

#### 4.2D: Injection attack preventions using hibernate validator

- In a Spring Web application, we used regular expressions (regex) to produce appropriate input checking to avoid injection attacks including XSS. The assignment's synopsis is provided below
- To avoid such injection, the use of regex pattern must be adjusted to becoming more complex as shown below. The new regex will strip off all probable script tags, encoded character or some wrong inputs that lead to an XSS attack. My main purpose was to update the regex to block all of these complex patterns. There were even more powerful regex patterns that allowed for script tags detection, encoded characters, and incorrect inputs formation. Before devising this altered regex, I experimented with a few different patterns:

"^(?!.\*(|<|>|&|&#[xX]?[0-9a-fA-F]+;|javascript:|\\(\\)|%|script|on\\w+=)).\*\$"

**Client.java was updated.**



#### Validations and Fields

- After which two more fields were added to the online application:
  - ✓ Customer Text section: To avoid potentially aggressive comment or a script like monologue, we limited the amount of text which can be typed in this part to 200 characters at most.
  - ✓ Email Address: Only valid email addresses with the.com domain extension should be entered in this box and other characters than @ should be typed here.

- Deployed the web application in the web browser using the Tomcat server to execute the deployment of the web application.

Client Registration Form

localhost:8382/05-coachwebapp-spring/client/showForm

Name:

Age:

Email:

Custom Request:

- I tried to enter XSS scripts in the name box and they were prohibited there. As it was expected—I got an error message when I entered the improper email format.

Client Registration Form — Mozilla Firefox

localhost:8080/05-coachwebapp-spring/workout/processForm

Name: Kenisha "<script>alert('XSS')"; incorrect format

Age: 21

Email: keni29gmail.com Enter a valid email

Custom Request: This is my process form

- Client-side validation: As for the input fields, we're implementing HTML5 input validation to the email where we specify the correct format. For the text part, characterized blocking and character limit was used in the current study.
- Server-side validation: To prevent XSS and all kinds of injection threats, we utilized Hibernate Validator to increase the security checks on all inputs.

## New version of ClientController.java

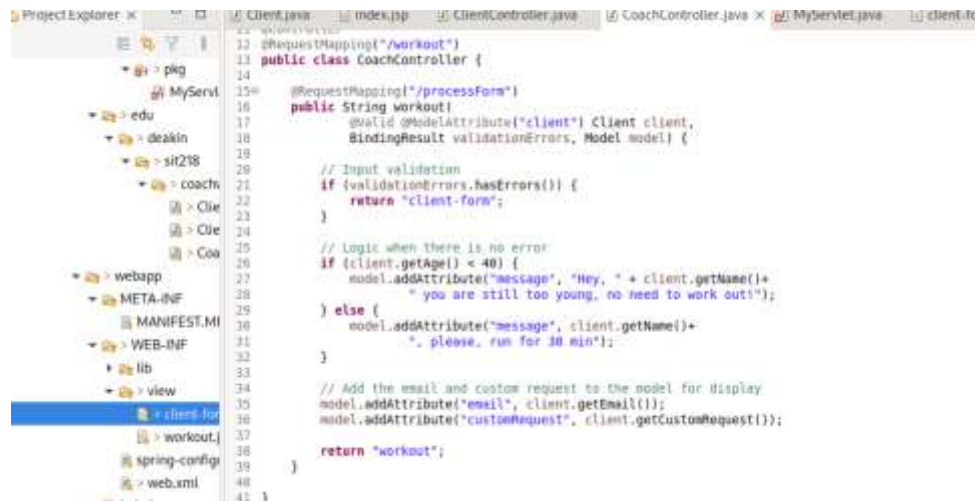


```

1 package edu.deakin.sh218.coachwebapp;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 @RequestMapping("/client")
7 public class ClientController {
8
9     @RequestMapping("/showForm")
10    public String showForm(Model model) {
11        //Create a client object
12        Client client = new Client();
13
14        //add client object to model
15        model.addAttribute("client", client);
16
17        return "client-form";
18    }
19 }

```

## New version of CoachController.java

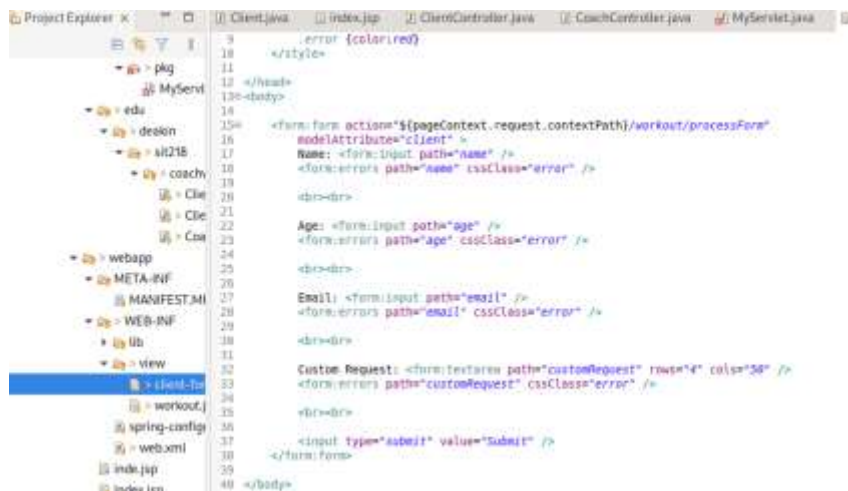


```

12 @RequestMapping("/workout")
13 public class CoachController {
14
15     @RequestMapping("/processForm")
16     public String workout(
17         @Valid @ModelAttribute("client") Client client,
18         BindingResult validationErrors, Model model) {
19
20         // Input validation
21         if (validationErrors.hasErrors()) {
22             return "client-form";
23         }
24
25         // Logic when there is no error
26         if (client.getAge() < 40) {
27             model.addAttribute("message", "Hey, " + client.getName() +
28                 " you are still too young, no need to work out!");
29         } else {
30             model.addAttribute("message", client.getName() +
31                 ". please, run for 30 min!");
32         }
33
34         // Add the email and custom request to the model for display
35         model.addAttribute("email", client.getEmail());
36         model.addAttribute("customRequest", client.getCustomRequest());
37
38         return "workout";
39     }
40 }

