

Une première application Apollo basée sur HTML

par Olivier Bugalotto ([Mes articles](#))

Date de publication : 16/05/2007

Article expliquant comment réaliser une application Apollo basée sur HTML

- I - Avant-propos
- II - Utiliser Eclipse
- III - Accéder à Apollo depuis javascript
- IV - Les sources

I - Avant-propos

La technologie Apollo nous permet de réaliser des applications entièrement écrites avec HTML/Javascript/AJAX.

Le runtime d'Apollo utilise la librairie **WebKit**, cette librairie permet aux développeurs d'intégrer facilement un moteur de rendu HTML. Nous la retrouvons comme fonctions du système d'exploitation MAC OS X et notamment dans le navigateur **Safari**.

L'environnement de développement Flex Builder 2 ne propose pas de projet d'application basée sur du HTML, il faut donc utiliser deux outils : Apollo Debug Launcher (ADL) pour déboguer l'application et Apollo Developer Tool (ADT) pour la packager sous la forme d'un fichier d'installation (fichier d'extension .air).

Le développement d'une application HTML sous Apollo demande au minimum deux fichiers : un fichier de configuration XML (voir : **Votre première application Apollo**) et une page HTML.

Voici le fichier de configuration, qui est identique à celui d'une application Apollo basée sur Flex :

```
<?xml version="1.0" encoding="utf-8"?>
<application xmlns="http://ns.adobe.com/apollo/application/1.0.M3" appId="HelloApollo"
version="0.1">
  <properties>
    <name>Hello world!</name>
    <publisher>Iteratif</publisher>
    <description>HTML-based application Apollo</description>
    <copyright>&#xA9; 2007</copyright>
  </properties>
  <rootContent systemChrome="standard" visible="true" width="160"
height="180">HTMLApollo.html</rootContent>
</application>
```

Et, bien entendu, une page HTML :

```
<!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>
<style>
body {
  margin: 0;
}

.container {
  background-image:url(default.png);
  width:150px;
  height:150px;
}

.helloText {
  font: 14px "Arial";
  font-weight: bold;
  position: absolute;
  top: 24px;
  left: 30px;
}
</style>
</head>
<body>
  <div class="container">
```

```
<h1 class="helloText">Hello, World!</h1>
</div>
</body>
</html>
```

Pour tester, il suffit d'utiliser l'outil en ligne de commande ADL :

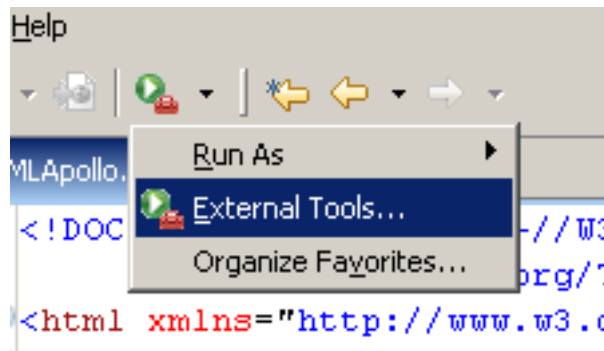
```
adl HelloWorld-app.xml
```

II - Utiliser Eclipse

Pour ce tutoriel, nous allons utiliser le plugin Aptana pour Flex builder 2 qui est un véritable environnement de développement intégré dédié au web : **Aptana** (*il suffit de suivre les instructions pour installer ce plugin*).

Lorsque le plugin est installé, il nous suffit de créer un simple projet et d'y ajouter notre fichier de configuration d'application XML et notre page HTML.

Nous allons en suite configurer l'appel à ADL à l'aide de Flex builder 2 en configurant un outil externe de la manière suivante :



Nous devons définir une nouvelle configuration :

External Tools

create, manage, and run configurations

Run a program

Configurations:

Name: New_configuration (1)

Location: C:\Program Files\Adobe\Flex Builder Moxie\sdk\bin\adl.exe

Working Directory: \${workspace_loc:/HTMLApollo}

Arguments: HelloApollo-app.xml


Note: Enclose an argument containing spaces using double-quotes ("").

Le chemin d'accès vers ADL

Le dossier de votre projet

Le fichier de configuration de l'application

Bien entendu, nous lui donnons un nom : HTMLApollo et nous testons... logiquement, nous ne devrions pas avoir de problème.

