Web

AJAX
JSON
PHP BDD

AJAX

- C'est quoi AJAX ?
- **A**synchronous **JA**vascript **X**ML
- AJAX réutilise des technologies
- HTML + CSS + XML + Javascript DOM

AJAX

- AJAX n'est **pas une** technologie
  - Ensemble d'outils
  - Construire des pages web dynamiques côté client

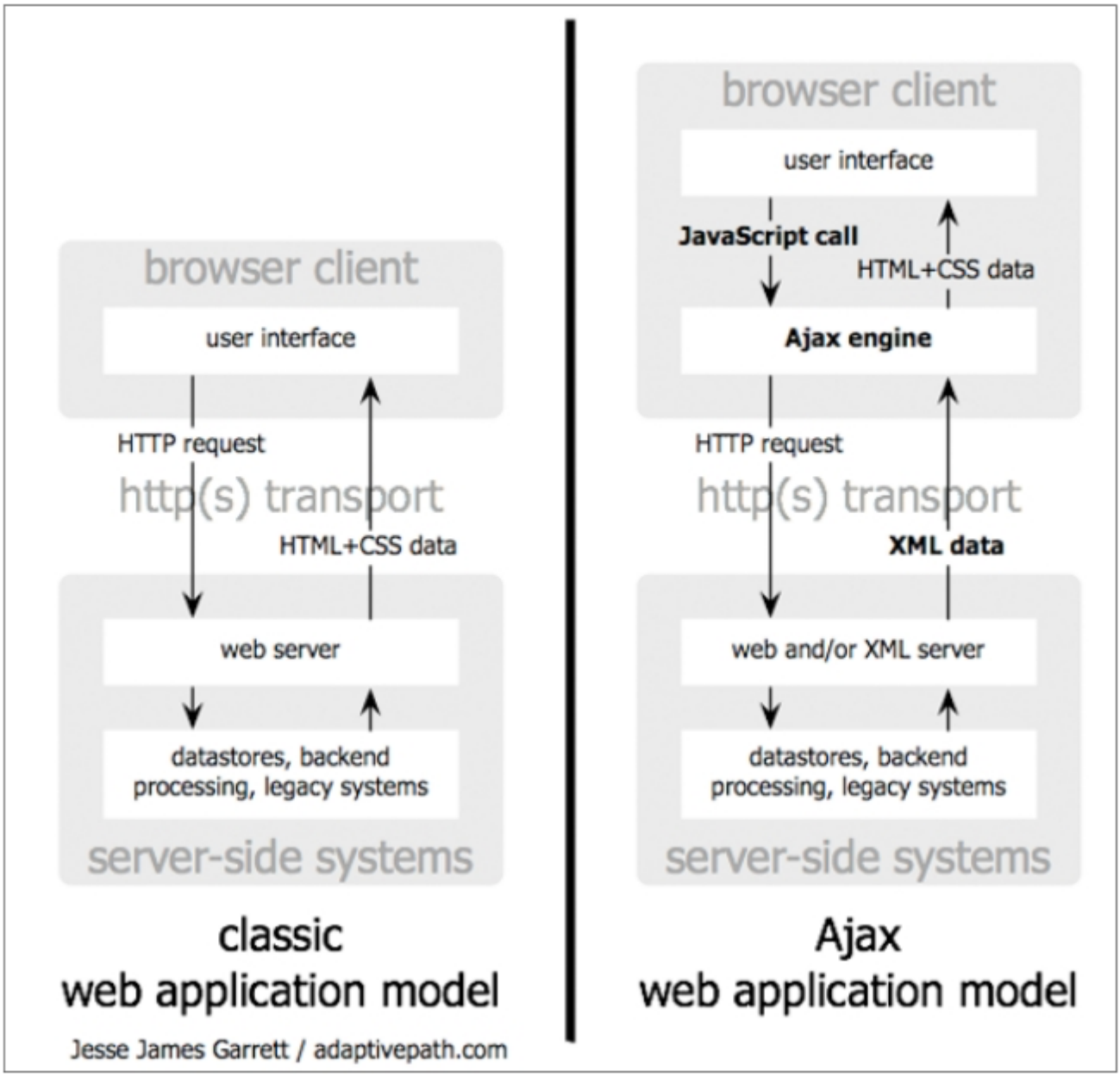
AJAX

- Utilité : modifier partiellement une page
- Quand ? Sur réception de données du serveur
- Pas besoin de recharger complètement la page

