**Spring Validation**

Spring Validation is an important feature of the Spring Framework that helps ensure the correctness and integrity of data. It can be used to validate input data in web forms, data models, or request parameters. Spring provides both declarative and programmatic ways of performing validation.

**Types of Validation in Spring:**

1. Spring's Built-in Validation (JSR-303/JSR-380 - Bean Validation API): Spring supports Bean Validation via annotations, typically using the Hibernate Validator as the default implementation.
2. Custom Validation: You can create custom validators by implementing the ConstraintValidator interface and providing your own validation logic.
3. Programmatic Validation: This is done using the Validator interface provided by Spring.

**Add the following dependency:-**

***Spring Web**

***Spring Boot DevTools**

***Thymeleaf**

***Validation**

**1. DemoApplication.java**

```
package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

   public static void main(String[] args) {

      SpringApplication.run(DemoApplication.class, args);

   }

}
```

**2. PersonForm.java**

```
package com.example.demo;
```

```
import jakarta.validation.constraints.*;
public class PersonForm {
    @NotNull
    @Size(min = 3, max = 30, message = "Name must be between 3 and 30 characters")
    private String name;
    @NotNull(message = "Age cannot be null")
    @Min(value = 18, message = "Age must be at least 18")
    @Max(value = 100, message = "Age must be less than or equal to 100")
    private Integer age;
    @NotNull(message = "Phone cannot be null")
    @Pattern(regexp = "^\\d{10}$", message = "Phone number must be 10 digits")
    private String phone;
    @NotBlank(message = "Email cannot be empty")
    @Email(message = "Enter a valid email address")
    private String email;
    // Getters and Setters
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
```

```
    }
    public String getPhone() {
        return phone;
    }
    public void setPhone(String phone) {
        this.phone = phone;
    }
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
    @Override
    public String toString() {
        return "PersonForm [name=" + name + ", age=" + age + ", phone=" + phone
+ ", email=" + email + "]";
    }
}
```

**3. PersonFormController.java**

```
package com.example.demo;
import com.example.demo.PersonForm;
import jakarta.validation.Valid;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.GetMapping;
```

```
import org.springframework.web.bind.annotation.PostMapping;
@Controller
public class PersonFormController {
    @GetMapping("/personForm")
    public String showForm(Model model) {
        model.addAttribute("personForm", new PersonForm());
        return "personForm";  // View name (personForm.html)
    }
    @PostMapping("/personForm")
    public String submitForm(@Valid PersonForm personForm, BindingResult result, Model model) {
        if (result.hasErrors()) {
            return "personForm";  // Return the form view with validation errors
        }
        model.addAttribute("person", personForm);
        return "formSuccess";  // View for success (formSuccess.html)
    }
}
```

**4.formSuccess.html**

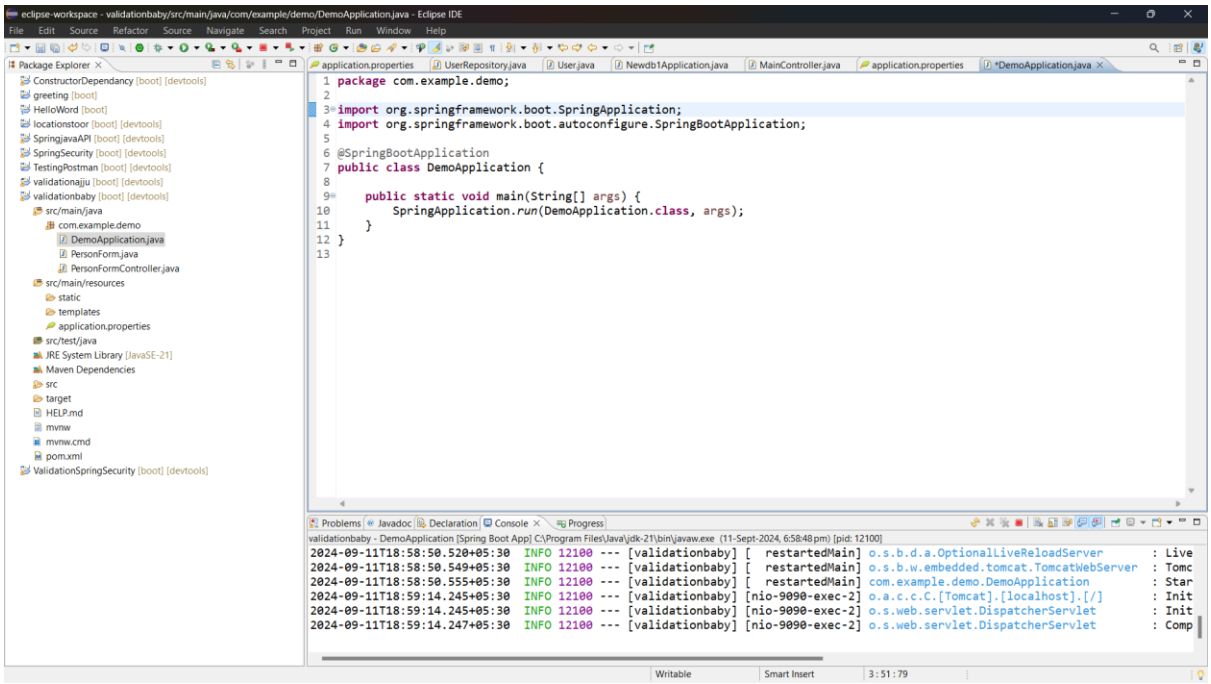
```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Form Success</title>
</head>
<body>
    <h1>Form Submission Successful</h1>
    <p th:text="'Name: ' + ${person.name}"></p>
    <p th:text="'Age: ' + ${person.age}"></p>
```

