

FIT5032 - Internet Applications Development

WEEK 06B - REMOTE ATTRIBUTE & FORM COLLECTION

Last updated: 14th July 2018

Author: Jian Liew

Housekeeping

Before we begin, it is highly recommended for you to use your own personal computer for this subject. If you are planning to use the Monash computers, it is highly recommended to save all your work properly. Towards the end of the semester, there will be a portfolio submission where you need to showcase all the work you have done so far. To prevent difficulties during this portfolio submission, it is suggested that you keep all your work neat and organised based on a **week by week structure**.

Tutorial Structure

The tutorials in this unit are designed to be of a **self-paced and self-taught structure**. So, the aim of your tutor is to aid you in your learning experience. He or she will not go through all the materials that are covered in the labs and will **not do the exercise one by one as a class**. If help is needed, you can either post on the Moodle forums or you can email your tutor asking for clarification. This is slightly different in comparison to other units, where your tutor will lead the discussions. **Please take note of this, you will only be provided help if you make it known that you need help.**

There is no need to complete this tutorial. It functions as a supplementary material to showcase how you can use the remote attribute to perform remote validation. It will also showcase how to use FormCollection instead of ModelBinding.

Objectives

Estimated Time To Complete - 1 hours

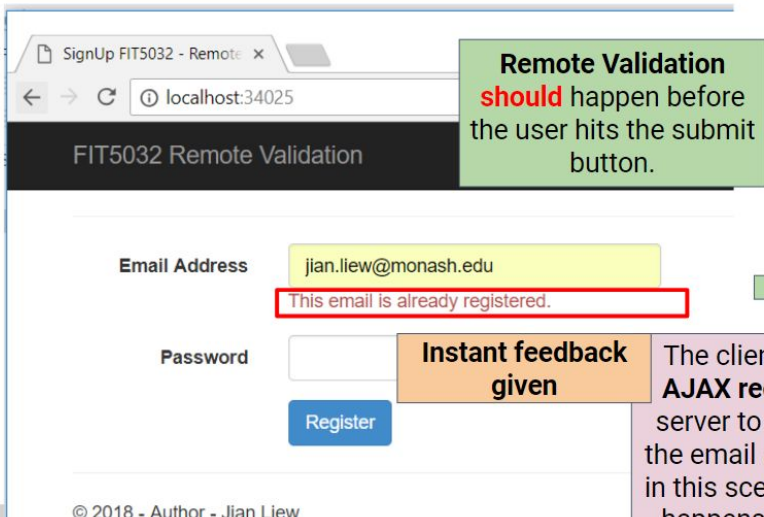
Upon the completion of this tutorial, you will gain a basic understanding of

- How to use Remote attribute
- **Using Form Collection instead of Model Binding**
- Creating a dropdown list and binding it to the model

Doubtfire Submission

- **None**

Remote Validation



Remote Validation
 should happen before the user hits the submit button.

Email Address: jian.liew@monash.edu
 This email is already registered.
 Password:
 Register

