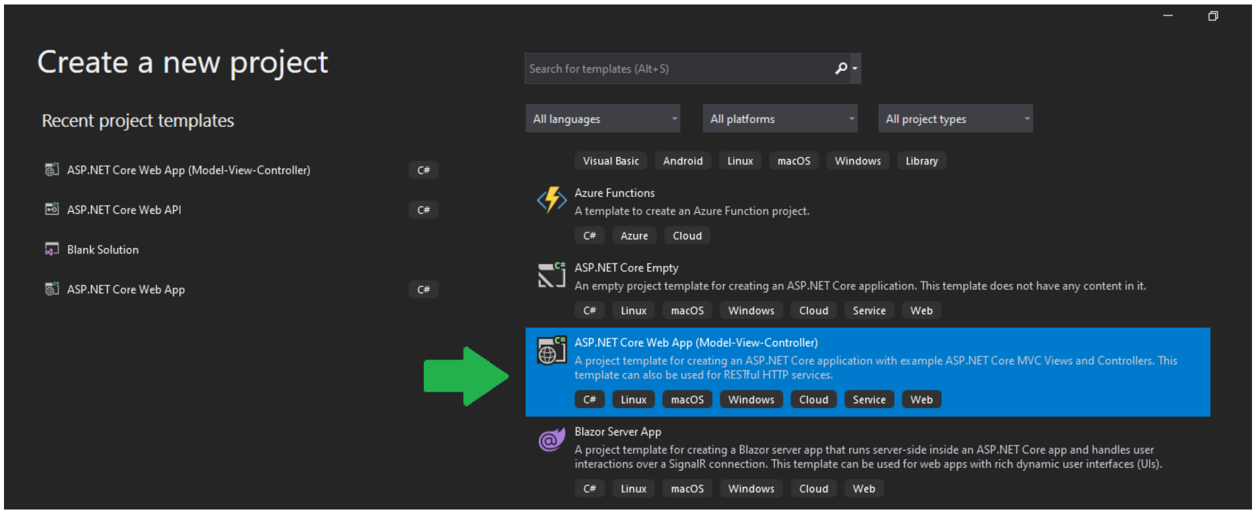
Server Side Validations in ASP.NET Core MVC when form post is via AJAX

Let us see how we can fire validations when we have a form post via jQuery AJAX in ASP.NET Core MVC

1) Create a sample ASP.NET Core MVC Project - I have VS 2019 with .Net Core version 3.1



Create a new project

Recent project templates

ASP.NET Core Web App (Model-View-Controller) C#
ASP.NET Core Web API C#
Blank Solution
ASP.NET Core Web App C#

Search for templates (Alt+S)

All languages | All platforms | All project types

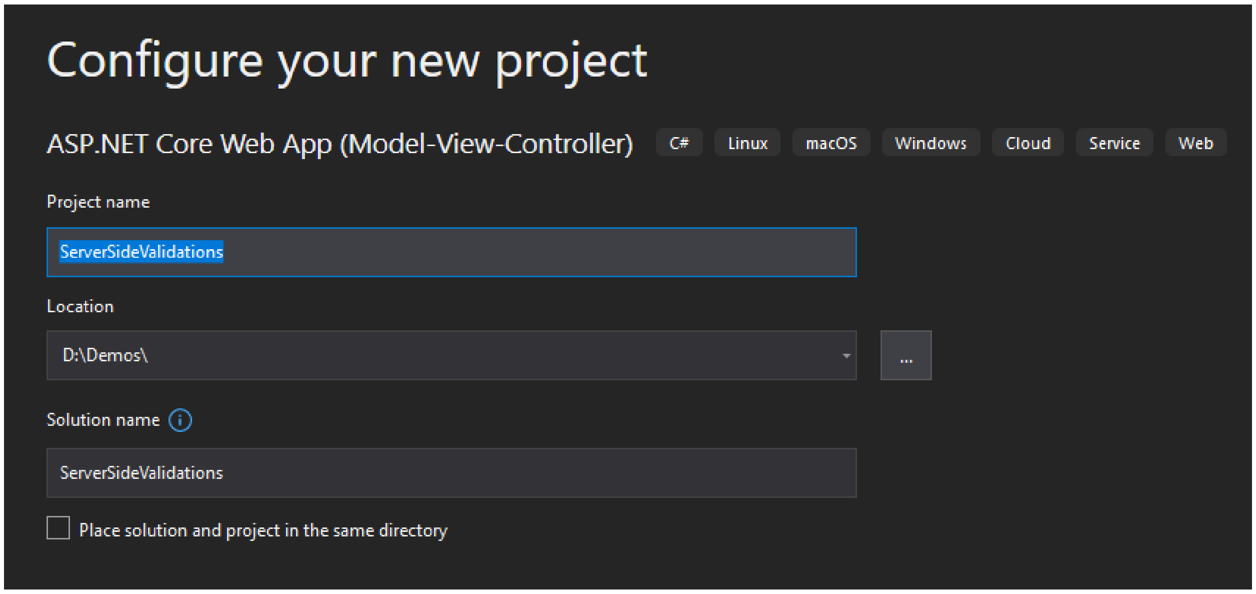
Visual Basic | Android | Linux | macOS | Windows | Library

Azure Functions
A template to create an Azure Function project.
C# | Azure | Cloud

ASP.NET Core Empty
An empty project template for creating an ASP.NET Core application. This template does not have any content in it.
C# | Linux | macOS | Windows | Cloud | Service | Web

ASP.NET Core Web App (Model-View-Controller)
A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.
C# | Linux | macOS | Windows | Cloud | Service | Web

Blazor Server App
A project template for creating a Blazor server app that runs server-side inside an ASP.NET Core app and handles user interactions over a SignalR connection. This template can be used for web apps with rich dynamic user interfaces (UIs).
C# | Linux | macOS | Windows | Cloud | Web



