Une introduction à ASP.NET Mobile

Sommaire

1. ASP.NET un framework adapté aux Mobiles

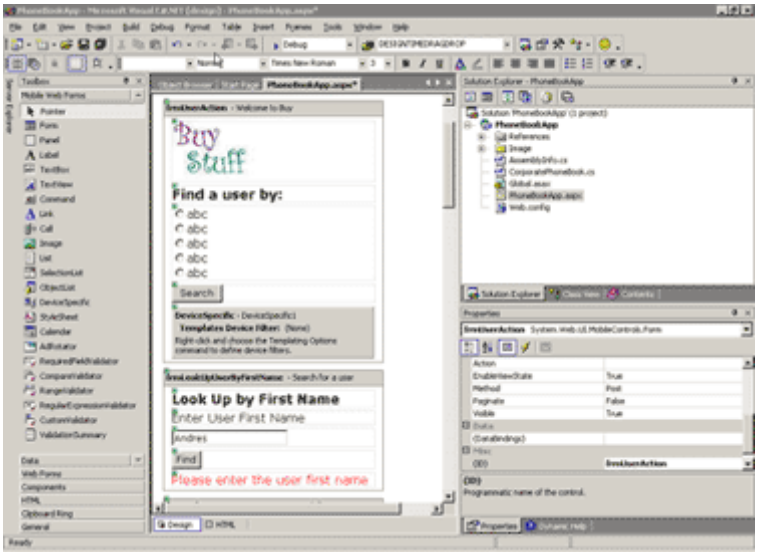
Le framework ASP.NET assure la prise en charge des périphériques mobiles qui n'était pas gérée par son prédécesseur ASP. Cette fonctionnalité est implémentée dans la partie "**Microsoft Mobile Internet Toolkit (MMIT)**". Bien que Mobile Internet Toolkit soit intégré au framework dans la version 1.1, il possède un versionning indépendant.

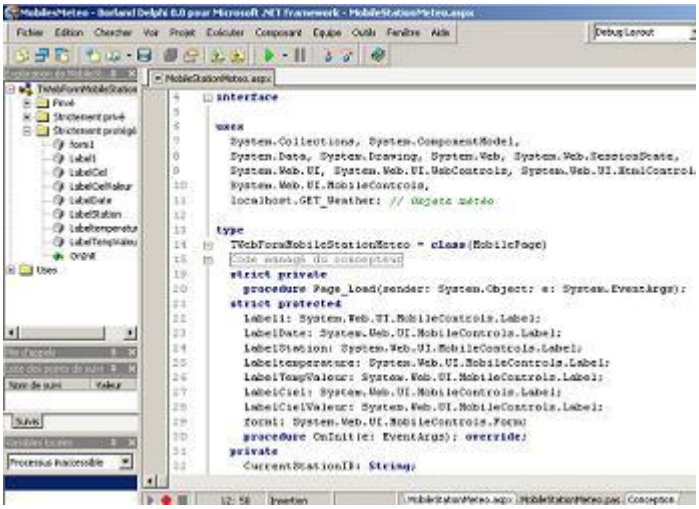
Cette partie du framework n'a rien à voir avec le "Compact Framework" qui permet de développer pour Windows CE.

Dans l'incontournable espace de nommage "**System.Web.UI.WebControls**", ASP.NET propose de nombreuses classes communes au développement web. De-même l'espace de nommage "**System.Web.UI.MobileControls**" correspond aux classes dédiées périphériques mobiles et implémentées dans **Mobile Web Forms Controls**. Ces classes supportent de nombreux périphériques. Le code de l'application .NET développée avec ces classes est le même pour l'ensemble des clients mobiles. La prise en charge des spécificités d'un périphérique (format, navigateur, nombre de lignes de l'affichage, taille de l'écran...) est traitée via le fichier "**Machine.config**" qui indexe les différents **USER_AGENT** des périphériques gérés par le framework. Via ce fichier ASP.NET détecte quel est le périphérique client. Cette spécialisation peut être étendue. Les formats de sortie actuellement implémentés sont : **Html 3.2** dédié au PDA, **cHTML** pour les téléphones i-mode (japon) et **Wml 1.1** pour le WAP.

2. Outils de développement ASP.NET Mobile

Quel que soit l'outil de développement utilisé, le développement visuel **WYSIWYG** n'est pas réellement envisageable dans ce type de développement. En effet en raison de la variété importante des périphériques clients, il est difficile d'uniformiser le rendu.

