Exercise 10 : Validation

Objective

In this exercise you will apply data annotation attributes to provide validation for your model data. You will then use html helpers to provide validation assistance in the views. Next you will protect your application from script injection attacks. Finally, you will implement exception handling in your application.

This exercise will take around 20 minutes.

Referenced material

This exercise is based on material from the chapter "Validation".

Actions

1	Open the 'Begin' solution in Visual Studio and compile (Shift Ctrl+B). The 'Begin' solution is identical to the 'End' solution from the previous lab.

2	In the Solution Explorer, locate and open the Models/Post.cs file, and have a look at the code. You created this class in an earlier exercise, and it is used in association with Entity Framework Code First to provide Post entities to the application:
	`public class Post`
	`{`
	`    public int PostId { get; set; }`
	`    public int ThreadId { get; set; }`
	`    public string Title { get; set; }`
	`    public string UserName { get; set; }`
	
	`    public string PostBody { get; set; }`
	
	`    [Column(TypeName = "smalldatetime")]`
	`    public DateTime PostDateTime { get; set; }`
	
	`    // Navigation Properties`
	
	`    public virtual Thread Thread { get; set; }`
	`}`
	As things stand, neither ASP.NET MVC nor Entity Framework have any way of knowing what the valid values would be for an entity of this type

3	Test the application to highlight the validation problem: Press F5 to build and run the project. Browse to the Posts page and click Create New at the top of the page. Select a Thread from the drop down list, but leave the Post Title, Post Body and User Name empty. Enter a sensible date (something from this year will be fine) and click Create. The form submits, allowing a blank post into the system. The post has no title, body or user, and is useless. We will fix these issues by applying data annotations to the model.

4	Add the following attributes to the Post model:
	`public class Post`
	`{`
	`    public int PostId { get; set; }`

```
        public int ThreadId { get; set; }

        [Required]
        [StringLength(100)]
        public string Title { get; set; }

        [Required]
        public string UserName { get; set; }

        [Required]
        public string PostBody { get; set; }

        [Column(TypeName = "smalldatetime")]
        public DateTime PostDateTime { get; set; }

        // Navigation Properties

        public virtual Thread Thread { get; set; }
    }
```

5 Any time we change our model, we must create a migration and apply it to the database.
Go into the Nuget Package Manager Console, and enter the following command:

```
add-migration AddValidationAttributes -context ForumDbContext
```

6 To apply our migration, we are going to run the SeedDatabase program from an earlier lab. Open SeedDatabase/Program.cs, and look at the lines highlighted in blue below:

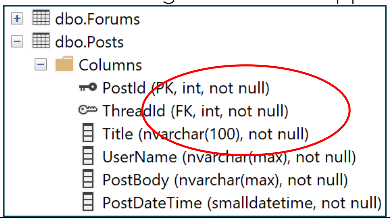
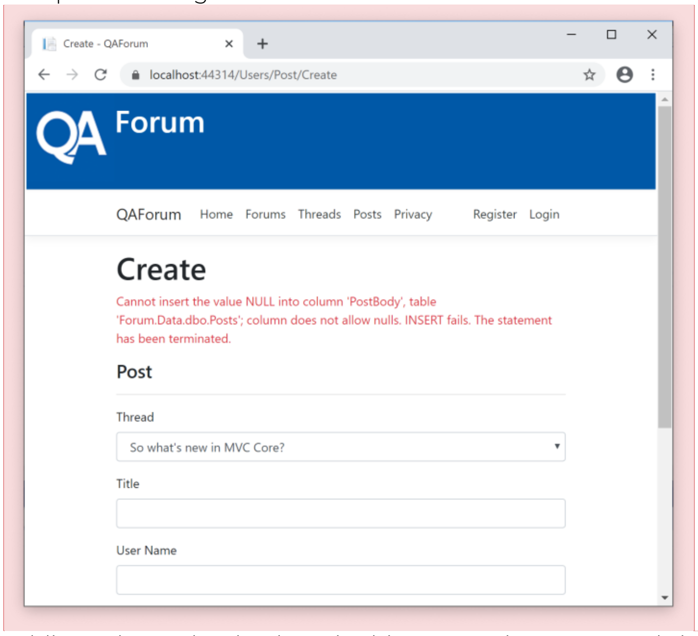
```
static void Main(string[] args)
{
    ....
    using(ForumDbContext ctx = new ForumDbContext(optionsBuilder.Options))
    {
        ctx.Database.EnsureDeleted();
        ctx.Database.Migrate();
        InitForums(ctx);
    }
}
```

