Aula 03 – Criação e Validação de Formulários MVC

Disciplina: Programação Web
Prof. Allbert Velleniche de Aquino Almeida
E-mail: allbert.almeida@fatec.sp.gov.br
Site: http://www.allbert.com.br
/allbert.almeida

Agenda

- Criação de formulários;
- Métodos **GET** e **POST**;
- DataAnnotations;
- Validação de formulários:
  - Validation Summary;
  - Validation MessageFor;
  - RemoteValidation;

Criando formulário

- Vamos criar um Controller "Cadastro" e ver as várias possibilidades de construção dos formulários;

Formulário #1

- Criar uma action “Cliente” no controller;
- Adicionar a View para essa action;

Formulário #1

- Adicionar o código html do formulário:

```
<form action="~/Cadastro/Cliente/" method="post">
    <div class="row">
        <div class="col-md-6">
            Nome:
            <input name="nome" type="text" class="form-control" />
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            E-mail:
            <input name="email" type="email" class="form-control" />
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            <input type="submit" value="Salvar" class="btn btn-primary" />
        </div>
    </div>
</form>
```

GET e POST

- O método **GET** envia o formulário ou parâmetros através da URL;
- O método **POST** faz o encapsulamento no cabeçalho HTTP e envia para o servidor;

Formulário #1

- No controller adicionamos outra action "Cliente", agora do tipo "HttpPost":

```
[HttpPost]
public ActionResult Cliente(string nome, string email)
{
    ViewBag.Nome = nome;
    ViewBag.Email = email;
    return View();
}
```

Formulário #1

- Na view "Cliente", após o </form>, inserimos as ViewBags para visualizarmos a submissão do formulário:

```
<br />
@ViewBag.Nome<br />
@ViewBag.Email
```

Formulário #2

- Nesse exemplo #2 vamos associar um formulário a um modelo;
- Na pasta "Models" vamos criar uma classe chamada "Livro", com as propriedades do tipo string: Titulo e Editora;
- Criar uma action "Livro" no controller;

Formulário #2

- Vamos associar a view ao formulário:
- Com botão direito do mouse sobre a action "Livro" no controller >> Add View >> Modelo: Empty e Classe do modelo: Livro
- Na view adicionada a primeira linha deve ser:

```
@model WebApplication1.Models.Livro
```

Formulário #2

- Adicionar o código html do formulário (Helpers):

```
<form action="~/Cadastro/Livro/" method="post">
    <div class="row">
        <div class="col-md-6">
            @Html.LabelFor(m => m.Titulo)
            @Html.EditorFor(m => m.Titulo, new { htmlAttributes = new { @class =
"form-control" } })
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            @Html.LabelFor(m => m.Editora)
            @Html.EditorFor(m => m.Editora, new { htmlAttributes = new { @class
= "form-control" } })
        </div>
    </div>
    <div class="row">
        <div class="col-md-6">
            <input type="submit" value="Salvar" class="btn btn-primary" />
        </div>
    </div>
</form>
```

Formulário #2

- No controller adicionamos outra action "Livro", agora do tipo "HttpPost", recebendo o objeto da classe "Livro":

```
[HttpPost]
public ActionResult Livro(Livro livro)
{
    ViewBag.Titulo = livro.Titulo;
    ViewBag.Editora = livro.Editora;
    return View();
}
```

Formulário #2

- Na view "Livro", após o </form>, inserimos as ViewBags para visualizarmos a submissão do formulário:

```
<br />
@ViewBag.Titulo<br />
@ViewBag.Editora
```

DataAnnotations

- Uma maneira simples de implementar as regras de negócio no cliente e servidor;
- Uma propriedade com atributos Data Annotations pode ser customizada para que se torne um campo obrigatório, para que tenha uma formatação de exibição, para que esteja de acordo com o modelo de dados da sua aplicação, e também para que sejam aplicadas regras específicas, como limite de idade, caracteres permitidos em um campo, etc.