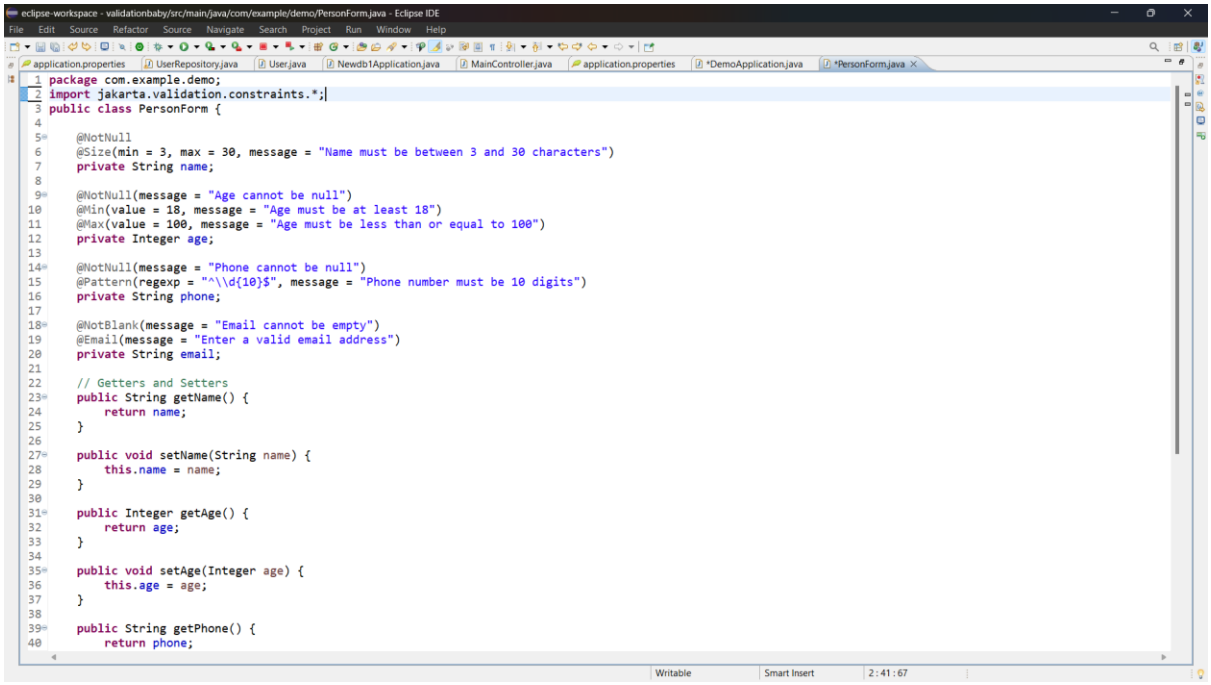
```
    <p th:text="'Phone: ' + ${person.phone}"></p>
    <p th:text="'Email: ' + ${person.email}"></p>
</body>
</html>
```

