Cours .Net
ASP.NET

Lemettre Arnaud
Arnaud.lemettre@gmail.com

EPITA
ÉCOLE D'INGÉNIEURS EN INFORMATIQUE

Microsoft® .NET

Sommaire

- Introduction
- Concept
- Composition d'une page
- User control
- Event
- DataBinding
- Session
- Ajax
- Déploiement
- Site MapMenu

Introduction

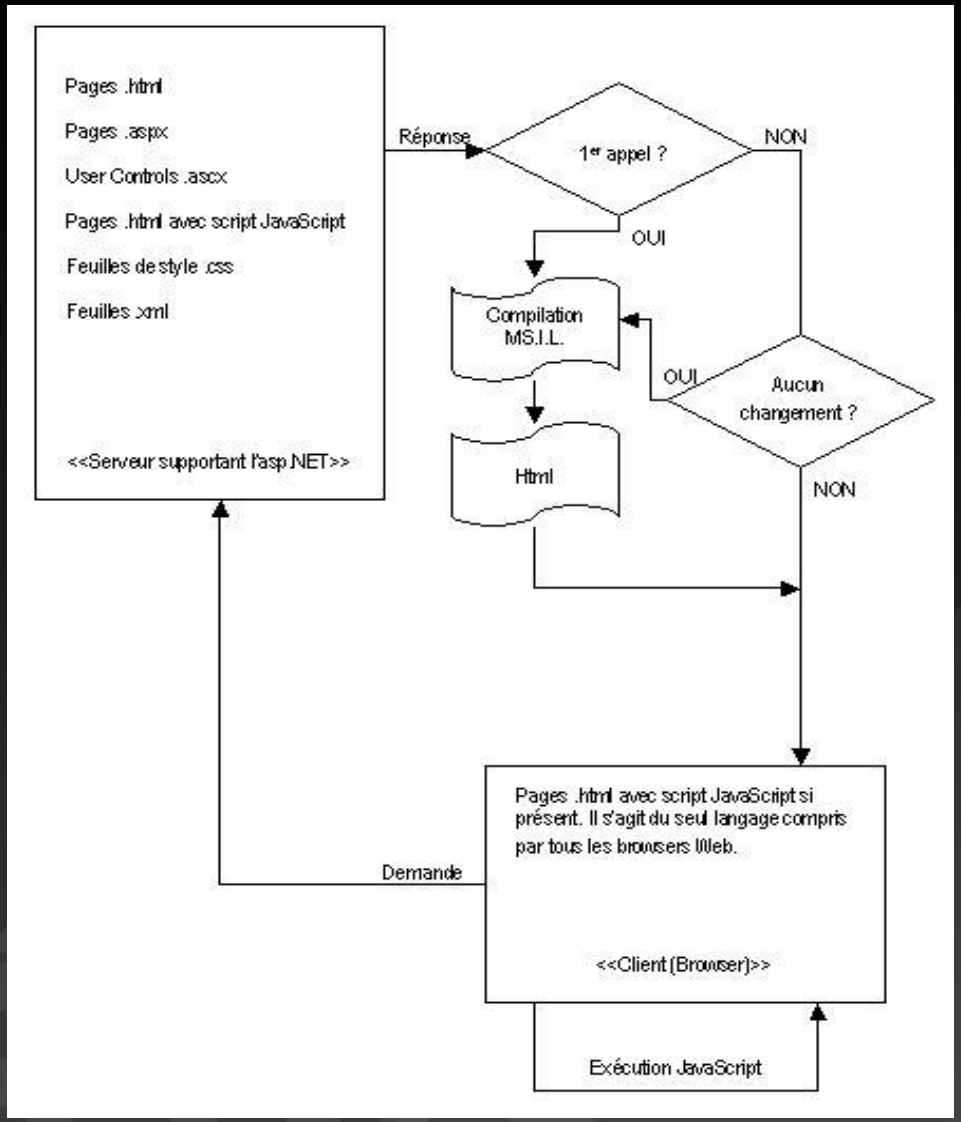
- ASP.NET est un ensemble de technologies de programmation web créées par Microsoft. Les programmeurs peuvent utiliser ASP.NET pour créer des sites web dynamiques, des applications web ou des web services XML. La technologie est accessible grâce à l'installation d'un serveur web compatible ASP (IIS).

Concept

Concept

- Interaction des différentes couches

Concept



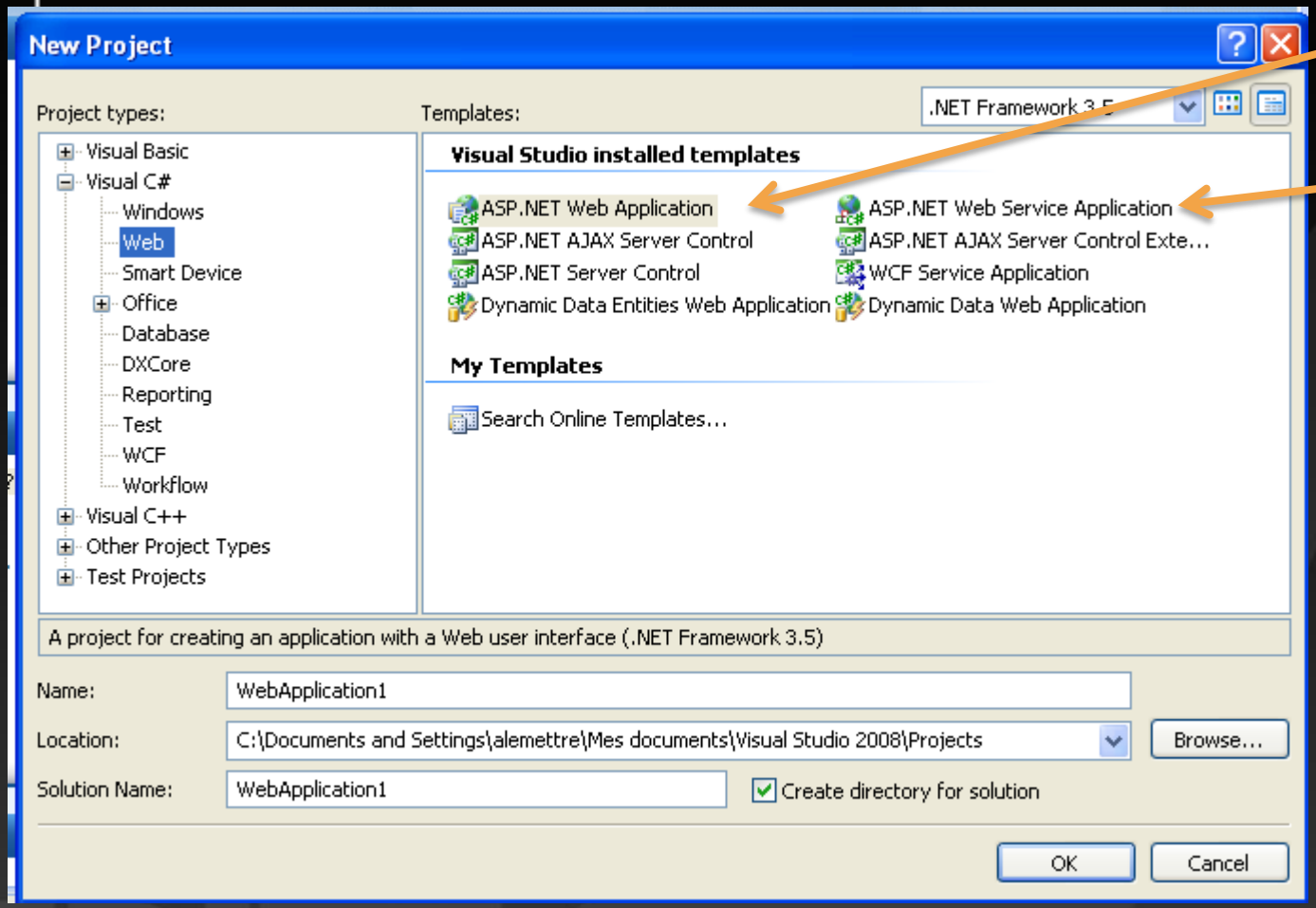
- Processus d'affichage d'une page

PRÉSENTATION

Présentation

- Il y a deux façons pour réaliser un projet Web :
  - Site web asp.net
  - Application web asp.net
- L'un ou l'autre templates sont équivalents sauf à quelques détails prés.

