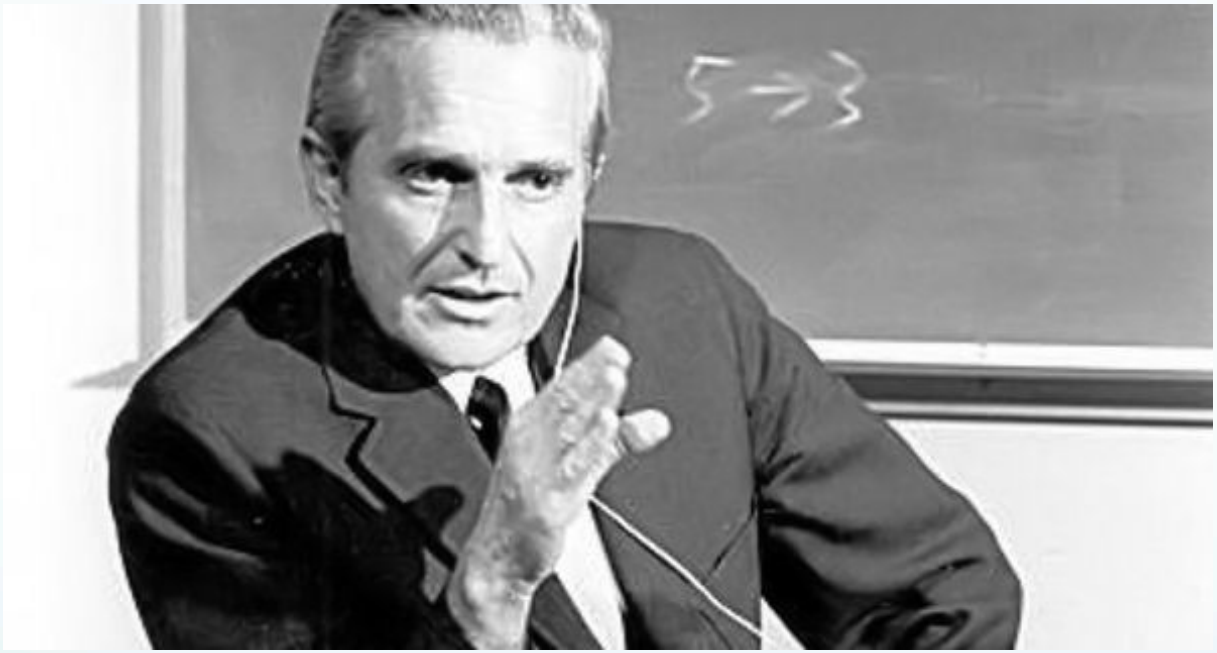
Web Applications and Cloud Computing

Pascal Molli, Nantes University, GDD team

Engelbart 1997 : Turing Award



"For an **inspiring vision** of the future of interactive computing and the **invention of key technologies** to help realize this vision."

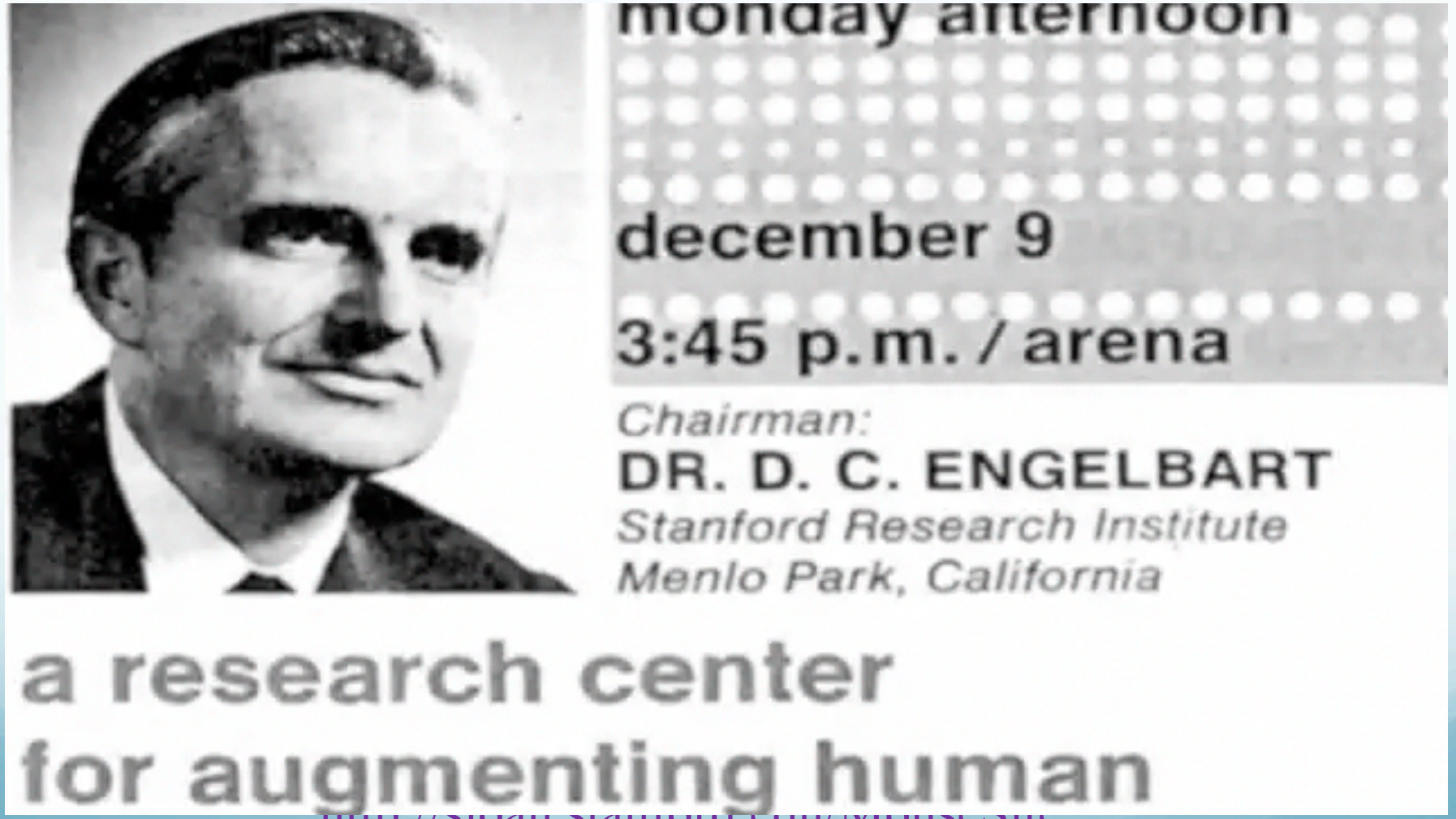
« For an inspiring vision …»

- 1962 : « La complexité et l'urgence des problèmes auxquels nous devons faire face croissent plus vite que notre capacité à les comprendre et à les résoudre. C'est un problème essentiel, nous pouvons prendre des mesures stratégiques, collectivement »
- L'approche d'Engelbart :
  - « Augmenter les capacités intellectuelles humaines en augmentant notre **intelligence collective**… »

« For an inspiring vision …»

- Dans les années 60, Engelbart développe l'outil NLS basé sur 2 principes:
  1. « **Develop knowledge Collection:** » Codifier et structurer les connaissances.
     - En 1962, approche centrée autour des hypertextes
  2. «**Use networked improvement communities** »
     - Des communautés pour développer et maintenir les connaissances.
     - Création du premier « groupware », collecticiel

« … and **key technologies**… »



http://sloan.stanford.edu/MouseSite

Augmenter notre intelligence collective...

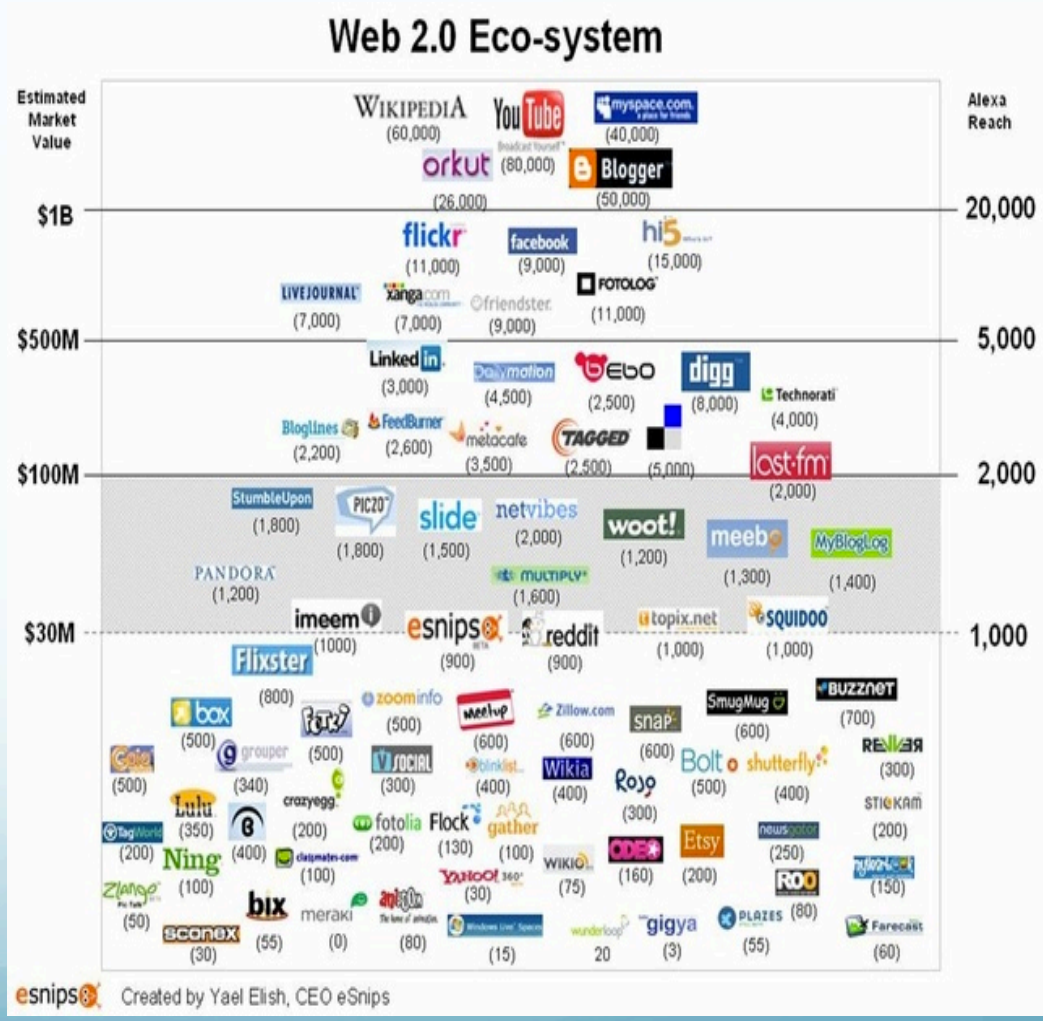
- « **Develop knowledge Collection:** » Codifier et structurer les connaissances.
- « **Use networked improvement communities:** » des communautés pour développer et maintenir les connaissances
- « **Co-evolution of Tools and Work:** » Construire de nouveaux outils, les utiliser, adapter nos pratiques aux outils, adapter les outils aux nouvelles pratiques que nous inventons.
- « **Innovate how we work collectively:** » repenser la manière dont les personnes travaillent ensemble

