

Programmation avancée en c#.NET

Ing.Meryem OUARRACHI

Plan du module

Programmation WEB

- ☐ Généralités outils Web
- ☐ **ASP MVC**
- ☐ ASP.Net core
- ☐ Angular

Génie logiciel en .Net

BI en Self Service

Programmation distribuée avancée

- ☐ Web API
- ☐ GraphQL

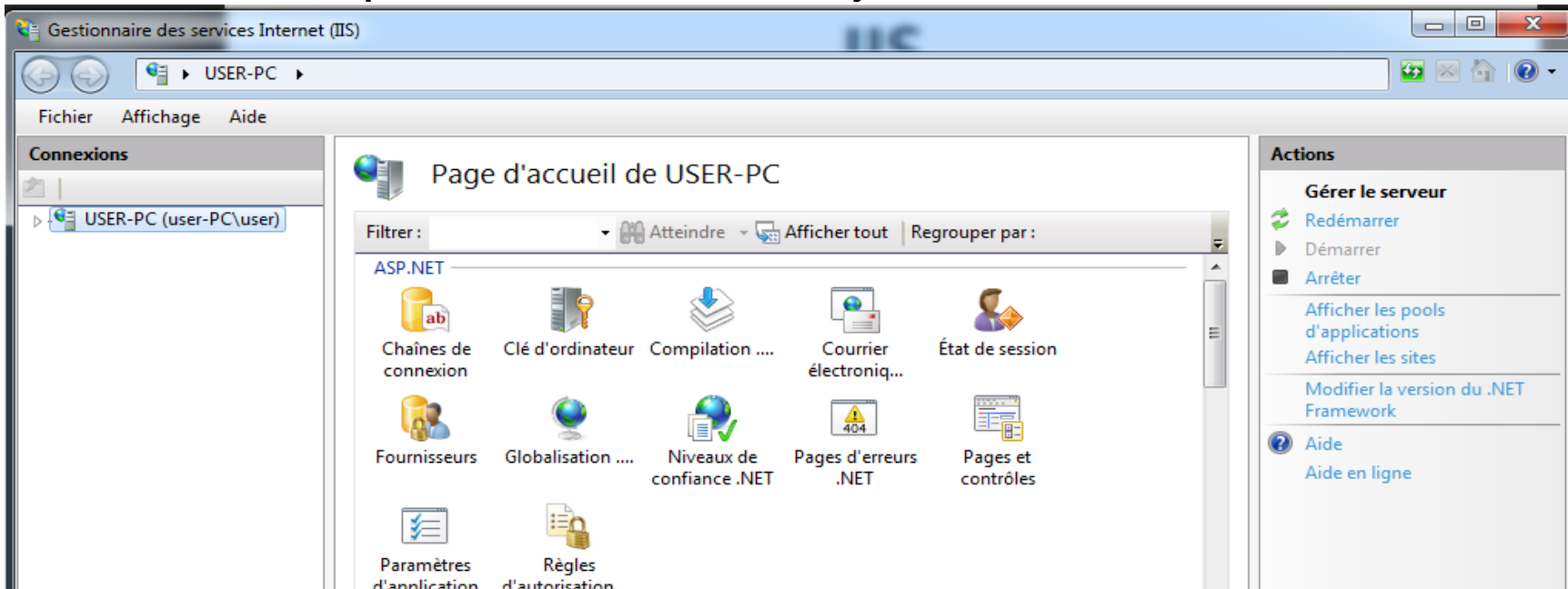
ASP WebForms

Objectif

- .Net possède un ensemble de fonctionnalités dédiées à la création et à la gestion de sites Web.
- ASP.NET permet de créer des sites Web dynamiques.
- IIS est le serveur web pour les projets web en .Net

IIS

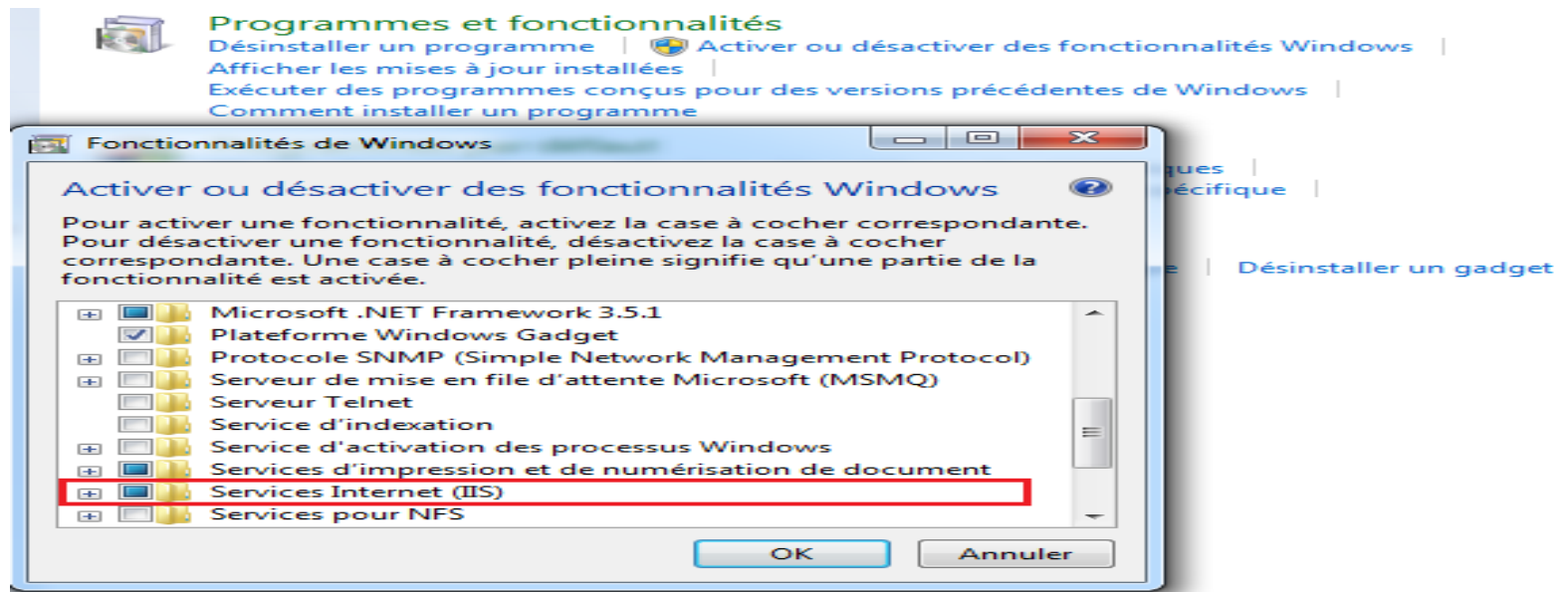
- S'installe avec le système d'exploitation windows
- Pour ouvrir IIS → taper **inetmgr** sur la barre de recherche mais assurer que le service est déjà activé



IIS

-Activation de IIS

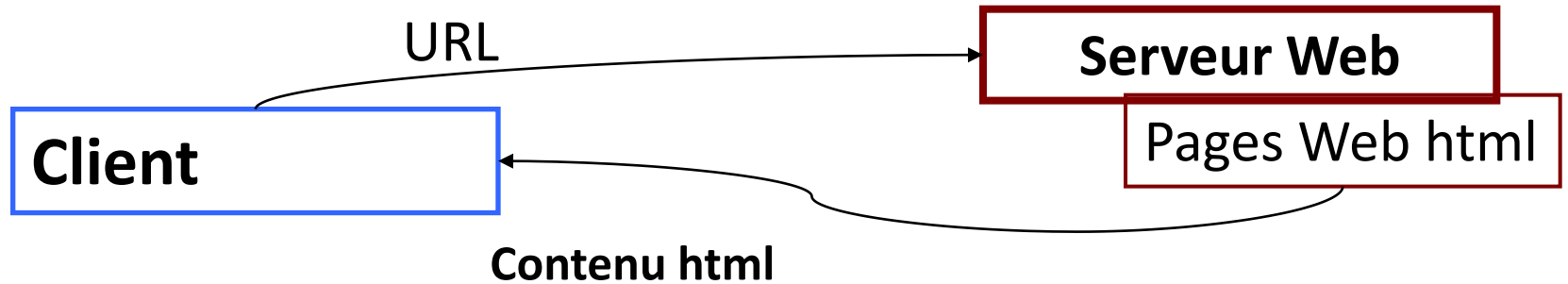
Panneau de configuration → ajouter supprimer des programmes → Activer ou désactiver des fonctionnalités windows



