ASP.Net MVC

JavaScript, Ajax et Web API

JQuery, JQuery UI, Web API

Visual studio ♥ JavaScript

- ☺ VS fait de l'inférence de type
  - ☺ + Commentaires XML /// deprecated, field, param, returns, summary, var, reference, ...

```
<field name="fieldName" static="true|false" type="FieldType" integer="true|false" domElement="true|false"
  mayBeNull="true|false" elementType="ArrayElementType" elementInteger="true|false" value="code"
  elementDomElement="true|false" elementMayBeNull="true|false" helpKeyword="keyword" locid="descriptionID">
    description
</field>
<param name="parameterName" type="ParameterType" integer="true|false" domElement="true|false" value="code"
  mayBeNull="true|false" elementType="ArrayElementType" elementInteger="true|false" elementDomElement="true|false"
  elementMayBeNull="true|false" locid="descriptionID" parameterArray="true|false" optional="true|false">
    description
</param>
<var type="ValueType" integer="true|false" domElement="true|false" mayBeNull="true|false"
  elementType="ArrayElementType" elementInteger="true|false" elementDomElement="true|false"
  elementMayBeNull="true|false" helpKeyword="keyword" locid="descriptionID">description</var>
<summary locid="descriptionID">description</summary>
<reference path="~/Scripts/bootstrap.js" />
...
```

```
function () {
    $compteur.html(Number($compteur.html()) + 1);
});
);
cript>
```

▲ 1 of 2 ▼ jQuery html(**htmlString htmlString**)
Set the HTML contents of each element in the set of matched elements.
htmlString: *A string of HTML to set as the content of each matched element.*

- ☺ Les références sont regroupées dans ~/_references.js

Intellisence

- Exemple

```
function Complexe(a, b) {
  /// <param name='a' type='Number' mayBeNull='false' optional='false'>Partie réelle.</param>
  /// <param name='b' type='Number' mayBeNull='false' optional='true'>Partie imaginaire.</param>
  /// <summary>Fabrique un nombre imaginaire</summary>

  /// <field name='I' type='Complexe' static='true'>La valeur 'i', sqrt(-1)</field>

  /// <field name='a' type='Number'>Partie réelle</field>
  /// <field name='b' type='Number'>Partie imaginaire</field>
  this.a = a;
  this.b = b ? b : 0;
}
Complexe.prototype.module = function () {
  ///<summary type='Number'>Calcul le module du nombre complexe</summary>
  ///<var type='Number'>a au carré</var>
  var a2 = this.a * this.a;
  ///<var type='Number'>b au carré</var>
  var b2 = this.b * this.b;
  return Math.sqrt(a2 + b2);
};
Complexe.prototype.convertirEnPoint = function () {
  /// <summary>Fabrique un objet {x:a, y:b}</summary>
  /// <returns value='{x:0, y:0}'>Le point</returns>
  return { x: this.a, y: this.b };
}
Complexe.I = new Complexe(0, 1);
```

JQuery / JQuery UI en deux mots

- Framework JavaScript
  - Encapsuler les éléments du DOM HTML
  - Ajout de fonctionnalités
  - Licence MIT
- Installation par package NuGet
  - En Asp.Net MVC, enregistrable comme bundle
  - Référencé par le Layout par défaut
- Extensible
  - Nombreux plugin libres
  - JQuery UI = regroupement de plugins
    - Très gros mais personnalisable

Initialisation de JQuery

- L'objet JQuery $ offre les fonctionnalités de base
  - La recherche d'éléments DOM par filtre CSS retourne un objet JQuery qui est un tableau d'éléments DOM et offre des méthodes pour toucher tous ses éléments

```
<span id="compteur">0</span> <br />
<input type="button" value="Incrémenter" id="incrementer" />
@section scripts {
  <script type="text/javascript">
    $(function () {
      alert("JQuery est initialisé, on peut y aller !");
      var $compteur = $("#compteur");
      $("#incrementer").click(
        function () {
          $compteur.html(Number($compteur.html()) + 1).removeClass("hidden");
      });
    });
  </script>
}
```