**5.personForm**

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>Person Form</title>
</head>
<body>
    <h1>Person Form</h1>
    <form action="#" th:action="@{/personForm}" th:object="${personForm}" method="post">
        <div>
            <label for="name">Name:</label>
            <input type="text" th:field="*{name}" />
            <p th:if="${#fields.hasErrors('name')}" th:errors="*{name}">Name Error</p>
        </div>
        <div>
            <label for="age">Age:</label>
            <input type="number" th:field="*{age}" />
            <p th:if="${#fields.hasErrors('age')}" th:errors="*{age}">Age Error</p>
        </div>
        <div>
            <label for="phone">Phone:</label>
```

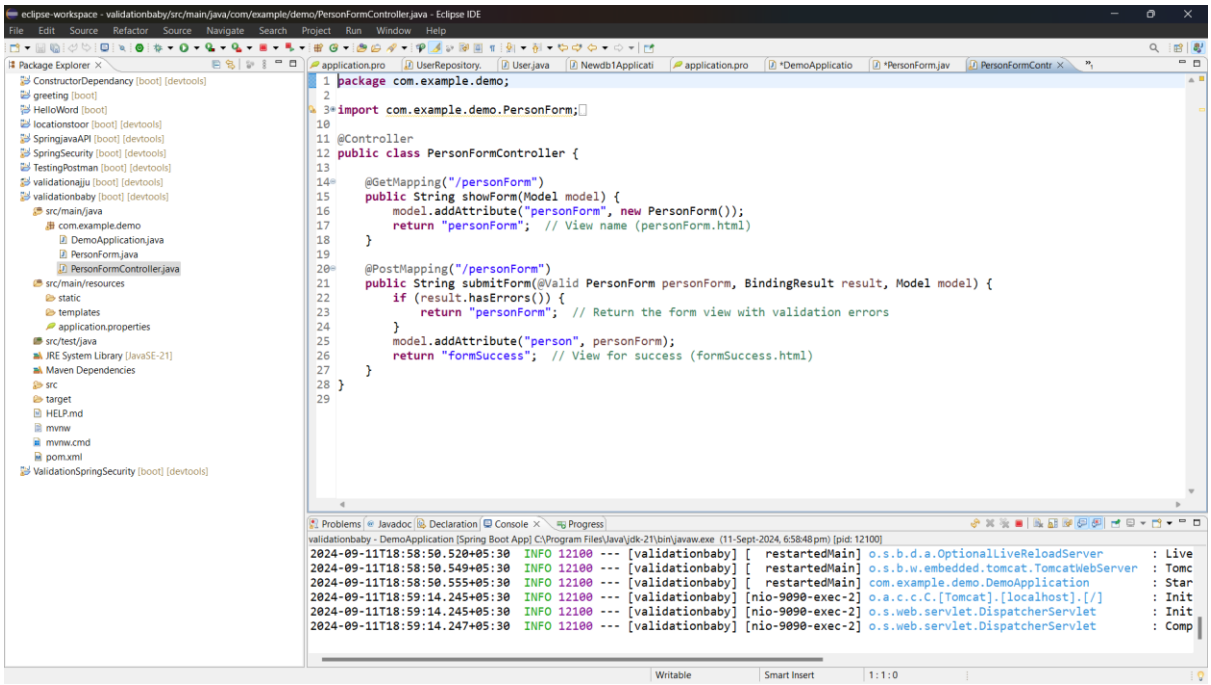
```
        <input type="text" th:field="*{phone}" />
        <p  th:if="${#fields.hasErrors('phone')}"  th:errors="*{phone}">Phone Error</p>
    </div>
    <div>
        <label for="email">Email:</label>
        <input type="email" th:field="*{email}" />
        <p  th:if="${#fields.hasErrors('email')}"  th:errors="*{email}">Email Error</p>
    </div>
    <div>
        <button type="submit">Submit</button>
    </div>
  </form>
</body>
</html>
```





```java
package com.example.demo;
import jakarta.validation.constraints.*;
public class PersonForm {

    @NotNull
    @Size(min = 3, max = 30, message = "Name must be between 3 and 30 characters")
    private String name;

    @NotNull(message = "Age cannot be null")
    @Min(value = 18, message = "Age must be at least 18")
    @Max(value = 100, message = "Age must be less than or equal to 100")
    private Integer age;

    @NotNull(message = "Phone cannot be null")
    @Pattern(regexp = "^\\d{10}$", message = "Phone number must be 10 digits")
    private String phone;

    @NotBlank(message = "Email cannot be empty")
    @Email(message = "Enter a valid email address")
    private String email;