Configure your new project

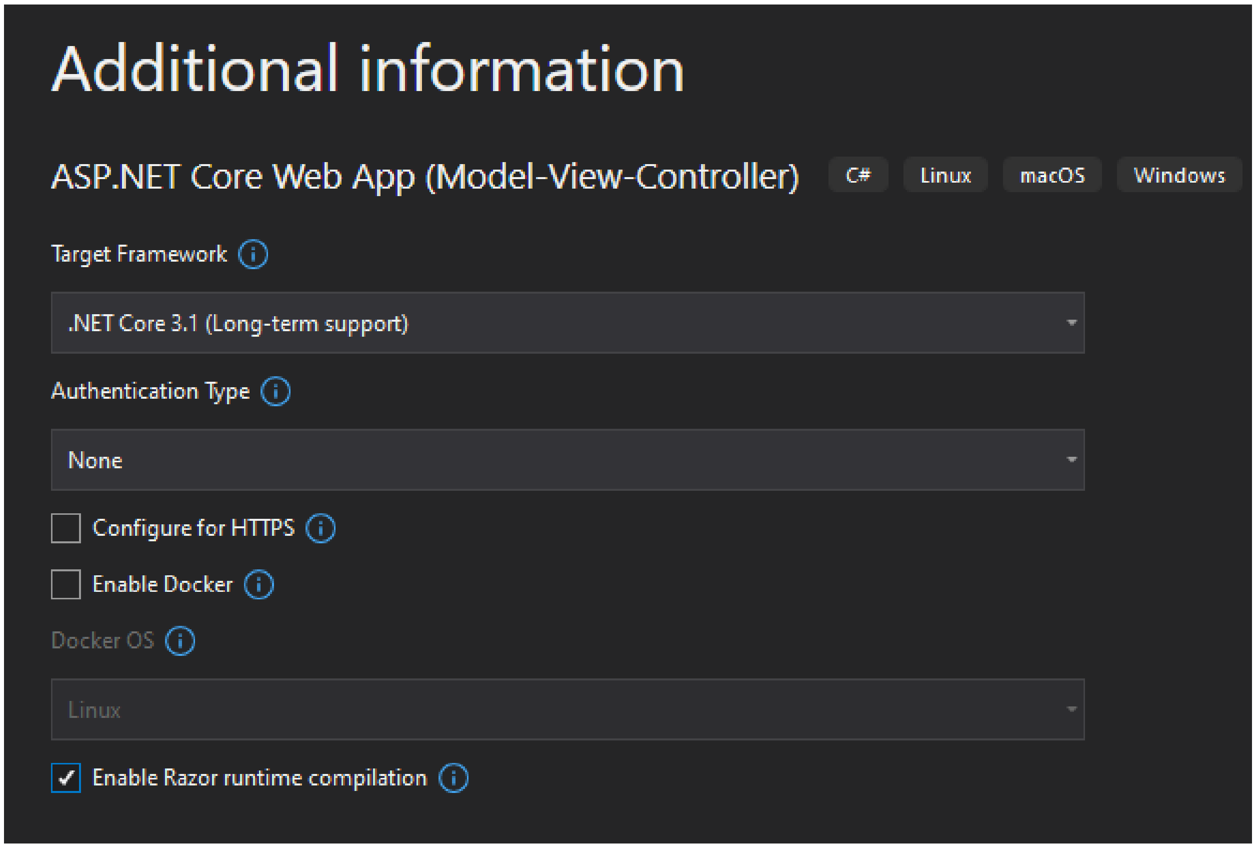
ASP.NET Core Web App (Model-View-Controller) C# | Linux | macOS | Windows | Cloud | Service | Web

Project name: ServerSideValidations

Location: D:\Demos\

Solution name: ServerSideValidations

☐ Place solution and project in the same directory


Additional information

ASP.NET Core Web App (Model-View-Controller) C# Linux macOS Windows

Target Framework: .NET Core 3.1 (Long-term support)

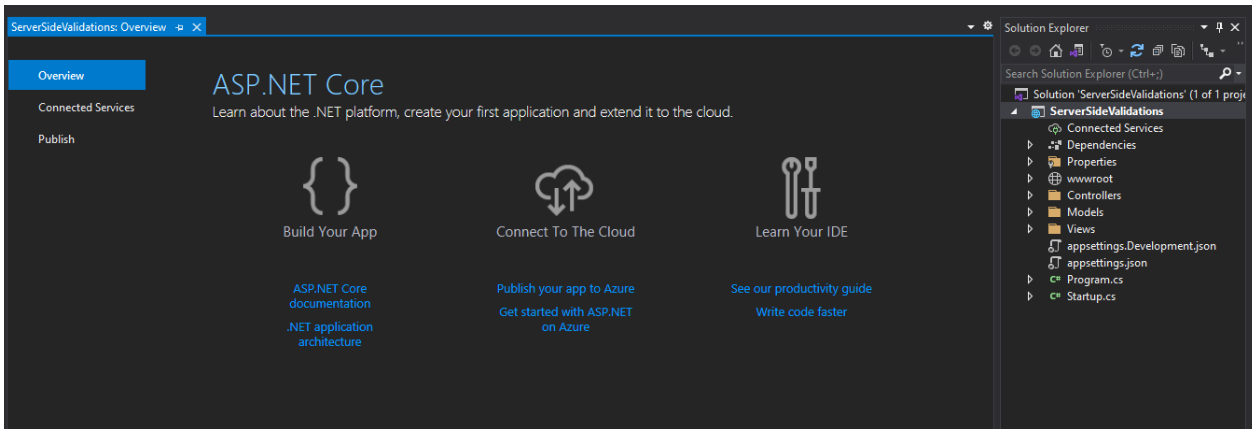
Authentication Type: None

[ ] Configure for HTTPS

[ ] Enable Docker

Docker OS: Linux

[✓] Enable Razor runtime compilation


ServerSideValidations: Overview

Overview
Connected Services
Publish

ASP.NET Core
Learn about the .NET platform, create your first application and extend it to the cloud.

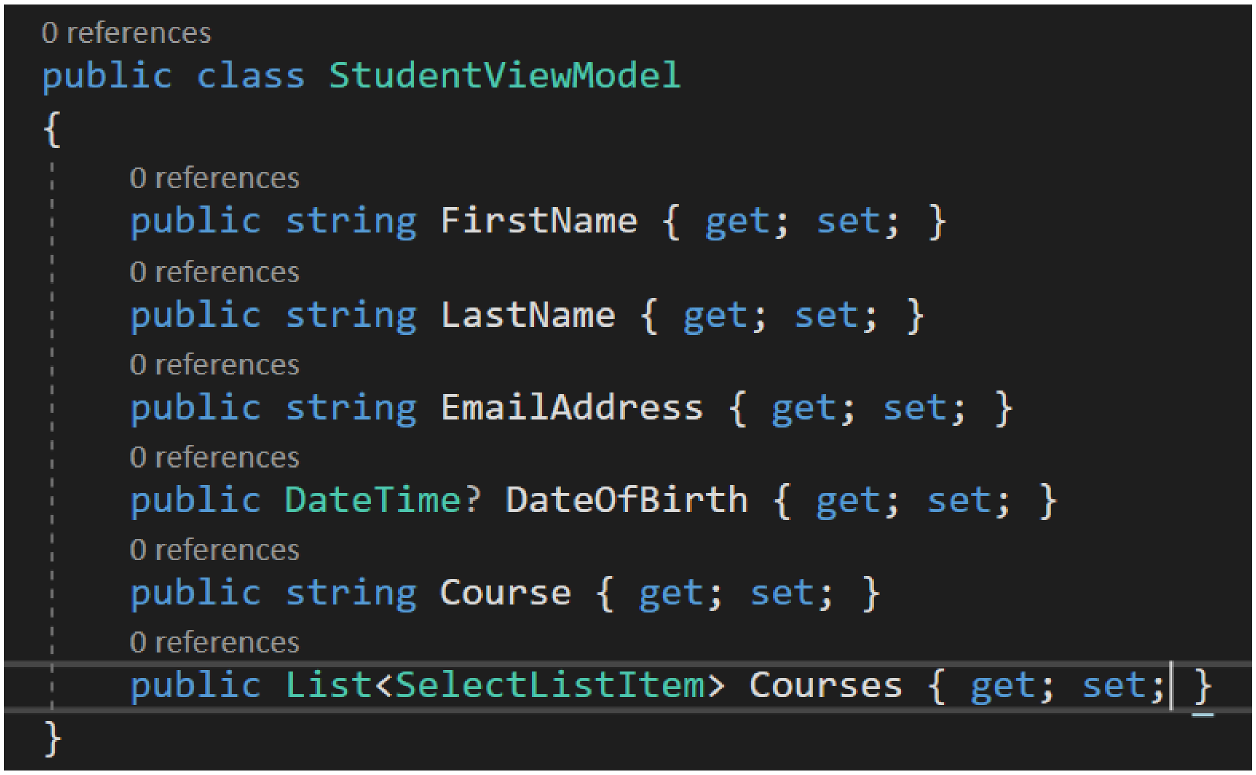
Build Your App: ASP.NET Core documentation; .NET application architecture
Connect To The Cloud: Publish your app to Azure; Get started with ASP.NET on Azure
Learn Your IDE: See our productivity guide; Write code faster

