ASP.NET MVC : introduction

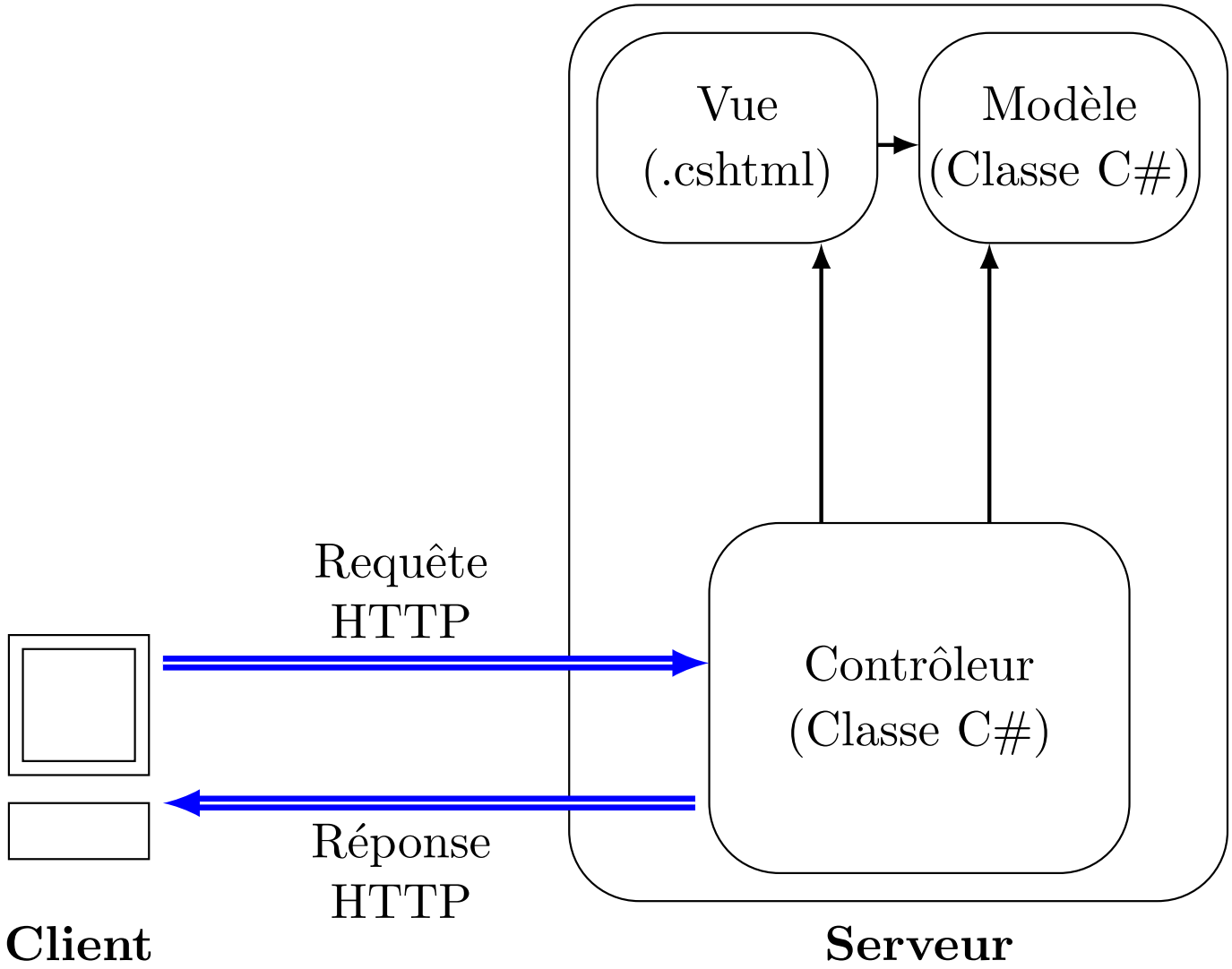
Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

