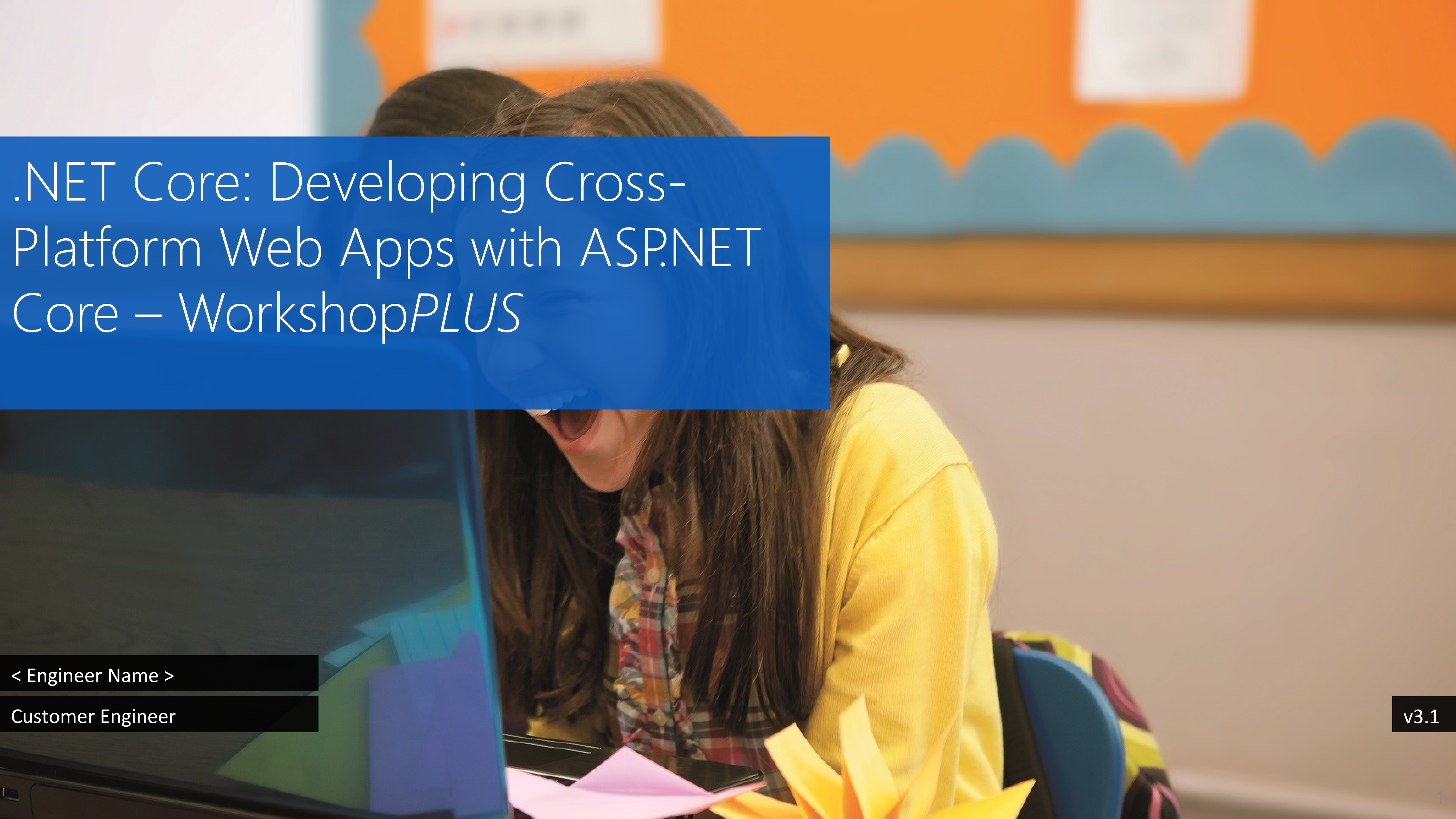


.NET Core: Developing Cross-Platform Web Apps with ASP.NET Core – Workshop*PLUS*

< Engineer Name >

Customer Engineer

v3.1

Module 7: Validation

Module Overview

Module 7: Validation

Section 1: Validation Fundamentals

Lesson: Overview

What is Validation?

