IC4 : Réaliser une sonde de surveillance des équipements du Réseau.

- Mazeau Fabiola
- Mourouvin Mickaël

IUT Saint-Pierre, La Réunion

Réseaux & Télécommunications

f.mazeau@rt-iut.re

m.mourouvin@rt-iut.re

08/04/2013

Ce compte-rendu résume ce que nous avons réalisé durant les séances de IC4 :

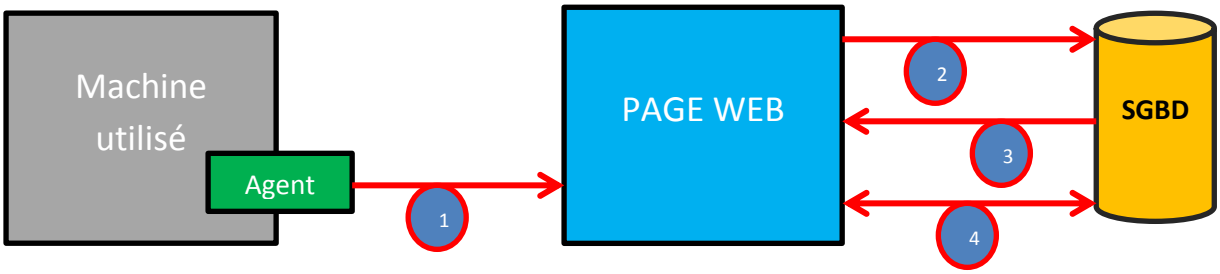
- Une Sonde de récupération de données en java
- Une base de données SQL
- Site WEB : affichage de données récupérées

Tuteurs :

- Mr. Hoareau
- Mr. Pajaniaye

Objectif : Réaliser une Sonde Réseau en langage Java où on récupère des données sur la machine utilisé puis on envoie ses données à une base de données où elle stocke et traite ses données sur page Web de supervision.

Schéma de Principe :



1. L'agent envoie les données récupéré par la sonde JAVA à la page WEB.
2. La page WEB stocke ses données dans la base de données qu'il est connecté.
3. La base de données exécute les traitements qu'on souhaite effectuer (requête) et affiche les résultats.
4. La base et la page web reste connecté : on peut traiter la base de données en temps réel : effacer entrée, sélection d'une ligne particulière…

I) Réalisation de la Sonde Java :

La **réalisation de la sond**e est la première étape, la sonde est **un programme** qui doit récupérer des **informations** de la machine sur laquelle elle tourne. Puis, **on formate les données** récupérées pour les envoyer **au couple page WEB + base de données**.

On souhaite **récupérer** les informations de base **qui définit un PC** par exemple sur **un réseau LAN** : c'est-à-dire son adresse **IP**, son adresse **Mac** et son **interface** connecté.

- **Récupération de l'adresse IP :**

```
// Classe pour Récupérer l'adresse IP de la machine
 public static String getIpAddress(){
     String IpAddress = null;
     try {
         IpAddress = InetAddress.getLocalHost().toString().split("/")[1];
     } catch (final Exception e){
         }
     return IpAddress;
 }
```

Explication : On récupère l'adresse en chaine de caractère (toutes les classes sont d'ailleurs en format de chaine de caractères pour l'envoi des données vers le serveur Web). En utilisant la méthode InetAddress.getLocalHost( ), on récupère le nom de l'ordinateur avec l'adresse IP. Toutefois dans la classe doit récupérer uniquement l'adresse IP, on coupe la chaine de caractère récupérée avec comme séparateur le caractère spécial « / ». On retourne l'adresse IP en type String.