ASP.NET MVC


Vue
(.cshtml)
Modèle
(Classe C#)
Requête
HTTP
Contrôleur
(Classe C#)
Réponse
HTTP
Client
Serveur

ASP.NET MVC

Déroulement

1. Le client envoie une requête depuis la vue (son navigateur).
2. Le contrôleur intercepte et analyse la requête du client.
3. Le contrôleur détermine quelle partie du modèle est concernée afin d'effectuer les traitements nécessaires.
4. Le modèle s'occupe de l'interaction avec les données, applique les règles métier et renvoie les données (dénués de toute présentation) au contrôleur.
5. Le contrôleur sélectionne la vue correspondante et lui injecte les données.
6. La vue affiche les données au client.

ASP.NET MVC

Exemple d'applications web **ASP.NET MVC**

- **Stack OverFlow** (forum de discussion entre développeurs)
- **CodePlex** (dépôt de projets)

ASP.NET MVC

Étapes

- Créer un nouveau projet `Fichier > Nouveau > Projet`
- Cliquer sur `Installé` et choisir `C#`
- Étendre la rubrique `Web` et sélectionner `Application web ASP.NET (.NET Framework)`
- Remplir le champs `Nom :` par `FirstAspNet`
- Valider puis sélectionner un modèle `Vide` et cocher la case `MVC`
- Valider et attendre la fin de création du projet

ASP.NET MVC

Plusieurs types de projets

- `Vide` sans cocher la case `MVC` : projet complètement vide.
- `Vide` en cochant la case `MVC` : projet avec des répertoires vides (`Models`, `Views`, `Controllers`...).
- `MVC` : projet avec des répertoires (`Models`, `Views`, `Controllers`...) non vides + inclusion de Bootstrap et jQuery.
- `WEB API` : projet non vide pour la création de services RESTful.
- ...

ASP.NET MVC

Structure du projet

- Une application MVC ⇒
  - `Models` : où on va placer les classes qui vont interagir avec les données
  - `Views` : où on va placer les pages HTML qui serviront de réponse à la requête utilisateur
  - `Controllers` : où on va mettre les classes qui contiendront les actions à exécuter dès réception d'une requête
- `packages.config` : fichier de configuration des packages `NuGet`.
- `App_Start` : où on va placer les fichiers de configuration. Il contient `RouteConfig.cs` qui servira de mapping route/contrôleur/action
- `Global.asax` : point d'entrée vers l'application. Il fait appel à `RouteConfig` présent dans `App_Start`

ASP.NET MVC

Si on exécute, on obtient une erreur 404

- Aucune action associée à la route `/`.
- D'ailleurs, le répertoire `Controllers` est vide $\equiv$ pas de contrôleur $\Rightarrow$ pas d'action.

ASP.NET MVC

Si on exécute, on obtient une erreur 404

- Aucune action associée à la route `/`.
- D'ailleurs, le répertoire `Controllers` est vide $\equiv$ pas de contrôleur $\Rightarrow$ pas d'action.

Solution : créer un contrôleur (et une action) et l'associer à cette route

ASP.NET MVC

Contrôleur dans une application **ASP.NET MVC**

- Classe **C#** héritant de la classe `Controller`
- Ayant des méthodes `public` appelées `Action`
- Nom suffixé par `Controller`

ASP.NET MVC

Pour créer un 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 (par exemple : `HomeController`) et cliquer sur `Add`
- Une classe `HomeController` qui hérite de `Controller` (du namespace `Web.Mvc`) est générée

ASP.NET MVC

Pour créer un 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 (par exemple : `HomeController`) et cliquer sur `Add`
- Une classe `HomeController` qui hérite de `Controller` (du namespace `Web.Mvc`) est générée

Un répertoire `Home` a été créé dans `Views`

ASP.NET MVC

Code généré pour `HomeController`

```
public class HomeController : Controller
{
    // GET: Home
    public ActionResult Index()
    {
        return View();
    }
}
```

ASP.NET MVC

Modifions la méthode `Index()` de `HomeController`

```
public class HomeController : Controller
{
    // GET: Home
    public string Index()
    {
        return "Hello World !!";
    }
}
```

Exécutons une deuxième fois le projet ⇒ `Hello World !!` est affiché.

