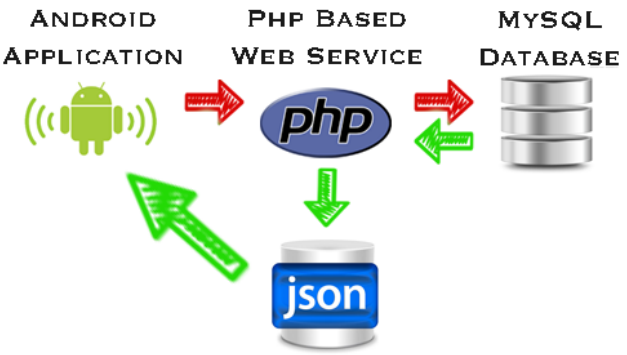
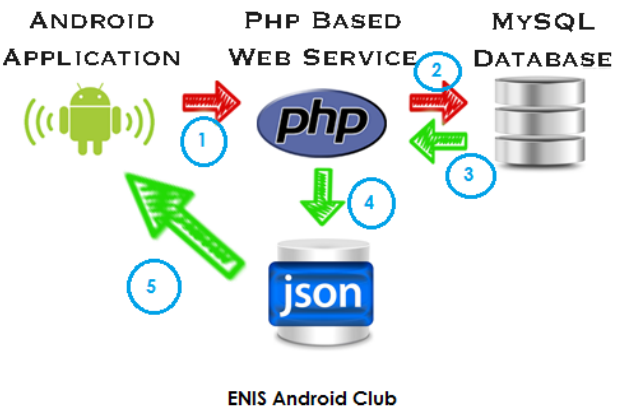



Bienvenue ,voici le nouvel tutoriel , après un retard, qui s'intéresse sur la communication avec des bases de données sous Android pour stocker des données . On a vu la dernière fois l'utilisation du base de données interne d' Android SQLite ( Tuto ).
Cette fois ci , on va voir comment utiliser une base de données externe (MySQL) .
Donc on va commencer par les outils nécessaires afin d'atteindre notre but :

1. Eclipse avec Sdk android (Android Studio)
2. WampServer ou Xampp ( Apache+ PHP + MySQL) ou autres , installé sur Pc.
3. Extension Advanced Rest Client ajoutée à Google Chrome pour tester nos websevices .
4. Patience

## Partie 0 :Avant propos :




Dans cette partie, nous allons expliquer comment vous pouvez intégrer PHP et MYSQL avec votre application Android. Ceci est très utile dans le cas où vous avez un serveur web, et que vous voulez accéder à ses données sur votre application Android.

MySQL est utilisé comme une base de données au serveur web et PHP est utilisé pour extraire des données de la base de données.

Notre application va communiquer avec la page PHP avec des paramètres nécessaires (1) puis PHP communique avec la base de données MySQL et va chercher le résultat (2+3) et enfin retourner les résultats pour nous sous une forme donnée(4+5).

Ici on comprend que notre application Android ne sait communiquer avec notre base de données qu'avec PHP comme intermédiaire .

Donc il faut tout d'abord préparer notre base , ensuite écrire nos scripts PHP qui nous permettre de connecter à la base , extraire les données et les retourner .

## Partie 1 : Préparation de la base de données MySQL :

Après l'installation de nos outils & surtout Wamp (Wampp etc.) , il faut créer notre base .
Comme d'habitude , on va créer une base avec des champs globale comme le tutoriel deSQLite.

**Nom base de données :** BaseExemple

**Nom de table :** TableExemple

| Colonnes | Type | Clé |
|---|---|---|
| Col1 | INTEGER | PRIMARY KEY |
| Col2 | TEXT | |
| Col3 | TEXT | |
| Col4 | TEXT | |

Il suffit d'aller à localhost phpmyadmin : soit on crée la base manuellement (nom: BaseExemple) soit avec un petit code de PHP:


```
CREATE DATABSE BaseExemple;
```

Notre base est maintenant crée. De même façon on crée la table avec ses champs.

```
CREATE TABLE TableExemple (
col1 int(11) primary key auto_increment,
col2 varchar(255) not null, col3 varchar(255) not null, col4 varchar(255) not null);
```

Notre base maintenant est préparée.

## Partie 2 : Préparation des scripts PHP :

Dans cette partie , on va écrire les scripts PHP , qui vont nous permettre de faire plusieurs opérations :

1. Connexion à la base : (connexion.php)
2. Ajouter un élément à la base (ajout_bd.php)
3. Supprimer une donnée de la base (suppression_db.php)
4. Mise à jour d'une donnée de la base (update_db.php)
5. Afficher les données insérées dans la base (affichage_db.php)

Bon jusqu'ici , vous devez avoir comme même des connaissances sur PHP .
Mais n'inquiétez vous , c'est facile .


