An overview of Web development
(for PDL)

Master 1 - MIAGE

Mathieu Acher

Maître de Conférences

mathieu.acher@irisa.fr

UNIVERSITÉ DE RENNES 1

istic Informatique Électronique

Material

http://mathieuacher.com/teaching/iPDL-MIAGE1/

Back to PDL

Fournisseurs

Gestionnaire

Clients

Système de gestion de catalogue web de produits

Plateforme d'échange

Gestion des produits

configurateur

A rendre

- Livrable d'analyse (A1)
- Livrable de conception (D1)
- Livrable de validation (V1)
- Code (C1)
- ~~Livrable qualité (Q1)~~

Code (C1)

- Contenu
  - Code source de l'application
    - Incluant jeu de tests
  - Instructions sur comment installer, exécuter et tester l'application
- « Le code sera à rendre de façon électronique et non sur papier »
  - Soit via une archive mail
  - Soit en pointant la forge, le github ou googlecode
    - En s'assurant que j'ai bien les droits

Code (C1)

- Code source de l'application
  - Java (ou PHP)
  - Liberté totale pour les "frameworks"
    - JEE, Play!, GWT, Symfony, etc.
- Bonnes propriétés
  - Code documenté, respectant une certaine convention
  - Design patterns
  - Maven ou ANT-like pour générer la documentation, exécuter les tests, et déployer l'application

Code (C1) & Soutenance

- Date limite de rendu du “code”
  - **15 janvier 23:59:59**
  - ~ 1 jour avant la soutenance

- Soutenance 16 janvier
  - 15’ de présentation
  - 5’ de questions
    - Individuelles ou au groupe
  - Défense de votre travail
  - Retour d’expérience
    - A1, D1, V1, C1

Motivation

A rendre

- Livrable d'analyse (A1)
- Livrable de conception (D1)
- Livrable de validation (V1)
- Code (C1)
- ~~Livrable qualité (Q1)~~

Application Web


https://www.google.fr
+Mathieu
Recherche
Images
Maps
Play
YouTube
Actualités
Gmail
Drive
Agenda
Plus
Google
France
Recherche Google
J'ai de la chance

Google web

Web Images Maps Shopping Plus Outils de recherche

Environ 14 990 000 000 résultats (0,20 secondes)

World Wide Web - Wikipédia
fr.wikipedia.org/wiki/World_Wide_Web
Le World Wide **Web** (WWW), littéralement la « toile (d'araignée) mondiale », communément appelé le **web**, le **Web**, et parfois la toile, est un système hypertexte ...
Terminologie - Architecture - Types de ressource - Conception

WEB.DE - E-Mail-Adresse kostenlos, FreeMail, Nachrichten & Services
web.de/ - Traduire cette page
Das beliebteste Internetportal Deutschlands mit Angeboten rund um Suche, Kommunikation (E-Mail, De-Mail & mehr), Information und Services.

Définition > Internet - Web
www.futura-sciences.com/fr/definition/t/.../internet_3983/
Définition : Internet - Internet est un réseau informatique mondial constitué d'un ensemble de réseaux nationaux, régionaux et privés. L'ensemble utilise un ...

Web - 20 Minutes
www.20minutes.fr/**web**/
15 éléments – Sauter aux éditions locales; Sauter à la navigation; Sauter au ...
Google lance son «graph du savoir» en France Hier à 7h11 Facebook ...
Facebook: Avoir beaucoup d'amis génère du stress Mardi à 15h46 le 26 ...

WebGirondins.com - le site des supporters des Girondins de ...
www.**web**girondins.com/
Site entièrement consacré au club du FC Girondins de Bordeaux. Toute l'actualité par et pour les supporters.

ARTE Live Web

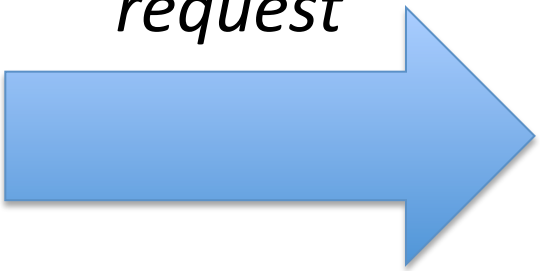
liveweb.arte.tv/fr
29 mai 2009
Présenté à la Biennale de la Danse de Lyon, au Musée du Quai Branly, et sur ARTE Live **Web**, le Lac des Cygnes ...

Autres vidéos pour **web** »

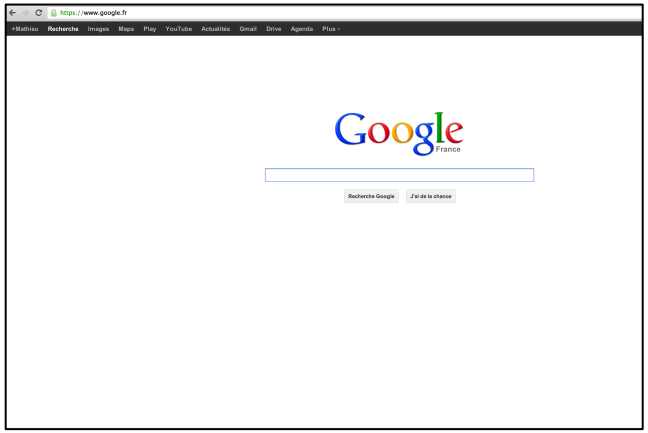
Paris Web – Accueil
www.paris-**web**.fr/
Paris **Web**, la conférence francophone des gens qui font le **web**, explore les thèmes de l'accessibilité **Web**, du design numérique et des standards ouverts.

Web browser

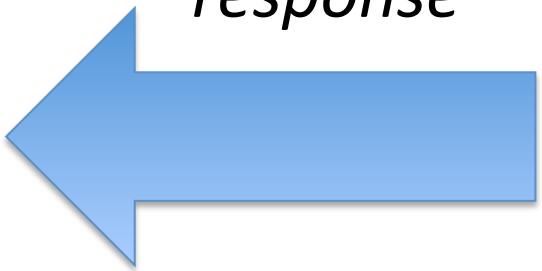
request



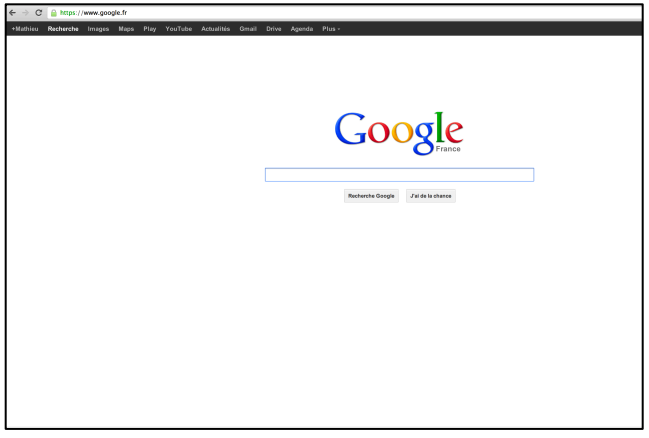
HTTP server



response



Web browser



request

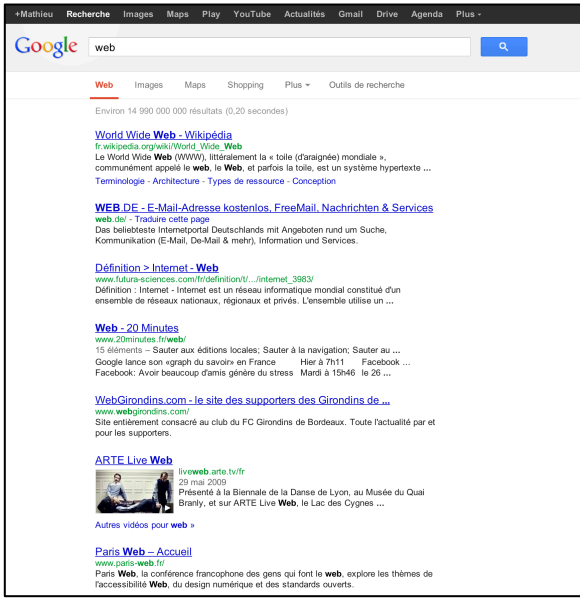
response

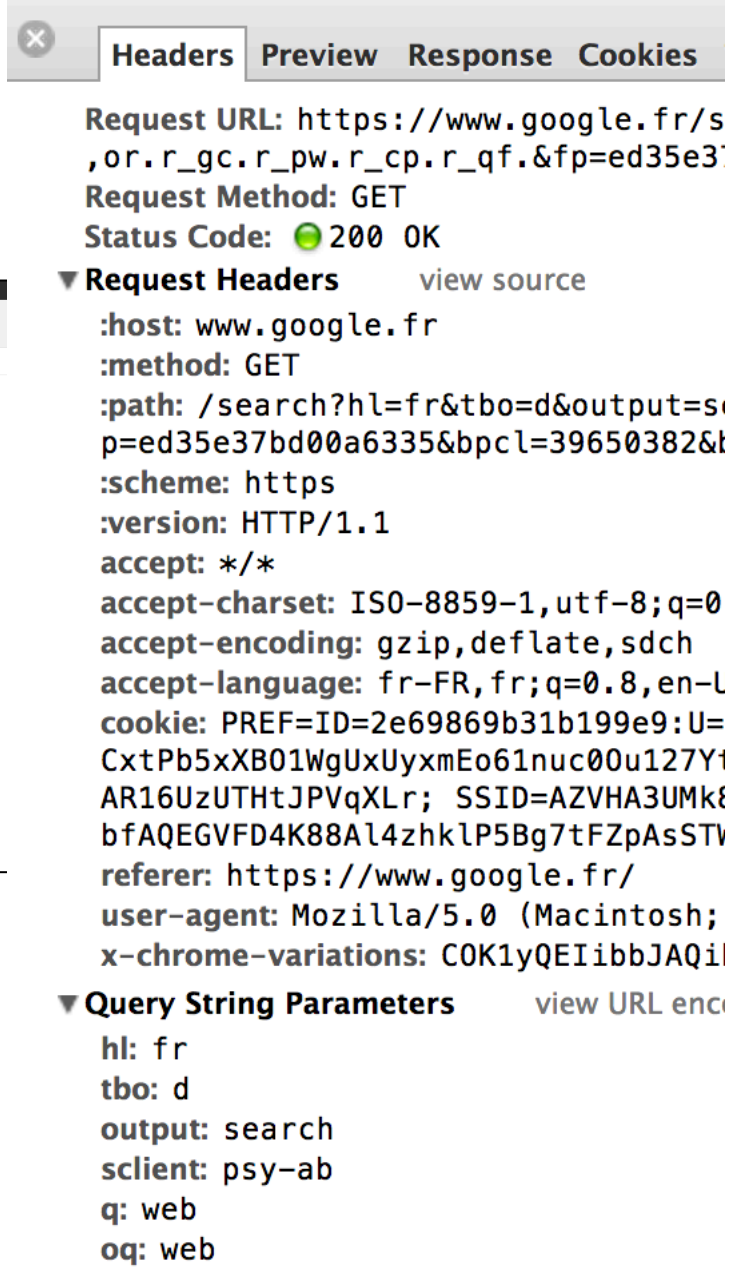
Request URL: https://www.google.fr/
Request Method: GET
Status Code: 200 OK
Request Headers view source
:host: www.google.fr
:method: GET
:path: /
:scheme: https
:version: HTTP/1.1
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
accept-encoding: gzip,deflate,sdch
accept-language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
cookie: PREF=ID=2e69869b31b199e9:U=b1fcad55eed1904b:FF=0:LD=fr:TM=1339164992:
CxtPb5xXBO1WgUxUyxmEo61nuc0Ou127YtQK7YYOPh-qm62ys5084UXT7NGSUy3xO_EWnF0LH9w
AR16UzUTHtJPVqXLr; SSID=AZVHA3UMk87RYchN6; APISID=1iFaN2NJgQOzCeZL/AOPySPpw
bfAQEGVFD4K88Al4zhklP5Bg7tFZpAsSTW-jY4SpVzy90DJnrzUQKCyMB-7FIqHzxDILxgOl_iQ
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.11
x-chrome-variations: COK1yQEIibbJAQibtskBCKW2yQEIp7bJAQiqtskBCLe2yQEIu4PKAQ==