Because of these two lines, when we run the SeedDatabase program, the existing database (and all the data in it) will be deleted, and then the initial set of data will be re-added.

Run the program now. Set it to be the startup program, press F5, and when it finishes, set the startup program back to QAForum.

(Note that, in a real-life situation, you would be unlikely to want to delete your data every time you apply a migration. There are two options to prevent this happening. Either comment out the two lines highlighted in blue above, or run the `update-database` command from the package manager console instead of running SeedData. In either case, applying a migration which disallows NULLs, with NULLs in the database as we currently have, will cause errors, and the NULLs will need to be replaced with an appropriate value first. We are choosing to delete the database here to avoid having to worry about this.)

7 Use the Object Explorer in SQL Server Management Studio to investigate the columns in the Posts table in the database. Notice how all the columns now have

	“not null” shown next to them. Additionally, the Title column has had the maximum length of the 100 applied:  dbo.Forums dbo.Posts Columns PostId (PK, int, not null) ThreadId (FK, int, not null) Title (nvarchar(100), not null) UserName (nvarchar(max), not null) PostBody (nvarchar(max), not null) PostDateTime (smalldatetime, not null)
8	Run the program again, and try to add a post in the same manner as earlier, with no title, body or username. This time, the post does not save. Instead, an exception occurs when we try to add the post into the database, and the exception message is shown on the screen:  Create - QAForum localhost:44314/Users/Post/Create QA Forum QAForum Home Forums Threads Posts Privacy Register Login Create Cannot insert the value NULL into column 'PostBody', table 'Forum.Data.dbo.Posts'; column does not allow nulls. INSERT fails. The statement has been terminated. Post Thread: So what's new in MVC Core? Title: User Name: Adding rules to the database in this manner is an extremely important part of setting up your application. It ensures that there is no way for invalid data (at least, the specific type of invalid data that we have restricted) to get into the database, even if a buggy program were to connect to the database and attempt to save such invalid data. But it’s far from the end of the job. Not only does it result in some error messages being shown that are not particularly user-friendly, it also requires the data to be sent from the web browser to the server, and then on to the database, before validation takes places. Surely we should be trying to validate the data on the client (the web browser), so that errors can be shown without the user having to wait for the data to be sent to the server?

Commented [PB1]: .NET 7.0 version automatically does validation in the browser and doesn't display a high level message but indicates each individual field that is in error

9

Add similar validation attributes to Areas/Users/ViewModels/PostWriteViewModel, as shown here:

```
public class PostWriteViewModel
{
    [Display(Name = "Thread")]
    public int ThreadId { get; set; }

    [Required(AllowEmptyStrings = false,
                    ErrorMessage = "Specify a title for your post")]
    [StringLength(100, ErrorMessage = "Please shorten your post's title")]
    public string Title { get; set; }

    [Required(AllowEmptyStrings = false,
        ErrorMessage = "Please specify the user name of the post's owner")]
    [Display(Name = "User Name")]
    public string UserName { get; set; }

    [Required(AllowEmptyStrings = false,
                    ErrorMessage = "Specify the body of your post")]
    [MinLength(10,
        ErrorMessage = "You need to write a bit more to make a valid post")]
    [Display(Name = "Post Body")]
    public string PostBody { get; set; }

    [Display(Name = "Date of Post")]
    public DateTime PostDateTime { get; set; }
    ....
```