PRÉSENTATION



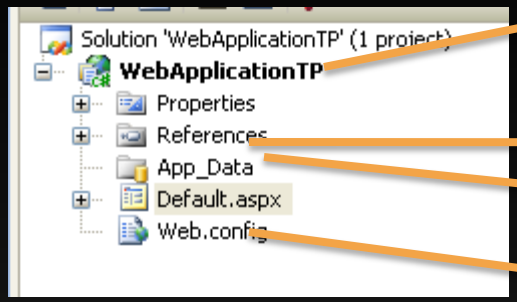
Site web

Permet d'héberger des web services de type asmx

Note : les * Server Control permettent de redéfinir des contrôles personnalisés pour les inclure dans les pages

Les autres projets seront vus dans le cours de l'année.

PRÉSENTATION



Racine du site

Références pour les assemblies

Dossier asp.net

Fichier de configuration

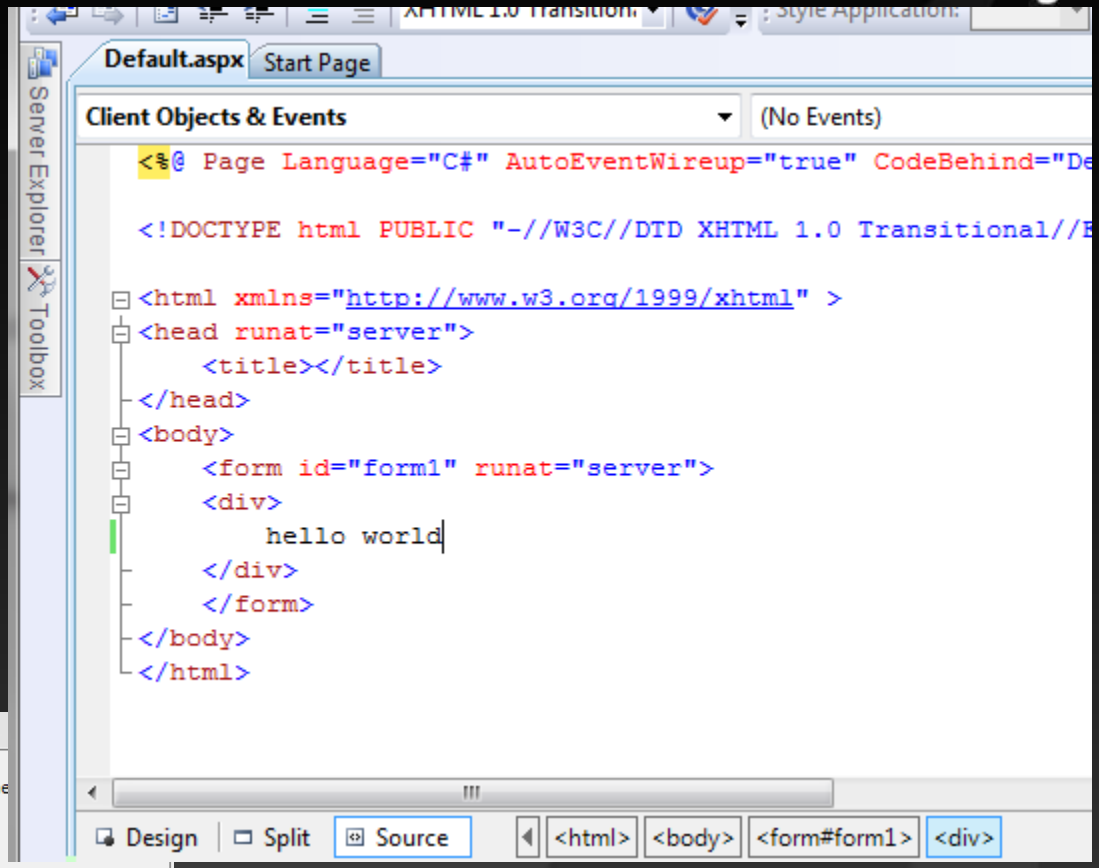
Il existe plusieurs types de dossiers pour les environnements ASP.NET

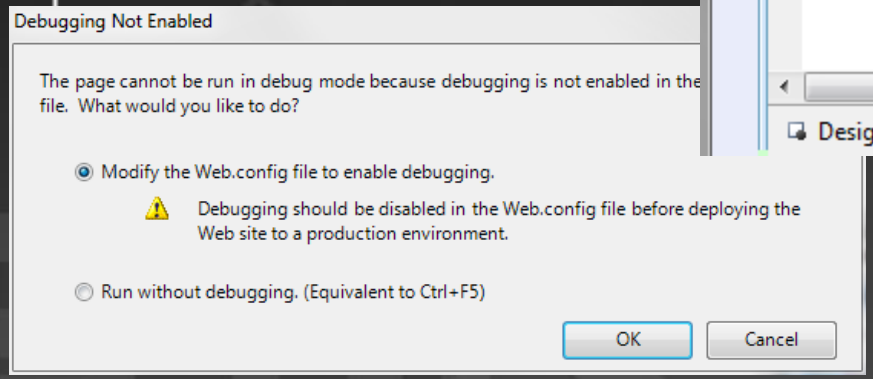
Nom	Description
App_code	Contient le code source (*.cs , ….)
App_data	Contient les fichiers données (*.mdf, *.xml, …)
App_GlobalResources	Contient les fichiers de ressources (langue , ….)
App_Themes	Contient les différents thèmes de l'application (*.css, *.jpg, …)
App_WebReferences	Contient les fichiers pour les web services
bin	Contient les dll de l'application après compilation.

Présentation

- Faire son premier hello world !

Appuyer sur F5 pour compiler.





Cliquer sur ok pour afficher la page.

PRÉSENTATION

Le détail de la page :

Langage utilisé pour développer

Indique que c'est un contrôle serveur et que l'on pourra interagir avec.