Response Headers view source
cache-control: private, max-age=0
content-encoding: gzip
content-type: text/html; charset=UTF-8
date: Thu, 06 Dec 2012 04:55:00 GMT
expires: -1
server: gws
status: 200 OK
version: HTTP/1.1
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block

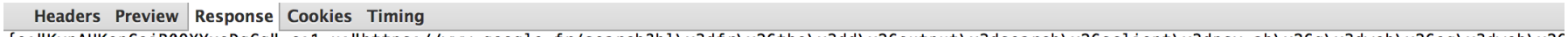
HTTP server

Web browser





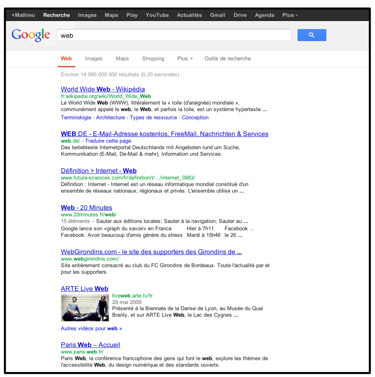
HTTP server



request

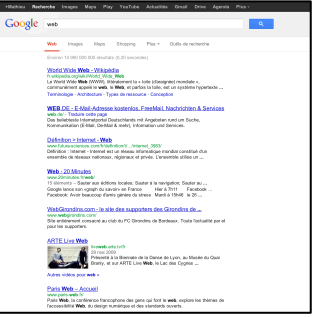
HTTP server

Web browser



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency
s	GET	200 OK	application/json	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	0B 2.12KB	65ms 65ms
s	GET	200 OK	application/json	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	0B 121.77KB	355ms 63ms
gen_204	GET	204 No Content	text/html	/:3 Script	241B 0B	58ms 57ms
s	GET	200 OK	application/json	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	0B 1.36KB	63ms 62ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/gif;base...	GET	Success	image/gif	/xjs/_/js/s/c,sb,wta,cr,cdos,j Script	(from cache)	0ms 0ms
data:image/jpeg;bas...	GET	Success	image/jpeg	/#hl=fr&tbo=d&output=searc Script	0B 1.67KB	0ms 0
data:image/jpeg;bas...	GET	Success	image/jpeg	/#hl=fr&tbo=d&output=searc Script	0B 1.94KB	0ms 0
data:image/jpeg;bas...	GET	Success	image/jpeg	/#hl=fr&tbo=d&output=searc Script	0B 2.20KB	0ms 0
data:image/jpeg;bas...	GET	Success	image/jpeg	/#hl=fr&tbo=d&output=searc Script	0B 2.88KB	0ms 0
data:image/jpeg;bas...	GET	Success	image/jpeg	/#hl=fr&tbo=d&output=searc Script	0B 2.05KB	0ms 0

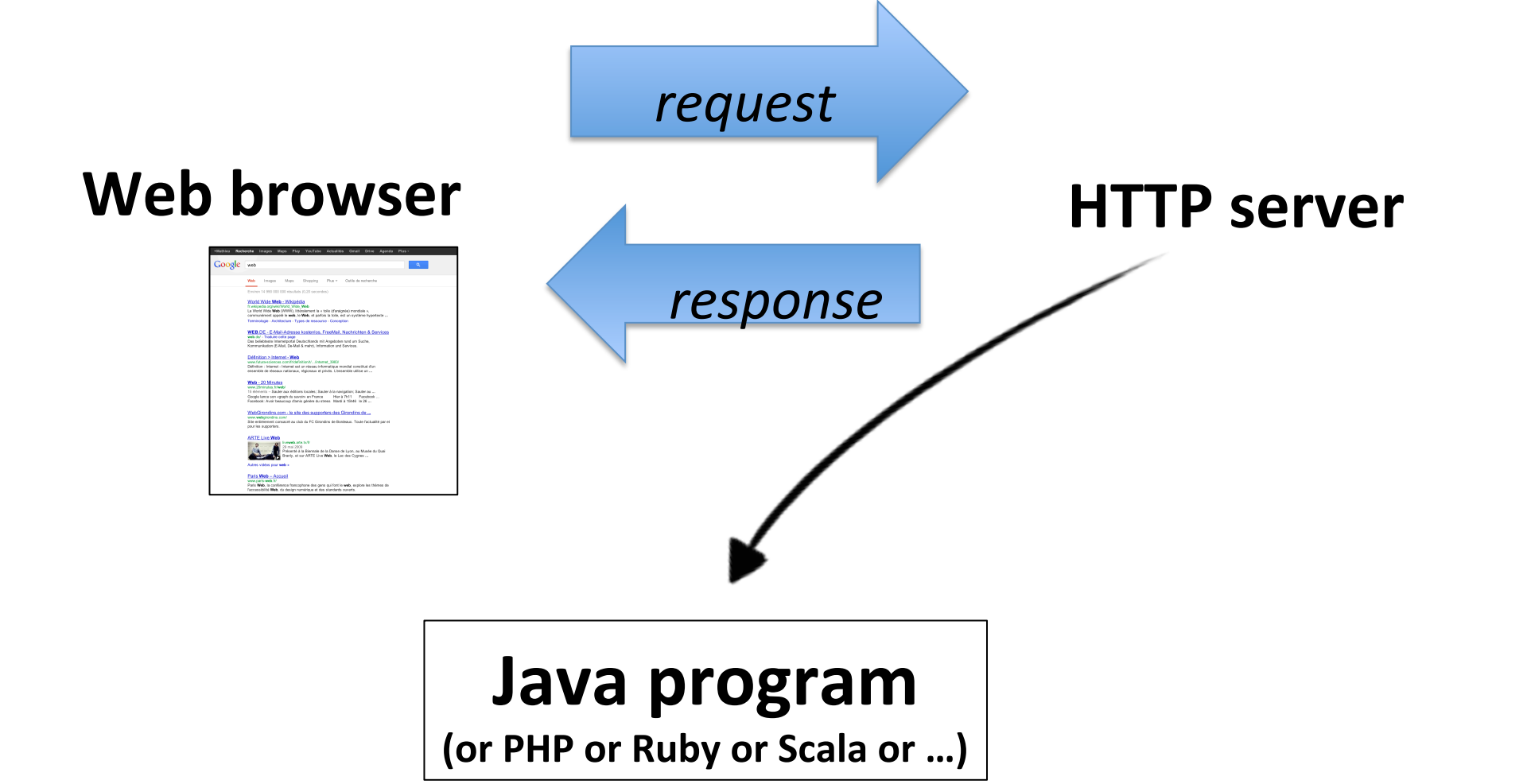
response



```
<!DOCTYPE html>
<html itemscope="itemscope" itemtype="http://schema.org/WebPage">
  <head>…</head>
  <body onload="try{if(!google.j.b){document.f&&document.f.q.focus();document.gbqf&&document.gbqf.q.focus();}}catch(e){}if(document.images)new Image().src='/images/nav_logo114.png'" dir="ltr" alink="#dd4b39" bgcolor="#fff" id="gsr" link="#12c" text="#222" vlink="#61c" style class="tbo vsh">
    <div id="pocs" style="position: absolute; z-index: 986; left: 126px; top: 79px; display: none;" class="sft">…</div>
    <div id="cst" style="display: none;">…</div>
    <a href="/setprefs?prev=https://www.google.fr/&sig=0_6x3nmUugcPvrsYlGwHkIz4c1HUY%3D&suggon=2" style="left:-1000em;position:absolute">…</a>
    <textarea name="csi" id="csi" style="display:none"></textarea>
    <script>if(google.j.b)document.body.style.visibility='hidden';</script>
    <div id="mngb">…</div>
    <iframe src="/blank.html" onload="google.j.l()" onerror="google.j.e()" name="wgjf" style="display:none">…</iframe>
    <textarea name="wgjc" id="wgjc" style="display:none"></textarea>
    <textarea name="wwcache" id="wwcache" style="display:none"></textarea>
    <textarea name="hcache" id="hcache" style="display:none"></textarea>
    <div id="main" style>
      <div id="ignore" style></div>
      <div>
        <div id="cnt" class>
          <script>…</script>
          <div class="mw">…</div>
          <div id="bst" style="display: none;">…</div>
          <div id="top_nav" style>…</div>
          <div id="appbar" style>…</div>
          <div class="mw" id="ucs" style></div>
          <div class="mw">
            <div id="rcnt" style="clear:both;position:relative;zoom:1">…</div>
            <div class="tsf-p" id="foot" role="contentinfo" style>…</div>
            <div id="bfoot" style>…</div>
          </div>
          <div id="xfoot" style>…</div>
          <div id="lfoot" style></div>
        </div>
      </div>
    </div>
    <script>…</script>
    <script data-url="/extern_chrome/ed35e37bd00a6335.js" id="ecs">…</script>
    <div id="xjsd">…</div>
    <div id="xjsi">…</div>
    <script>…</script>
    <script src="/xjs/_/js/s/sy8,qf,lor/rt=j/ver=ml8H-8_903g.en_US./d=0/sv=1/rs=AItRST0536Ewq15GFZZEVzACJOaXm-0N9w"></script>
    <table cellspacing="0" cellpadding="0" style="width: 572px; top: 78px; position: absolute; text-align: left; display: none; left: 126px;" class="gstl_0 gssb_c" dir="ltr">…</table>
    <script src="//ssl.gstatic.com/gb/js/smm_a79c05aaa286c9a4782660d65a6dd540.js" async></script>
    <script src="//ssl.gstatic.com/gb/js/abc/gci_91f30755d6a6b787dcc2a4062e6e9824.js" async gapi_processed="true"></script>
    <style type="text/css">…</style>
```

```
<head>
  <meta itemprop="image" content="/images/google_favicon_128.png">
  <title>web 2.0 - Recherche Google</title>
  <script src="https://apis.google.com/_/abc-static/_/js/gapi/googleapis_client,plusone/rt=j/ver=_CZX0pAF71I.en./sv=1/am=!7O-CuSNjNcIpPOou/d=1/cb=gapi.loaded_0" async></script>
  <script>
    (function(){
    window.google={kEI:"TCrAULeJLYi40QXqjYG4Cg",getEI:function(a){for(var b;a&&(!a.getAttribute||!(b=a.getAttribute("eid")));)a=a.parentNode;return b||google.kEI},https:function(){return"h
    b,c,j){var d=new Image,f=google.lc,e=google.li,g="";d.onerror=d.onload=d.onabort=function(){delete f[e];f[e]=!c&&-1==b.search("&ei=")&&(g="&ei="+google.getEI(j));c=c||"/gen_204?atyp=
    pm:"p",pl:[],mc:0,sc:0.5,u:"1cfeb9e1"},Toolbelt:{},y:{},x:function(a,b){google.y[a.id]=[a,b];return!1}};
    window.onpopstate=function(){google.j.psc=1};for(var h="ad api bc is p pa ac pc pah ph sa sifp slp spf spn x z zz".split(" "),i=0,k;k=h[i++];)(function(a){google.j[a]=function(){goog
    window.chrome||(window.chrome={}),window.chrome.sv=2.00,window.chrome.searchBox||(window.chrome.searchBox=[]),window.chrome.searchBox.onsubmit=function(){google.x({id:"psyapi"},functio
    window.google.sn="webhp";window.google.timers={};window.google.startTick=function(a,b){window.google.time[a]={t:{start:(new Date).getTime()},bfr:!(!b)}};window.google.tick=function(a
    (function(){'use strict';var d=null,j=this;var l="undefined"!=typeof navigator&&/Macintosh/.test(navigator.userAgent);var q="undefined"!=typeof navigator&&/iPhone|iPad|iPod/.test(naviga
  </script>
  <style>…</style>
  <style id="gstyle">…</style>
  <style>…</style>
  <script>…</script>
  <style type="text/css">…</style>
</head>
```

