Les sélecteurs d’action en ASP.NET MVC

MOHAMED AIROUCHE

Les sélecteurs d'action

- Un sélecteur d'action est l'attribut qui peut être appliqué aux méthodes d'action.

- Il aide le moteur de routage à sélectionner la bonne méthode d'action pour gérer une requête HTTP.

- MVC 5 inclut les attributs de sélecteur d'action suivants :

  1. ActionName
  2. NonAction
  3. ActionVerbs

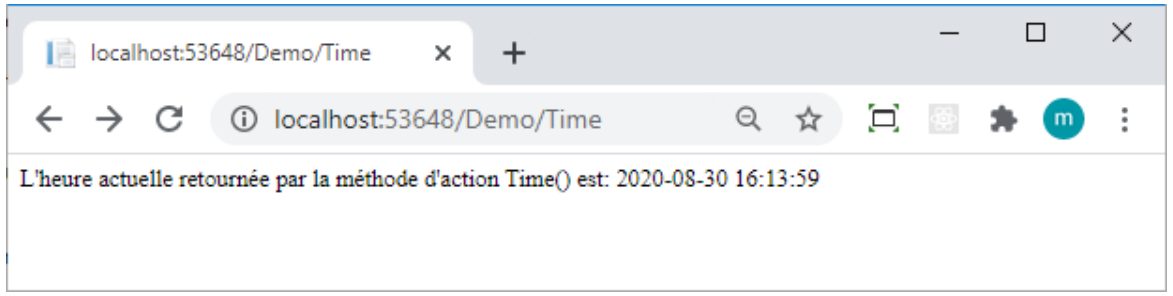
  - L'attribut ActionName est appliqué pour une action pour donner un nouveau nom pour une méthode. Ce nom sera utilisé dans l'URL pour appeler cette méthode.

  - Il permet également aux développeurs d'utiliser un nom d'action différent de celui de la méthode.

Les sélecteurs d'action

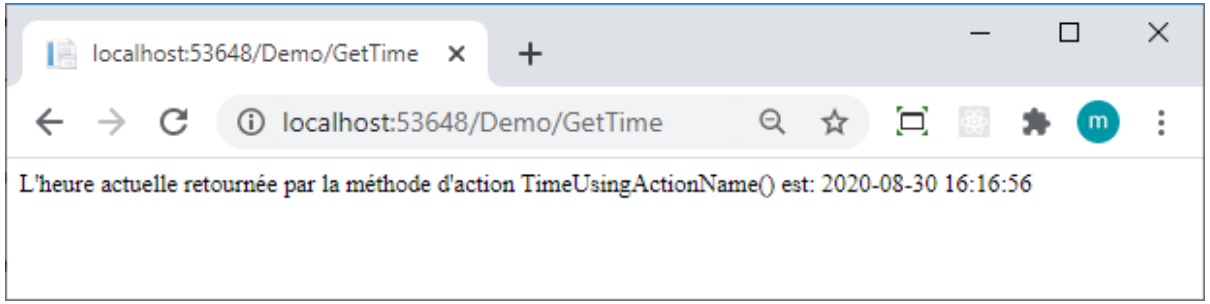
- Exemple de méthode d'action qui n'utilise pas l'attribut ActionName :

```
public String Time()
{
     return @"L'heure actuelle retournée par la
     méthode d'action Time() est : " + DateTime.Now;
}
```



- Exemple de méthode d'action qui utilise l'attribut ActionName :