Validation

- Validating user inputs and enforcing business rules/logic is a core requirement of most web applications
- Server-side validation
  - Should be done with or without client-side validation
  - Model Validation with Data Annotations
- Client-side validation
  - Unobtrusive validation
  - Extending
  - Remote
- “Don’t Repeat Yourself”

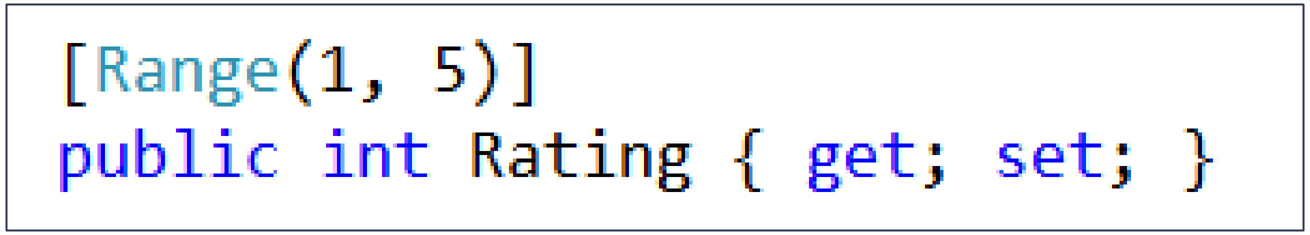
Data Annotations

- The attribute is declared on the server-side property via metadata
- Built-in validation attributes

Attribute	Description
CompareAttribute	Compares the value of two model properties. Validation succeeds if they are equal
RemoteAttribute	Leverages jQuery Validate to call an action on the server to perform server-side validation with AJAX
RequiredAttribute	Indicates that a value is required
RangeAttribute	Indicates the numeric range constraints for the field value
RegularExpressionAttribute	A data field value must match the specified
StringLengthAttribute	Specifies the maximum string length

Range Attribute

Example: Range Attribute in Model Metadata



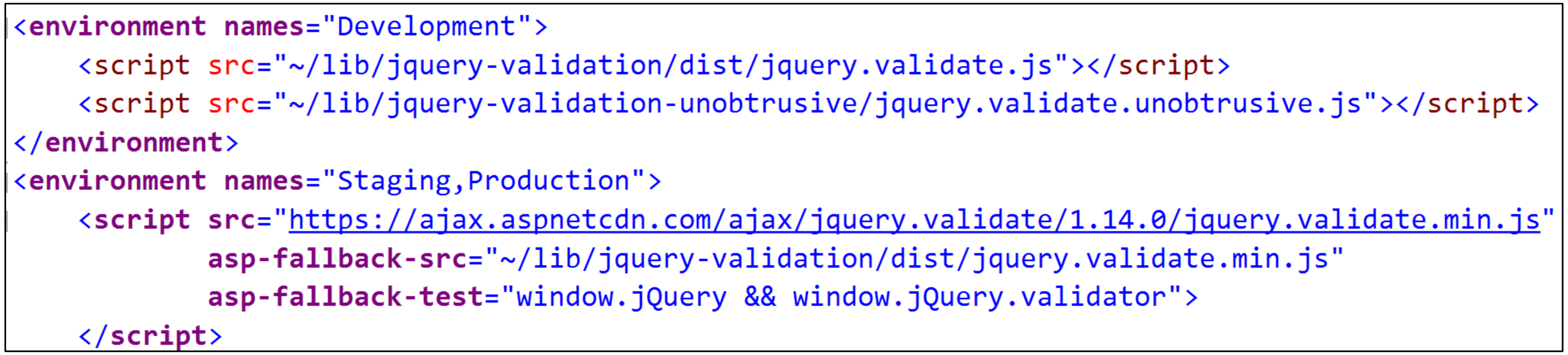
```
[Range(1, 5)]
public int Rating { get; set; }
```