HTML + Javascript + CSS



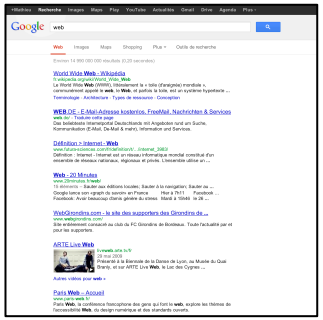
~ treat requests and user inputs (e.g., input forms)
~ generate HTML/JS/CSS stuff accordingly

Can be highly complex and require a combination of technology (user session, data storage, business logic, user interface concerns, performance, reliability, etc.)

request

response

Web browser



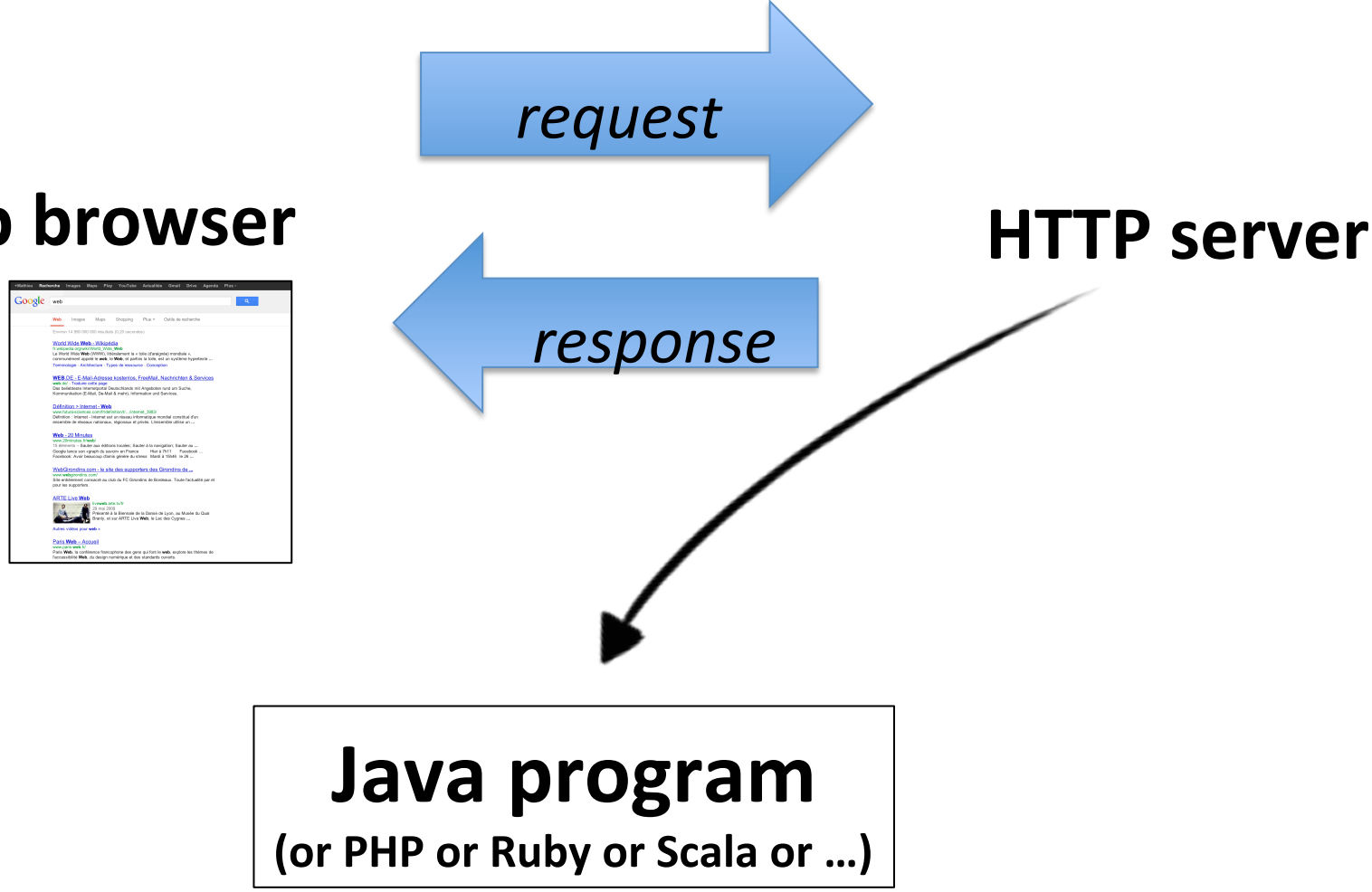
HTTP server

Java program

(or PHP or Ruby or Scala or …)

JEE (Servlets+JSP)

Play! framework



Disclaimer: this is just an overview with a quick focus on two technologies

#1 You can consider other technologies as well
(the principles are likely to be the same)

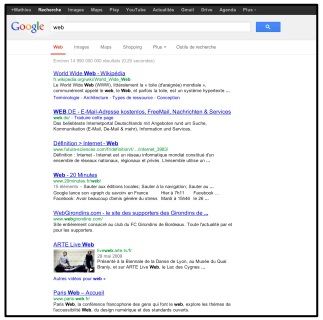
#2 You will have to learn and practice more by yourself
(available for any questions)

JEE (Servlets + JSP)

request

response

Web browser



HTTP server

Java servlet

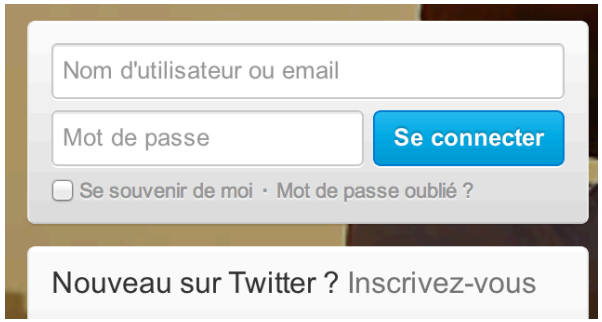
(and Java ecosystem actually)

Java Servlet

- Java program
  - rely on Java ecosystem
- Receive and treat HTTP requests
  - inputs (e.g., input forms)
  - resources (e.g., images)
- Generate web pages “on the fly”
  - Response to “clients” (browser)
  - HTML code + (JavaScript + CSS)

Web browser

HTTP server



request

response

Java servlet

```
<form action="https://twitter.com/sessions" class="signin" method="post">
  <div class="placeholding-input username">
    <input type="text" id="signin-email" class="text-input email-input" name="session[username_or_email]" title="Nom d'utilisateur ou email" autocomplete="on" tabindex="1">
    <label for="signin-email" class="placeholder">Nom d'utilisateur ou email</label>
  </div>
  <table class="flex-table password-signin">…</table>
  <div class="remember-forgot">…</div>
  <input type="hidden" name="return_to_ssl" value="true">
  <input type="hidden" name="scribe_log">
  <input type="hidden" name="redirect_after_login" value="/">
  <input type="hidden" value="6af33df72c59cf3ace4375a2dc7d2016cb96d726" name="authenticity_token">
</form>
```