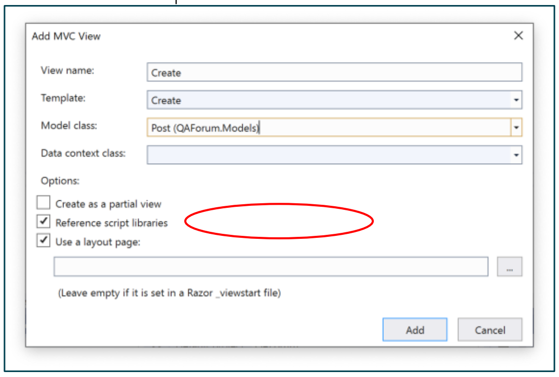
We've applied all the same attributes to the view-model as we applied to the model, to ensure that the view is aware of all the restrictions. We've also added some extra attributes that apply restrictions which can't be applied by SQL Server, and we've added user-friendly messages to each attribute.

10

Open the /Areas/Users/Views/Post/Create view. Scroll to the bottom, and notice these lines:

```
@section Scripts {
    @{await Html.RenderPartialAsync("_ValidationScriptsPartial");}
}
```

These were added when we created the view, but only because we ticked the "Reference Script Libraries" checkbox:


Add MVC View

View name: Create

Template: Create

Model class: Post (QAForum.Models)

Data context class:

Options:

☐ Create as a partial view

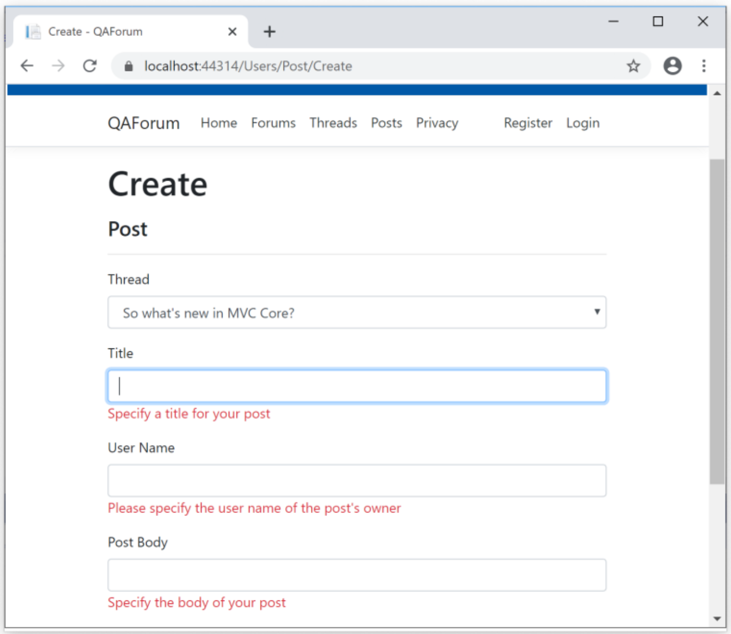
☑ Reference script libraries

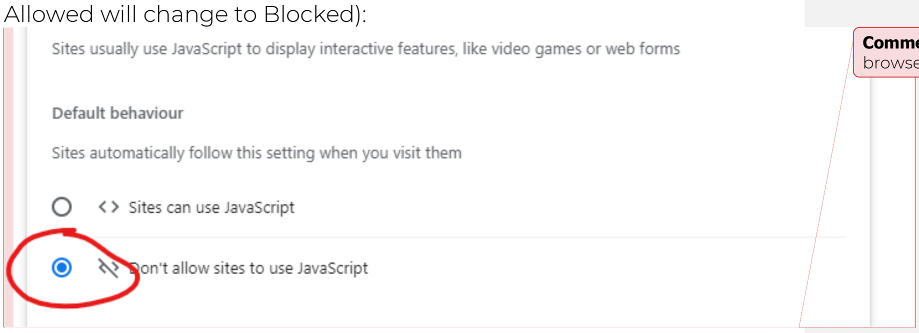
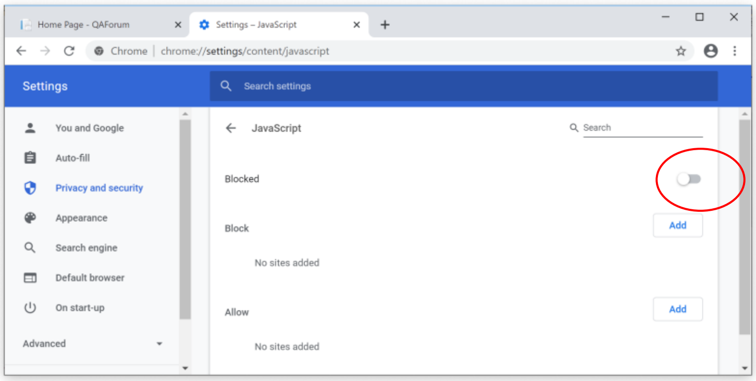
☑ Use a layout page:

