Ecole Nationale des Sciences Appliquées de Safi

A.U:2016-2017

Programmation avancée en c#.NET

Meryem OUARRACHI

Plan du module

- Programmation Web

-ASP WebForms

-Web Service

-ASP MVC

- WPF
- Programmation RIA en Silverlight
- Informatique décisionnelle en .Net

CHAPITRE3:

ASP MVC

AJAX

Implementation d'ajax en ASP MVC

Pour activer la mise à jour partielle de la page:

Comme le principe d'Ajax dans le PHP, l'idée est d'écrire les résultats dans une balise div vide sans actualiser la page.

1. Ajouter les scripts d'ajax en Scripts

```
@section scripts
{
    <script type="text/javascript" src="~/Scripts/jquery-1.7.1.js"></script>
    <script type="text/javascript" src="~/Scripts/jquery.unobtrusive-ajax.js"></script>
}
```

Implementation d'ajax en ASP MVC

2.Utiliser Ajax.BeginForm() qui permet de soumettre un formulaire de manière asynchrone (équivalent script manager en ASP WebForms)

```
@Ajax.BeginForm("AfficherDetails","Ajax",
                    new AjaxOptions { HttpMethod = "POST", UpdateTargetId = "div1",
                        InsertionMode = InsertionMode.Replace,
                         })
<h1>Enter Product Code</h1>
<h2>
     Code du produit: @Html.TextBox("Code")<br />
    <input type="submit" value="Show Details" />
    <input type="reset" value="Clear"/>
</h2>
@{Html.EndForm();}
<br />
<br />
<div id="div1">
</div>
```

Implementation d'ajax en ASP MVC

3.On crée l'action responsable de restitution de la page:

```
public PartialViewResult AfficherDetails()
{
    string code = Request.Form["Code"];
    Product prod = new Product();
    foreach (Product p in prodList)
    {
        if (p.ProdCode == code)
        {
            prod = p;
            break;
        }
    }
    return PartialView("_AfficherDetails", prod);
}
```

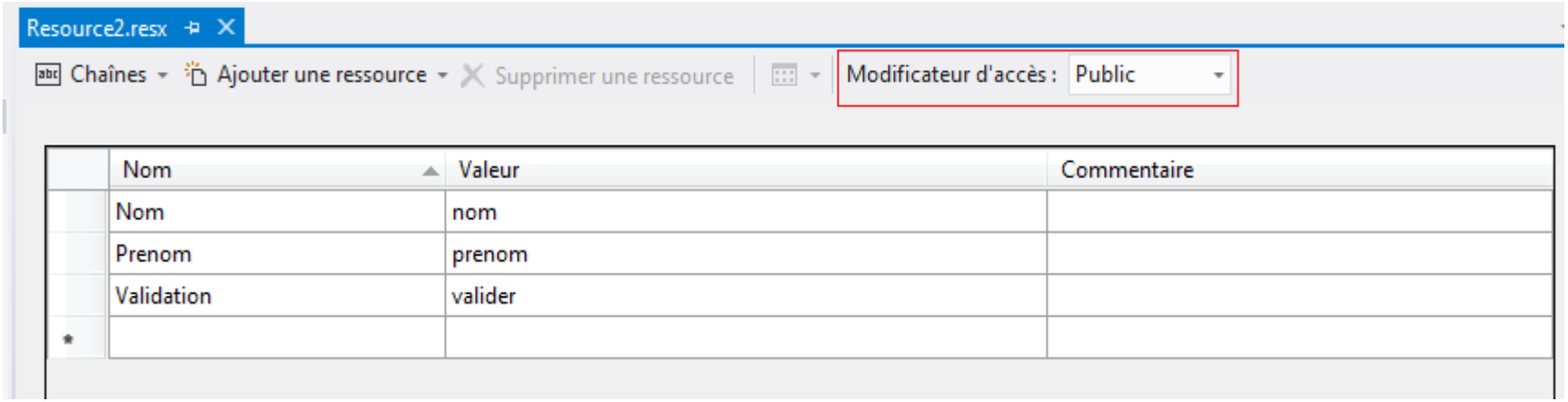
4.On définit _AfficherDetails responsable de l'affichage de l'élément sélectionné

INTERNATIONALISATION EN ASP MVC

Application multilingue

•Etape1: Créer des fichiers ressources des contenus des vues:

1. Créer un dossier Resources dans le projet
2. Créer des fichiers resources dans le dossier précédent
3. Maintenant au lieu d'appeler un nom on va l'appeler à partir de ressource file