Treat input forms

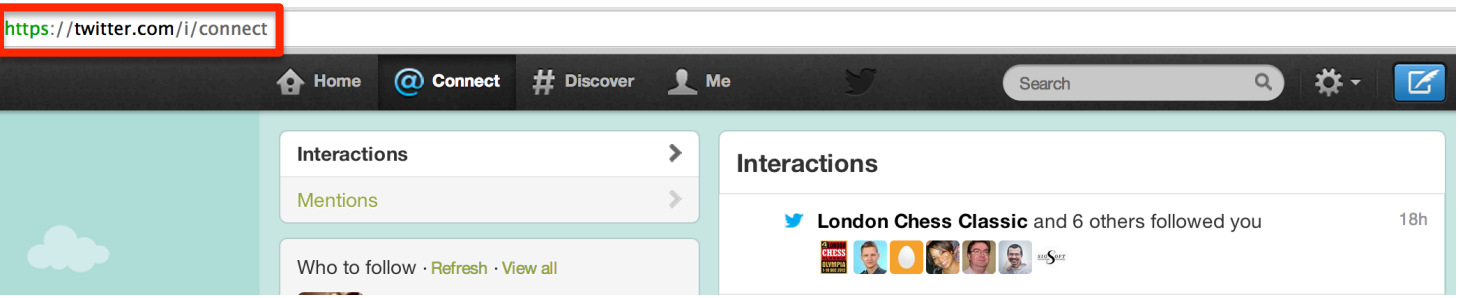
Identify/Retrieve User (if everything goes fine)

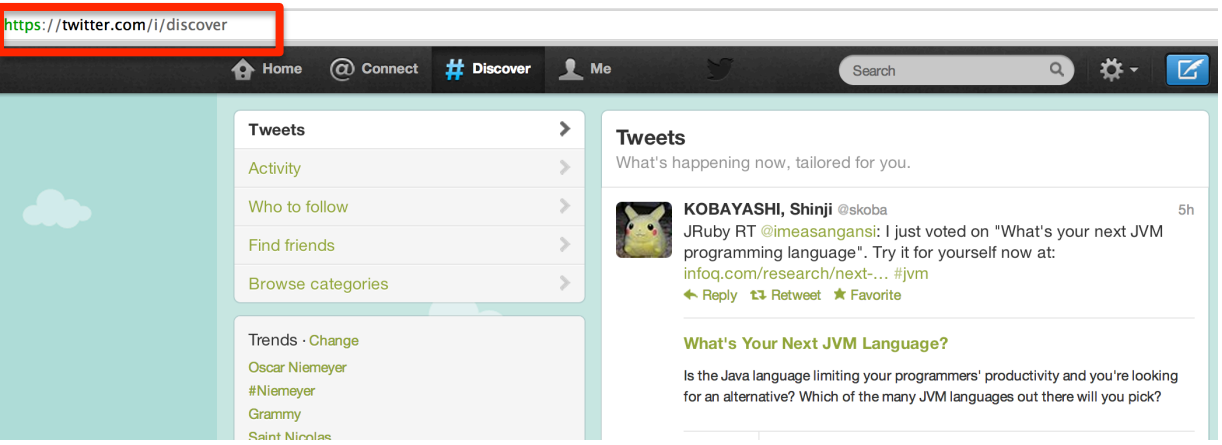
Find associated Followers/tweets/ads

Generate HTML stuff

The server responds according to the **URL** of the request (« **context** »)

Different applications/processings according to different contexts

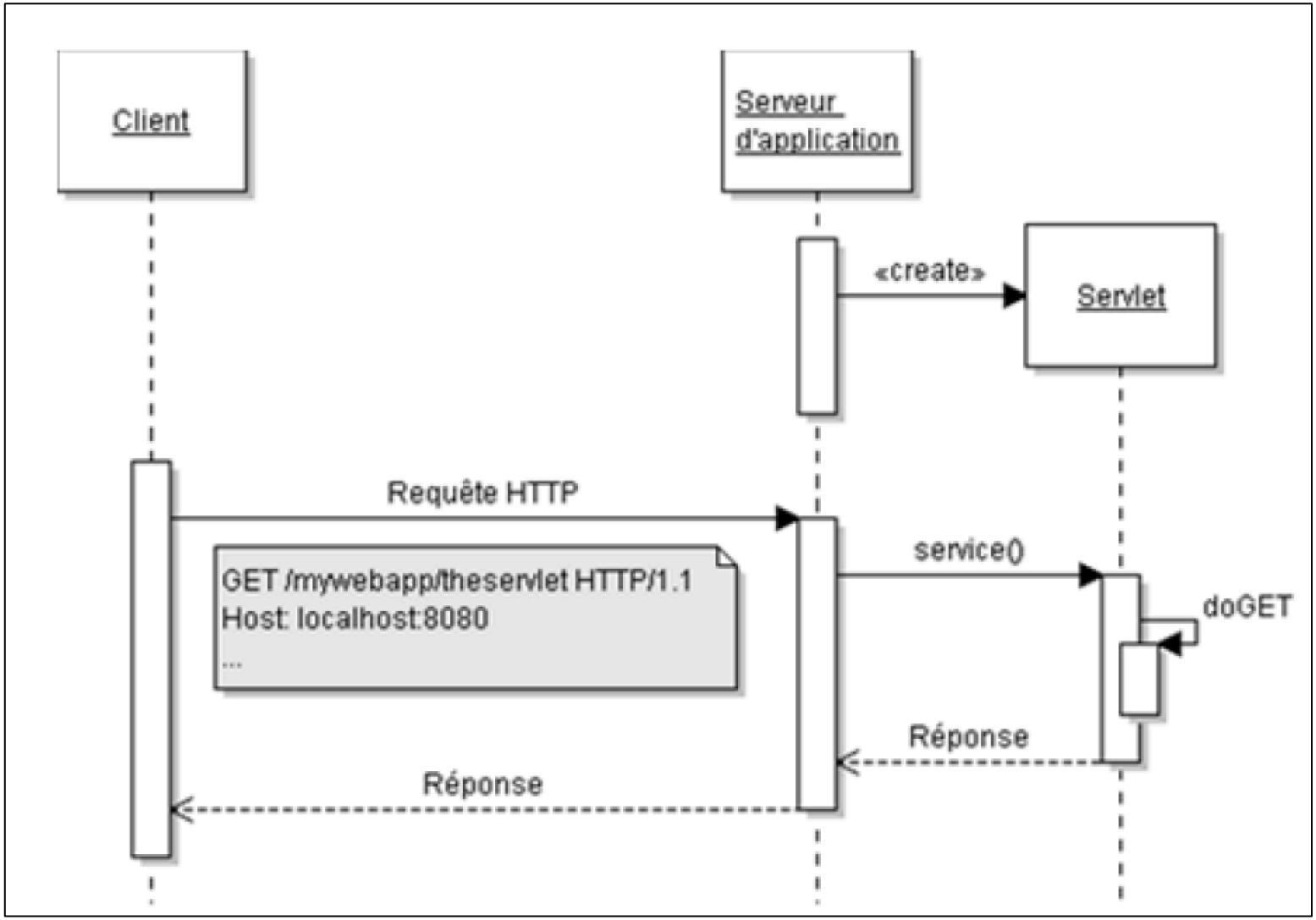




HTTP server

Java servlets

Client, Server and Java Servlet



Java Servlet (Hello World)

```java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {

  public void doGet (HttpServletRequest request,
  HttpServletResponse response)
  throws ServletException, IOException {

  response.setContentType("text/html");
  PrintWriter pw = response.getWriter();
  pw.println("<html>");
  pw.println("<head><title>Hello World</title></title>");
  pw.println("<body>");
  pw.println("<h1>Hello World</h1>");
  pw.println("</body></html>");

  }
}
```

Java Servlet (zoom on “doGet”)

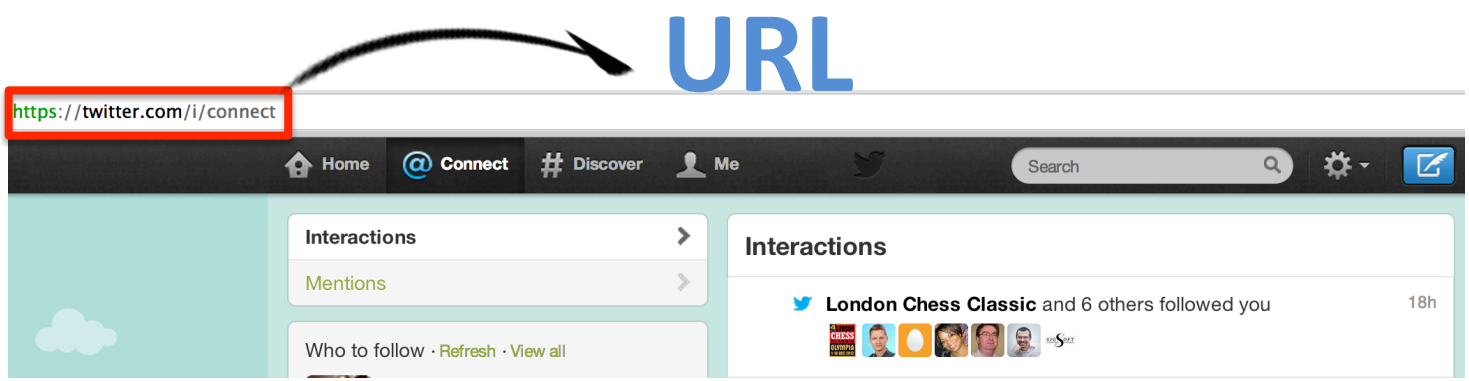
```
1  public void doGet(HttpServletRequest request,
2  HttpServletResponse response)
3  throws ServletException, IOException {
4  // use "request" for reading parameters
5  // and headers of HTTP request
6  ...
7  // treat the request
8  ...
9  // use response to specify the response status
10 // (including headers of the response)
11 ...
12 PrintWriter out = response.getWriter();
13 // send the content of the response
14 ...
15 }
```

Servlet Container

- HTTP server does not know how to execute the Java code of a servlet (obvious)
  - HTTP requests, that's it
- A servlet container is needed and manages a set of servlets
  - Management of servlet names (class names)
  - Creation and initialization of servlets
  - Deletion of servlets
- HTTP server delegates the requests to the container

