DAR - Cours 03
Technologies Serveur

Romain Demangeon

APR, LIP6, UPMC

02/10/2017

UPMC SORBONNE UNIVERSITÉS

Retour sur les approches

Approche services (SOAP) vs. approche ressources (REST).

- Esprit:
  - une bibliotheque de méthodes vs. un catalogue de données.
- Avantages:
  - sécurisé, formel, robuste vs. simple, pratique, léger.
- Utilisation:
  - limité, formel, entreprises, finance vs. vaste, ouvert, web, mobile.

Projet DAR 2017

Ce qui doit être fait:

- Choix du sujet de projet. Validation.
- Cas d'utilisation: portée, fonctionnalités, utilisation.
- Choix d'une ou plusieurs API.

A faire cette semaine:

- Choix d'un hébergement.
- Découpage du projet.
- Structure de la base de données.
- Installation de conteneur de servlet.
- Installation d'une base de données.
- Configuration d'un ORM.
- Rédaction de code serveur (agilité).

UPMC SORBONNE UNIVERSITÉS

Plan

1. Serveurs Web
   - CGI,
   - Servlets Java,
   - JSP.
2. Persistance (Bases de données)
   - Relationnelles,
   - NoSQL.

UPMC SORBONNE UNIVERSITÉS

Serveur Web HTTP

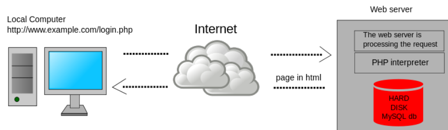
Définition

Un serveur web (logiciel) est une suite de programmes informatique qui stockent, fabriquent et délivrent des pages webs à des clients en suivant le protocole HTTP.

- Gérer des requêtes HTTP.
- Générer des réponses HTTP.
- Majoritairement sur le Web, mais pas seulement.

Fonctionnalités modernes:

- Hébergement virtuel: un serveur pour plusieurs sites,
- Gestion des gros fichiers: taille supérieur à 2Gb,
- Limitation de bande passante: pour ne pas saturer le réseau,
- Langage Serveur: génération dynamique de page web.



UPMC SORBONNE UNIVERSITÉS

Traduction de chemin / Parts de Marché

- Requête saisie dans le navigateur:

  `http://www.saucisse.com/chemin/fichier.html`

- Requête envoyée par le navigateur:

  ```
  GET /chemin/fichier.html HTTP/1.1
  Host: www.saucisse.com
  ```

- Traduction du serveur web:

  `/home/www/chemin/fichier.html`

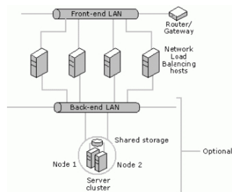
Parts de marché - Sites actifs - (Septembre 2015)

1. Apache (Apache): 50% (↓)
2. nginx (NGINX Inc.): 14% (↑)
3. IIS (Microsoft): 10% (↓)
4. GWS (Google): 8% (–)

UPMC SORBONNE UNIVERSITÉS

Concurrence

- Le modèle Clients/Serveur implique des connexions simultanées.
- Le serveur HTTP gère les requêtes séparément. La concurrence est gérée au niveau de la persistance.
- La base de données gère le partage d'information entre requêtes et assure la cohérence à travers le modèle transactionnel.
- Le serveur web a donc une limite de charge en nombre de connexions et en requêtes par seconde.
  - échecs quand la charge maximale est atteinte,
  - plusieurs moyens pour résoudre le problème (pare-feu, caches, noms de domaines, *load balancer*)



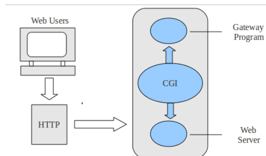
Common Gateway Interface (CGI)

Principe: Méthode historique pour déléguer la génération dynamique de contenu web à des fichiers exécutables.

- Au lieu de retourner le contenu d'un fichier (html), le serveur exécute un programme et retourne le résultat.

CGI: standard industriel qui explicite comment transmettre la requête (arguments) au serveur et récupérer la réponse générée.

- Introduit en 1993 (www-talk), formalisé en 1997 (RFC 3875).
- Les serveurs web (comme Apache) supportent des scripts CGI en plusieurs langages différents:
  - C, C++, Perl, Python, Java, ...
- Exemple: recherche sur un site web (argument: chaîne).



CGI: Page fixe générée dynamiquement

```
#!/soft/python-2.7/bin/python

import cgi
import cgitb
cgitb.enable()  # for troubleshooting

#print header
print "Content-type: text/html"
print
print "<?xml version=\"1.0\" encoding=\"UTF-8\"?>"
print "<!DOCTYPE html>"
print "<html>"
print "<head>"
print "<title>Python CGI test</title>"
print "</head>"
print "<body>"
print "<p>Hello, world!</p>"
print "</body>"
print "</html>"
```

- la page générée est écrite directement sur la sortie standard.