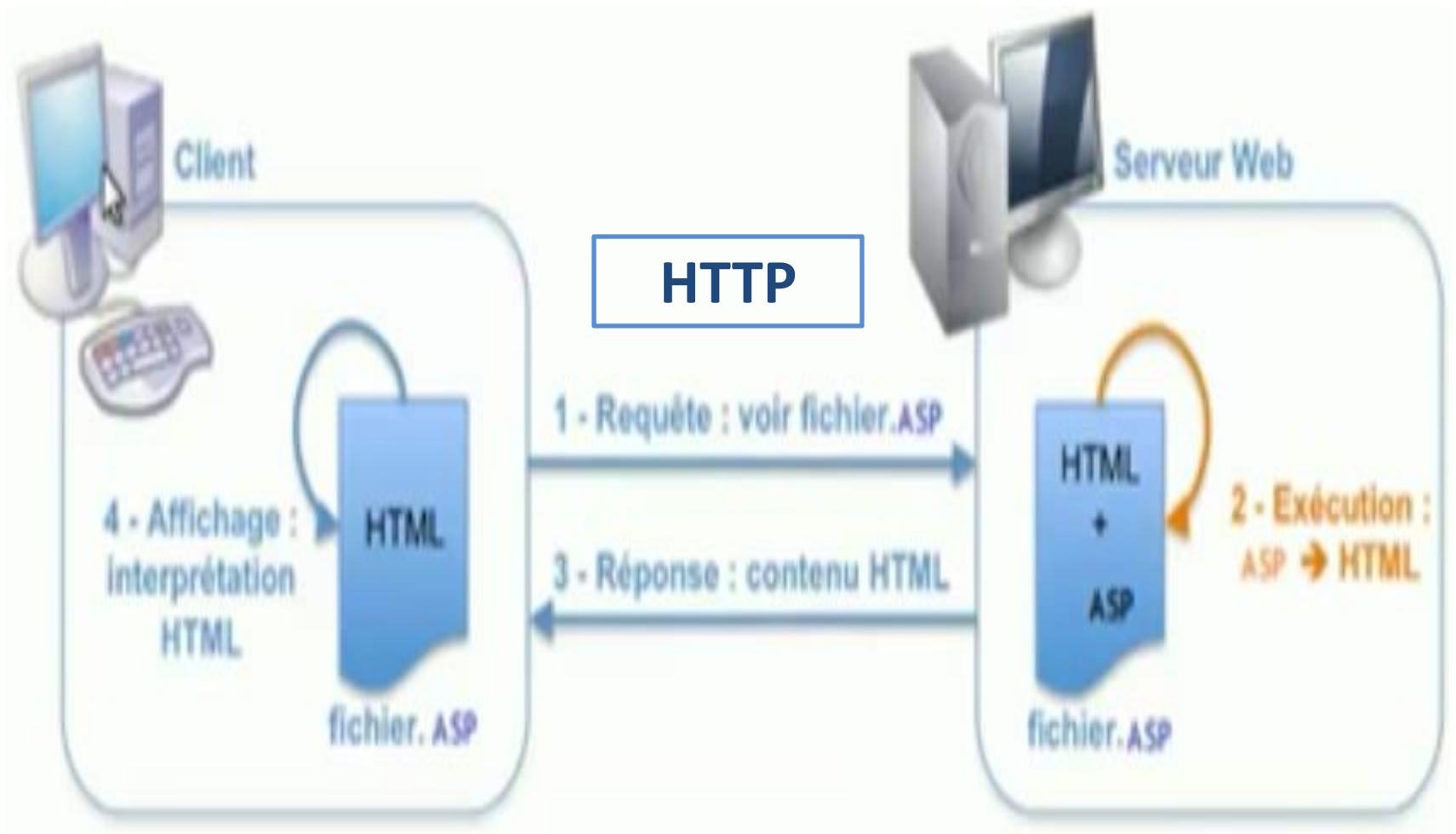
Privilèges d'ASP

- ASP webForms a une bonne structure et architecture pour la programmation(Bibliothèque située à une place, les pages Web à une autre, le code à une autre).
- Plus sécurisée :Selon le *cabinet de sécurité WhiteHat Security*, la sécurité est légèrement meilleure avec *ASP .NET* qu'avec *JSP*, à cause du fait qu'il y a une meilleure orientation de la sécurité pour les développeurs. Mais, les chiffres sont très proches l'un de l'autre, la densité de vulnérabilité est de 27,2 pour le .NET et de 30,0 pour le Java.
- Solution utilisée par des nombreux gouvernements et institutions financières.

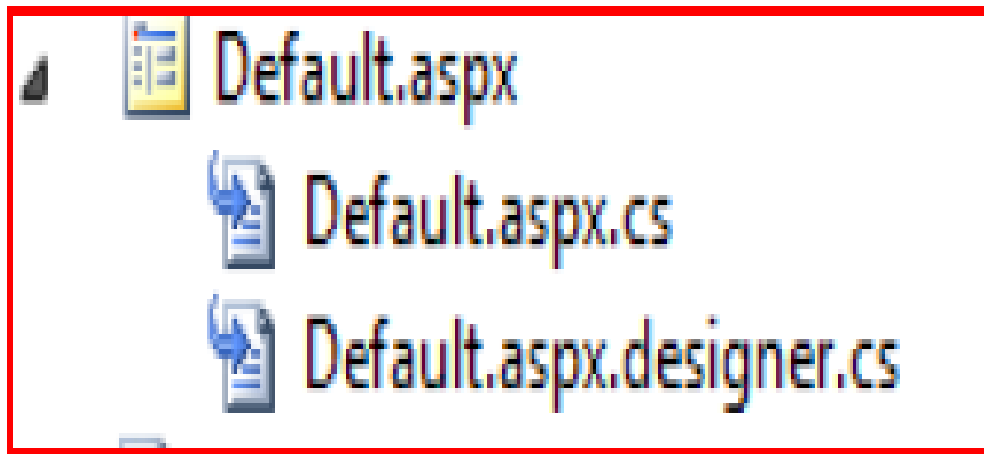
Architecture WEB, site statique



Architecture WEB, site dynamique



Structure d'un projet ASP



En tête et Directive

- Le Doctype est exactement le même qu'en HTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

- L'ASP.NET a aussi sa ligne qui définit certains paramètres

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebForm5.aspx.cs"  
Inherits="WebApplication5.WebForm5" %>
```

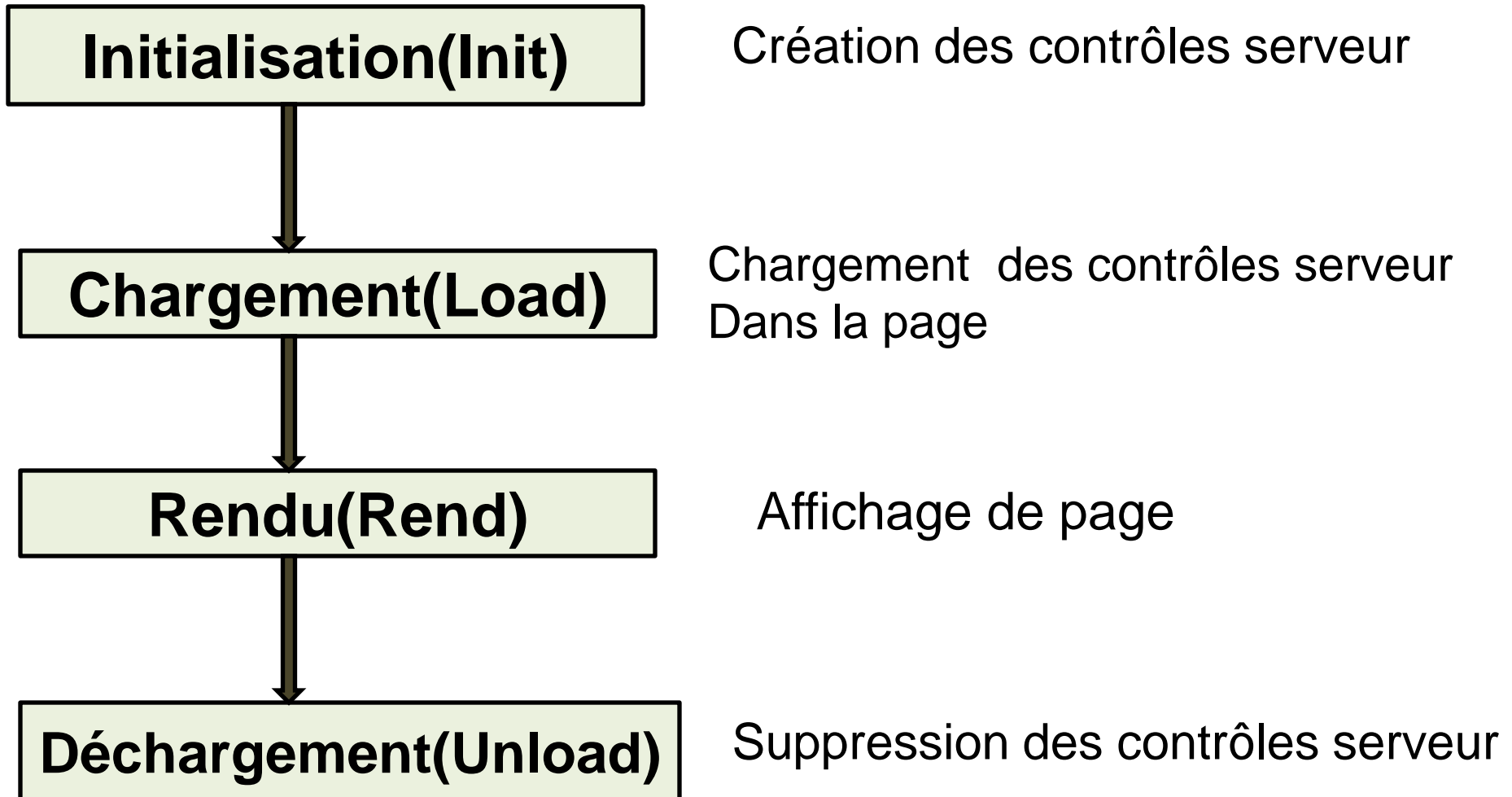
En tête et Directive

- On va utiliser le C# comme langage `Language="C#"`.
- Le nom de la page du code behind, correspondant à cette page `.aspx`
- On définit le namespace et le nom de la classe lié à la page.
- Behind: est le code dans lequel on écrira les instructions qui seront exécutées sur le serveur.

Contrôle Serveur

- Un contrôle serveur est une balise (HTML ou ASP) associé à la propriété `runat="server"`.
- Cette balise sera associé à un objet lors de l'exécution du code par le serveur.
- Un contrôle serveur est programmable depuis le code behind pour répondre à des événements.

Cycle de vie d'une page ASP



La notion dePostBack

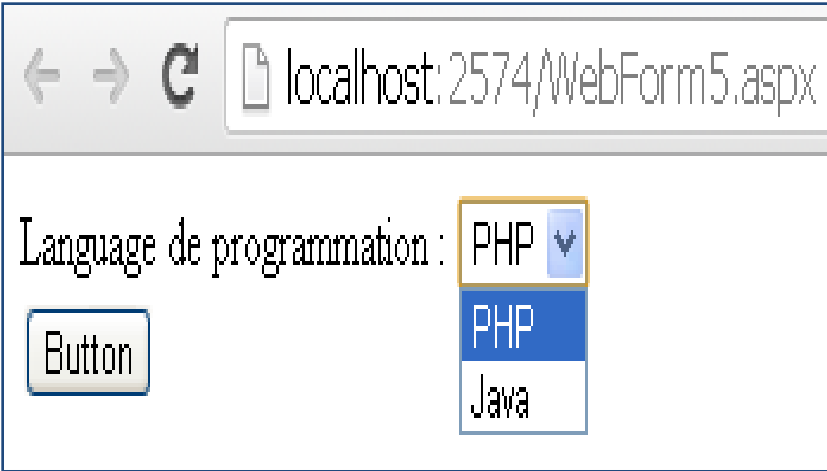
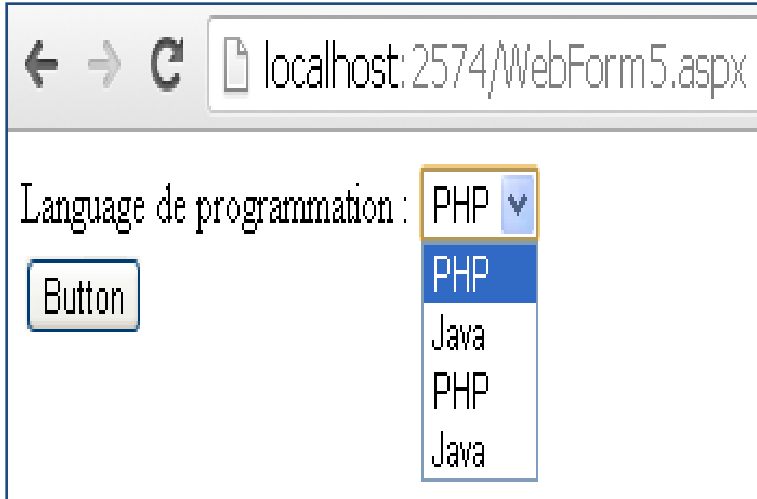
-Il s'agit d'une publication HTTP Post qui consiste à envoyer une mise à jour au navigateur

Exemple1:

```
public partial class WebForm5 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DropDownList1.Items.Add("PHP");
        DropDownList1.Items.Add("Java");
    }
}
```

La notion dePostBack

Résultat:

Lors du premier chargement de page	Après le click sur le bouton
 <p>Language de programmation : PHP ▼</p> <p>Button</p>	 <p>Language de programmation : PHP ▼</p> <p>Button</p>

La notion dePostBack

- La propriété `Page.IsPostBack`: teste si c'est le premier ou deuxième envoi de la page
- `Page.IsPostBack=True`**: il s'agit d'une publication
- `Page.IsPostBack=false`**: Premier accès

ViewState

Exemple2:

```
<asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>

protected void Page_Load(object sender, EventArgs e)
{
}

protected void Button1_Click(object sender, EventArgs e)
{
    Label1.Text = "bonjour";
}

protected void Button2_Click(object sender, EventArgs e)
{
}
```

Résultat:

Label	Button	Button	Premier chargement
<hr/>			
bonjour	Button	Button	Click sur button1 qui change le texte de label
<hr/>			
bonjour	Button	Button	Click sur le button2 qui ne déclenche aucun événement

ViewState

- Le ViewState est un système de maintien de la persistance des données pour les pages ASP.NET.
- La classe Control contient un attribut appelé « ViewState » qui est un dictionnaire d'état des objets qui contient tous les états des contrôles.
- Restaure automatiquement les états des contrôles seulement quand la page est rechargée ➔ PostBack.

Html généré la première fois

```
<form method="post" action="WebForm6.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTc1NDI3NTEyM2Rk9jcYiMA8MK/2MzjZIHUrLfx+05vu3aVrPaEot40iZA=" />
</div>

<div class="aspNetHidden">

<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWAwKZtJnsDAKM54rGBgK7q7GGC07mVqpXE5V+1W12NraDDakYterzeb+grjPhV5lma1bG" />
</div>
<div>
<span id="Label1">Label</span>
<input type="submit" name="Button1" value="Button" id="Button1" />
<input type="submit" name="Button2" value="Button" id="Button2" />
</div>
</form>
```

Html généré la deuxième fois

```
<form method="post" action="WebForm6.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKMTc1NDI3NTEyMw9kFgICAw9kFgICAQ8PFgIeBFRleHQFB2JvbmpvdXJkZGQGYfJ322PFhFqx0TwJg+kJT/zUQY/mo6HB6uKUQr9NQg==" />
</div>

<div class="aspNetHidden">

<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEWAwKkqcuZAQKM54rGBgK7q7GGCJP6chPUQuKteG4fTowK0qY/6YZOW4ptlAT61nYvK1YY" />
</div>
<div>
<span id="Label1">bonjour</span>
<input type="submit" name="Button1" value="Button" id="Button1" />
<input type="submit" name="Button2" value="Button" id="Button2" />
</div>
</form>
```

ViewState

Exemple2:

Si on désactive le viewState « **EnableViewState=false** »

Label Premier chargement

bonjour Click sur button1 qui change le texte de label

Label Click sur le button2 qui ne déclenche aucun événement

Navigation entre les pages

-Navigation directe: en utilisant un lien hypertexte(tags a href)
ou quelques composants d'ASP:

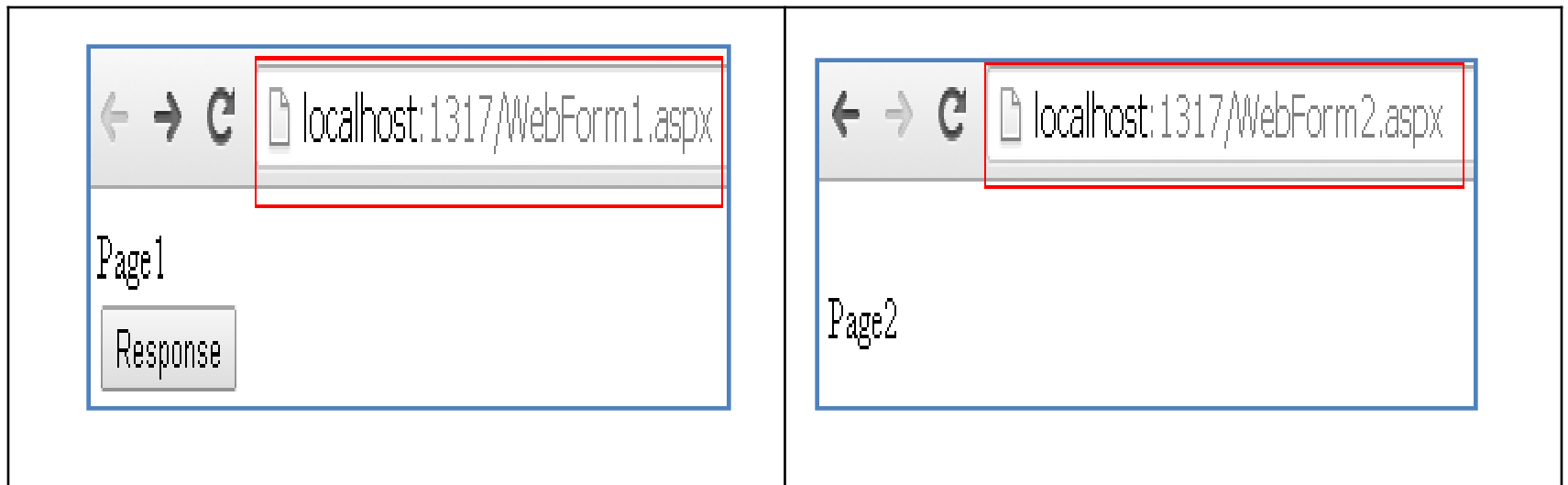
```
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/WebForm2.aspx">HyperLink  
</asp:HyperLink>  
<asp:LinkButton ID="LinkButton1" runat="server" PostBackUrl="~/WebForm2.aspx">LinkButton  
</asp:LinkButton>  
<asp:Button ID="Button1" runat="server" PostBackUrl="~/WebForm2.aspx" Text="Button" />
```

-Navigation basée sur le code: en utilisant **Response.Redirect**
ou **Server.Transfer** .

Response.Redirect

- L'objet **Response** permet ainsi de manipuler l'ensemble des informations à destination au navigateur du client.
- La méthode **Response.Redirect** envoie au navigateur client l'adresse de la page vers laquelle il doit être redirigé

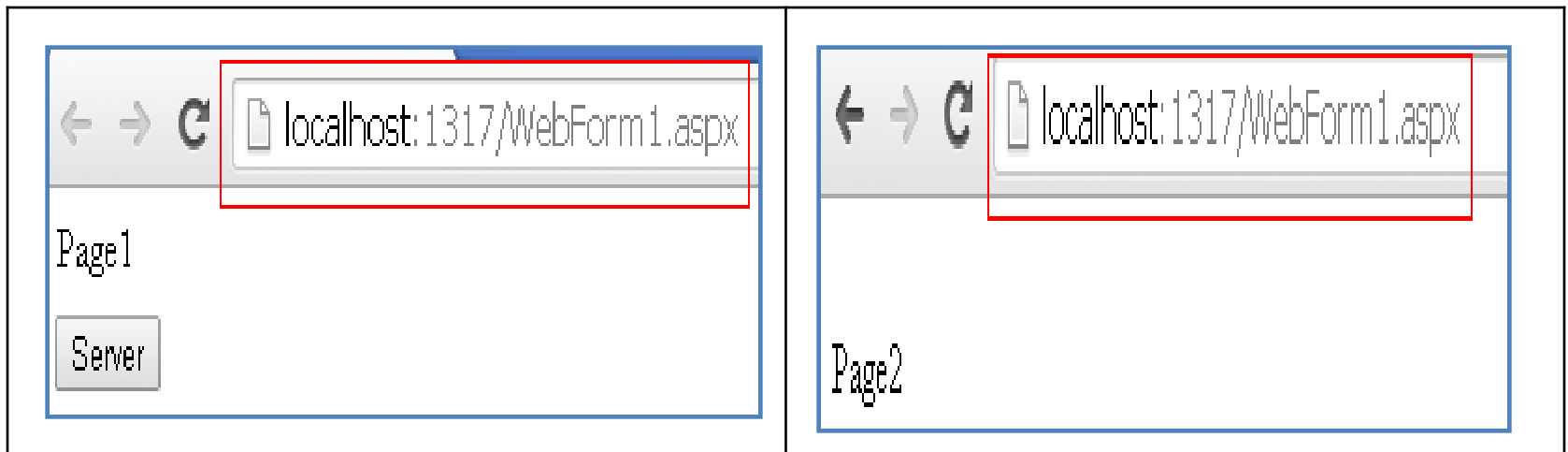
Syntaxe: **Response.Redirect ("URL")**



Server.Transfer

- L'objet Server est utilisé pour accéder aux propriétés et aux méthodes du serveur.
- Server.Transfer envoie toutes les informations qui ont été rassemblées par le traitement d'une page ASP, vers une autre page sans effectué un retour vers le client.

Server.Transfer ("URL")



Response.Redirect VS Server.Transfer

Response.Redirect	Server.Transfer
-URL se change alors Le client connaît la redirection.	-maintient l'URL d'origine dans le navigateur.
-Ne préserve pas la chaîne de requête et les variables de formulaire de la demande initiale	Préserve la chaîne de requête et les variables de formulaire de la demande initiale
-Allers-retours supplémentaires pour le serveur à chaque demande	-Evite les allers-retours inutiles vers le serveur

L'objet Request

- C'est un objet qui manipule les informations envoyées de client vers le serveur:
- Les informations qui passent dans un formulaire.

Request.Form["IdControle"]

- Les informations qui passent dans un lien.

**Request.QueryString [" Nom de variable envoyé dans
le lien "]**

L'objet Request

Exemple:

```
Server.Transfer("WebForm2.aspx?cle1=v1" +  
    "&cle2="+TextBox1.Text);
```

- *Dans WebForm2.Aspx*

```
Label1.Text = Request.QueryString["cle1"];
```

```
Label2.Text = Request.QueryString["cle2"];
```

Gestion d'Etat

- **Problème:**

- http est un protocole sans état.
- Chaque appel d' url ignore les précédents.

- **Besoin:**

- Lors de l' affichage d' une page, il est possible de personnaliser cet affichage en fonction de l' utilisateur.
- On peut conserver des mots passe(informations) pour éviter des identifications.

Gestion d'Etat

- **Solution:** Moyen pour conserver les informations.
-Deux cas de figures :

Gestion des états coté client	Gestion des états coté Serveur
<ul style="list-style-type: none">-View State- HiddenField-Query Strings- Cookies	<ul style="list-style-type: none">- Session- Application- Base de donnée

Viewstate

- **Exemple:**

Ecrivez un programme permettant de conserver le contenu des TextBox dans un viewstate

Nom:	<input type="text" value="meryem"/>
Prénom:	<input type="text" value="ouarrachi"/>
	<input type="button" value="viewstate"/>
Bonjour meryem ouarrachi	

Viewstate

- **Exemple:**

```
protected void Button3_Click(object sender, EventArgs e)
{
    ViewState["nom"] = TextBox1.Text;
    ViewState["prenom"] = TextBox2.Text;
    Label1.Text = "Bonjour" + " " + ViewState["nom"] + " " + ViewState["prenom"];
}
```


HiddenField

- Il est parfois utile de créer des champs cachés (Hidden Field) dans lesquels seront stockés des valeurs que l'on pourra utiliser après unPostBack .
- A la différence des ViewStates, n'ont ni compression, ni chiffrement ou hachage, un utilisateur avancé pourra consulter voir modifier les informations contenus dans nos HiddenField

HiddenField

- Exemple:

```
protected void Button4_Click(object sender, EventArgs e)
{
    HiddenField1.Value = TextBox1.Text;
    HiddenField2.Value = TextBox2.Text;
    Label2.Text = "Bonjour" + " " + HiddenField1.Value + " " + HiddenField2.Value;
}
```

- Résultat:

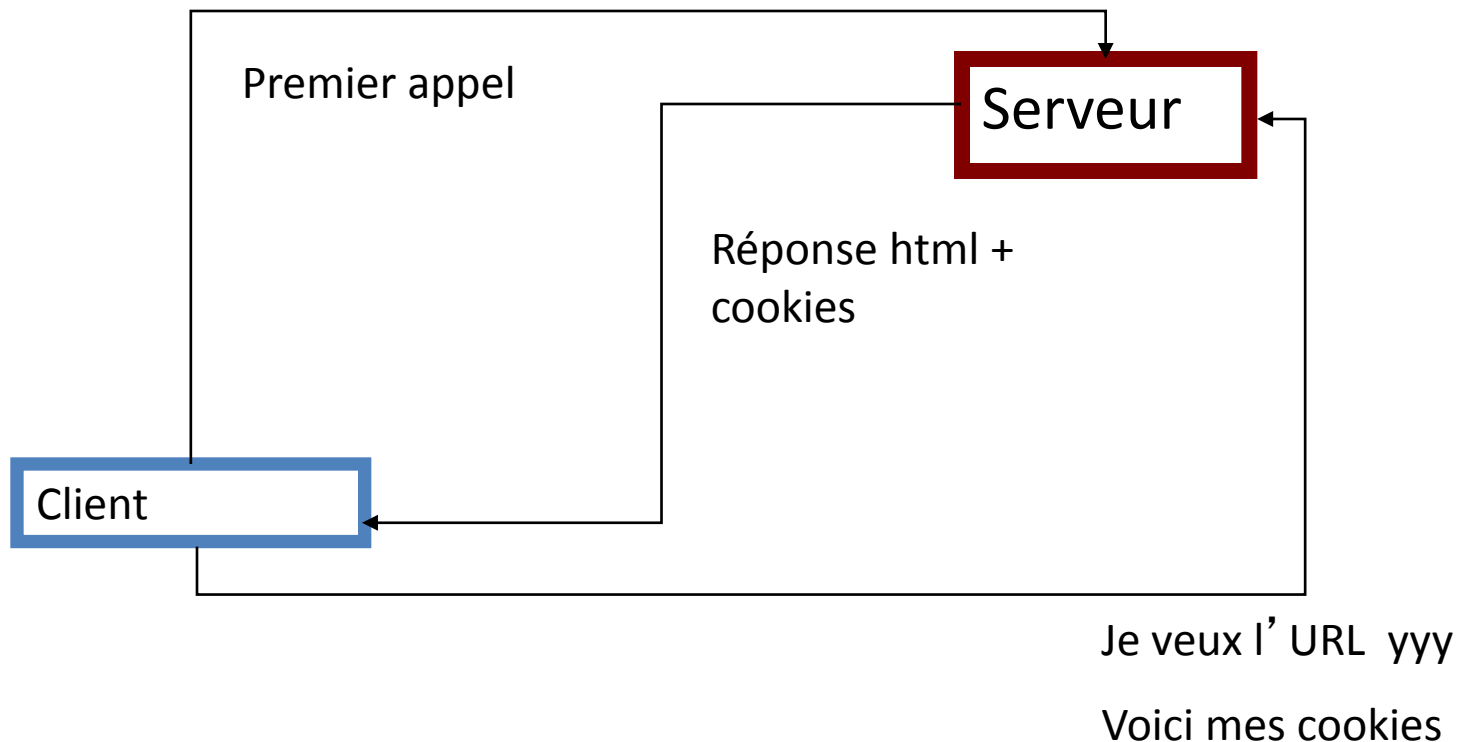


The screenshot shows a web browser window with the address bar displaying 'localhost:2256/page1.aspx'. The page contains a form with two text input fields. The first field, labeled 'Nom:', contains the text 'meryem'. The second field, labeled 'Prénom:', contains the text 'ouarrachi'. Below these fields is a button labeled 'HiddenField'. At the bottom of the page, the text 'Bonjour meryem ouarrachi' is displayed, which is the result of the code execution.

Les cookies

-Sont des données stockées dans un fichier texte dans le système de fichier client .

-Fonctionnement:



Stocker un cookie sur le client

- Automatique, géré par le navigateur.
- Le navigateur renvoie automatiquement les cookies concernés par l' URL demandée.
- Les cookies périmées sont automatiquement supprimées.

Les cookies

- Création:

Response.Cookies["nomCookies"].Value = valeur;

- Récupérer un cookie

Request.Cookies["nomCookies"].Value

- Vérifier l'existence d'un cookie:

if (Request.Cookies["nomCookies"] == null)

Les cookies

- Fixer la date d'expiration

```
DateTime dateExpir = DateTime.Now.AddMonths(3);  
Response.Cookies[" nomCookies "].Expires = dateExpir;
```

- Supprimer un cookie