Instant feedback given

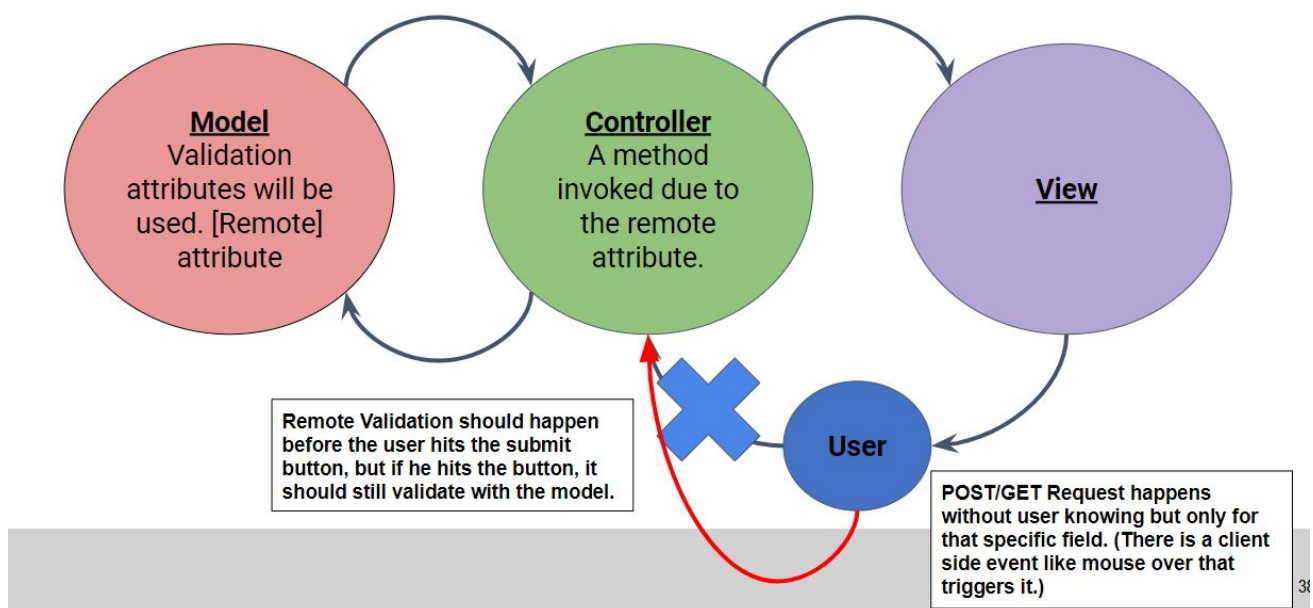
The client makes an **AJAX request** to the server to determine if the email address exist in this scenario. All this happens without the users' knowledge.

Server Side
 The method in the server side will accept either a POST or GET request and return the status.