UPMC SORBONNE UNIVERSITÉS

CGI: Page fixe générée dynamiquement

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";
print "<font size=+1>Environment</font>\n";
foreach (sort keys %ENV)
{
  print "<b>$_</b>: $ENV{$_}<br>\n";
}

1;
```

- récupère les variables d'environnements .

```
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,
*/*;q=0.8"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.7"
HTTP_ACCEPT_ENCODING="gzip, deflate"
HTTP_ACCEPT_LANGUAGE="en-us,en;q=0.5"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="example.com"
HTTP_USER_AGENT="Mozilla/5.0 (Windows NT 6.1; WOW64; rv:5.0)
Gecko/20100101 Firefox/5.0"
QUERY_STRING="var1=value1&var2=with%20percent%20encoding"
REMOTE_ADDR="127.0.0.1"
REMOTE_PORT="63555"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv.pl/foo/bar?var1=value1&var2=with%
20percent%20encoding"
...
```

CGI: Passer des arguments en utilisant GET

```
#!/usr/bin/perl

    local ($buffer, @pairs, $pair, $name, $value, %FORM);
    # Read in text
    $ENV{'REQUEST_METHOD'} =~ tr/a-z/A-Z/;
    if ($ENV{'REQUEST_METHOD'} eq "GET")
    {
$buffer = $ENV{'QUERY_STRING'};
    }
    # Split information into name/value pairs
    @pairs = split(/&/, $buffer);
    foreach $pair (@pairs)
    {
($name, $value) = split(/=/, $pair);
$value =~ tr/+/ /;
$value =~ s/%(..)/pack("C", hex($1))/eg;
$FORM{$name} = $value;
    }
    $first_name = $FORM{first_name};
    $last_name  = $FORM{last_name};

print "Content-type:text/html\r\n\r\n";
print "<html>";
print "<head>";
print "<title>Bonjour</title>";
print "</head>";
print "<body>";
print "<h2>Bonjour $first_name $last_name </h2>";
print "</body>";
print "</html>";1;
```

requête GET sur:

`http://www.saucisse.com/hello_get.cgi?first_name=Annie&last_name=Cordy`

Limites et alternatives

- Originellement, un processus est créé sur le serveur pour chaque requête.
  - les scripts doivent parfois être interprétés/compilés
  - les variables d'environnement sont recréées,
  - surcharge sur serveur.
- Pas d'état sur le serveur (doit être dans les requêtes ou la BD).
- Scripts dépendants de la plateforme.
- Solutions à ce problème:
  - FastCGI (1996) et SimpleCGI gardent le modèle, mais réduisent le nombre de processus créés.
  - un modèle de plus haut-niveau comme les Servlets Java

UPMC SORBONNE UNIVERSITÉS

Servlets Java

UPMC SORBONNE UNIVERSITÉS

Servlets

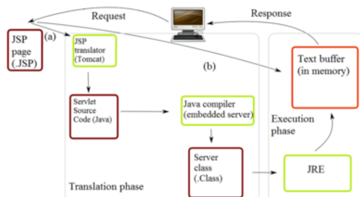
Définition

Un Servlet est une classe Java permettant de créer dynamiquement du contenu Web au sein d'un serveur HTTP.

- Permet d'étendre les fonctions d'un serveur (accès à une BD, transactions). Utilisations principales:
  - traiter ou stocker des données soumises sous forme HTML,
  - proposer du contenu dynamique (*e.g.* résultats de *query*),
  - gérer l'état d'une session (*e.g.* panier d'achat).
- Les Servlets sont chargés (démarrage du serveur, à la première requête) puis restent actifs en attendant d'autres requêtes.
- Créés par Sun Microsystems en 1997.
- Version actuelle: 3.1 (mai 2013)

Conteneur de Servlets

- Le conteneur est le composant su serveur web qui interagit avec les servlets.
- Il est responsable
  - du cycle de vie des servlets
  - de relier les URL aux servlets,
  - de s'assurer que l'utilisateur a les bons droits d'acces.
- Les interactions entre les servlets et le conteneur sont décrites dans la Servlet API (`javax.servlet`)



Cycle de vie d'un servlet

1. un utilisateur saisit une requête pour visiter une certaine URL.
   1.2 le navigateur génère une requête HTTP.
   1.3 le navigateur envoie la requête HTTP au serveur.
2. la requête HTTP est reçue par le serveur web et transférée au conteneur.
   2.2 le conteneur lie la requête HTTP au servlet adéquat.
   2.3 le conteneur récupère le servlet et le charge dans son espace d'adresse.
3. le conteneur invoque la méthode `init()` du servlet.
   - uniquement quand il est chargé pour la 1ère fois.
   - on peut passer des arguments pour configurer le servlet.
4. le conteneur invoque la méthode `service()` du servlet.
   - utilisée pour traiter la requête.
   - le servlet peut accéder aux données fournie dans la requête HTTP.

   4.1 le servlet peut générer une réponse HTTP.
5. le servlet reste disponible dans le conteneur pour traiter d'autres requêtes.
   - `service()` utilisée à chaque fois.
6. le conteneur peut décider de décharger le servlet de sa mémoire.
   - les algorithmes de décision sont spécifiques au conteneur.
7. le conteneur appelle la méthode `destroy` du servlet.
   - des données peuvent être sauvegardées dans la BD.
8. la mémoire allouée au servlet (et ses objets) est disponible pour le ramase-miettes.

UPMC SORBONNE UNIVERSITÉS

Interface `HttpServlet`