## 0 : Mon endroit de travail ? :

Avant de commencer à coder , il faut que nous à avons un endroit où on va mettre nos fichiers PHP .
Dans le répertoire où on a installé Wamp (ou Xampp etc. ) , on crée un dossier *"enis_android_club"* dans le dossier *www (pour Wamp)* ou *htdocs* (pour Xampp).


## 1 : Fichier de connexion à la base : Connexion.php :

Pour faciliter les choses, la fonction de connexion à la base , on va la mettre dans une fichier puisqu'on va se connecter à la base à chaque opération .
Donc on a besoin du nom de serveur (localhost ou l'addresse IP de votre serveur),nom de la base de données (BaseExemple) ,le login et le mot de passe de la base (lorsqu vous connectez à phpmyadmin au début ).
On crée un fichier "connexion.php" dans le répertoire *"enis_android_club"* ot on coller cette code :

```
<?php
class CONNEXION_DB {
  function __construct() {
        $this->connection(); // connexion à la base
    }
   function __destruct() {
        $this->fermer(); // fermer la connexion
```

```
  }
 function connection() {
       // connexion à la base , ici : "zied" = mon mot de passe
       $connexion = mysql_connect("localhost", "root", "zied") or die(mysql_error());
        // selection de la base
       $db = mysql_select_db("BaseExemple") or die(mysql_error()) or die(mysql_error());
       return $connexion;
   }
  function fermer() {
      mysql_close(); //Fermer la connexion
   }
}
?>
```

## 2 :Pause :

Avant de commencer à coder les restes de fichiers , il y a des notions qu'on doit les apprendre.

### *A. C'est quoi JSON ?*

**JSON** (*JavaScript Object Notation*) est un format de données textuelles, générique, dérivé de la notation des objets du langage JavaScript. Il permet de représenter de l'information structurée comme le permet XML par exemple. En faite c'est le retour des scripts PHP lorsqu'on les appelles par notre application Android .

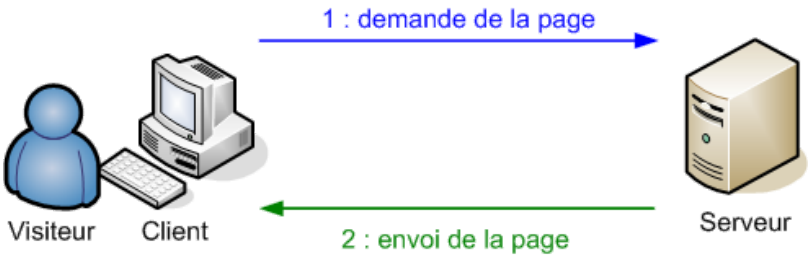
A titre d'exemple et comparaison avec XML (de Wiki) :

```
{
    "menu": {
        "id": "1",
        "value": "File",
        "popup": {
            "menuitem": [
                { "value": "New" },
                { "value": "Open" },
                { "value": "Close" }
            ]
        }
    }
}
```

```
<menu id="1" value="File">
  <popup>
    <menuitem value="New" />
    <menuitem value="Open" />
    <menuitem value="Close" />
  </popup>
</menu>
```

### *B. C'est quoi une requête HTTP ?*

Les flèches représentent les requêtes HTTP:




Il y a plusieurs types de requêtes , on s'intéresse à 2 types:

1. **GET:** C'est la méthode la plus courante pour demander une ressource. Une requête **GET** est sans effet sur la ressource, il doit être possible de répéter la requête sans effet.
2. **POST :** Cette méthode doit être utilisée lorsqu'une requête modifie la ressource

Par exemple , si on va récupérer des données de la base on utilise des requêtes HTTP **GET**, et si on veut insérer des données , on utilise **POST**. Simple non !

### *C. & pourquoi ces deux paragraphes précédents ?*

L'intérêt d'introduire les deux notions , c'est pour vous dire qu'avec notre application Android , on va faire des requêtes HTTP pour communiquer avec notre base MySQL via les fichiers php (**WebServices**) . Le retour sera en format **JSON** (elle est le plus léger pour le transformer en données utilisables par la suite dans notre application : on appelle cette transformation: **Parser** (parsing) )


N'inquiétez-vous !