Range Attribute Rendered Output

- Example: HTML rendered in View with jQuery unobtrusive validation attributes

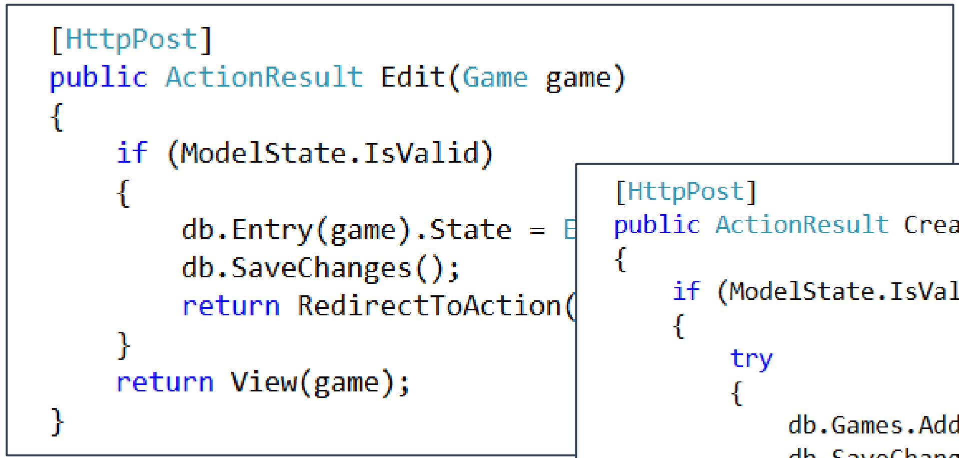
```
 <input class="text-box single-line" data-val="true" data-val-
number="The field Rating must be a number." data-val-range="The
field Rating must be between 1 and 5." data-val-range-max="5"
data-val-range-min="1" data-val-required="The Rating field is
required." id="Rating" name="Rating" type="number" value="" />
```

- Example: HTML rendered script references

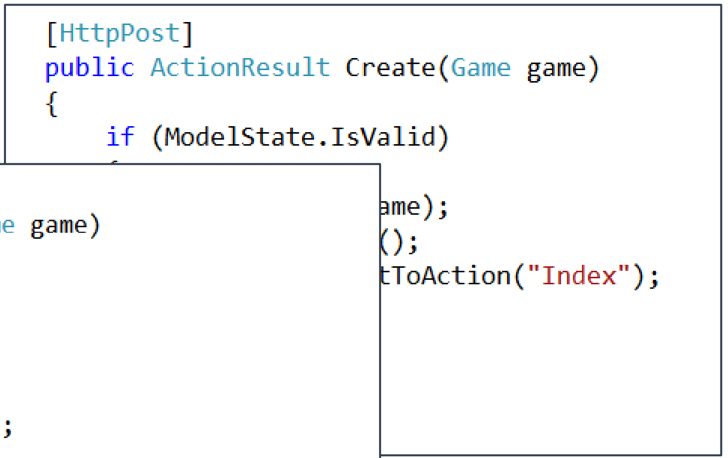


```
<environment names="Development">
    <script src="~/lib/jquery-validation/dist/jquery.validate.js"></script>
    <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment names="Staging,Production">
    <script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"
            asp-fallback-src="~/lib/jquery-validation/dist/jquery.validate.min.js"
            asp-fallback-test="window.jQuery && window.jQuery.validator">
    </script>
```

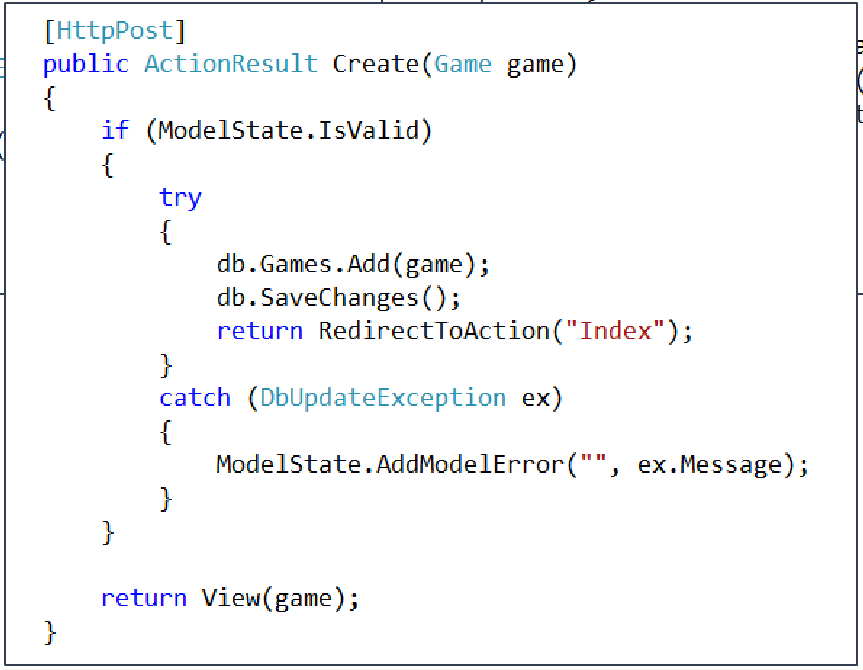
Data Annotations & ModelState



```
[HttpPost]
public ActionResult Edit(Game game)
{
    if (ModelState.IsValid)
    {
        db.Entry(game).State = E
        db.SaveChanges();
        return RedirectToAction(
    }
    return View(game);
}
```



```
[HttpPost]
public ActionResult Create(Game game)
{
    if (ModelState.IsValid)

                                ame);
                                ();
                                tToAction("Index");
```



```
[HttpPost]
public ActionResult Create(Game game)
{
    if (ModelState.IsValid)
    {
        try
        {
            db.Games.Add(game);
            db.SaveChanges();
            return RedirectToAction("Index");
        }
        catch (DbUpdateException ex)
        {
            ModelState.AddModelError("", ex.Message);
        }
    }

    return View(game);
}
```