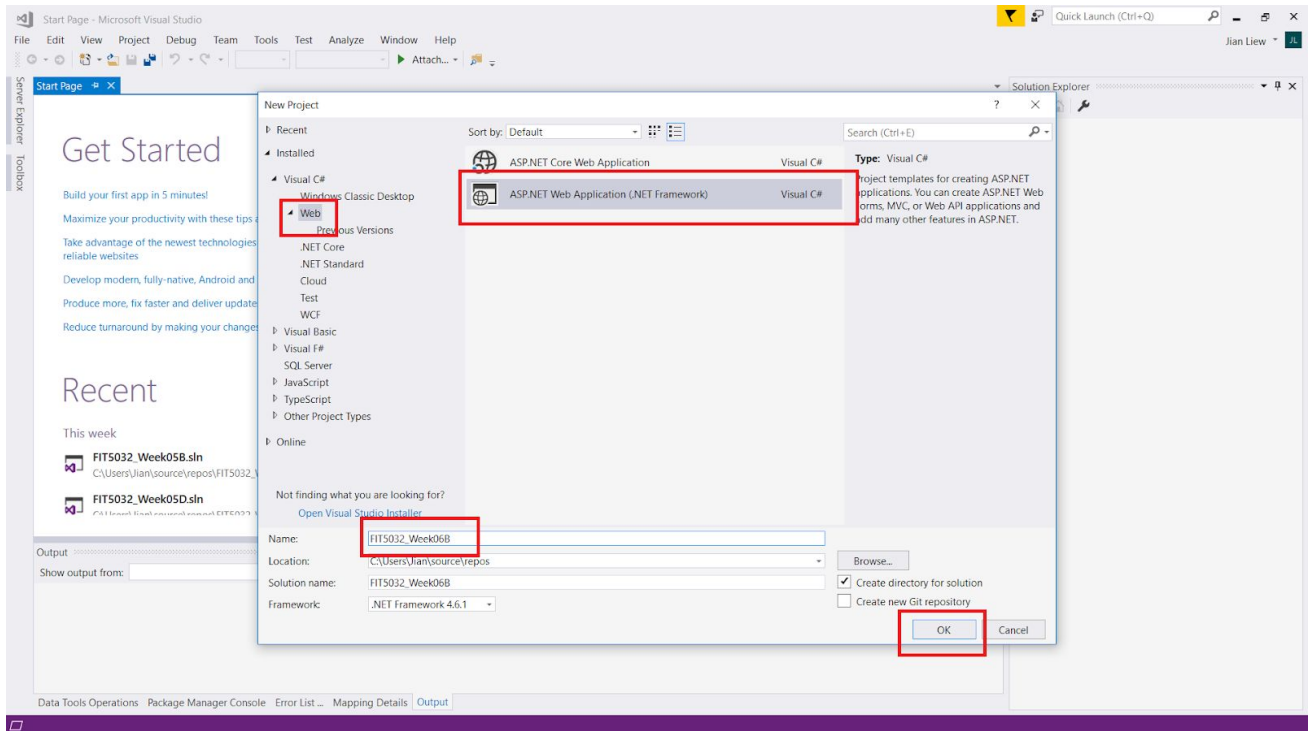
© 2018 - Author - Jian Liew

MONASH University 37

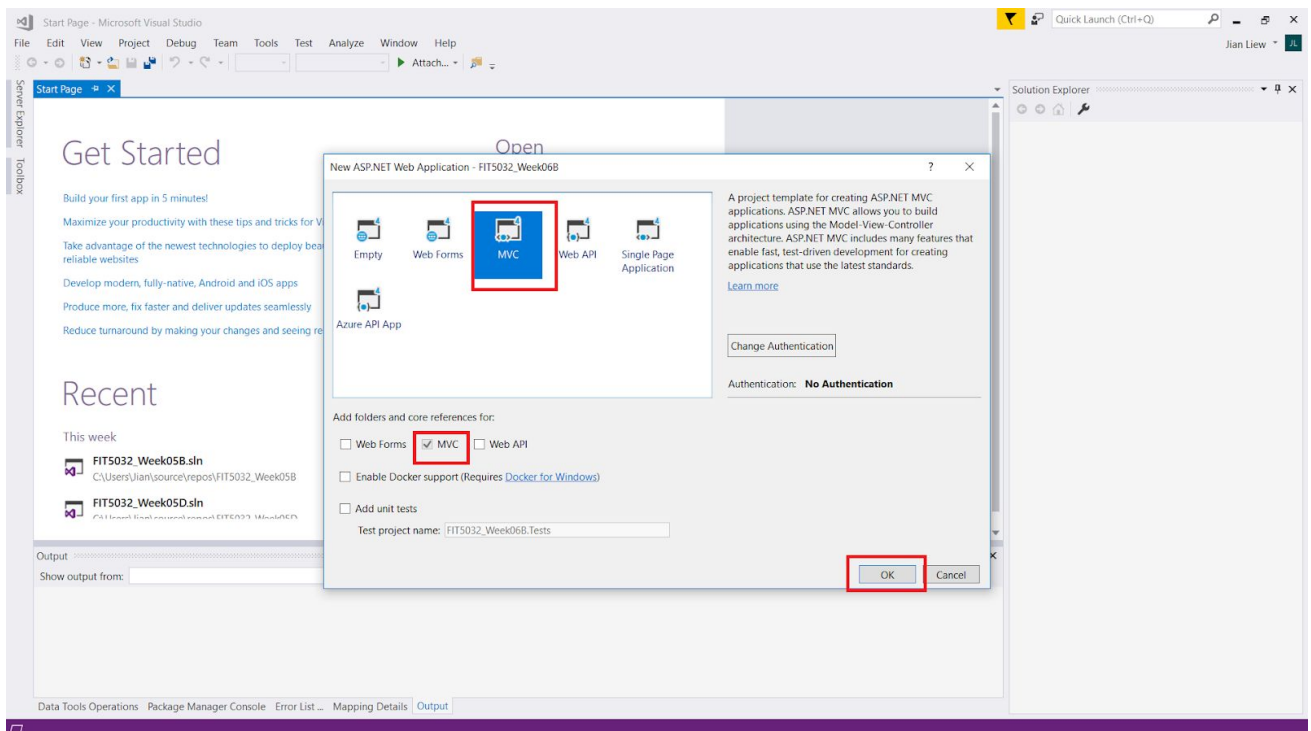
One of the key features of using the remote validation is that **you do not actually see your browser do a POST or GET request**. In other words, you won't see your browser loading. This is because remote validation **post only certain data to the server** instead of the entire form. (In other words, you won't see your browser load or refresh)



