

ASP.NET MVC : deuxième partie

Nouhaila Bensalah

Chercheuse en IA/NLP

nouhaila.bensalah@etu.fstm.ac.ma



- 1 Création d'une page web dynamique avec ASP.NET
 - Création d'une page web dynamique avec ASP.NET: Modèle
 - Création d'une page web dynamique avec ASP.NET: Contrôleur
 - Création d'une page web dynamique avec ASP.NET: Vue

- 2 Création d'une page web avec ASP.NET: Contrôleur suite

- 3 Création d'une page web avec ASP.NET: Modèle suite

1 Création d'une page web dynamique avec ASP.NET

- Création d'une page web dynamique avec ASP.NET: Modèle
- Création d'une page web dynamique avec ASP.NET: Contrôleur
- Création d'une page web dynamique avec ASP.NET: Vue

2 Création d'une page web avec ASP.NET: Contrôleur suite

3 Création d'une page web avec ASP.NET: Modèle suite

Objectif

- Réalisation d'une petite application web afin d'intégrer la notion de Modèle-Vue-Contrôleur.
- Dans cette application, le visiteur rentrera son nom dans un champ de texte modifiable et lorsqu'il va cliquer sur le bouton **Valider**, un message va s'afficher: "Bonjour" suivi du texte saisi.

- 1 **Création d'une page web dynamique avec ASP.NET**
 - **Création d'une page web dynamique avec ASP.NET: Modèle**
 - Création d'une page web dynamique avec ASP.NET: Contrôleur
 - Création d'une page web dynamique avec ASP.NET: Vue
- 2 Création d'une page web avec ASP.NET: Contrôleur suite
- 3 Création d'une page web avec ASP.NET: Modèle suite

Réalisation d'une petite application web afin d'intégrer la notion de MVC

Décrire les informations: création du modèle

- Faire un clic droit sur le nom du projet dans l'Explorateur de solutions.
- Aller dans Ajouter — > Class.
- Choisir Class.
- Saisir Visiteur dans Nom : et valider.

```
class Visiteur
{
    public string Prenom { get; set; }
}
```

Modèle

On distingue trois types :

- *Modèle de données* : constitué des classes qui interagissent avec une base de données pour l'objectif de stocker ou récupérer des données.
- *Modèle Business* : des classes qui implémentent des fonctionnalités représentant des règles de traitements.
- *Modèle de vue* : des classes utilisées pour passer des informations d'un contrôleur à une vue.

- 1 **Création d'une page web dynamique avec ASP.NET**
 - Création d'une page web dynamique avec ASP.NET: Modèle
 - **Création d'une page web dynamique avec ASP.NET: Contrôleur**
 - Création d'une page web dynamique avec ASP.NET: Vue
- 2 Création d'une page web avec ASP.NET: Contrôleur suite
- 3 Création d'une page web avec ASP.NET: Modèle suite

Création d'une page web dynamique avec ASP.NET

Réalisation d'une petite application web afin d'intégrer la notion de MVC

Création du contrôleur:

Création du contrôleur

- Faire un clic droit sur **Controllers** et aller dans **Ajouter** → **Contrôleur**.
- Sélectionner **MVC 5 Controller - Empty** et cliquer sur **Ajouter**.
- Saisir un nom pour le contrôleur (**SalutationController**) et cliquer sur **Add**.

Création d'une page web dynamique avec ASP.NET

Contenu de: SalutationController.cs

```
public class SalutationController : Controller
{
    public ActionResult Index()
    {
        Visiteur client = new Visiteur();

        return View(client);
    }

    [AcceptVerbs(HttpVerbs.Post)]
    public ActionResult Index(Visiteur visiteur)
    {
        Visiteur client = new Visiteur();

        string prenom = "";

        prenom = Request.Form["prenom_visiteur"];

        client.Prenom = prenom; // Model binding: allows you to map request information (example: forms) to named parameters

        ViewData["message"] = "Bonjour, " + prenom;

        return View("Index", client);
    }
}
```

Création d'une page web dynamique avec ASP.NET

Récupération des données: Contrôleur

Property	Type	Description
Request.QueryString	NameValueCollection	GET variables sent with this request
Request.Form	NameValueCollection	POST variables sent with this request
Request.Cookies	HttpCookieCollection	Cookies sent by the browser with this request
Request.HttpMethod	string	The HTTP method (verb, such as GET or POST) used for this request
Request.Headers	NameValueCollection	The full set of HTTP headers sent with this request
Request.Url	Uri	The URL requested
Request.UserHostAddress	string	The IP address of the user making this request
RouteData.Route	RouteBase	The chosen <code>RouteTable.Routes</code> entry for this request
RouteData.Values	RouteValueDictionary	Active route parameters (either extracted from the URL or default values)
HttpContext.Application	HttpApplicationStateBase	Application state store
HttpContext.Cache	Cache	Application cache store
HttpContext.Items	IDictionary	State store for the current request

Transfert des données: contrôleur

- **ViewBag, ViewData:**

permet de stocker des données dans une action de contrôleur qui sera utilisée dans la vue correspondante.