Validation

- ValidationMessage
- ValidationSummary

```
@Html.ValidationMessageFor(model => model.Rating)
```

```
@Html.ValidationMessage("GameName", "some message")

@Html.ValidationSummary()
```

```
<span asp-validation-for=“Rating" class="text-danger"></span>
```

```
<div asp-validation-summary="ValidationSummary.All" class="text-
danger"></div>
```

Remote Attribute

```
[Remote("IsGameNameUnique", "Games", AdditionalFields = "GameId", ErrorMessage = "Game Name
must be a unique name!")]
         public string GameName { get; set; }
```

```
 public ActionResult IsGameNameUnique(string gameName, int? gameId)
        {
            var game = db.Games.FirstOrDefault(o => o.GameName == gameName);
            if (game == null)
                return Json(true, JsonRequestBehavior.AllowGet);

            return Json(game.GameId == gameId, JsonRequestBehavior.AllowGet);
        }
```

Validation != Security

- Include List

```
 // include list
 [HttpPost]
 public ViewResult Edit([Bind(Include = "GameName")]
Game game)
 {
    // ...
 }
```

- Bind against interface

```
 [HttpPost]
 public ActionResult Create(Game game)
 {
if (TryUpdateModel<IGameModel>(game))
      {
```

- Use ViewModel (Model-View-ViewModel (MVVM))

```
 [HttpPost]
 public ActionResult Create(GameViewModel game)
 {
```

Custom Attributes

- Custom Attribute with Client-Side Validation

```
public class UrlValidAttribute : ValidationAttribute, IClientValidatable
    {
        public override bool IsValid(object value)
        {
            if (value == null || ((string)value).ToLowerInvariant().Contains("microsoft"))
                return false;
            return true;
        }

        public IEnumerable<ModelClientValidationRule>
            GetClientValidationRules(ModelMetadata metadata, ControllerContext context)
        {
            yield return new ModelClientValidationRule
            {
                ErrorMessage = this.ErrorMessage,
                ValidationType = "urlvalid"
            };
        }
    }
```

Custom Validation

- Custom Client-Side Validation - In Views

```
@section Scripts {

    @Scripts.Render("~/bundles/jqueryval")

    <script type="text/javascript">
        // -- jQuery validation method
        jQuery.validator.addMethod('urlvalidCheck', function (value, element, params) {
            return (!/microsoft/.test(value));
        }, '');

        // add the unobtrusive adapter
        jQuery.validator.unobtrusive.adapters.add('urlvalid', {}, function (options) {
            options.rules['urlvalidCheck'] = true;
            options.messages['urlvalidCheck'] = options.message;
        });
    </script>
}
```

Client-side Validation

- Built-in jQuery validation methods

Validation Method	Description
minlength(length) Returns: Boolean	Makes the element require a given minimum length
maxlength(length) Returns: Boolean	Makes the element require a given maximum length
min(value) Returns: Boolean	Makes the element require a given minimum
max(value) Returns: Boolean	Makes the element require a given maximum
email( ) Returns: Boolean	Makes the element require a valid email
url( ) Returns: Boolean	Makes the element require a valid URL
dateISO( ) Returns: Boolean	Makes the element require a ISO date
number( ) Returns: Boolean	Makes the element require a decimal number
digits( ) Returns: Boolean	Makes the element require digits only
creditcard( ) Returns: Boolean	Makes the element require a creditcard number
accept(extension) Returns: Boolean	Makes the element require a certain file extension
equalTo(other) Returns: Boolean	Is Equal To

