


Microsoft
.net

Services réseaux




COURS N°7 : .NET – IHM « WEB CLIENT »

Antony BRUGERE
a.brugere@gmail.com


Frédéric CHASSAGNE
frederic.chassagne@capgemini.com

Plan du cours



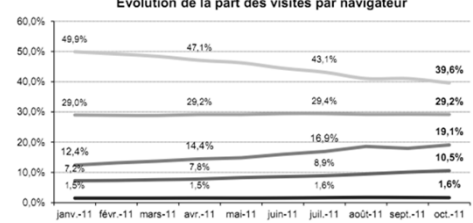
- Avant-propos : Développer pour tous les navigateurs
- Architecture .net
- Quelques objets ASP.net
- Les master pages
- Gérer son plan de site
- Les thèmes

Avant-propos : Développer pour tous les navigateurs



HTML et Javascript = interprétés par plusieurs navigateurs.


Evolution de la part des visites par navigateur



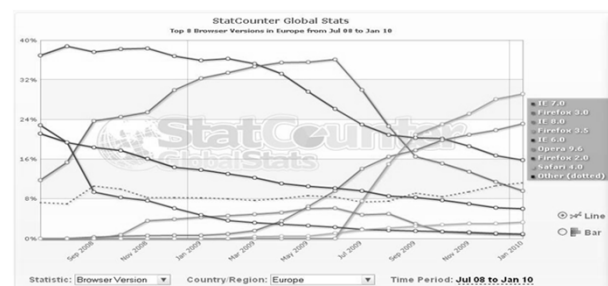
Month	Internet Explorer	Mozilla Firefox	Google Chrome	Apple Safari	Autres
janv.-11	49.9%	29.0%	12.4%	7.2%	1.5%
févr.-11	47.1%	29.2%	14.4%	7.8%	1.5%
avr.-11	43.1%	29.4%	16.9%	8.9%	1.6%
oct.-11	39.6%	29.2%	19.1%	10.5%	1.6%

Mais la réalité est un peu plus compliquée :

Avant-propos : Développer pour tous les navigateurs



Beaucoup de version possible par navigateur :



StatCounter Global Stats
Top 8 Browser Versions in Europe from Jul 08 to Jan 10

Legend: IE 7.0, Firefox 3.0, IE 8.0, Firefox 3.5, IE 9.0, Opera 9.6, Firefox 2.0, Safari 3.0, Other (dotted)

Statistic: Browser Version | Country/Region: Europe | Time Period: Jul 08 to Jan 10

Avant-propos : Développer pour tous les navigateurs

- En résumé : faire un site qui fonctionne sur tous les navigateurs (+smartphone) : c'est impossible ! Sauf si vous faites un... bloc de texte.
- Il existe des "bonnes pratiques" pour éviter au maximum les problèmes de compatibilité.

Avant-propos : Développer pour tous les navigateurs

HTML5 : Terme très commercial en fait. Ajoute tout simplement les objets Canvas, audio et vidéo à l'HTML (comme flash).

Javascript : agrégat de technologie (dom et ecma script et pleins API) non fiable car exécuté dans un environnement non maîtrisé.

Avant-propos : Développer pour tous les navigateurs

Les solutions :

- Utiliser les standards du web (W3C). Des plugins comme "HTML Validator" existent pour vous aider.
- Oublier le "pixel perfect".
- Utiliser les framework JavaScript (jQuery) et préprocesseur CSS (960gs).
- Afficher un message à l'utilisateur si le javascript n'est pas activé ou si sa version de navigateur n'est pas prise en compte.
- Utiliser la méthode "dégradation harmonieuse" ou "amélioration progressive".

Avant-propos : Développer pour tous les navigateurs

- trouver le plus petit dénominateur commun (sélecteur css marche partout).
- responsive web design : visite et utilisable quelque soit le contexte d'utilisation (navigateur + résolution). utiliser media query dans css (selectionne une feuille de style et javascript selon le contexte).
- User agent : a ne pas utiliser ! Car parfois renvoie la mauvaise information ! Demander plutôt "qu'est ce que tu sais faire ?" plutôt que "qui es-tu ?"
- utiliser le modernizr : détection de fonctionnalité, chargement de fichier conditionnel (polyfills JS : remplace des trucs non géré par certains navigateurs)
- productivité : ne pas lier css ou html et javascript.