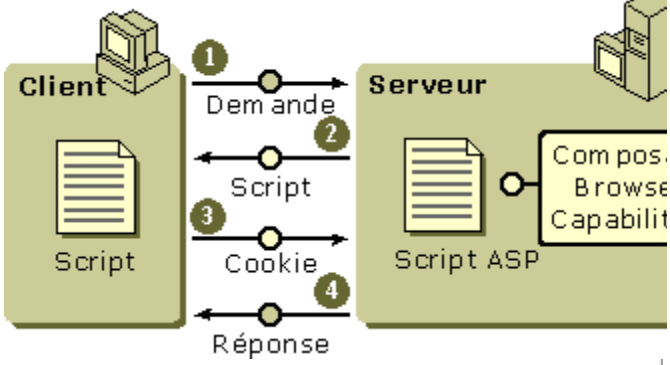
Microsoft Visual Studio .NET: l'intégration avec le système est complète (web serveur, languages "#", filtre/adaptateur, émulateur) avec l'aide des services du kit **Mobile Internet Designer** (inclus dans MMIT). C'est actuellement l'outil le plus complet pour développer des applications ASP.NET Mobile. **Microsoft Webmatrix** : la version 0.6 inclut le support des langages C#, J# et Visual Basic ainsi que l'accès aux données SQL Serveur/MSDE/Access. Cet éditeur **gratuit**permet de développer des applications ASP.NET Mobile.	

Borland Delphi .NET : avec la version Delphi 8.Net la création de form et de contrôle ASP.NET Mobile n'est pas proposée. La prise en charge du développement ASP.NET Mobile n'est pas complète. En conception visuelle, l'EDI Delphi ne reconnait pas les contrôles de type mobiles et produit une erreur. Cette erreur est liée à la partie conception visuelle. La compilation est néanmoins correcte.	

3. Mobiles et Contraintes Web

Les exigences d'un système adapté aux périphériques mobiles sont nombreuses et complexes en fonction du (ou des) périphérique(s) cible(s).

3.1. Détection et redirection d'un client type "périphérique Mobiles"



La redirection vers une page en fonction d'un périphérique type (Desktop ou Mobiles) n'est pas intégrée au framework. Un site web comportant une partie accessible aux clients se connectant avec un périphérique mobile doit prendre ce besoin explicitement en charge dans le code. Le paramètre du fichier config.web "**useFullyQualifiedRedirectUrl**" par défaut à true est nécessaire pour certains Mobiles.

Exemple de code standard inclu dans la page par défaut du site et permettant de faire cette redirection :

```
procedure TWebFormDefault.Page_Load(sender:
System.Object; e: System.EventArgs);
begin
  if (Request.Browser['IsMobileDevice']=
'true' )  then
Response.Redirect(DEFAULT_MOBILEINDEX)
  else
Response.Redirect(DEFAULT_DESKTOPINDEX);
end;
```

Dans le cas du traitement d'un **PostBack**, il faut utiliser la méthode "**RedirectToMobilePage**" de MobilePage.

3.2. Authentification

ASP.NET fournit trois modes d'authentification paramètrables dans le fichier web.config du site : le mode d'authentication par **formulaire, windows et passport**. Les authentifications spécifiques "au monde Windows" ne sont généralement pas implémentées par les périphériques mobiles. Sur IIS

l'authentification "Anonyme" et "de Base" (login/password en clair) fonctionnement généralement. Par contre l'authentification NTLM est un charabia indécryptable pour les mobiles.

Dans le fichier **config.web** du site, la section **authentication** permet de spécifier le mode d'authentification utilisé par le site.

En général l'authentification par formulaire est utilisée :

```
<authentication mode="Forms">
 <forms name=".ASPXAUTH" loginURL="logon.aspx"
protection="All" timeout="60"/>
</authentication>
```

3.3. Session

Idéalement située sur le serveur, la session est liée à la configuration du client. Par exemple un navigateur n'acceptant pas les cookies persistants passera l'état d'authentification le temps d'une requête. La requête suivante, le cookie non présent (puisque refusé), n'indexera plus ce client comme authentifié... Cette limitation est encore plus marquée pour les mobiles. Les cookies ne sont pas supportés par les périphériques mobiles comme les téléphones portables. ASP.NET intègre un paramètrage avancé des différents modes de session. Dans la section <system.web> du fichier de configuration la section **sessionState** indique au framework comment celle-ci est gérée. Voici les différents paramètres possibles :