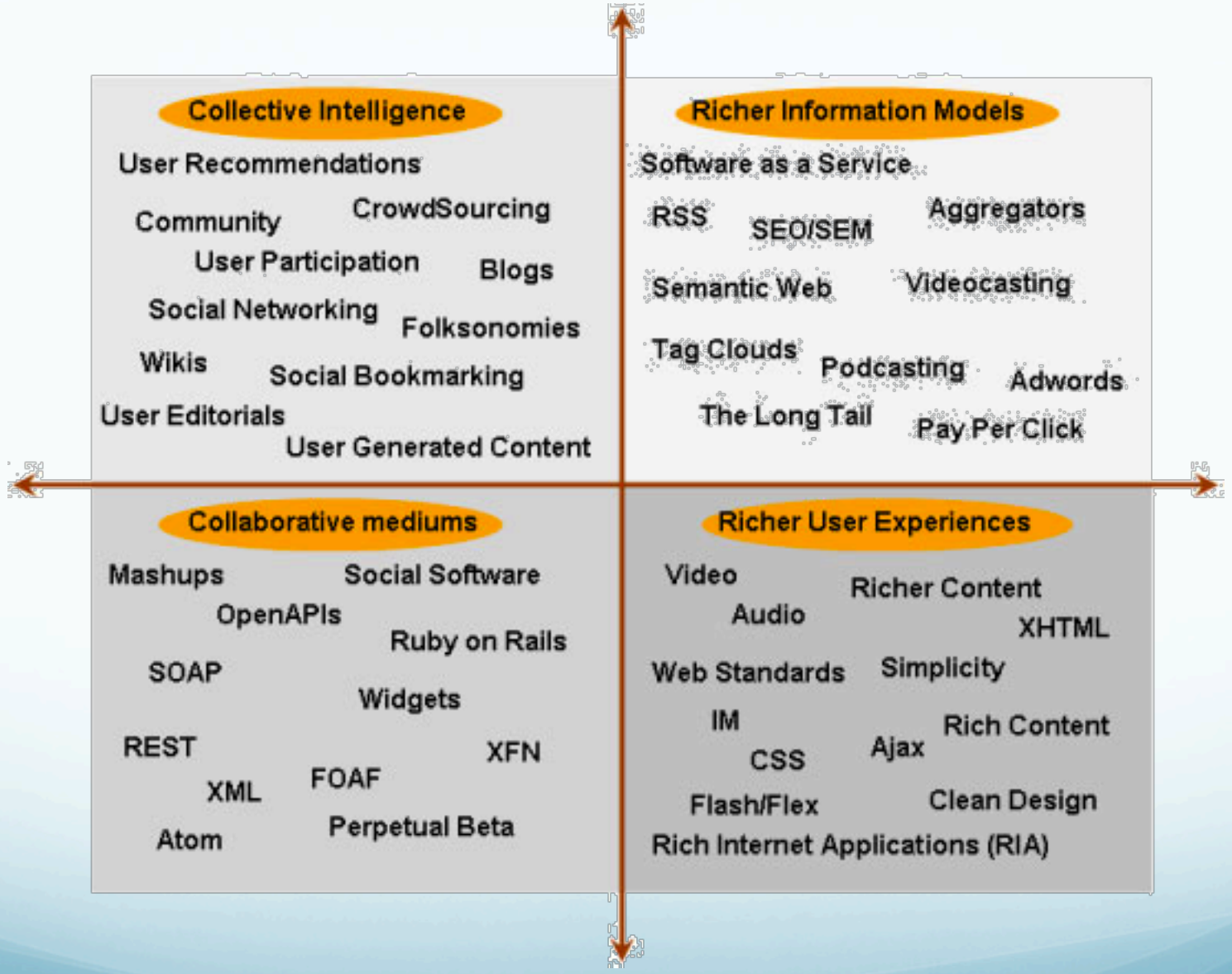
1968

- NLS permet d'initier le processus de co-évolution:
  - Hypertexte pour la répresentation des connaissances
  - Environnement collaboratif graphique et convivial pour visualiser et éditer ces connaissances

Web 2.0: un web participatif

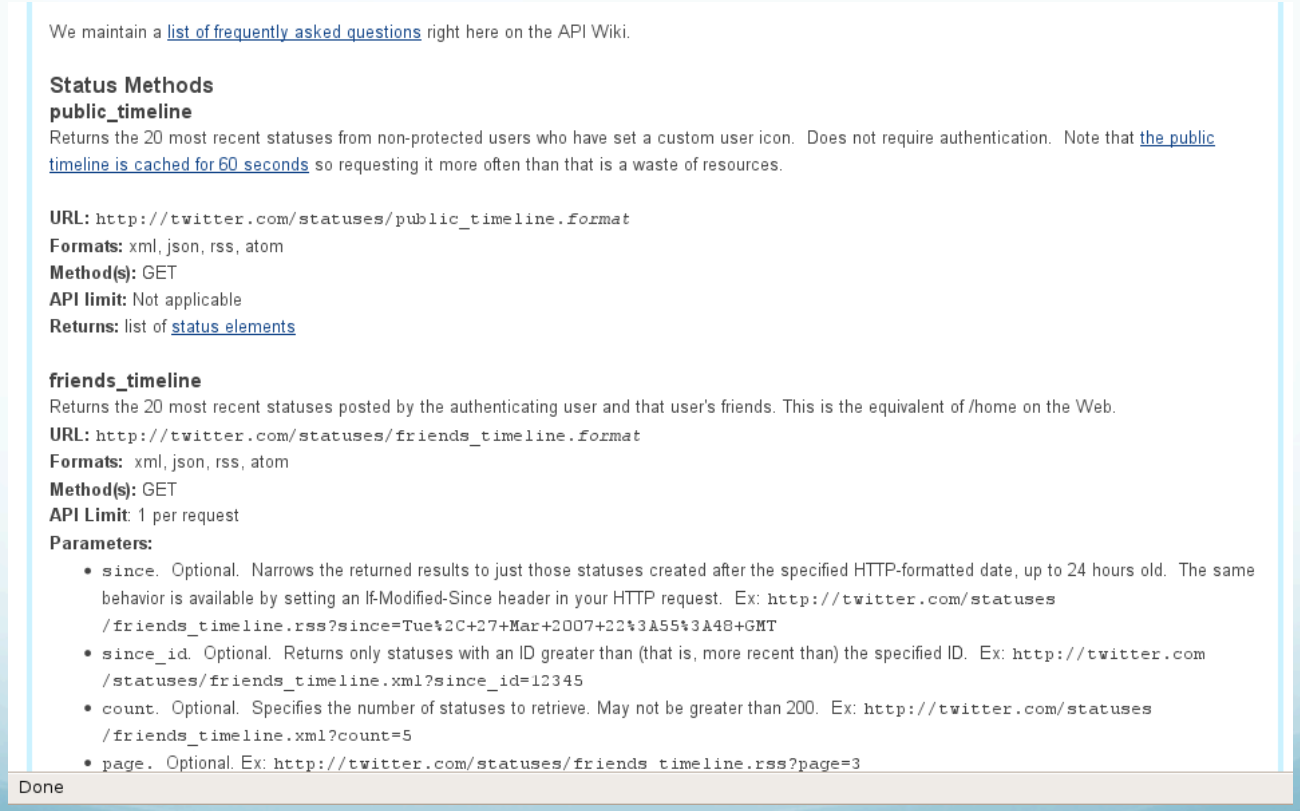
- Folksonomy-
- Rich User Experience-
- User Participation –
- Long tail
- Software as a service –
  - Web API, Mashup
- Mass Participation - Universal web access leads to differentiation of concerns from the traditional internet userbase.




Collective Intelligence
User Recommendations
Community
CrowdSourcing
User Participation
Blogs
Social Networking
Folksonomies
Wikis
Social Bookmarking
User Editorials
User Generated Content
Richer Information Models
Software as a Service
RSS
SEO/SEM
Aggregators
Semantic Web
Videocasting
Tag Clouds
Podcasting
Adwords
The Long Tail
Pay Per Click
Collaborative mediums
Mashups
Social Software
OpenAPIs
Ruby on Rails
SOAP
Widgets
REST
XFN
FOAF
XML
Perpetual Beta
Atom
Richer User Experiences
Video
Richer Content
Audio
XHTML
Web Standards
Simplicity
IM
Rich Content
Ajax
CSS
Flash/Flex
Clean Design
Rich Internet Applications (RIA)

Service Web/Rest

- Exposition des API
- Exemple : Twitter



We maintain a list of frequently asked questions right here on the API Wiki.

Status Methods

public_timeline

Returns the 20 most recent statuses from non-protected users who have set a custom user icon. Does not require authentication. Note that the public timeline is cached for 60 seconds so requesting it more often than that is a waste of resources.

