

Regex Application

Tema 1

Regex Application este o aplicatie creata folosind ASP NET (MVC). Aceasta aplicatie peermite testarea unei expresii regulate.

Pagina Principală:

Regular expressions

Pattern

Text1

Text2

Matched

Matches

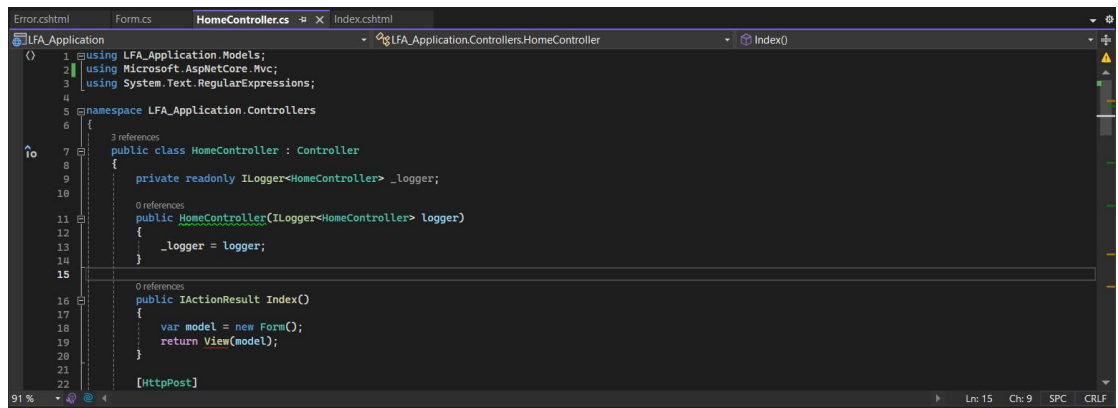
Apply

Aplicatia contine 3 input-uri:

- primul input reprezinta pattern-ul care trebuie introdus,
- al doilea input reprezinta Text1
- al treilea input reprezinta Text2

Matched indica faptul ca Text1 este in match cu pattern-ul. Acesta are o valoare default "Not verified".

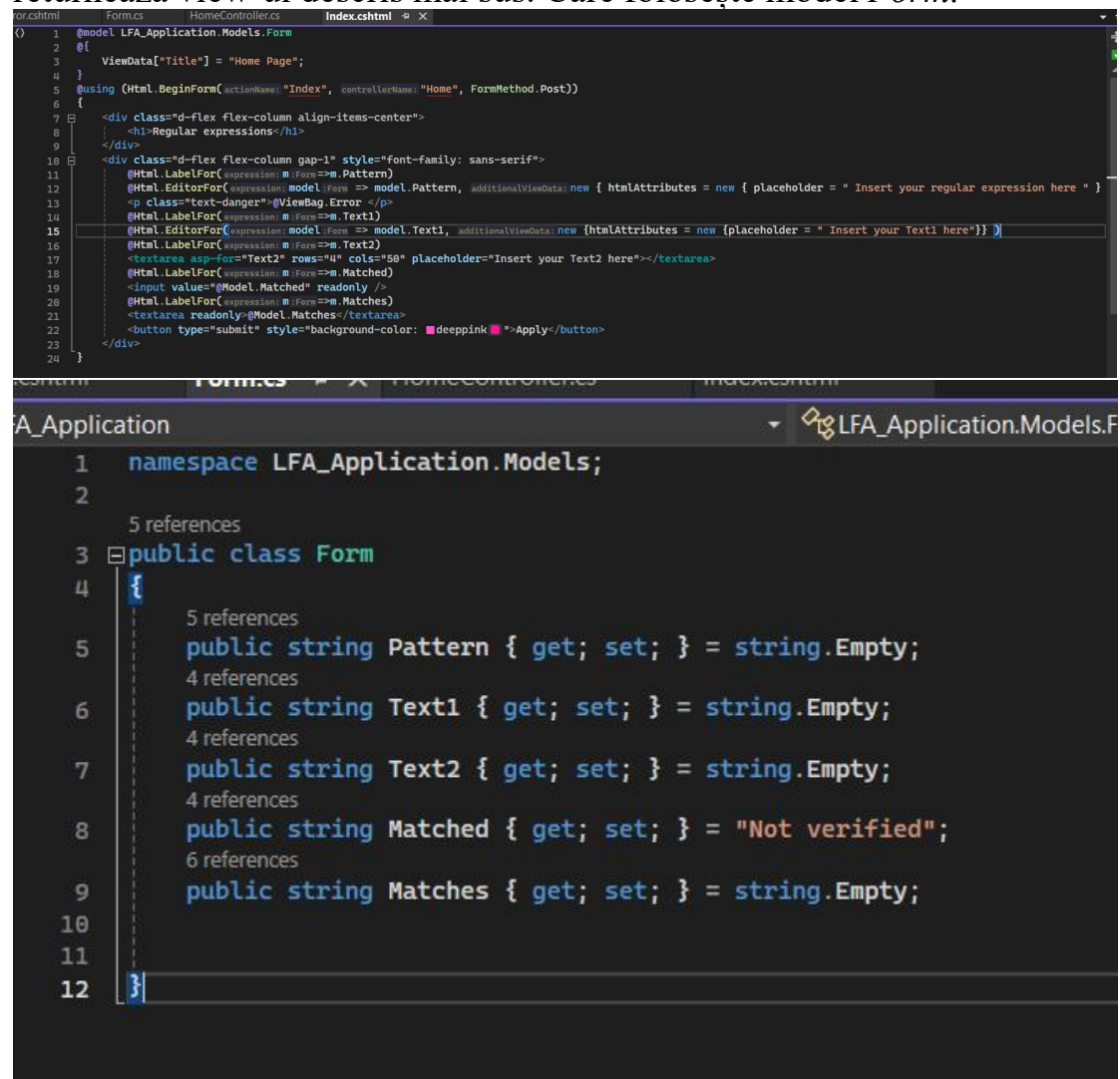
Matches extrage din Text2 toate stringurile care indeplinesc pattern-ul si le afiseaza.



```
1 using LFA_Application.Models;
2 using Microsoft.AspNetCore.Mvc;
3 using System.Text.RegularExpressions;
4
5 namespace LFA_Application.Controllers
6 {
7     public class HomeController : Controller
8     {
9         private readonly ILogger<HomeController> _logger;
10
11         public HomeController(ILogger<HomeController> logger)
12         {
13             _logger = logger;
14         }
15
16         public IActionResult Index()
17         {
18             var model = new Form();
19             return View(model);
20         }
21
22         [HttpPost]
```

```
21  
22 [HttpPost]  
23 0 references  
24 public IActionResult Index(Form model)  
25 {  
26     if (string.IsNullOrEmpty(model.Pattern) && string.IsNullOrEmpty(model.Text1) &&  
27         string.IsNullOrEmpty(model.Text2))  
28     {  
29         return View(model);  
30     }  
31  
32     if (!IsValidRegexPattern(model.Pattern))  
33     {  
34         ViewBag.Error = "Pattern-ul introdus nu este valid.";   
35         return View(model);  
36     }  
37     else  
38     {  
39         ViewBag.Error = "";  
40     }  
41  
42     if (!ModelState.IsValid)  
43     {  
44         return View(model);  
45     }  
46  
47     Regex regex = new Regex(model.Pattern);  
48  
49     var forText1_Match = regex.Match(model.Text1);  
50     if (forText1_Match.Success)  
51     {  
52         model.Matched = "Matched";  
53     }  
54     else  
55     {  
56         model.Matched = "DisMatched";  
57     }  
58  
59     var forText2_MatchCollection = regex.Matches(model.Text2);  
60  
61     foreach (Match match in forText2)  
62     {  
63         if (string.IsNullOrEmpty(model.Matches))  
64         {  
65             model.Matches = match.ToString();  
66         }  
67         else  
68         {  
69             model.Matches = model.Matches + ", " + match.ToString();  
70         }  
71     }  
72  
73     return View(model);  
74  
75  
76 1 reference  
77 private bool IsValidRegexPattern(string pattern)  
78 {  
79     try  
80     {  
81         Regex.Match(input: "", pattern);  
82         return true;  
83     }  
84     catch (ArgumentException)  
85     {  
86         return false;  
87     }  
88 }  
89 }
```