HTTP server

Java
servlets



- Client (browser) won't specify a direct reference to a Servlet but rather an URL
- The web application should establish a correspondence between an URL and a servlet
  - **Mapping** URL-Servlet
- The corresponding container of the servlet will execute the servlet

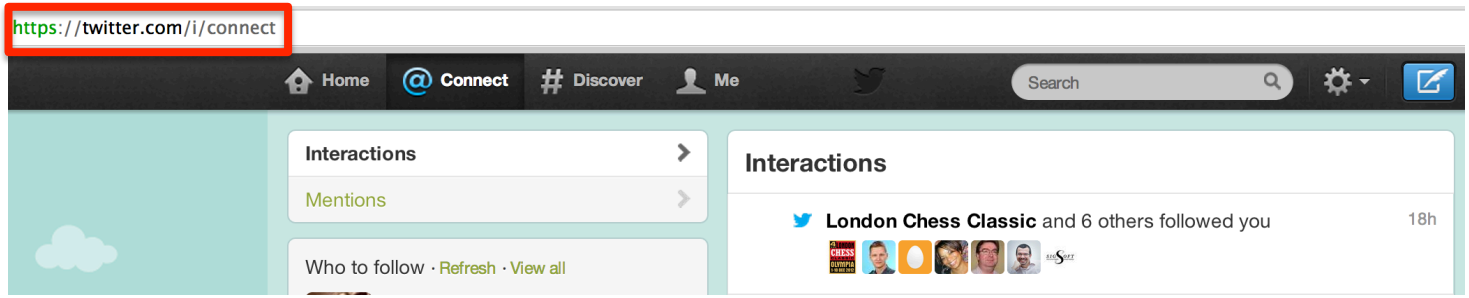
Mapping URL-Servlet

- Two solutions
  - Java annotations
  - web.xml (configuration file)

Mapping URL-Servlet

- **Java annotations** (since Servlet 3.0)

```
1  @WebServlet(urlPatterns={"/connect"})
2
3  public class Servlet1 extends HttpServlet {
4  ...
5  }
```



Mapping URL-Servlet

- **web.xml** (configuration file, since the beginning)
  - Included in the WAR archive that packages all code and resources of the applications

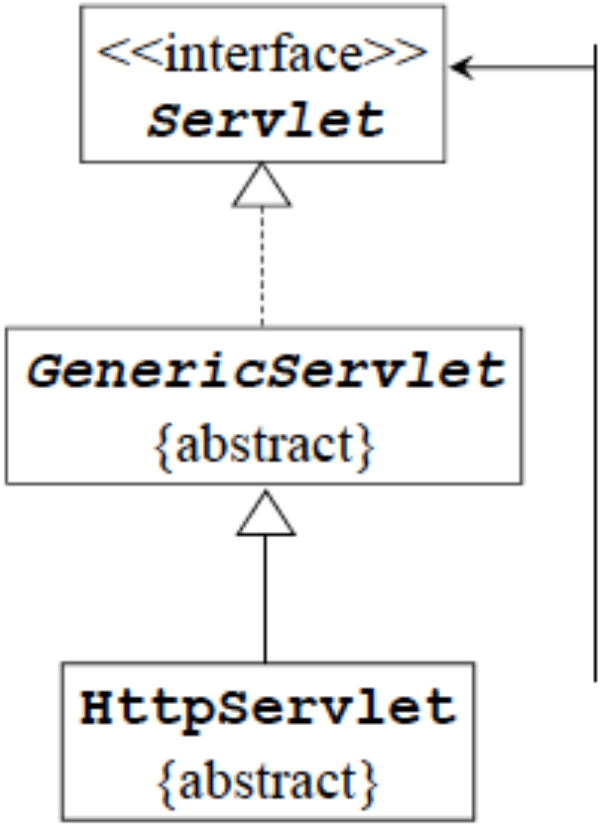
```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <display-name>HelloWorld Application</display-name>
    <description>
        This is a simple web application with a source code organization
        based on the recommendations of the Application Developer's Guide.
    </description>

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>examples.Hello</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

Mapping URL-Servlet

- **web.xml** (configuration file, since the beginning)
- Joker can be used
  - <url-pattern>/users/*</url-pattern>

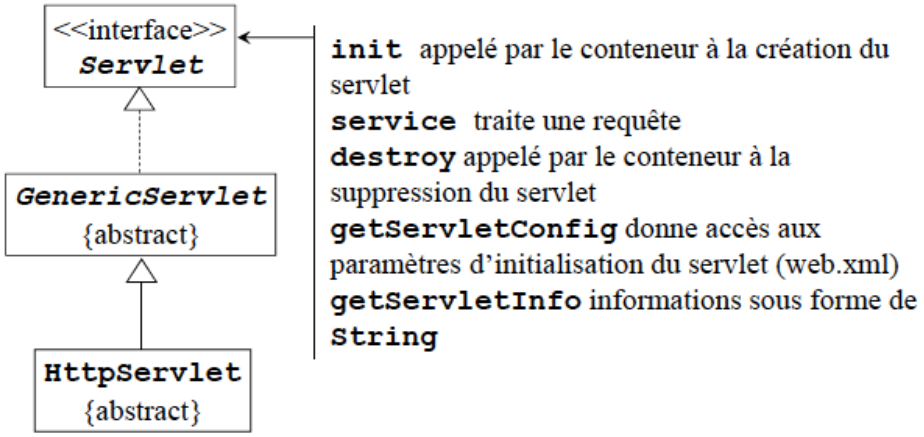
```xml
<?xml version="1.0" encoding="ISO-8859-1" ?>

<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd"
    version="2.4">

    <display-name>HelloWorld Application</display-name>
    <description>
        This is a simple web application with a source code organization
        based on the recommendations of the Application Developer's Guide.
    </description>

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>examples.Hello</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>

</web-app>
```

HttpServlet (1)



init appelé par le conteneur à la création du servlet
service traite une requête
destroy appelé par le conteneur à la suppression du servlet
getServletConfig donne accès aux paramètres d'initialisation du servlet (web.xml)
getServletInfo informations sous forme de **String**

HttpServlet (2)



- service() method of HttpServlet delegates the treatment to another method, depending on the kinf of HTTP request sent by the client
  - doGet(), doPost(), doPut(), doDelete(), doHead(), doOptions(), doTrace()

Writing a Servlet

- Boil down to…
- Writing a class that inherits from HttpServlet
- Override at least one method (doGet, doPost, etc.)
- Eventually override init or destroy if the servlet manages a set of resources that need to be initialized or removed

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class HelloWorld extends HttpServlet {
6
7    public void doGet (HttpServletRequest request,
8    HttpServletResponse response)
9    throws ServletException, IOException {
10
11   response.setContentType("text/html");
12   PrintWriter pw = response.getWriter();
13   pw.println("<html>");
14   pw.println("<head><title>Hello World</title></title>");
15   pw.println("<body>");
16   pw.println("<h1>Hello World</h1>");
17   pw.println("</body></html>");
18
19   }
20 }
```

Writing a Servlet

- Boil down to…
- Writing a class that inherits from HttpServlet
- Override at least one method (doGet, doPost, etc.)
- Eventually override init or destroy if the servlet manages a set of resources that need to be initialized or removed

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4
5  public class HelloWorld extends HttpServlet {
6
7    public void doGet (HttpServletRequest request,
8    HttpServletResponse response)
9    throws ServletException, IOException {
10
11   response.setContentType("text/html");
12   PrintWriter pw = response.getWriter();
13   pw.println("<html>");
14   pw.println("<head><title>Hello World</title></title>");
15   pw.println("<body>");
16   pw.println("<h1>Hello World</h1>");
17   pw.println("</body></html>");
18
19   }
20 }
```

Treating input forms (1)

Servlet in action

- HTTP recalls
  - With GET (within the URL):
    - http://www.truc.com/abonnement?id=23378&nom=Toto&op=d&v=56
  - With POST
    - Parameters are directly in the HTTP request

```
▼<form action="https://twitter.com/sessions" class="signin" method="post">
  ▼<div class="placeholding-input username">
      <input type="text" id="signin-email" class="text-input email-input" name="session[username_or_email]" title="Nom d'utilisateur ou email" autocomplete="on" tabindex="1">
      <label for="signin-email" class="placeholder">Nom d'utilisateur ou email</label>
    </div>
  ▶<table class="flex-table password-signin">…</table>
  ▶<div class="remember-forgot">…</div>
    <input type="hidden" name="return_to_ssl" value="true">
    <input type="hidden" name="scribe_log">
    <input type="hidden" name="redirect_after_login" value="/">
    <input type="hidden" value="6af33df72c59cf3ace4375a2dc7d2016cb96d726" name="authenticity_token">
  </form>
```

- Methods to get the parameters are independent of the kind of request (POST or GET)
  - As a result doGet can call doPost (or the other way)

Treating input forms (2)

Servlet in action

```
1  public void doGet(HttpServletRequest request,
2  HttpServletResponse response)
3  throws ServletException, IOException {
4  // use "request" for reading parameters
5  // and headers of HTTP request
6  ...
7  // treat the request
8  ...
9  // use response to specify the response status
10 // (including headers of the response)
11 ...
12 PrintWriter out = response.getWriter();
13 // send the content of the response
14 ...
15 }
```

- **HttpServletRequest**
  - Interface, represents a request of the client
- Several methods are provided
  - String getParameter (String nomParam)
  - String[] getParameterValues()
    - e.g. multivalues
  - Enumeration<String> getParameterNames()

http://www.truc.com/abonnement?id=23378&nom=Toto&op=d&v=56

```
1 <input type="text" name="nom">
2 <input type="checkbox" name="ville"
3 value="Nice">Nice<br>
4 <input type="checkbox" name="ville"
5 value="Paris">Paris<br>
```

```
1 String nom = req.getParameter("nom");
2 String[] villes =
3 req.getParameterValues("ville");
```

JSP (J