Un **service web** est un **programme informatique** de la famille des **technologies web** permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribués. Il s'agit donc d'un ensemble de fonctionnalités , pour nous ce sont les fonctions

écrites en php (CRUD). (wiki)

## 3 : Fichier d'ajout d'une ligne de données à la base : ajout_bd.php :

Dans le même endroit ou on a crée le premier fichier , on crée un autre nommé "ajout_bd.php".
Pour ajouter des données à la base , il faut que nous envoyons les données à insérer via le requête HTTP POST (valeurs de col2 , col3 et col4, et concernant col1 c'est ID )
Dans cette fonction , on va établir une connexion à la base. Puis on va tester si on a bien reçu les champs nécessaires ou non , si oui , on insère et un renvoie une réponse sous forme JSON indiquant que l'opération est effectuée avec succès.
Notre réponse sera sous cette forme :

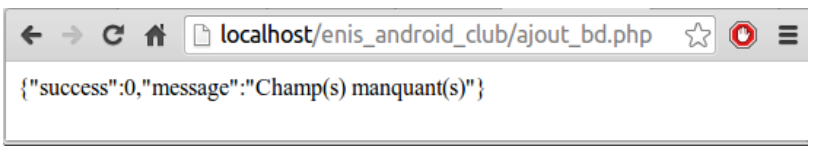
```
{
  "success": 0 / 1
  "message": "message ici ("état d'insertion)"
}
```

Copier / coller ce code dans le fichier :

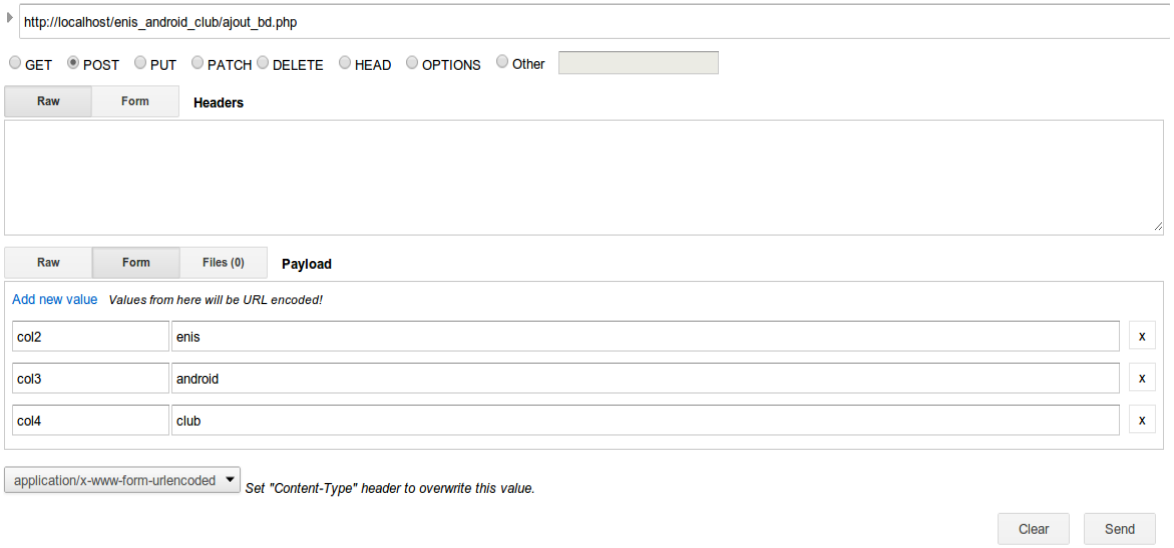
```
<?php
/*
 Requête HTTP Post
 */

// tableau de réponse JSON (array)
$reponse = array();

// tester si les champs sont valides
if (isset($_POST['col2']) && isset($_POST['col3']) && isset($_POST['col4'])) {

    $valeur_col2 = $_POST['col2'];
    $valeur_col3 = $_POST['col3'];
    $valeur_col4 = $_POST['col4'];

    // inclure la classe de connexion
    require_once __DIR__ . '/connexion.php';

    // connxion à la base
    $db = new CONNEXION_DB ();

    // requéte pour insérer les données
```

```
    $resultat = mysql_query("INSERT INTO TableExemple(col2, col3, col4) VALUES('$valeur_col2', '$valeur_col3',
'$valeur_col4')");

    // tester si les données sont bien insérées
    if ($resultat) {
        // Données bien insérées
        $reponse["success"] = 1;
        $reponse["message"] = "Données bien insérées";

       // afficher  la reponse JSON
        echo json_encode($reponse);
    } else {
        // errur d'insertion
        $reponse["success"] = 0;
        $reponse["message"] = "Oops! Erreur d'insrtion.";

      // afficher  la réponse JSON
        echo json_encode($reponse);
    }
} else {
    // Champ(s) manquant(s)
    $reponse["success"] = 0;
    $reponse["message"] = "Champ(s) manquant(s)";

    // afficher  la réponse JSON
    echo json_encode($reponse);
}

?>
```