Formulário #2

- Na classe "Livro" iremos adicionar referência:

```
using System.ComponentModel.DataAnnotations
```

- Sobre cada propriedade adicionamos as regras:

```
[Required]
[StringLength(100, MinimumLength =3,ErrorMessage ="O
título deve ter entre 3 e 100 caracteres")]
[Display(Name ="Título do livro")]
public string Titulo { get; set; }
[Required]
[RegularExpression("^[a-zA-Z ]+$", ErrorMessage ="Somente
letras")]
public string Editora { get; set; }
```

Formulário #2

- Na view "Livro" iremos trocar a tag form por:

```
@using (Html.BeginForm()) { }
```

- Dentro { } inserimos o formulário e:

```
@Html.AntiForgeryToken()
```

- Adicionamos para cada campo:

```
@Html.ValidationMessageFor(model =>
model.Titulo, "", new { @class = "text-
danger" })
```

Formulário #2

- Na action "Livro" :

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Livro(Livro livro) {
    if (ModelState.IsValid) {
        ViewBag.Titulo = livro.Titulo;
        ViewBag.Editora = livro.Editora;
        return View();
    }
    return View(livro);
}
```

Formulário #3

- Nesse exemplo #3 vamos associar um formulário a um modelo;
- Quando criamos um modelo podemos associar este a um modelo de view pré definida (cadastro, edição, exclusão, lista, entre outros);

Formulário #3

- Vamos criar um novo modelo. Na pasta models adicionem uma classe "Aluno"

```
public class Aluno {
        [Required]
        [Range(100,200)]
        public int Matricula { get; set; }
        [Required]
        public string Nome { get; set; }
        [Required]
        public string Curso { get; set; }
        public bool Ativo { get; set; }
}
```

Formulário #3

- Adicionem uma ActionResult "Aluno"
- Com botão direito do mouse sobre a action "Aluno" no controller >> Add View >> Modelo: Create e Classe do modelo: Aluno
- Na view adicionada a primeira linha deve ser:

```
@model WebApplication1.Models.Aluno
```

Formulário #3

- Na action "Aluno" :

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Aluno(Aluno aluno)
{
    if (ModelState.IsValid)
    {
        ViewBag.Aluno = aluno;
    }
    return View(aluno);
}
```

