# ClubUMLSpring2014 Ameya Joshi & Sahil Patil

**Team Validation Sprint 3**

The validation team has been working on logically integrating the JavaScript into single files. The JavaScript for the validation of different pages which have similar validations logically have been grouped together into one file.

* The JavaScript for index.jsp and register.jsp has similar functionality related to account information and logins. Hence, the JavaScript for the validation of these two pages have been merged together.
* A JavaScript was created for validation of the display.jsp. We plan to integrate the JavaScript used for validation of managePolicy.jsp with this file.

While implementing these functionalities, we have *edited* the following files:

1. Index.jsp & register.jsp.
2. Register.js: New functionalities have been added for validation.
3. Display.jsp & managePolicy.jsp: Functionalities added to perform checks on the data being added/uploaded.

During the procedure, we *created* some new files to accommodate these additional features:

1. ValidateEmail.java: This java class is responsible for making sure that the email being used in account registration is not already present in the database to make sure that one email is used to create only one account.
2. Display.js: This file contains the check for file upload to make sure that the file being uploaded is of the type “Ecore” or “XMI”.

Apart from this, Ameya also acted as the third random person selected for the testing for the Rationale Management team and carried out manual testing on the dialog box feature and database changes made by the team.

**Plan of Action:**

* We plan to integrate the JavaScript for managePolicy.jsp into the one used for display.jsp.
* Manual test the code changes made until now by three people.
* Get the project ready to be merged with the master repo.

**Code changes:**

1. **Display.js:**
2. /\*\*
3. \* **@author** AmeyaCJoshi
4. \* **@description** This javascript file helps in validation of Display.jsp
5. \*/
6. **function** checkError() {
7. **var** block = document.getElementById("errorMsg");
8. **var** fileName = document.getElementById("file1");
9. **var** flag = **false**;
10. //var pattern = new RegExp ("/^.\*\.(ecore|ECORE)$/");
11. //flag = pattern.test(fileName);
12. **var** ext = "";
13. **var** a = fileName.value.split(".");
14. **if** (a.length === 1 || (a[0] === "" && a.length === 2)) {
15. ext = "";
16. }
17. ext = a.pop();
18. **if** (ext == "ecore" || ext == "ECORE" || ext == "xmi" || ext == "XMI")
19. flag = **true**;
20. **if** (!flag) {
21. alert("File type must be either .ecore or .xmi!");
22. }
23. }
24. **function** checkFields() {
25. **if** (type == "downloadButton") {
26. **if** ($(".myCheckBox:checked").length == 1) {
27. **return** **true**;
28. }
29. alert("Please select 1 diagram for download!");
30. **return** **false**;
31. }
32. **if** (type == "compareButton") {
33. **var** check = document.getElementsByName("check");
34. **var** checked = [];
35. **for** (**var** i = 0; i < check.length; i++) {
36. **if** (check[i].checked) {
37. checked.push(check[i]);
38. }
39. }
40. // Check for 2 Ecore; XMI not supported++
41. **var** bothValid = **true**;
42. **if** (checked.length == 2) {
43. **if** (checked[0].id == "sequence" || checked[1].id == "sequence") {
44. bothValid = **false**;
45. }
46. **if** (checked[0].id == "class" || checked[1].id == "class") {
47. bothValid = **false**;
48. }
49. }
50. **if** (checked.length == 2 && bothValid) {
51. // DisplayDiagram looks for ID numbers in the checked.value fields
52. **return** **true**;
53. }
54. alert("Please select 2 Ecore diagrams to compare");
55. **return** **false**;
56. }
57. **if** (type == "displayButton") {
58. **if** ($(".myCheckBox:checked").length == 1) {
59. **return** **true**;
60. }
61. alert("Please select 1 diagram to display");
62. **return** **false**;
63. }
64. // merge function
65. **if** (type == "mergeButton") {
66. **var** req = document.getElementById("req");
67. **var** form = document.getElementById("requestForm");
68. **var** check = document.getElementsByName("check");
69. **var** checked = [];
70. **for** (**var** i = 0; i < check.length; i++) {
71. **if** (check[i].checked) {
72. checked.push(check[i]);
73. }
74. }
75. // Check for 2 XMI class; Ecore or sequence not supported
76. **var** bothValid = **true**;
77. **if** (checked.length == 2) {
78. **if** (checked[0].id == "sequence" || checked[1].id == "sequence") {
79. bothValid = **false**;
80. }
81. **if** (checked[0].id == "Ecore" || checked[1].id == "Ecore") {
82. bothValid = **false**;
83. }
84. }
85. **if** (checked.length == 2 && bothValid) {
86. **var** reqO = {};
87. reqO.Request = "Refresh";
88. reqO.Diagram1 = checked[0].value;
89. reqO.Diagram2 = checked[1].value;
90. req.value = JSON.stringify(reqO);
91. form.submit();
92. **return** **false**;
93. }
94. alert("Please select 2 XMI Class diagrams to merge.");
95. **return** **false**;
96. }
97. // end
98. }
99. **function** displayClassDiagramFields(element) {
100. **var** option = element.options[element.selectedIndex].text;
101. **var** selectLabel = document.getElementById("DiagramSelectLabel");
102. **var** fileinput2 = document.getElementById("file2");
103. **var** fileinput3 = document.getElementById("file3");
104. **if** (option == "ECORE") {
105. fileinput2.style.display = "none";
106. fileinput3.style.display = "none";
107. selectLabel.innerHTML = "Class Diagram Format: (.ecore)";
108. } **else** **if** (option == "XMI") {
109. fileinput2.style.display = "block";
110. fileinput3.style.display = "block";
111. selectLabel.innerHTML = "Class Diagram Format: (.di, .notation, .uml)";
112. }
113. }
114. **function** toggleChecked(status) {
115. $(".myCheckBox").each(**function**() {
116. $(**this**).attr("checked", status);
117. })
118. }