AJAX

- La requête se fait en JavaScript
- Lorsque le serveur traite la requête, des données sont échangées
- Données transmises à une fonction Javascript cliente de manière asynchrone

AJAX


browser client
user interface
HTTP request
http(s) transport
HTML+CSS data
web server
datastores, backend processing, legacy systems
server-side systems
classic web application model
Jesse James Garrett / adaptivepath.com
browser client
user interface
JavaScript call
HTML+CSS data
Ajax engine
HTTP request
http(s) transport
XML data
web and/or XML server
datastores, backend processing, legacy systems
server-side systems
Ajax web application model

AJAX

- 2 modes de communication
- **synchrone** : l'exécution de Javascript est "gelée" en attendant la réponse du serveur
- **asynchrone** : l'exécution de Javascript continue sans attendre la réponse du serveur. Cette dernière sera traitée quand elle arrivera

AJAX

- **L'objet XMLHTTPRequest**
- Objet Javascript utilisé permettant de ...
  - lancer une requête sur le serveur
  - attendre et récupérer des données depuis le serveur

Objet XMLHTTPRequest

- Fonctions
  - open : établit la connexion
  - send : envoie une requête au serveur
- Attributs
  - Données texte : responseText
  - Données XML : responseXML
  - readyState : État de traitement de la requête
  - status : code de disponibilité de la page
  - onreadystatechange : fonction activée lors d'un changement d'état
- Création d'un nouvel objet XMLHTTPRequest à chaque connexion au serveur

XMLHTTPRequest - Fonctions

- open(type, url, boolComm)
  - **type** : type de requête GET ou POST
  - **url** : la page script à exécuter
  - **mode** : true (asynchrone) / false (synchrone)
- send("chaîne")
  - **null** pour une requête GET
  - **"p1=val1&p2=val2&..."** : pour POST
- onreadystatechange
  - Fonction exécutée lors d'un changement d'état

XMLHTTPRequest - Attributs

- **.readyState**
  - **0** : non initialisé
  - **1** : connexion établie
  - **2** : requête reçue
  - **3** : réponse en cours
  - **4** : terminé

Création de l'objet XMLHTTPRequest

```
function getXHR() {

  var xhr = null;
  if (window.XMLHttpRequest) // FF & autres
     xhr = new XMLHttpRequest();
  else if (window.ActiveXObject) { // IE < 7
        try {
          xhr = new ActiveXObject("Msxml2.XMLHTTP");
        } catch (e) {
          xhr = new ActiveXObject("Microsoft.XMLHTTP");
        }
  } else { // Objet non supporté par le navigateur
     alert("Votre navigateur ne supporte pas AJAX");
     xhr = false;
  }

  return xhr;

}
```

AJAX – Exemple - GET

```
function execution() {

  var xhr = getXhr();

  // Quoi faire au changement d'état
  xhr.onreadystatechange = function() {
     // Procéder si on a tout reçu
     // et que le serveur est ok
     if (xhr.readyState == 4 && xhr.status == 200){
        // traitement réalisé avec la réponse...
        reponse = xhr.responseText;
        ...
     }
  }
  // cas de la méthode get
  xhr.open("GET","pagecible.php",true) ;
  xhr.setRequestHeader(
    'Content-Type',
    'application/x-www-form-urlencoded ;charset=utf-8'
  );
  xhr.send(null);

}
```

AJAX – Exemple - POST

```
function execution(mavar) {

  var xhr = getXhr();
  xhr.onreadystatechange = function() {

   if (xhr.readyState == 4 && xhr.status == 200) {
        reponse = xhr.responseText;
        ...
     }
  }

  // cas de la méthode post
  xhr.open("POST","pagecible.php", true) ;
  xhr.setRequestHeader(
    'Content-Type',
    'application/x-www-form-urlencoded ;charset=utf-8'
  );

  xhr.send("val1="+mavar);

}
```

AJAX

- Avantages
  - Réactivité car toute la page n'est pas rechargée
  - Bonne compatibilité
- Inconvénients
  - Ergonomie limitée : pas de maintient de l'historique des pages visitées par AJAX
  - Indexation des pages limitée
  - Temps de latence potentiellement pénalisant
  - Javascript obligatoire

JSON

- **J**ava**S**cript **O**bject **N**otation
- Format populaire sur le Web
  - Plus utilisé que .CSV (Comma Separated Values)
- XML est plus lourd mais peut être approprié pour des données structurées et complexes
- Intégrable facilement dans du code Javascript