URL: `http://twitter.com/statuses/public_timeline.format`
Formats: xml, json, rss, atom
Method(s): GET
API limit: Not applicable
Returns: list of status elements

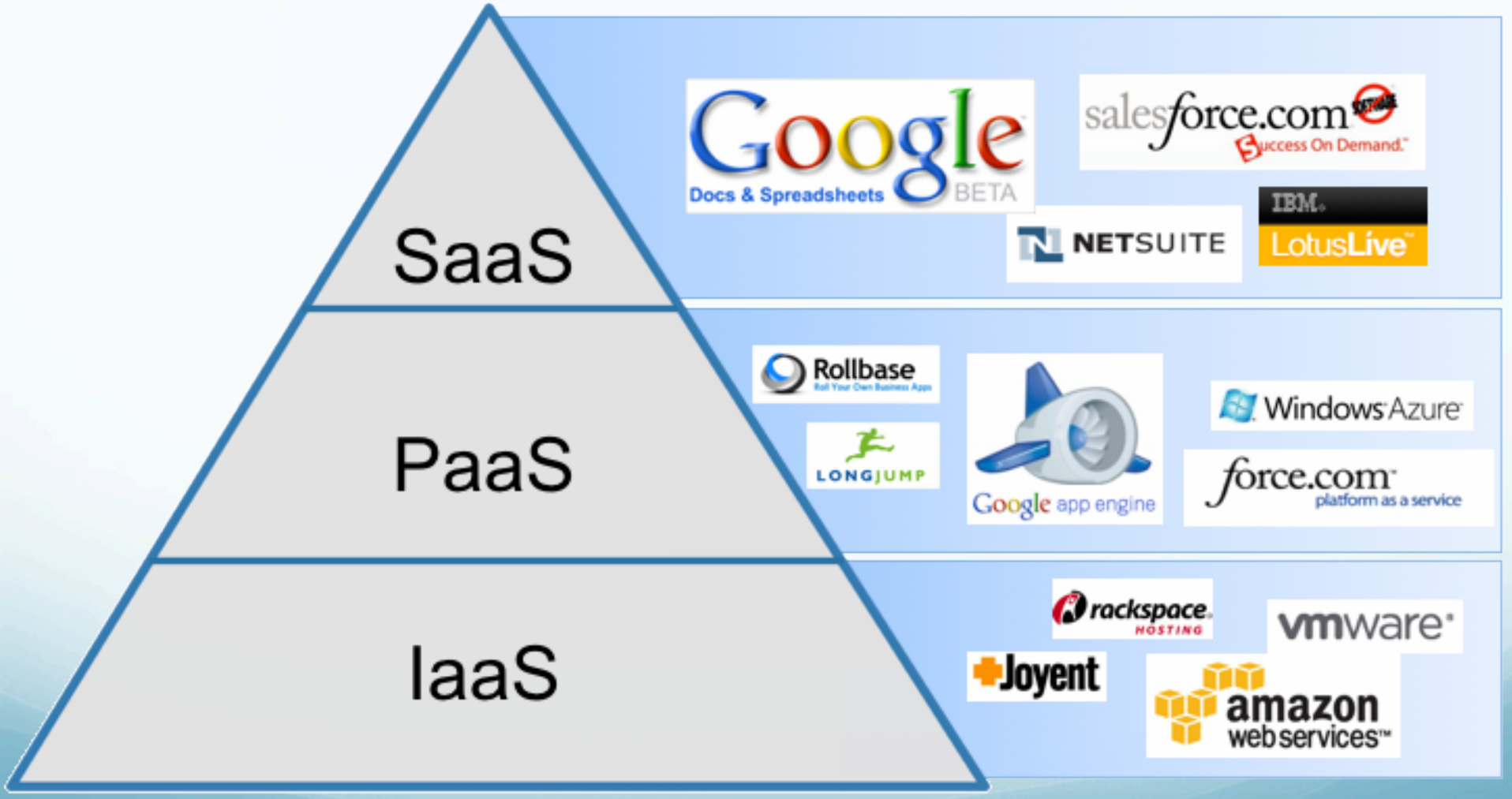
friends_timeline

Returns the 20 most recent statuses posted by the authenticating user and that user's friends. This is the equivalent of /home on the Web.
URL: `http://twitter.com/statuses/friends_timeline.format`
Formats: xml, json, rss, atom
Method(s): GET
API Limit: 1 per request
Parameters:

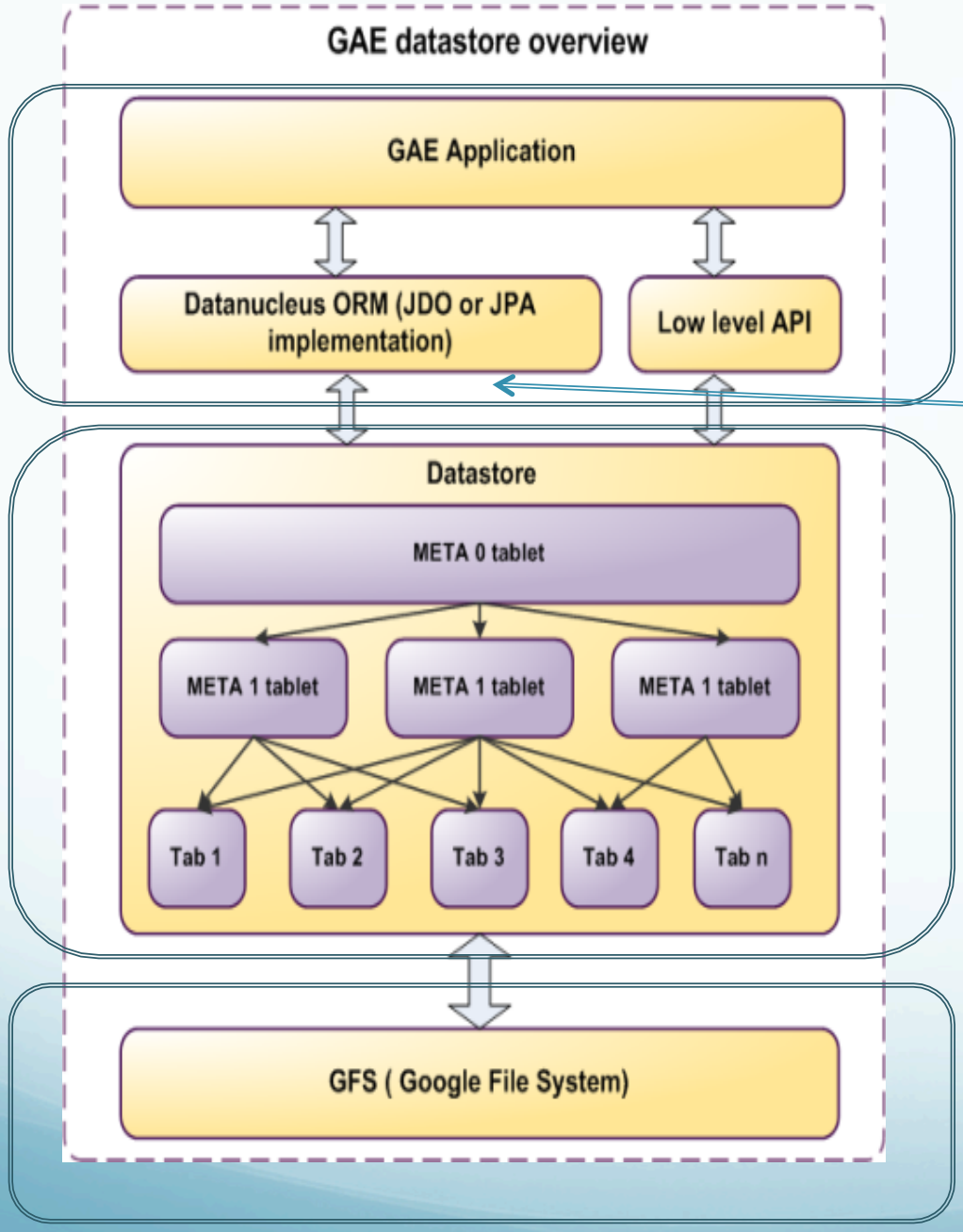
- `since`. Optional. Narrows the returned results to just those statuses created after the specified HTTP-formatted date, up to 24 hours old. The same behavior is available by setting an If-Modified-Since header in your HTTP request. Ex: `http://twitter.com/statuses/friends_timeline.rss?since=Tue%2C+27+Mar+2007+22%3A55%3A48+GMT`
- `since_id`. Optional. Returns only statuses with an ID greater than (that is, more recent than) the specified ID. Ex: `http://twitter.com/statuses/friends_timeline.xml?since_id=12345`
- `count`. Optional. Specifies the number of statuses to retrieve. May not be greater than 200. Ex: `http://twitter.com/statuses/friends_timeline.xml?count=5`
- `page`. Optional. Ex: `http://twitter.com/statuses/friends_timeline.rss?page=3`

Done

Powered by the cloud: *(a)s(a)(Service)



Cloud infrastructure



This run on one VM (that can be relocated for energy optimization

This runs on thousands data nodes and some master nodes: This is really big table…: manage data updates and queries

JDO/JPA: loose coupling with BigTable -

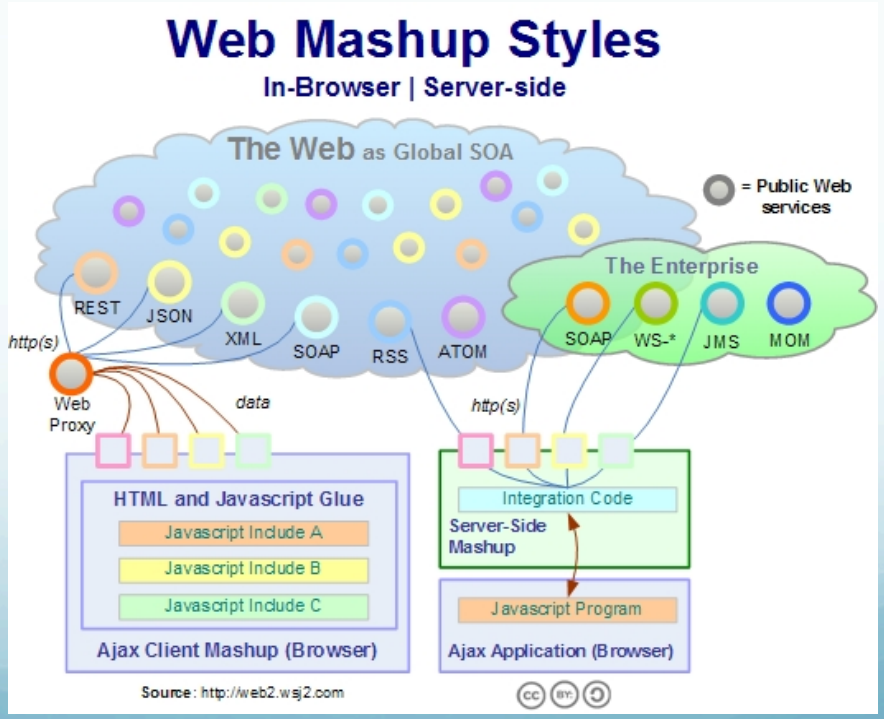
move on dynam

This runs on thousands data nodes with some master nodes: manage file read/write and failures