DataType Attribute

- Use **DataType** Attribute to leverage the existing jQuery validators, or add them to custom client validation rules by name

```
[DataType(DataType.CreditCard)]
public string CreditCard { get; set; }

[DataType(DataType.EmailAddress)]
public string Email { get; set; }

[DataType(DataType.Url)]
public string Url { get; set; }
```

Handling Validation Errors in Web API

- Web API does not automatically return an error to the client when validation fails
- Use the controller action to check for model state, and respond appropriately through HTTP.

```
[HttpPost]
public void CreateTodoItem([FromBody] TodoItem item)
{
            if (!ModelState.IsValid)
            {
                HttpContext.Response.StatusCode = 400;
            }
}
```

Module 7: Validation

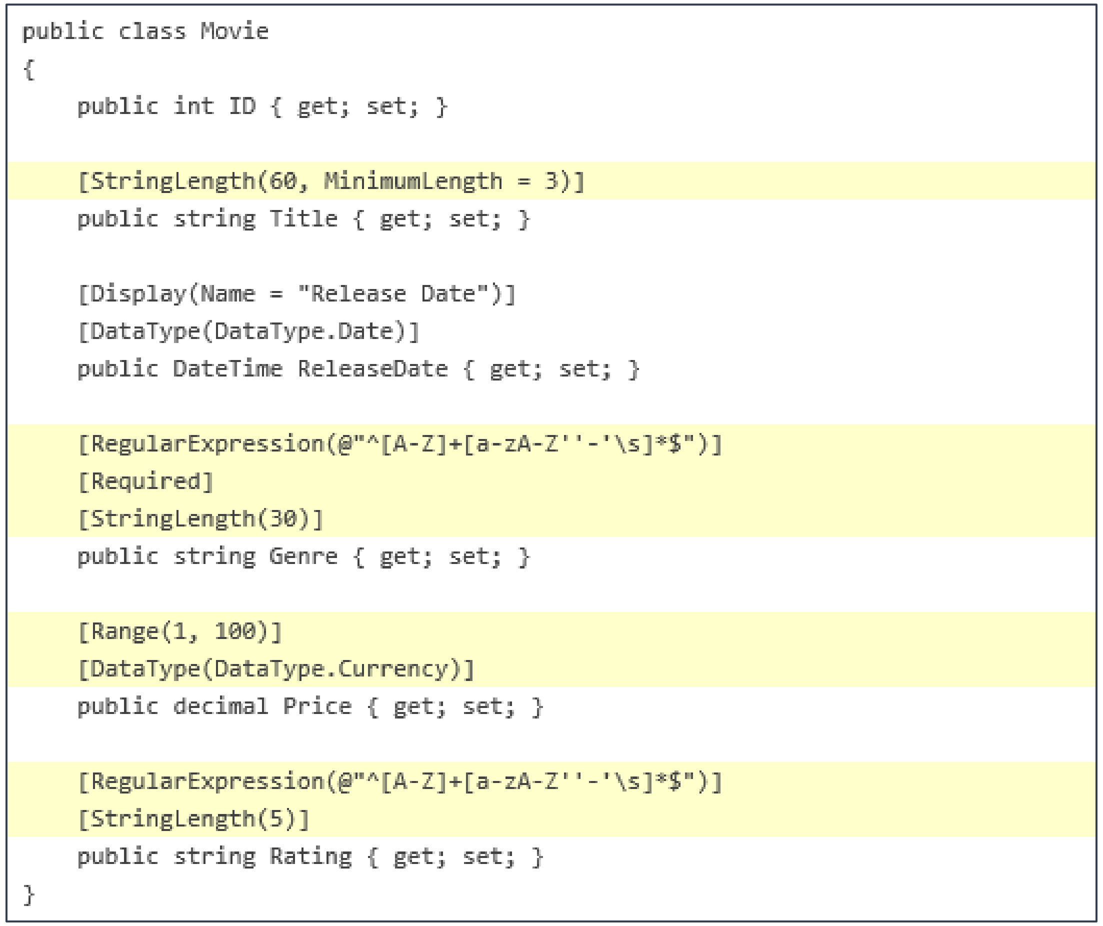
Section 2: Don’t Repeat Yourself Principle

Lesson: Example Scenario

Don’t Repeat Yourself!