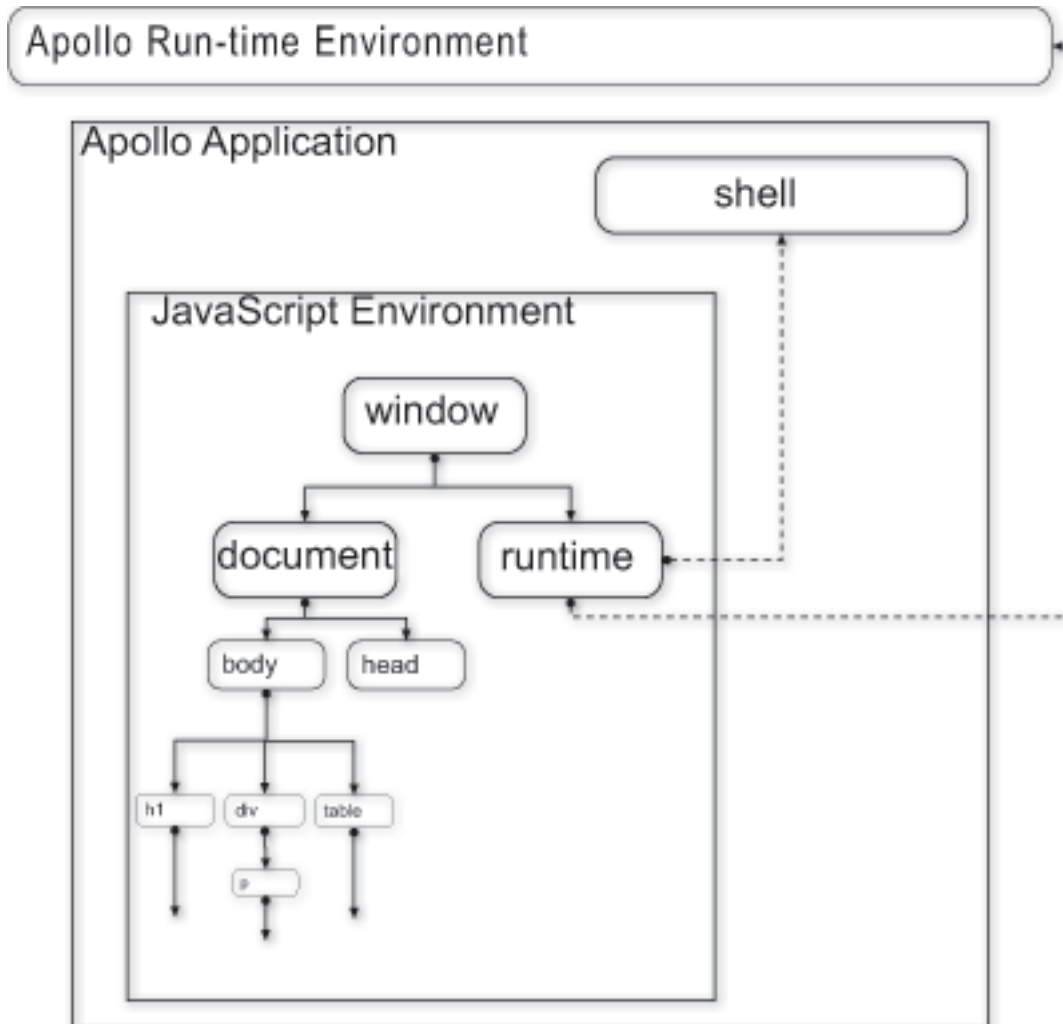
 *Aptana permet de lire les fichiers AS, ce qui peut poser problème avec des projets flex et actionscript, nous devons donc désactiver l'utilisation de fichier AS par l'éditeur d'Aptana. Pour se faire, il suffit de se rendre dans les préférences (Window > Preferences). Aller*

dans Aptana > File View et retirer l'extension .as. Ensuite, aller dans General > Editors > File associations et associer l'extension .as à l'éditeur de flex.

Nous pouvons faire la même chose pour utiliser le débogueur fdb mais il ne peut pas actuellement se connecter à une application basée sur HTML... mais oui ce n'est qu'une version alpha :-)

III - Accéder à Apollo depuis javascript

Voici la structure d'une application HTML :



Nous remarquons dans cette structure classique un nouvel objet à travers la propriété runtime de window qui est la liaison entre une application HTML et Apollo. Ainsi, grâce à cet objet runtime, nous pouvons utiliser les objets proposés par Apollo, par exemple en accédant à la classe Capabilities :

```

<html>
<head>
<title>Apollo Runtime</title>
<script>
var capabilities = runtime.flash.system.Capabilities;

function onInit() {
output.innerHTML = "<h1>Capabilities</h1>"
output.innerHTML += "Langage : " + capabilities.language + "<br/>";
output.innerHTML += "PlateForme : " + capabilities.os + "<br/>";
output.innerHTML += "Version : " + capabilities.version + "<br/>";
output.innerHTML += "Larg. Écran : " + capabilities.screenResolutionX + "px<br/>";
output.innerHTML += "Haut. Écran : " + capabilities.screenResolutionY + "px<br/>";
}
  
```

```
</script>
</head>
<body onload="onInit()">
  <div id="output"></div>
</body>
</html>
```

Les possibilités concernant l'utilisation du WebKit dans Apollo sont assez prometteuses, vivement la beta 2 pour pouvoir contrôler la fenêtre d'une application HTML et pleins d'autres méthodes non encore disponible à savoir la possibilité de lancer des programmes externes depuis l'objet Shell avec la méthode `exec()` ... et pleins d'autres choses :-)

IV - Les sources

En attendant voici les codes sources de ce tutorial : **HTML-Based Apollo Application (miroir)**