Solution Explorer
Search Solution Explorer (Ctrl+;)

```
Solution 'ServerSideValidations' (1 of 1 proje
  ServerSideValidations
    Connected Services
    Dependencies
    Properties
    wwwroot
    Controllers
    Models
    Views
    appsettings.Development.json
    appsettings.json
    Program.cs
    Startup.cs
```

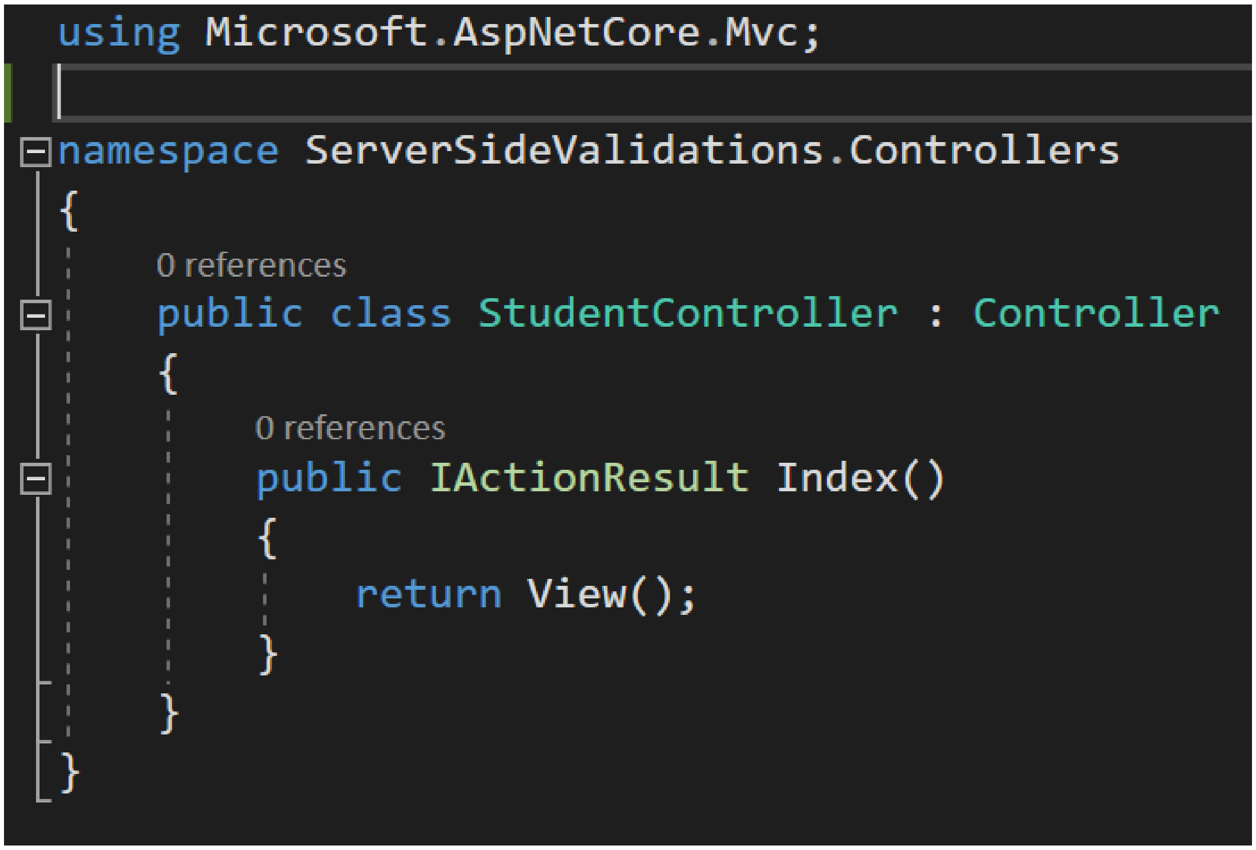
2) The project is created in step 1 - Let us now create a form and try to submit the form using jQuery AJAX.
   For simplicity purposes I will create a Student Form, with few basic details.
   Let us first create a model to mimic the student details.

   StudentViewModel is created as below, takes in the first name, last name, email address, date of birth and a list of courses in which the student wishes to enroll.



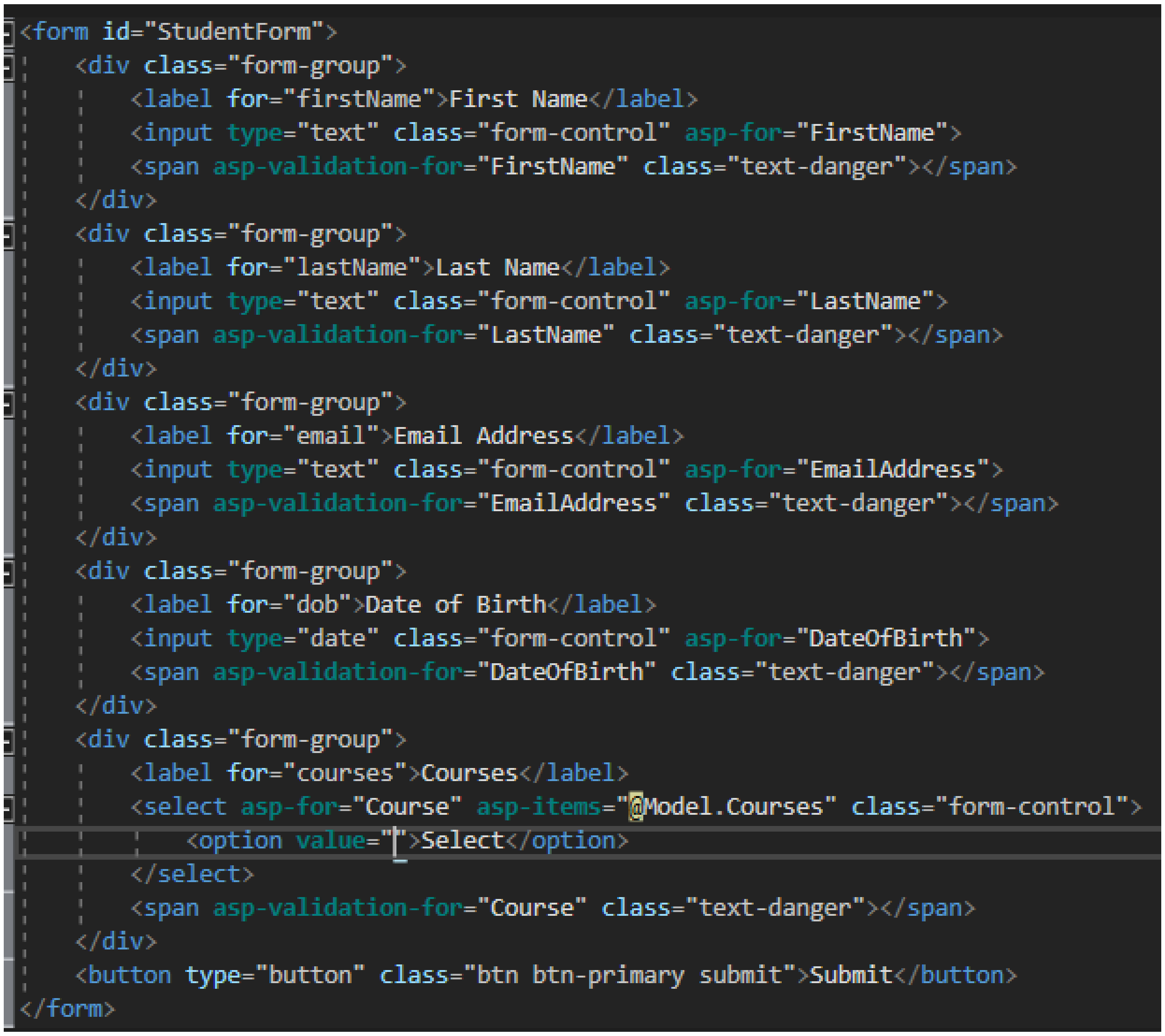
```
0 references
public class StudentViewModel
{
    0 references
    public string FirstName { get; set; }
    0 references
    public string LastName { get; set; }
    0 references
    public string EmailAddress { get; set; }
    0 references
    public DateTime? DateOfBirth { get; set; }
    0 references
    public string Course { get; set; }
    0 references
    public List<SelectListItem> Courses { get; set; }
}
```