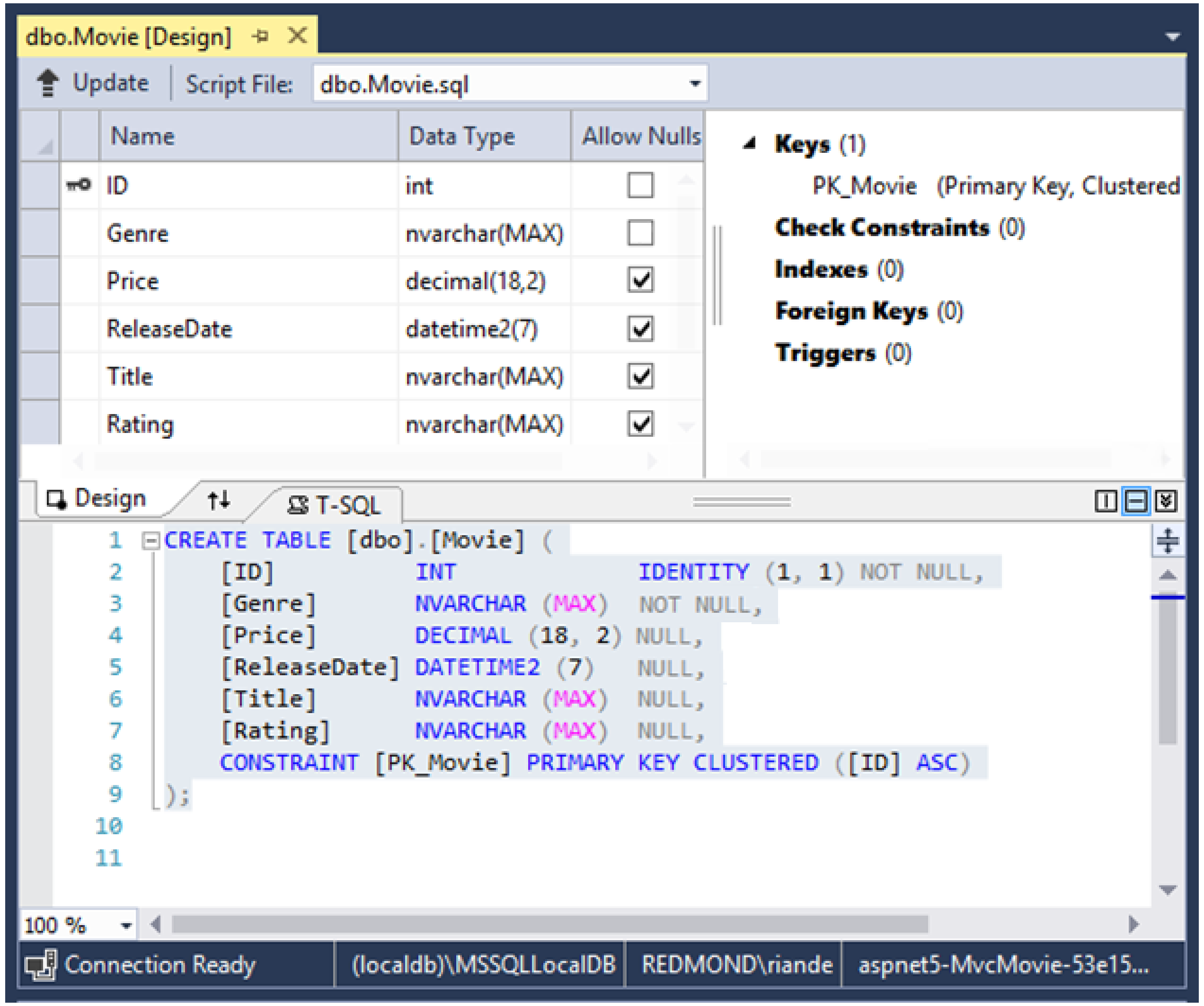
1. Define a Model

Movie Model



```
public class Movie
{
    public int ID { get; set; }

    [StringLength(60, MinimumLength = 3)]
    public string Title { get; set; }

    [Display(Name = "Release Date")]
    [DataType(DataType.Date)]
    public DateTime ReleaseDate { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z''-'\s]*$")]
    [Required]
    [StringLength(30)]
    public string Genre { get; set; }

    [Range(1, 100)]
    [DataType(DataType.Currency)]
    public decimal Price { get; set; }

    [RegularExpression(@"^[A-Z]+[a-zA-Z''-'\s]*$")]
    [StringLength(5)]
    public string Rating { get; set; }
}
```