**2) ValidateEmail.java:**

/\*\*

\* **@author** AmeyaCJoshi

\* Information class that contains all the features of ValidateServlet.

\* **@description** This class is used to make sure that one email id is used

\* to create only one account.

\*

\*/

**package** controller;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**import** java.sql.ResultSet;

**import** java.sql.SQLException;

**import** java.sql.Statement;

**import** java.util.logging.Level;

**import** java.util.logging.Logger;

**import** javax.servlet.RequestDispatcher;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

**import** repository.UserDAO;

**import** domain.User;

**public** **class** ValidateEmail **extends** HttpServlet {

/\*\*

\* Processes requests for both HTTP

\* <code>GET</code> and

\* <code>POST</code> methods.

\*

\* **@param** request servlet request

\* **@param** response servlet response

\* **@throws** ServletException if a servlet-specific error occurs

\* **@throws** IOException if an I/O error occurs

\*/

**protected** **void** processRequest(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException

{

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

response.setContentType("text/html;charset=UTF-8");

String email = request.getParameter("email");

User checkEmailExist = UserDAO.*getUserEmail*(email);

**if** (checkEmailExist != **null**)

{

out.println("<font color='red'>Account for this email already exists!");

}

**else**

{

out.println("<font color='green'>Email valid!");

}

}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/\*\*

\* Handles the HTTP

\* <code>GET</code> method.

\*

\* **@param** request servlet request

\* **@param** response servlet response

\* **@throws** ServletException if a servlet-specific error occurs

\* **@throws** IOException if an I/O error occurs

\*/

@Override

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Handles the HTTP

\* <code>POST</code> method.

\*

\* **@param** request servlet request

\* **@param** response servlet response

\* **@throws** ServletException if a servlet-specific error occurs

\* **@throws** IOException if an I/O error occurs

\*/

@Override

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

processRequest(request, response);

}

/\*\*

\* Returns a short description of the servlet.

\*

\* **@return** a String containing servlet description

\*/

@Override

**public** String getServletInfo() {

**return** "Short description";

}// </editor-fold>

}