Attribut	Valeur	Description
mode	"InProc" ou "Stateserver" ou "SQLServer"	- "in-process" - persistant : "serveur d'état" ou "serveur SQL".
cookieless	"true"	- pas de cookie dans la gestion de

		session.
timeout	10	- clôture automatique après 10 minutes d'inactivité.

exemple de configuration sans cookie avec un timeout de 10 minutes

```
<sessionState mode="InProc" cookieless="true"
timeout="10"/>
```

ASP.NET permet de fonctionner avec les mobiles sans avoir à développer une gestion de session particulière comme il était nécessaire encore avec ASP et la gestion de session via l'url type "url?SessionId=ad1j8k7bbjhoo12871". Il est à noter que pour une session maintenue sans cookie, l'url présentée au client est préfixée de l'**ID de session** :

http://www.Delphinaute.be/ChapsAndChips/(agemaq55xturnxrpyv0rh2ag)/MobileDefault.aspx

Attention, des problèmes de redémarrage automatique de l'application ("recycling") sont référencés dans différentes situations par exemple lors du parcours de l'arborescence avec un anti-virus (**http://support.microsoft.com/default.aspx?scid=kb;en-us;324772** ou **http://support.microsoft.com/kb/316148/EN-US/**). Dans cette situation, avec le mode "InProc", les données de session sont réinitialisées et donc perdues ! La résolution classique est le stockage persistant (Stateserver ou SQLServer) de la session. Par exemple, en mode "**StateServer**" via le**service ASP.NET State Service** en ajoutant l'attribut "StateConnectionString" indiquant quel serveur est utilisé :

stockage de session via le service "ASP.NET State service"

```
<sessionState mode="StateServer"
stateConnectionString="tcpip=localhost:42424"
        cookieless="true" timeout="10"/>
```

3.4. ViewState

Avec ASP.NET, l'état des pages est automatiquement sauvegardé dans une zone **VIEWSTATE**. Cette zone d'information est postée au client. Le VIEWSTATE pouvant être important, en général les capacités des périphériques mobiles ne sont pas suffisantes pour gérer ces informations. Les caractéristiques du/des client(s) cible(s) déterminent le choix de l'utilisation de cette fonctionnalité intégrée.

4. Mobiles Spécifiques et Contrôles ASP.NET Mobile

Deux niveaux sont offerts pour personnaliser la prise en charge des périphériques par un contrôle Mobile ASP.NET. Avec le premier niveau, il est possible de spécifier des filtres (**filter**) dans le fichier web.config afin de spécialiser une partie de l'application pour les périphériques mobiles qui valident la condition du filtre. Le second niveau utilise les adaptateurs. Un adaptateur (**adapter**) est une implémentation spécifique pour un format de sortie d'un contrôle. Chaque contrôle ASP.NET Mobile est mappé sur un adaptateur pour chacun des formats de sortie. Il est possible de mettre en oeuvre un nouvel adaptateur personnalisé via son référencement dans le fichier web.config

4.1. deviceFilters/filter

Les filtres sont recensés dans le fichier web.config du site dans la section <system.web><**deviceFilters**>.

Exemple de la liste des filtres du projet Delphi Portal Starter Kit

```
<deviceFilters>
    <filter name="isJScript"
compare="javascript" argument="true"/>
    <filter name="isPocketIE"
compare="browser" argument="Pocket IE"/>
    <filter name="isHTML32"
compare="preferredRenderingType"
argument="html32"/>
</deviceFilters>
```

Le controle ASP.NET Mobile "**DeviceSpecific**" associé à "**Choice**" et son argument **"filter**" utilisent cette liste pour déterminer le filtre qui correspond au client. Cette logique correspond à une figure algorithmique du type "Case" (Pascal) ou du type "Switch" en C avec comme opérande le type de périphérique du client mobile du site.

Extrait du source du contrôle Mobile 'Text.ascx' du projet Delphi Portal Starter Kit:

```
<mobile:panel id=summary runat="server">
 <mobile:devicespecific id=DeviceSpecific1
runat="server">
   <choice filter="isJScript">
    <contenttemplate>
      <aspnetportal:title runat="server"/>
         <%#mobileSummary %>
            <asp:linkbutton id="LinkButton1"
runat="server"
               commandname="Details"
text="(+)" visible="<%# mobileDetails !=
String.Empty %>">
            </asp:linkbutton>
    </contenttemplate>
  </choice>
 </mobile:devicespecific>
</mobile:panel>
```

Le contenu du composant **panel** du contrôle **DeviceSpecific1** est variable en fonction de la validité du filtre appliqué au client. Dans cet exemple, le panel "summary"

sera enrichi de la partie**<contenttemplate>** dans le cas où le client est compatible JavaScript.

