FIAP

FÁBRICA DE DESENVOLVIMENTO

#19 – Tratamento de Erros e Testes

FIAP

FÁBRICA DE DESENVOLVIMENTO – 2018

Prof. Thiago T. I. Yamamoto

#19 – TRATAMENTO DE ERROS E TESTES

thiagoyama thiagoyama@gmail.com

CONTEÚDO PROGRAMÁTICO

#19 – TRATAMENTO DE ERROS E TESTES

- Tratamento de Exceções
- Custom Errors
- Testes
- Teste de Unidade

FIAP

TRATAMENTO DE ERROS

TRATAMENTO DE EXCEÇÕES

FIAP

- Podemos tratar as exceções com o bloco **try.. catch..**
  - Não lance Exception(), prefira criar as suas próprias exceções;
  - Não capture exceções com catch vazio;

CUSTOM ERRORS

FIAP

- A configuração de página de erro fica no arquivo **web.config.**
- Na sessão **<customErrors>** a propriedade **mode**:
  - **Off:** exibe uma página com detalhes de um erro sempre que um erro não tratado por try.. catch.. ocorre. Essa tela é chamada de “tela amarela da morte”.
  - **On:** exibe uma página de erro configurada;
  - **RemoteOnly:** a página com detalhes de erro é exibida para usuários locais e uma página de erro configurada é exibida para usuários remotos;

CONFIGURAÇÃO

- Podemos definir o redirecionamento padrão através do atributo **defaultRedirect;**
- Podemos definir um redirecionamento para cada código de erro, adicionando a **tag error**;

```
<system.web>
    <customErrors mode="On" defaultRedirect="Erro/Default">
      <error statusCode="404" redirect="Erro/NotFound"/>
    </customErrors>
  </system.web>
```

REDIRECIONAMENTO

- O redirecionamento é feito para uma **action** de um **controller**:

```
public class ErroController : Controller
{
        public ActionResult Default()
        {
            return View();
        }

        public ActionResult NotFound()
        {
            return View();
        }
}
```

FIAP

TESTES

TESTES

- Testar o código implementado é fundamental no ciclo de desenvolvimento de sistemas;
- Existem vários tipos de testes: unitário, stress, integração, usabilidade, segurança, etc.
- **Teste de Unidade (Unit Test):**
  - Um dos primeiros testes que o desenvolvedor deve realizar;
  - Desenvolvido durante a implementação do código;
  - Responsável por testar a menor unidade de software desenvolvido;

TESTES UNITÁRIOS

FIAP

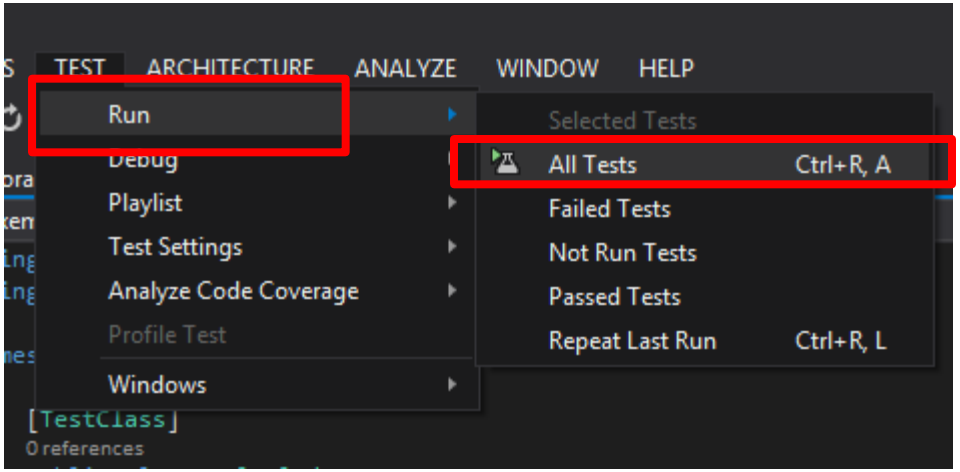
- Um dos benefícios de utilizar o padrão de projetos MVC é a facilidade da implementação de testes unitários;
- É possível testar cada camada da aplicação;
- Estrutura de um teste de unidade:
  - *Arrange*: inicializar os objetos;
  - *Act*: chamar a funcionalidade a ser testada;
  - *Assert*: analizar o resultado produzido pela funcionalidade;

EXEMPLO

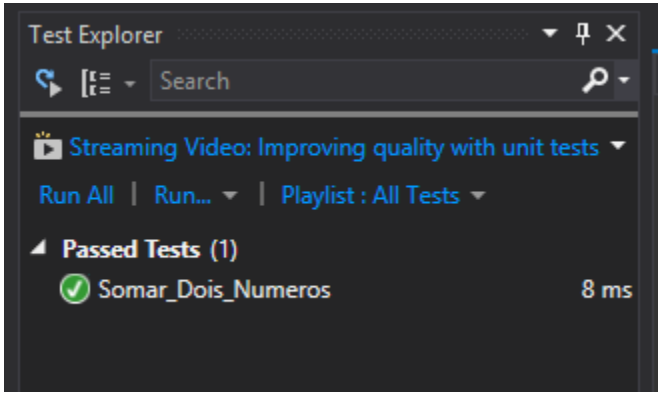
- Teste do método somar da classe Calculadora;

```
[TestClass]
public class CalculadoraTest
{
    [TestMethod]
    public void Somar_Dois_Numeros()
    {
        Calculdarora calc = new Calculdarora();
        var resultado = calc.Somar(10, 20);
        Assert.AreEqual(resultado, 30);
    }
}
```