3) Let us create a StudentController to encompass the actions related to the Student.



```
using Microsoft.AspNetCore.Mvc;

namespace ServerSideValidations.Controllers
{
    0 references
    public class StudentController : Controller
    {
        0 references
        public IActionResult Index()
        {
            return View();
        }
    }
}
```

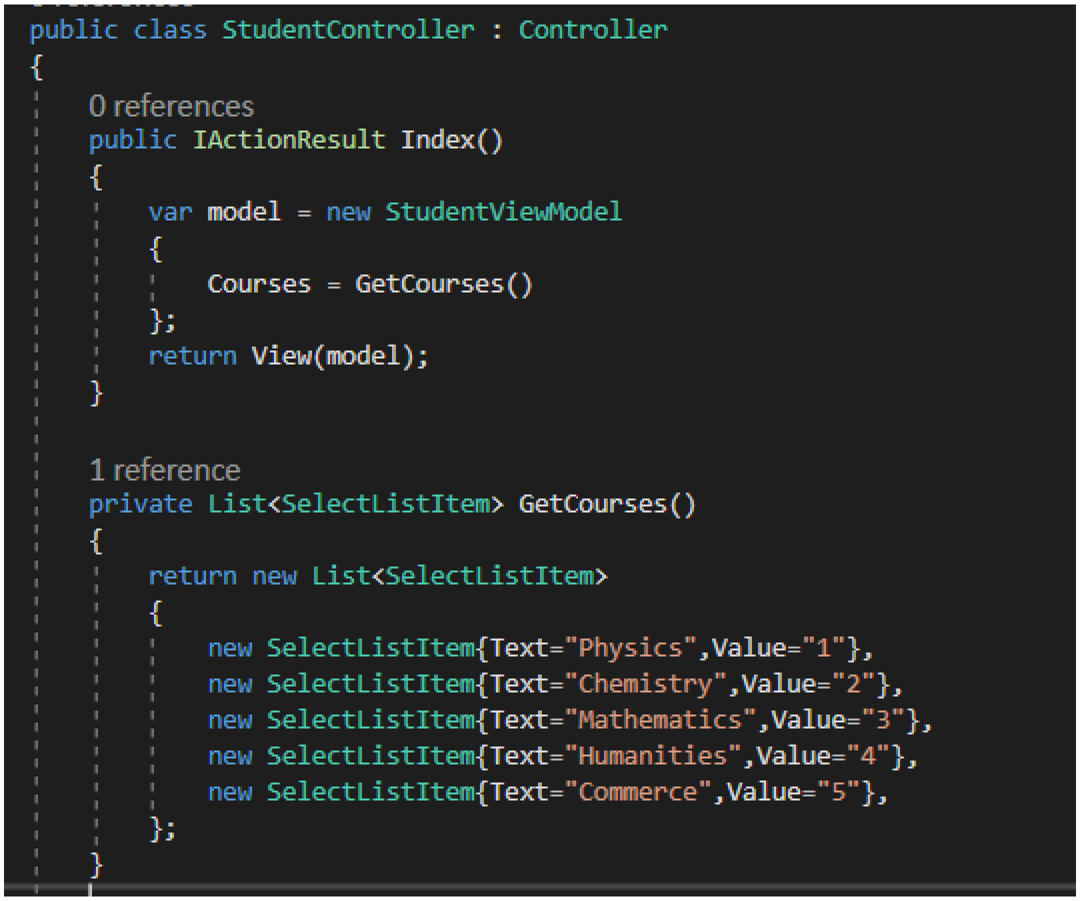
4) Let us create the Student Form in the Index.cshtml in Views/Student folder.



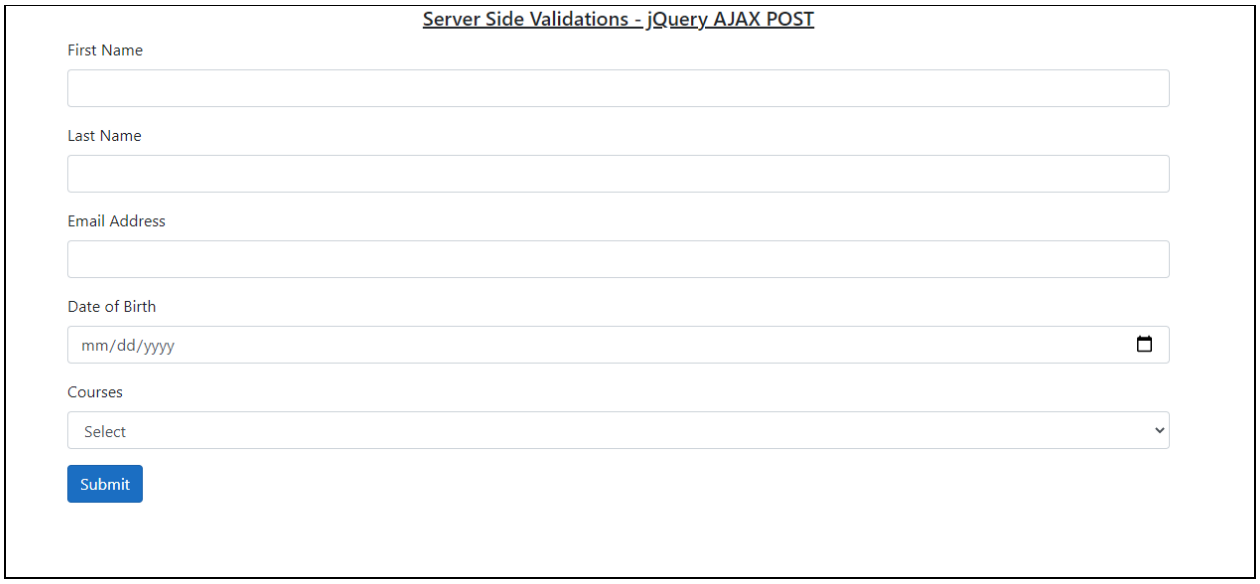
```
<form id="StudentForm">
    <div class="form-group">
        <label for="firstName">First Name</label>
        <input type="text" class="form-control" asp-for="FirstName">
        <span asp-validation-for="FirstName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label for="lastName">Last Name</label>
        <input type="text" class="form-control" asp-for="LastName">
        <span asp-validation-for="LastName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label for="email">Email Address</label>
        <input type="text" class="form-control" asp-for="EmailAddress">
        <span asp-validation-for="EmailAddress" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label for="dob">Date of Birth</label>
        <input type="date" class="form-control" asp-for="DateOfBirth">
        <span asp-validation-for="DateOfBirth" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label for="courses">Courses</label>
        <select asp-for="Course" asp-items="@Model.Courses" class="form-control">
            <option value="">Select</option>
        </select>
        <span asp-validation-for="Course" class="text-danger"></span>
    </div>
    <button type="button" class="btn btn-primary submit">Submit</button>
</form>
```

We have strongly typed the Student Form with the StudentViewModel and are using tag helpers like asp-for, asp-validation-for to bind the input and span controls with the individual model properties.