```
public abstract class HttpServlet extends
GenericServlet {
  public HttpServlet();

  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException;

  protected void service(HttpServletRequest req, HttpServletResponse resp)
    throws ServletException, IOException;
}
```

- des méthodes similaires pour `doPost`, `doPut`, `doDelete`, `doOptions`, `doTrace`.
- `service` transfere la requête à la méthode `do` correspondante.

Interfaces requêtes et réponses du Servlet

```
public interface HttpServletRequest extends ServletRequest {
  public Cookie[] getCookies();
  public String getHeader(String name);
  public String getParameter(String name);
  public BufferedReader getReader() throws IOException;
...
}

public interface HttpServletResponse extends ServletResponse {
  public void addCookie(Cookie cookie);
  public String encodeURL(String url);
  public void sendError(int sc, String msg) throws IOException;
  public void sendRedirect(String location) throws IOException;
  public void setHeader(String name, String value);
  public void setStatus(int sc);
  public void setContentType(String type);
  public ServletOutputStream getOutputStream()
    throws IOException;
  public PrintWriter getWriter() throws IOException;
...
}
```

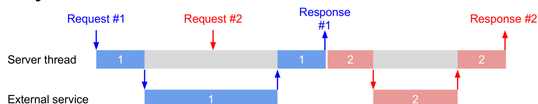
Exemple de Servlet

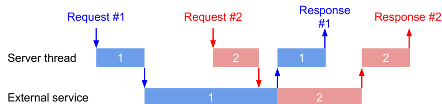
```
public class ExampServlet extends HttpServlet {
  public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      out.println("<title>Example</title><body>");
      String DATA = request.getParameter("DATA");
      if(DATA != null){
        out.println(DATA);
      } else {
        out.println("No text entered.");
      }
      out.println("<p>Return to <a href='index.html'>home</a>");
      out.close();
    }
  }
```

- utilise la méthode `getParameter` de la requête.
- utilise `getWriter` de la réponse pour écrire le contenu.

Asynchronie

- dans les Servlets standards, un thread serveur est créé par requête client.
- il faut s'assurer qu'aucun thread serveur n'attend trop longtemps,
  - soit en utilisant un service externe (BD, connexion), soit en attendant un évènement client,
  - sinon on risque de surcharger le serveur (e.g. limite de pool de threads).
- les Servlets Java Asynchrones permettent de libérer le thread dans l'attente d'une opération externe:
  - c'est elle qui devra renvoyer la réponse.





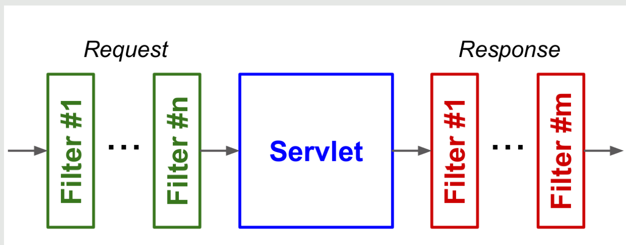
UPMC SORBONNE UNIVERSITÉS

Filtres

- un filtre est un morceau de code réutilisable qui modifie ou adapte les requêtes et les réponses d'un Servlet.

Exemples

- Authentification/Blocage basé sur les identifiants de l'utilisateur,
- Suivi (*tracking*) des utilisateurs,
- Conversion/Redimensionnement d'images,
- Compression de données, . . .



Exemple: un compteur