JQuery.animate

- Transition de propriétés de style

```
<div class="animer">DANGER !</div>
<div class="animer">DANGER AUSSI !</div>
@section scripts {
<script type="text/javascript">
  $(function () {                                          ← JQuery est initialisé, on peut y aller !
    var lancer = function () {
      $(".animer").animate({ fontSize: '20px', opacity: '1' }, 500, "linear").promise().done(
        function () {
          setTimeout(
            function () {
              $(".animer").animate({ fontSize: '10px', opacity: '0.5' }, 500, "linear").promise().done(
                function () { setTimeout(lancer, 5000); }
                );
            }, 1000);
        }
        );
    }
    lancer();
  });
</script>
}
```

Toggle

- Pour inverser un état

```
<input type="button" value="Montrer" id="btMontrer" class="btToggle" style="display:none" />
<input type="button" value="Cacher" id="btCacher" class="btToggle" />
<div id="zoneAToggler">Exemple de "<em>togglage</em>"</div>

@section scripts {
  <script type="text/javascript">
    function toggler(event) {
    $("#zoneAToggler").toggleClass("invisible");
      $(".btToggle").toggle(3000);
    }
    $(function () {
      $(".btToggle").click(toggler);
    }
</script>
}
```

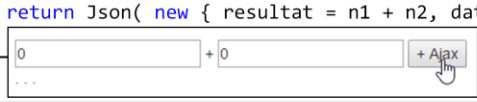
$.ajax

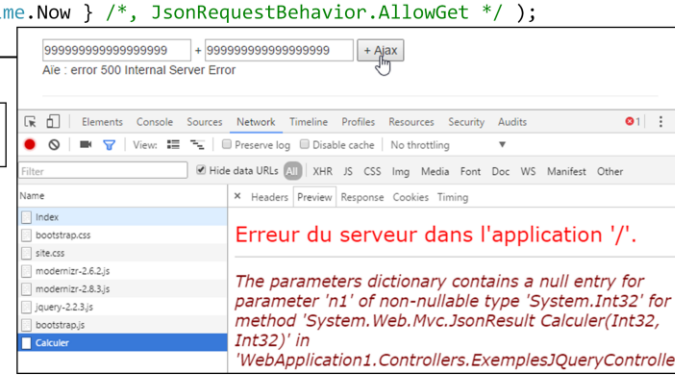
- Calculs http asynchrones
  - Vue

```
<input type="number" id="n1" value="0" /> + <input type="number" id="n2" value="0" />
<input type="button" value="+ Ajax" onclick="additionnerAjax(this)" /> <br />
<div id="resultatAjax"></div>
```

  - Contrôleur

```
public JsonResult Calculer( int n1, int n2 ) {
  Thread.Sleep( 3000 );
  return Json( new { resultat = n1 + n2, date = DateTime.Now } /*, JsonRequestBehavior.AllowGet */ );
}
```





$.ajax

- Calculs http asynchrones
  - JavaScript

```
function additionnerAjax(btSender) {
  ///<param name='btSender' value='document.createElement("input")' />
  btSender.disabled = true;
  var $resultatAjax = $("#resultatAjax");
  $resultatAjax.html( ". . ." );
  $.ajax( {
      url: "@Url.Action( "Calculer" )",
      method: "POST",            ← Asp.Net MVC n'accepte que le POST sans l'option "JsonRequestBehavior.AllowGet"
      data: { n1: Number($("#n1").val()), n2: Number($("#n2").val()) }
    } )
  .success(function (reponse, messageSuccess, objetRequete) {
    ///<param name='reponse' value="{resultat: 0, date:'/Date(123456)/'}" />
    btSender.disabled = false;
    ///<field name="date" type="Date" />
    var date =  eval( "new " + reponse.date.replace( /\//g, "" ) );
    $resultatAjax.html("OK : " + messageSuccess + " " + objetRequete.status +
      " résultat = " + reponse.resultat + " date brute " + reponse.date +
      " date : " + date.toLocaleString());
  })
  .error(function(objetRequete, messageError, strStatus ) {
    btSender.disabled = false;
    $resultatAjax.html( "Aïe : " + messageError + " " + objetRequete.status + " " + strStatus );
  });
}
```

