

- ◆ Model: Use [UIHint("CKEditor")]

```
public partial class BaiViet
{
    [Key]
    [Display(Name="Mã bài viết")]
    public string maBaiViet { get; set; }
    [Display(Name="Tựa bài viết")]
    [Required(ErrorMessage="Phải nhập tiêu đề bài viết")]
    public string tuaBaiViet { get; set; }
    [Display(Name="Nội dung tóm tắt")]
    [Required(ErrorMessage = "Phải nhập nội dung tóm tắt")]
    public string noiDungTomTat { get; set; }
    [UIHint("CkEditor")]
    [Required(ErrorMessage = "Phải nhập nội dung chính")]
    [Display (Name="Nội dung chính")]
    [AllowHtml]
    public string noiDungChinh { get; set; }
    public Nullable<System.DateTime> ngayDang { get; set; }
    public string maLoai { get; set; }
```

- ◆ Download CKEditor library:
<http://ckeditor.com/>
- ◆ Copy CKEditor library to Script folder
- ◆ Create EditorTemplates in Shared Folder

- ◆ Create CKEditor.cshtml in View/Shared/

EditorTemplates:

- ▶  Shared
- ▶  EditorTemplates
 - [@] CkEditor.cshtml
 - [@] _Layout.cshtml

- ◆ Code file:

```
<script src="~/Scripts/ckeditor/ckeditor.js"></script>
<textarea class="ckeditor" name="editor1"></textarea>
```

- ◆ Controller:

```
public class BaiVietController : Controller
{
    Personal_FSoftEntities1 db = new Personal_FSoftEntities1();
    //Action nhập bài viết
    public ActionResult NhapBaiViet()
    {
        ViewBag.LoaiBaiViet = new SelectList(db.LoaiBaiViets, "maLoai", "tenLoai");
        return View();
    }
}
```

◆ Controller:

```
[HttpPost]
public ActionResult NhapBaiViet(BaiViet bv)
{
    bv.maBaiViet = "BV_" + DateTime.Now.ToString("yyyyMMddhhmmss");
    bv.ngayDang = DateTime.Now;

    if (ModelState.IsValid)
    { //xử lý nhập bài viết
        db.BaiViets.Add(bv);
        db.SaveChanges();
        return RedirectToAction("Index");
    } //trường hợp sai dữ liệu, chọn lại Loại bài viết trên DropDownList
    ViewBag.LoaiBaiViet = new SelectList(db.LoaiBaiViets, "maLoai",
        "tenLoai", bv.maLoai);
    //trả lại trạng thái trên View với các thông tin bài viết
    return View(bv);
}
```

- ◆ View:

```
@model prjMVCEExample.Models.BaiViet
@{
    ViewBag.Title = "NhapBaiViet";
    Layout = "~/Views/Shared/Admin_Layout.cshtml";
}
<h2>NhapBaiViet</h2>

@using (Html.BeginForm())
{
    .....
    @Html.LabelFor(model => model.noDungChinh, new
    { @id = "control-label col-md-2" })
    <div class="col-md-10">
        @Html.TextAreaFor(model => model.noDungChinh, new
        { @class = "ckeditor", @name = "editor1" })
        @Html.ValidationMessageFor(model => model.noDungChinh)
```

NhapBaiViet

BaiViet

Tựa bài viết

Nội dung tóm tắt

Nội dung chính

The image shows the CKEditor interface. At the top is a toolbar with various icons for text formatting, such as bold, italic, underline, and alignment. Below the toolbar is a larger text area for the main content. A status bar is visible at the bottom of the editor window.

maLoai

 Công nghệ thông tin ▾

Create

Action Filter

Table of Contents

- ◆ Action Filters

- ◆ [Authorize]
- ◆ [AllowAnonymous]
- ◆ [ValidateAntiForgeryToken]
- ◆ [ValidateInput]
- ◆ [HandleError]
- ◆ [InitializeSimpleMembership]

Action Filter

- ◆ Action filter is used to perform the tasks occur at the time of implementation of an action different.
- ◆ Some Action Filter is available by MVC
 - ◆ [ValidateInput]
 - ◆ [Authorize]
 - ◆ [ValidateAntiForgeryToken]
 - ◆ [HandleError]