```
1  <html>
2
3  <head>
4  <title> Cu
5  </head>
6
7  <font face
8
9  The curren
10 <br />
11
12 <%= new
13
14 </font>
15
16 </body>
17 </html>
18
```

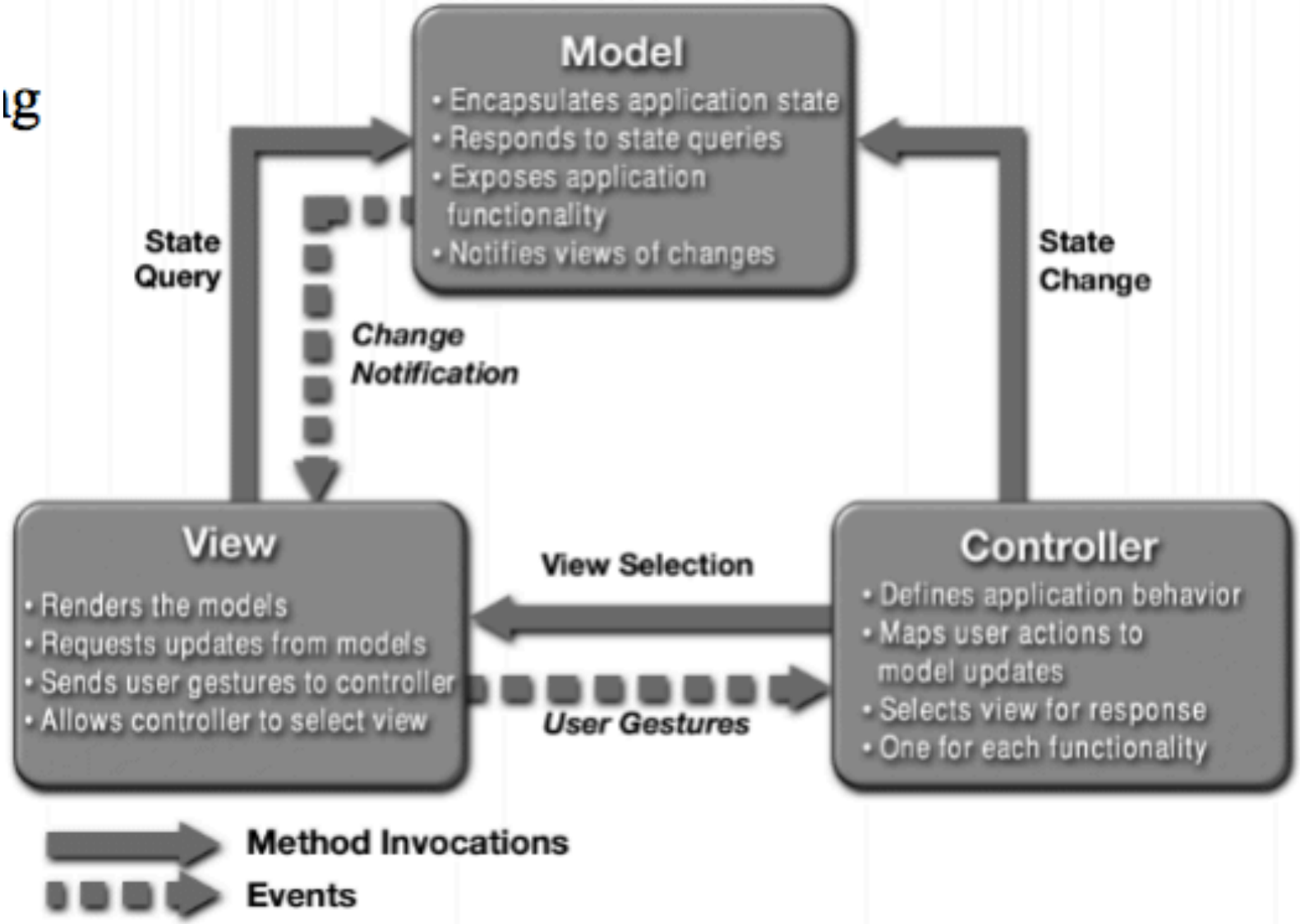
```
public User(String first, String last, String email)
    {
        firstName = first;
        lastName = last;
        emailAddress = email;
    }

    public void setFirstName(String f)
    {
        firstName = f;
    }

    }
    public String getFirstName()
    {
        return firstName;
    }

    public void setLastName(String l)
    {
        lastName = l;
    } The code for the User bean class (cont.)
    public String getLastName()
    {
        return lastName;
    }

    public void setEmailAddress(String e)
    {
        emailAddress = e;
    }

    public String getEmailAddress()
    {
        return emailAddress;
    }
}
```

JSP + Javabean

```
public User(String first, String last, String email)
    {
        firstName = first;
        lastName = last;
        emailAddress = email;
    }

    public void setFirstName(String f)
    {
        firstName = f;
    }

    }
    public String getFirstName()
    {
        return firstName;
    }

    public void setLastName(String l)
    {
        lastName = l;
    } The code for the User bean class (cont.)
    public String getLastName()
    {
        return lastName;
    }