5) Let us create a list of dummy courses and then populate the student form with an empty model, so that the form displays on the default INDEX action of the StudentController.



```
public class StudentController : Controller
{
    0 references
    public IActionResult Index()
    {
        var model = new StudentViewModel
        {
            Courses = GetCourses()
        };
        return View(model);
    }

    1 reference
    private List<SelectListItem> GetCourses()
    {
        return new List<SelectListItem>
        {
            new SelectListItem{Text="Physics",Value="1"},
            new SelectListItem{Text="Chemistry",Value="2"},
            new SelectListItem{Text="Mathematics",Value="3"},
            new SelectListItem{Text="Humanities",Value="4"},
            new SelectListItem{Text="Commerce",Value="5"},
        };
    }
```



Server Side Validations - jQuery AJAX POST

First Name

Last Name

Email Address

Date of Birth

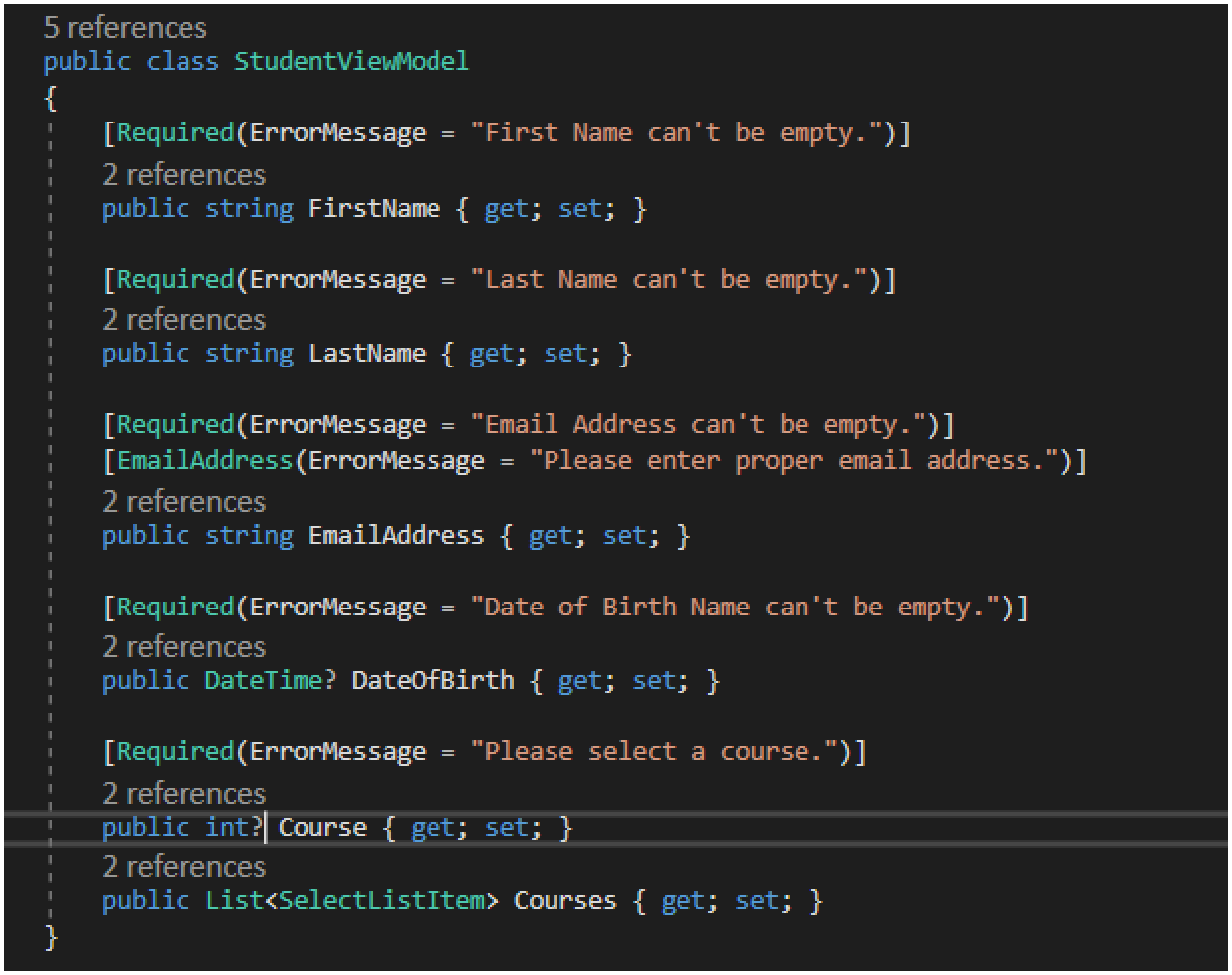
mm/dd/yyyy

Courses

Select

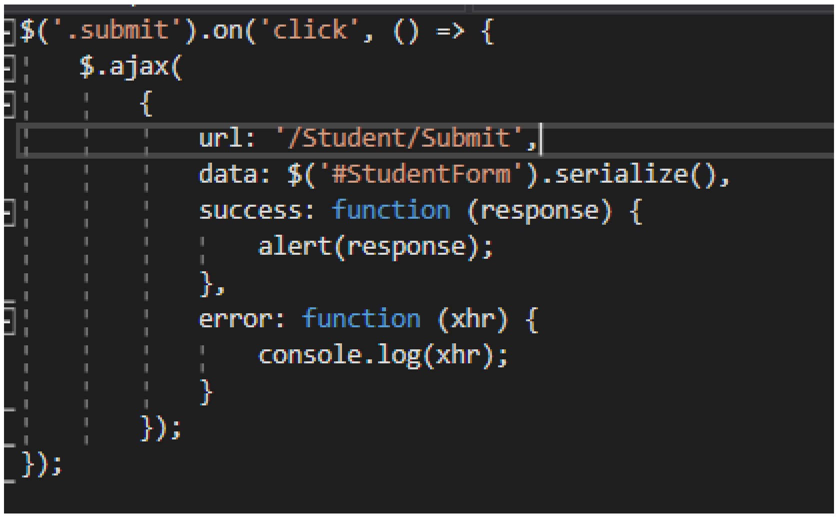
Submit

6) Let us decorate the StudentViewModel with DataAnnotations provided by MVC.



```
5 references
public class StudentViewModel
{
    [Required(ErrorMessage = "First Name can't be empty.")]
    2 references
    public string FirstName { get; set; }

    [Required(ErrorMessage = "Last Name can't be empty.")]
    2 references
    public string LastName { get; set; }

    [Required(ErrorMessage = "Email Address can't be empty.")]
    [EmailAddress(ErrorMessage = "Please enter proper email address.")]
    2 references
    public string EmailAddress { get; set; }

    [Required(ErrorMessage = "Date of Birth Name can't be empty.")]
    2 references
    public DateTime? DateOfBirth { get; set; }

    [Required(ErrorMessage = "Please select a course.")]
    2 references
    public int? Course { get; set; }
    2 references
    public List<SelectListItem> Courses { get; set; }
}
```