Application multilingue

Etape2: Appeler le contenu de fichier ressource

NomProjet.Resources.NomFichierResources.NomPropriété

Exemple:

```
@{
    ViewBag.Title = MvcMultilingue5.Resources.Resource1.Accueil;
    Layout = "~/Views/Shared/_Layout.cshtml";
}
```

Application multilingue

Etape3: Déterminer la langue à afficher

```
Thread.CurrentThread.CurrentUICulture = new
CultureInfo(lang);
```

```
Thread.CurrentThread.CurrentCulture =
CultureInfo.CreateSpecificCulture(lang);
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

1. Définir une classe permettant de gérer les langues: cette classe va contenir des méthodes permettant de:

-changer la langue du thread courant selon le choix linguistique de l'utilisateur→Sauvegarder le choix dans un cookie

-Si aucune langue n'est spécifiée dans l'URL, la langue du Thread d'exécution sera modifiée en utilisant par défaut la langue du navigateur de l'utilisateur.

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

1. Définir une classe permettant de gérer les langues:

```
public class GestionLanguages
{       //la liste des langues possibles
  public static List<Languages> AvailableLanguages = new List<Languages>
 {
  new Languages{ LangFullName = "English", LangCultureName = "en"},
  new Languages{ LangFullName = "Français", LangCultureName = "fr"},

  };
            //voir si la langue demandée est autorisée
  public static bool IsLanguageAvailable(string lang)
  {
  return AvailableLanguages.Where(a => a.LangCultureName.Equals(lang)).FirstOrDefault() != null ? true : false;
  }
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

1. Définir une classe permettant de gérer les langues:

```
//Récupérer langue par défaut
public static string GetDefaultLanguage()
{
    return AvailableLanguages[0].LangCultureName;
}
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

1. Définir une classe permettant de gérer les langues:

```
//changer la langue
public void SetLanguage(string lang)
{
    try
    {
        if (!IsLanguageAvailable(lang))
            lang = GetDefaultLanguage();
        var cultureInfo = new CultureInfo(lang);
        Thread.CurrentThread.CurrentUICulture = cultureInfo;
        Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(cultureInfo.Name);

        HttpCookie langCookie = new HttpCookie("culture", lang);
        langCookie.Expires = DateTime.Now.AddYears(1);
        HttpContext.Current.Response.Cookies.Add(langCookie);

    }
    catch (Exception ex)
    {

    }
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

2. Créer une classe de type Controller:

Cette classe va être appeler lors de l'exécution de n'importe quel Controller→Redéfnir la méthode BeginExecuteCore

-Par la suite les controllers doivant hériter de cette classe au lieu de la classe controller

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

2. Créer une classe de type Controller:

```
public class MyBaseController : Controller
{
    protected override IAsyncResult BeginExecuteCore(AsyncCallback callback, object state)
    {
        string lang = null;
        HttpCookie langCookie = Request.Cookies["culture"];
        if (langCookie != null)
        {
            lang = langCookie.Value;
        }
        else
        {  //pour détecter la culture à partir du navigateur de l'utilisateur.
           var userLanguage = Request.UserLanguages;

            var userLang = userLanguage != null ? userLanguage[0] : "";
            if (userLang != "")     { lang = userLang;  }
            else        { lang = SiteLanguages.GetDefaultLanguage(); }
        }
        new SiteLanguages().SetLanguage(lang);
        return base.BeginExecuteCore(callback, state);
    }
}
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

3. Ajouter des composants permettant de basculer entre les langues:

```
@{
 foreach (var i in MvcMultilingue5.GestionLanguages.AvailableLanguages)
 {
 @Html.ActionLink(i.LangFullName, "ChangeLangue", "Home", new{lang = i.LangCultureName}, null)
 }
 }
```

Application multilingue

- **Mécanisme de sélection de la langue dans l'IU**

3. Ajouter des composants permettant de basculer entre les langues:

```
public class HomeController : MyBaseController
{
    public ActionResult ChangeLangue(string lang)
    {
        new GestionLanguages().SetLanguage(lang);
        return Redirect(Request.UrlReferrer.ToString());
        //Obtient des informations sur l'URL de la précédente requête du client
        //qui était liée à l'URL actuelle.
    }
```

Application multilingue

- Localisation des messages d'erreur de validation et DisplayAttribute

```
public class EtudiantModel
{[Required(ErrorMessageResourceType = typeof(MvcMultilingue5.Resources.Resource1),
    ErrorMessageResourceName = "NameRequired")]


   [Display(Name = "Nom",ResourceType=typeof(MvcMultilingue5.Resources.Resource1))]
   public string FirstName { get; set; }
```