$.ajax

- Mises à jour partielles asynchrones
  - Contrôleur

```
public PartialViewResult MettreAJour( int nb ) {
  ViewBag.Nb = nb;
  return PartialView();
}
```

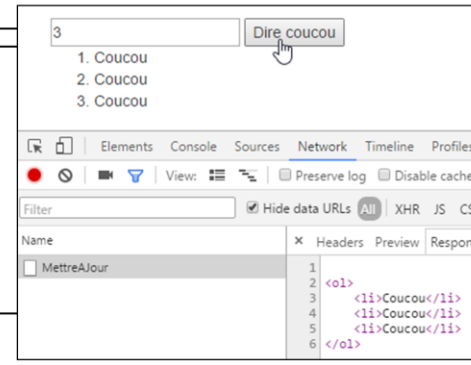
  - Vue **partielle**

```
@{ int nb = ViewBag.Nb; }
<ol>
  @for( int i = 0; i < nb; i++ ) {
    <li>Coucou</li>
  }
</ol>
```

  - Vue principale

```
<input type="number" id="nbCoucou" value="0" />
<input type="button" value="Dire coucou" onclick="DireCoucouAjax()" />
<div id="zoneMaJPartielle"></div>
```

```
function DireCoucouAjax() {
  $.ajax( {
    url: "@Url.Action( "MettreAJour" )",
    method: "POST",
    data: { nb: Number($("#nbCoucou").val()) }
  })
  .success(function (fragmentHtmlResultat) {
    $("#zoneMaJPartielle").html(fragmentHtmlResultat);
  });
}
```



Ajax et validation

- Déclencher la validation côté client
  - Les contrôles à valider doivent être dans un formulaire

```
$("form").valid();
```

- Si l'ajax retourne du Html que vous injectez dans un form ➔ forcer l'interprétation des attributs de validation des nouveaux éléments

```
$.validator.unobtrusive.parse("form");
```

  - Pas nécessaire si vous mettez à jour vous-même le contenu des contrôles avec les champs de l'objet JSON reçu par ajax
- Effacer les erreurs

```
$(".field-validation-error").removeClass("field-validation-error").addClass("field-validation-valid").html("");
```

Helper Ajax

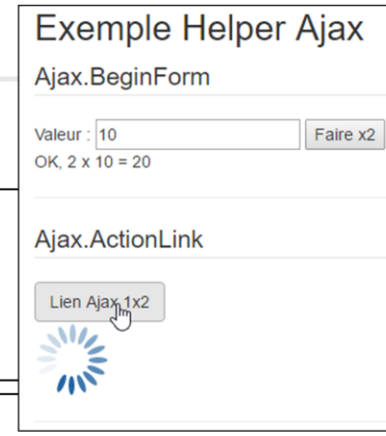
- Moins de contrôle qu'avec $.ajax…
- Demande le script jquery.unobtrusive-ajax.js
  - Package NuGet Microsoft.jQuery.Unobtrusive.Ajax

Ajax. … (très nombreuses surcharges)	Info
`MvcForm BeginForm( this AjaxHelper ajaxHelper, string actionName, string controllerName, object routeValues, AjaxOptions ajaxOptions, object htmlAttributes )`	La soumission du formulaire sera asynchrone. Les paramètres supplémentaires sont passés par l'objet routeValues. L'élément à mettre à jour est précisé dans les AjaxOptions.
`MvcHtmlString ActionLink( this AjaxHelper ajaxHelper, string linkText, string actionName, string controllerName, object routeValues, AjaxOptions ajaxOptions, object htmlAttributes )`	Le lien déclenche l'action en manière asynchrone. L'élément à mettre à jour est précisé dans les AjaxOptions.