    public void setEmailAddress(String e)
    {
        emailAddress = e;
    }

    public String getEmailAddress()
    {
        return emailAddress;
    }
}
```

```
<jsp:useBean id="user" scope="session"
class="business.User"/>
 <table cellspacing="5" cellpadding="5" border="1">
     <tr>
         <td align="right">First name:</td>
         <td><jsp:getProperty name="user"
             property="firstName"/></td>
     </tr>
     <tr>
         <td align="right">Last name:</td>
         <td><jsp:getProperty name="user"
             property="lastName"/></td>
     </tr>
     <tr>
         <td align="right">Email address:</td>
         <td><jsp:getProperty name="user"
             property="emailAddress"/></td>
     </tr>
 </table>
```

ıg

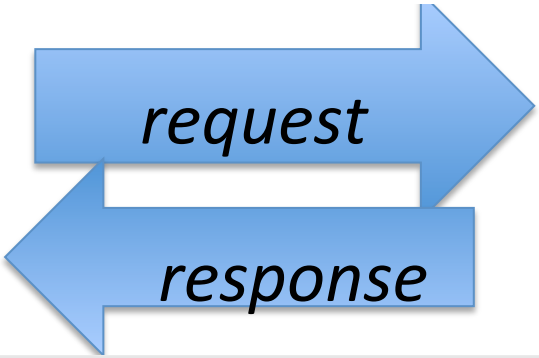

Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
State Change
Change Notification
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

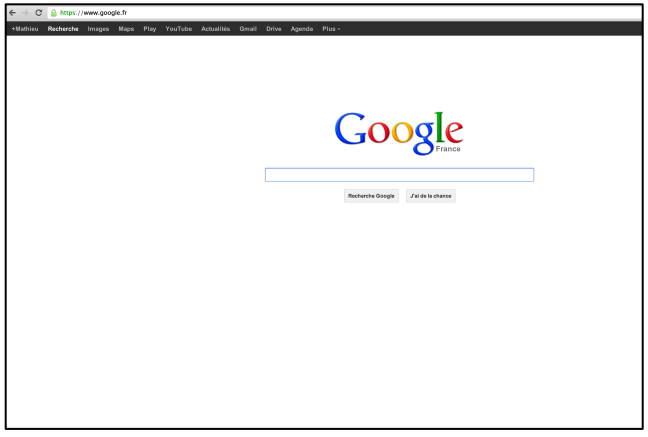
Play!

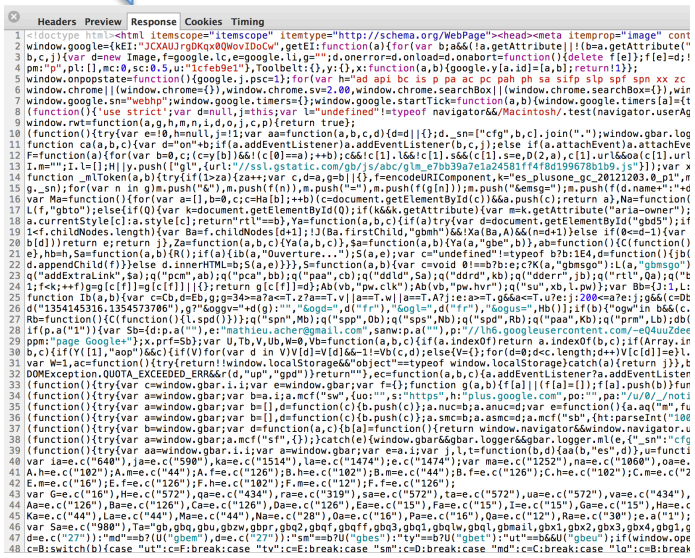
First, let us talk about REST architecture (revisiting HTTP)

(revisiting)

Web browser







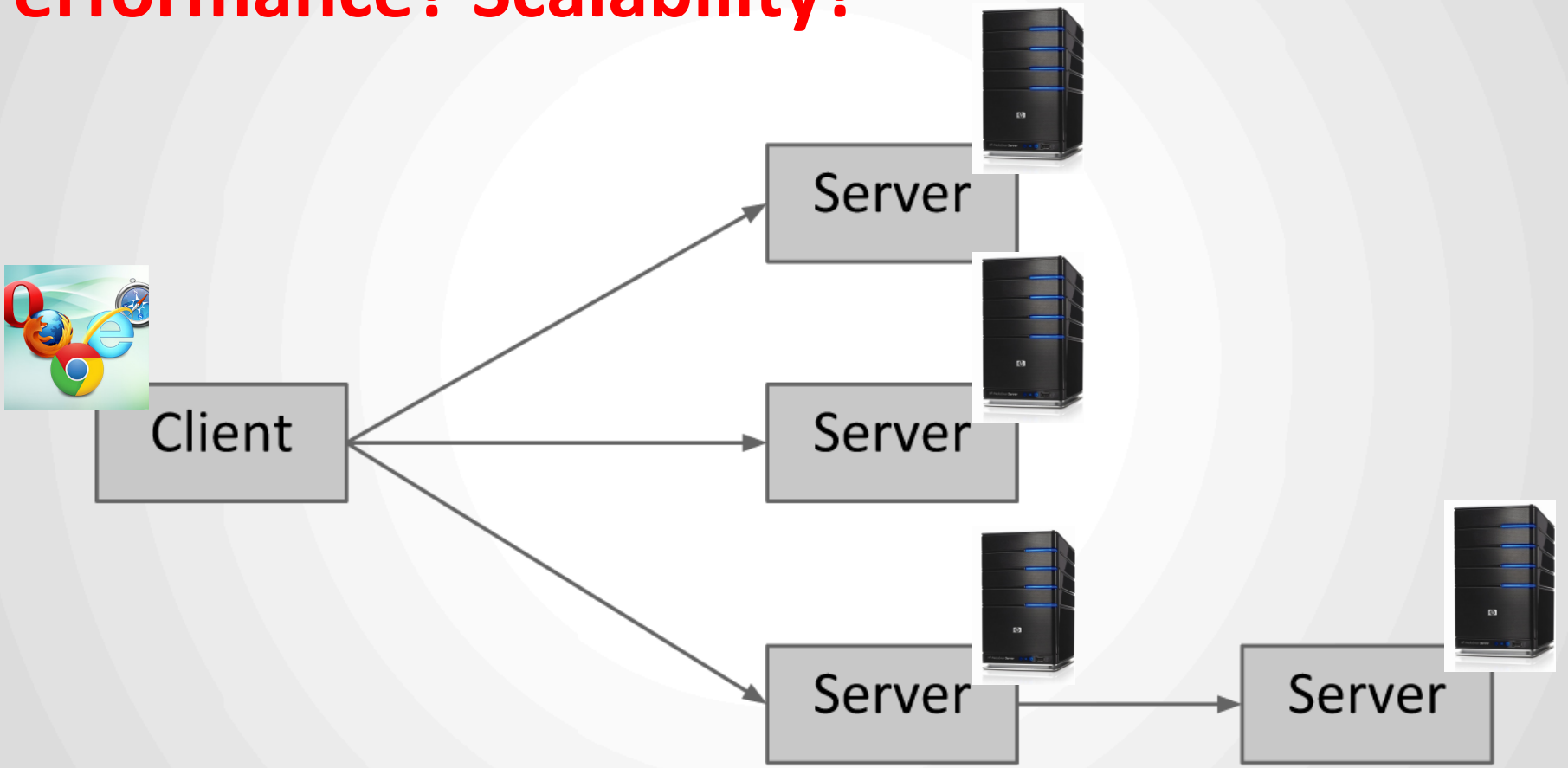
Request URL: https://www.google.fr/
Request Method: GET
Status Code: 200 OK
le.com/ t Headers view source
:host: www.google.fr
:method: GET
:path: /
:scheme: https
:version: HTTP/1.1
accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
accept-charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
accept-encoding: gzip,deflate,sdch
accept-language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
cookie: PREF=ID=2e69869b31b199e9:U=b1fcad55eed1904b:FF=0:LD=fr:TM=1339164992:
CxtPb5xXBO1WgUxUyxmEo61nuc0Ou127YtQK7YYOPh-qm62ys5084UXT7NGSUy3xO_EWnF0LH9w
AR16UzUTHtJPVqXLr; SSID=AZVHA3UMk87RYchN6; APISID=1iFaN2NJgQOzCeZL/AOPySPpw
bfAQEGVFD4K88Al4zhklP5Bg7tFZpAsSTW-jY4SpVzy90DJnrzUQKCyMB-7FIqHzxDILxgOl_iQ
user-agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.11
x-chrome-variations: COK1yQEIibbJAQibtskBCKW2yQEIp7bJAQiqtskBCLe2yQEIu4PKAQ==

▼ Response Headers view source
cache-control: private, max-age=0
content-encoding: gzip
content-type: text/html; charset=UTF-8
date: Thu, 06 Dec 2012 04:55:00 GMT
expires: -1
server: gws
status: 200 OK
version: HTTP/1.1
x-frame-options: SAMEORIGIN
x-xss-protection: 1; mode=block

HTTP server

Web Architecture

Performance? Scalability?



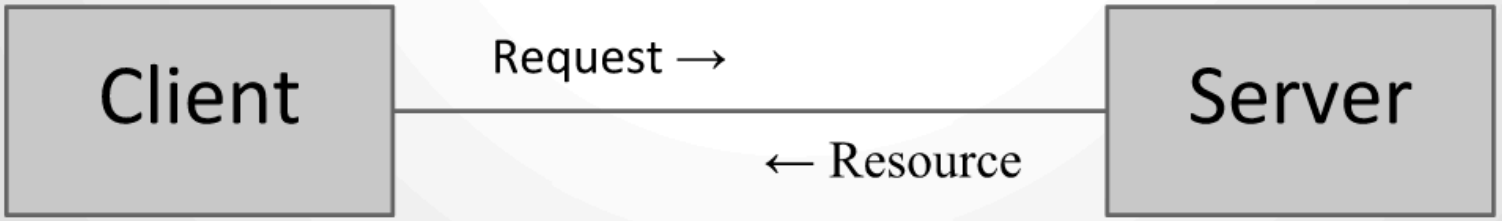
REST

Representational State Transfer

- Roy Fielding, 2000
- Architectural style for distributed systems
- Decrease coupling between Web Services
- Improve scalability
- **Stateless**

REST & HTTP (basic)

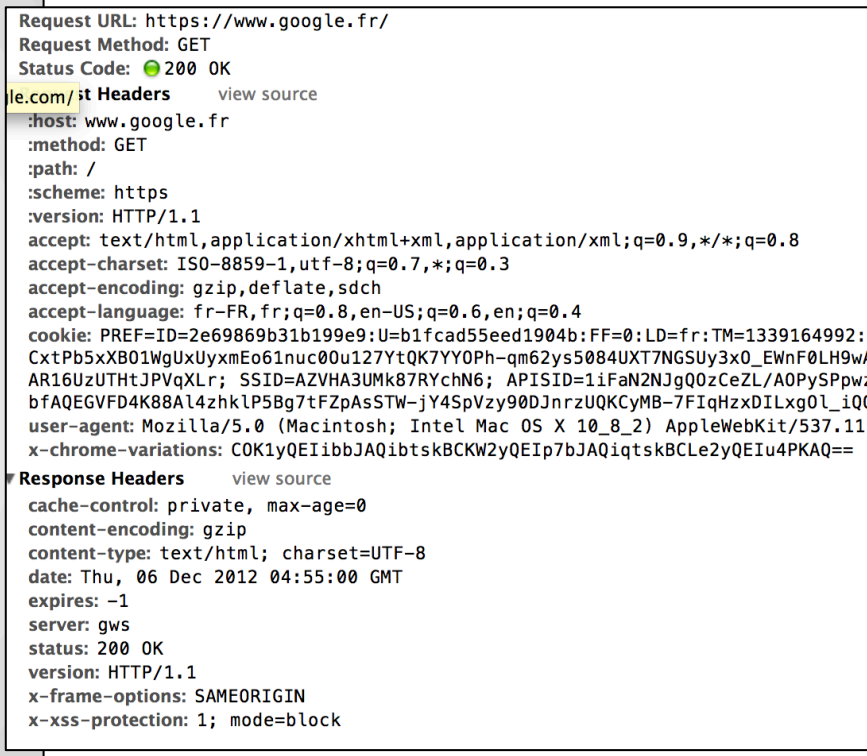
- An HTTP request is a self-descriptive message
- Resources are identified by an URL and manipulated through their representations



HTTP request (basics)

HTTP Request

- URL: `http://myapp.com/foo/bar?baz=bah`
- Verb
  - GET, get the current resource state, nullipotent
  - POST, create a new resource
  - PUT, update an existing resource, idempotent
  - DELETE, delete a resource, idempotent
- Headers
  - `Accept`, `Accept-Language`, `Accept-Encoding`
  - `Content-Type`
  - `Cookie`
  - `If-Modified-Since`
  - `User-Agent`
- Body



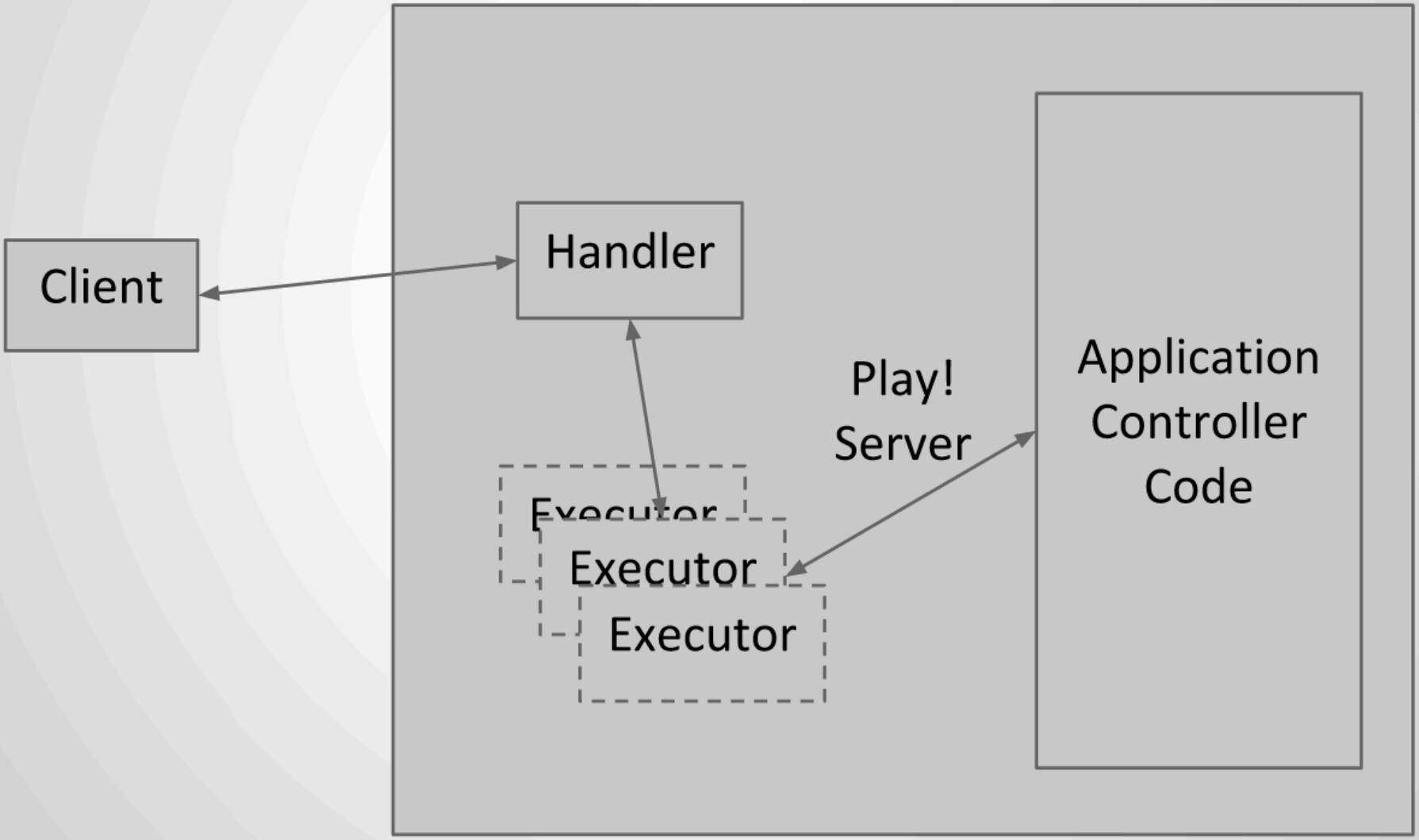
HTTP Response

- Status
  - 2xx, success (e.g. 200 OK, 206 Partial Content)
  - 3xx, redirection (e.g. 303 See Other, 304 Not Modified)
  - 4xx, client error (e.g. 404 Not Found)
  - 5xx, server error (e.g. 500 Internal Server Error, 503 Service Unavailable)
- Headers
  - Content-Type, Content-Length
  - Expires, ETag, Last-Modified, Cache-Control
  - Set-Cookie
  - Location
- Body

Play!

- **Web framework** for Java and Scala
- **Stateless** (client session stored in cookies) **REST principles**
- **Asynchronous** programming model based on Futures
- **Reactive** programming model for stream processing
- **Modern** HTTP support (WebSockets, Server-Sent Events, Chunked transfer encoding)
- Version 2.0 released in March 2012
- More than 2,000 followers and 500 forks

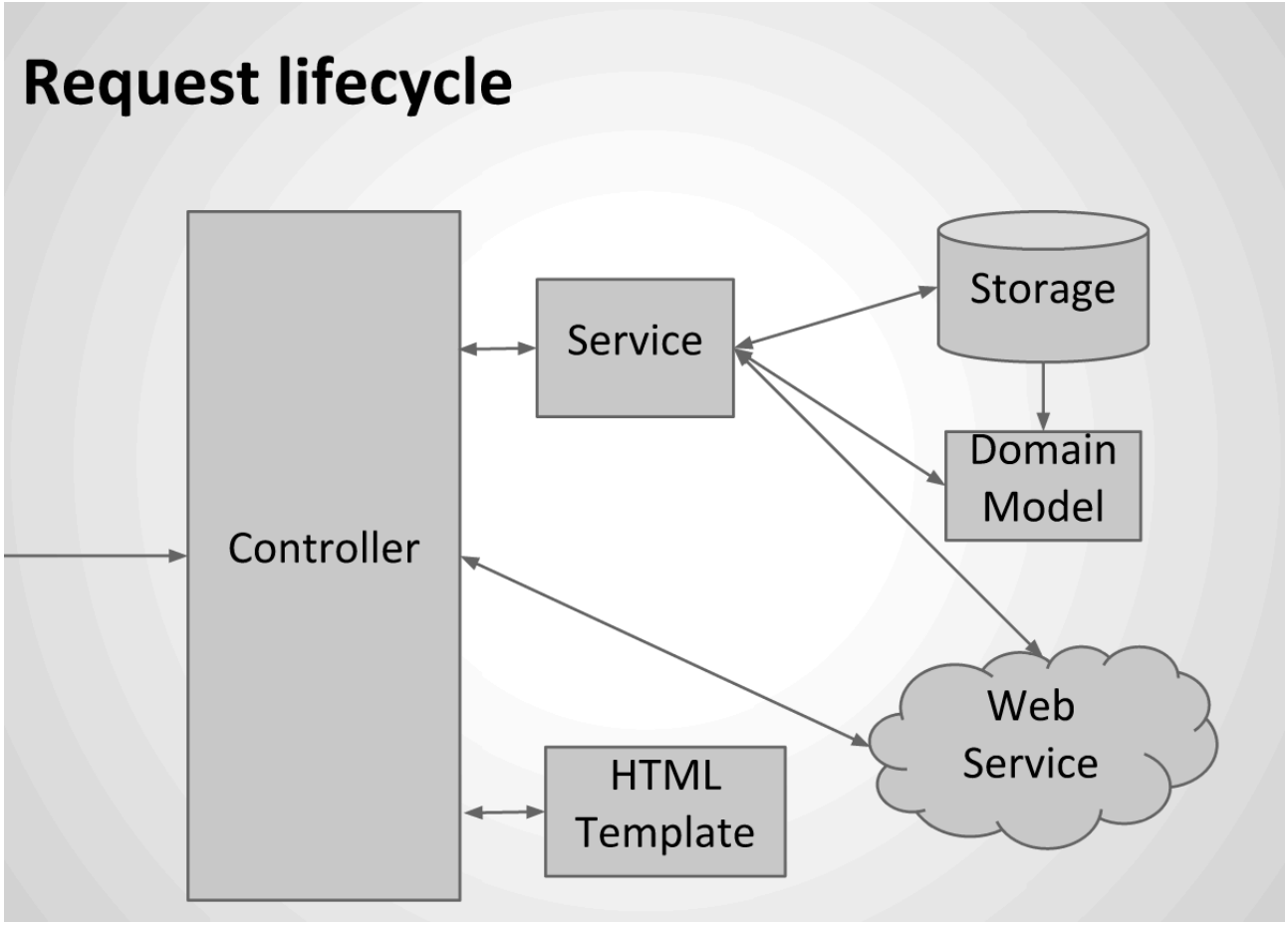
Request lifecycle



Play! framework

offers an integrated solution

- controller facilities for treating HTTP requests (controllers)
- template language/engine (views)
- specification/serialization of domain model (models)



Organization of a Play! Project (main excerpt)