- **Récupération de l'adresse MAC :**

```
// Classe pour récupérer l'adresse Mac
public static String getMACAddress(){
    String MacAddress = null;
    try{
        InetAddress Ip =InetAddress.getLocalHost();
        NetworkInterface ni = NetworkInterface.getByInetAddress(Ip);
        byte [] mac = ni.getHardwareAddress();
        if (mac!=null) {
            // Et si elle existe on la formate afin de la rendre lisible :
                StringBuilder sb = new StringBuilder();
                for (byte b : mac) {
                    sb.append(String.format("%02X:", b));
                }
           //enlève le dernier caractère de la chaine
                MacAddress = sb.substring(0, sb.length()-1);
            }
    }catch(final Exception e){
    }
    return MacAddress;
}
```

Explication : L'**adresse Mac** est le second paramètre qu'on a essayé de récupérer. Le formatage de l'adresse Mac a été une notion difficile (pour nous), en effet on peut **récupérer** l'adresse Mac, mais son **affichage (classique),** nous a forcé à traiter la variable « mac » dans une boucle pour réaliser l'affichage du MAC en **format chaine de caractère**. Lors du formatage, pour chaque bloc (@MAC : 6 octets), on applique « **:** » à la fin pour séparer les octets, on doit donc **enlever** ce **dernier caractère** spécial pour afficher correctement l'adresse MAC.

- **Récupération de l'interface connectée :**

```
//Classe pour Récupérer le nom de l'interface connecté
public static String getIface(){
    String Interface = null;
    try{
        final InetAddress addr = InetAddress.getLocalHost();
        NetworkInterface dest = NetworkInterface.getByInetAddress(addr);
        Interface = ((NetworkInterface) dest).getName();
    } catch(final Exception e){
    }
    return Interface;
}
```

Explication : L'interface récupérée est **le nom de l'interface connecté au réseau.** La méthode **getLocalHost()** retourne l'adresse IP de la machine local. On peut **par** cette information retourner l'interface **lié à** cette adresse

On a ensuite voulu récupérer d'autres **informations supplémentaires** qu'on souhaite par la suite envoyer vers notre base de données.

- **Récupération du nom de l'ordinateur :**

```
// Classe pour Récupérer le nom de l'ordinateur
public static String getComputerFullName() {
    String hostName = null;
    try {
        final InetAddress addr = InetAddress.getLocalHost();
        hostName = new String(addr.getHostName());
        } catch(final Exception e) {
            }
    return hostName;
}
```

Référence: http://forum.hardware.fr/hfr/Programmation/Java/java-recuperer-hostname-sujet_56350_1.htm

- **Récupération de la capacité du disque dur :**

```
//Récupérer la capacité total du disque dur
public static String getTotalspace(){
    String TotalSpace = null;
    try{
          File[] racines = File.listRoots();

          long sum = 0;
          for (int i = 0; i < racines.length; i++){
              sum += racines[i].getTotalSpace()/(1024 * 1024 * 1024);
              TotalSpace = String.valueOf(sum);
          }
    }catch(final Exception e){
    }
    return TotalSpace;
}
```

Explication : C'est la classe la **plus difficile** qu'on a réalisé à notre avis. Car on doit récupérer la capacité totale du disque dur. Mais au départ on récupère uniquement une **partition du disque** dur. Il faut donc **lister tous** les partitions du disque dur et **récupérer** leurs tailles. Ensuite on additionne, chaque partition.

Note : Il y a néanmoins **une faille** qu'on n'a pas pu résoudre, on liste les périphériques de stockage (utilisation de la méthode : listRoots()), donc lors d'**ajout d'une clé USB** ou disque dur externe il **additionne** sa capacité avec la **capacité de stockage** interne de l'ordinateur.

- **Récupération de l'espace libre total du disque dur :**

```
//récupérer l'espace libre du disque dur
public static String getFreespace(){
    String FreeSpace = null;
    try{
          File[] racines = File.listRoots();
          long sum = 0;
          for (int i = 0; i < racines.length; i++){
              sum += racines[i].getFreeSpace()/(1024 * 1024 * 1024);
              FreeSpace = String.valueOf(sum);
          }
    }catch(final Exception e){
    }
    return FreeSpace;
}
```

Explication : Cette classe est **quasi-identique** la différence est l'utilisation de la méthode **getFreeSpace()** qui récupère l'**espace libre** du disque.