- Membres de la classe `AjaxOptions`
  - `AllowCache, Confirm, HttpMethod, **InsertionMode**, LoadingElementDuration, LoadingElementId, OnBegin, OnComplete, OnFailure, **OnSuccess**, **UpdateTargetId**, Url`

Helper Ajax

- Exemple

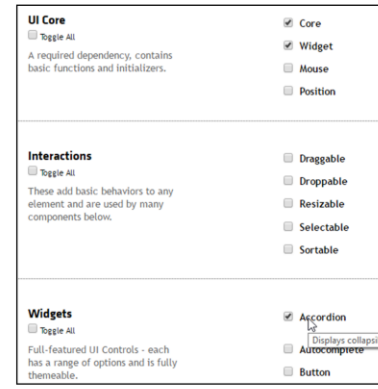


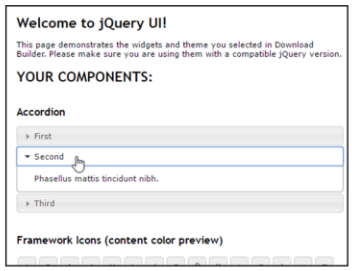

```
public class ExempleHelperAjaxController : Controller {
  public ActionResult Index() { return View(); }
  public ActionResult ActionAjax( int valeur, int fois ) {
    Thread.Sleep( 1500 );
    return Content( $"OK, {fois} x {valeur} = {fois * valeur}" );
  }
}
```

```
<fieldset> <legend>Ajax.BeginForm</legend>
  @using( Ajax.BeginForm( "ActionAjax", "ExempleHelperAjax", new { fois = 2 },
    new AjaxOptions() { HttpMethod = "POST", InsertionMode = InsertionMode.Replace, UpdateTargetId="DivResultat"},
    new { id = "formAjax" } ) ) {
    <span>Valeur : </span><input type="text" name="valeur" value="" />
    <input type="submit" value="Faire x2" />
  }
  <div id="DivResultat"><br />  </div>
</fieldset>
<hr />
<fieldset> <legend>Ajax.ActionLink</legend>
  @Ajax.ActionLink("Lien Ajax 1x2", "ActionAjax", "ExempleHelperAjax", new { valeur = 1, fois = 2 },
    new AjaxOptions() { LoadingElementId = "Patientez", UpdateTargetId = "DivResultat2",
    InsertionMode = InsertionMode.Replace }, new { @class = "btn btn-default" } )
</fieldset>
<img src="~/Images/Patientez.gif" id="Patientez" style="display:none" />
<div id="DivResultat2"></div>
@section scripts {
  <script src="~/Scripts/jquery.unobtrusive-ajax.js"></script>
}
```

Deux mots de JQuery UI

- Ensemble d'éléments d'interface utilisateur, d'effets, de widgets, et de thèmes
  - Bâtis sur JQuery
  - Gros package NuGet
    - Alternative : le site permet de fabriquer un sous-ensemble personnalisé
      - ☺ Dépendances automatiquement cochées !
    - Page de test





Types	Éléments
Interactions	Draggable, Droppable, Resizable, Selectable, Sortable
Widgets	Accordion, Autocomplete, Button, Datepicker, Dialog, Menu, Progressbar, Selectmenu, Slider, Spinner, Tabs, Tooltip
Effects	Add Class, Color Animation, Easing, Effect, Hide, Remove Class, Show, Switch Class, Toggle, Toggle Class
Utilities	Position, Widget Factory

Deux mots de JQuery UI

- Créer son bundle (si téléchargement maison)

~/App_Start/BundleConfig.cs

```
bundles.Add( new StyleBundle( "~/Content/JQueryUI" ).Include(
  "~/Scripts/jquery-ui-1.11.4.custom/jquery-ui.css",
  "~/Scripts/jquery-ui-1.11.4.custom/jquery-ui.structure.css",
  "~/Scripts/jquery-ui-1.11.4.custom/jquery-ui.theme.css"
  ) );
bundles.Add( new ScriptBundle( "~/bundles/jquery-ui" ).Include(
  "~/Scripts/jquery-ui-1.11.4.custom/jquery-ui.js" ) );
```