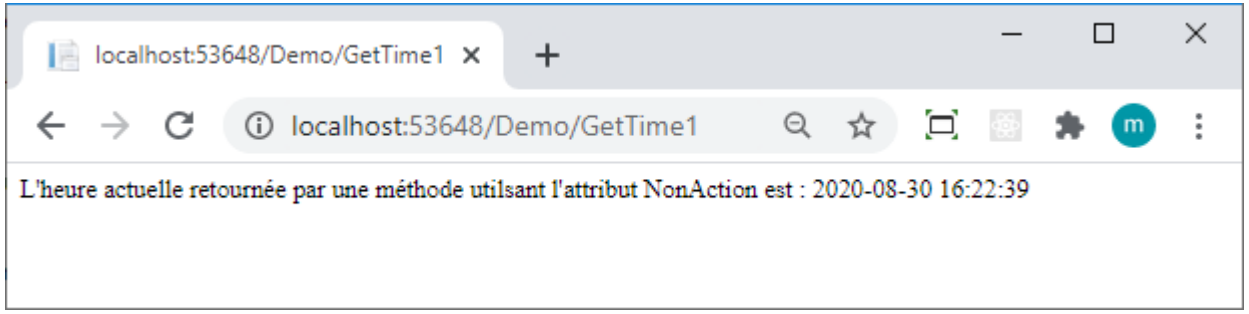
```
[ActionName("GetTime")]
public String TimeUsingActionName()
{
     return @"L'heure actuelle retournée par la
     méthode d'action TimeUsingActionName() est : "
     + DateTime.Now;
}
```



Les sélecteurs d’action

- NonAction est un autre attribut intégré, qui indique qu’une méthode publique d’un contrôleur n’est pas une méthode d’action. Il est utilisé quand vous voulez qu’une méthode ne soit pas traitée comme une méthode d’action.

```
/* exemple montrant l’utilisation de l’attribut NoAction*/
[ActionName("GetTime1")]
public String TimeUsingNonActionMethod()
{
     return GetTimeMethod();
}
[NonAction]
public String GetTimeMethod()
{
   return @"L’heure actuelle retournée par une méthode utilisant NonAction attribut est : «
      + DateTime.Now;
}
```



Les sélecteurs d'action

- Le sélecteur ActionVerbs est utilisé lorsque vous souhaitez contrôler la sélection d'une méthode d'action basée sur une méthode de requête HTTP.

- Par exemple, vous pouvez définir deux méthodes d'action différentes avec le même nom, mais une méthode d'action répond à une requête HTTPGet et une autre méthode d'action répond à une requête HTTPPost.

- Le Framework MVC supporte différents ActionVerbs, tel que :
  1. HttpGet : ne sers que la commande GET.
  2. HttpPost : ne sers que la commande POST.
  3. HttpPut : ne sert que la commande PUT.
  4. HttpDelete : ne sers que la commande DELETE.
  5. HttpOptions : ne sers que la commande OPTIONS.
  6. HttpPatch : ne sers que la commande PATCH.

- Vous pouvez appliquer ces attributs à la méthode action pour indiquer le type de requête HTTP que la méthode d'action prend en charge.
- Si vous n'appliquez aucun attribut, il considère qu'il s'agit d'une requête GET par défaut.

Rappel sur la transmission des données

1. Utilisation de la méthode *GET* pour transmettre les données d'un formulaire

La commande *GET* utilise l'adresse URL pour transmettre les données au serveur. Chaque demande de connexion d'un navigateur à une nouvelle page Web utilise la commande *GET*. Il est possible de rattacher des données à une adresse URL en intercalant un point d'interrogation (?). Les données sont visibles par tous les utilisateurs. Il est principalement utilisé lorsque vous ne publiez pas de données sensibles sur le serveur telles que le nom d'utilisateur, le mot de passe, les informations de carte de crédit, etc.

Note : Par défaut, c'est la méthode *GET* qui est utilisée si la propriété **method** n'est pas spécifiée dans la balise **<form>**.

Si nous ne spécifions pas d'attribut **action**, c'est le fichier courant qui sera de nouveau chargé en mémoire lorsque nous soumettrons le formulaire.

Rappel sur la transmission des données

2. Utilisation de la méthode *POST* pour transmettre les données d'un formulaire

La commande *POST* fonctionne différemment. Cette méthode ne subit pas les limitations des adresses URL. Elle masque les informations de l'URL et ne lie pas les données à l'URL. Elle est plus sécurisée que la méthode GET. Cela est utile lorsque vous transmettez des informations sensibles au serveur.

Le navigateur peut en effet utiliser la commande *POST* pour indiquer au serveur qu'il dispose de données, par exemple du contenu de zones de formulaire.