Action Filter

- ◆ Also we can write Action Filter for private purposes
 - ◆ [InitializeSimpleMembership]
- ◆ Filters can be used to filter:
 - ◆ A single action
 - ◆ The actions in a Controller
 - ◆ All actions of the project

[ValidateInput]

- ◆ HTML data is sended to Action
 - ◆ True: can send HTML data
 - ◆ False: receive error message
- ◆ Example:

```
public class HomeController : Controller
{
    [ValidateInput(false)]
    public ActionResult Send(Mail mail)
    {
```

[Authorize]

- ◆ [Authorize] is used to validate accessing the action with appropriate authorities
- ◆ There are 3 types:
 - ◆ [Authorize]: allow access after logging
 - ◆ [Authorize(users="u1,u2,...")]: allow access with user u1, u2, ...
 - ◆ [Authorize(roles="r1,r2,...")]: allow access with users have roles r1, r2, ...

[Authorize]

```
[Authorize(Roles="Administrator")]
public ActionResult Manage()
{
    return View();
}
```

```
[Authorize(Users = "thphuong")]
public ActionResult Config()
{
    return View();
}
```

```
[Authorize]
public ActionResult MyAccount()
{
    return View();
}
```

[Authorize]

Must login

```
[Authorize]  
[InitializeSimpleMembership]  
public class AccountController : Controller  
{  
    public ActionResult ChangePassword()...
```

```
[AllowAnonymous]  
public ActionResult Login(string returnUrl)...
```

allow
anonymous
access

[ValidateAntiForgeryToken]

- [ValidateAntiForgeryToken]: prevent request forgery

```
<form action="/home/savedata" method="post">  
    <%= Html.AntiForgeryToken() %>  
    <input type="submit" value="Submit from App1" />  
</form>
```

```
[HttpPost]  
[ValidateAntiForgeryToken]  
public ViewResult SaveData()  
{  
    return View();  
}
```

[HandleError]

- ◆ Simply Try...Catch approach

```
public ActionResult TestMethod()
{
    try
    {
        //...
        return View();
    }
    catch (Exception e)
    {
        //Handle Exception;
        return View("Error");
    }
}
```

- ◆ *Problem with the above approach is we cannot reuse the exception handling logic across multiple action methods.*

[HandleError]

- ◆ Global level Exception Handling

1. Web.config:

```
<system.web>
  <customErrors mode="On" defaultRedirect="Error" />
```

- ◆ TestMethod:

```
public ActionResult TestMethod()
{
    int x = 0;
    x /= x;
    return View();
}
```

[HandleError]

1. FilterConfig class

```
public class FilterConfig
{
    public static void RegisterGlobalFilters(GlobalFilterCollection filters)
    {
        filters.Add(new HandleErrorAttribute());
    }
}
```

2. Where is RegisterGlobalFilters method invoked?

[HandleError]

1. In Global.asax file in Application_Start
RegisterGlobalFilters method is invoked.

```
public class MvcApplication : System.Web.HttpApplication
{
    protected void Application_Start()
    {
        AreaRegistration.RegisterAllAreas();
        WebApiConfig.Register(GlobalConfiguration.Configuration);
 FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
        RouteConfig.RegisterRoutes(RouteTable.Routes);
    }
}
```

[HandleError]

- ◆ HandleError at controller level?

```
[HandleError(View = "TestMethod")]
public class TestController : Controller
{
    public ActionResult TestMethod()
    {
        int x = 0;
        x /= x;
        return View();
    }
}
```