ASP.NET MVC

Contenu de `RouteConfig`

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "
       Index", id = UrlParameter.Optional }
);
```

ASP.NET MVC

Contenu de `RouteConfig`

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "
       Index", id = UrlParameter.Optional }
);
```

Remarques

- `controller` : mot-clé obligatoire permettant de préciser le nom du contrôleur à exécuter
- `action` : mot-clé obligatoire permettant de préciser le nom de la méthode à exécuter

ASP.NET MVC

testez les URL suivantes

- /index
- /Home
- /home
- /HOME
- /home/index
- /home/index/2
- /home/index2

ASP.NET MVC

Contenu de `RouteConfig`

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index"
       , id = UrlParameter.Optional }
);
```

ASP.NET MVC

Contenu de `RouteConfig`

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index"
       , id = UrlParameter.Optional }
);
```

Explication

Si une route de cette forme ({controller}/{action}/{id}) est saisie

- le contrôleur nommé `controllerController` sera instancié
- La méthode `action` définie dans ce même contrôleur sera exécutée
- On peut même accepter un paramètre `id` (optionnel)

ASP.NET MVC

Si une (ou plus) de ces 3 parties manque à l'appel, on peut définir des valeurs par défaut :

```
defaults: new { controller = "Home", action = "
    Index", id = UrlParameter.Optional }
```

Dans ce cas :

- Le contrôleur `HomeController` sera instancié
- La méthode `Index()` sera exécutée

Saisir cette URL `http://localhost:53515/` engendrera l'exécution de la méthode (action) `Index()` définie dans `HomeController`

ASP.NET MVC

Ce contenu

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "
       Index", id = UrlParameter.Optional }
);
```

ASP.NET MVC

Ce contenu

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "
       Index", id = UrlParameter.Optional }
);
```

peut être remplacé par (les clés peuvent être supprimées)

```
routes.MapRoute(
    "Default",
    "{controller}/{action}/{id}",
    new { controller = "Home", action = "Index", id
       = UrlParameter.Optional }
);
```

ASP.NET MVC

Et cela ?

```
routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
```

- permet d'ignorer les requêtes qui ont la forme `/uneChaine.axd/n-importe-quoi` et qui essayent d'accéder à un fichier d'extension `axd`.

ASP.NET MVC

Créons un un deuxième contrôleur `SecondController`

```
public class SecondController : Controller
{
    // GET: Second
    public String Index()
    {
        return "Bonjour Index";
    }
    public String Action()
    {
        return "Bonjour";
    }

}
```

ASP.NET MVC

Les URL suivantes

- /second : affiche `Bonjour Index`
- /second/action : affiche `Bonjour`
- / : affiche `Hello world`
- /second/second : génère une erreur 404

ASP.NET MVC

Il suffit de déclarer un paramètre dans la signature de la méthode

```
public class HomeController : Controller
{
    public string Index(string id)
    {
        return $"Hello {id}";
    }
}
```

ASP.NET MVC

Il suffit de déclarer un paramètre dans la signature de la méthode

```
public class HomeController : Controller
{
    public string Index(string id)
    {
        return $"Hello {id}";
    }
}
```

En allant à `/home/index?id=Wick`, un `Hello Wick` sera affiché.

ASP.NET MVC

Il suffit de déclarer un paramètre dans la signature de la méthode

```
public class HomeController : Controller
{
    public string Index(string id)
    {
        return $"Hello {id}";
    }
}
```

En allant à `/home/index?id=Wick`, un `Hello Wick` sera affiché.
En allant à `/home/index/Wick`, le même message sera affiché.

