# Les technologies Asynchrones

## I) Introduction :

Quand nous effectuons une demande dont le résultat peut se faire attendre, on peut parfois faire autre chose avant d'obtenir le résultat de la demande.

Exemple : j'attends le bus

- soit je reste debout sans rien faire : je suis en mode synchrone.
- Soit j'ouvre mon livre de math et je révise : je suis en mode asynchrone.

  Les E/S sont souvent lentes pour ne pas perdre de temps cette méthode permet à notre programme de travailler en attendant la fin de l'opération d'E/S.

## II) Définition :

Appel d'une fonction :

- Méthode synchrone : la fonction bloque le programme.
- Méthode asynchrone : la fonction se termine rapidement et le processus qui gère la fonction déclenchera un événement pour nous informer quand la lecture sera achevée.

## III) En pratique :

- créer au moins une fonction « callback » qui sera appelé depuis le processus Asynchrone.
- Créer le processus Asynchrone en lui passant la fonction « callback ».
- Lancer le processus Asynchrone.
- Faire autre chose en tenant éventuellement compte de variable modifiée par la fonction « callback ».

## Exemples :

1. lecture asynchrone d'un fichier.

```
// ouverture du flux en lecture
FileStream asfileStream = new FileStream("Asynchrone.exe", FileMode.Open,
FileAccess.Read);
// variables nécessaires à une opération asynchrone
byte[] Buffer = new byte[512];
// fonction delegate : permet d'avoir une référence sur un objet fonction
 AsyncCallback callBack = new AsyncCallback(ProcessRead);
// création d'un objet permettant d'identifier cette opération asynchrone
object state = new object();
// variable de communication
bLue = false;
// appel non bloquant, la fonction callback sera appelé quand la lecture sera complète
IAsyncResult asRes = asfileStream.BeginRead(Buffer, 0, Buffer.Length, callBack, state);

// faire autre chose en attendant que la lecture du fichier soit complète : des calculs par
exemple
      while (!bLue)
            Console.Write("A");

// appel bloquant termine la lecture si celle-ci n'est pas terminée et retourne le nombre
d'octets lus
int nbOctetsLus = asfileStream.EndRead(asRes);
Console.WriteLine(" Nombre d'octets lu du fichier :{0:d4}", nbOctetsLus);

// fermeture du flux
asfileStream.Close();
```

2. Cas d'un appel asynchrone AJAX ( *Asynchronous Javascript And Xml* ):

```
<script type ="text/javascript" language ="javascript" >

host = "localhost";

// fonction callback
function Recu()
{
   if (oReq.readyState == 4)
      document.getElementById( "case4" ).innerHTML = oReq.responseText;
}

// fonction exemple
function MaJPHPAsynchrone( id)
{
 // objet minimum base des techniques AJAX
 // il existe des bibliotheques javascript pour faciliter AJAX
 oReq  = new XMLHttpRequest();
 if (oReq.overrideMimeType)
     oReq.overrideMimeType('text/xml');
 // préparation de la requete au serveur HTTP distant
 oReq.open("GET", "http://"+host+"/CoucouId.php"+ "?Id="+id, true);
 // Donner la fonction qui sera appelée quand le résultat sera arrivé
 oReq.onreadystatechange = Recu;
 // appel de requete asynchrone
 oReq.send();
}

</script>

<p id = "case4" onclick = "MaJPHPAsynchrone( id);" style =" border-
width :medium">
Appel cliquer ici
</p>
```

## IV) Quand utiliser (ou ne pas ) une technologie asynchrone :

1. Utiliser le mode asynchrone à chaque fois qu'un blocage est pénalisant.
2. Attention le mode asynchrone consomme des ressources.
3. Attention : un appel asynchrone fait travailler deux processus en concurrence !

4. Dans le cas de AJAX : il ne faut pas confondre le fait :
   - de modifier le contenu de la page sans la recharger entièrement. Ce qui peut se faire de manière synchrone.
   - et celui de charger du contenu de manière asynchrone.

Autres exemples :

- Requête SQL : les opération de requête peuvent être consommatrice de temps.
- Affichage d'une barre de progression pendant une opération longue.
- Traitement multimédia.
- Temps réel.
- ...