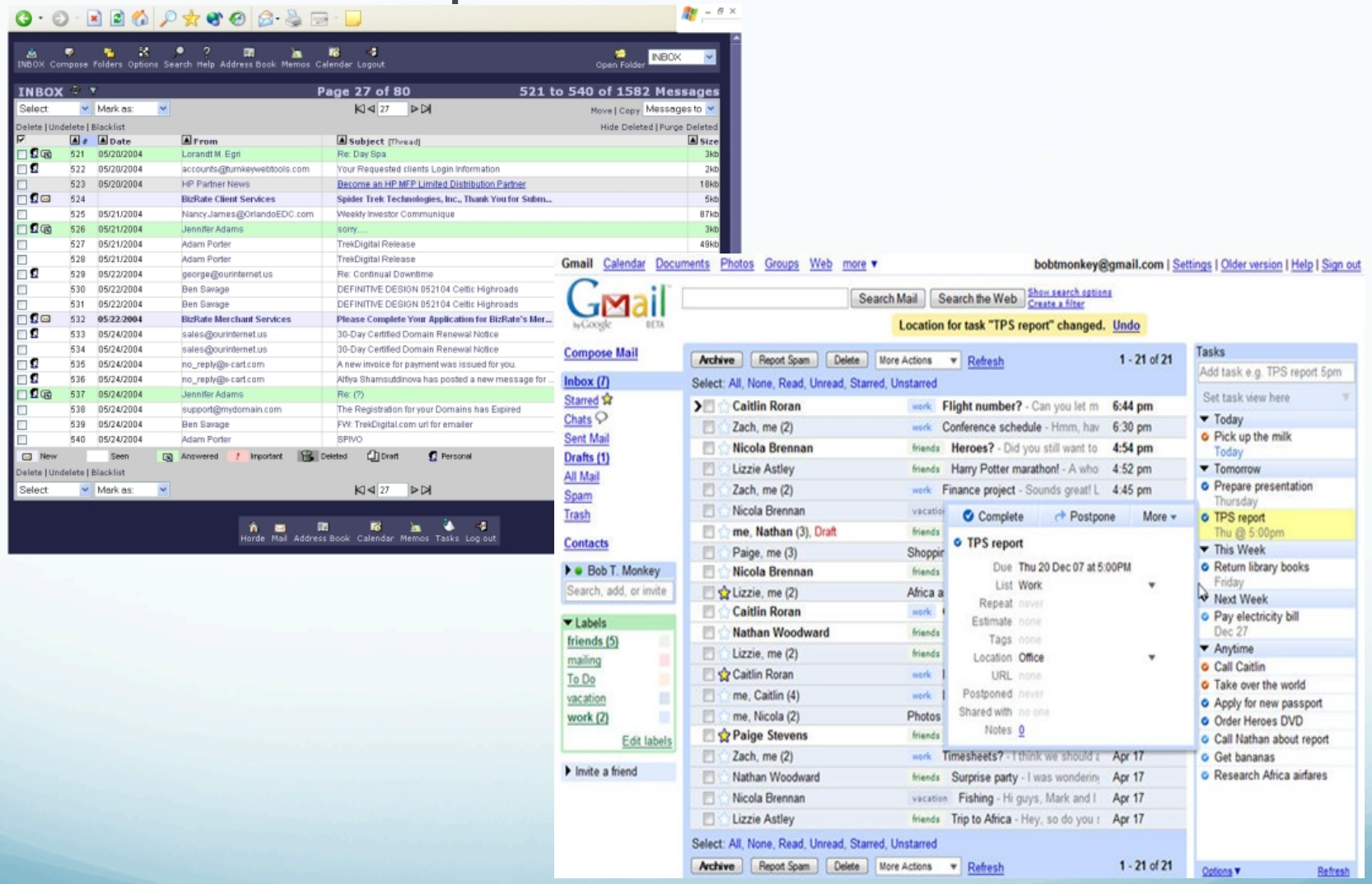
http://blog.xebia.fr/2009/07/16/la-persistance-dans-google-app-engine-partie-une-le-datastore/

Mashups

- Composition d'application Web
- Composition/Filtre de sources de données
- Composition d'applications
- Google Maps + bases de données
- Yahoo Pipes
- ...



Rich User Experience



Problèmes Web 2.0

- Beaucoup de connaissances sont produites,
- Beaucoup de bruits aussi,
- La charge de maintenance et de synthèse de ces connaissances repose uniquement sur les communautés.
- Les machines peuvent nous aider…

Augmenter notre intelligence collective...

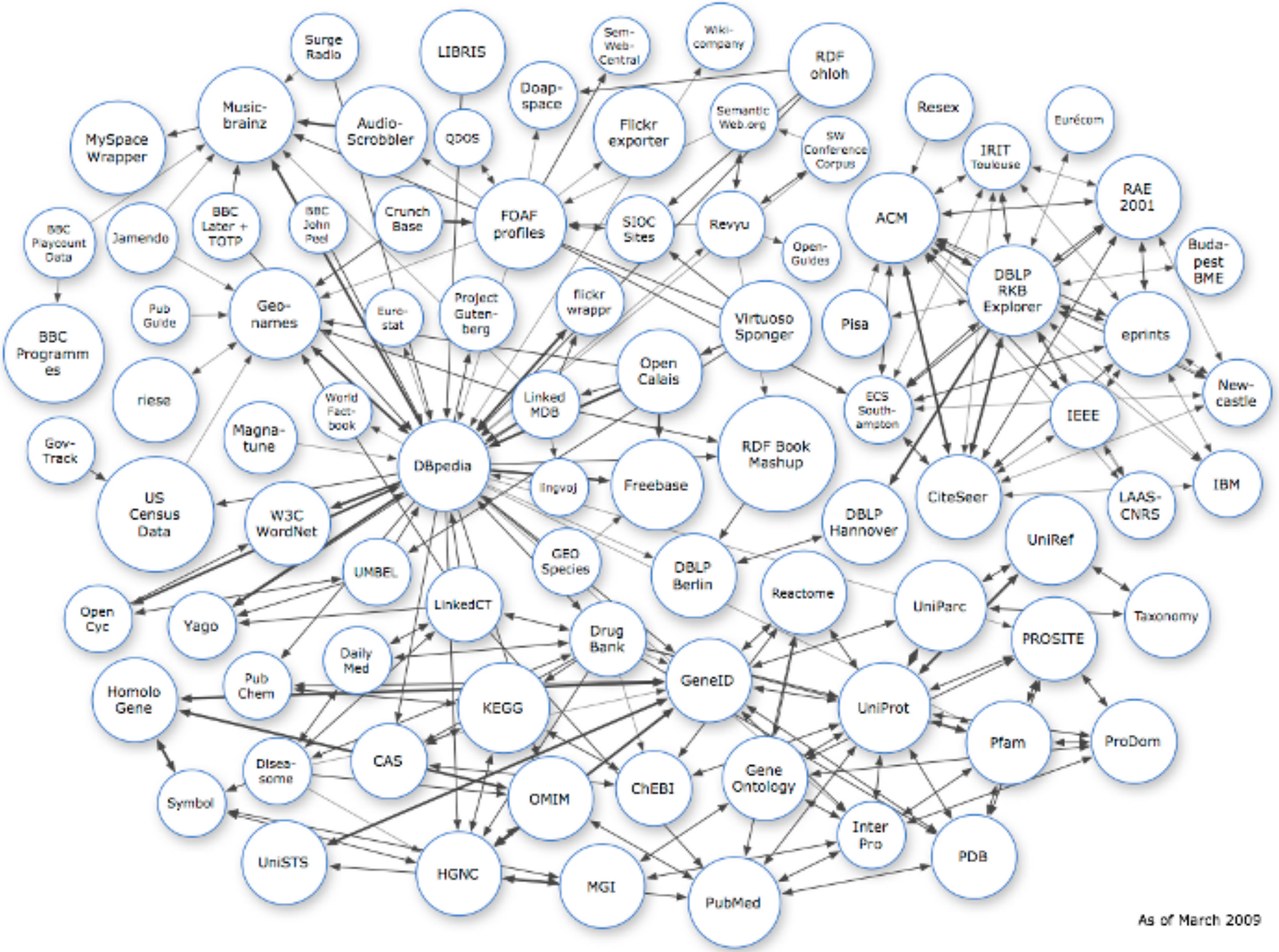
- « **Develop knowledge Collection:** » Codifier et structurer les connaissances.
- « **Use networked improvement communities:** » des communautés pour développer et maintenir les connaissances
- « **Co-evolution of Tools and Work:** » Construire de nouveaux outils, les utiliser, adapter nos pratiques aux outils, adapter les outils aux nouvelles pratiques que nous inventons.
- « **Innovate how we work collectively:** » repenser la manière dont les personnes travaillent ensemble

Co-évolution : Web 3.0

Rendre le Web compréhensible par les machines

- Description formelle des connaissances humaines domaine par domaine. Des groupes d'experts collaborent à l'élaboration d'ontologies
- Les sites webs publient des informations sémantiques ou sont parcourus par des robots pour « peupler » les ontologies
- Les ontologies peuvent être connectées entre elles, permettent le raisonnement formel et donc l'interrogation par les humains et les machines.

- **Engelbart : « Develop knowledge Collection: »** Codifier et structurer les connaissances.


Surge Radio
LIBRIS
Sem-Web-Central
Wiki-company
RDF ohloh
Doap-space
Music-brainz
Audio-Scrobbler
QDOS
Flickr exporter
Semantic Web.org
SW Conference Corpus
MySpace Wrapper
Resex
Eurécom
IRIT Toulouse
RAE 2001
ACM
BBC Playcount Data
Jamendo
BBC Later + TOTP
BBC John Peel
Crunch Base
FOAF profiles
SIOC Sites
Revyu
Open-Guides
Buda-pest BME
DBLP RKB Explorer
Pub Guide
Geo-names
Euro-stat
Project Guten-berg
flickr wrappr
Virtuoso Sponger
Pisa
eprints
BBC Programmes
riese
Open Calais
New-castle
World Fact-book
Linked MDB
ECS South-ampton
IEEE
Magna-tune
Gov-Track
DBpedia
RDF Book Mashup
Ingvoj
Freebase
CiteSeer
LAAS-CNRS
IBM
US Census Data
W3C WordNet
DBLP Hannover
UniRef
UMBEL
GEO Species
DBLP Berlin
Reactome
UniParc
Taxonomy
Open Cyc
Yago
LinkedCT
Drug Bank
PROSITE
Daily Med
Pub Chem
GeneID
Homolo Gene
KEGG
UniProt
Pfam
ProDom
Disea-some
CAS
ChEBI
Gene Ontology
Symbol
OMIM
Inter Pro
UniSTS
HGNC
MGI
PubMed
PDB
As of March 2009

Search results

From Wikipedia, the free encyclopedia

capitals europe Search

Content pages Multimedia Help and Project pages Everything Advanced

You may create the page "Capitals europe", but consider checking the search results below