    // Getters and Setters
    public String getName() {
        return name;
    }

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

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }

    public String getPhone() {
        return phone;
    }
```

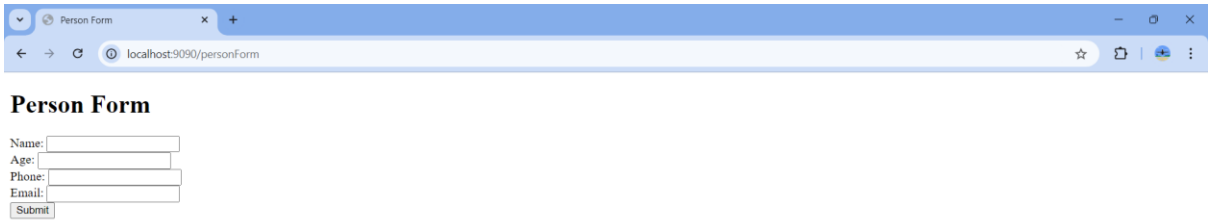
```java
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "PersonForm [name=" + name + ", age=" + age + ", phone=" + phone + ", email=" + email + "]";
    }
}
```

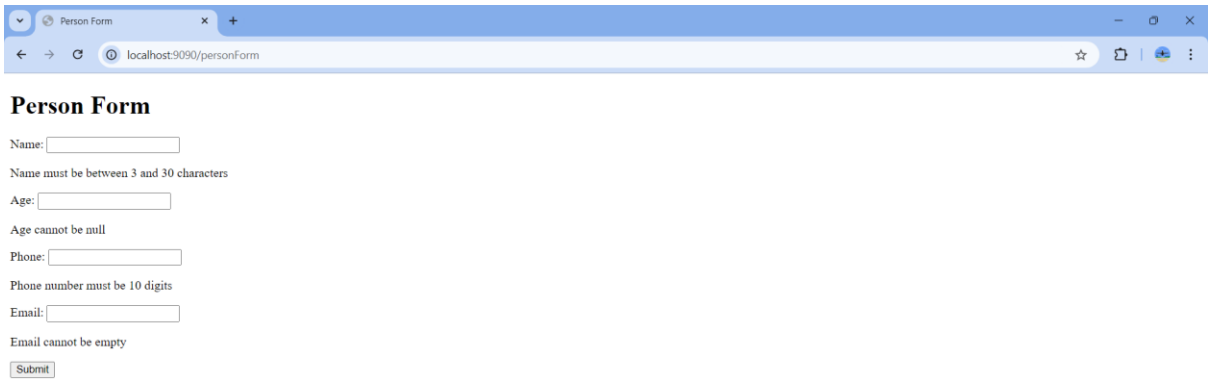


```java
package com.example.demo;

import com.example.demo.PersonForm;

@Controller
public class PersonFormController {

    @GetMapping("/personForm")
    public String showForm(Model model) {
        model.addAttribute("personForm", new PersonForm());
        return "personForm";  // View name (personForm.html)
    }

    @PostMapping("/personForm")
    public String submitForm(@Valid PersonForm personForm, BindingResult result, Model model) {
        if (result.hasErrors()) {
            return "personForm";  // Return the form view with validation errors
        }
        model.addAttribute("person", personForm);
        return "formSuccess";  // View for success (formSuccess.html)
    }
}
```

```
2024-09-11T18:58:50.520+05:30  INFO 12100 --- [validationbaby] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : Live
2024-09-11T18:58:50.549+05:30  INFO 12100 --- [validationbaby] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomc
2024-09-11T18:58:50.555+05:30  INFO 12100 --- [validationbaby] [  restartedMain] com.example.demo.DemoApplication        : Star
2024-09-11T18:59:14.245+05:30  INFO 12100 --- [validationbaby] [nio-9090-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Init
2024-09-11T18:59:14.245+05:30  INFO 12100 --- [validationbaby] [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet        : Init
2024-09-11T18:59:14.247+05:30  INFO 12100 --- [validationbaby] [nio-9090-exec-2] o.s.web.servlet.DispatcherServlet        : Comp
```

OUTPUT:


Person Form
localhost:9090/personForm
Person Form
Name:
Age:
Phone:
Email:
Submit


Person Form
localhost:9090/personForm
Person Form
Name:
Name must be between 3 and 30 characters
Age:
Age cannot be null
Phone:
Phone number must be 10 digits
Email:
Email cannot be empty
Submit