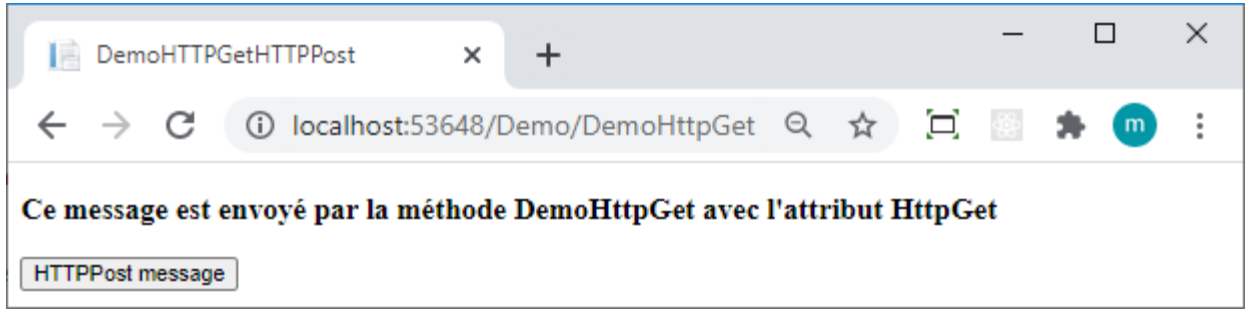
Exemple d’utilisation des sélecteurs d’action

```
/* exemple montrant l’utilisation de l’attribut ActionVerbs*/
[HttpGet]
public ActionResult DemoHttpGet()
{
     ViewBag.message = "Ce message est envoyé par la méthode DemoHttpGet avec l’attribut HttpGet";
     return View();
}

[HttpPost]
public ActionResult DemoHttpPost()
{
     ViewBag.message = "Ce message est envoyé par la méthode DemoHttpPost avec l’attribut HttpPost";
     return View();
}
```

Exemple d'utilisation des sélecteurs d'action

- La vue DemoHttpGet.cshtml :

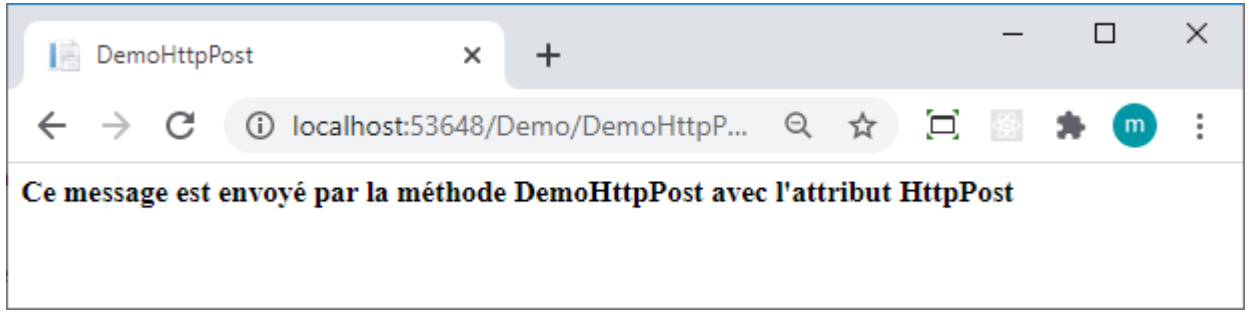



```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>DemoHttpGet</title>
</head>
<body>
   <div>
      <h3>@ViewBag.message</h3>
      @using (Html.BeginForm("DemoHttpPost", "Demo", FormMethod.Post))
      {
         <input type="submit" name="btn" value="HTTPPost message" id="btn" />
      }
      </div>
</body>
</html>
```

Exemple d'utilisation des sélecteurs d'action

- La vue DemoHttpPost.cshtml :




```
<!DOCTYPE html>

<html>
<head>
    <meta name="viewport" content="width=device-width" />
    <title>DemoHttpGet</title>
</head>
<body>
   <div>
      <h3>@ViewBag.message</h3>
   </div>
</body>
</html>
```