```

## New version of HTML page



```

9     <error {color:red}>
10     </style>
11
12     </head>
13     <body>
14
15         <form:form action="${pageContext.request.contextPath}/workout/processForm"
16             modelAttribute="client" >
17             <form:input path="name" />
18             <form:errors path="name" cssClass="error" />
19
20             <br><br>
21             Age: <form:input path="age" />
22             <form:errors path="age" cssClass="error" />
23
24             <br><br>
25             Email: <form:input path="email" />
26             <form:errors path="email" cssClass="error" />
27
28             <br><br>
29             Custom Request: <form:textarea path="customRequest" rows="4" cols="50" />
30             <form:errors path="customRequest" cssClass="error" />
31
32             <br><br>
33             <input type="submit" value="Submit" />
34         </form:form>
35
36     </body>

```

## New version of Client.java

```
package edu.deakin.sit218.coachwebapp;

import javax.validation.constraints.Email;
import javax.validation.constraints.Max;
import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Pattern;
import javax.validation.constraints.Size;

public class Client {

    @NotNull(message = "is required")
    @Size(min = 3, message="is required")
    @Pattern(regexp = "^(?!.*(<|>|&lt;|&gt;|&|&#xX)?[0-9a-fA-F]+;|javascript:|\\(\\)|%|script|on\\w+=)).*$",
            message="incorrect format")
    protected String name;

    @NotNull(message = "is required")
    @Min(value=18, message = "You must be 18 years old or older")
    @Max(value=120, message = "Vampires are not allowed")
    protected int age;

    @NotNull(message = "Email is required")
    @Email(message = "Enter a valid email")
    protected String email;

    @NotNull(message = "Message is required")
    @Size(max = 200, message = "Request cannot be more than 200 characters")
    protected String customRequest;

    // Getters and setters for all fields
    public String getName() {
        return name;
    }

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

    public int getAge() {
        return age;
    }

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

```
    }

    public String getEmail() {
        return email;
    }

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

    public String getCustomRequest() {
        return customRequest;
    }

    public void setCustomRequest(String customRequest) {
        this.customRequest = customRequest;
    }
}
```

## New version of CoachController.java

```
package edu.deakin.sit218.coachwebapp;

import javax.validation.Valid;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/workout")
public class CoachController {

    @RequestMapping("/processForm")
    public String workout(
        @Valid @ModelAttribute("client") Client client,
        BindingResult validationErrors, Model model) {

        if (validationErrors.hasErrors()) {
            return "client-form";
        }

        if (client.getAge() < 40) {
            model.addAttribute("message", "Hey, " + client.getName()+
                " you are still too young, no need to work out!");
        } else {
            model.addAttribute("message", client.getName()+
                ", please, run for 30 min");
        }

        model.addAttribute("email", client.getEmail());
        model.addAttribute("customRequest", client.getCustomRequest());

        return "workout";
    }
}
```

## New version of HTML page

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<!DOCTYPE html>
<html>
<head>
<title>Client Registration Form</title>

    <!-- Inline CCS -->
    <style>
        .error {color:red}
    </style>

</head>
<body>

    <form:form action="${pageContext.request.contextPath}/workout/processForm"
        modelAttribute="client" >
        Name: <form:input path="name" />
        <form:errors path="name" cssClass="error" />

        <br><br>

        Age: <form:input path="age" />
        <form:errors path="age" cssClass="error" />

        <br><br>

        Email: <form:input path="email" />
        <form:errors path="email" cssClass="error" />

        <br><br>

        Custom Request: <form:textarea path="customRequest" rows="4" cols="50" />
        <form:errors path="customRequest" cssClass="error" />

        <br><br>

        <input type="submit" value="Submit" />
    </form:form>

</body>
</html>
```

## New version of ClientController.java

```
package edu.deakin.sit218.coachwebapp;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@RequestMapping("/client")
public class ClientController {

    @RequestMapping("/showForm")
    public String showForm(Model model) {
        //Create a client object
        Client client = new Client();

        //add client object to model
        model.addAttribute("client", client);

        return "client-form";
    }
}
```