Pe ruta “/” se apelează constroller-ul *Home*. Din metoda Index, se returnează view-ul descris mai sus. Care folosește model *Form*.



The image shows two parts of a Visual Studio Code editor. The top part displays the `Index.cshtml` file, which is a Razor view. It starts with a model declaration `@model LFA_Application.Models.Form` and a `@{}` block containing `ViewData["Title"] = "Home Page";` and a `@using` directive for `Html.BeginForm`. The view body contains HTML markup for a form with two text inputs and a submit button. The bottom part shows the `LFA_Application.Models` namespace, where the `Form` class is defined. It has five public string properties: `Pattern`, `Text1`, `Text2`, `Matched`, and `Matches`, each with a getter and setter and an initial value.

```
1 @model LFA_Application.Models.Form
2 @{
3     ViewData["Title"] = "Home Page";
4 }
5 @using (Html.BeginForm(actionName: "Index", controllerName: "Home", FormMethod.Post))
6 {
7     <div class="d-flex flex-column align-items-center">
8         <h1>Regular expressions</h1>
9     </div>
10
11     <div class="d-flex flex-column gap-1" style="font-family: sans-serif">
12         @Html.LabelFor(expression: m => m.Pattern)
13         @Html.EditorFor(expression: model.Form => model.Pattern, additionalViewData: new { htmlAttributes = new { placeholder = " Insert your regular expression here " } })
14         <p class="text-danger">@ViewBag.Error </p>
15         @Html.LabelFor(expression: m => m.Text1)
16         @Html.EditorFor(expression: model.Form => model.Text1, additionalViewData: new {htmlAttributes = new {placeholder = " Insert your Text1 here"}} >
17         @Html.LabelFor(expression: m => m.Text2)
18         <textarea asp-for="Text2" rows="4" cols="50" placeholder="Insert your Text2 here"></textarea>
19         @Html.LabelFor(expression: m => m.Matched)
20         <input value="@Model.Matched" readonly />
21         @Html.LabelFor(expression: m => m.Matches)
22         <textarea readonly="@Model.Matches">/textarea>
23         <button type="submit" style="background-color: #deeppink">Apply</button>
24     </div>
25 }
```

```
1 namespace LFA_Application.Models;
2
3 5 references
4 public class Form
5 {
6     5 references
7     public string Pattern { get; set; } = string.Empty;
8     4 references
9     public string Text1 { get; set; } = string.Empty;
10    4 references
11    public string Text2 { get; set; } = string.Empty;
12    4 references
13    public string Matched { get; set; } = "Not verified";
14    6 references
15    public string Matches { get; set; } = string.Empty;
16 }
17 }
```

Pentru a valida texturile, folosim Regex din System.Text. Folosim o metodă de validare pentru a valida pattern-ul. Dacă acesta este valid, se formează o instanță de regex care ne ajută să validăm primul text și să extragem match-urile pentru al doilea text.