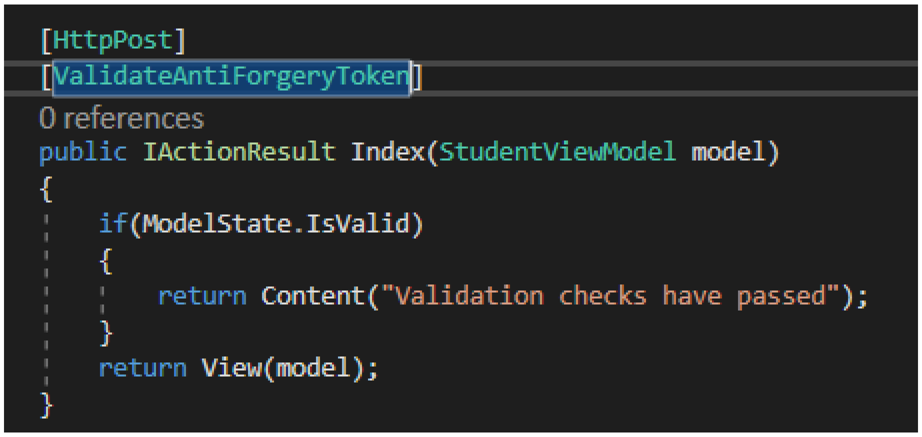
7) Let us create the AJAX code and set the skeleton for submitting the Student Form site.js:



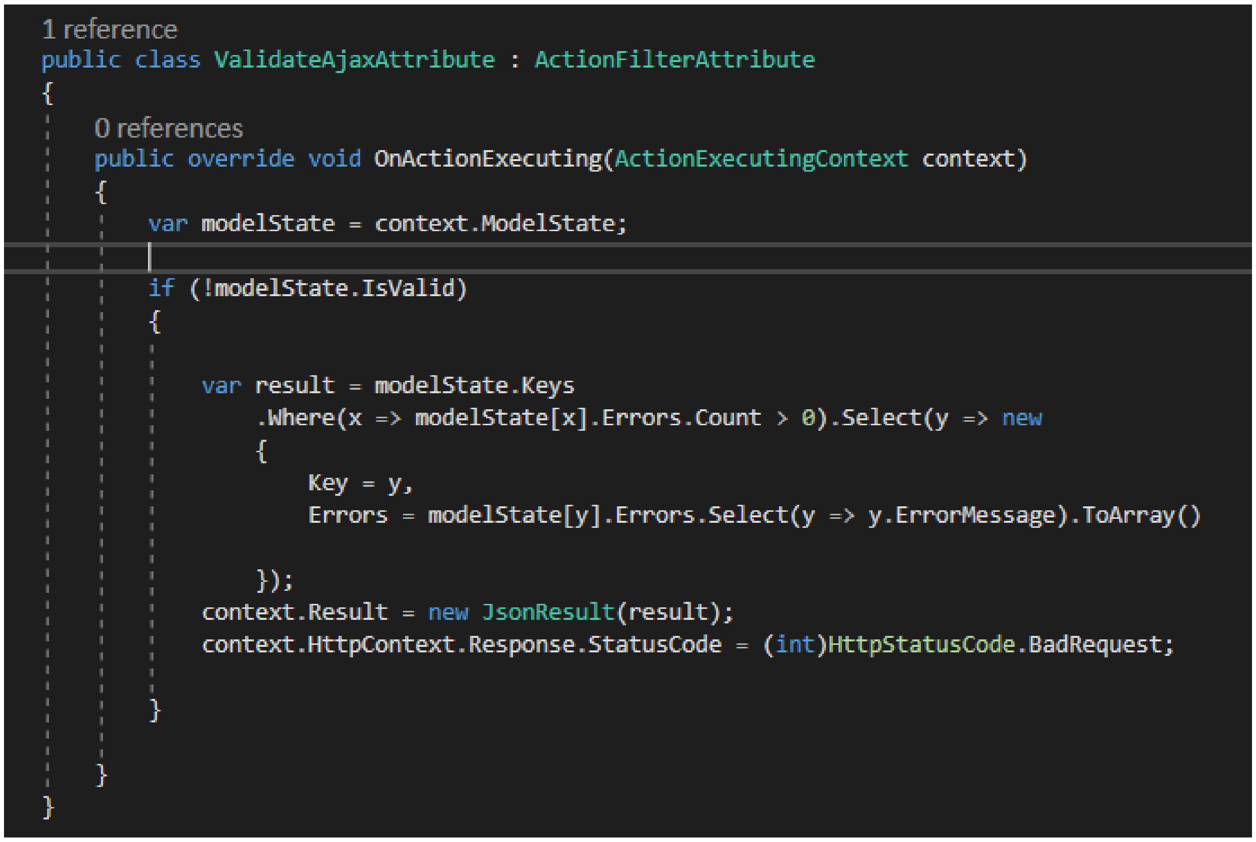
```
$('.submit').on('click', () => {
    $.ajax(
        {
            url: '/Student/Submit',
            data: $('#StudentForm').serialize(),
            success: function (response) {
                alert(response);
            },
            error: function (xhr) {
                console.log(xhr);
            }
        });
});
```

.

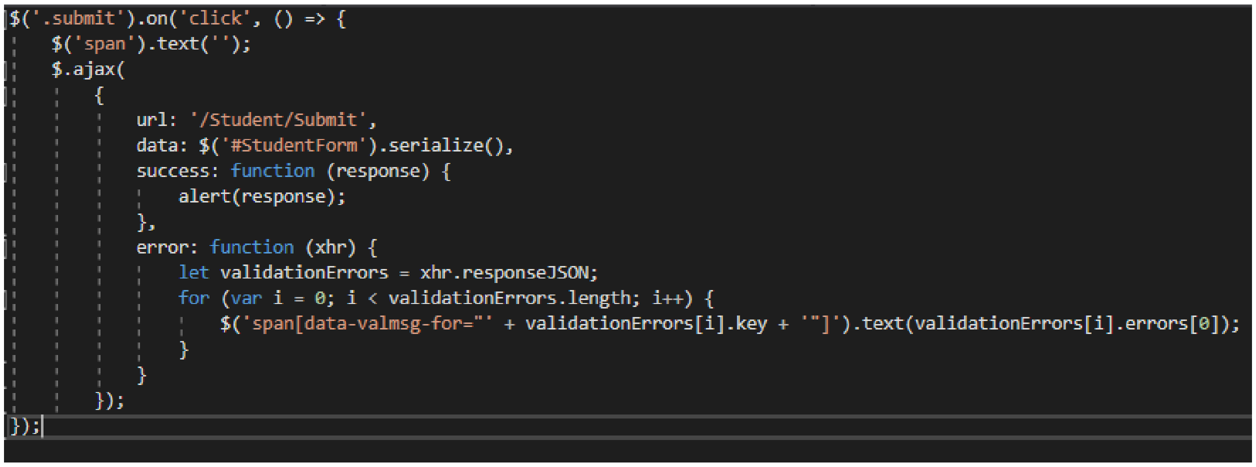
Post Method in StudentController: I have made the method very simple, the onus of this content is to demonstrate validations when submission is via AJAX.


```
[HttpPost]
[ValidateAntiForgeryToken]
0 references
public IActionResult Index(StudentViewModel model)
{
    if(ModelState.IsValid)
    {
        return Content("Validation checks have passed");
    }
    return View(model);
}
```