European capital (redirect from **European Capitals**)
European capital may refer to: the **capitals** of the **European** countries , see List of **capitals** by continent the Capital of the **European** Union **...**
362 B (36 words) - 14:07, 28 April 2009

List of national **capitals** (redirect from **Europe capitals**)
See also List of historical **capitals** of China . c. c | c | none Unrecognized, self-declared country. d. d | d | none Unofficial; see Politics **...**
16 KB (570 words) - 07:56, 13 October 2009

Location of **European** Union institutions
The governing institutions of the **European** Union (EU) are not concentrated in a **...** referred to as the joint **capitals** of **Europe**, for example **...**
57 KB (8,073 words) - 22:23, 27 August 2009

Vienna
It is the 9th largest city by population in the **European** Union . **...** premier Volleyball organisations, and the Vienna **Capitals** (Ice Hockey). **...**
51 KB (6,314 words) - 12:36, 9 October 2009

WIKIPEDIA
The Free Encyclopedia

navigation
- Main page
- Contents
- Featured content
- Current events
- Random article

search
capitals europe
Go Search

interaction
- About Wikipedia
- Community portal
- Recent changes
- Contact Wikipedia
- Donate to Wikipedia
- Help

toolbox
- Upload file
- Special pages

Google

capitals europe

Rechercher

Recherche avancée
Préférences

Rechercher dans : Web Pages francophones Pages : France

Web Résultats **1 - 10** sur un total d'environ **4 610 000** pour **capitals europe** (**0,25** secondes)

Capital Cities of **Europe** - [Traduire cette page]

23 Aug 2009 ... Links to official or near official sites of the states and **capital** cities of **Europe**. Include figures of the population of the **capitals**.
www.nationsonline.org/.../**capitals_europe**.htm - En cache - Pages similaires -

Capitales-Europe (Union **Européenne**)

capitales des pays d'**Europe** membres de l'Union **européenne**.
www.liensutiles.org/villes**europe**.htm - En cache - Pages similaires -

europe capital cities map and information page - [Traduire cette page]

... North America | South America | World Atlas | WIN $100 here dot **europe capital** city ... Moscow is the **capital** city of the entire Russian Federation. ...
www.worldatlas.com/.../**europe**/eucaps.htm - En cache - Pages similaires -

Résultats d'images pour **capitals europe** - Signaler des images

Trouver les **capitales** d'**Europe**

WolframAlpha™ computational... knowledge engine

capitals europe

Assuming Europe | Use Europe with Russia and Turkey or Europe with Russia instead

Input interpretation: *Mathematica* form

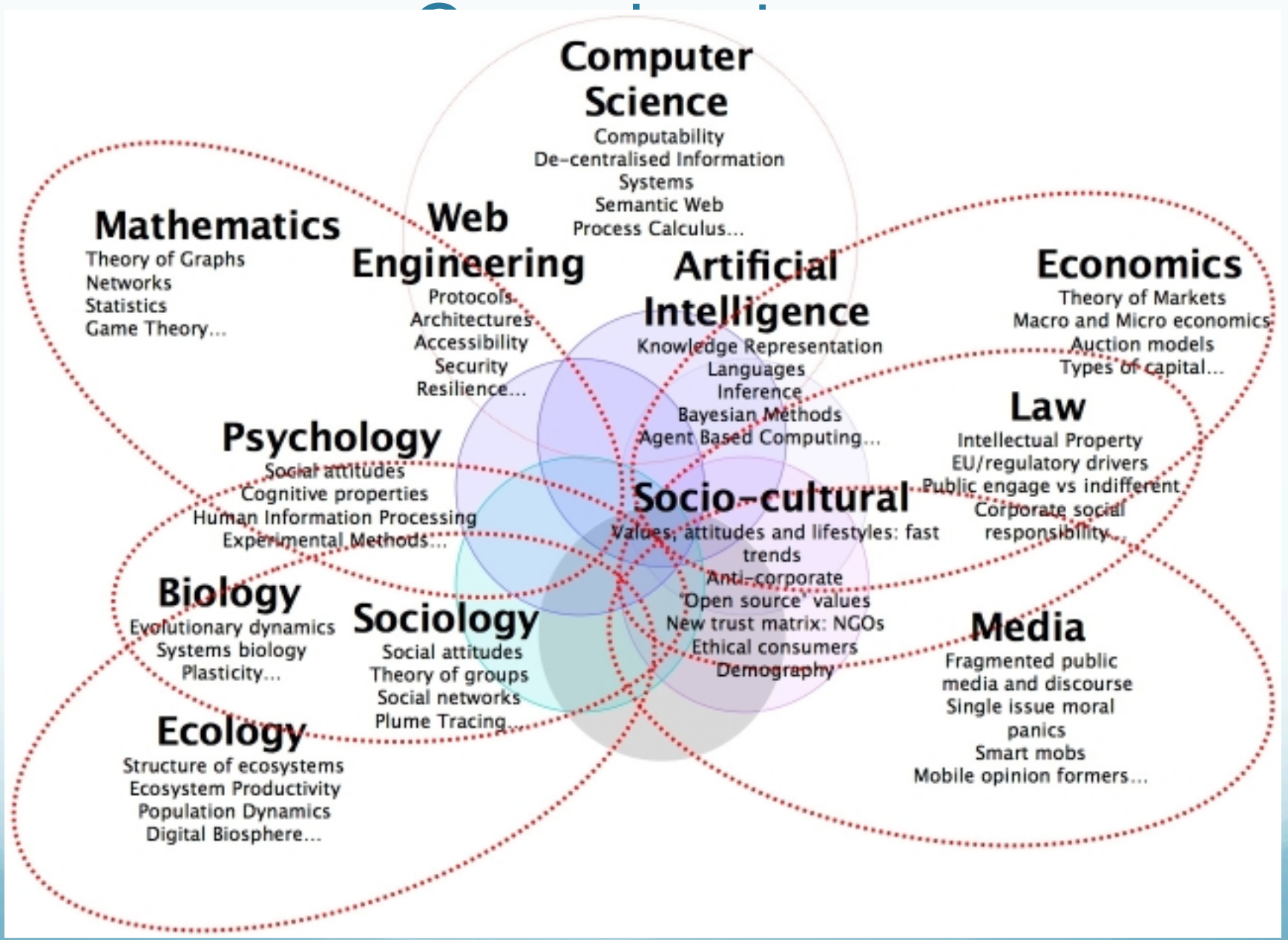
Europe | capital city

Table: Less | More

Albania	Tirana
Andorra	Andorra la Vella
Austria	Vienna
Belarus	Minsk
Belgium	Brussels
Bosnia and Herzegovina	Sarajevo, Federacija Bosna i Hercegovina
Croatia	Zagreb, Grad Zagreb
Cyprus	Nicosia, Government controlled area
Czech Republic	Prague

Conclusion...

- Web is alive and grows...
  - It is in the same time a poison and a drug (B. Stiegler)
  - How its evolution can produce a better world?


Computer Science
Computability
De-centralised Information Systems
Semantic Web
Process Calculus...
Mathematics
Theory of Graphs
Networks
Statistics
Game Theory...
Web Engineering
Protocols
Architectures
Accessibility
Security
Resilience...
Artificial Intelligence
Knowledge Representation
Languages
Inference
Bayesian Methods
Agent Based Computing...
Economics
Theory of Markets
Macro and Micro economics
Auction models
Types of capital...
Law
Intellectual Property
EU/regulatory drivers
Public engage vs indifferent
Corporate social responsibility...
Psychology
Social attitudes
Cognitive properties
Human Information Processing
Experimental Methods...
Socio-cultural
Values, attitudes and lifestyles: fast trends
Anti-corporate
'Open source' values
New trust matrix: NGOs
Ethical consumers
Demography
Biology
Evolutionary dynamics
Systems biology
Plasticity...
Sociology
Social attitudes
Theory of groups
Social networks
Plume Tracing...
Media
Fragmented public media and discourse
Single issue moral panics
Smart mobs
Mobile opinion formers...
Ecology
Structure of ecosystems
Ecosystem Productivity
Population Dynamics
Digital Biosphere...

Conclusion

- But Web is also a piece of engineering
  - On top of internet (IETF)
  - With many specific standards (W3C)
  - Largely powered by cloud computing
- Start studying it...

W3C

Standards
Participate
Membership
About W3C
Member Home

STANDARDS

W3C standards define an **Open Web Platform** for application development that has the unprecedented potential to enable developers to build rich interactive experiences, powered by vast data stores, that are available on any device. Although the boundaries of the platform continue to evolve, industry leaders speak nearly in unison about how HTML5 will be the cornerstone for this platform. But the full strength of the platform relies on many more technologies that W3C and its partners are creating, including CSS, SVG, WOFF, the Semantic Web stack, XML, and a variety of APIs.

W3C develops these technical specifications and guidelines through a process designed to maximize consensus about the content of a technical report, to ensure high technical and editorial quality, and to earn endorsement by W3C and the broader community.

If you are learning about Web technology, you may wish to start with the introduction below, and follow links for greater detail.

Web Design and Applications

Web Design and Applications involve the standards for building and Rendering Web pages, including HTML, CSS, SVG, Ajax, and other technologies for Web Applications ("WebApps"). This section also includes information on how to make pages accessible to people with disabilities (WCAG), to internationalize them, and make them work on mobile devices.

Web Architecture

Web Architecture focuses on the foundation technologies and principles which sustain the Web, including URIs and HTTP.

Semantic Web

In addition to the classic "Web of documents" W3C is helping to build a technology stack to support a "Web of data," the sort of data you find in databases. The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term "Semantic Web" refers to W3C's vision of the Web of linked data. Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS.

XML Technology

XML Technologies including XML, XML Namespaces, XML Schema, XSLT, Efficient XML Interchange (EXI), and other related standards.

Web of Services

Objectives

- Imagine and Write a web applications **that works** using open data nantes
- Write on the cloud… (Google app Engine)

Developping Web Applications with Servlets

Quels sont les problèmes ?

- Notion d'application
- Clients multiples et inconnus
  - Concurrence d'accès
  - Sécurité
  - Passage à l'échelle
- Protocole HTTP (stateless)
  - Identification des clients
  - Maintien des sessions

Web Frameworks

Česky
Deutsch
Español
فارسی
한국어
Bahasa Indonesia
Italiano
Magyar
日本語
Русский
Svenska
中文

Edit links

- List of Ajax frameworks
- Web application framework
- Comparison of web application frameworks

A

- AccDC
- ActiveVFP
- Agavi
- AIDA/Web
- Ajax framework
- Ametys CMS
- Ample SDK
- Apache Shale
- Apache Sling
- Apache Tapestry
- Apache Wicket
- AppFlower
- AppFuse
- Template:Application frameworks
- Apusic OperaMasks
- ASP.NET
- ASP.NET MVC Framework
- ASP.NET Razor view engine
- ATL Server
- Axiom CMS

B

- Barracuda (Java)
- Base One Foundation Component Library

F cont.