ASP.NET MVC

Il suffit de déclarer un paramètre dans la signature de la méthode

```
public class HomeController : Controller
{
    public string Index(string id)
    {
        return $"Hello {id}";
    }
}
```

En allant à `/home/index?id=Wick`, un `Hello Wick` sera affiché.
En allant à `/home/index/Wick`, le même message sera affiché.
Oui mais le nom du paramètre `id` n'est pas significatif. On veut l'appeler `nom`

ASP.NET MVC

Solution : modifier le `RouteConfig`

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{nom}",
    defaults: new { controller = "Home", action = "
       Index", nom = UrlParameter.Optional }
);
```

Modifions aussi le contrôleur

```
    public class HomeController : Controller
    {
        public string Index(string nom)
        {
            return $"Hello {nom}";
        }
    }
```

ASP.NET MVC

Attention, quand le paramètre est de type numérique

```
routes.MapRoute(
    name: "Default",
    url: "{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "
       Index", id = UrlParameter.Optional }
);
```

Modifions aussi le contrôleur

```
    public class HomeController : Controller
    {
        public string Index(int id)
        {
            return $"Hello {id}";
        }
    }
```

ASP.NET MVC

Les URL suivantes

- / : génère une erreur
- /home : génère une erreur
- /home/index : génère une erreur
- /home/index/2 : affiche `Hello 2`

Constat

- Le paramètre `id` est obligatoire car un entier ne prend pas la valeur null (il est non `nullable`)

ASP.NET MVC

Pour éviter le problème précédent, il faut modifier la signature de l'action

```
public class HomeController : Controller
{
    public string Index(int ? id)
    {
        return $"Hello {id}";
    }
}
```

ASP.NET MVC

Exercice

- Modifier le programme pour qu'il affiche `n` fois le message `Hello` + `nom` + `prénom`
- `nom`, `prénom` et `n` étant des paramètres récupérés de la requête
- Si `n` n'a pas été précisé par l'utilisateur ou s'il est inférieur ou égal à zéro, le message sera affiché une seule fois

ASP.NET MVC

Si on veut mettre en forme le message ? il faut utiliser HTML

```
public string Index(int? id)
{
    return @"
        <html>
            <head>
                <title> Hello World </title>
            </head>
            <body>
                <p> <b> Hello " + id + @" </b></p>
            </body>
        </html> ";
}
```

ASP.NET MVC

Remarques

- Le contrôleur fait les calculs
- Le contrôleur met en forme la réponse
- HTML + C# dans la même page

ASP.NET MVC

Remarques

- Le contrôleur fait les calculs
- Le contrôleur met en forme la réponse
- HTML + C# dans la même page

Solution

Utiliser des vues (pages HTML) qui seront appelées par le contrôleur

ASP.NET MVC

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

```
RedirectToAction("action", "controller", new {
   paramName = value});
```

ASP.NET MVC

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
```

ASP.NET MVC

Pour récupérer la route courante dans le contrôleur

```
string host = HttpContext.Request.RawUrl;
```

ASP.NET MVC

Pour récupérer la route courante dans le contrôleur

```
string host = HttpContext.Request.RawUrl;
```

Remarque

En allant à `https://localhost:44395/home/index/2`, la variable `host` récupère la valeur `/home/index/2`.

ASP.NET MVC

Autres informations à récupérer

- `HttpContext.Request.HttpMethod` : pour récupérer le verbe `HTTP` utilisé pour exécuter l'action
- `HttpContext.Request.Url.AbsolutePath` : pour récupérer le chemin absolu demandé (comme `/home/index/2`)
- `HttpContext.Request.Url.AbsoluteUri` : pour récupérer l'URI complète (comme `https://localhost:44395/home/index/2`)
- `HttpContext.Request.Url.Host` : pour récupérer l'adresse IP du visiteur
- ...

ASP.NET MVC

Pour récupérer la route courante dans le contrôleur

```
@Request.Url.AbsoluteUri
<!-- affiche https://localhost:44395/Home/Index/2 -->

@Request.RawUrl
<!-- affiche /Home/Index/2  -->
```

ASP.NET MVC

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

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

ASP.NET MVC

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.

ASP.NET MVC

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.

ASP.NET MVC

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

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

ASP.NET MVC

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

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

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