2. Generated DB Schema



Name	Data Type	Allow Nulls
ID	int	☐
Genre	nvarchar(MAX)	☐
Price	decimal(18,2)	✓
ReleaseDate	datetime2(7)	✓
Title	nvarchar(MAX)	✓
Rating	nvarchar(MAX)	✓

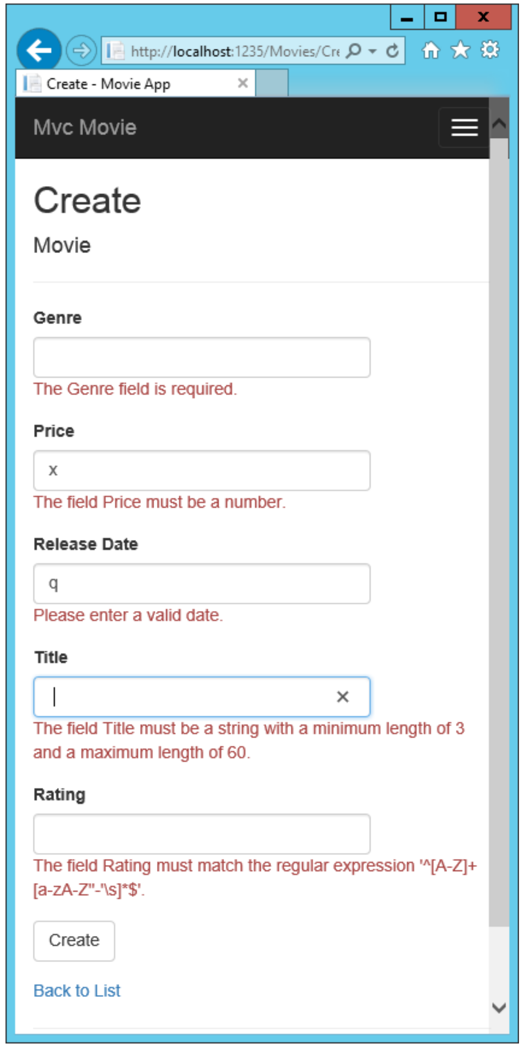
Keys (1)
PK_Movie (Primary Key, Clustered
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

```
CREATE TABLE [dbo].[Movie] (
    [ID]          INT            IDENTITY (1, 1) NOT NULL,
    [Genre]       NVARCHAR (MAX)  NOT NULL,
    [Price]       DECIMAL (18, 2) NULL,
    [ReleaseDate] DATETIME2 (7)   NULL,
    [Title]       NVARCHAR (MAX)  NULL,
    [Rating]      NVARCHAR (MAX)  NULL,
    CONSTRAINT [PK_Movie] PRIMARY KEY CLUSTERED ([ID] ASC)
);
```

3. Scaffolded Views with Validation

```
<form asp-action="Create">
    <div class="form-horizontal">
        <h4>Movie</h4>
        <hr />
        <div asp-validation-summary="ValidationSummary.ModelOnly" class="text-danger"></div>
        <div class="form-group">
            <label asp-for="Genre" class="col-md-2 control-label"></label>
            <div class="col-md-10">
                <input asp-for="Genre" class="form-control" />
                <span asp-validation-for="Genre" class="text-danger" />
            </div>
        </div>
    @*Markup removed for brevity.*@
        <div class="form-group">
            <label asp-for="Rating" class="col-md-2 control-label"></label>
            <div class="col-md-10">
                <input asp-for="Rating" class="form-control" />
                <span asp-validation-for="Rating" class="text-danger" />
            </div>
        </div>
        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Create" class="btn btn-default" />
            </div>
        </div>
    </div>
</form>
```

4. Validation Messages on UI


http://localhost:1235/Movies/Cre

Create - Movie App

Mvc Movie

Create

Movie

Genre

The Genre field is required.

Price

x

The field Price must be a number.

Release Date

q

Please enter a valid date.