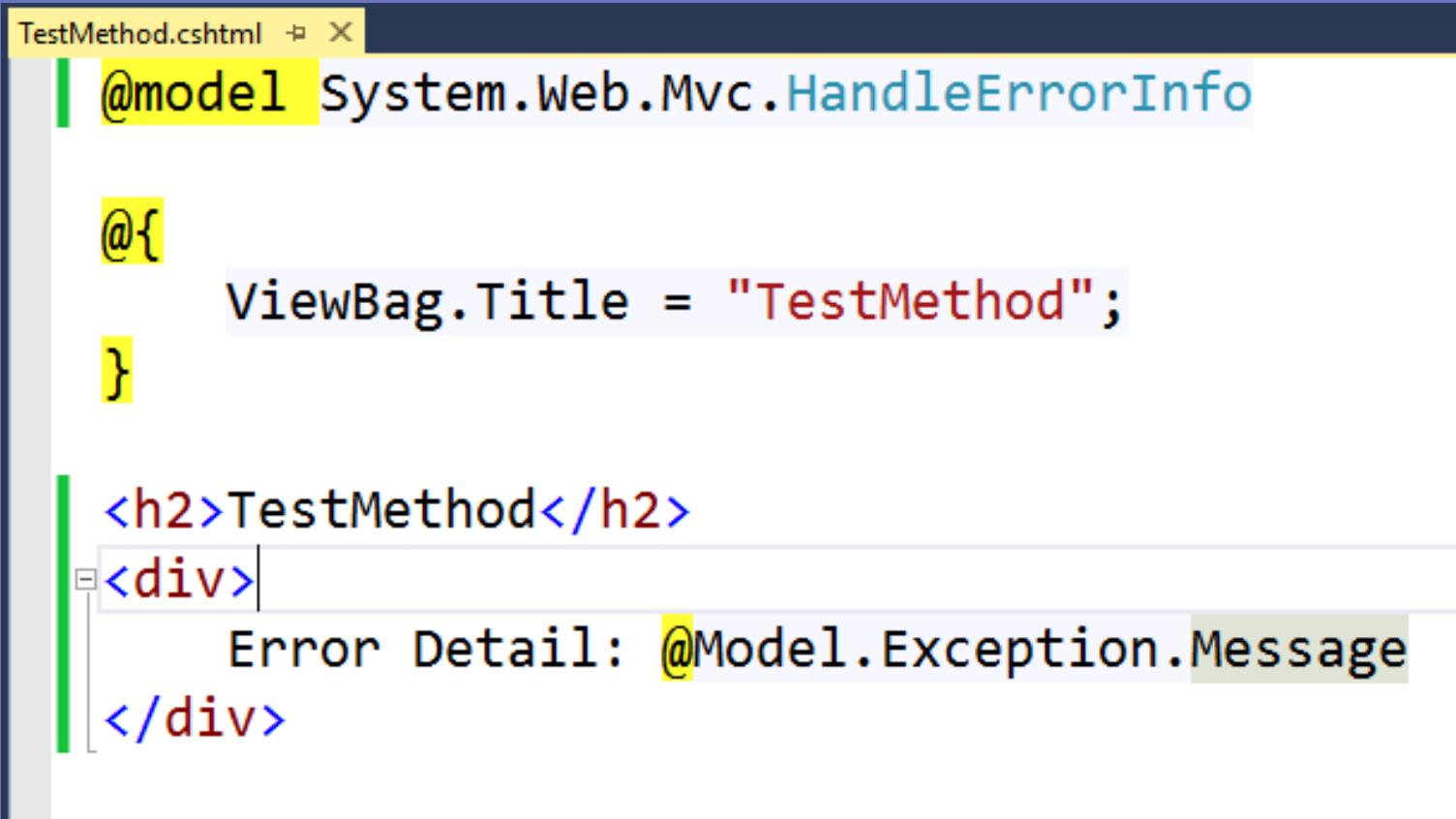
[HandleError]

- ◆ HandleError at Action level

```
public class TestController : Controller
{
    [HandleError(View = "TestMethod")]
    public ActionResult TestMethod()
    {
        int x = 0;
        x /= x;
        return View();
    }
}
```

[HandleError]