Step 1

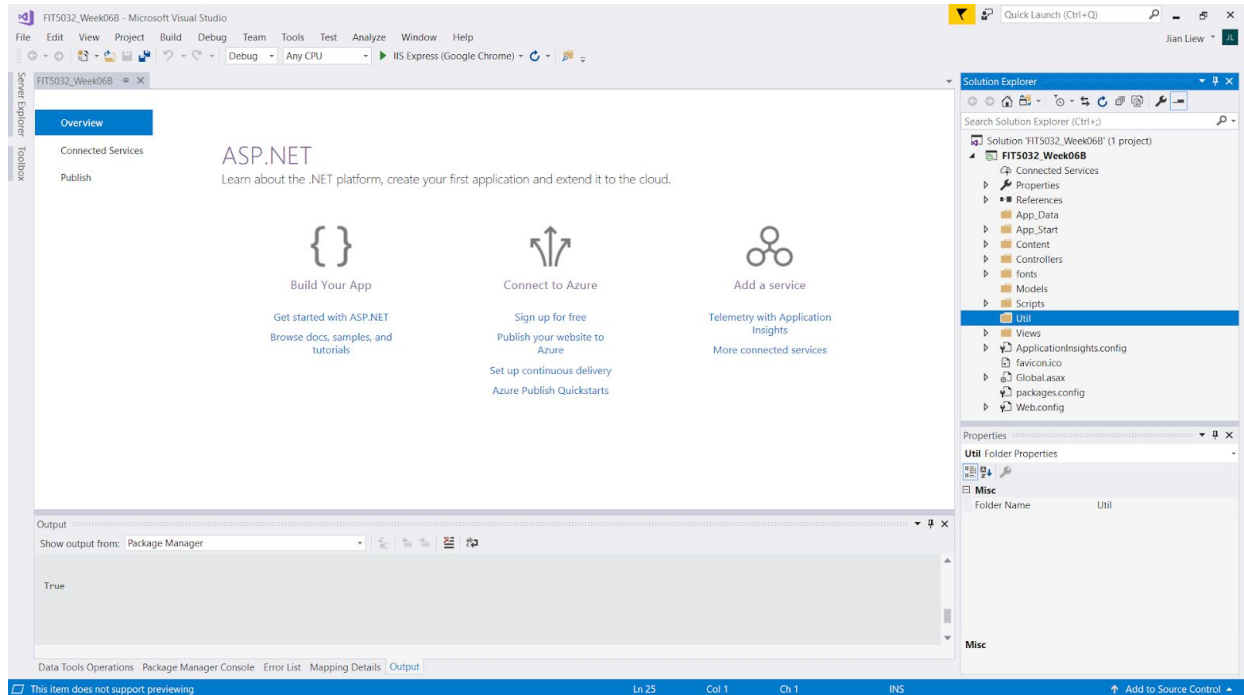


Step 2



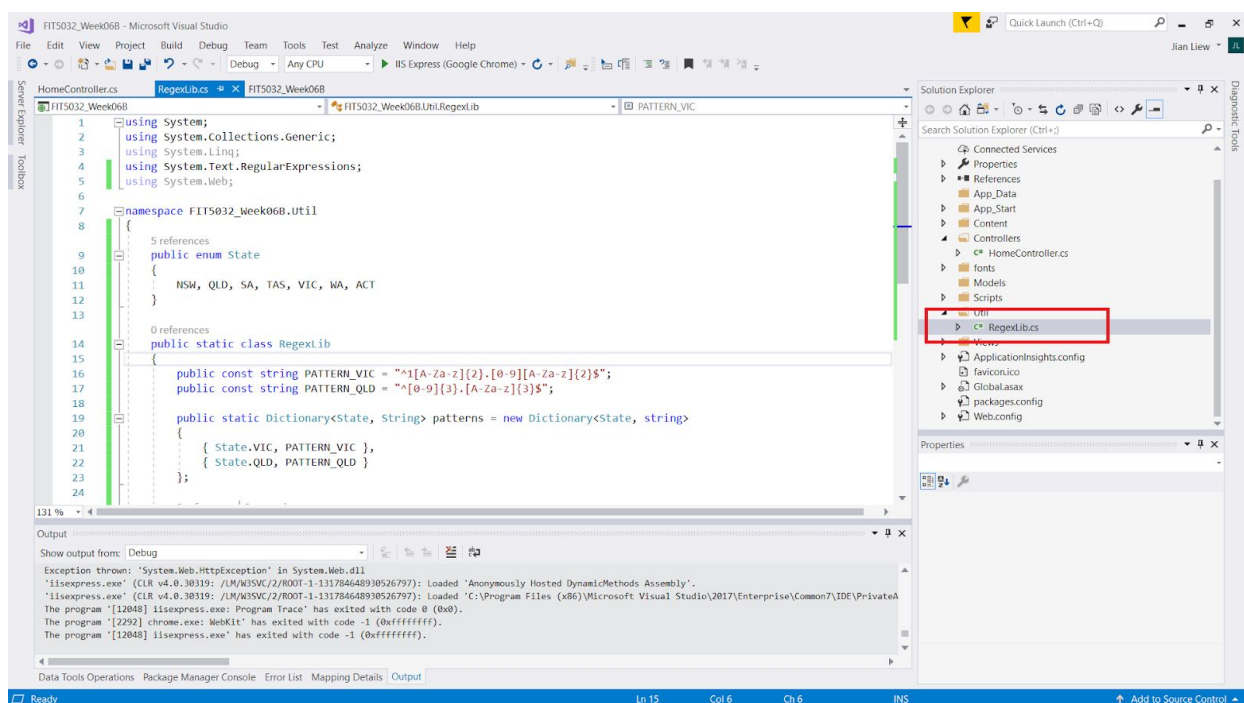
Step 3

Create a new Folder called "Util"



Step 4

Then create a .cs called **RegexLib.cs**



Step 5

The codes for RegexLib.cs are given as follows.

```
using System;
using System.Collections.Generic;
using System.Text.RegularExpressions;

namespace FIT5032_Week06B.Util
{
    public enum State
    {
        NSW, QLD, SA, TAS, VIC, WA, ACT
    }

    public class RegexLib
    {
        public const string PATTERN_VIC = "^1[A-Za-z]{2}[0-9][A-Za-z]{2}$";
        public const string PATTERN_QLD = "^1[0-9]{3}[A-Za-z]{3}$";

        public static Dictionary<State, String> patterns = new Dictionary<State, string>
        {
            { State.VIC, PATTERN_VIC },
            { State.QLD, PATTERN_QLD }
        };

        public static bool IsValidCarPlate(string patternToValidate, State state)
        {
            patterns.TryGetValue(state, out string pattern);
            if (null != pattern)
                return Regex.IsMatch(patternToValidate, pattern);
            return false;
        }
    }
}
```

Step 6

Create a ViewModel called CarPlateViewModel.cs. The codes are given as follows.

```
using FIT5032_Week06B.Util;
using System;
using System.ComponentModel.DataAnnotations;
using System.Web.Mvc;

namespace FIT5032_Week06B.Models
{
    public class CarPlateViewModel
    {
        public State State { get; set; }

        [Required]
        [MaxLength(10)]
        [Remote("CheckCarPlate", "CarPlate",
            AdditionalFields = "State",
            HttpMethod = "POST",
            ErrorMessage = "Invalid Car Plate")]
        public String CarPlate { get; set; }
    }
}
```

Step 7

Create a Folder called CarPlate under Views and create a Index.cshml. The codes are as follows

```
@model FIT5032_Week06B.Models.CarPlateViewModel
@{
    ViewBag.Title = "CarPlate";