EXECUTANDO OS TESTES



Execute os testes através da opção: **Test -> Run -> All Tests**



O Resultado é apresentado no **Test Explorer**

MÉTODO ASSERT

Method	Parameters	Notes
`AreEqual()`	`Object expected` `Object actual`	Verifies that two objects are equal. One is the expected result, while the other is the actual result produced in the Act part of the unit test.
`AreEqual<T>()`	`T expected` `T actual`	Verifies that two generic type data are equal.
`AreNotEqual()`	`Object notExpected` `Object actual`	Verifies that two objects are not equal.
`AreNotEqual<T>()`	`T notExpected` `T result`	Verifies that two generic type data are not equal.
`AreNotSame()`	`Object notExpected` `Object actual`	Verifies that two object variables reference to different objects.
`AreSame()`	`Object expected` `Object actual`	Verifies that two object variables reference to the same object.
`Equals()`	`Object objA` `Object objB`	Verifies whether two objects are equal.
`IsFalse()`	`bool condition`	Verifies if the condition is false.
`IsTrue()`	`bool condition`	Verifies if the condition is true.
`IsNull()`	`Object object`	Verifies if the object is null.
`IsNotNull()`	`Object object`	Verifies if the object is not null.
`IsInstanceOfType()`	`Object value` `Type expectedType`	Verifies if the object is an instance of the specified type.

http://msdn.microsoft.com/pt-br/library/microsoft.visualstudio.testtools.unittesting.assert.aspx

EXEMPLO DE TESTE

- Podemos fazer um ajuste para que o teste utilize um banco separado;
- Vamos adicionar um construtor no Context para que seja possível passar a string de conexão do banco de dados;

```
public EcommerceContext(string stringConnection)
            : base(stringConnection)
{
    Database.SetInitializer(new
        DropCreateDatabaseAlways<EcommerceContext>());
}
```

EXEMPLO DE TESTE

- Na classe de teste, podemos adicionar um método que é executado antes dos testes, para inicialização.

```
[TestClass]
public class ClienteRepositoryTest
{
     ClienteRepository rep;
     EcommerceContext context;

     [TestInitialize]
     public void init()
     {
          context = new EcommerceContext("Teste");
          rep = new ClienteRepository(context);
     }

    //...
}
```

EXEMPLO DE TESTE - BUSCA

```
[TestMethod]
public void Buscar_Por_Nome()
{
        var cli = new Cliente
        {
            Nome = "Thiago",
            Sobrenome = "Yamamoto",
            Plano = new Plano
            {
                Nome = "Teste"
            },
            Email = "Thiagoyma@gmai.com",
            EnderecoCobranca = new Endereco
            {
                Logradouro = "Teste"
            }
            };
            rep.Add(cli);
            context.SaveChanges();

            var cliente = rep.SearchByName("Thiago");
            Assert.AreEqual(cliente.Count, 1);
            foreach (var item in cliente)
            {
                Assert.AreEqual(item.Nome, "Thiago");
            }
}
```

EXEMPLO DE TESTE - EXCEPTIONS

- Podemos testar exceptions esperados:

```
[TestMethod]
[ExpectedException(typeof(DbUpdateException))]
public void Cadastrar_Sem_Endereco_Exception()
{
     var cli = new Cliente
     {
          Nome = "Thiago"
     };
     rep.Add(cli);
     context.SaveChanges();
}
```

EXEMPLO DE TESTE - CONTROLLER

FIAP

- Podemos testar os controllers, e verificar o resultado (ViewResult);
- Através do resultado temos acesso ao Model, View, ModelState TempData, ViewBag e etc..

```
[TestMethod]
public void Cadastrar_Get()
{
      ClienteController controller = new ClienteController();
      var result = (ViewResult)controller.Cadastrar();
      Assert.IsNotNull(result);
      Assert.AreEqual("", result.ViewName);
}
```

EXEMPLO DE TESTE - CONTROLLER

FIAP

- Teste de cadastro através da action do controller:

```
[TestMethod]
public void Cadastrar_Post()
{
      ClienteController controller = new ClienteController();
      var cliente = new ClienteViewModel
      {
            Nome = "Thiago",
            Sobrenome = "Yamamoto",
            Logradouro = "Teste"
      };
      var result = (ViewResult) controller.Cadastrar(cliente);
      Assert.IsNotNull(result);

      var modelResultado = result.Model as ClienteViewModel;
      Assert.AreEqual(modelResultado.Mensagem,
                  "Cliente Cadastrado com sucesso!");
}
```

EXEMPLO DE TESTE - CONTROLLER

- Teste de validação do cadastro através da action do controller:

```
[TestMethod]
public void Cadastrar_Nome_Obrigatorio()
{
        ClienteController controller = new ClienteController();
        var cliente = new ClienteViewModel
        {
             Sobrenome = "Yamamoto",
             Logradouro = "Teste"
        };
        var result = (ViewResult)controller.Cadastrar(cliente);
        Assert.IsNotNull(result);

        var modelValido = result.ViewData.ModelState.IsValid;
        Assert.IsFalse(modelValido);

        var chaveErros = result.ViewData.ModelState.Keys;
        foreach (var item in chaveErros)
        {
             Assert.AreEqual("Nome", item);
        }
}
```

FIAP

"Para conquistar o sucesso, você precisa aceitar todos os desafios que vierem na sua frente. Você não pode apenas aceitar os que você preferir"
- Mike Gafka