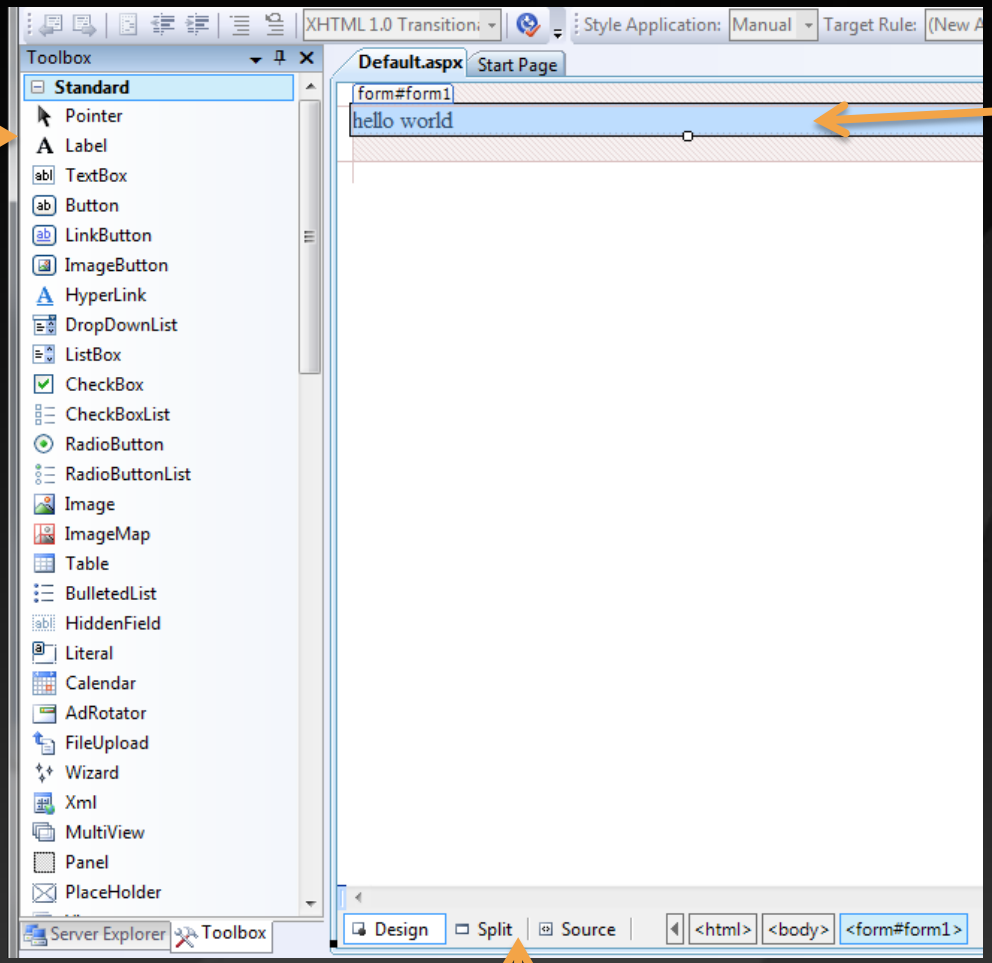
Code behind associé, à la page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="WebApplicationTP._Default" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        hello world
    </div>
    </form>
</body>
</html>
```

Composition d'une page

Composition d'une page

Composants à drag & drop sur l'interface



La page

Permet de passer de l'interface graphique au code aspx

Composition d'une page

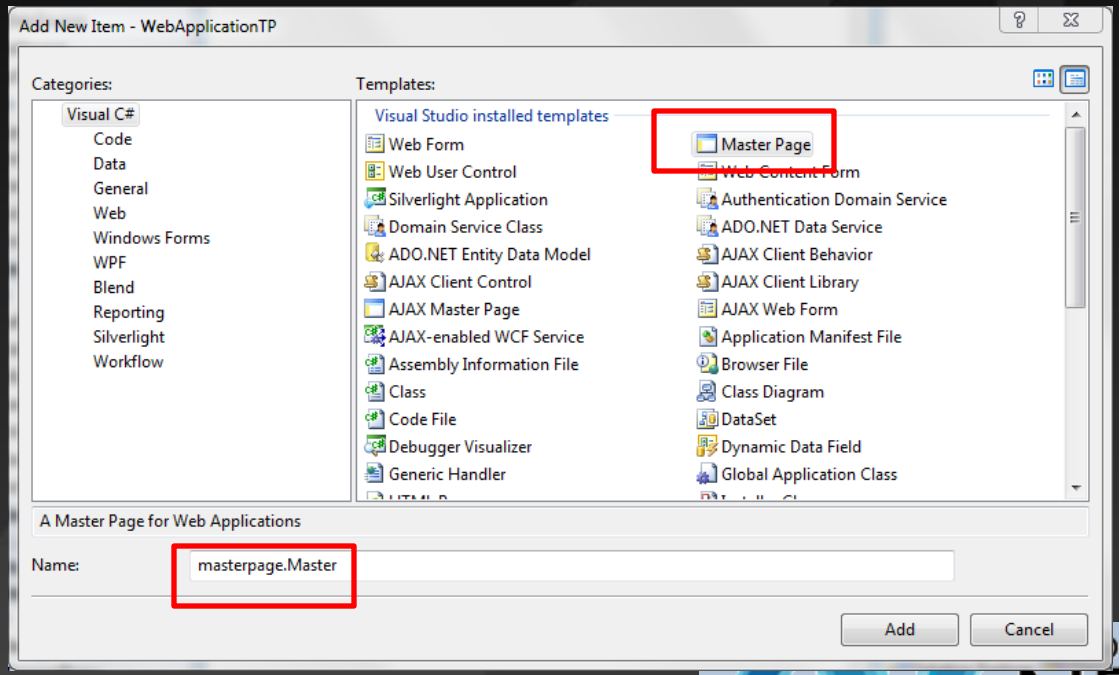
- Tout comme dans d'autres technologies, il faut éviter au maximum de répéter du code, c'est également le cas pour le design.

- Pour cela asp.net introduit le principe de masterpage.

Composition d'une page

- La master page est une page qui permettra de centraliser toutes les fonctionnalités communes. Dans un site web il peut exister plusieurs masterpages.

Ajout -> new item

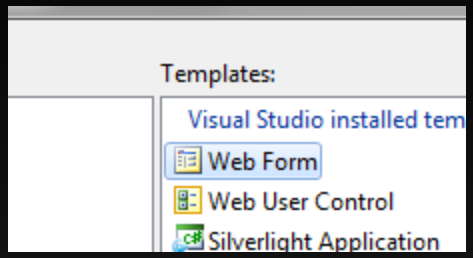


Composition d'une page

```
<%@ Master Language="C#" AutoEventWireup="true" CodeBehind="masterpage.master.cs"
    Inherits="WebApplicationTP.masterpage" %>

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

<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
<body>
<%--Elements qui apparaitront sur toutes les pages--%>
    <form id="form1" runat="server">
    <div>
    Menu
        <asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
        <%--A completer dans les autres pages--%>
        </asp:ContentPlaceHolder>
    </div>
    </form>
</body>
</html>
```

Composition d'une page

Puis après il faut rajouter une webform pour créer une nouvelle page




```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs" Inherits="WebApplicationTP.WebForm1"
    MasterPageFile="~/masterpage.Master" %>

<asp:Content ID="content1" ContentPlaceHolderID="head" runat="server">

</asp:Content>

<asp:Content ID="content2" ContentPlaceHolderID="ContentPlaceHolder1"
    runat="server">
<%--composants de la page--%>
<asp:Label Text="plop" runat="server" />
</asp:Content>
```

Composition d'une page

Au niveau de la masterpage, on peut par exemple rajouter notre feuille de style CSS. Par exemple :

```
body
{
}

p
{
    color:Green;
    font-size:12px;
}
```

Dans la master page :

```
<head runat="server">
    <title></title>
    <link rel="Stylesheet"
  type="text/css"
  href="Stylesheet.css" />
    <asp:ContentPlaceHolder
  ID="head" runat="server">
    </asp:ContentPlaceHolder>
</head>
```

Composition d'une page

Puis dans la page qui utilise la masterpage :

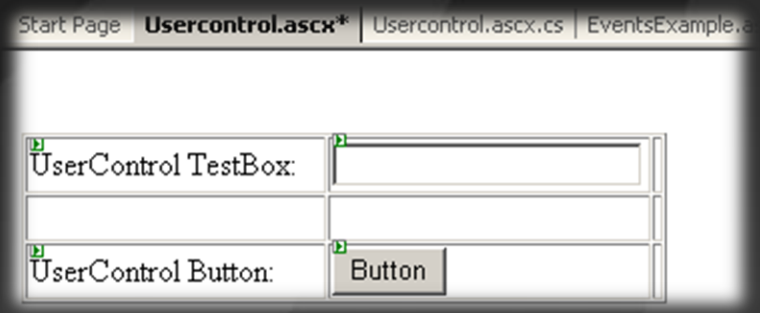
```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs" Inherits="WebApplicationTP.WebForm1"
    MasterPageFile="~/masterpage.Master" %>