    Layout = "~/Views/Shared/_Layout.cshtml";
}
<h2>CarPlate</h2>
@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()
    <div class="form-horizontal">
        <h4>CarPlateViewModel</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })

        <div class="form-group">
            @Html.LabelFor(model => model.State, htmlAttributes: new { @class = "control-label col-md-2" })
            <div class="col-md-10">
                @Html.DropDownListFor(model => model.State, new SelectList(new string[] { "ACT",
                "VIC", "ACT" }, "VIC"), htmlAttributes: new { @class = "form-control" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.CarPlate, htmlAttributes: new { @class =
            "control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.CarPlate, new { htmlAttributes = new { @class =
                "form-control" } })
                @Html.ValidationMessageFor(model => model.CarPlate, "", new { @class =
                "text-danger" })
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Submit" class="btn btn-default" />
            </div>
        </div>
    </div>
}

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Step 8

Create Controller called CarPlateController.cs

```
using FIT5032_Week06B.Util;
using System;
using System.Web.Mvc;

namespace FIT5032_Week06B.Controllers
{
    public class CarPlateController : Controller
    {
        // GET: CarPlate
        public ActionResult Index()
        {
            return View();
        }

        // POST: CarPlate
        [HttpPost]
        public ActionResult Index(FormCollection formCollection)
        {
            try
            {
                // TODO: You can obtain values with form collections instead of model too.
                String state = formCollection["state"];
                String carPlate = formCollection["carPlate"];
                // Your logic here.