(Leave empty if it is set in a Razor _viewstart file)

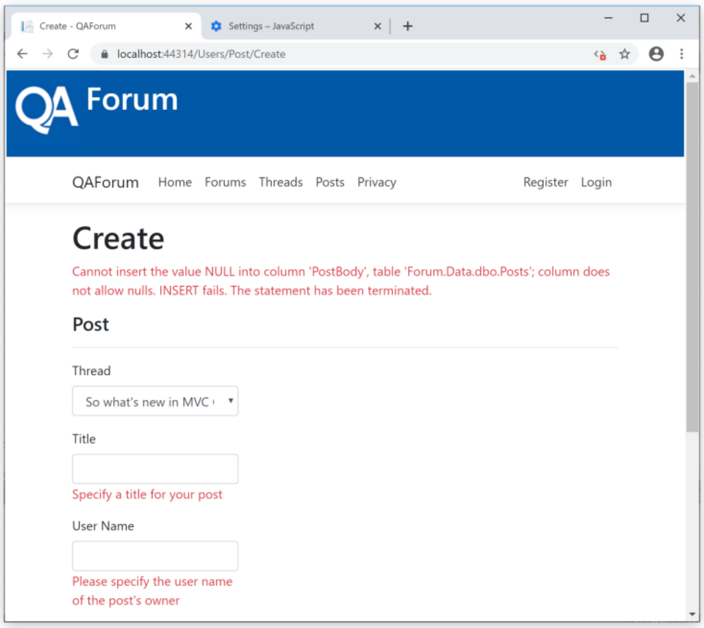
These lines include the JavaScript libraries which are needed to enable client-side validation. If we had created the view without this checkbox, client-side

	validation would not work unless we included the JavaScript libraries using some other technique.
	
	Scroll up a little, and take a look at the code for displaying the data.
	
	Each section has a line similar to this:
	`<div class="form-group">`
	`    <label asp-for="Title" class="control-label"></label>`
	`    <input asp-for="Title" class="form-control" />`
	`    <span asp-validation-for="Title" class="text-danger"></span>`
	`</div>`
	This line is the one responsible for displaying the error message if the user enters invalid data.

11	Once again, run the program and try to add a post with no data. This time, we get a much more user-friendly experience:  Create - QAForum localhost:44314/Users/Post/Create QAForum Home Forums Threads Posts Privacy Register Login Create Post Thread So what's new in MVC Core? Title Specify a title for your post User Name Please specify the user name of the post's owner Post Body Specify the body of your post Type a few letters in the Post Body, and watch the error message change as you type (it should change from "Specify the body of your post" to "You need to write a bit more to make a valid post"). The message changes without you needing to press the Submit button – proof that it must be handled by the client, since the data is not being sent to the server.