```
public final class HitCounterFilter implements Filter {
   private FilterConfig filterConfig = null;
   public void init(FilterConfig filterConfig)
      throws ServletException {
      this.filterConfig = filterConfig;}
   public void destroy() {
      this.filterConfig = null;}
   public void doFilter(ServletRequest request,
      ServletResponse response, FilterChain chain)
      throws IOException, ServletException {
      if (filterConfig == null)
         return;
      StringWriter sw = new StringWriter();
      PrintWriter writer = new PrintWriter(sw);
      Counter counter = (Counter)filterConfig.getServletContext().
         getAttribute("hitCounter");
      writer.println("Nombre d'utilisation: " +counter.incCounter());
      writer.flush();
      filterConfig.getServletContext().
         log(sw.getBuffer().toString());
      ...
      chain.doFilter(request, response);
      ...}}
```

Exemple: changer l'encodage d'une requête

```
public void doFilter(ServletRequest request,
   ServletResponse response, FilterChain chain) throws
   IOException, ServletException {
   String encoding = selectEncoding(request);
   if (encoding != null)
      request.setCharacterEncoding(encoding);
   chain.doFilter(request, response);
}
public void init(FilterConfig filterConfig) throws
   ServletException {
   this.filterConfig = filterConfig;
   this.encoding = filterConfig.getInitParameter("encoding");
}
protected String selectEncoding(ServletRequest request) {
   return (this.encoding);
}
```

Sessions

Principes

Une Session permet de stocker des informations sur une suite de requêtes du même utilisateur sur le serveur.

- le protocole HTTP est sans état par nature.

Une session peut être maintenue par un serveur:

- à l'aide d'un *cookie*: information envoyée par le serveur lors d'une réponse et retournée par le navigateur à la requête suivante.
- en réécrivant l'URL (en ajoutant un identifiant de session à la fin de chaque URL)

HTTPSession

- la classe Java `HttpSession` propose une interface de haut-niveau pour la gestion des sessions, construites sur la réécriture d'URL et les *cookies*.
  - utilisation de `request.getSession(true)`.

Exemple de session

```
public class SessionCount extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
    HttpSession session = request.getSession(true);
    response.setContentType("text/text");
    PrintWriter out = response.getWriter();
    Integer count = new Integer(0);;
    if (session.isNew()) {
      out.println("Bienvenue");
    } else {
      out.println("Rebonjour");
      Integer previousCount =
        (Integer) session.getValue("count");
      if (oldAccessCount != null) {
        count = new Integer(previousCount.intValue() + 1);
      }
    }
    session.putValue("count", count);
    out.println("Compteur: " + count.toString());
  }
```

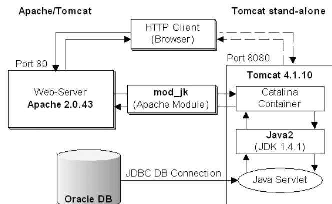
- interface de haut-niveau: utilisation des méthodes de `session`

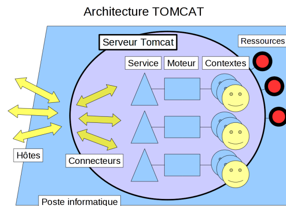
UPMC SORBONNE UNIVERSITÉS

Apache Tomcat

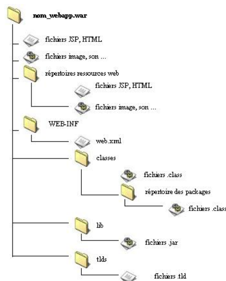
- Serveur Web libre (Apache) et Conteneur de Servlets (Tomcat), implémentant les Servlets et le JSP.

- Principaux composants:
  - Catalina: Conteneur Servlet,
  - Coyote: Connecteur HTTP,
  - Jasper: Moteur JSP,
  - Cluster: *load balancer*.
- Version 3 1999, Version 8 2014.





Application Web (Approche Servlet)



- Une application web (dans le contexte Servlet) est un fichier archive `.WAR` (essentiellement .JAR) contenant des Servlets (classes Java) et leurs ressources associées servie par un Conteneur de Servlet.
- la définition de l'application web est contenue dans un fichier web.xml

Exemple de web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
 <!DOCTYPE web-app
     PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.2//EN"
     "http://java.sun.com/j2ee/dtds/web-app_2_2.dtd">

 <web-app>
     <servlet>
         <servlet-name>HelloServlet</servlet-name>
         <servlet-class>mypackage.HelloServlet</servlet-class>
     </servlet>

     <servlet-mapping>
         <servlet-name>HelloServlet</servlet-name>
         <url-pattern>/Appli/Bonjour</url-pattern>
     </servlet-mapping>

     <resource-ref>
         <description>
             Resource reference to a factory for javax.mail.Session
             instances that may be used for sending electronic mail messages,
             preconfigured to connect to the appropriate SMTP server.
         </description>
         <res-ref-name>mail/Session</res-ref-name>
         <res-type>javax.mail.Session</res-type>
         <res-auth>Container</res-auth>
     </resource-ref>
 </web-app>
```

- Application (Servlet) qui requiert une ressource annexe (pour gérer les sessions mail).