```
-Response.Cookies[" nomCookies "].Expires = DateTime.Now;
```

Les cookies

- Visualiser cookies



Les sessions

- Permettent de stocker temporairement (pendant toute la durée de vie de la session) des informations transmissibles de page en page.
- Chaque connexion possède un espace sur le serveur pour stocker des variables de session.

Fonctionnement de session

- La première fois que vous ajoutez une variable de session, le serveur vous ouvre une session et génère un cookie de session comportant un identifiant.
- Dans les rappels ultérieurs votre navigateur renvoie le cookie de session, ce qui permet au navigateur de retrouver votre espace de session.
- A la fermeture du navigateur, le cookie de session est détruit.

Session: mise en oeuvre

- Ajouter une variable de session:

Session["sessNom"] = valeur

- Vérifier l'existence d'une variable de session:

if (Session["sessNom"] != null)

- Relire une variable de session:

textbox1.Text = Session["sessNom"];

Session: mise en oeuvre

- Supprimer une session:

Session["sessNom"] = null

ou

Session.clear();

Les évènements de session

La session possède deux évènements.

- Session_Start:** exécuté quand une nouvelle session démarre

- Session_End:** exécuté quand une session est fermé.

Fichier Global.asax

```
void Session_Start(object sender, EventArgs e)
{
    // Code that runs when a new session is started
}

void Session_End(object sender, EventArgs e)
{
    // Code that runs when a session ends.
}
```

Application

Définition:

- il s'agit de l'objet représentant l'application web elle-même.
- Cet objet permet le partage d'information entre plusieurs utilisateurs d'une application.

Exemple:

- nombre d'utilisateurs connectés simultanément.

Application

Fonctionnement:

- L'objet Application se situe dans la mémoire du serveur Web.
- crée lorsque l'application reçoit sa première requête, et reste actif jusqu'à ce que la dernière Session soit libérée.

Application

Comment initialiser et lire ces variables?

- Affecter une variable d'application:

Application ["clé"] = valeur

- Récupérer une variable d'application:

Variable = Application ["clé"]

Événements d'Application

- **Événement `Application_OnStart`:**

- Rôle:**

Cet événement est déclenché lorsque l'application démarre, c'est-à-dire à la mise en service du site web.

- Exemple d'utilisation:**

On utilisera cet événement pour initialiser des variables d'application comme un compteur de visiteurs, par exemple.

Événements d'Application

- **Événement `Application_OnEnd`:**

- Rôle:**

Cet événement est déclenché lorsque l'application s'arrête c.à.d une fois que toutes les sessions en cours ont été arrêtées.

- Exemple d'utilisation:**

On utilisera par exemple cet événement pour enregistrer quelque part le nombre de visiteurs, afin de reprendre ce nombre au prochain démarrage.

Événements d'Application

Fichier global.asax:

```
void Application_Start(object sender, EventArgs e)
{
    // Code that runs on application startup
    Application["compteur"] = 0;
}

void Application_End(object sender, EventArgs e)
{
    // Code that runs on application shutdown
}
```

Utiliser un SGBD

- Utiliser les instructions de ADO.NET.

ou

- Travailler en mode assistant:

1. SqlConnection : composant ajouté à partir de Toolbox, il représente un objet utilisé pour accéder aux tables d'une base de données.


Utiliser un SGBD

2. Configurer le sqlDataSource

asp:sqldatasource#SqlDataSource1

SqlDataSource - SqlDataSource1

Configurer la source de données - SqlDataSource1

 **Configurer l'instruction Select**

Comment voulez-vous extraire les données de votre base de données ?

☐ Spécifiez une instruction SQL personnalisée ou une procédure stockée

☒ Spécifiez les colonnes d'une table ou d'une vue

Nom :

Colonnes :

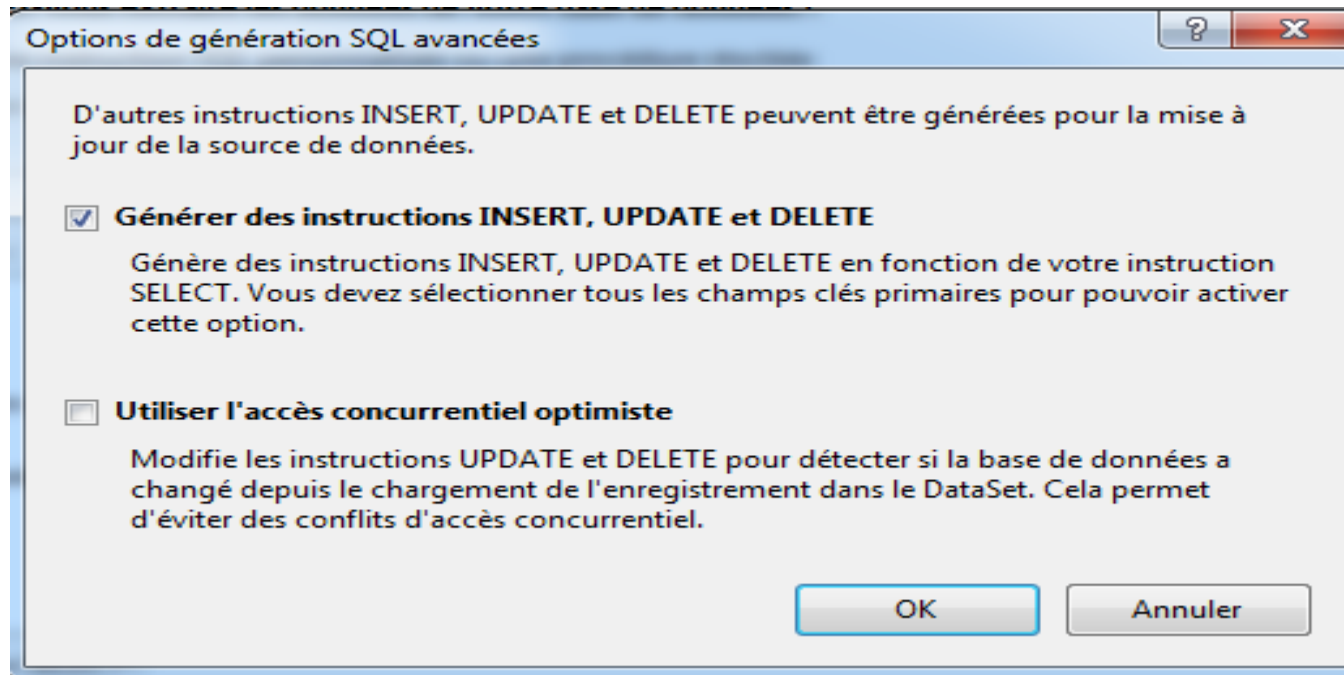
<input checked="" type="checkbox"/> *
<input type="checkbox"/> cne
<input type="checkbox"/> nom
<input type="checkbox"/> prenom
<input type="checkbox"/> sexe
<input type="checkbox"/> date_naiss
<input type="checkbox"/> id_fil

☐ Retourner seulement des lignes uniques

Utiliser un SGBD

3. Pour ajouter les instructions de mise à jour (Insert, Update, Delete).

Cliquer sur Options avancées



Utiliser un SGBD (mode assistant)

○GridView:

- Permet d'afficher les données dans un tableau
- dataSource pour la lier à un contrôle SqlDataSource.

○DetailView:

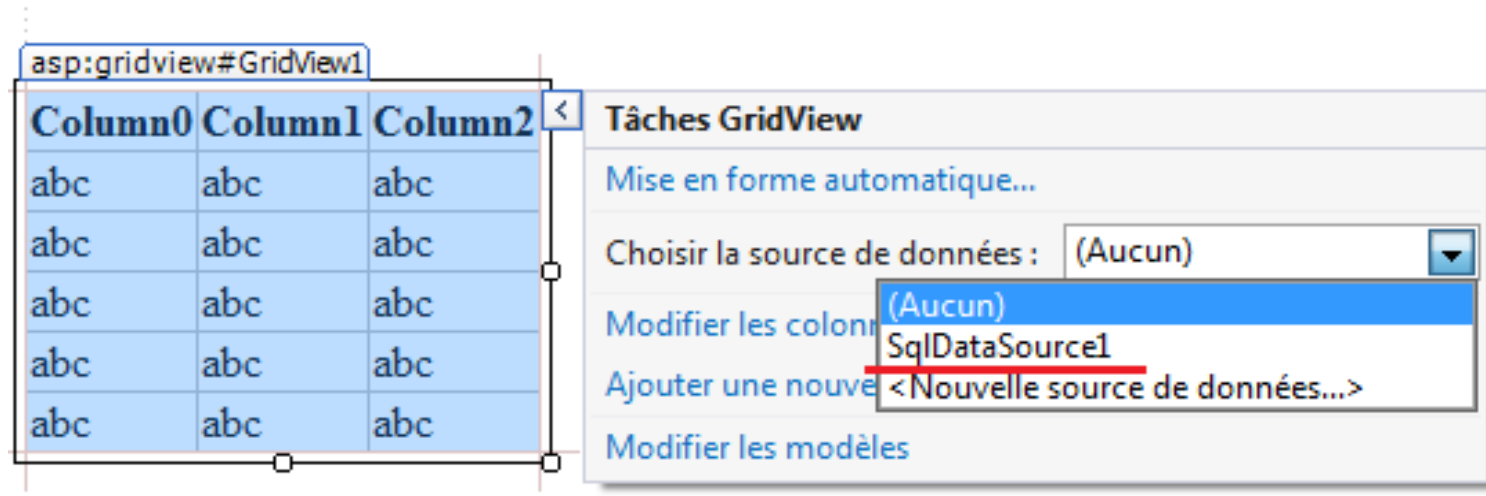
- affiche les données enregistrement par enregistrement.

○DataList:

- affiche les informations sous forme d'une liste.

Utiliser un SGBD (mode assistant)

- ❑ Lier les composants précédents au SQLDataSource:



Les validateurs

-Sont des contrôles dédiés à la validation càd permettant la vérification des champs d'un formulaire

•Les propriétés des validateurs

- **ControlToValidate** : Permet d'associer à un validateur le contrôle qu'il doit valider.

-**Display** : Permet de définir la façon dont vont s'afficher les messages d'erreurs des validators(static;dynamic;None).

-**Text** : Permet d'indiquer le message d'erreur qui sera affiché

Les validateurs

- Les propriétés des validateurs

-ErrorMessage : Permet d'indiquer le message d'erreur qui sera affiché. Cette propriété est utilisée notamment pour le contrôle ValidationSummary.

-ValidationGroup : Permet de grouper les contrôles pour que ne soient validés que les contrôles qui font parti du même groupe. Cela permet de définir plusieurs zones de validation dans une page.

-EnableClientScript : Permet d'indiquer si l'on souhaite utiliser une validation cliente ou non. (Vrai par défaut)

Exécution de validation

- Il peut s'exécuter:
 - Sur le client: le java script
 - Sur le serveur: Causesvalidation=True
 - Sur le client, puis le serveur (conseillé).