<asp:Content ID="content1" ContentPlaceHolderID="head" runat="server">

</asp:Content>

<asp:Content ID="content2" ContentPlaceHolderID="ContentPlaceHolder1"
    runat="server">
     <%--composants de la page--%>
<p>texte vert</p>
<asp:Label Text="test" runat="server" />
</asp:Content>
```

Composition d'une page

- La construction d'une page se fait après, par simple drag & drop de composants.

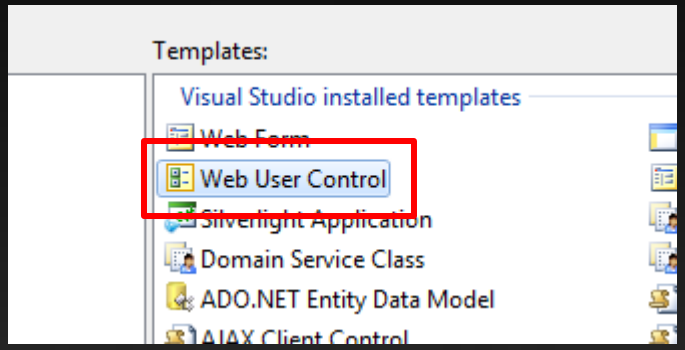
- Cependant afin d'éviter la répétition de code et de contrôles, on peut construire des users controls. Ces composants en regroupent plusieurs qui peuvent être ré utilisés dans différentes pages.

LES USERS CONTROLS



Les users controls

- C'est un composant comme un autre qu'il faut ajouter au projet :



Puis mettre les différents composants par exemple : boutons, combo box, …

Les users controls

Pour pouvoir l'utiliser ensuite dans une autre forme :

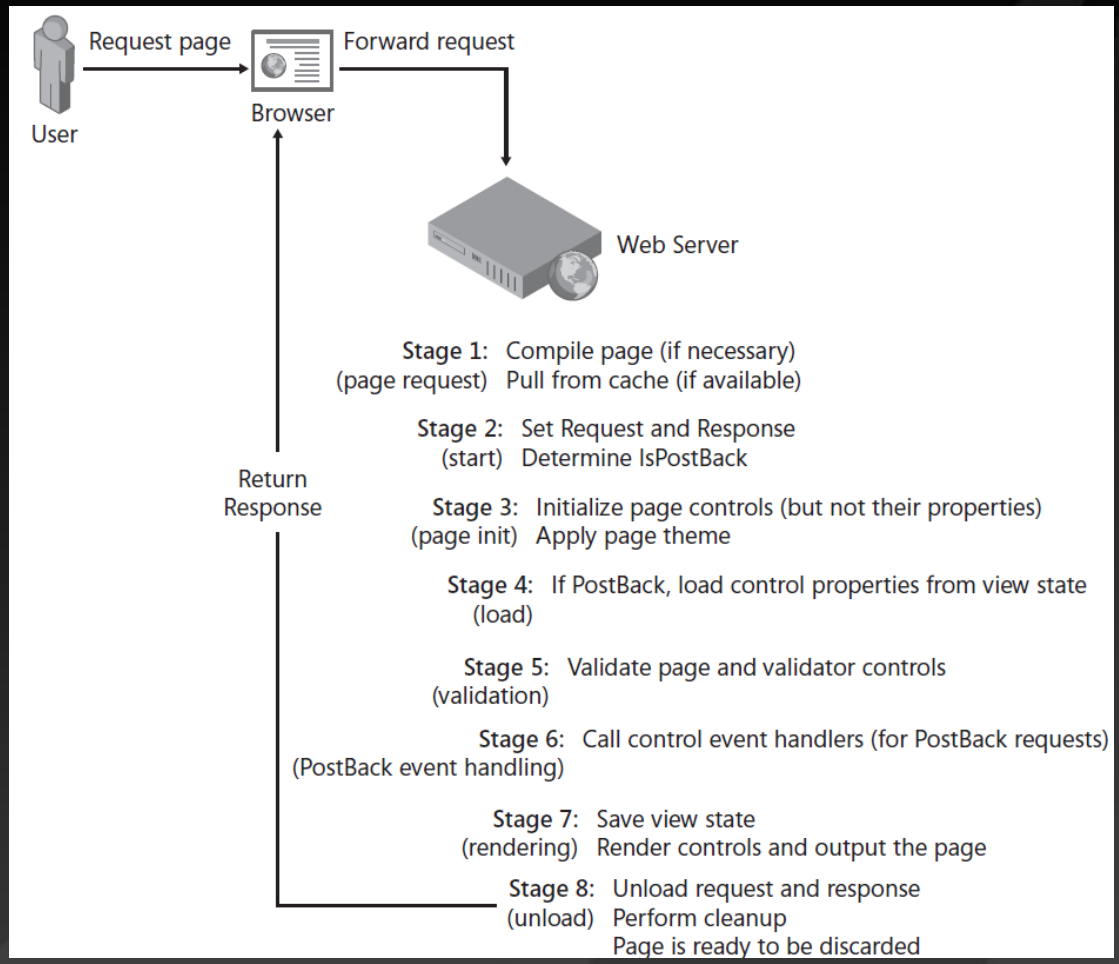
```
<%@ Page Language="C#" AutoEventWireup="true"
    CodeBehind="WebForm1.aspx.cs" Inherits="WebApplicationTP.WebForm1"
    MasterPageFile="~/masterpage.Master" %>
<%@ Register Src="~/WebUC.ascx" TagName="userControl" TagPrefix="uc" %>

<asp:Content ID="content1" ContentPlaceHolderID="head" runat="server">
</asp:Content>
<asp:Content ID="content2" ContentPlaceHolderID="ContentPlaceHolder1"
    runat="server">
     <%--composants de la page--%>
<p>texte vert</p>
<asp:Label Text="texte en vert" runat="server" />
<uc:userControl runat="server" />
</asp:Content>
```

Les users controls

- Maintenant que nous avons une page, un user control, des composants, il est intéressant de comprendre le mécanisme qui lie ces éléments afin de mieux pouvoir aborder la suite :

LES USERS CONTROLS

Cycle de vie d'une page aspx.



Les users controls

- La programmation sous asp.net est de type événementiel, c'est-à-dire à la réponse d'un événement par exemple clic d'un bouton, une action est déclenchée par le serveur qui retourne ensuite au client la réponse.

- En reprenant le code du bouton de notre user control on a :

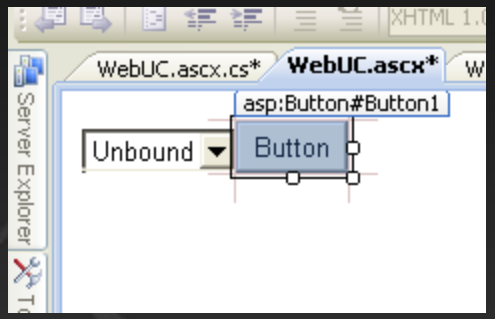
Les users controls

Rajout de l'évènement clic :

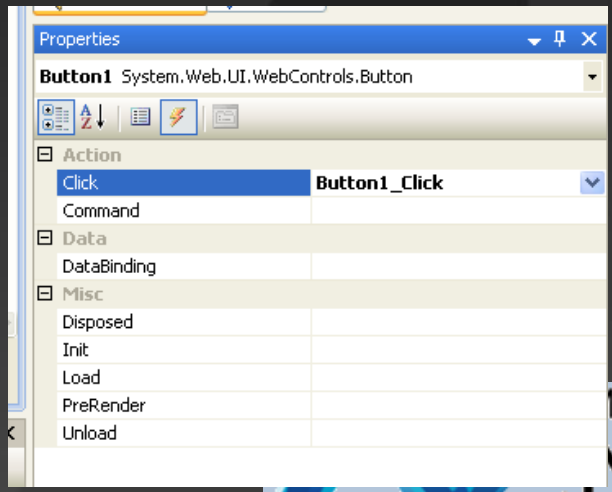
soit directement dans le code :

```
<asp:Button ID="Button1" runat="server"
   Text="Button" onclick="Button1_Click" />
```

Soit dans le designer en faisant un double click sur le bouton



Ou en passant par la fenêtre des propriétés, qui donne accès également aux autres Events !



Les users controls

La signature de la fonction est la suivante :

```
protected void Button1_Click(object sender, EventArgs e)
{

}
```

Nous allons maintenant charger une liste dans la dropdownlist, pour cela, on va d'abord construire une classe :

```
public class Country
{
    #region variables
    private string _name;
    private int _id;
    #endregion


    #region getter / setter
    public string Name
    {
        get { return _name; }
        set { _name = value; }
    }

    public int Id
    {
        get { return _id; }
        set { _id = value; }
    }
    #endregion
}
```

Puis on complète la fonction :

```
protected void Button1_Click(object sender, EventArgs e)
{
    List<DBO.Country> list = new List<WebApplicationTP.DBO.Country>();

    list.Add(new WebApplicationTP.DBO.Country() { Id = 0, Name = "France" });
    list.Add(new WebApplicationTP.DBO.Country() { Id = 1, Name = "Anglettere" });
    list.Add(new WebApplicationTP.DBO.Country() { Id = 2, Name = "Allemagne" });

    DropDownList1.DataSource = list;
    DropDownList1.DataTextField = "Name";
    DropDownList1.DataBind();


}
```

Les users controls

- Maintenant si on compile, on obtient ceci :

Le post-back est le modèle utilisé en ASP.NET pour transmettre les données d'un formulaire HTML avec la méthode POST. Ce modèle consiste à afficher et à traiter les données recueillies du formulaire sur la même page ASPX.

Poster un formulaire sur lui-même permet aux contrôles de la page de non seulement obtenir une programmation événementielle, mais aussi de conserver leur état entre chaque Post. Le postback est également utilisé par défaut en ASP.NET 2.0

Les users controls

Utilisation du post back, pour compléter la textbox

```
protected void Page_Load(object sender, EventArgs e)
{
    if (IsPostBack)
    {
        if (DropDownList1.SelectedItem != null)
        {
            TextBox1.Text = DropDownList1.SelectedItem.Text;
        }
    }
}
```

Interaction avec les pages

Interaction avec les pages

- Une application web est rarement composée d'une seule page. Effectivement la plupart du temps, il y a une multitude de pages, et des actions nécessitent de passer des paramètres de page et en page.

Interaction avec les pages

Provoquer le changement de page :

```
protected void Button2_Click(object sender, EventArgs e)
{
    Response.Redirect("~/WebFormPicture.aspx");
}
```

Le « ~ » correspond à la racine du serveur.
Maintenant, si nous voulons passer des paramètres à cette page soit on peut les passer dans l'adresse soit directement les placer dans la session de l'utilisateur.