4.2. mobileControls/device

Les adaptateurs personnalisés sont référencés dans le fichier Web.config du site dans la section <system.web>**<mobileControls>**. Ces adaptateurs (cf. Design Pattern) correspondent à une implémentation personnalisée pour un contrôle et un format de sortie.

Liste des adaptateurs recensés du fichier web.config du projet Delphi Portal Starter Kit :

```
<mobileControls>

  <device name="PortalHtmlDeviceAdapters"
inheritsFrom="HtmlDeviceAdapters">
    <control
name="MobileControls.TabbedPanel,ASPNET.Starte
rKit.delphiPortal.PortalCSSDK"

adapter="MobileControls.HtmlTabbedPanelAdapter
,ASPNET.StarterKit.delphiPortal.PortalCSSDK"/>
    <control
name="MobileControls.LinkCommand,ASPNET.Starte
rKit.delphiPortal.PortalCSSDK"

adapter="MobileControls.HtmlLinkCommandAdapter
,ASPNET.StarterKit.delphiPortal.PortalCSSDK"/>
  </device>

  <device name="PortalChtmlDeviceAdapters"
inheritsFrom="ChtmlDeviceAdapters">
    <control
name="MobileControls.TabbedPanel,ASPNET.Starte
rKit.delphiPortal.PortalCSSDK"

adapter="MobileControls.ChtmlTabbedPanelAdapte
r,ASPNET.StarterKit.delphiPortal.PortalCSSDK"/
>
```

```
  </device>

  <device name="PortalWmlDeviceAdapters"
inheritsFrom="WmlDeviceAdapters">
    <control
name="MobileControls.TabbedPanel,ASPNET.Starte
rKit.delphiPortal.PortalCSSDK"

adapter="MobileControls.WmlTabbedPanelAdapter,
ASPNET.StarterKit.delphiPortal.PortalCSSDK"/>
  </device>

</mobileControls>
```

Dans cet exemple, pour chacun des trois formats (html, chtml, wml) supportés par ASP.NET Mobile un nouveau "device" est recensé. Pour chacun de ces "device", le contrôle *"TabbedPanel"* est recensé avec un attribut "**adapter**" qui référence dans quelle classe de quel assemblage le nouveau contrôle est implémenté. Le contrôle *"TabbedPanel"* est disponible pour les trois formats standards. Le contrôle "*LinkCommand*" est uniquement disponible pour le format html.

5. Tests

5.1. Conditions de test

Avant de démarrer les tests, il est important de vérifier les différents points abordés précédemment. L'authentification ne devrait pas causer d'effet de bords. Par contre une mauvaise anticipation des différents autres paramètres peut avoir des conséquences dramatiques sur les tests.

5.1. Outils de test

Une page dédiée Mobiles est visualisable avec votre navigateur habituel. Cependant pour tester le rendu réel ainsi que la compatibilité avec le système cible, il est fortement conseillé d'utiliser un émulateur ou encore mieux de tester avec le périphérique cible. Chaque constructeur fournit généralement un émulateur permettant de vérifier la compatibilité avec ses périphériques. D'autres éditeurs proposent des émulateurs multipériphériques. Ces émulateurs permettent de simuler des périphériques génériques ou un périphérique plus spécifiquement ciblé. Les **SDK Microsoft Pocket PC et SmartPhone** sont intégrés dans environnement Visual Studio .NET. Nous utiliserons **Openwave V7 simulator** pour tester le projet accompagnant cet article en raison de sa mise en oeuvre trés simple et indépendante d'un outil de développement particulier.

6. Exemple d'application ASP.NET Mobile : "Météo"

Choix d'une station	Météo pour la station
	

6.0. Téléchargement du projet Delphi .NET : "Météo"

Téléchargement du projet Delphi .NET

6.1. Objectifs

Cette application ASP.NET Mobile est un exemple ayant pour cible différents périphériques : les navigateurs "du monde Desktop" et les périphériques Mobiles. Cet exemple simple met en oeuvre des contrôles Mobiles standards et un exemple de filtre.