	Client-side validation is ideal, because it gives the user immediate feedback. But it relies on JavaScript, which can easily be disabled by users, so it's important that we consider what to do when this happens. With the program running, disable JavaScript in your web browser. In Google Chrome, you can do this going to …/Settings/Privacy and Security/Site Settings/JavaScript, and then sliding the Allowed slider to the left (the word Allowed will change to Blocked):  Sites usually use JavaScript to display interactive features, like video games or web forms Default behaviour Sites automatically follow this setting when you visit them ○ Sites can use JavaScript [✓ Don't allow sites to use JavaScript]  Home Page – QAForum Settings – JavaScript Chrome | chrome://settings/content/javascript Settings Search settings You and Google Auto-fill Privacy and security Appearance Search engine Default browser On start-up Advanced JavaScript Search Blocked Block Add No sites added Allow Add No sites added If you're using a different web browser and you can't find the setting to disable JavaScript, ask your tutor for help.
12	Go back to the Post/Create page (be sure to refresh it if it's already showing), and again try to add a post with no content. Once again, you will see the SQL error message when you try to save the post – proof that the invalid data has been sent to the server. (You also see the same client-side messages as when

Commented [PB2]: UI different in later version of browser (Edge)

	JavaScript is enabled. These are added statically – not by JavaScript – when the view is re-created after the exception is caught.) 

Create - QAForum

Settings – JavaScript

localhost:44314/Users/Post/Create

QAForum Home Forums Threads Posts Privacy Register Login

Create

Cannot insert the value NULL into column 'PostBody', table 'Forum.Data.dbo.Posts'; column does not allow nulls. INSERT fails. The statement has been terminated.

Post

Thread

So what's new in MVC

Title

Specify a title for your post

User Name

Please specify the user name of the post's owner

As you can see, the existing server-side code isn't doing a bad job of handling the invalid data. But that is only because it's trying to save the data to the database. Ideally, the server should check that the data is valid *before* it goes to the database, and only send it to the database if it passes these checks.

This is very easy to achieve using MVC. Add the following code to the HttpPost Create() method in the PostController:

```
public ActionResult Create(PostWriteViewModel viewModel)
{
    try
    {
        if (ModelState.IsValid)
        {
            // TODO: Add insert logic here
            Post post = new Post
            {
                ThreadId = viewModel.ThreadId,
                Title = viewModel.Title,
                UserName = viewModel.UserName,
                PostBody = viewModel.PostBody,
                PostDateTime = viewModel.PostDateTime
            };
            context.Posts.Add(post);
            context.SaveChanges();

            return RedirectToAction(nameof(Index));
        }
        else
        {
            viewModel.ThreadSelectListItems=context.GetThreadsSelectListItems();
            return View(viewModel);
        }
    }
```

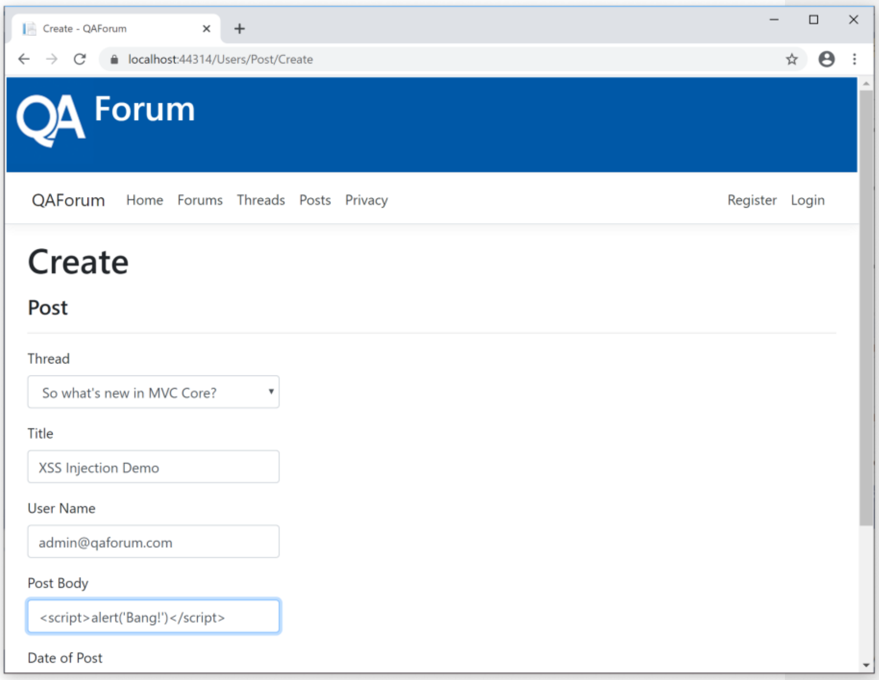
Add the invalid post again, and check that you still see the user-friendly errors from the view-model, but that you no longer see the SQL Server error message. Notice that, if you start typing a post body, the message no longer dynamically updates until you submit the form – there is simply no way that the text on the page can dynamically change with JavaScript disabled.