```
protected void Button2_Click(object sender, EventArgs e)
{
    Session["Pays"] = TextBox1.Text;
    Response.Redirect("~/WebFormPicture.aspx");
}
```

Interaction avec les pages

Pour récupérer le paramètre sur la nouvelle page :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["Pays"] != null)
    {
        LabelPays.Text = Session["Pays"].ToString();
    }
}
```

Comme son nom l'indique l'objet session existe le temps d'une session d'utilisateur. En configuration par défaut la session sera renouvelée toutes les 20min.

Si on avait passé le paramètre par adresse, la récupération donnerait ceci :

```
protected void Page_Load(object sender, EventArgs e)
{
    LabelPays.Text = Request.QueryString["Pays"];

}
```


Interaction avec les pages

- Il existe encore bien d'autres manières de passer des paramètres aux pages :
  - Application (Global à toute l'application)
  - ViewState
  - Cookies
  - HiddenFields
  - …

Interaction avec les pages

- Une autre problématique qui se pose souvent est la manipulation de fichiers dans une application web.
- En local, le fonctionnement ne pose aucun problème. Ce qui n'est pas le cas chez le client …
  Effet démo ? … non => généralement mal codé. L'une des erreurs les plus fréquentes est de mettre les chemins en dur en considérant qu'un répertoire d'IIS est toujours au même endroit.

Interaction avec les pages

Pour cela ASP.NET fournit une méthode qui permet de retrouver le chemin physique à partir du chemin virtuel. Le problème des chemins absolus est ainsi évité.
Par exemple :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["Pays"] != null)
    {
        LabelPays.Text = Session["Pays"].ToString();
    }
    else
    {
        TextReader tr = new StreamReader(Server.MapPath("~") + "listePays.txt");
        LabelPays.Text = tr.ReadLine();
    }
}
```

AJAX

AJAX

- Les applications web deviennent de plus en plus interactives. Les technologies pour rendre la navigation plus ergonomique n'ont pas cessées d'évoluer. Il y a maintenant des technologies comme Silverlight, Flex, … Cependant, elles ont beaucoup de contraintes. Une autre alternative est l'AJAX, pour Asynchronous JavaScript and XML.

AJAX n'est pas une technologie !

AJAX

- Microsoft a compris l'intérêt de l'AJAX, c'est pourquoi il fournit différentes briques pour faire de l'AJAX.
  - Microsoft AJAX Library (Regroupement de bibliothèques de JS)
  - ASP.NET AJAX Controls Server (contrôles serveurs, permettant l'update de pages partielles)
  - AJAX Control Toolkit (Ensemble de composants développés par la communauté)

AJAX

- Pour l'utilisation d'AJAX dans un projet web, il faut inclure un scriptManager. Pour éviter d'alourdir l'ensemble de l'application, cet objet peut être commun à l'ensemble des pages, et peut donc se mettre directement dans une master Page.

AJAX

Utilisation du script Manager.

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

Maintenant, nous allons mettre en pratique le fait de ne pouvoir recharger qu'une seule partie de la page lorsque cela est nécessaire.

AJAX

```
<asp:UpdatePanel runat="server" ID="firstPanel">
    <ContentTemplate>
        <asp:Label Text="texte en vert" runat="server" />
        <uc:userControl runat="server" />
        <div>
            <asp:Label runat="server" ID="LabelDateTime" /><br />
            Codename :
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
            Name :
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br />
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="insert" />
            <asp:Button ID="Button2" runat="server" Text="clear" OnClick="Button2_Click" />
        </div>
    </ContentTemplate>
</asp:UpdatePanel>
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
    <Triggers>
        <asp:PostBackTrigger ControlID="Button1" />
    </Triggers>
    <ContentTemplate>
        <asp:GridView runat="server" ID="gridView" AutoGenerateColumns="False" DataSourceID="XmlDataSource1">
            <Columns>
                <asp:BoundField DataField="CodeName" HeaderText="CodeName" SortExpression="CodeName" />
                <asp:BoundField DataField="Name" HeaderText="Name" SortExpression="Name" />
            </Columns>
        </asp:GridView>
        <asp:XmlDataSource ID="XmlDataSource1" runat="server" DataFile="~/XmlPays.xml" XPath="Pays/name">
        </asp:XmlDataSource>
        <asp:Label runat="server" ID="Label1" />
    </ContentTemplate>
</asp:UpdatePanel>
```

AJAX

```
protected void Page_Load(object sender, EventArgs e)
{
    LabelDateTime.Text = DateTime.Now.ToLongTimeString();
    Label1.Text = DateTime.Now.ToLongTimeString();
    gridView.DataBind();
}

protected void Button1_Click(object sender, EventArgs e)
{
    XDocument xdoc = XDocument.Load(Server.MapPath("~") + "XmlPays.xml");
    xdoc.Root.AddFirst(new XElement("name", new XAttribute("CodeName", TextBox1.Text), new
     XAttribute("Name", TextBox2.Text)));

    xdoc.Save(Server.MapPath("~") + "XmlPays.xml");

    gridView.DataBind();