6.2. Outils utilisés : .NET 1.1, Delphi 8 .NET, C#

La version du framework utilisée est **Microsoft .NET Framework 1.1.**

Delphi 8 .NET est utilisé pour le développement de ce projet de démonstration. Il est à noter que Delphi 8 ne gère pas la conception visuelle des **MobilePage** : il n'est pas possible de visualiser la page créée dans l'onglet "conception" de cet EDI. A la première ouverture de la fiche, le programme indique une erreur de "paramètre incorrect". A la seconde tentative la fiche s'ouvre mais l'onglet "conception" est généralement vide. Ceci n'a aucune conséquence notable.

ASP.NET & l'utilisation de composants spécifiques
La déclaration suivante dans le code la page ASPx indique au framework où est référencé le contrôle ("UnControlMobile") :

<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls" Assembly="System.Web.Mobile" %>
Ce contrôle est défini dans la page via une balise du type : *<mobile:UnControlMobile/>*

Les étapes suivies avec Delphi 8 .Net pour construire une page Mobiles du projet accompagnant cet article sont résumées ci dessous. Ces étapes permettent de contourner l'erreur Delphi de conception avec "**le concepteur de premier niveau**" qui (a priori) correspond à un concepteur typé "WebForm".

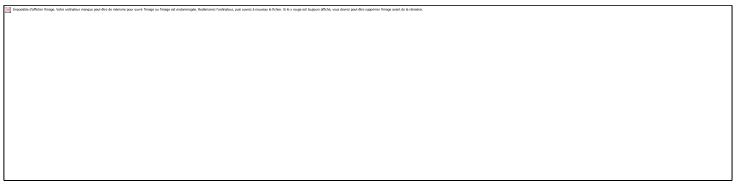
Etape	Procédure
1	Créer une page ASP.NET classique contenant vos contrôles mobiles (onglet code aspx).
2	Passer en "conception" afin de recencer dans le code behind les contrôles serveurs utiles à la page.
3	Vérifier dans le code behind (.pas) la déclaration automatique des contrôles serveurs utilisés dans le code behind.
4	Transformer la déclaration de la page en une déclaration de page mobile dans le code behind et dans le code ASPx : "class(System.Web.UI.Page) devient "class(MobilePage)"). De même la déclaration de la webform (<form ..>) dans le code ASPx doit être modifiée pour correspondre à une mobileform (<mobile:Form>).
5	Compiler

Le web service "GET_Weather.asmx" est développé en C#. Aucune modification n'a été apportée au code source original.

6.3. Web Service Météo et Client Proxy Soap Delphi

Cet exemple utilise le webservice **"GlobalWeather" de CapeScience** pour obtenir les informations météorologiques. Le web service proxy, développé par Bernhard Gaul @**http://www.bgxcomponents.com**, est nécessaire sur le serveur où est installé l'application ASP.NET Mobile. Il chaîne les demandes vers le webservice CapeScience : c'est un Web Service Proxy.

L'importation du WSDL dans Delphi est effectuée via l'ajout d'une référence Web pointant sur le web service proxy local.



chaînage des requêtes vers le webservice GlobalWeather : proxy

Dans la définition du proxy client soap (**'Web References\localhost\localhost.GET_Weather.pas'**) automatiquement créée par Delphi lors de l'importation, une ligne de code supplémentaire est nécessaire dans le constructeur :

```
Type
....
GET_Weather =
class(System.Web.Services.Protocols.SoapHttpCl
ientProtocol)
....
....

constructor GET_Weather.Create;
begin
  inherited Create;

Self.Credentials:=System.Net.CredentialCache.D
efaultCredentials; // ++
  Self.Url :=
'http://localhost/meteows/GET_Weather.asmx';
end;
```