JavaServer Pages (JSP)

Une technologie permettant aux développeurs webs de créer dynamiquement des pages webs en HTML, XML ou autres.

- similaire à PHP (langage de programmation), mais en utilisant les Servlets et Conteneurs de Servlets.
- les JSP sont converties en Servlets à l'exécution.
  - les JSP requièrent donc un Conteneur de Servlets.

```
<%@page contentType="text/html"%>
<%@page errorPage="erreur.jsp"%>
<%-- Importation d'un paquetage (package) --%>
<%@page import="java.util.*"%>
<html>
 <head><title>Page JSP</title></head>
 <body>
  <%-- Déclaration d'une variable globale à la classe --%>
  <%! int nombreVisites = 0; %>
  <%-- Définition de code Java --%>
  <% Date date = new Date();
     nombreVisites++; %>
  <h1>Exemple de page JSP</h1>
  <%-- Impression de variables --%>
  <p>Au moment de l'exécution de ce script, nous sommes le <%= date %>.</p>
  <p>Cette page a été affichée <%= nombreVisites %> fois !</p>
 </body>
</html>
```

Exemple: Servlet généré par le code JSP

```
package org.apache.jsp;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.jsp.*;
import org.apache.jasper.runtime.*;
import java.util.*;
 public class example_jsp extends HttpJspBase {

 int nombreVisites = 0;
  private static java.util.Vector _jspx_includes;

  public java.util.List getIncludes() {
    return _jspx_includes;
  }

  public void _jspService(HttpServletRequest request,
                          HttpServletResponse response)
        throws java.io.IOException, ServletException {
    JspFactory _jspxFactory = null;
    javax.servlet.jsp.PageContext pageContext = null;
    HttpSession session = null;
    ServletContext application = null;
    ServletConfig config = null;
    JspWriter out = null;
    Object page = this;
    JspWriter _jspx_out = null;
    try {
      _jspxFactory = JspFactory.getDefaultFactory();
      response.setContentType("text/html;
                                 charset=ISO-8859-1");
      pageContext = _jspxFactory.getPageContext(this,
request, response, "erreur.jsp", true, 8192, true);
```

```
  application = pageContext.getServletContext();
  config = pageContext.getServletConfig();
  session = pageContext.getSession();
  out = pageContext.getOut();
  _jspx_out = out;
  out.write("<body>\n\n");
  out.write("\n");
  out.write("\n\n");
  out.write("\n");
Date date = new Date();
nombreVisites++;
  out.write("\n");
  out.write("<h1>Exemple de page JSP");
  out.write("</h1>\n");
  out.write("\n");
  out.write("<p>Au moment de l'exécution de
      ce script, nous sommes le ");
  out.print( date );
  out.write(".");
  out.write("</p>\n");
  out.write("<p>Cette page a été affichée ");
  out.print( nombreVisites );
  out.write(" fois !");
  out.write("</p>\n");
  out.write("</body>\n");
  out.write("</html>\n");
} catch (Throwable t) {
  out = _jspx_out;
  if (out != null && out.getBufferSize() != 0)
    out.clearBuffer();
  if (pageContext != null) pageContext.
    handlePageException(t);  } finally {
  if (_jspxFactory != null) _jspxFactory.
    releasePageContext(pageContext);}}}
```

UPMC SORBONNE UNIVERSITÉS

Tags dans JSP

JSP permet aux développeurs d'ajouter leurs propres étiquettes (*tags*) qui exécutent des actions spécifiques en utilisant la *JSP tag extension API*.

- Une classe Java écrite par les développeurs qui implémente l'interface `Tag` et propose une description XML de la bibliothèque tag (*Tag Library Descriptors*).
- La description spécifie les étiquettes et les classes java utilisées pour les implémenter.

Code JSP:

```
<%@ taglib uri="/WEB-INF/taglib.tld"
      prefix="mytaglib" %>
<mytaglib:hello name="Bob">
  You're welcome :)
</mytaglib:hello>
```

Code TLD:

```
<tag>
  <name>hello</name>
  <tagclass>HelloTag</tagclass>
  <bodycontent>JSP</bodycontent>
  <attribute>
    <name>name</name>
  </attribute>
</tag>
```

Code Java:

```
public class HelloTag extends TagSupport {
  private String name = null;
  public void setName (String string) {
    this.name = string;
  }

  public int doStartTag() throws JspException {
    pageContext.getOut().println(
      "Hello " + this.name + " !");
   return EVAL_BODY_INCLUDE;
  }
}
```

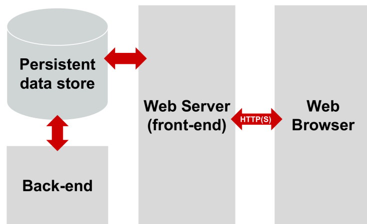
Le retour `EVAL_BODY_INCLUDE` écrit le corps du tag dans le JSP.