Avant-propos : Développer pour tous les navigateurs

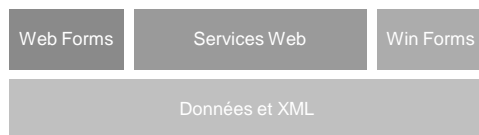
Attention : les vieux navigateurs mettent longtemps à mourir !

Architecture .net

- Web IHM = Client léger ≠ Win IHM = smart client
- Smart client = rafraichissement tps réel
- Web IHM = rafraichissement à la demande
- Avantage Web Forms
 - Pas d'installation, rien ne s'exécute sur le client
- Inconvénient
 - Accès limité aux ressources de la machine

Architecture .net

- En .net : Volonté de coder des objet de haut niveau
- Développement « Web app » et « Win app » très proche



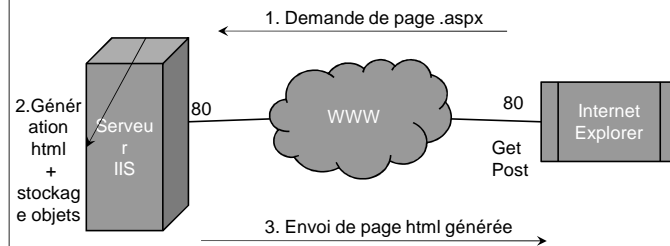
➡ Modèle de programmation unifié

Architecture .net

- ASP.Net
 - Ensemble de pages .aspx
 - 2 possibilités de stockage de code source
 - Code-inline : Contenu dans la page.aspx elle-même
 - Code behind : Contenu dans un fichier .cs à côté
- 1 page aspx = html + contrôles serveur

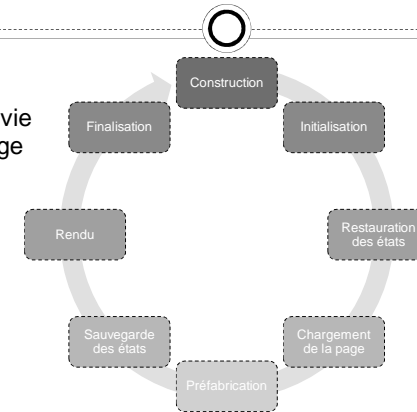
Architecture .net

- Asp.net : un modèle simple



Architecture .net

Cycle de vie
D'une page



Quelques objets ASP.net

- Web = stateless
- Gros problème = maintien des informations de page en page
- Mécanismes possibles
 - Querystring
 - Cookies
 - Viewstate / controlState
 - Session / Application
 - SGBD

Quelques objets ASP.net

- Querystring
 - Syntaxe
`http://localhost/instruments.aspx?ident=21001`
 - Code de récupération
`Value = Request.QueryString["ident"];`

Quelques objets ASP.net

- Viewstate
 - Sorte de Hashtable côté client
 - Stocké dans le flux de retour de la page
 - Syntaxe

```
ViewState["EditionMode-" + this.ID] = Value;
```

Quelques objets ASP.net

- Session / Application
 - Sorte de Hashtable côté serveur
 - Stocké en mémoire
 - Session : 1 par client connecté
 - Persistance : la session
 - Application : 1 par appli asp.net
 - Persistance : tant que le serveur tourne
 - Syntaxe

```
Session["EditionMode-" + this.ID] = Value;
```

Quelques objets ASP.net

- Contrôle serveur
 - Dérive de System.Web.UI.Control
 - Intérêt = capacité à générer 1 fragment de code html
 - Méthode Render(Htmltextcontrol)
 - Méthode RenderChildrenhtmltextcontrol
- 2 sortes :
 - Contrôles html
 - Contrôles Web
- Point commun : runat = "server"