- Référencer son bundle en cas de besoin

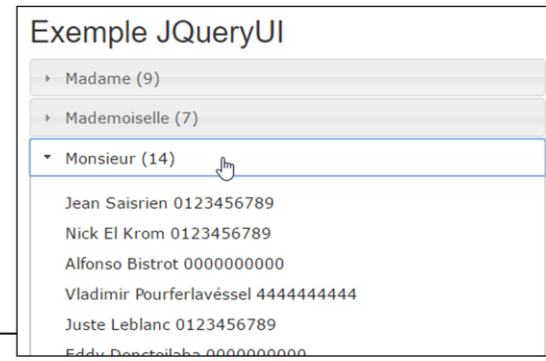
```
@section head {
  @Styles.Render( "~/Content/JQueryUI" )
}

@section scripts {
  @Scripts.Render( "~/bundles/jquery-ui" )
}
```

Deux mots de JQuery UI

- Exemple

```
public ActionResult Index() {
  using( var dc = new AnnuaireEntities() ) { return View( dc.Titres.Include( t => t.Personnes ).ToList() ); }
}
```

```
@using WebApplication1.Models
@{ ViewBag.Title = "Exemple JQueryUI";
  IEnumerable<Titre> titres = Model;
}
@section head { @Styles.Render( "~/Content/JQueryUI" ) }
<h2>@ViewBag.Title</h2>
<div class="panel" id="accordion">
  @foreach( var t in titres ) {
    <div class="panel-heading"> @($"{t.Civilite} ({t.Personnes.Count})") </div>
    <div class="panel-body">
      @foreach( var p in t.Personnes ) {
        <p>@($"{p.Prenom} {p.Nom} {p.Telephone}")</p>
      }
    </div>
  }
</div>
@section scripts {
  @Scripts.Render( "~/bundles/jquery-ui" )
  <script type="text/javascript">
    $("#accordion").accordion();
  </script>
}
```



Exercice

- Filtrer les personnes et les parcourir
  - Utiliser EF
  - Afficher une personne à la fois, ordre Personne.Id
- Ajouter un mode édition pour le téléphone
  - Impossible de se déplacer tant qu'on n'a pas validé ou abandonné
  - La mise à jour doit être faite en ajax
- Une crise de courage ?
  - Éditer tous les champs !

Web API

- Services HTTP facile
  - Tous types de clients : JS, Clients lourds, mobiles
  - Idéal pour construire des services REST full avec .Net
    - REpresentational State Transfer
      - L'URL pointe le contrôleur et l'objet
      - Le verbe HTTP exécute l'action
    - Le routage Asp.Net MVC est bien utile
- Mise en œuvre
  - Contrôleur héritant de `ApiController`
  - Après la création du 1er Web API Controller, ajouter dans Global.asax.cs

```
System.Web.Http.GlobalConfiguration.Configure( WebApiConfig.Register );
```

Web API - Exemple

- Structure + lectures (verbe GET)

```
public class ExempleProductsWebApiController : ApiController {

  private NorthwindEntities DC = new NorthwindEntities();
  protected override void Dispose( bool disposing ) {
    if( disposing )
      DC.Dispose();
    base.Dispose( disposing );
  }

  public IQueryable<object> GetProducts() {                          ← Convention : préfixer par le verbe HTTP
    return DC.Products.Select( p => new { p.ProductID, p.ProductName } );   ← View Model de flémard...
  }

  public IHttpActionResult GetProduct( int id ) {
    var productVM = DC.Products.Select( p =>
      new {                                                          ← View Model de flémard...
        p.ProductID,
        p.ProductName,
        p.Category.CategoryName,
        p.Category.Description,
        p.UnitPrice,
        p.UnitsInStock } )
      .FirstOrDefault( p => p.ProductID == id );
    if( productVM == null ) return NotFound();
    return Ok( productVM );
  }
```