                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

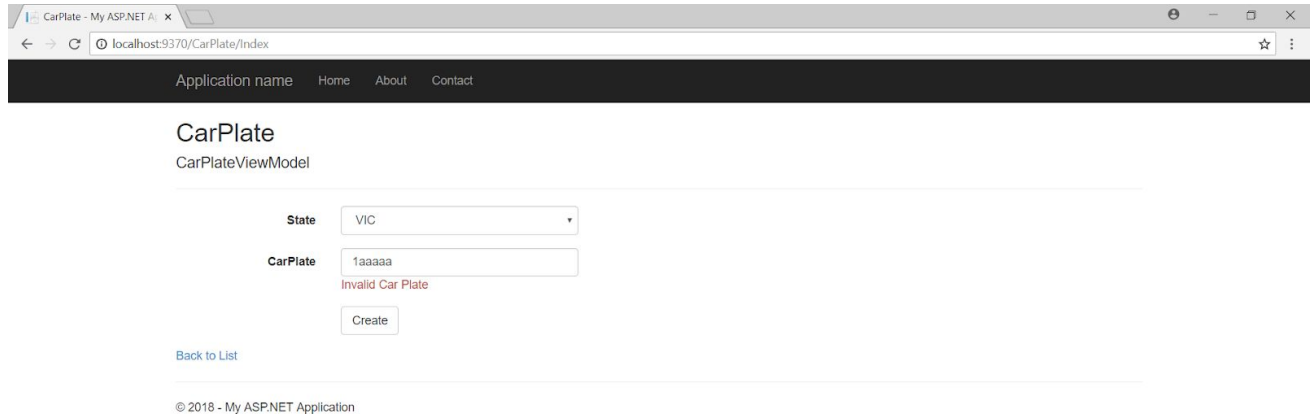
        [HttpPost]
        public ActionResult CheckCarPlate(string carPlate, State state)
        {
            try
            {
                return Json(RegexLib.IsValidCarPlate(carPlate, state));
            }
            catch (Exception ex)
            {
                return Json(false);
            }
        }
    }
}
```


Step 9

You can then add a link at the shared layout.

Step 10

You can then test it out.



The screenshot shows a web browser window with the address bar displaying 'localhost:9370/CarPlate/Index'. The page has a dark navigation bar with links for 'Application name', 'Home', 'About', and 'Contact'. Below this, the title 'CarPlate' is displayed, followed by 'CarPlateViewModel'. The form contains a 'State' dropdown menu set to 'VIC' and a 'CarPlate' text input field containing '1aaaaa'. A red error message 'Invalid Car Plate' is visible below the input field. A 'Create' button is positioned below the input field. A 'Back to List' link is located at the bottom left of the form area. The footer of the page reads '© 2018 - My ASP.NET Application'.

Explanation

- 1) The main objective of having remote validation is to **increase user experience**. This is a simple example of what could be done. Notice that if you **do not need to click the "Create"** button to trigger a validation.
- 2) A better use case for this, is for example, when you would like to check if an item is still "In stock". (So, if you want to do this, you must call the model layer for this information)
- 3) This example here, also shows you that you do not always need to use model binding if you do not want. At the end of the day, the values are still obtained when the form is posted, so you can actually use a form collection instead. You can put a breakpoint if you want to see the values of the FormCollection.
- 4) This tutorial also shows you an easy way to use a dropdownlist (select box) for your enum values that you might use.
- 5) It is also important to remember that enums in C# do not accept String
- 6) **The example, given is a very simple example which has flaws, can you notice where the flaw is?**

Conclusion

This supplementary material showcases the remote validation. Remote validation is often times used to increase usability of a web application.

It also showcases that, it is not needed to always bind a model when you are submitting a form as you can use a `FormCollection` instead.