Quelques objets ASP.net

- Evenement postback et non postback
 - Postback = appel de POST ie aller retour IIS
 - Non postback : événement stocké et envoyé lors du prochain postback
- Request
 - QueryString
- Response
 - Redirect("monUrl.aspx")

Quelques Objets ASP.net

- Base d'un appli Web = Objet Page
- Page = conteneur des composants
- Equivalent à un Form en Win appli
- Events Principaux
 - OnInit
 - OnLoad
 - Executer à chaque aller retour client/serveur

Quelques objets ASP.net

- Principaux contrôles Web
 - Button
 - Checkbox
 - FileUpload
 - Image
 - Label
 - Panel
 - ...

Quelques objets ASP.net

- Validation des données
 - Hérite de baseValidator
 - Implémentation totale côté client
 - Exemples :
 - CompareValidator
 - RangeValidator
 - RequiredFieldValidator
 - RegularExpressionValidator

Quelques objets ASP.net

- Propriété de base des validateurs
 - ControlToValidate
 - Display
 - ForeColor
 - IsValid
 - ValidationGroup

Quelques objets ASP.net

- L'accès aux données
 - Identique à l'accès en Win appli
 - Seule différence = objets de présentations
 - 2 objets ihm principaux
 - Datagrid
 - GridView
- ➡ Initialisation par affectation de la propriété DataSource

Quelques objets ASP.net

- Alternative au Datagrid = Repeater
- Avantages
 - Maîtrise des contenus affichés
 - Possibilité d'ajouter des labels, Textbox, Combobox, ...
 - Possibilité d'un affichage alterné des lignes
 - Gestion personnalisée des entêtes et pied de pages
 - Gestion de DataTable ou Collection d'objet (IEnumerable)

Quelques objets ASP.net

- Fonctionnement
 - Code = Affectation de la propriété DataSource & bind des données
 - Design = Ecriture de la balise <asp:repeater ... ></asp:repeater> contenant :
 - <HeaderTemplate>
 - <ItemTemplate>
 - <AlternatingItemTemplate>
 - <SeparatorTemplate>
 - <FooterTemplate>

Quelques objets ASP.net

- Affectation d'une propriété / cellule =
 <%# DataBinder.Eval(Container.DataItem, "Nom") %>
- Exemple


```
<ItemTemplate>
<tr><td>
  <p class="PanelValue">
    <%# DataBinder.Eval(Container.DataItem, "Titre") %>
  </p>
</td>
<td>
  <cc1:Image ID="imgCover" runat="server" ImageUrl= "<%#
  "GetPhoto.aspx?Code=" + DataBinder.Eval(Container.DataItem, « photold") %>" />
  </td>
</tr>
</ItemTemplate>
```

Les master pages

- Concept nouveau (asp.net 2.0)
- Idée = mutualiser les contenus commun entre plusieurs pages
- Héritage de page
 - Fusion lors de la demande de la page
- Utilité
 - Factorisation des entêtes
 - Factorisation des menus

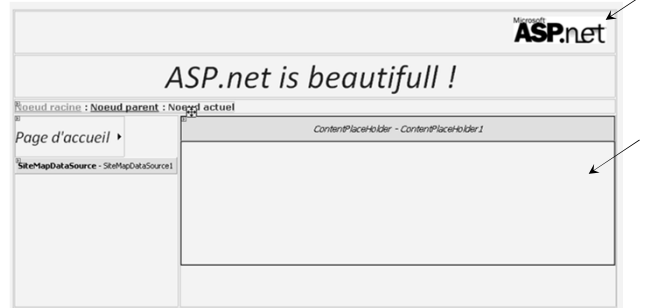
Les master pages

• Mode d'emploi

- Ajouter depuis visual studio 1 master page
- Nom par défaut : MasterPage.master
- 1 master page = 1 page + 1 ContentPlaceHolder
- Chaque page doit définir son master
 - Par visual studio
 - Par code
 - `<%@ Page Language="C#" MasterPageFile="~/dotnet.master" ... CodeFile="Default.aspx.cs" >`