UPMC SORBONNE UNIVERSITÉS

Persistance: Base de Données Relationnelles

La Persistance

- La plupart des applications webs ont besoin de stocker des informations entre les sessions.
- Les informations doivent pouvoir être récupérées par différents clients.
  - donc être stockées sur le serveur.
- la persistance (ou base de données) est:
  - la mémoire de l'application web,
  - son point de synchronisation principal.



Modèle relationnel

Une persistance relationnelle est un ensemble de tables comportant des colonnes fixes (les champs) et un nombre arbitraire de lignes (les entrées).

Hypothetical Relational Database Model

PubID	Publisher	PubAddress
03-4472822	Random House	123 4th Street, New York
04-7733903	Wiley and Sons	45 Lincoln Blvd, Chicago
03-4859223	O'Reilly Press	77 Boston Ave, Cambridge
03-3920886	City Lights Books	99 Market, San Francisco

AuthorID	AuthorName	AuthorBDay
345-28-2938	Haile Selassie	14-Aug-92
392-48-9965	Joe Blow	14-Mar-15
454-22-4012	Sally Hemmings	12-Sept-70
663-59-1254	Hannah Arendt	12-Mar-06

ISBN	AuthorID	PubID	Date	Title
1-34532-482-1	345-28-2938	03-4472822	1990	Cold Fusion for Dummies
1-38482-995-1	392-48-9965	04-7733903	1985	Macrame and Straw Tying
2-35921-499-4	454-22-4012	03-4859223	1952	Fluid Dynamics of Aquaducts
1-38278-293-4	663-59-1254	03-3920886	1967	Beads, Baskets & Revolution

Composants des Persistances Relationnelles

- Tables (tableaux à double entrée)
- Clefs primaires: champs identifiant uniquement une entrée.
- Clefs étrangères: identifie une colonne d'une table comme référençant une colonne de clef primaires d'une autre table.
- Index: système permettant de retrouver les données.
- Un langage de requêtes: SQL
- Transactions: opération changeant l'état de la base de données de manière atomique, cohérente, isolée et durable.

- Parts de marché de systèmes de gestion de base de données relationnelles (RDBMS):
  - Oracle Database (Oracle Corp): 70%
  - Microsoft SQL Server (Microsoft): 68%
  - MySQL (Oracle Corp): 50%
  - IBM DB2 (IBM): 39%

UPMC SORBONNE UNIVERSITÉS

Mapping Objet-Relationnel

Le mapping objet-relationnel est une technique de programmation qui crée l'illusion d'une base de données objet à partir d'une base de données relationnelle en définissant des correspondances entre les objets d'un langage et les entrées de la base de données.

Correspondance 1 Entrée = 1 Objet

Table Persons

int id	string name	date birthdate
42	Bob	2013-01-01

```
class Person {
  int id;
  String name;
  Date birthdate;
}
```

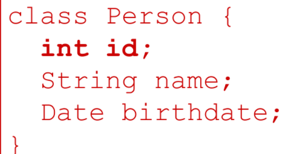
Problemes courants:

- anomalies de typage entre langages de programmations,
- retrouver l'entrée depuis l'objet et vice-versa,
- synchroniser l'entrée et l'objet,
- représenter les collections, les sous-objets,
- représenter l'héritage.

Lier Objets et Entrées

Table Persons

int id	string name	date birthdate
42	Bob	2013-01-01



- Utilisation des clefs primaire pour les lier.
- Synchronisation avec des méthodes `set`/`get`.

UPMC SORBONNE UNIVERSITÉS

Représenter les collections

```
class Album {
  String title;
  Collection<Track> tracks;
}
```

Table Album

ind id	string title
42	Combat Rock
43	Tata Yoyo
44	La Bonne du Curé

```
class Track {
  String title;
}
```

Table Track

ind id	string title	int album
101	Rock the Casbah	42
102	Tata Yoyo	43
103	Queen of the Disco	43

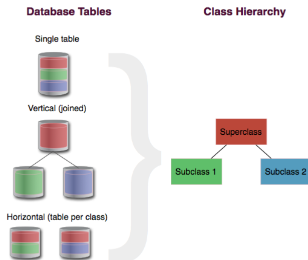
- Utilisation d'une colonne supplémentaire.
- Création d'une table supplémentaire pour la collection.