- **TempData:**

permet de stocker des données pendant la durée de la requête HTTP dans une action du contrôleur qui sera utilisée dans une autre action du contrôleur. Une fois la redirection effectuée, les données sont automatiquement supprimées.

Création d'une page web avec ASP.NET

TempData: Exemple

```
public ActionResult Test() {  
    TempData["test"] = "object";  
  
    return View(); } // a controller action that will consume this data
```

ViewBag: Exemple

```
public ActionResult Test() {  
    ViewBag.Test = "object";  
  
    return View(); } // et dans la vue: @ViewBag.Test
```

ViewData: Exemple

```
public ActionResult Test() {  
    ViewData["Test"] = "object";  
  
    return View(); } // et dans la vue: @ViewData["Test"]
```

- 1 **Création d'une page web dynamique avec ASP.NET**
 - Création d'une page web dynamique avec ASP.NET: Modèle
 - Création d'une page web dynamique avec ASP.NET: Contrôleur
 - **Création d'une page web dynamique avec ASP.NET: Vue**
- 2 Création d'une page web avec ASP.NET: Contrôleur suite
- 3 Création d'une page web avec ASP.NET: Modèle suite

Réalisation d'une petite application web afin d'intégrer la notion de MVC

Création de la vue:

Création de la vue

- Faire un clic droit sur la méthode **Index** et cliquer sur **Ajouter une vue**.
- Sélectionner **Vue MVC 5** et cliquer sur **Ajouter**.
- Sélectionner : **Empty** pour le *modèle*, **Visiteur** pour la *classe du modèle* et cliquer sur **Ajouter**.

Création d'une page web dynamique avec ASP.NET

Contenu Index.cshtml

```
<html>
<head>

    <meta name="viewport" content="width=device-width" />

    <title>Formulaire</title>
</head>
<body>

    <div>

        @using (Html.BeginForm()){

            <h1>Bonjour MVC Learner </h1>

            <p>Veuillez entrez votre nom?</p>

            @Html.TextBox("prenom_visiteur") //c'est la syntaxe d'un moteur de vue appelé Razor, On peut utiliser <input
type="text" name="prenom_visiteur" id="prenom_visiteur" />

            <input type="submit" value="Valider" /> }

            <p>@ViewData["message"]</p>

        </div>

    </body>
</html>
```


Transfert des données: Vue

- **ViewBag, ViewData:**

permet de stocker des données dans une action de contrôleur qui sera utilisée dans la vue correspondante.

- **TempData:**

permet de stocker des données pendant la durée de la requête HTTP dans une action du contrôleur qui sera utilisée dans une autre action du contrôleur. Une fois la redirection effectuée, les données sont automatiquement supprimées.

Création d'une page web dynamique avec ASP.NET

TempData: Exemple

```
public ActionResult Test() {  
    TempData["test"] = "object";  
  
    return View(); // a controller action that will consume this data}
```

ViewBag: Exemple

```
public ActionResult Test() {  
    ViewBag.Test = "object";  
  
    return View(); } // et dans la vue: @ViewBag.Test
```

ViewData: Exemple

```
public ActionResult Test() {  
    ViewData["Test"] = "object";  
  
    return View(); } // et dans la vue: @ViewData["Test"]
```

- 1 Création d'une page web dynamique avec ASP.NET
 - Création d'une page web dynamique avec ASP.NET: Modèle
 - Création d'une page web dynamique avec ASP.NET: Contrôleur
 - Création d'une page web dynamique avec ASP.NET: Vue
- 2 Création d'une page web avec ASP.NET: Contrôleur suite
- 3 Création d'une page web avec ASP.NET: Modèle suite

Pour rediriger vers une autre action, on utilise la méthode
RedirectToAction

```
RedirectToAction("action", "controller", new { paramName = value});  
//Ce qui génère la route suivante controller/action/value
```

Pour récupérer la route courante dans le contrôleur

```
string host = HttpContext.Request.RawUrl; //En allant à  
https://localhost:Numéro_de_port/home/index/9, la variable host récupère  
la valeur /home/index/9
```

Autres informations à récupérer

- **HttpContext.Request.HttpMethod** : pour récupérer le verbe HTTP utilise pour executer l'action
- **HttpContext.Request.Url.AbsolutePath** : pour récupérer le chemin absolu demande (comme /home/index/9)
- **HttpContext.Request.Url.AbsoluteUri** : pour récupérer l'URL complète (comme https://localhost:Numéro_de_port/home/index/2)
- **HttpContext.Request.Url.Host** : pour récupérer l'adresse IP du visiteur
- ...

Pour modifier le nom d'une action, on utilise le décorateur **ActionName**

```
public class HomeController : Controller
{
    [ActionName("first")]
    public string Index()
    {
        return $"Hello first";
    }
}
```

En allant à `/home/first`, un Hello first sera affiché.

En allant à `/home/index`, une erreur 404 sera affichée.

Pour ne pas considérer une méthode du contrôleur comme une action, on utilise le décorateur **NonAction**

```
public class HomeController : Controller
{
    [NonAction]
    public string DoSomething()
    {
        //...
    }
}
```

En allant à `/home/first`, un Hello first sera affiché.