Les master pages

• Exemple d'1 master page



Les master pages

- Master page = sorte de classe mère de page
- Membres publiques accessibles depuis les objets page
- Intérêt
 - Mutualisation de propriétés ou méthodes

```
public string Title
{
    get{ return lblTitle.text;}
    set{lblTitle.Text = value;}
}
```

Dans la
master

```
((ASP.dotnet_master)Master).Title =  
"ASP.net is cool !";
```

Depuis la page

Les master pages

- Alternative au cast (ASP.maMasterPage)
 - Directive @MasterType dans la page
`<%@ MasterType VirtualPath="~/dotnet.master" %>`
- Modification dynamique d'1 masterpage
 - Sur event Page_PreInit
 - Syntaxe
`MasterPageFile = "~/dotnet.master";`

Gérer son plan de site

- Concept nouveau (asp.net 2.0)
- Ensemble de composants de navigation
 - Menu
 - Treeview
 - SiteMapPath
- Point de départ = Web.sitemap

Gérer son plan de site

- Web.sitemap = fichier XML de navigation
- Exemple

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0">
  <siteMapNode url="Default.aspx" title="Page d'accueil" description="Accueil">
    <siteMapNode url="Recherche.aspx" title="Recherche" description="Recherche">
      <siteMapNode url="Search.aspx" title="Basique" description="Recherche basique">
        />
      </siteMapNode>
    </siteMapNode>
    <siteMapNode url="About.aspx" title="About" description="About" />
  </siteMapNode>
</siteMap>
```

Gérer son plan de site

- Dans la page contenant le menu
- Dropper un composant SiteMapDataSource
 - Interface entre fichier xml et composant ihm
 - Rien à coder si fichier = Web.sitemap
 - Sinon renseigner la propriété SiteMapProvider
- Dropper le composant Navigation
 - Rien à coder sauf forme

Gérer son plan de site

- Définition possible de plusieurs .sitemap
 - Ajouter un nœud sitemap dans web.config
 - Définir les providers
 - Renseigner la propriété SiteMapProvider des connecteurs utilisés

Gérer son plan de site

- Exemple d'ajout de providers

```
<siteMap defaultProvider="MonSiteMap1" enabled="true">
  <providers>
    <add
      name="MonSiteMap1"
      type="System.Web.XmlSiteMapProvider"
      siteMapFile="~/MonSiteMap1.sitemap" />

    <add
      name="MonSiteMap2"
      type="System.Web.XmlSiteMapProvider"
      siteMapFile="~/MonSiteMap2.sitemap" />
    </providers>
  </siteMap>
</system.web>
</configuration>
```

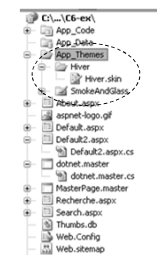
Les thèmes

- CSS
 - Skin html
 - Essentiellement côté client
- Contrôles serveur
 - Notion de thème
 - Fichier .skin
 - Optionnel
 - Fichier CSS
 - Répertoire d'images

Les thèmes

- Dans Visual Studio
 - Répertoire commun App_Theme
 - 1 répertoire par thème
 - Nom du répertoire = nom du fichier
- Modification dynamique d'1 thème
 - Sur l'événement Page_PreInit
 - Syntaxe

```
_theme = Request.QueryString["Theme"];
Page.Theme = _theme;
```



Les thèmes



- Exemple de fichier .skin

```
<asp:Label Runat="server" ForeColor="#004000" Font-Names="Verdana"
Font-Size="X-Small" />
<asp:Textbox Runat="server" ForeColor="#004000"
Font-Names="Verdana"
Font-Size="X-Small" BorderStyle="Solid"
BorderWidth="1px"
BorderColor="#004000" Font-Bold="True" />
<asp:Button Runat="server" ForeColor="#004000"
Font-Names="Verdana"
Font-Size="X-Small" BorderStyle="Solid"
BorderWidth="1px"
BorderColor="#004000" Font-Bold="True"
BackColor="#FFEECo" />
```