Title

The field Title must be a string with a minimum length of 3 and a maximum length of 60.

Rating

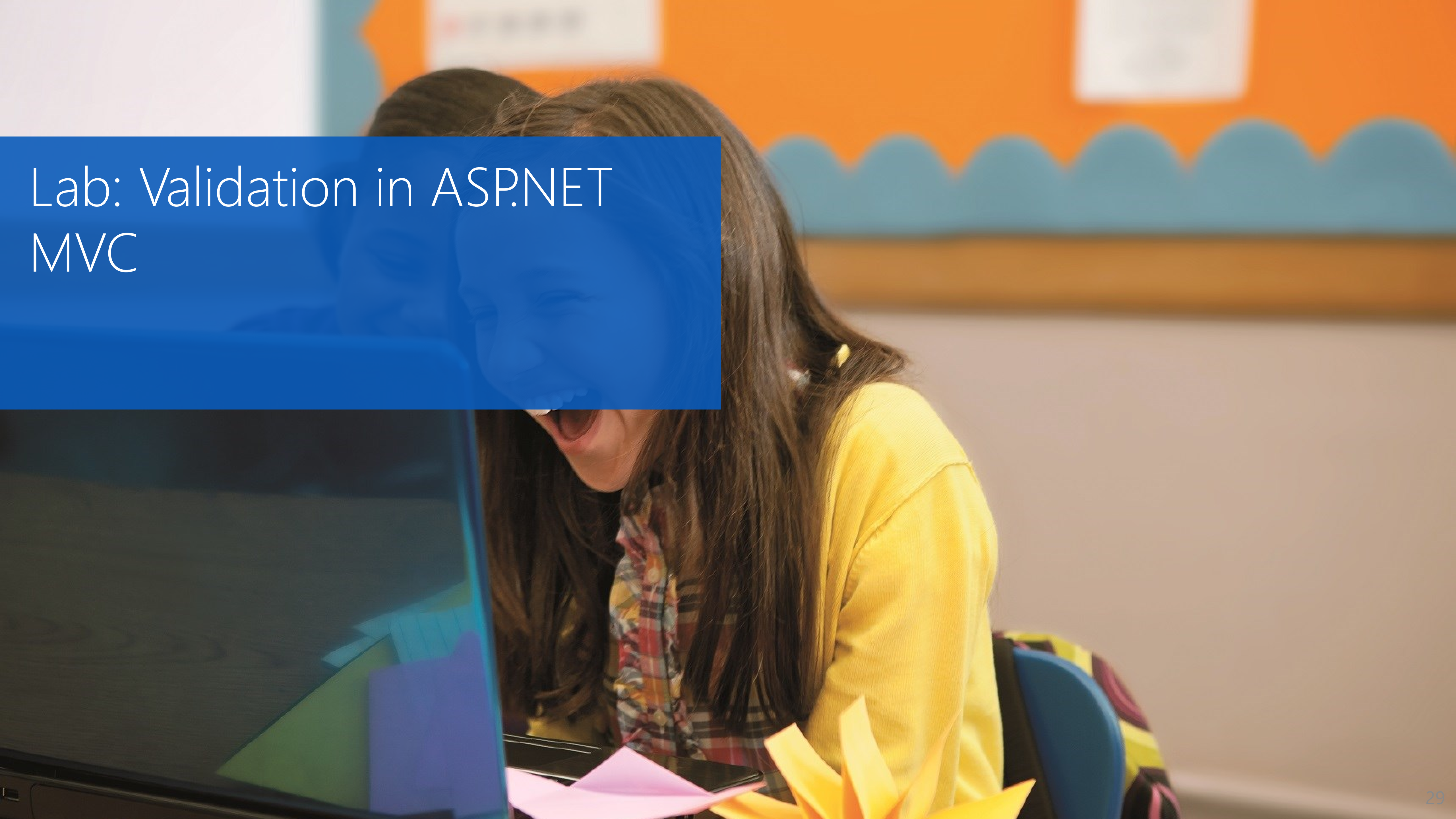
The field Rating must match the regular expression '^[A-Z]+[a-zA-Z''-'\s]*$'.

Create

Back to List

Module Summary

- In this module, you learned about:
  - Validation
  - Data Annotations
  - Client-Side and Server Side Validation
  - Validation != Security



Lab: Validation in ASP.NET MVC

Microsoft