Format JSON

-
```
{"employes":[
    {"prenom":"Martin", "nom":"Davis"},
    {"prenom":"Anna", "nom":"Duel"},
    {"prenom":"Tim", "nom":"Jones"}
]}
```


- Testez votre JSON sur http://json.parser.online.fr

Équivalent en XML

```
<employes>
    <employe>

      <prenom>Martin</prenom>
                <nom>Davis</nom>
    </employe>
      <employe>

      <prenom>...</prenom>
                <nom>...</nom>
    </employe>
      ...
</employes>
```

Javascript

- var text = '{"nom":"Jesus Cris","rue":"777 Hell","tel":"+1 402 983-5400"}'
- `var obj = JSON.parse("text");`

PHP

- **$json** =

  '{"prenom":"Roger",
  "nom":"Federer",
  "age":33}';

- var_dump(json_decode(**$json**));

- Résultat: object(stdClass)#1 (3) {
  ["prenom"]=> string(5) "Roger"
  ["nom"]=> string(7) "Federer"
  ["age"]=> int(33)
}

PHP - Oracle

- Oracle est un type de base de données (comme MySQL, PostgreSQL, etc.)
- Le module OCI doit être configuré
- Cette extension donne accès à des fonctions de manipulation de bases de données Oracle depuis PHP
- La méthode la mieux supportée par Oracle est OCI

- Références
  - PHP – http://php.net
  - OCI - http://php.net/manual/fr/ref.oci8.php

Processus typique

- Connexion à une base de données
- Création de la requête
- Exécution de la requête
- Récupération (au besoin) des résultats
- Fermeture de la connexion

Fonctions

- oci_connect — Établit une connexion avec un serveur Oracle
- oci_parse — Prépare une requête SQL avec Oracle
- oci_error — Retourne la dernière erreur Oracle
- oci_execute — Exécute une requête
- oci_statement_type – Type de requête
- oci_bind_by_name – Lie (sécuritairement) une variable PHP au marqueur
- oci_define_by_name — Associe une variable PHP à une colonne d'une requête pour un fetch
- trigger_error – Déclenche une erreur personnalisée.
- htmlentities – Encode les caractères en HTML.

Connexion à la BD (Oracle)

```
$conn = oci_connect(
'SYSTEM', 'password', 'localhost/XE');
if (!$conn) {
    $e = oci_error();
    trigger_error(htmlentities($e['message'],
ENT_QUOTES),        E_USER_ERROR);
}
```

Préparation d'une requête

```
$sql = "SELECT * FROM atable";
$stid = oci_parse($conn, $sql);
if (!$stid) {
    $e = oci_error($conn);
    trigger_error(htmlentities($e['message'],
        ENT_QUOTES), E_USER_ERROR);
}
```

Exécution d'une requête

```
$r = oci_execute($stid);
if (!$r) {
    $e = oci_error($stid);

trigger_error(htmlentities($e['message
'],      ENT_QUOTES), E_USER_ERROR);
}
```

Exemple – type de requête

```
$stid = oci_parse($conn, 'DELETE FROM departments WHERE department_id = 13;');
if (oci_statement_type($stid) == "DELETE") {
    trigger_error('Non autorisé', E_USER_ERROR);
}
else {
    oci_execute($stid);  // efface la ligne
}
```

Exemple - Insertion

```
$stid = oci_parse($conn,
"INSERT INTO mytab (id, text)
VALUES(:monid, :montext) );
$id = 1;
$text = "Données à insérer";
oci_bind_by_name($stid, ":monid", $id);
oci_bind_by_name($stid, ":montext", $text);
oci_execute($stid);
```

Exemple - Sélection

Récupérer des valeurs dans des variables PHP.

```
$stid = oci_parse($conn,
 'SELECT * FROM mytab');
oci_define_by_name($stid, 'ID', $id);
oci_define_by_name($stid, 'NOM', $nom);
oci_execute($stid);
while (oci_fetch($stid)) {
    echo $id . " = " . $nom . "<br>\n";
}
```

Exemple – Insertion (Row ID)

```
$sql = "INSERT INTO mytab (id, name)
VALUES(:monid, :monnom)
RETURNING ROWID INTO :rid";
$stid = oci_parse($conn, $sql);
$rowid = oci_new_descriptor
($conn, OCI_D_ROWID);

oci_bind_by_name($stid, ":monid",   $id, 10);
oci_bind_by_name($stid, ":monnom", $name, 3
2); //32: longueur
oci_bind_by_name($stid, ":rid", $rowid, -
1, OCI_B_ROWID);
```