- **Récupération de la date d'envoi :**

```
//récupérer la date d'envoi du post
public static String getDate(){
    String Date = null;
    try{
        Date now = Calendar.getInstance().getTime();
        DateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd kk:mm:ss");
        Date = dateFormat.format(now);
    }catch(final Exception e){
    }
    return Date;
}
```

Explication : On récupère **la date d'envoi** du post à la seconde près. Le formatage est réalisé par la méthode **SimpleDateFormat**, on peut regarder dans l'**API** pour choisir de quelle manière on souhaite écrire la date.

Conclusion : Dans cette partie, on a réalisé la **récupération des données** qu'on souhaite envoyer vers la **page de traitement du post**. On voit que toutes les informations récupérés ont été par la suite formatés en chaine de caractères pour l'envoi. C'est pour **respecter les paramètres** de la méthode Post() qu'on va commencer à construire par la suite.

II) Connectivité et envoi de la sonde :

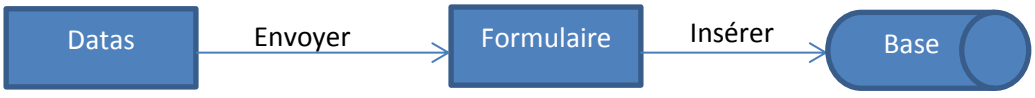
Ici, on va maintenant essayer d'effectuer une **connectivité avec le serveur WEB** qui traitera des données envoyées via à la **méthode POST()** par la sonde.

```
public static String post(String adress,List<String> keys,List<String> values) throws IOException{
    String result = "";
    OutputStreamWriter writer = null;
    BufferedReader reader = null;
    try {
    //encodage des paramètres de la requête en fonction des List<string>
        String data="";
        for(int i=0;i<keys.size();i++){
            if (i!=0) data += "&";
            data +=URLEncoder.encode(keys.get(i), "UTF-8")+"="+URLEncoder.encode(values.get(i), "UTF-8");
            }
    //création de la connection avec le serveur web
        URL url = new URL(adress);
        URLConnection conn = (URLConnection) url.openConnection();
        conn.setDoOutput(true);
    //envoi de la requête POsT
        writer = new OutputStreamWriter(conn.getOutputStream());
        writer.write(data);
        writer.flush();
    //lecture de la réponse
        reader = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        String ligne;
        while ((ligne = reader.readLine()) != null) {
            result+=ligne;
        }
    }catch (Exception e) {
        e.printStackTrace();
    }
    return result;
}
```

La méthode Post() prend en **paramètres l'adresse du serveur web** qui traitera les données, des **noms donnés** au champs de la page PHP qu'on envoie les données, et des **valeurs associées**.

En effet les deux paramètres de la méthode **stockent les informations** qu'on a récupérées. Après la connexion avec la page WEB, les post remplissent les champs post de la page web. On a réalisé simplement une méthode qui **remplissent automatiquement des champs d'un formulaire** c'est pour cela que les informations récupérés sont formatés en chaine de caractères.

Principe :




```
// envoi à l'adresse du serveur
    String url = post("http://192.168.60.138/tot/IC4/hello.php",keys,values);
     System.out.println(url);
```