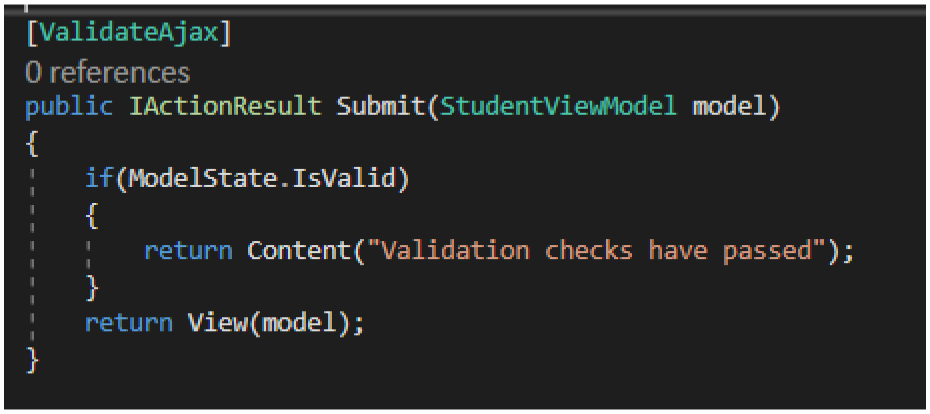
8) Let us create an attribute at server end, which will examine the ModelState errors, and return them so that we can display them for the inputs which have failed the validation check.
The MVC engine would have thrown/displayed the validation messages by itself if we were using button/input type=”submit”.
But since we are using AJAX to post the form (type=”button”), we would have to rely on custom code to return a model with errors, grab this model in the error callback of AJAx function and write some logic to display the validation errors.


```
1 reference
public class ValidateAjaxAttribute : ActionFilterAttribute
{
    0 references
    public override void OnActionExecuting(ActionExecutingContext context)
    {
        var modelState = context.ModelState;

        if (!modelState.IsValid)
        {

            var result = modelState.Keys
                .Where(x => modelState[x].Errors.Count > 0).Select(y => new
                {
                    Key = y,
                    Errors = modelState[y].Errors.Select(y => y.ErrorMessage).ToArray()

                });
            context.Result = new JsonResult(result);
            context.HttpContext.Response.StatusCode = (int)HttpStatusCode.BadRequest;

        }

    }
}
```

9) Let us now grab this model in the error callback of the ajax call.



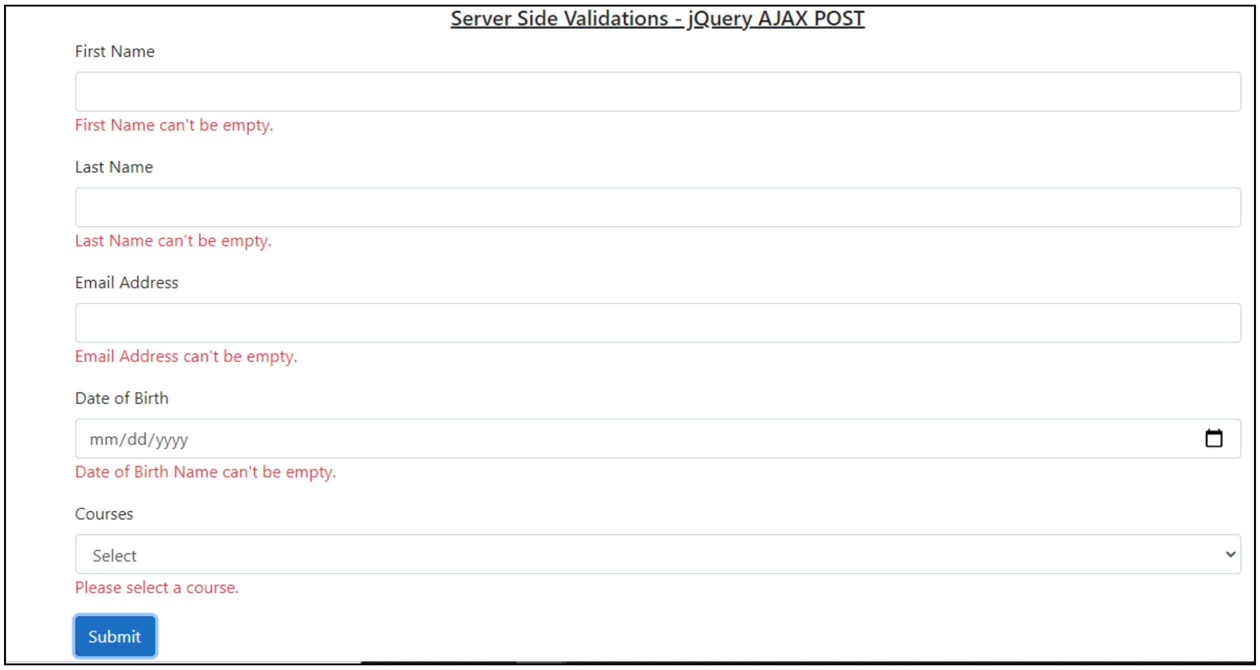
```
$('.submit').on('click', () => {
    $('span').text('');
    $.ajax(
        {
            url: '/Student/Submit',
            data: $('#StudentForm').serialize(),
            success: function (response) {
                alert(response);
            },
            error: function (xhr) {
                let validationErrors = xhr.responseJSON;
                for (var i = 0; i < validationErrors.length; i++) {
                    $('span[data-valmsg-for="' + validationErrors[i].key + '"]').text(validationErrors[i].errors[0]);
                }
            }
        });
});
```



```
[ValidateAjax]
0 references
public IActionResult Submit(StudentViewModel model)
{
    if(ModelState.IsValid)
    {
        return Content("Validation checks have passed");
    }
    return View(model);
}
```

10) This is how the flow works :- The ajax call serializes the form and tries to hit the url '/Student/Submit', it encounters the ValidateAjax attribute over there and tries to weigh in whether the ModelState is valid or not - if its not valid, the validation errors are combined and returned back to the AJAX call. In the error callback of the AJAX code, we find the span tags as per the input element name, and populate the text property of the span with the validation error received.

11) Student form with all the validation errors:



Server Side Validations - jQuery AJAX POST

First Name

First Name can't be empty.

Last Name

Last Name can't be empty.

Email Address

Email Address can't be empty.

Date of Birth

mm/dd/yyyy

Date of Birth Name can't be empty.

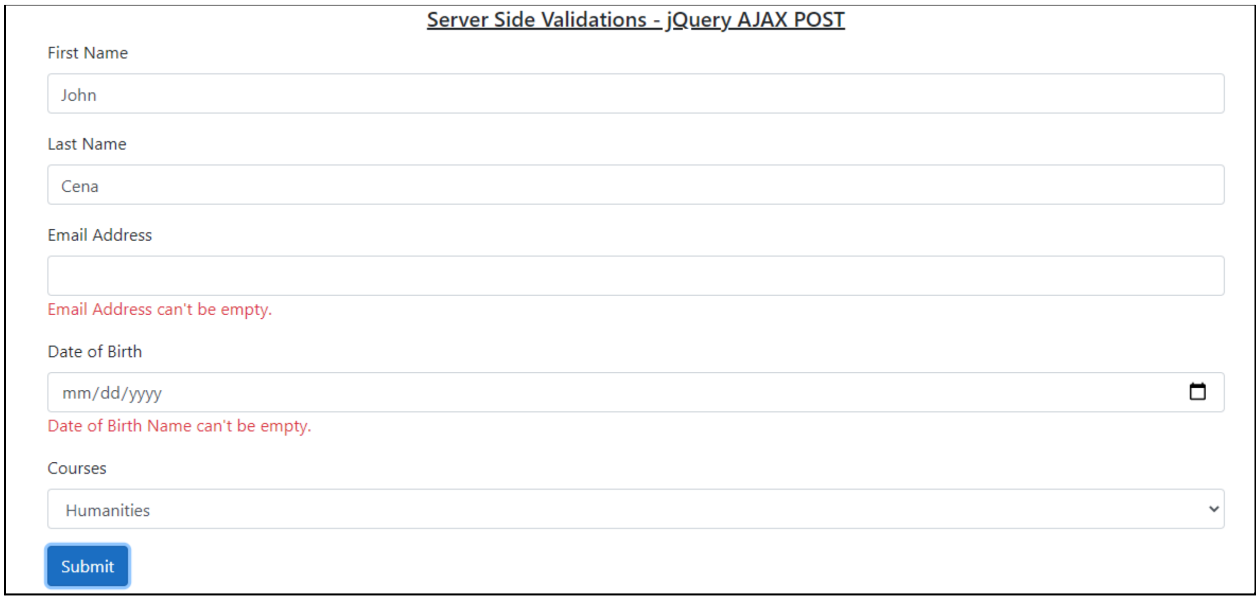
Courses

Select

Please select a course.