- Fluid UI
- Framework One
- FuelPHP
- Fusebox (programming)

G

- Geomajas
- Google Apps Script
- Grails (framework)
- Gurupa

H

- Hamlets
- Hammerkit
- Hazaar MVC
- Hobo (software)
- Horde (software)
- Humax Raiya Programming

I

- IBM WebSphere Application Server
- IBM WebSphere Application Server Community Edition
- ICEfaces
- ItsNat

J

- Java Platform, Enterprise Edition
- JavaServer Faces
- Javeline platform

P cont.

- PHPClasses repository
- PhpCodeGenie
- PHPixie
- PHPRunner
- PhpWebSite
- Play Framework
- Plone (software)
- PRADO Framework
- ProcessWire
- Project Zero

Q

- Qcodo
- Quinoa Framework

R

- Ramaze
- Reasonable Server Faces
- RichFaces
- RIFE
- Rivista
- RNA Framework
- Ruby on Rails
- Run BASIC

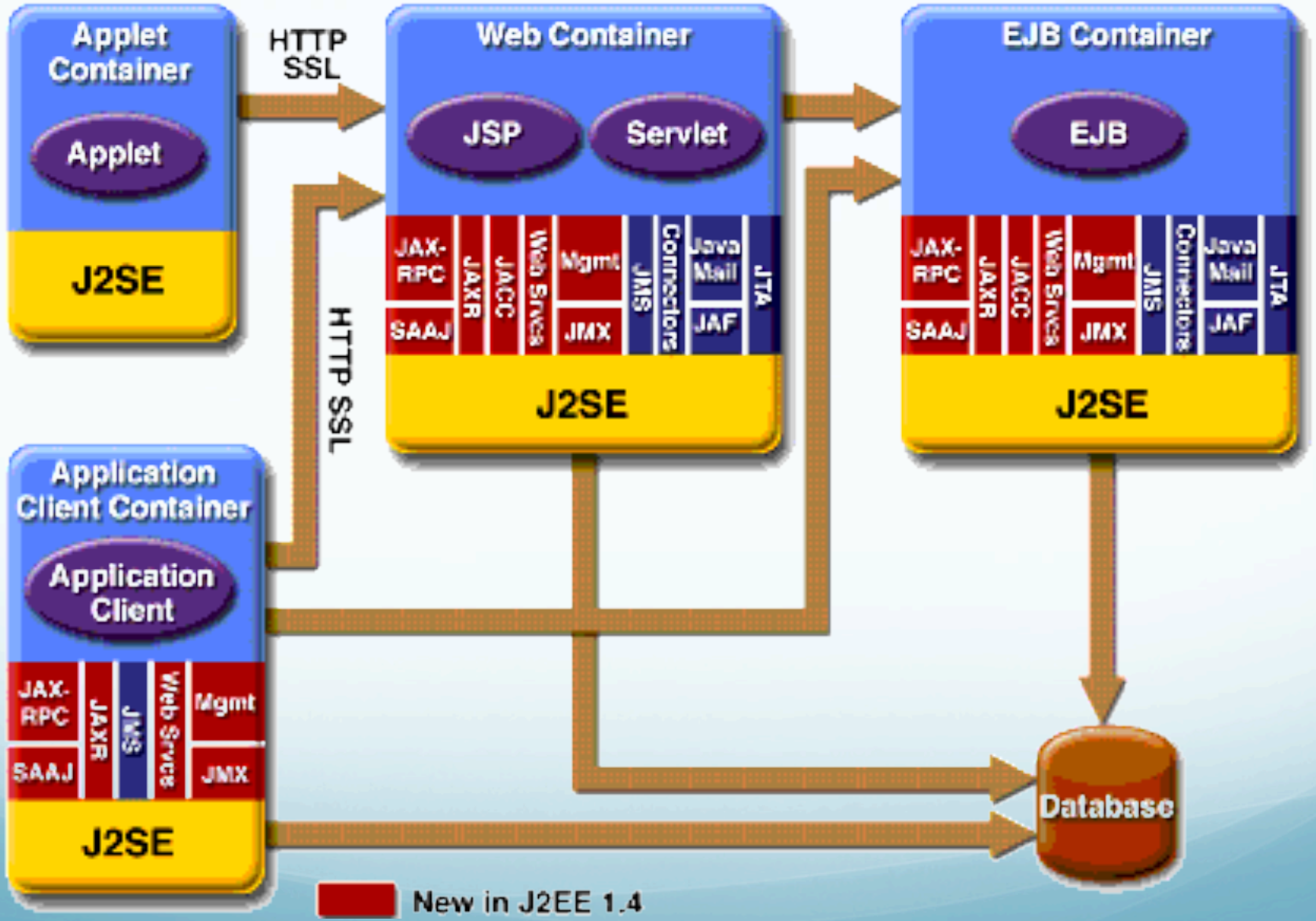
S

- Scalatra
- Seagull PHP Framework
- Seaside (software)

Serveur d'application JEE

- Un modèle à composant
  - Component programming
- Fournit un ensemble de services pour l'écriture d'application distribuées en Java
  - Nommage
  - Transactions
  - Sécurité
  - Conte

Pourquoi Java ?

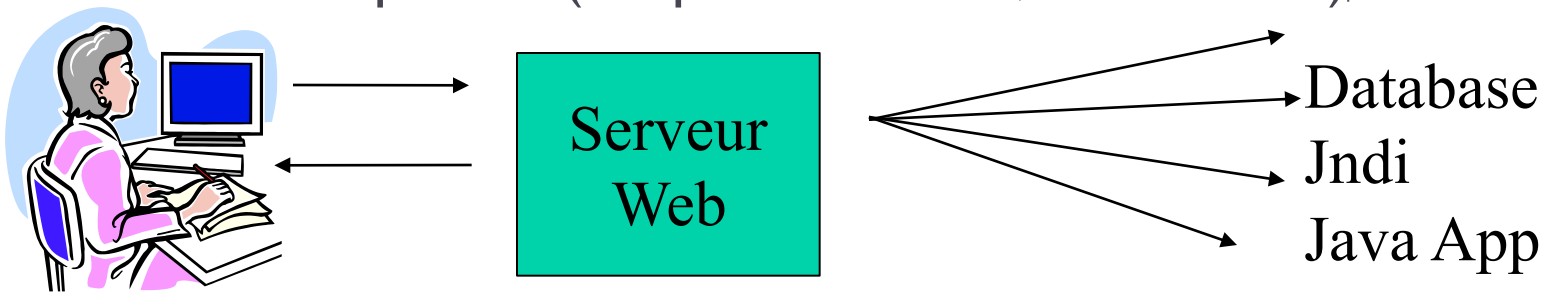

Applet Container
Applet
J2SE
HTTP
SSL
Web Container
JSP
Servlet
JAX-RPC
SAAJ
JAXR
JACC
Web Srvcs
Mgmt
JMX
JMS
Connectors
Java Mail
JAF
JTA
J2SE
EJB Container
EJB
JAX-RPC
SAAJ
JAXR
JACC
Web Srvcs
Mgmt
JMX
JMS
Connectors
Java Mail
JAF
JTA
J2SE
HTTP SSL
Application Client Container
Application Client
JAX-RPC
SAAJ
JAXR
JMS
Web Srvcs
Mgmt
JMX
J2SE
Database
New in J2EE 1.4

Les Servlets

- Permettre la programmation d'applications coté serveur
- Permettre l'extension d'un serveur Web en java
- Permettre la construction d'appli Web dynamique
- Equivalent des CGI en java

Fonctionnement d'un servlet

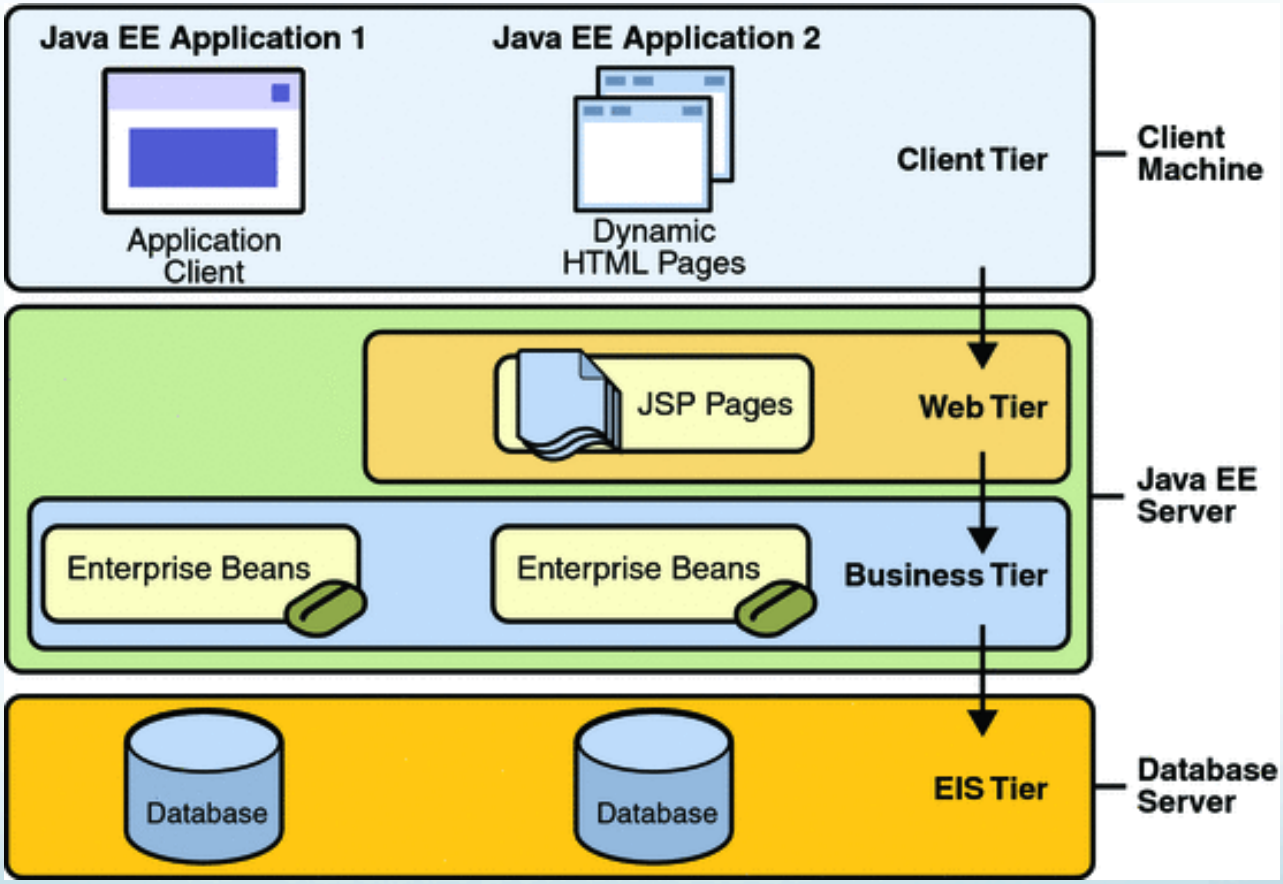
- Un servlet lit les données envoyées par un client Web (transmises par le serveur)
  - Données explicites (Formulaire)
  - Données implicites (Request Header)
- Il génère un résultat
- Il envoit le résultat au client
  - Données explicites (Page HTML)
  - Données implicites (Response Header, Status code)



