,

# Programmation Web Côté Client avec *JavaScript* et *jQuery*

Rémy Malgouyres
LIMOS UMR 6158, IUT, département info
Université Clermont 1
B.P. 86
63172 AUBIERE cedex
http://malgouyres.org/

## Tous mes cours sur le *Web* sont sur le *Web* :

**Cours de programmation *WEB* sur les documents hypertexte *HTML/CSS* :**

http://malgouyres.org/programmation-html-css

**Tutoriel sur le *CMS Drupal* :**

http://malgouyres.org/tutoriel-drupal

**Cours de programmation *WEB* côté serveur en *PHP* :**

http://malgouyres.org/programmation-php

**Cours de programmation *WEB* côté client en *JavaScript* :**

http://malgouyres.org/programmation-javascript

**Cours sur l'administration de serveurs (Serveurs *WEB* avec *apache*, *SSL*, *LDAP*...) :**

http://malgouyres.org/administration-reseau

# Table des matières

# Architectures client/serveur et *API*

## Architecture d'une application multi plate-formes

Une application multi plate-formes contemporaine cherchera à se structurer suivant (voir figure 1) :

1. Une application sur un serveur (*API*) qui traitera les données et assurera la persistance ;

2. Une application sur chaque type de client, qui utilise ce serveur via des requêtes, et gère l'interface (par exemple une *Interface Homme Machine (IHM)*).

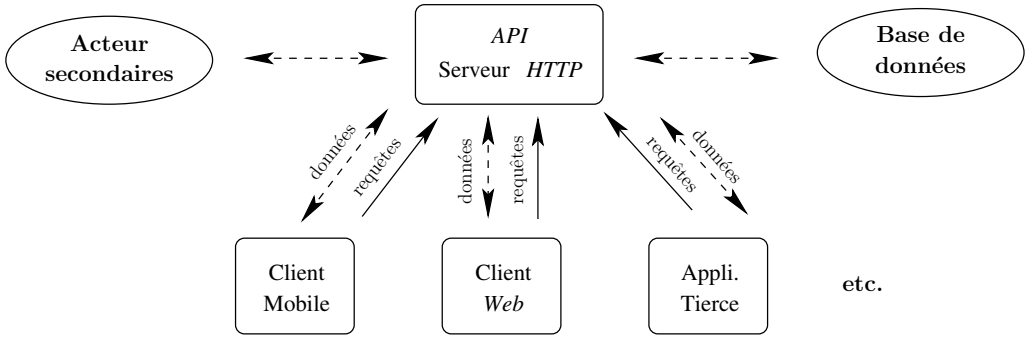


FIGURE 1 : La structure typique d'une application multi plate-formes

On aura dans la mesure du possible intérêt à limiter le plus possible le travail côté client pour les raisons suivantes :

1. L'implémentation côté client dépend de la plate-forme et l'implémentation sur le serveur constitue un forme de *factorisation* du code.

2. Sur certaines plate-formes, comme dans le cas des applications web en *JavaScript*, la sécurité et la confidentialité côté client sont très mauvaises, alors que nous pouvons implémenter la sécurité côté serveur.

Cependant, dans la mesure du possible, les opérations peu sensible, par exemple concernant l'ergonomie, se feront côté client pour limiter les coûts d'infrastructure (nombre de serveurs...) et améliorer la réactivité de l'application.

# Le cas de l'application *Web*

Dans ce cours, nous étudions le développement d'applications *Web* (auxquelles on accède via un navigateur internet), avec une architecture client/serveur dans laquelle (voir la figure 2) :

- Notre *API* est un serveur *HTTP* implémenté en *PHP* avec une architecture *MVC* et *DAL* ;
- Notre application côté client est en *JavaScript* (qui s'est imposé comme un langage standard côté client), et utilise la librairie *jQuery* pour la gestion des événements, des vues, et des interactions (requêtes et échange de données au format *JSON*) avec le serveur.

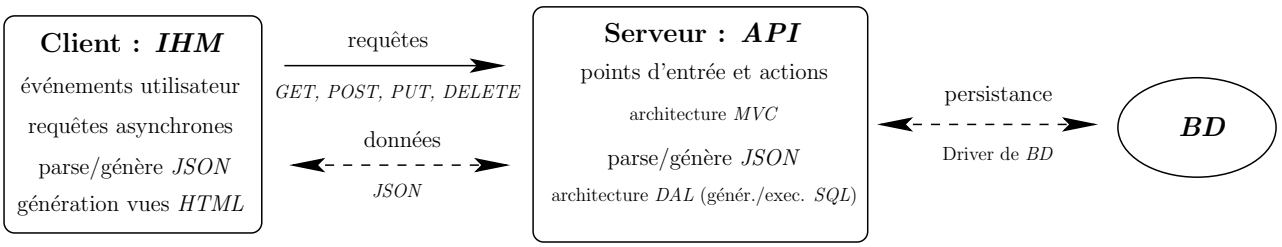


FIGURE 2 : L'architecture *client/serveur* de notre application *Web*

# Chapitre 1

# Premiers pas en *JavaScript*

## 1.1 Balise `<script>` et Hello world en *JavaScript*

Une première manière d'insérer un script *JavaScript* dans un fichier *HTML* est de mettre le code *JavaScript* dans une balise `<script></script>`. On a alors accès au document dans le code *JavaScript* et on peut sortir du code *HTML* :

exemples/bases/ex01_helloWorld.html

```
1  <!doctype HTML>
2  <html lang="fr">
3  <head>
4  <meta charset="UTF-8" />
5  <title>Hello World en Javascript</title>
6  </head>
7  <body>
8  <p>
9  <script>
10   document.write("Hello world !");
11 </script>
12 <p>
13 </body>
14 </html>
```

Une première manière d'insérer un script *JavaScript* dans un fichier *HTML* est de mettre le code *JavaScript* dans un fichier `.js` séparé, qui est inclus dans le *HTML* au niveau du header par une balise `<script src=''...''></script>`.

exemples/bases/ex02_helloWorld.html

```
1  <!doctype HTML>
2  <html lang="fr">
3  <head>
4    <meta charset="UTF-8" />
5    <title>Hello World en Javascript</title>
6    <script src="./ex02_helloWorld.js"></script>
7  </head>
8  <body>
9  </body>
10 </html>
```

**Ceci est un exemple, cliquez sur le lien de téléchargement pour obtenir le cours complet.**