Have a go at adding similar code to the Edit Post code, and check it works as expected.
Make sure you enable JavaScript again once you finish this step!

Preventing JavaScript injection attacks

Add a post. For the post body, use the following text:

```
<script>alert('Bang')</script>
```



Create - QAForum

localhost:44314/Users/Post/Create

Create

Post

Thread: So what's new in MVC Core?

Title: XSS Injection Demo

User Name: admin@qaforum.com

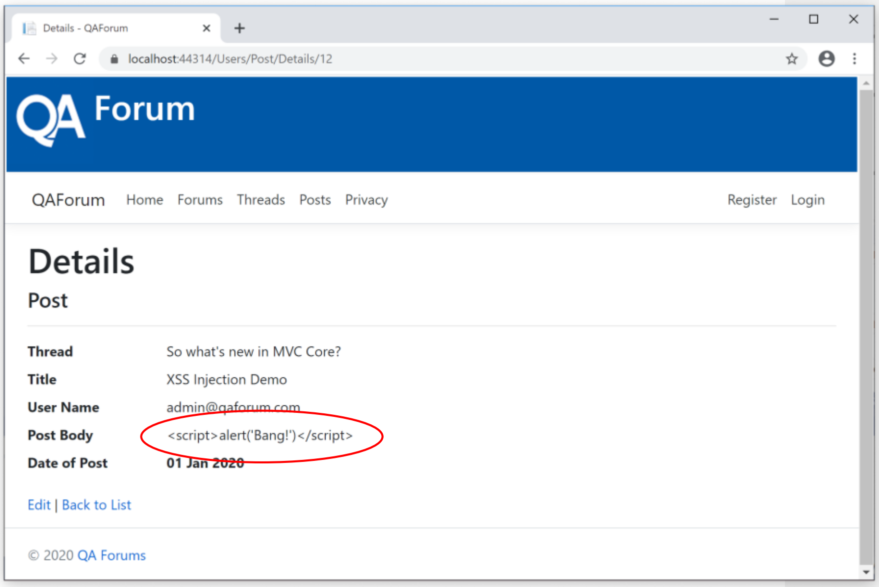
Post Body: `<script>alert('Bang!')</script>`

Date of Post

The text that was used for the post body includes some JavaScript. If users were allowed to enter JavaScript, and then that JavaScript were to be sent to another user's web browser and allowed to run, the web site would be open to a type of attack called "Cross Site Scripting".

Let's see whether MVC allows this to happen.

Go to the Posts screen, and choose the Details option next to your new post. Notice that the JavaScript does not run – instead, it's just shown to the user as shown here:


Details - QAForum
localhost:44314/Users/Post/Details/12
QA Forum
QAForum Home Forums Threads Posts Privacy
Register Login
Edit | Back to List
© 2020 QA Forums

Details

Post

Thread	So what's new in MVC Core?
Title	XSS Injection Demo
User Name	admin@qaforum.com
Post Body	`<script>alert('Bang!')</script>`
Date of Post	01 Jan 2020

It's vital that any untrusted data (i.e. data that is entered by a user) is "encoded" before it is displayed in a user's web browser. This ensures that scripts in data will be displayed, but not run. As you can see, this happens by default in MVC – no special action is required by the programmer to protect their website.

You can read more about this important topic here: https://docs.microsoft.com/en-us/aspnet/core/security/cross-site-scripting?view=aspnetcore-3.1