```
app                   → Application sources
 └ assets             → Compiled asset sources
    └ stylesheets     → Typically LESS CSS sources
    └ javascripts     → Typically CoffeeScript sources
 └ controllers        → Application controllers
 └ models             → Application business layer
 └ views              → Templates
conf                  → Configurations files and other non-compiled resources
 └ application.conf   → Main configuration file
 └ routes             → Routes definition
public                → Public assets
 └ stylesheets        → CSS files
 └ javascripts        → Javascript files
 └ images             → Image files
```

Model
View
Controller

~ mapping URL – controller action

Example: routes definition

mapping URL – controller action

routes (file)

```
# Hello action
GET    /hello/:name          controllers.Application.hello(name)
```

controller

```
package controllers;

import play.*;
import play.mvc.*;

public class Application extends Controller {

  public static Result hello(String name) {
      return ok("Hello " + name + "!");
  }

}
```

http://localhost:9000/hello?name=World

Example: routes definition

mapping URL – controller action

```
# Home page
GET     /                           controllers.Application.index()

# Tasks
GET     /tasks                      controllers.Application.tasks()
POST    /tasks                      controllers.Application.newTask()
POST    /tasks/:id/delete           controllers.Application.deleteTask(id: Long)
```

```
public class Application extends Controller {

  public static Result index() {
    return ok(index.render("Your new application is ready."));
  }

  public static Result tasks() {
    return TODO;
  }

  public static Result newTask() {
    return TODO;
  }

  public static Result deleteTask(Long id) {
    return TODO;
  }

}
```

Controller action and result

```
package controllers;

import play.*;
import play.mvc.*;

public class Application extends Controller {

  public static Result hello(String name) {
      return ok("Hello " + name + "!");
  }

}
```

```
Result ok = ok("Hello world!");
Result notFound = notFound();
Result pageNotFound = notFound("<h1>Page not found</h1>").as("text/html");
Result badRequest = badRequest(views.html.form.render(formWithErrors));
Result oops = internalServerError("Oops");
Result anyStatus = status(488, "Strange response type");
```

Model

```
package models;

import java.util.*;
import javax.persistence.*;

import play.db.ebean.*;
import play.data.format.*;
import play.data.validation.*;

@Entity
public class Task extends Model {

  @Id
  @Constraints.Min(10)
  public Long id;

  @Constraints.Required
  public String name;

  public boolean done;

  @Formats.DateTime(pattern="dd/MM/yyyy")
  public Date dueDate = new Date();

  public static Finder<Long,Task> find = new Finder<Long,Task>(
    Long.class, Task.class
  );

}
```

Persistance
and Querying for free

Annotations define the constraints on the model data
Models can be serialized in a database (eg SQL)

Model and Controller

```java
package models;

import java.util.*;
import javax.persistence.*;

import play.db.ebean.*;
import play.data.format.*;
import play.data.validation.*;

@Entity
public class Task extends Model {

  @Id
  @Constraints.Min(10)
  public Long id;

  @Constraints.Required
  public String name;

  public boolean done;

  @Formats.DateTime(pattern="dd/MM/yyyy")
  public Date dueDate = new Date();

  public static Finder<Long,Task> find = new Finder<Long,Task>(
    Long.class, Task.class
  );

}
```

code that can be used in the controller actions

```java
// Find all tasks
List<Task> tasks = Task.find.all();

// Find a task by ID
Task anyTask = Task.find.byId(34L);

// Delete a task by ID
Task.find.ref(34L).delete();

// More complex task query
List<Task> tasks = find.where()
    .ilike("name", "%coco%")
    .orderBy("dueDate asc")
    .findPagingList(25)
    .getPage(1);
```

Template Language/Engine

Mix of HTML (+ Javascript + CSS)

and

Application code (models)

```
<div class="article">
  <span>@article.name</span>
  <span>@article.price €</span>
</div>
<ul>
  @for(article <- articles) {
    <li>@show(article)</li>
  }
</ul>
@if(isAdmin) {
  <button>Delete</button>
}
```

Much more here:

http://www.playframework.org/documentation/

If you choose Play!

start with a simple tutorial
don't try to master everything about the framework
get simple results ASAP

Advices for the PDL

- Models
  - Rely on your previous work (« livrable de conception »)
  - Facilities are offered/integrated by Play! but be careful
    - The learning curve is high
    - « Magics » can be a nightware
    - You may prefer control the way models are serialized
- Views
  - Simple HTML pages
    - as a client, I don't care about the usability and look and feel
  - Templates (JSP or Play! template engines)
- Controllers
  - HTTPServlet (Servlet) or Controller in Play!

Fournisseurs

Gestionnaire

Clients

Système de gestion de catalogue web de produits

Plateforme d'échange

Gestion des produits

configurateur