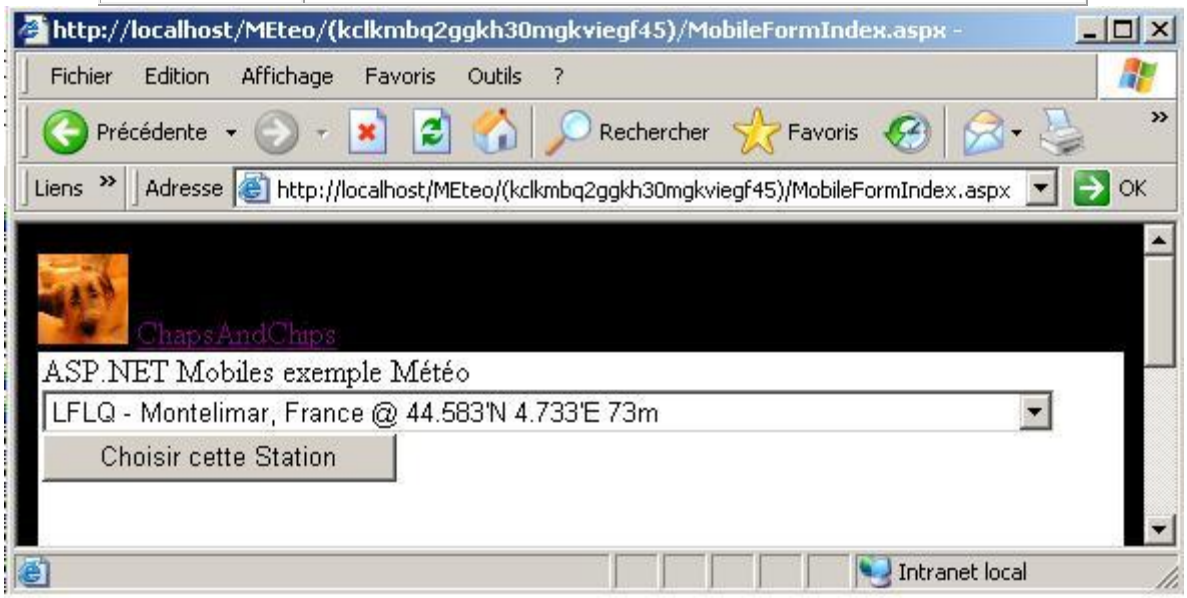
Ce code copie les informations d'authentification système du contexte de sécurité en cours dans le client SOAP delphi. Pour les applications ASP.NET, les informations d'authentification par défaut sont les informations d'authentification utilisateur de l'utilisateur en cours ou de l'utilisateur ayant effectué un emprunt d'identité. Cette affectation permet au client du web service de négocier avec le serveur pour déterminer quel mode d'authentification utiliser (NTLM et/ou Kerberos).

6.4. Installation et test de l'exemple

Etapes	Etapes à suivre pour installer la démonstration
1	Après avoir décompressé le fichier ZIP contenant le source du projet, créer un répertoire virtuel "meteows" pointant sur "<chemin d'installation>\Meteo\GET_Weather". Ce site correspond au webservice proxy local.
2	Créer un second répertoire virtuel (l'application ASP.NET Mobile) pointant sur "**<chemin d'installation>\Projet"**. Ce site correspond à l'application ASP.NET Mobile
3	Fixer la sécurité des répertoires virtuels : - pour "meteows" (étape1) seule l'authentification "Windows" est selectionnée. - pour l'application ASP.NET Mobile (étape2) fixer la sécurité sur "Authentification de base" et "Anonyme" - pour "meteows" (étape1) seule l'authentification "Windows" est selectionnée.

	Description des exigences et outils pour tester l'application "Météo"

Exigences	Il est nécessaire d'être connecté à Internet afin que le web service proxy accède au web service WeatherReport de CapeScience. La configuration de type "StateServer" exige aussi que le service "ASP.NET State Service" soit démarré
Outils	Vous pouvez utiliser Internet Explorer ou votre navigateur favori pour vous connecter à l'application. Dans ce cas, c'est un flux HTML3.2 qui sera transmis à votre navigateur. Pour tester le flux WML vous pouvez utiliser OpenWave V7 simulator ou un émulateur de votre choix.



Test de l'application avec Microsoft Internet Explorer

6.5. Remarques

Cet exemple est assez "minimaliste" mais peut-être trés facilement étendu. Par exemple la transmission des données des stations météo mériterait d'être conserver en cache afin de ne pas accéder à chaque requête (de choix de station) au webservice "ReportWeather de CapeScience". De même le "look and feel" de l'application n'est pas "très travaillé", le but de cette démonstration n'étant pas de faire du design web. Cette partie est

facilement améliorable. Ces améliorations constituent un bon exercice de test afin de vérifier vos connaissances d'ASP.NET Mobile.

7. Conclusion

Dans cette présentation générale, les différentes composantes mises en oeuvre et permettant de développer avec ASP.NET Mobile sont présentées. La configuration d'un site adapté aux périphériques mobiles est détaillée pour les aspects fondamentaux. Vous êtes maintenant paré à proposer à la flotte de commerciaux un outil d'alerte ou à affronter les portables d'une tribu de jeunes ! Dans un prochain article, les différents contrôles inclus dans le runtime seront plus spécifiquement abordés. L'extension d'un contrôle Mobiles sera implémentée dans le projet exemple. A bientôt