}

protected void Button2_Click(object sender, EventArgs e)
{
    TextBox2.Text = "";
    TextBox1.Text = "";
}
```

AJAX

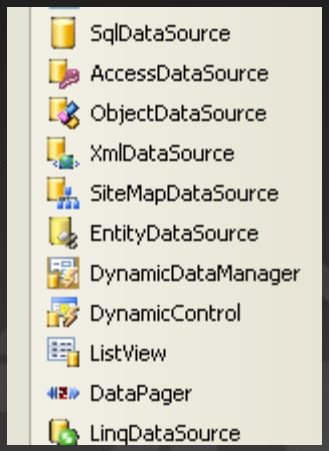
Le format XML :

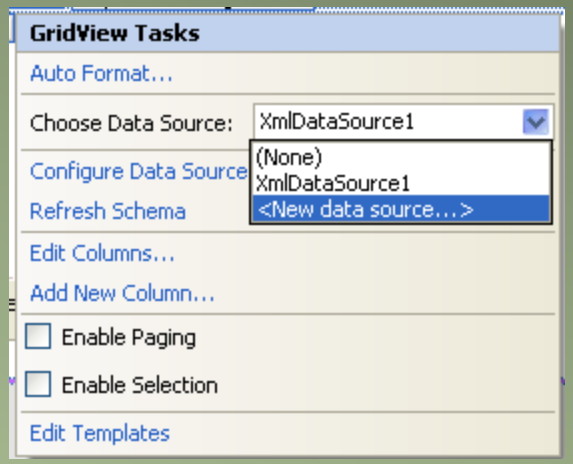
```
<?xml version="1.0" encoding="utf-8"?>
<Pays>
  <name CodeName="fr-Fr" Name="France" />
  <name CodeName="EN-US" Name="USA" />
</Pays>
```

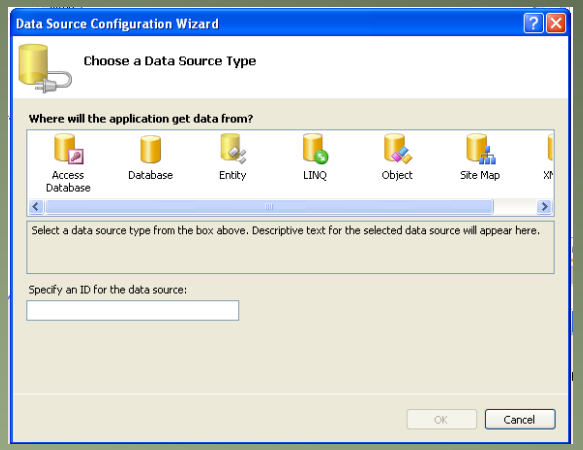
Dans l'exemple fourni juste avant, on peut voir l'utilisation d'un grid view relié directement à une source de données qui va interroger notre fichier XML.
Lors de l'appui sur le bouton insert, l'ensemble de la page est rechargée alors que dans le cas du bouton cancel seul le 1er panel est rechargé.

AJAX

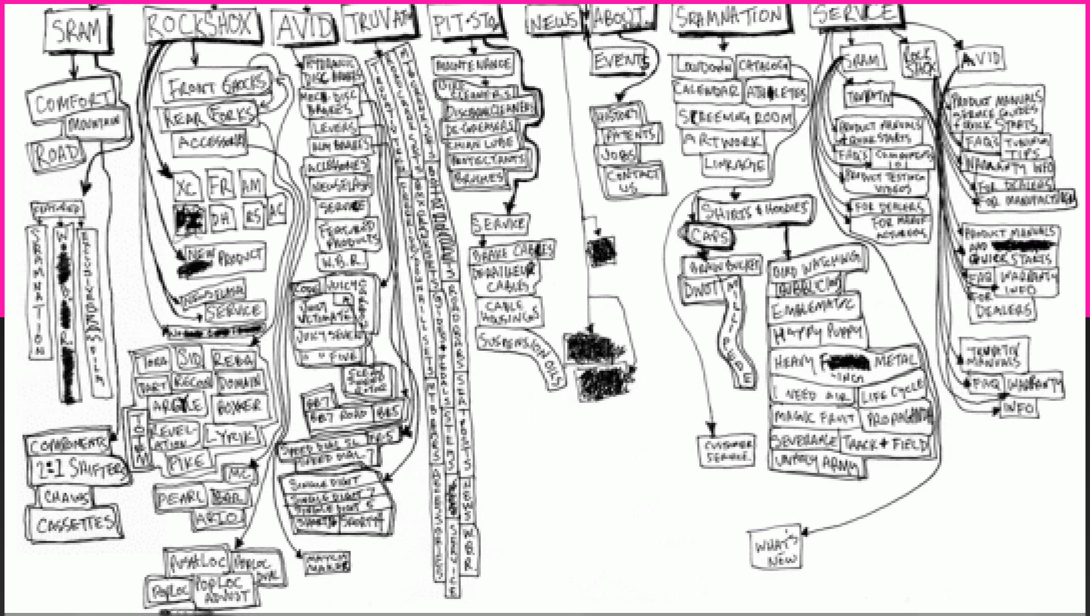
- Le grid view ne s'utilise pas qu'au travers de l'AJAX, on peut l'utiliser tout simplement sur une page. Il peut permettre de modifier directement des valeurs en base, s'il est associé à la bonne source de données.







SiteMap

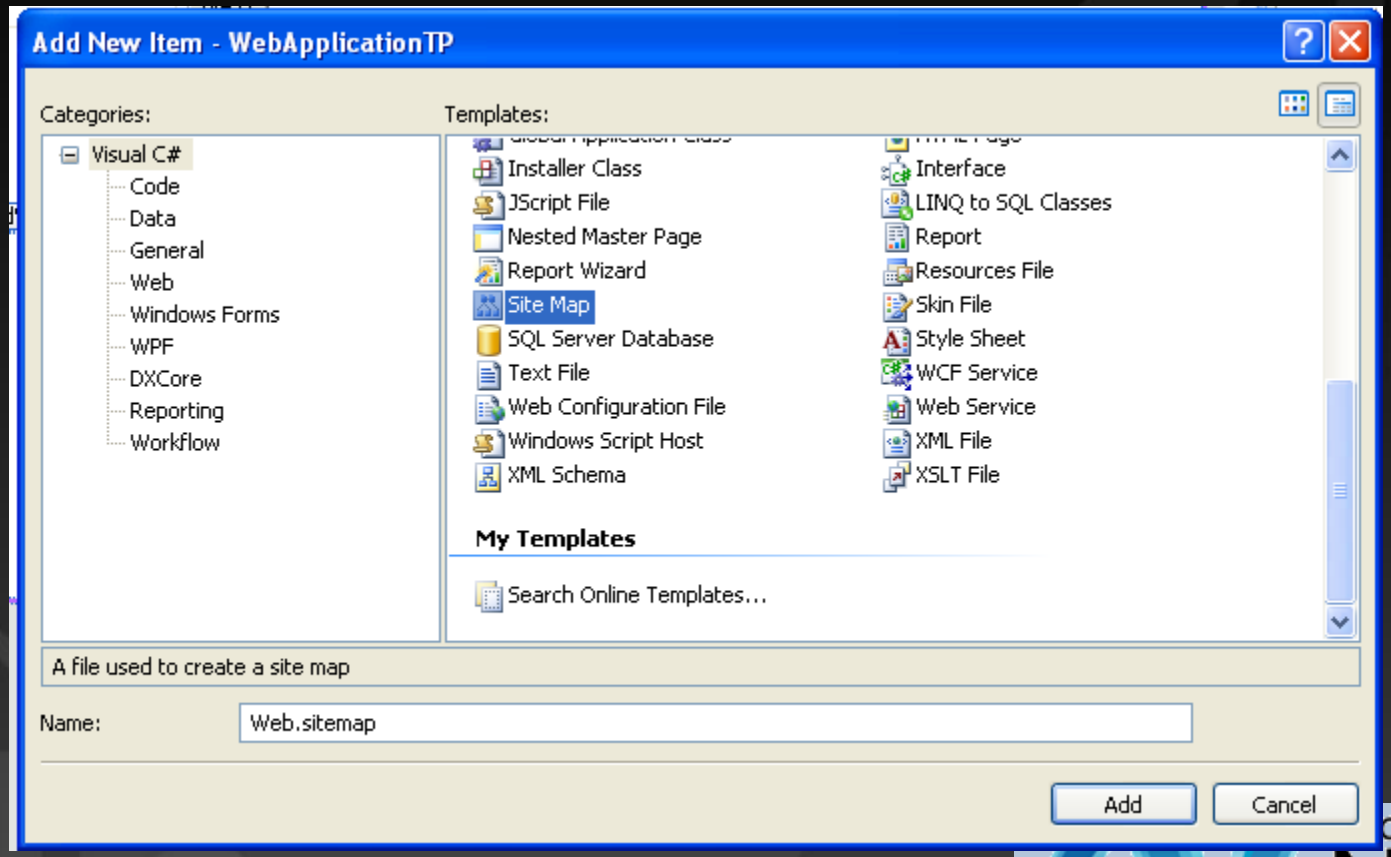

SRAM
ROCKSHOX
AVID
PIT-STOP
NEWS
ABOUT
SRAMNATION
SERVICE
COMFORT
MOUNTAIN
ROAD
EVENTS
CALENDAR
ATHLETES
SCREENING ROOM
ARTWORK
CAPS
CHAINS
CASSETTES
WHAT'S NEW

SiteMap

- Le SiteMap est un fichier XML qui se place à la racine de votre application web. Il décrit l'arborescence de votre application, avec les URLs, description de pages, …
- Il sert à configurer les menus, et les composants de navigation de votre application

SiteMap

- Ajout d'un nouvel élément au projet : sélectionner Site Map puis ajouter



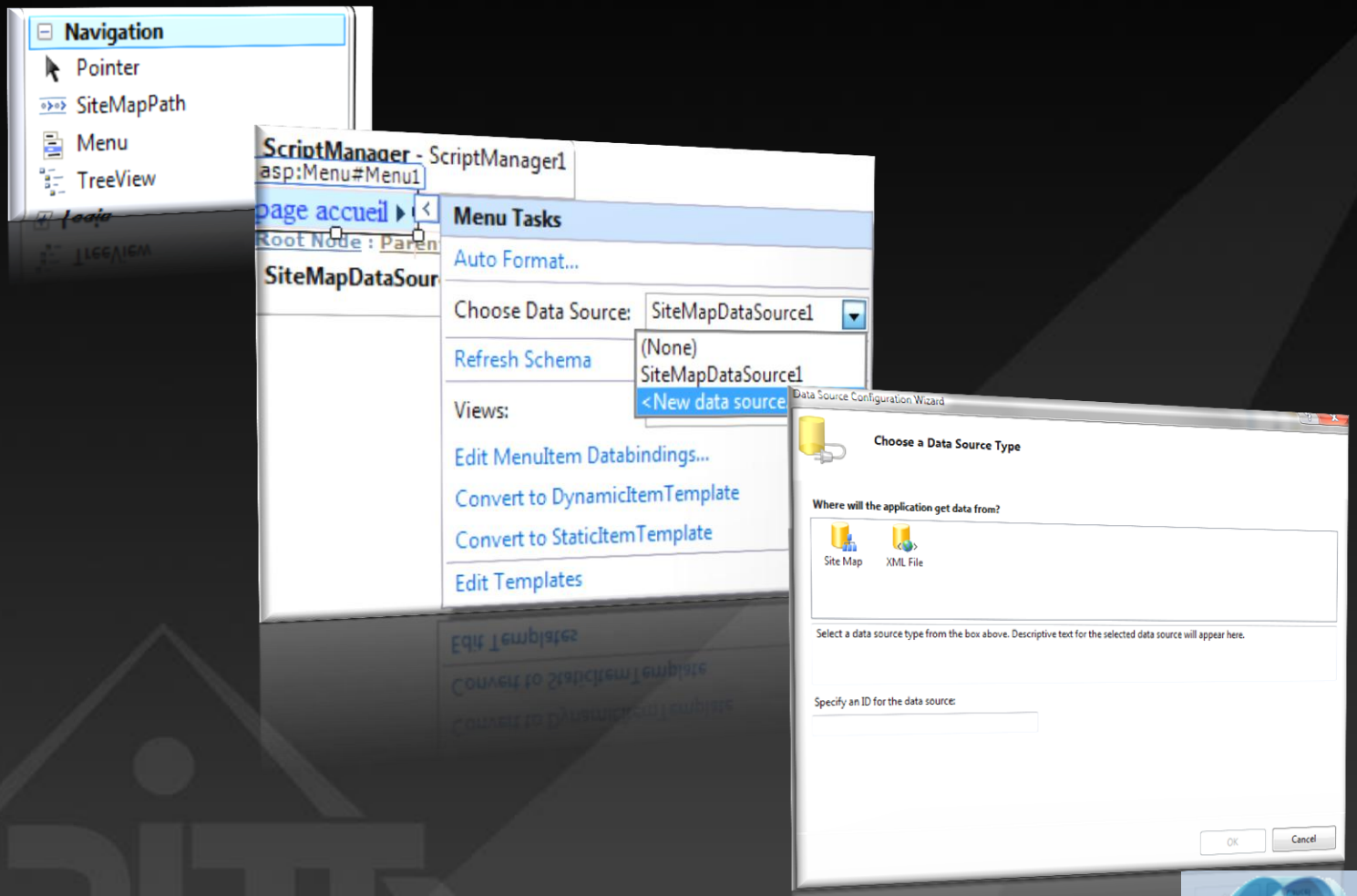
SiteMap

Exemple de contenu d'un site map :

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
  <siteMapNode url="Default.aspx" title="page accueil"  description="1er
    page">
     <siteMapNode url="WebForm1.aspx" title="forme d'exemple"
    description="avec masterpage" >
       <siteMapNode url="WebFormPicture.aspx" title="page avec une image"
    description="page avec une image" />
     </siteMapNode>
     <siteMapNode url="SecondWebForm.aspx" title="page de test"
    description="page de test pour le site map" />
  </siteMapNode>
</siteMap>
```