Exemple - Update

```
$sql =
"UPDATE mytab SET salary = :salary
WHERE ROWID = :rid";
$upd_stid = oci_parse($conn, $sql);
oci_bind_by_name
($upd_stid, ":rid", $rowid, -1, OCI_B_ROWID);
oci_bind_by_name
($upd_stid, ":salary", $salary,   32);
```

Fonctions – Résultats de requêtes

- oci_fetch_row — Lit la prochaine ligne d'une requête sous forme de tableau numérique
- oci_fetch_array — Lit une ligne d'un résultat sous forme de tableau associatif ou numérique
- oci_fetch_assoc — Lit une ligne d'un résultat sous forme de tableau associatif
- oci_fetch_object — Lit une ligne d'un résultat sous forme d'objet
- oci_fetch — Lit la prochaine ligne d'un résultat Oracle dans un buffer interne
- oci_fetch_all — Lit plusieurs lignes d'un résultat dans un tableau multi-dimensionnel

Exemple – fetch row/array

```
while (($row = oci_fetch_row($stid)) != false) {
    echo $row[0] . " " . $row[1] . "<br>\n";
}
while (($row = oci_fetch_array($stid)) != false) {
    // Indexation numérique et associative
    echo $row[0] . " et " . $row['DEPARTMENT_ID']
   . " sont identiques<br>";
    echo $row[1] . " et " . $row['DEPARTMENT_NA
ME'] . " sont identiques<br>";
}
```

Exemple – fetch assoc/object

```
while (($row = oci_fetch_assoc($stid)) != false) {
    echo $row['DEPARTMENT_ID'] . " " . $row['DEPART
    MENT_NAME'] . "<br>";
}
while (($row = oci_fetch_object($stid)) != false)
{
    echo $row->ID . "<br>\n";
   echo $row->DESCRIPTION . "<br>\n";
}
```

- Note - oci_fetch_assoc équivaut à oci_fetch_array() avec le mode OCI_ASSOC + OCI_RETURN_NULLS.

Exemple - Récupération de résultats

```
$select_stmt = "select username from user_table";
...
echo "<table border='1'>\n";
while ($row = oci_fetch_assoc($stid)) {
    echo "<tr>\n";
    echo "<td>". $row["USERNAME"] . "</td>\n";
    echo "</tr>\n";
}
echo "</table>\n";
```

Lignes et colonnes

- oci_num_rows — Retourne le nombre de lignes affectées durant la dernière commande
- oci_num_fields — Retourne le nombre de colonnes dans un résultat Oracle
- oci_field_name — Retourne le nom d'un champ Oracle
- oci_field_type — Retourne le type de données d'un champ Oracle
- oci_result($stid, 'COLONNE') — Retourne la valeur d'une colonne dans un résultat

Exemple – Types de colonnes (champs)

```
$ncols = oci_num_fields($stid);
for ($i = 1; $i <= $ncols; $i++) {
    $column_name  =
oci_field_name($stid, $i);
    $column_type  =
oci_field_type($stid, $i);
    echo "<tr>";
    echo "<td>$column_name</td>";
    echo "<td>$column_type</td>";
    echo "</tr>\n";
}
```

Déconnexion

oci_free_statement($stid) — Libération des ressources

oci_close($conn); — Fermeture de la connexion

oci_commit — Confirme l'exécution de la requête
(par défaut, se fait après un oci_execute)

oci_rollback — Annule les transactions Oracle en cours

Rappels SQL

- PHP + OCI permettent d'exécuter n'importe quelle requête SQL
- Rappels sur SQL http://www.w3schools.com/sql/

```
SELECT
Shippers.ShipperName,
COUNT(Orders.OrderID)
 AS NumberOfOrders FROM Orders
LEFT JOIN Shippers
ON Orders.ShipperID=Shippers.ShipperID
GROUP BY ShipperName;
```

Débogage

- var_dump(…); est votre ami:

  ini_set('display_errors', 'On');
  error_reporting(E_ALL);

- $data = array('Ben' => 7,'Alice' => '9');
- echo '<pre>'; var_dump($data); echo '</pre>';

Configuration de OCI

- Préparez votre environnement de travail Oracle.
- À l'aide des instructions d'installation du module OCI.
- Jetbrains PhpStorm