- ◆ View:



```
TestMethod.cshtml
@model System.Web.Mvc.HandleErrorInfo

@{
    ViewBag.Title = "TestMethod";
}



## TestMethod

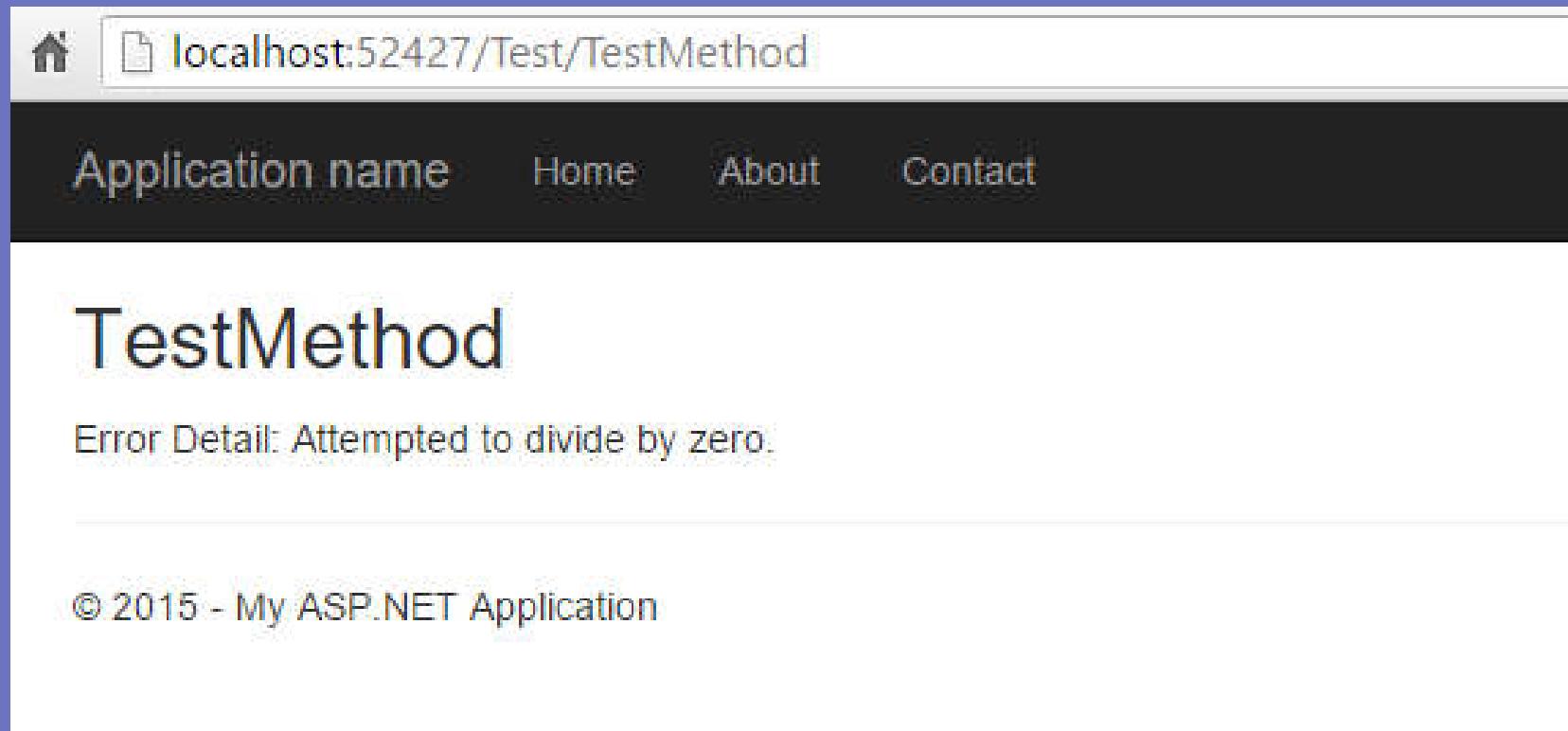


Error Detail: @Model.Exception.Message


```

[HandleError]

- ◆ Web Result:



[HandleError]

- ◆ Different views for Different Exceptions
 - ◆ For that we will use overloaded version of HandleErrorAttribute constructor as follows.

```
[HandleError(ExceptionType=typeof(DivideByZeroException),View="TestMethod")]
[HandleError(ExceptionType=typeof(NullReferenceException),View="NullError")]
public class TestController : Controller
{
```

Questions?