```
protected void Button1_Click(object sender, EventArgs e)
{
    Page.Validate();
    if (Page.IsValid)
    {
        Response.Redirect("accueil.aspx");
    }
}
```

Les principaux validateurs

- **RequiredFieldValidator**: permet de tester si un champ est rempli ou non.
- **RangeValidator**: vérifie si la valeur d'un contrôle d'entrée se trouve dans une plage de valeurs spécifiée.
 - On peut comparer des Integer, Double, Date ... Pour préciser quel type on attend, on va utiliser la propriété **Type**. Les bornes à valider seront saisies grâce aux propriétés **MinimumValue** et **MaximumValue**.

Les principaux validateurs

- **CompareValidator:** permet de comparer la valeur entrée par l'utilisateur avec
 - une valeur : ValueToCompare et Operator(Equal,NotEqual...)
 - la valeur d'un autre contrôle: ControleToCompare
- On peut également se servir de ce contrôle pour vérifier qu'une donnée saisie est d'un type particulier: on utilisera Operator=DataTypeCheck et on change le Type.

Les principaux validateurs

- **RegularExpressionValidator:** permet de vérifier une entrée à partir d'une expression régulière. Par exemple, pour vérifier un email.

Les principaux validateurs

- **CustomValidator:** Effectue une validation définie par l'utilisateur sur un contrôle d'entrée dans l'événement suivant:

ServerValidate(object source, serverValidateEventArgs args)

-args:représente les données associées à cet evt;il possède deux propriétés:

- ✓ **value:** renvoie la valeur de la commande d'entrée(ex:valeur saisi dans un TextBox)
- ✓ **IsValid:** tester si la validation est réussi ou non.

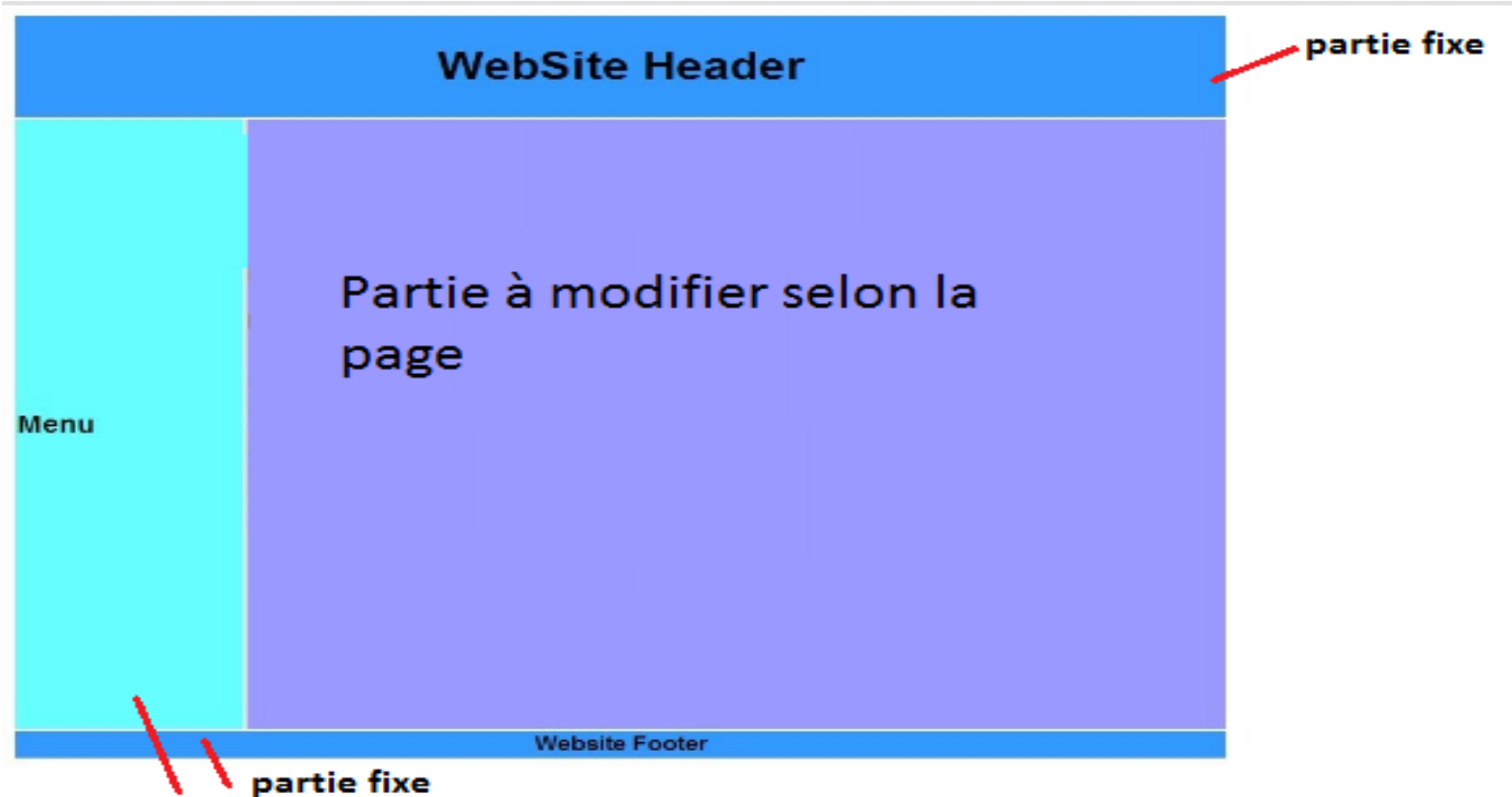
Les validateurs

- **ValidationSummary**: ce n'est pas un validator comme les autres, il n'effectue pas de validation mais propose de récapituler les différentes erreurs survenues lors de la validation de la page.

-Il se base uniquement sur le contenu des propriétés **ErrorMessage** de chaque validator qui n'a pas passé la validation.

Les pages maîtres.

Dans un site web, on aura souvent une partie qui se répète dans toutes les pages comme l'entête/pied de page, menu



Les pages maîtres.

- On peut réaliser cela via les pages maîtres.
- Elles permettent de définir une mise en page et les éléments à inclure dans toutes les pages.
- Il est logique de commencer le projet par la construction d'une page maître.

Construire une page maître

- Ajouter un nouvel élément de type page maître.
- Définir les éléments de mise en page, table ou div.
- Inclure les éléments constant: décorations et contrôles utilisateur.
- « **Place Holder** » pour inclure les éléments variables.
- Après on crée un nouvel élément webform use Master page (ou contentPage)et on coche notre master page et juste dans la partie de placeHolder où on peut inclure le contenu de cette page.

Contrôle utilisateur

-En plus d'utiliser des contrôles serveur Web dans les pages Web ASP.NET, on peut créer nos propres contrôles personnalisés et réutilisables, à l'aide des mêmes techniques que celles qui sont utilisées pour créer les pages Web ASP.NET. Ces contrôles sont appelés contrôles utilisateur.

-**construire user controle:** Ajouter nouvel élément→web user control→et après on peut le glisser dans d'autres pages.Toute modification dans le user control sera transmis aux autres pages qui l'utilisent(centralisation d'administration)

Web.siteMap

- SiteMap est une représentation de la structure de navigation d'un site, fournie par un ou plusieurs fournisseurs de plan de site(**SiteMapDataSource**)
- La manière la plus simple de créer un plan de site consiste à créer un fichier XML nommé Web.sitemap qui organise les pages du site de manière hiérarchique.
- Création:** ajouter nouvel élément de type Web siteMap

Web.siteMap

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="~/Default.aspx" title="Accueil">
    <siteMapNode url="~/Page1.aspx" title="Page 1">
      <siteMapNode url="~/Page11.aspx" title="Page 1.1" />
      <siteMapNode url="~/Page12.aspx" title="Page 1.2" />
    </siteMapNode>
    <siteMapNode url="~/Page2.aspx" title="Page 2" />
    <siteMapNode url="~/Page3.aspx" title="Page 3" />
  </siteMapNode>
</siteMap>
```



```
- Accueil
  - Page 1
    - Page 1.1
    - Page 1.2
  - Page 2
  - Page 3
```


Web.siteMap

- Les contrôles disponibles sont :

-TreeView qui affiche l'arborescence du sitemap.

-Menu qui permet d'afficher la navigation sous la forme d'un menu.

-SiteMapPath qui permet d'indiquer à l'utilisateur où il se trouve dans la hiérarchie du site.