SiteMap

Pour pouvoir profiter du menu sur toutes les pages, sans duplication de code on peut le placer directement dans la masterpage.



Déploiement

Déploiement

- Notre site fonctionne maintenant dans un serveur intégré à Visual Studio. Cependant, ce n'est pas une solution pour un mode de production. Le but est de faire fonctionner l'application dans un serveur IIS.
- Pour cela il faut installer sur votre poste un serveur d'IIS (Ajout ; suppression / composant windows)

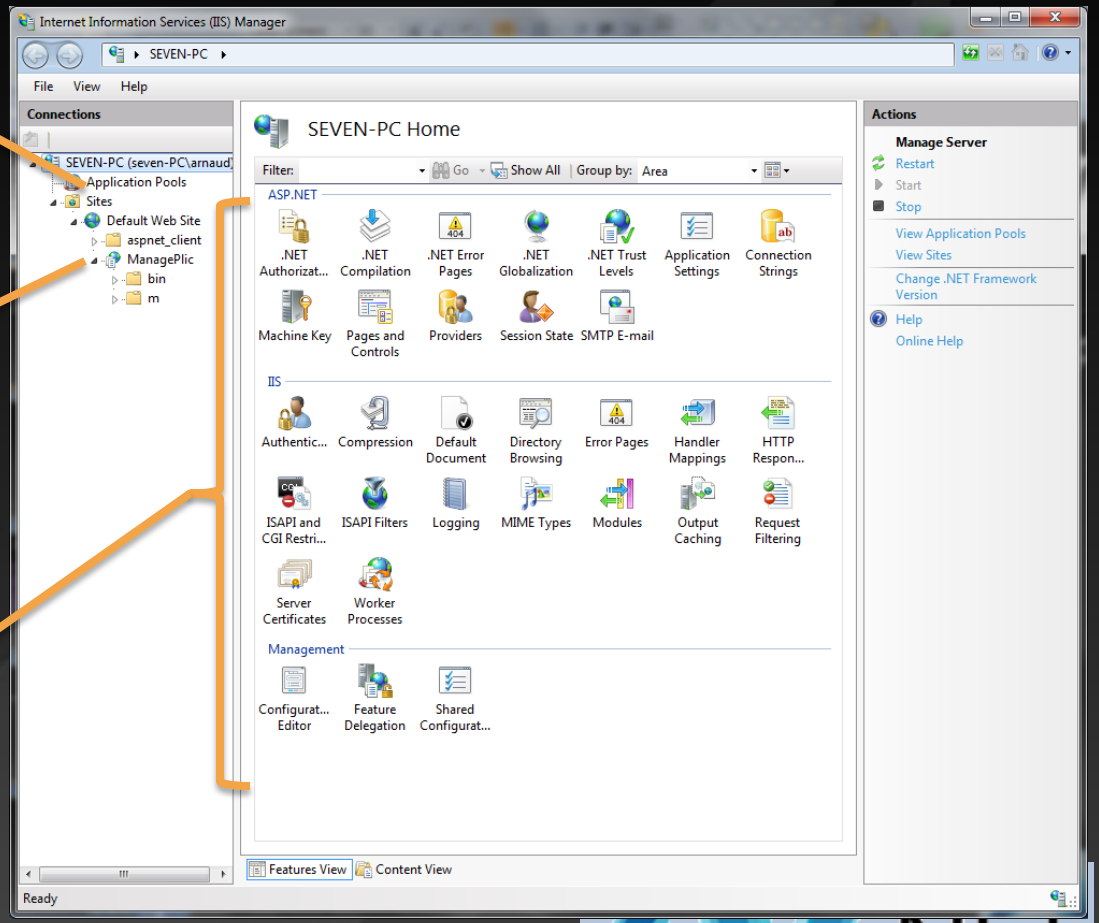
Déploiement

- Pour créer une application web il faut passer par l'outil inetmgr.

Les pools d'applications permettent de pouvoir contrôler un ensemble d'applications (Arrêt, …), dans un même contexte.

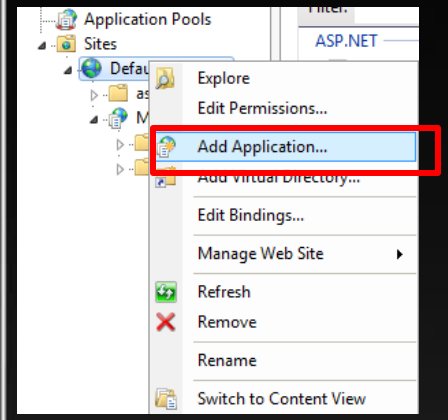
Liste des applications

Options permettant de configurer une application web.