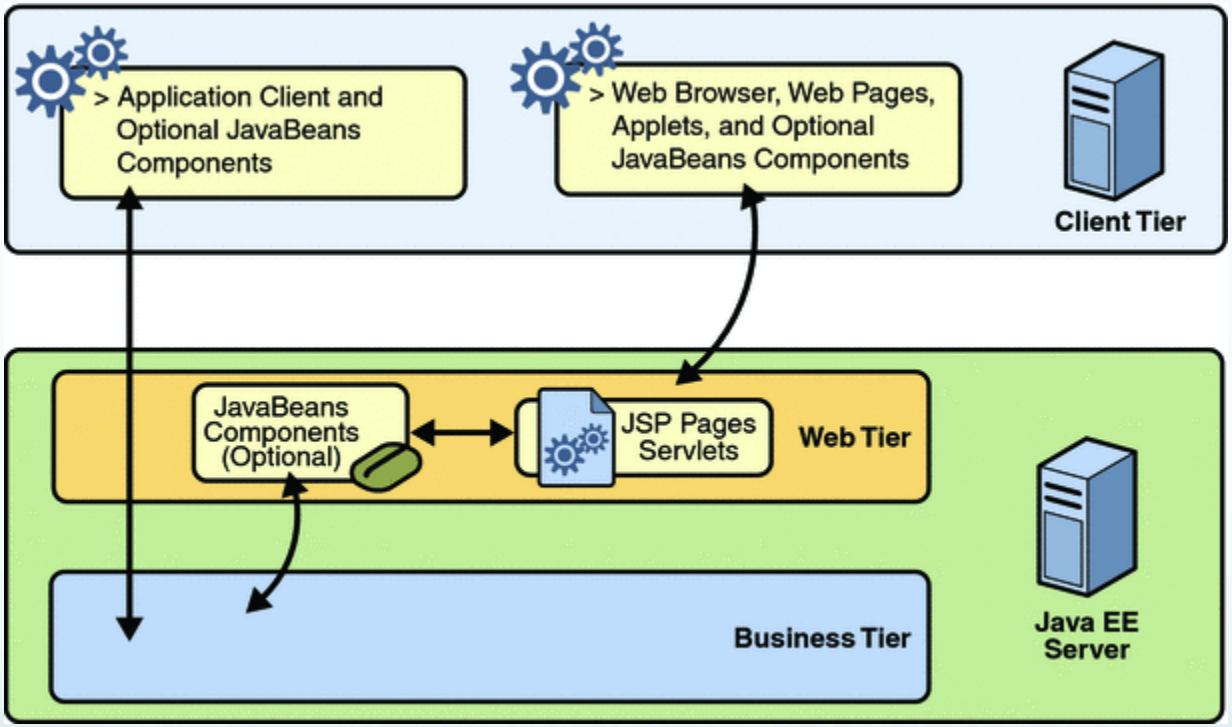
Avantage des servlets

- Efficacité
  - Utilisent des threads plutôt que des processus comme les CGI
- Pratique
  - Librairie très développée (moins que PHP mais plus consistante et standardisée)
- Portable
  - Déployable quelque soit le serveur (ou presque)
- Sécurisé
  - Fonctionne dans une machine virtuelle (plus maitrisable)
- Pas cher
  - Nombreux serveurs gratuits

L’architecture



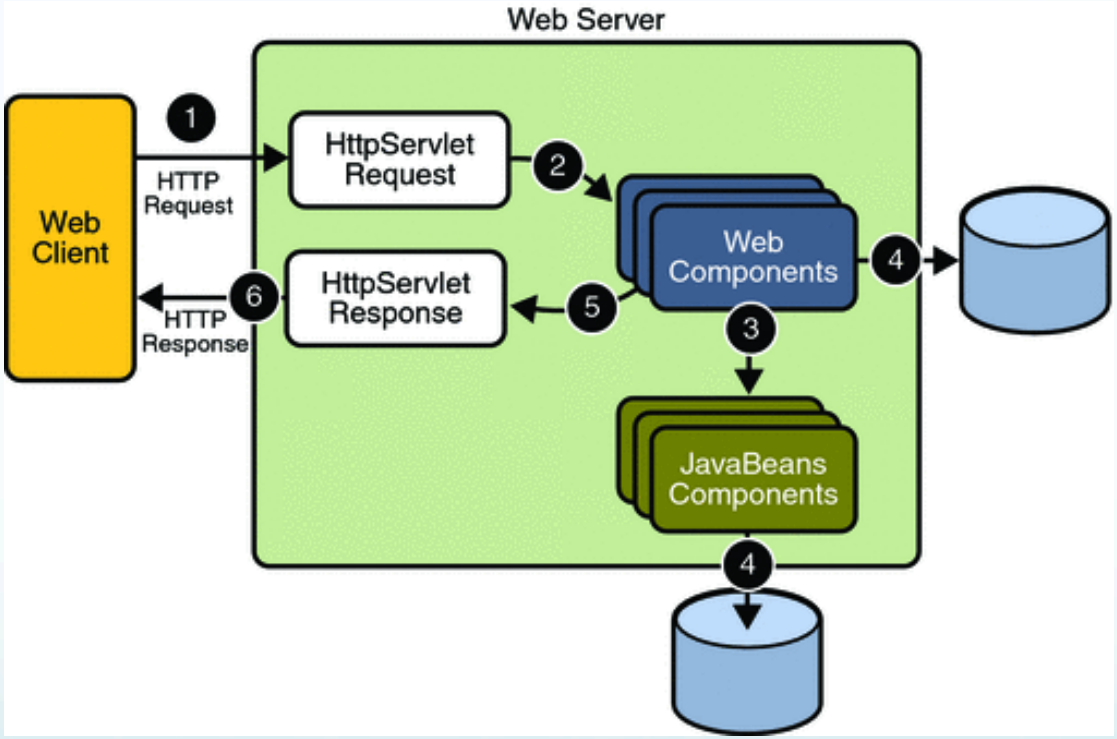
La partie Web



Les containers de servlet

- Les servlets sont des composants
  - Ils s'exécutent dans un container
  - Ils doivent se conformer à une interface prédéfinie
  - C'est le container qui décide de leur instantiation
- Les containers
  - Environnement d'exécution des servlets
  - Tomcat (jakarta.apache.org/tomcat)
  - Jetty
  - Weblogic
  - Glassfish

L’exécution d’un servlet

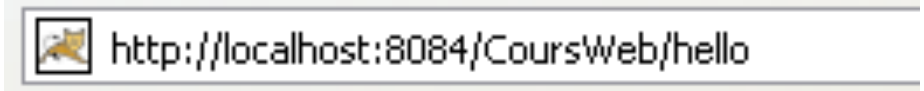


Un premier servlet

- Le servlet HelloWorld
- Un servlet surcharge la classe HttpServlet
- Il doit implanter au moins une des méthodes
  - doGet
  - doPost
  - doPut
  - doDelete

Le servlet HelloWorld

```
public class HelloWorld extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
    throws ServletException, IOException {
        PrintWriter out=response.getWriter();
        out.println("Hello World");
    }

}
```



Hello World

Le fichier de configuration

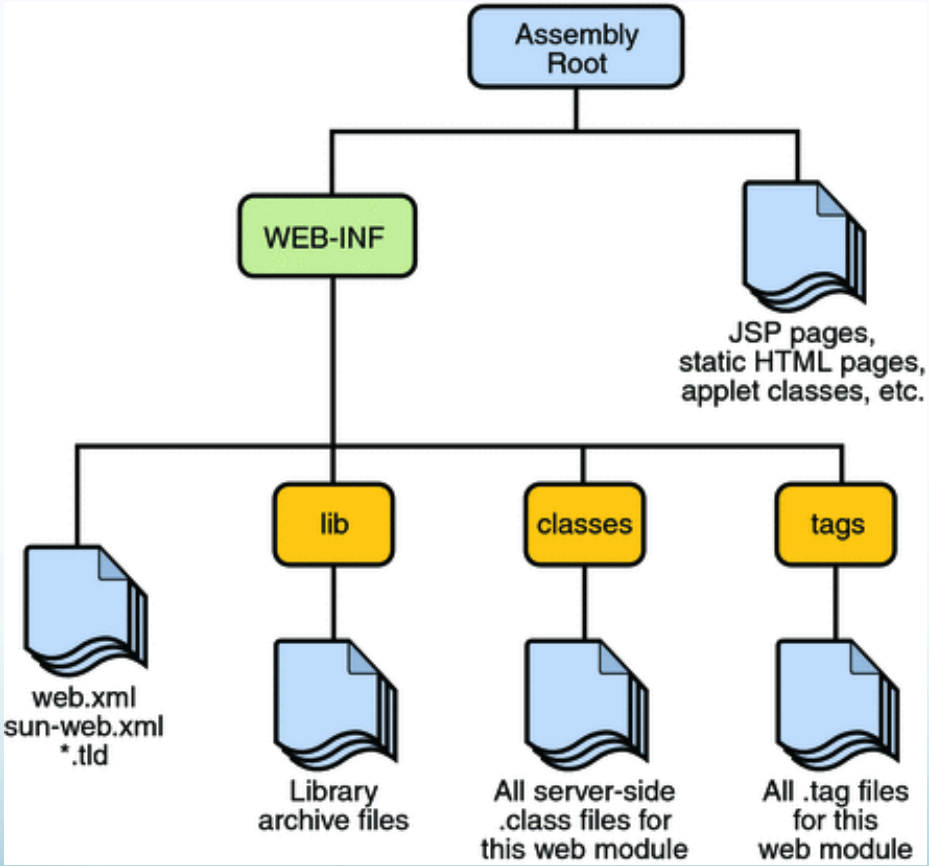
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi=
    <servlet>
        <servlet-name>HelloWorld</servlet-name>
        <servlet-class>exemple.HelloWorld</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <welcome-file-list>
        <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
    </web-app>
```

Servlet 3.0 : annotations