Web API - Exemple

- Opération d'écriture (verbes POST, PUT & DELETE)

```
  public IHttpActionResult PutProduct( int id, Product product ) {
    if( !ModelState.IsValid || id != product.ProductID ) return BadRequest( ModelState );
    DC.Entry( product ).State = EntityState.Modified;
    try {
      DC.SaveChanges();
    }
    catch( DbUpdateConcurrencyException ) {
      if( DC.Products.Any( p => p.ProductID == id ) ) throw;
      return NotFound();
    }
    return StatusCode( HttpStatusCode.NoContent );
  }

  public IHttpActionResult PostProduct( Product product ) {                    ← View Model de flémard...
    if( !ModelState.IsValid ) return BadRequest( ModelState );
    DC.Products.Add( product );
    DC.SaveChanges();
    return CreatedAtRoute(                                                     ← URI vers la ressource créée
      "DefaultApi", new { id = product.ProductID }, product );                 ←"DefaultApi" est le nom de la route
  }

  public IHttpActionResult DeleteProduct( int id ) {
    DC.Entry( new Product() { ProductID = id } ).State = EntityState.Deleted;
    if( DC.SaveChanges() != 1 ) return NotFound();
    return Ok( id );
  }
}
```

Web API

- Utilisation
  - En JavaScript
    - ☹ Envoyer une requête HTTP/JSON avec le bon verbe
    - ☹ L'URI REST rend les choses moins désagréables
    - ☹ La génération d'un éventuel proxy doit être faite manuellement
  - En C#
    - ☹ La même misère qu'en JS
    - ☹ On utilise classiquement un `HttpClient`, `WebClient` ou `HttpWebRequest`
    - ☹ On peut aussi utiliser Swagger (http://swagger.io/)
      - ☺ Swagger permet également la génération de documentation https://docs.asp.net/en/latest/tutorials/web-api-help-pages-using-swagger.html
  - En résumé
    - On regrette bien les Web Services *à Papa* (ASMX) + MS Ajax !

Web API – Exemple de client JS

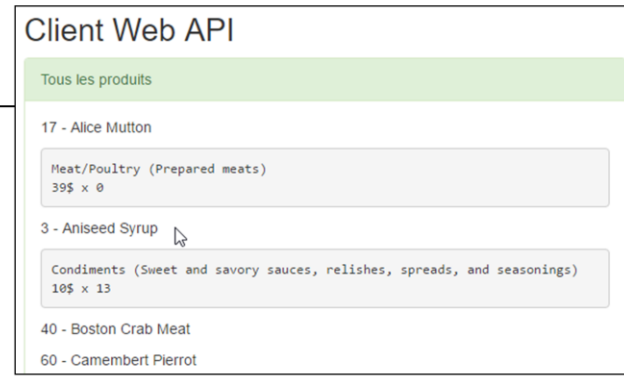
- Remonter une liste

```
<div class="panel panel-success">
  <div class="panel-heading">Tous les produits</div>
  <div class="panel-body" id="TousLesProduits"></div>
</div>

@section scripts {
  <script type="text/javascript">
    var urlWebApiProducts = "/api/ExempleProductsWebApi";
    $(function () {
      $.getJSON(urlWebApiProducts)                          ← Idem $.ajax, fixe le paramètre dataType à 'json'
        .done(function (produits) { /// <param name="produits" value="[{ProductID:0, ProductName:''}]" />
          $.each(produits, function (key, p) {
            $('<p>', { text: p.ProductID + " - " + p.ProductName, id: "p" + p.ProductID })
              .click(function () { afficherDetails( p.ProductID ) })
              .appendTo($('#TousLesProduits'));
          });
        }
      );
    });
  </script>
  [ . . . ]
}
```

Web API – Exemple de client JS

- Remonter un objet

```
@section scripts {
  <script type="text/javascript">
    [ . . . ]
    function afficherDetails(id) {
      $.getJSON(urlWebApiProducts + "/" + id)
        .done(function (p) {
          /// <param name="p" value="{ProductID:0, ProductName:'', CategoryName:'', Description:'', UnitPrice:0.0,
UnitsInStock:0}" />
          $("#p" + p.ProductID)
          .after( $("<pre>", { text: p.CategoryName + " (" + p.Description + ")\n"
                                                    + p.UnitPrice + "$ x " + p.UnitsInStock }) )
          .unbind("click")                                    ← On ne requête plus
          .click(function () { $(this).next().toggle() });    ← Déjà calculé, désormais on affiche ou on masque
        });
      }
  </script>
}
```