Déploiement

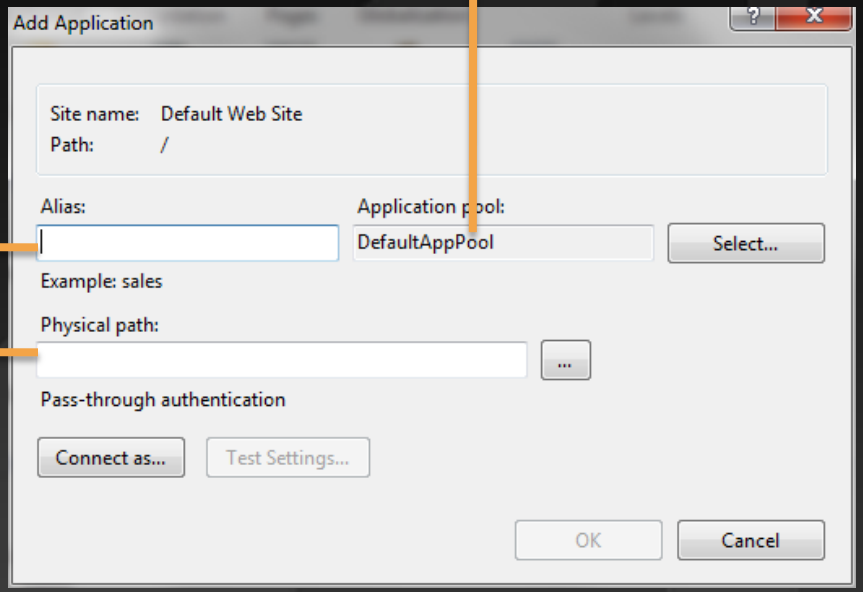
- Création d'une application.



Définition du pool d'application

Nom de l'application qui s'affichera dans le navigateur

Chemin du répertoire où le site est stocké. Par Défaut, le répertoire est placé : C:\inetpub\wwwroot\



Déploiement

- Les fichiers qui doivent être placés dans le répertoire sont les suivants :
  - Le dossier Bin
  - Les fichiers aspx, svc, js, image, css, web.config, …
  - Si la dll du serveur se trouve dans le dossier bin, alors les fichiers *.cs ne sont pas obligatoires.

Autre & ASP.NET 4.0

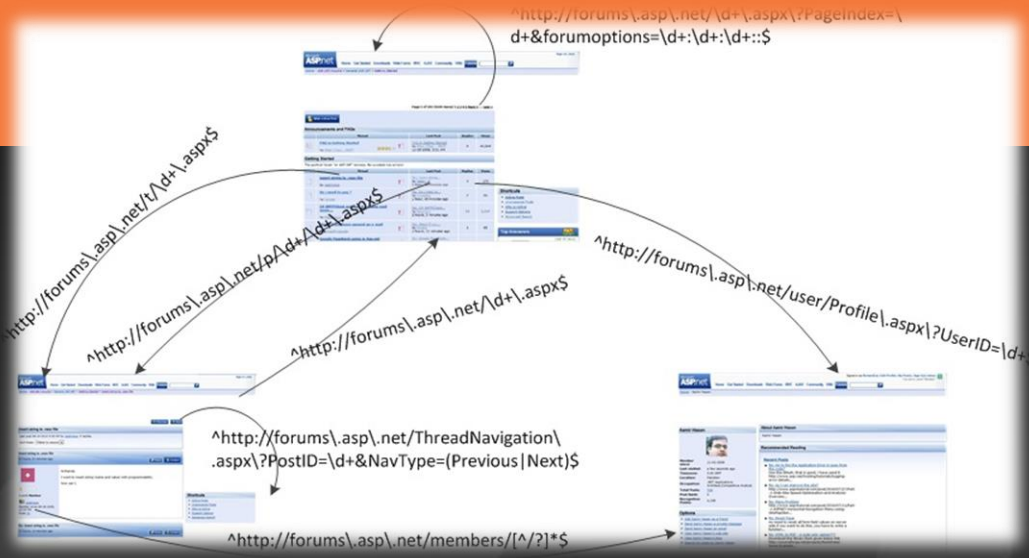
Autre & ASP.NET 4.0

- Nous n'avons pas pu tout voir d'ASP.NET… c'est un sujet vaste, il faudrait encore parler de :
  - ASP.NET MVC
  - Model View Presenter
  - Unity (Injection de Dépendance)

Autre & ASP.NET 4.0

- ASP.NET 4.0 apporte son lot d'amélioration :
  - Meilleur cache (possibilité de le configurer : mémoire, fichier, ...)
  - Définition des balises méta par programmation
  - Meilleur contrôle sur les ID générés.
  - Capacité de rendre les lignes sélectionnées persistantes dans les contrôles de données.
  - Meilleur contrôle sur le HTML rendu dans les contrôles FormView et ListView
  - Prise en charge des navigateurs et des appareils récemment sortis
  - Filtrage de données avec le contrôle QueryExtender
  - Prise en charge améliorée des normes Web et de l'accessibilité
  - ....

URL Routing


d+&forumoptions=\d+:\d+:\d+::$
^http://forums\.asp\.net/t/\d+\.aspx$
^http://forums\.asp\.net/p/\d+/\d+\.aspx$
^http://forums\.asp\.net/\d+\.aspx$
^http://forums\.asp\.net/user/Profile\.aspx\?UserID=\d+
^http://forums\.asp\.net/ThreadNavigation\
.aspx\?PostID=\d+&NavType=(Previous|Next)$
^http://forums\.asp\.net/members/[^/?]*$

URL Routing

- Url routing a été introduit avec l'ASP.NET 3.5 SP1 et popularisé entre autre grâce ASP.NET MVC. Ceci permet d'exposer des URL claires, facilitant le SEO.

- Une autre fonctionnalité de l'URL routing permet de configurer notre application ASP.NEt à accepter des URL ne correspondant pas à des fichiers (*.aspx) présents physiquement sur le disque.
  - Exemple d'URL traditionnelle :
    - http://localhost:2309/Products?category=voiture
  - Exemple d'URL avec routing :
    - http://localhost:2309/products/voiture

URL Routing

- Configuration :
  - Il faut déclarer les routes dès le début de l'application

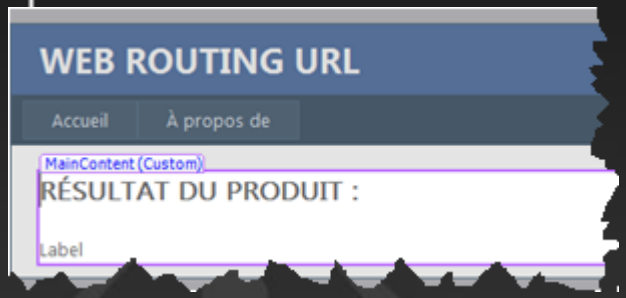
```
void RegisterRoutes(RouteCollection routes)
{
    routes.MapPageRoute(
        "products-browse", // Nom de la route
        "products/{category}", // Url avec les paramètres
        "~/Products.aspx" // Redirection vers la page
        );
}

void Session_Start(object sender, EventArgs e)
{
    // Code qui s'exécute lorsqu'une nouvelle session démarre
    RegisterRoutes(RouteTable.Routes);
}
```

- La fonction Session_Start est disponible dans le fichier global.asax.cs

URL Routing

Product.aspx :

```
<asp:Content ID="HeaderContent" runat="server" ContentPlaceHolderID="HeadC
ontent">
</asp:Content>
<asp:Content ID="BodyContent" runat="server" ContentPlaceHolderID="MainCon
tent">
    <h2>
        Résultat du produit :
    </h2>
    <p>
      <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
    </p>
</asp:Content>
```



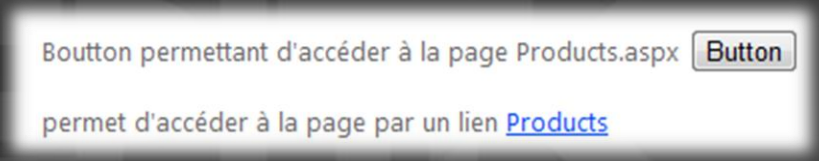

```
namespace WebRoutingUrl
{
    public partial class Products : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string category = Page.RouteData.Values["category"] as string;
            Label1.Text = category;
        }
    }
}
```

URL Routing

Utilisation dans une page web pour l'appel :

```
<p>
   Boutton permettant d'accéder à la page Products.aspx

    <asp:Button ID="Button1" runat="server" Text="Button" onclick="Button1_Click" />
</p>
<p>
permet d'accéder à la page par un lien <asp:HyperLink ID="HyperLink1" runat="server">Products</asp:HyperLink>
</p>
```

```
namespace WebRoutingUrl
{
    public partial class _Default : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            HyperLink1.NavigateUrl = Page.GetRouteUrl("products-browse", new { category = "voiture" });

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            Response.RedirectToRoute("products-browse", new { category = "voiture" });
        }
    }
}
```



QUESTIONS ?