-Création: on intègre ces composants à partir de toolbox et on affecte à leur propriété « DataSource »: le sitemap généré précédemment

Application multilingue

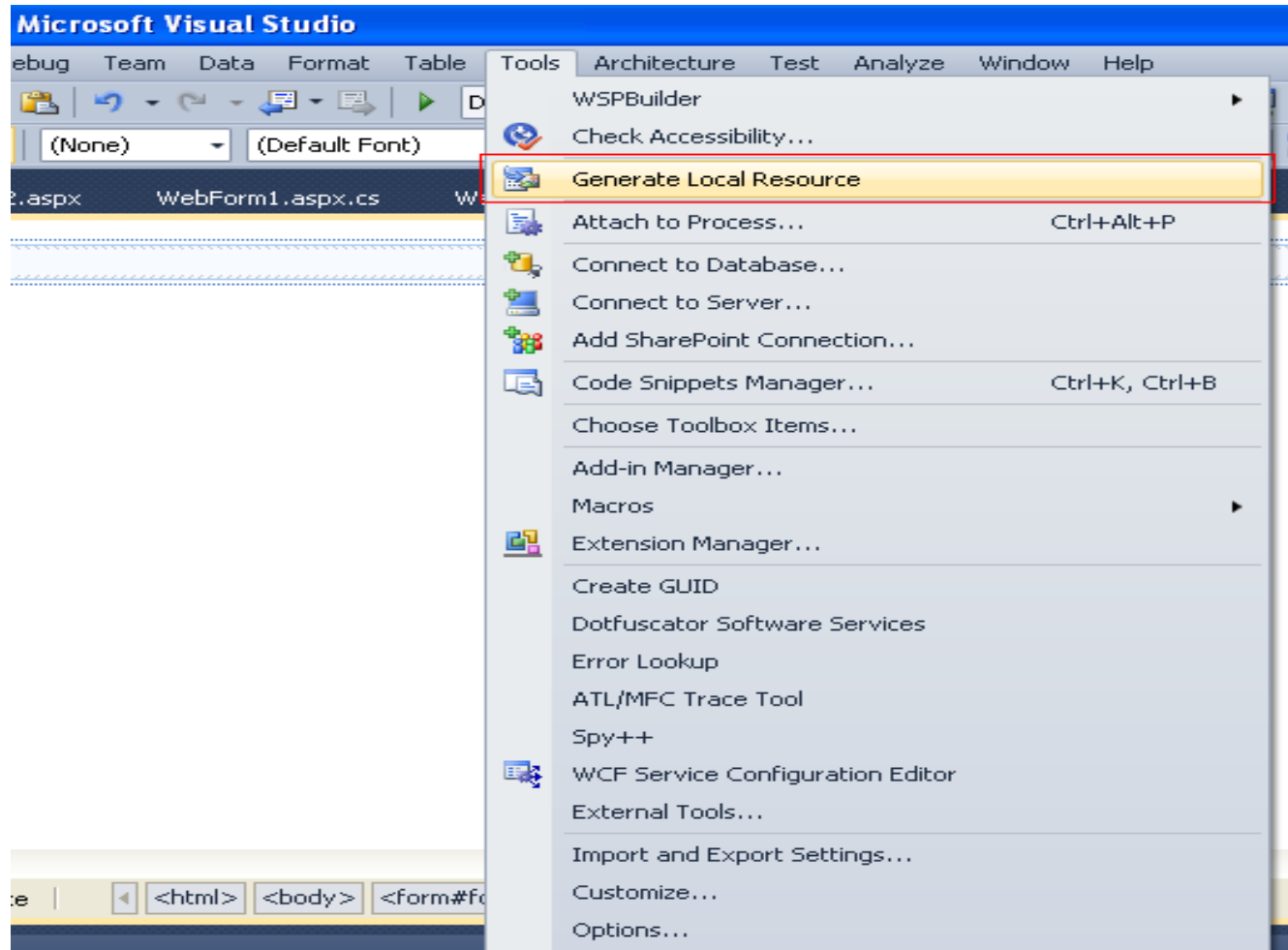
Pour avoir un site web multilingue:

➤ Etape1: La localisation

Le principe de la localisation est de générer un fichier qui va contenir une référence de chacun de nos contrôles de notre page web ainsi qu'une propriété qui contiendra du texte.

Application multilingue

- Générer un fichier de ressources locales



Application multilingue

- Générer un fichier de ressources locales

	Name ▲	Value	Comment
	AccueilResource2.ToolTip		
	BtnValidNomResource1.Text	Valider	
	BtnValidNomResource1.ToolTip	Validez s'il vous plait	
	GridView1Resource1.Caption	Titre de mon Tableau	
	GridView1Resource1.EmptyDataText		
	GridView1Resource1.ToolTip	Mon tableau	
	LblEntrerNomResource1.Text	Entrez votre nom :	
	LblEntrerNomResource1.ToolTip		
	LblTexteAccueilResource1.Text	Bienvenue	
▶	LblTexteAccueilResource1.ToolTip		
	PageResource1.Title	Page Test	
	TxtNameResource1.Text		
	TxtNameResource1.ToolTip		
*			

Remarque: ce fichier s'enregistre dans le dossier « App_LocalResource ».

Application multilingue

- Générer un fichier de ressources locales

```
culture="auto" meta:resourcekey="PageResource1" uiculture="auto"
```

Application multilingue

➤ **Etape2:** Pour ajouter une langue à votre application web, copiez votre fichier de ressource locale dans votre répertoire App_LocalResources puis renommez-le de la façon suivante :
(nom de la page).aspx.(nom de la langue selon la norme ISO 639-1).resx

Exemple: Default.aspx.fr.resx ; Default.aspx.en.resx;
Default.aspx.ar.resx

Application multilingue

➤ Etape3: Mettre en place plusieurs langue:

- La méthode InitializeCulture():Définit la Culture et UICulture pour le thread actuel de la page.
- La valeur Culture détermine les résultats de fonctions spécifiques à une culture, comme la mise en forme de la date, des nombres et de la monnaie, etc.
- La valeur UICulture détermine les ressources qui sont chargées pour la page.

Application multilingue

```
protected override void InitializeCulture()
{
    if (Request.Form["DropDownList1"] == "en-US")
    {
        Thread.CurrentThread.CurrentCulture = new
        CultureInfo("en");
        Thread.CurrentThread.CurrentUICulture =
        new CultureInfo("en");
    }

    else
    {
        Thread.CurrentThread.CurrentCulture = new
        CultureInfo("fr");
        Thread.CurrentThread.CurrentUICulture =
        new CultureInfo("fr");
    }
}
```


AJAX

AJAX

- En aspx à chaque appel, la page est entièrement reconstruite.
- Ajax permet de ne reconstruire qu'une partie de la page.

Exemple:

- le contenu d'un champ de formulaire peut être changé, sans avoir à recharger la page avec le titre, les images, le menu, etc.
- Recherche en google.

Principe d'AJAX

Asynchronous Javascript And XML(Javascript et Xml asynchrones):

- Asynchrone**:faire des appels asynchrones au serveur depuis le client.

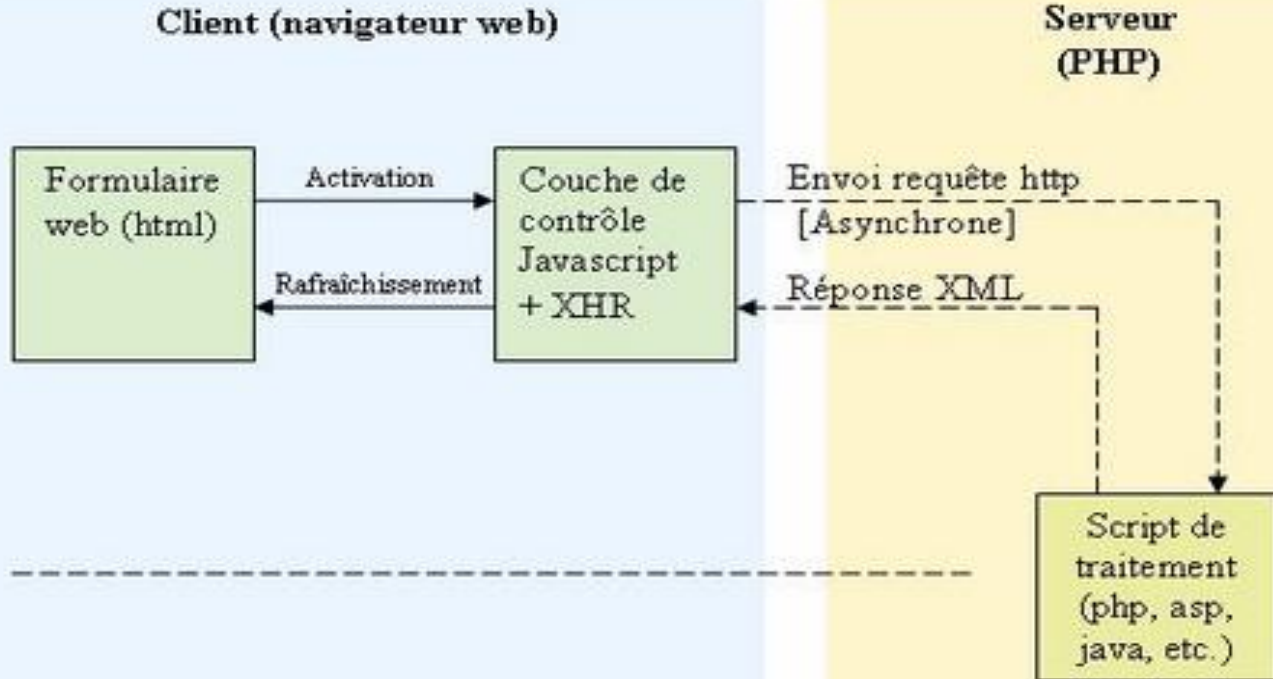
- Javascript** et **XML**:

- Lors de ces appels, le serveur retournera du XML qui sera "récupéré" par JavaScript et traité.