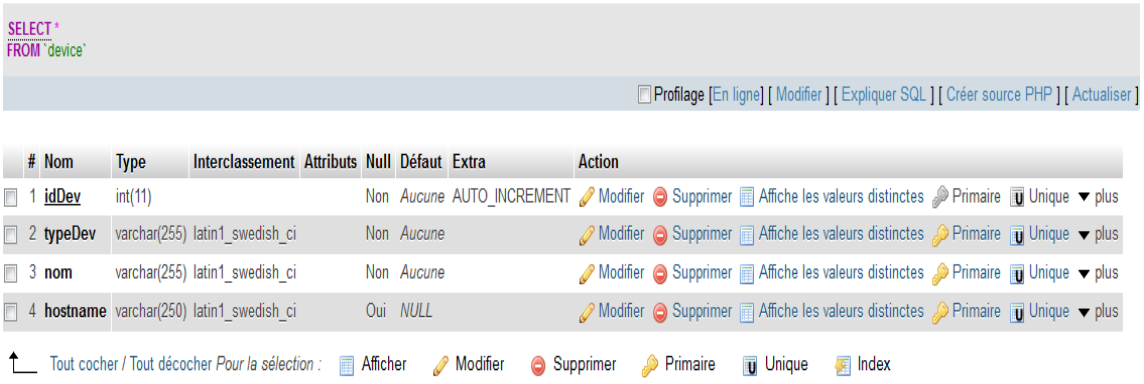
La **variable « url »** stocke l'adresse du « formulaire » du serveur qui va enregistrer les informations. C'est ici qu'il faudra modifier si le serveur change d'adresse.

Conclusion : A ce stade on communique (envoi via **méthode POST**) avec un serveur web, il faut maintenant **réaliser la base de données** en fonction de contraintes citées durant les séances.

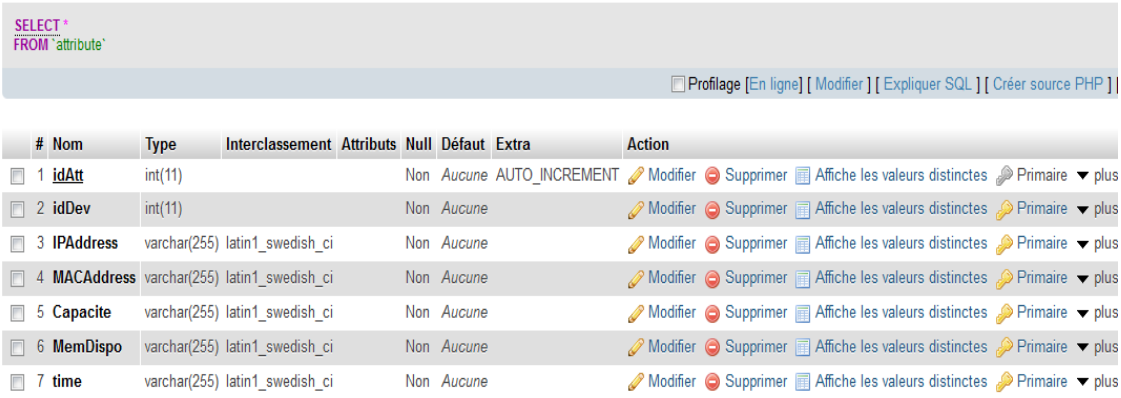
III) Réalisation de la base de données :

Pour la base de données, nous avons choisi une structure simple et efficace. Nous avons décidé de créer notre base de données à l'aide du logiciel PHPMyAdmin.
Nous avons séparé la base de données en deux tables :

- La table « device » qui regroupe les données essentielles sur les machines connectées tel que le nom de cette machine, son type et son nom.



- La table « attribute » qui regroupe des informations complémentaires (les attributs) de cette machine. Ainsi en la consultant, nous pouvons retrouver l'adresse IP de la machine, son adresse MAC, sa capacité et son espace disponible, ainsi que l'heure ou nous avons récolté les informations, c'est-à-dire l'heure à laquelle la machine s'est connectée ou encore l'heure à laquelle la sonde nous a envoyé les informations.