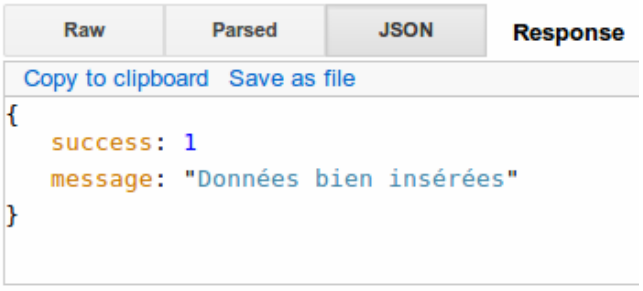
On passe maintenant à tester ce code  via l'extension Advanced REST client sur google chrome:
Si on le teste directement avec le navigateur , voici le résultat:



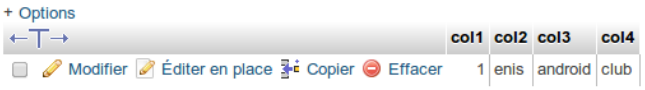

C'est évident puisqu'on a pas envoyé des données avec la requête HTTP.
Avec l'extension : (Type =POST)




Le résultat (après clique sur bouton *send*) :




et voici la vérification avec phpmyadmin:




## 4 : Fichier de suppression d'une ligne de données à la base : suppression_bd.php :

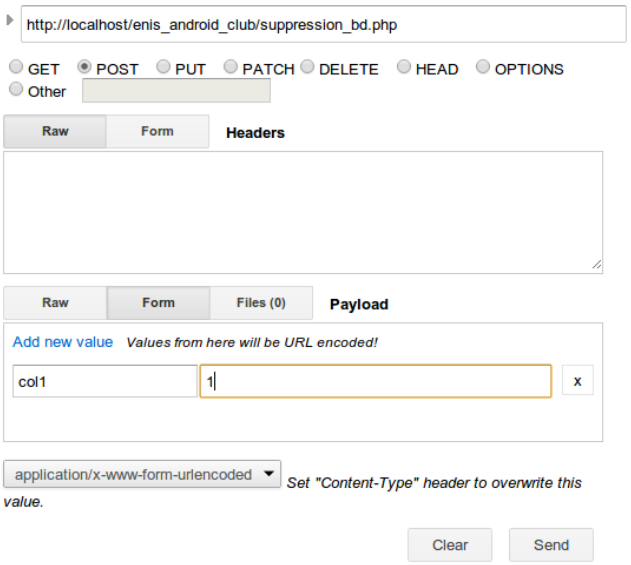
De même pour la fonction d'ajout , on va envoyer l'id de ligne à supprimer (col1) puis on vérifie s'il y a une telle valeur, si oui, on la supprime et envoyer une résultat par la suite.

On crée "`suppression_bd.php`" :

```
<?php

/*
 Requete HTTP Post
 */
// tableau de reponse JSON (array)
$reponse = array();
```

```
// tester s'il y a une donnée récue
if (isset($_POST['col1'])) {
    $valeur_col1 = $_POST['col1'];
    // inclure la classe de connexion
    require_once __DIR__ . '/connexion.php';

    // connexion à la base
    $db = new CONNEXION_DB ();

    // supprimer la ligne
    $resultat = mysql_query("DELETE FROM TableExemple WHERE col1 = $valeur_col1");

    // tester si la ligne est supprimée ou non
    if (mysql_affected_rows() > 0) {
        // ligne supprimée
        $reponse["success"] = 1;
        $reponse["message"] = "ligne supprimée";

        // afficher  la reponse JSON
        echo json_encode($reponse);
    } else {
        // ligne n'existe pas avec col1 =col1(récue)
        $reponse["success"] = 0;
        $reponse["message"] = "Erreur de suppression";

       // afficher  la reponse JSON
        echo json_encode($reponse);
    }
} else {
    // Champ manquant col1
    $reponse["success"] = 0;
    $reponse["message"] = "Champ manquant";

    // afficher  la reponse JSON
    echo json_encode($reponse);
}
?>
```

De même , on teste notre fonction :




## 5 : Les autres fonctions :

De la même façon pour les autres fonctions . Je vous fournit un lien de téléchargement des autres fonctions pour réduire le temps de lire de cette longue article .

Lien de téléchargement

## Partie 3 : Conclusion:

Jusqu'ici on a préparé nos scripts php afin de les utiliser par notre application Android.
L'idée c'est simple : si on veut afficher les données de la base , on fait appel au fichier correspondant , puis on récupère le résultat et la transformer.

Dans le prochain tutoriel , on va développer notre application Android , qui va consommer les web-services .
S'il y a un problème ou une question n'hésitez pas de le poser avec un petit commentaire ou un email à enisandroidclub@gmail.com.

Restez branchés ! A la prochaine 'Samedi'


CLUB
ANDROID
ENIS