Submit

12) Student Form with few validation errors.



Server Side Validations - jQuery AJAX POST

First Name

John

Last Name

Cena

Email Address

Email Address can't be empty.

Date of Birth

mm/dd/yyyy

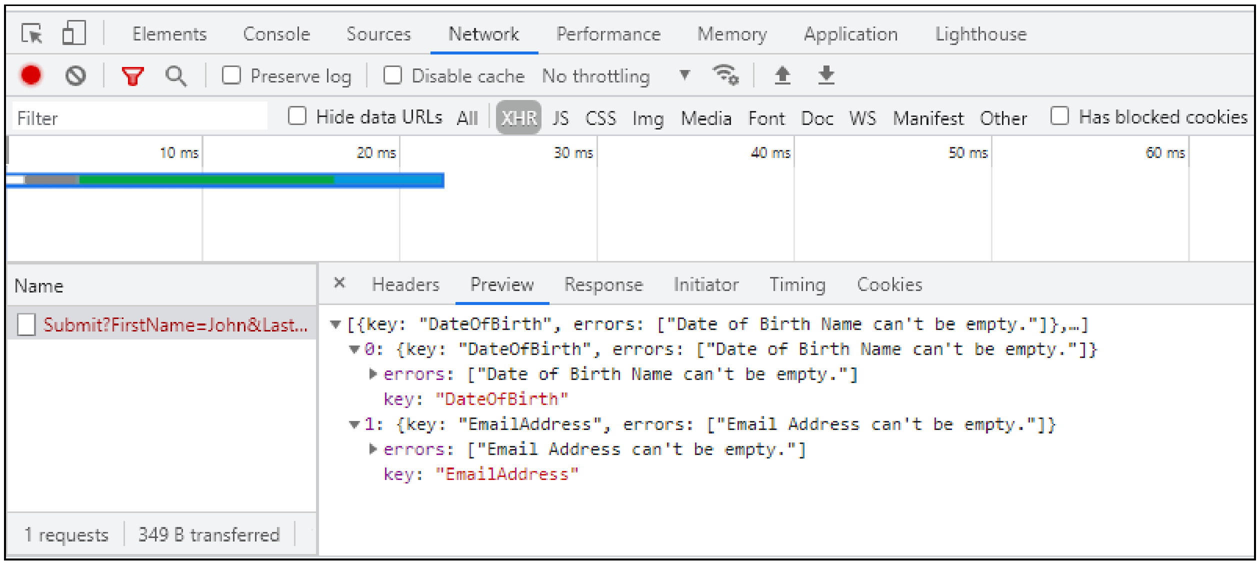
Date of Birth Name can't be empty.

Courses

Humanities

Submit

13) Validation error messages as inspected in the browser's console

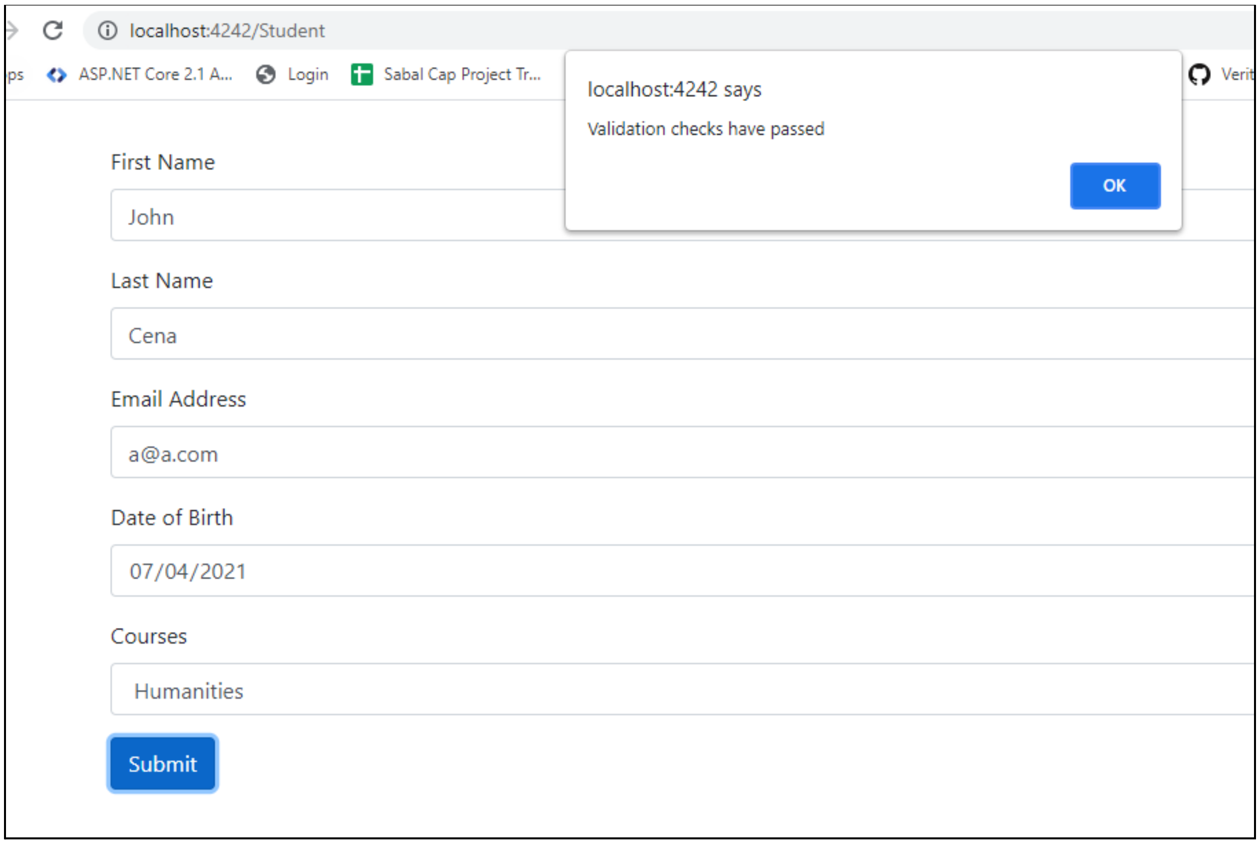


Submit?FirstName=John&Last...

```
▼[{key: "DateOfBirth", errors: ["Date of Birth Name can't be empty."]},…]
  ▼0: {key: "DateOfBirth", errors: ["Date of Birth Name can't be empty."]}
    ▶errors: ["Date of Birth Name can't be empty."]
     key: "DateOfBirth"
  ▼1: {key: "EmailAddress", errors: ["Email Address can't be empty."]}
    ▶errors: ["Email Address can't be empty."]
     key: "EmailAddress"
```

1 requests | 349 B transferred

14) Student Form with all validation checks passed.


localhost:4242/Student

localhost:4242 says

Validation checks have passed

First Name	John
Last Name	Cena
Email Address	a@a.com
Date of Birth	07/04/2021
Courses	Humanities

The full code can be found out at:
https://github.com/anurag1302/server-side-validation-jquery-ajax

Screencast of the solution can be found out at:
https://www.youtube.com/watch?v=bIS4nqdOzdo