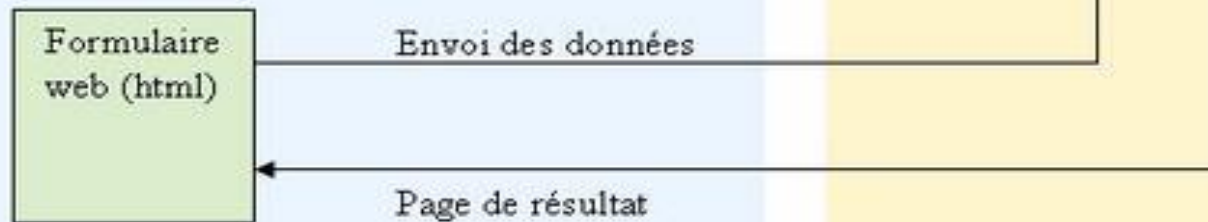
- L'objet **XMLHttpRequest** de javascript sert au dialogue asynchrone avec le serveur Web

Principe d'AJAX

**Avec
AJAX**

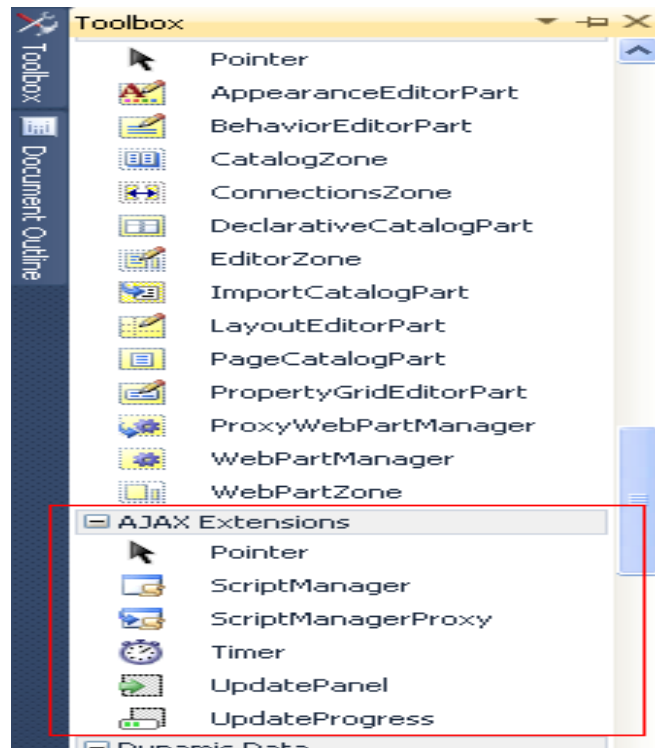


**Sans
AJAX**



ASP.net Ajax

Microsoft ASP .NET AJAX Extensions, présentés sous la forme de nouveaux contrôles serveur ASP .NET, et classes associées.



ASP.net Ajax

- Etape1:

- Insérer un script manager dans le code aspx:

- ```
<asp:ScriptManager ID="ScriptManager1" runat="server">
```

- ```
</asp:ScriptManager>
```

- Il permet:

- l'ajout des bibliothèques de scripts au navigateur.
 - L'appel des services web

ASP.net Ajax

Etape2:

-Encadrer les objets concernés par Ajax par des balises update Panel:

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
```

```
<ContentTemplate>
```

```
<asp:TextBox ID="txtA" runat="server"></asp:TextBox>
```

```
<asp:TextBox ID="txtResult" runat="server"></asp:TextBox>
```

```
</ContentTemplate>
```

```
</asp:UpdatePanel>
```

ASP.net Ajax

Etape2:

UpdatePanel: permet la restitution d'une section de page

Remarque:

-On peut pas utiliser une navigation de type
Server.Transfer dans un composant intégré en
Updatepanel

AJAX Control Toolkit

- C'est un ensemble de contrôles Web enrichis permettant d'apporter à cette dernière, de nouvelles fonctionnalités AJAX, au travers de contrôles ASP .NET.
- Il contient plus de 30 contrôles qui vous permettent de créer facilement des riches, des pages Web interactives.

AJAX Control Toolkit

-Pour l'intégrer il faut l'installer :

- Installation à partir d'un lien

<https://www.devexpress.com/Products/AJAX-Control-Toolkit/>

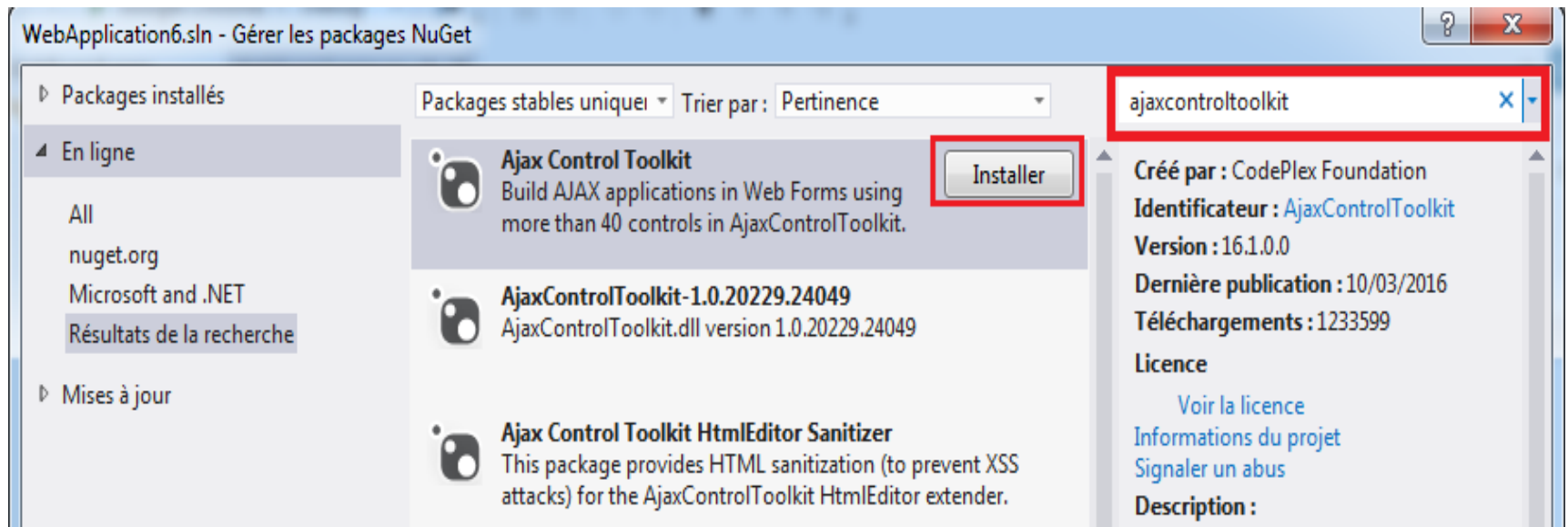
- Installation à partir NuGet Package Manager:

Une collection d'outils pour automatiser le processus de téléchargement, l'installation, la mise à niveau, la configuration et la suppression de paquets à partir d'un projet Visual Studio.

AJAX Control Toolkit

-Installation à partir NuGet Package Manager:

Outils-->Gestionnaire de package Nuget-->Gérer les packages
Nuget pour la solution



AJAX Control Toolkit

-Insérer les nouveaux éléments dans la boîte à outils de VS:

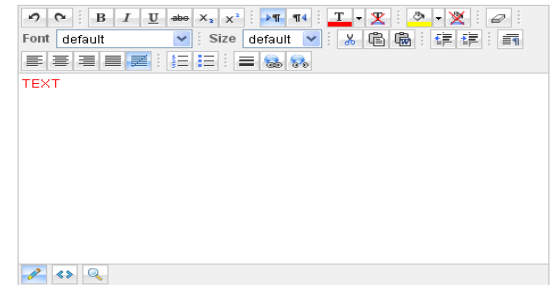
Click droit en toolbox→Ajouter un onglet→Insérer un nom

-Click droit sur le nouvel onglet→Choisir les
éléments→sélectionner le fichier AjaxControlToolkit.dll

AJAX Control Toolkit

Exemple:

-le contrôle **Editor**



-Le contrôle **AutoComplete** est un extender de TextBox qui fournira une auto-complétion de ce qui est écrit dans le TextBox. Il fonctionne grâce à un service web qui doit avoir pour signature :
public string[] RecupererListe(string Text){ ... }

Exemple

Exemple1

Exemple2

Exemple3

AJAX Control Toolkit

AutoComplete 1/2:

```
<%@ Register Assembly="AjaxControlToolkit" Namespace="AjaxControlToolkit" TagPrefix="ajaxToolkit" %>
```

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
</asp:ScriptManager>
<br />
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
  <ContentTemplate>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <ajaxToolkit:AutoCompleteExtender ID="AutoCompleteExtender1" runat="server"
      MinimumPrefixLength="2" CompletionInterval="100"
      EnableCaching="false" CompletionSetCount="10"
      FirstRowSelected="false" UseContextKey="true" ServiceMethod="GetCompletionList"
      TargetControlID="TextBox1">
    </ajaxToolkit:AutoCompleteExtender>
  </ContentTemplate>
</asp:UpdatePanel>
```

AJAX Control Toolkit

AutoComplete 2/2:

```
public partial class WebForm2 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
    [System.Web.Script.Services.ScriptMethod]
    [System.Web.Services.WebMethod]
    public static string[] GetCompletionList(string prefixText)
    {
        string[] s = { "Exemple1", "Exemple2", "Exemple3" };
        return s;
    }
}
```