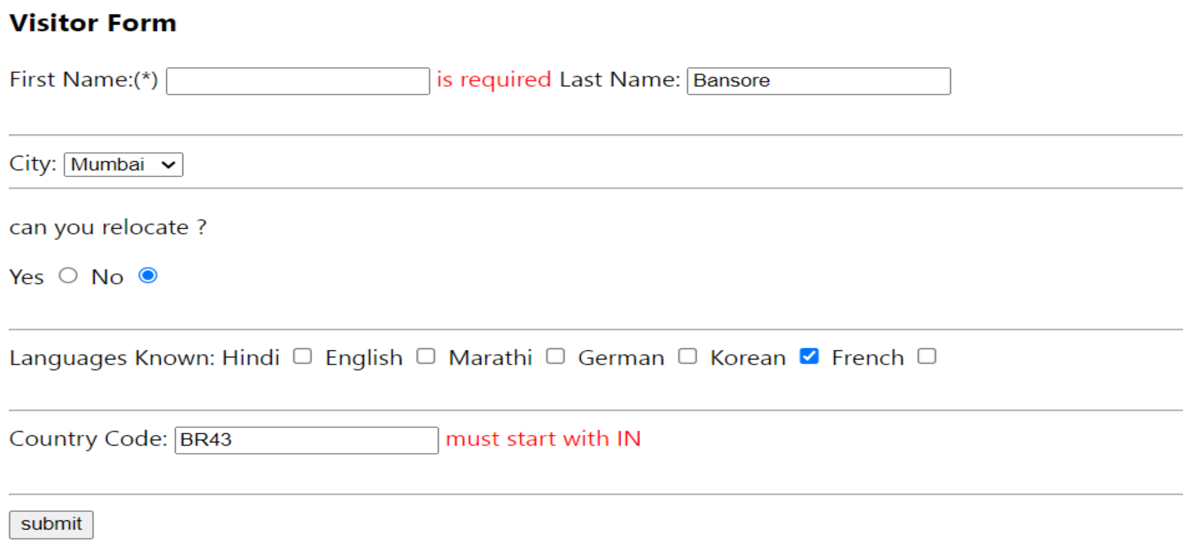
Screenshots of Server Response:

1. Below is the main menu page


Main Menu

Are you a visitor?

2. Below is the main form visitor will see when he click on the link:


Visitor Form

First Name:(*) [ ] is required Last Name: Bansore

City: Mumbai

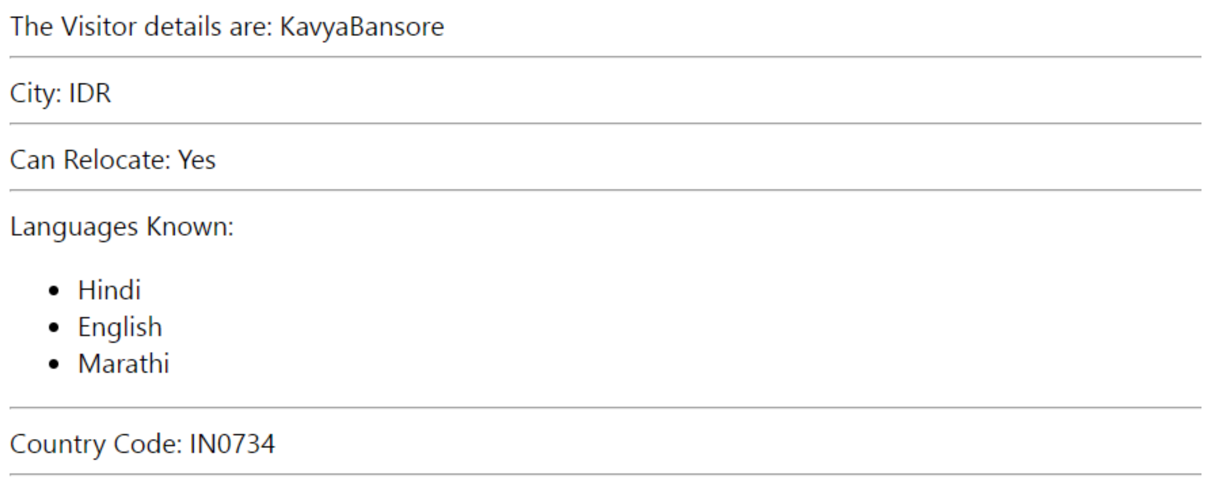
can you relocate ?

Yes ○ [✓ No]

Languages Known: Hindi ☐ English ☐ Marathi ☐ German ☐ Korean [✓] French ☐

Country Code: BR43 must start with IN

3. Once a visitor clicks on the submit button with required fields, the below page is displayed with entered information.


The Visitor details are: KavyaBansore

City: IDR

Can Relocate: Yes

Languages Known:

- Hindi
- English
- Marathi

Country Code: IN0734