Web API – Exemple de client .Net

- Utiliser un proxy

```
var proxy = new ExempleProductsWebApiClientDotNet();
var produitsDispo = proxy.ListerProduits();
foreach( var p in produitsDispo )
  Console.WriteLine( "{0,4} - {1}", p.ProductId, p.ProductName );
Console.WriteLine();
Console.Write( "Produit N°1 :" );
var p1 = proxy.LireProduit( 1 );
Console.WriteLine("{0} {1}\n{2} ({3})\n{4:0.00} x {5}", p1.ProductID, p1.ProductName, p1.CategoryName,
                                                        p1.Description, p1.UnitPrice ?? 0, p1.UnitsInStock ?? 0 );
Console.WriteLine();
Console.WriteLine("Ajout du produit test :");
var nv = new ProduitDeListe() { ProductName = "Test" };
var uri = proxy.AjouterProduit( nv );
Console.WriteLine( "  Le nouveau produit porte l'id {0}", nv.ProductId );
Console.WriteLine( "  Il est disponible ici : {0}", uri );
```

```
  17 - Alice Mutton
   3 - Aniseed Syrup
[ . . . ]
Produit N°1 : 1 Chai
Beverages (Soft drinks, coffees, teas, beers, and ales)
18,00 x 39

Ajout du produit test :
  Le nouveau produit porte l'id 1089
  Il est disponible ici : http://localhost:52599/api/ExempleProductsWebApi/1089
```

Web API – Exemple de client .Net

- Créer les modèles à la main…

```
class ProduitDeListe {
  public int ProductId { get; set; }
  public string ProductName { get; set; }
}

class ProduitDetaillé {
  public int ProductID { get; set; }
  public string ProductName { get; set; }
  public string CategoryName { get; set; }
  public string Description { get; set; }
  public decimal? UnitPrice { get; set; }
  public short? UnitsInStock { get; set; }
}
```

- Créer le proxy à la main…

Web API – Exemple de client .Net

```
class ExempleProductsWebApiClientDotNet : IDisposable {
  private JavaScriptSerializer Serialiseur = new JavaScriptSerializer();   ← Référencer System.Web.Extensions.dll
  private HttpClient Client = new HttpClient();
  public ExempleProductsWebApiClientDotNet() {
    Client.BaseAddress = new Uri( "http://localhost:52599/api/ExempleProductsWebApi/" );
    Client.DefaultRequestHeaders.Accept.Clear();
    Client.DefaultRequestHeaders.Accept.Add( new MediaTypeWithQualityHeaderValue( "application/json" ) );
  }
  public IList<ProduitDeListe> ListerProduits() {
    HttpResponseMessage reponse = Client.GetAsync( string.Empty ).Result;
    reponse.EnsureSuccessStatusCode();
    string chaineJson = reponse.Content.ReadAsStringAsync().Result;
    return Serialiseur.Deserialize<List<ProduitDeListe>>( chaineJson );
  }
  public ProduitDetaillé LireProduit( int id ) {
    var json = Client.GetStringAsync( id.ToString() ).Result; ← Alternative à reponse.Content.ReadAsStringAsync()
    return Serialiseur.Deserialize<ProduitDetaillé>( json );
  }
  public Uri AjouterProduit( ProduitDeListe p ) {
    string json = Serialiseur.Serialize( p );
    var reponse = Client.PostAsync(string.Empty,new StringContent(json,Encoding.UTF8,"application/json")).Result;
    reponse.EnsureSuccessStatusCode();
    var produitRetourné = Serialiseur.Deserialize<ProduitDeListe>( reponse.Content.ReadAsStringAsync().Result );
    p.ProductId = produitRetourné.ProductId;
    return reponse.Headers.Location;     ← L'uri de la nv ressource, fabriquée par le service avec CreatedAtRoute
  }
  public void Dispose() {
    Client.Dispose();
  }
}
```