IV) Réalisation de la page Web :

Pour la réalisation de la page Web, nous avons utilisé les outils Notepad++ pour la partie programmation Java et WampSserver/ localhost pour la mise en réseau du programme.
Notre page Web comprend plusieurs onglets :

- **Hello.php** : c'est la page qui se connecte à la base de données :

```php
//Connection à la base de données
$connection = mysql_connect('localhost','root', '') or die('Erreur de connexion '.mysql_error());

$db=mysql_select_db('ic4',$connection) or die('Erreur de selection '.mysql_error());
```

La sonde Java envoie ses données à la page hello.php, sui va ensuite inscrire les données envoyées dans la base de données prévue à cet effet.
Exemple de la table « device » :

```php
//Table device
if (isset($_POST['idDev']) && (isset($_POST['typeDev']))&& (isset($_POST['nom'])) &&(isset($_POST['hostname'])))
    $idDev= $_POST['idDev'];
    $typeDev = $_POST['typeDev'];
    $nom= $_POST['nom'];
    $hostname=$_POST['hostname'];

    //Insertion des données dans la base
    $query="INSERT INTO device VALUES ('$idDev', '$typeDev', '$nom', '$hostname')";

    //Affichage de la requête
    echo "requete =$query </br>";
    $requete = mysql_query($query, $connection) or die( mysql_error() ) ;

    //Affichage des résultats
    if($requete)
        {echo("Insertion réussie") ;}
    else
        {echo("Echec") ;}
}
```

Pour être certain de la bonne insertion dans la base de données, nous demandons dans notre code d'indiquer si l'insertion est réussie ou est un échec.

- **Index.php** : la page index.php correspond à la page d'accueil du site. Elle comprend le nom de la version du moniteur, les onglets de redirections (exemple vers la page contact.php), affiche aussi l'heure en temps réel :

```php
<p class="side-title">Current Hour:</p>
<p><script language="javascript">
        function heure()
        {
        dte = new Date();
        h = dte.getHours();
        m = dte.getMinutes();
        s = dte.getSeconds();

        if(h<10)
        { h = '0'+h; }
        if(m<10)
        { m = '0'+m; }
        if(s<10)
        { s = '0'+s; }

        date = ''+h+':'+m+':'+s+'';
        document.temps.time.value = date;
        }
        window.setInterval("heure()",1000);
```

- **Contact.php** : cette partie du code permet aux utilisateurs d'envoyer un mail à l'administrateur sans avoir besoin de se connecter à sa boite mail, sur le site même il devra renseigner son nom, son mail, l'objet du mail ainsi que le corps du message. Le mail sera directement redirigé vers la boite mail de l'administrateur. Cela est possible grâce à quelques lignes que l'on doit ajouter dans ce fichier :

```
ini_set("SMTP","mx.zeop.re");

$dest = 'f.mazeau@rt-iut.re';
```

Ces lignes indiquent le nom du serveur SMTP ainsi que l'adresse mail à laquelle les mails doivent être envoyés.
De même, nous devons modifier le fichier ini.d du php de WampServer :

```
[mail function]
; For Win32 only.
; http://php.net/smtp
SMTP = smtp.gmail.com
; http://php.net/smtp-port
smtp_port = 25

; For Win32 only.
; http://php.net/sendmail-from
sendmail_from = f.mazeau@rt-iut.re
```

Ainsi on modifie la partie SMTP et on renseigne l'adresse mail de l'administrateur.

- **Tableau.php** : la page tableau.php donne la structure du tableau qui contiendra les informations sur les machines connectées :

```
<h3>Table d'information</h3>
<thead>
    <tr>
        <th>HostName</th>
        <th>Adresse MAC</th>
        <th>Adresse IP</th>
        <th>Interface connectée</th>
        <th>mem.disp</th>
        <th>1st SendTime</th>
        <th>état</th>
    </tr>
```

- **Heur.php** : C'est code écrit en javascript qui permet d'afficher l'heure interne du serveur en temps réel.

Conclusion : Nous avons donc réalisé une structure qui permet de surveiller un réseau via à un agent qui tourne sur les machines qu'on souhaite surveiller à distance. Il est vrai qu'il manque de nombreuses fonctionnalités basiques du monitoring. Toutefois nos réalisations personnels sont les résultantes de ce que nous maitrisons dans le domaine de la programmation.

THE END