Table Album2Track

ind id	string title
42	101
43	102
43	103

Représenter l'héritage

- Héritage d'une table unique (pour toute la hierarchie),
- Superclasse de liaison: tables séparées,
  - Héritage de tables multiples (une table liée à toute les tables de la hierarchie ascendante),
- Héritage horizontal de tables (requiert une UNION).



Bases de Données relationnelles: Java

- JDBC (Java DataBase Connectivity)
  - le code du Servlet utilise l'API JDBC pour accéder au contenu de la base de données,
  - le pilote JDBC s'occupe de traduire les appels de l'API en requête SQL pour les RDBMS.
- Hibernate:
  - lie les classes Java aux tables de la base de données,
  - remplace les accès à la base de données par des méthodes de haut niveau.
  - HQL langage de requêtes orienté objet (polymorphismes, héritages)

Hibernate: Exemple de classe

```
public class Person {
  private int id, age;
  private String name;
  public Person() {}
  public Person(String name, int age) {
   this.name = name;
   this.age = age;
  }
  public int getId() { return id; }
  public void setId(int id) { this.id = id; }
  public String getName() { return firstName; }
  public void setName(String name) { ... }
  public int getAge() { return age; }
  public int setAge(int age) { this.age = age; }
}
```

- décrit un objet avec ses champs et ses méthodes get/set.

UPMC SORBONNE UNIVERSITÉS

Hibernate: Exemple

Création de table:

```
create table PERSON (
  id INT NOT NULL auto_increment,
  name VARCHAR(20) default NULL,
  age INT default NULL,
  PRIMARY KEY (id)
);
```

Exemple de fichier de liaison:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC ...>

<hibernate-mapping>
  <class name="Person" table="PERSON">
    <meta attribute="class-description">...</meta>
    <id name="id" type="int" column="id">
      <generator class="native"/>
    </id>
    <property name="name" column="name"
              type="string"/>
    <property name="age" column="age" type="int"/>
  </class>
</hibernate-mapping>
```

Bases de Données NoSQL

NoSQL (Not only SQL)

- Un mot générique pour désigner les technologies de base de données qui utilisent des modèles moins contraignants (sur la cohérence) que les modèles traditionnels relationnels.

- utilisé au départ pour les bases de données géantes (Google, Amazon).
- l'unité logique n'est plus la table.
- système de stockage clefs-valeurs

Exemples

- BigTable (Google)
- Dynamo (Amazon)
- HBase (Facebook)
- MongoDB (SourceForge.net)

PMC
BONNE UNIVERSITÉS

MongoDB

- système de gestion de bases de données orienté document
  - répartissable sur plusieurs ordinateurs,
  - efficace pour les requêtes simples dans de grosses bases,
  - ne nécessitant pas de schéma prédéfini de données.
- gratuit et libre (depuis 2009).
- développement commencé en 2007.
- utilisé par eBay, Foursquare, SourceForge

Modèle orienté-document

- une base MongoDB est un ensemble de collections (≃tables) constituées de documents (≃entrées).
- Le schéma de données est flexible:
  - les documents de la même collection n'ont pas forcément tous la même structure et les mêmes champs.
  - les champs communs à tous les documents d'une collection peuvent contenir des données différentes.
- l'atomicité est garantie au niveau du document.
- les documents sont du JSON en binaire (BSON)

```
{
"_id": ObjectId("4efa8d2b7d284dad101e4bc9"),
"Nom": "DUMONT",
"Prénom": "Jean",
"Âge": 43
},
```

Avantages

- les documents correspondent à des types de données natifs des langages de prog.
- les documents à l'intérieur des documents réduisent la nécessité de JOIN.
- les schémas dynamiques supportent le polymorphisme

Denormalisation

Les bases de données orientées document encourage le stockage d'information de manière dénormalisées pour éviter de faire trop de look-up.

Dénormalisé:

```
{ id: 42,
  title: "Tata Yoyo",
  tracks: [
  { id: 101
    title: "Tata Yoyo"},
  { id: 102
    title: "Cho Ka Ka o"},
  { id: 103
    title: "Frida Oum Papa"}
  ]
}
```

Normalisé:

```
{ id: 42
  title: "Tata Yoyo",
  tracks: [101, 102, 103]}
```

```
{ id: 101,
  title: "Tata Yoyo"}
```

```
{ id: 102,
  title: "Cho Ka Ka o"}
```

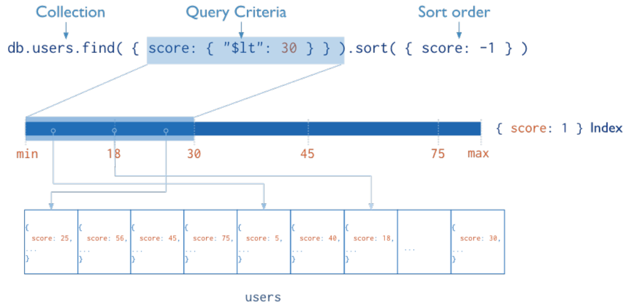
```
{ id: 102,
  title: "Frida Oum Papa"}
```

UPMC SORBONNE UNIVERSITÉS

Index MongoDB

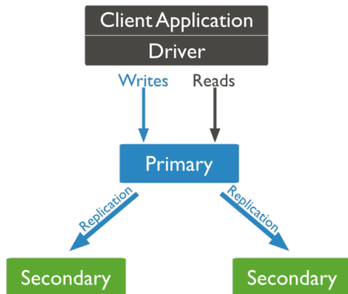
Les index sont des structures de données spéciales stockant une petite portion des données collections dans une forme facile à parcourir.

- les index stockent les valeurs d'un (de plusieurs) champ(s) spécifique(s), ordonnées par la valeur du champ.
- toutes les collections ont un index par défaut sur le champ `id_field`



Réplication

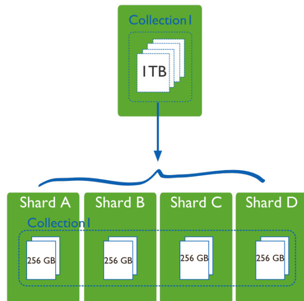
- la réplication ajoute de la redondance et augmente la disponibilité des données.
- la réplication de copies des données sur plusieurs serveurs permet la robustesse aux pannes serveurs.
  - un même client peut envoyer plusieurs ordres de lecture/écriture à des serveurs différents.



Eclatement (*Sharding*)

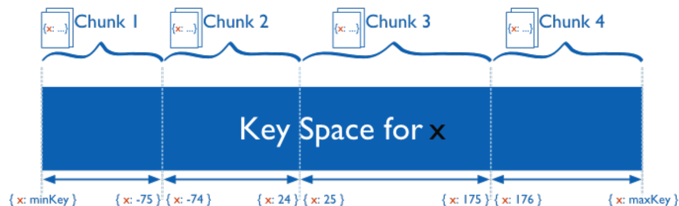
Permet de partitionner une collection dans une base de données pour distribuer une collection de documents sur plusieurs instances ou *shards*.

- la clef de *shard* détermine la distribution.
  - son choix est crucial pour une partition efficace.

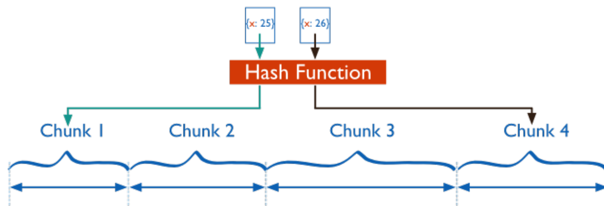


Partition (*Sharding*)

Partition sur intervalles:

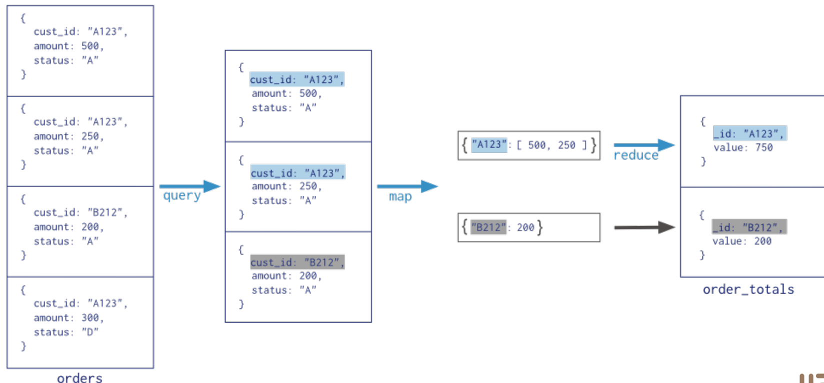


Partition par Hash:



Map-Reduce

```
   Collection
       ↓
db.orders.mapReduce(
          map      →   function() { emit( this.cust_id, this.amount ); },
          reduce   →   function(key, values) { return Array.sum( values ) },
                       {
          query    →     query: { status: "A" },
          output   →     out: "order_totals"
                       }
                     )
```



UPMC SORBONNE UNIVERSITÉS