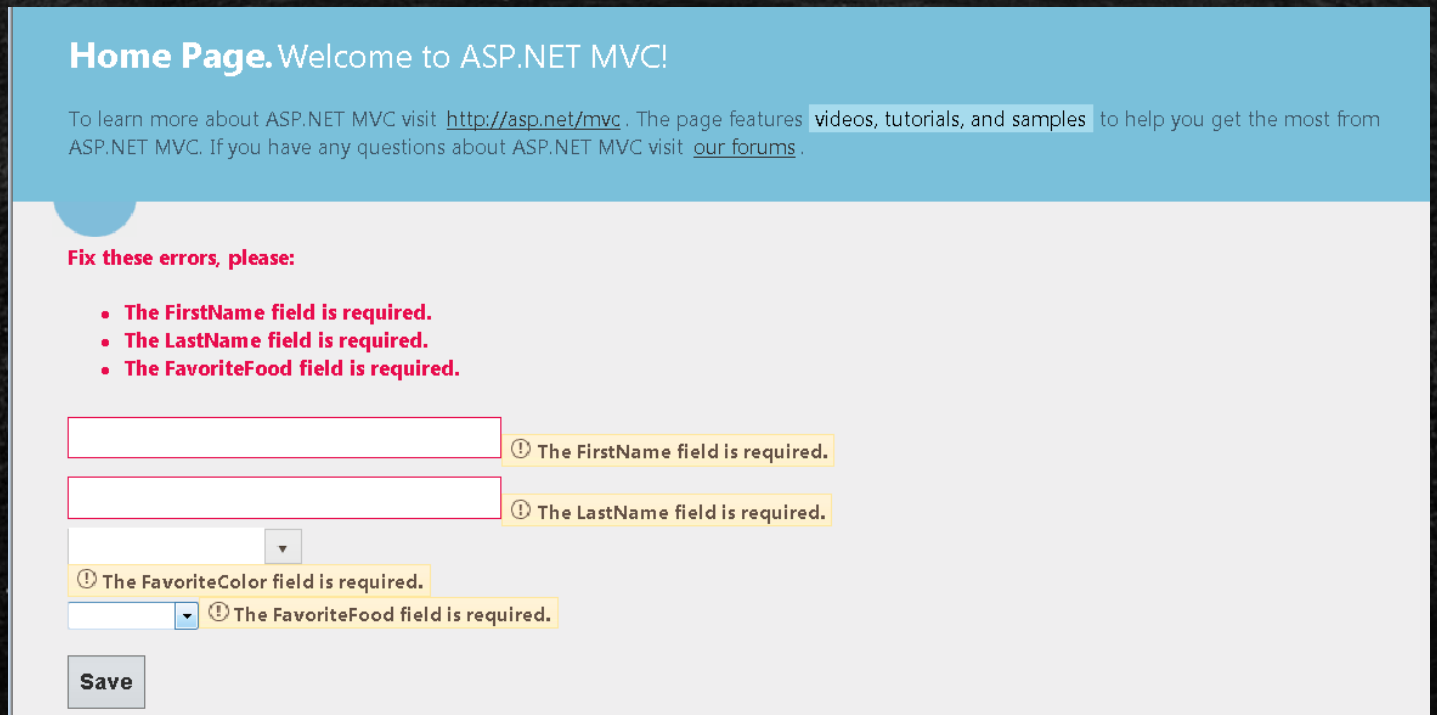
Formulário #3

- Na view "Aluno", após o "}", inserimos as ViewBags para visualizarmos a submissão do formulário:

```
<br />
@if (ViewBag.Aluno != null)
{
    @ViewBag.Aluno.Matricula<br />
    @ViewBag.Aluno.Nome<br />
    @ViewBag.Aluno.Curso<br />
    @ViewBag.Aluno.Ativo<br />
}
```

Validation Summary

- Exibe uma lista de todos os erros contidos no ModelState;



Formulário #3

- Alterando a action "Aluno", dentro do ModelState.IsValid:

```
if(aluno.Nome=="erro") {
   ModelState.AddModelError("", "Erro ao gravar no
banco de dados");
   return View(aluno);
}
```

Remote Validation

- O processo de validação remota permite validar os dados específicos de postagem para um servidor sem postar todos os dados do formulário;

Formulário #3

- Para validação remota iremos utilizar dois métodos no controller de "Cadastro":

```
[HttpPost]
public JsonResult VerificaMatricula(int Matricula) {
    return Json(ValidaMatricula(Matricula),
JsonRequestBehavior.AllowGet);
}

public bool ValidaMatricula(int Matricula) {
    return !Matricula.Equals(123);
}
```

Formulário #3

- No model “Aluno” , iremos adicionar sobre a propriedade “Matricula”:

```
[Remote("VerificaMatricula", "Cadastro",HttpMethod
="POST", ErrorMessage = "Matrícula já utilizada")]
```

- E adicionar a referência:

```
using System.Web.Mvc;
```

Formulário #3

- Na view "Aluno" , iremos adicionar antes do fechamento do formulário:

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

Formulário #3

- Para a validação funcionar no lado servidor, é necessário adicionar essa validação, dentro do ModelState.IsValid:

```
if (ValidaMatricula(aluno.Matricula)==false) {
    ModelState.AddModelError("Matricula",
"Matrícula já utilizada");
    return View(aluno);
}
```

Atividade Prática

- Crie uma classe modelo “Produto”, com as seguintes propriedades:
  - Descricao: Obrigatório, mínimo 5 caracteres e no máx. 200, utilize o display para acentuação do campo;
  - Preco: Tipo decimal, Obrigatório , utilize o display para acentuação do campo;
  - Marca: Opcional

Atividade Prática

- Adicione uma action “NovoProduto” e adicione view, utilizando o modelo “Create”;
- Receba os dados do formulário através da nova action do tipo HttpPost e escreva os valores submetidos em ViewBags;
- Escreva os valores na View;