En allant à `/home/index`, une erreur 404 sera affichée.

1 Création d'une page web dynamique avec ASP.NET

- Création d'une page web dynamique avec ASP.NET: Modèle
- Création d'une page web dynamique avec ASP.NET: Contrôleur
- Création d'une page web dynamique avec ASP.NET: Vue

2 Création d'une page web avec ASP.NET: Contrôleur suite

3 Création d'une page web avec ASP.NET: Modèle suite

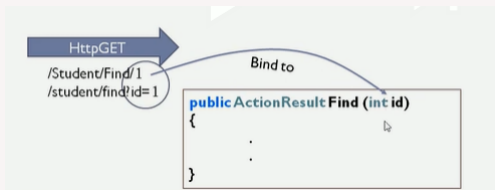
Model binding

With the Model binding, MVC framework converts the http request values to action method parameters.

Le processus du **model binding** s'exécute automatiquement si la l'action demandée porte des paramètres.

Ces paramètres peuvent être de:

- Primitive type



- Complex type (class, structure)

Récupération les données

Property	Type	Description
Request.QueryString	NameValueCollection	GET variables sent with this request
Request.Form	NameValueCollection	POST variables sent with this request
Request.Cookies	HttpCookieCollection	Cookies sent by the browser with this request
Request.HttpMethod	string	The HTTP method (verb, such as GET or POST) used for this request
Request.Headers	NameValueCollection	The full set of HTTP headers sent with this request
Request.Url	Uri	The URL requested
Request.UserHostAddress	string	The IP address of the user making this request
RouteData.Route	RouteBase	The chosen <code>RouteTable.Routes</code> entry for this request
RouteData.Values	RouteValueDictionary	Active route parameters (either extracted from the URL or default values)
HttpContext.Application	HttpApplicationStateBase	Application state store
HttpContext.Cache	Cache	Application cache store
HttpContext.Items	IDictionary	State store for the current request

Model binding: Include, Exclude and IsPropertyAllowed

- **Exclude** : La propriété Exclude est utilisée pour obtenir ou définir une liste délimitée par des virgules de noms de propriétés pour lesquelles la liaison n'est pas autorisée.
- **Include** : La propriété Include est utilisée pour obtenir ou définir une liste délimitée par des virgules de noms de propriétés pour lesquels la liaison est autorisée.
- **IsPropertyAllowed** : La méthode IsPropertyAllowed est utilisée pour déterminer si la propriété spécifiée est autorisée. Elle renvoie True si la propriété spécifiée est autorisée ; sinon, False.

Supposons qu'on a: Classe Produit

- ProductId
- ProductName
- ProductType

Include

```
public ActionResult Test([Bind(Include = "ProductName", ProductType")]  
Produit product) // Dans le contrôleur crée
```

Exclude

```
public ActionResult Test([Bind(Exclude = "ProductId")] Produit product)
```

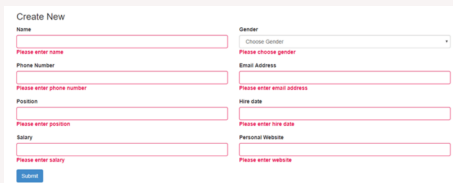
Validation des données

- Avant qu'une application ne stocke des données dans une base de données, il faut vérifier que les données ne présentent pas de menaces potentielles pour la sécurité (exemple: formatées par type et par taille) en utilisant des annotations.
- Le model binder intercepte toutes les erreurs, et les mettent dans le ModelState : une propriété du Controller.
- Exemple:
 - `ModelState.IsValid == false` // vérification au niveau de tout le modèle (dans le contrôleur crée)
 - `ModelState.IsValidField("FirstName") == false` // vérification au niveau de chaque propriété (dans le contrôleur crée)

Validation client vs Validation serveur

Dans MVC, la validation s'effectue à la fois sur le client et sur le serveur.

- La saisie de l'utilisateur qui est validée dans le navigateur avant d'être soumise au serveur est appelée **validation côté client**.



The screenshot shows a web form titled "Create New" with two columns of input fields. The left column contains fields for Name, Phone Number, Position, and Salary. The right column contains fields for Gender (a dropdown menu), Email Address, Hire date, and Personal Website. Each field has a red border and a red error message below it: "Please enter name", "Please enter phone number", "Please enter position", "Please enter salary", "Please choose gender", "Please enter email address", "Please enter hire date", and "Please enter website". A blue "Submit" button is located at the bottom left of the form.

- La validation côté serveur** entre en jeu lorsqu'un utilisateur malveillant peut soumettre des données par le biais de différents canaux ou encore soumettre des entrées dangereuses au serveur.

Annotations: Exemple

```
public class WebUser
{
    [Required]

    [StringLength(25)]

    public string FirstName { get; set; }
    [Required]

    [StringLength(50, MinimumLength = 3)]

    public string LastName { get; set; }

    [EmailAddress]

    public string MailAddress { get; set; }
}
```

See [This](#) for the other annotations