```
@WebServlet(name="FirstServlet", urlPatterns={"/first"})
public class FirstServlet extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
```

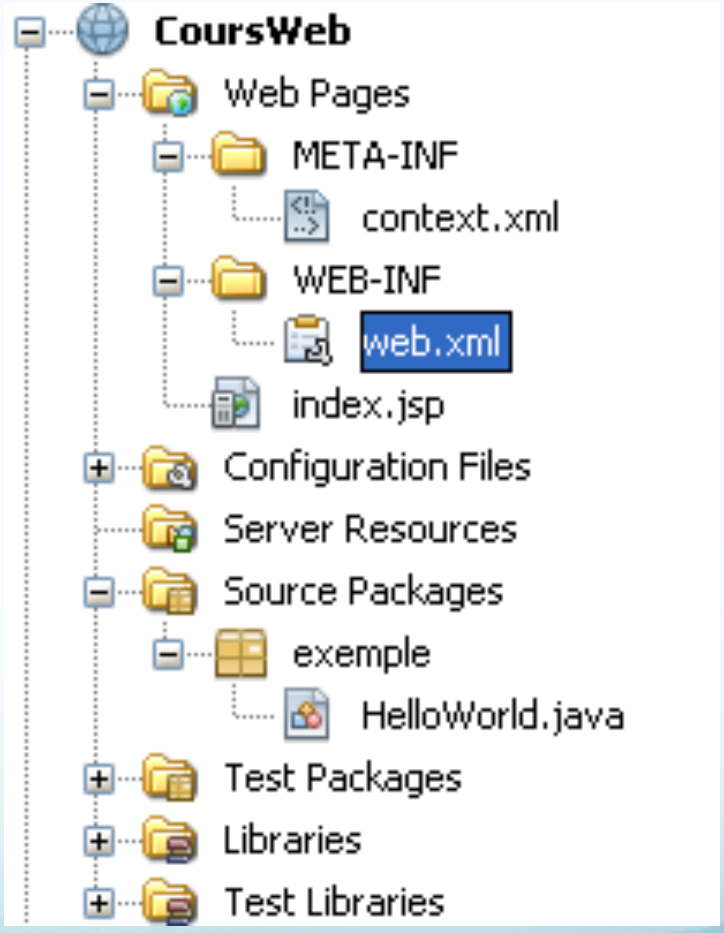
Un module Web

- Déployable et exécutable dans un serveur Web



L'application Web

- Une archive war
- Déployée dans le serveur Web
- web.xml = description de l'application



Structure d'une application

- Package déployable
- Composants Web : Servlet et JSP
- Ressources statiques (images)
- Classes java (helper)
- Librairies
- Descripteurs de déploiement (web.xml)

Les paramètres

- Les paramètres d'une requête sont accessibles dans l'objet (HttpServletRequest)request
  - request.getParameter("paramname");
- Il est possible de récupérer tous les noms des paramètres
  - request.getParameterNames();

Exemples paramètres

```
protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet ParamServlet</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet ParamServlet at " + request.getContextPath ()
        out.println("param1="+request.getParameter("param1"));
        out.println("param2="+request.getParameter("param2"));
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

Servlet ParamServlet at /CoursWeb

param1=test param2=parametre

Exemple POST

```
<html>
  <head>
    <title></title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body>
  <form action="myForm" method="POST">
  Le parametre : <input type="text" name="nom" value="" size="20" />
  <input type="submit" value="submit" />
</form>
  </body>
</html>
```

Exemple POST (suite)

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet MyForm</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("Hello "+request.getParameter("nom"));
        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}
```

```
<servlet>
    <servlet-name>MyForm</servlet-name>
    <servlet-class>exemple.MyForm</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>MyForm</servlet-name>
    <url-pattern>/myForm</url-pattern>
</servlet-mapping>
```

Hello rototo

La classe HttpServlet

L'interface HttpServletRequest

- Fournit les informations sur la requête du client au serveur
- Principales méthodes (pour l'instant):
  - String getParameter(String name)
  - Enumeration getParameterNames()
    - Retourne une énumération de tous les noms de paramètres
  - String[] getParameterValues()
    - Retourne un tableau contenant toutes les valeurs des paramètres
  - String getHeader(String name)
  - Enumeration getHeaderNames()
    - Retourne une énumération de tous les noms des propriétés du header
  - String[] getHeaderValues()
    - Retourne un tableau de toutes les valeurs du header

Les request headers

```
for (Enumeration e=request.getHeaderNames();e.hasMoreElements();){
    String name=(String) e.nextElement();
    out.println(name+" = "+request.getHeader(name)+"<br>");
}
```

host = localhost:8084
user-agent = Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.8.1.11) Gecko/20071127 Firefox/2.0.0.11
accept = text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
accept-language = fr,fr-fr;q=0.8,en-us;q=0.5,en;q=0.3
accept-encoding = gzip,deflate
accept-charset = ISO-8859-1,utf-8;q=0.7,*;q=0.7
keep-alive = 300
connection = keep-alive
cookie = moncookie="mon password"; JSESSIONID=AB4130236FA5B9E054232D468ED9F070

L'interface HttpServletResponse

- Objet fournissant les services pour envoyer la réponse à un client.
- Les principales méthodes sont:
  - java.io.PrintWriter getWriter()
    - Pour récupérer un printWriter qui permet d'envoyer du texte au client
  - public void setContentLength(int len)
    - Positionne le Content-Length Header
  - public void setContentType(java.lang.String type)
    - Position le Content-Type header (exemple text/html)
  - public void sendError(int sc, java.lang.String msg) throws java.io.IOException
    - Envoi un message d'erreur au client (cf code dans l'API)
  - public void setHeader(java.lang.String name, java.lang.String value)

web.xml

- Le fichier web.xml donne des instructions sur le déploiement du servlet dans le container
- web-app
  - La description d'une Web Application
- Servlet
  - La relation entre le nom du servlet et la classe qui l'implante
- Servlet-mapping
  - La relation entre le nom du servlet et l'url qui permet d'y accéder

Les paramètres d'initialisation

- Les servlets peuvent avoir des paramètres d'initialisation.
- Ces paramètres peuvent être changés sans avoir à recompiler l'application
  - <init-param>
    - <param-name>testValue</param-name>
    - <param-value>12</param-value>
    - <description>une valeur quelconque</description>
  - </init-param>
- Ces paramètres peuvent être utilisés avec la méthode
  - javax.servlet.getInitParameter()

Exemple

```
out.println("<h1>Servlet InitServlet at " + request.getContextPath () + "</h1>");
out.println("valeur = "+this.getInitParameter("valeur1"));
out.println("nombre = "+this.getInitParameter("nombre"));
```

```
<servlet>
    <servlet-name>InitServlet</servlet-name>
    <servlet-class>exemple.InitServlet</servlet-class>
    <init-param>
        <param-name>valeur1</param-name>
        <param-value>PAR DEFAUT</param-value>
    </init-param>
    <init-param>
        <param-name>nombre</param-name>
        <param-value>12</param-value>
    </init-param>
</servlet>
```

Servlet InitServlet at /CoursWeb

valeur = PAR DEFAUT nombre = 12

Le servlet mapping

- Permet de construire la relation entre un servlet et son URL
  - <servlet-mapping>
    - <servlet-name>Test</servlet-name>
    - <url-pattern>/Test/*</url-pattern>
  - </servlet-mapping>
- Tous les urls correspondant à http://host:port/webapp/url-pattern déclencherons l'exécution du servlet
- Exemples
  - /*.do
  - /Test
  - /cours/test/*

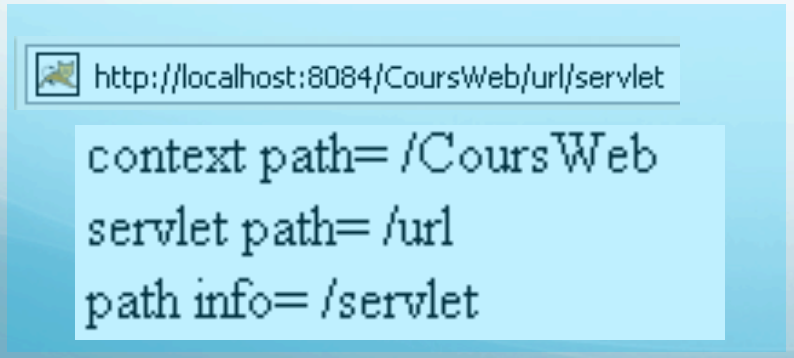
Les éléments du chemin de requête

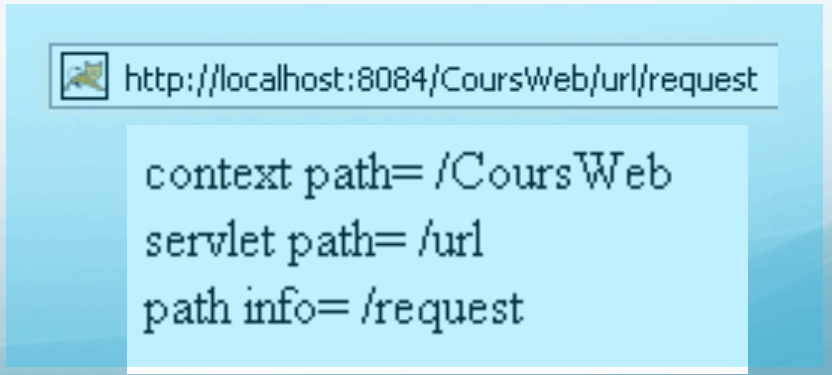
- ContextPath : le chemin du contexte de déploiement
- ServletPath : la section du chemin qui a déclenché le mapping
- PathInfo : la partie de la requête qui n'est ni le ContextPath ni le ServletPath
  - Request.getContextPath()
  - Request.getServletPath()
  - Request.getPathInfo()

Exemples

```
out.println("<h1>Servlet URLServlet at " + request.getContextPath () + "</h1>");
out.println("context path= "+request.getContextPath()+"<br>");
out.println("servlet path= "+request.getServletPath()+"<br>");
out.println("path info= "+request.getPathInfo()+"<br>");
```

```
<servlet-mapping>
    <servlet-name>URLServlet</servlet-name>
    <url-pattern>/url/*</url-pattern>
</servlet-mapping>
```





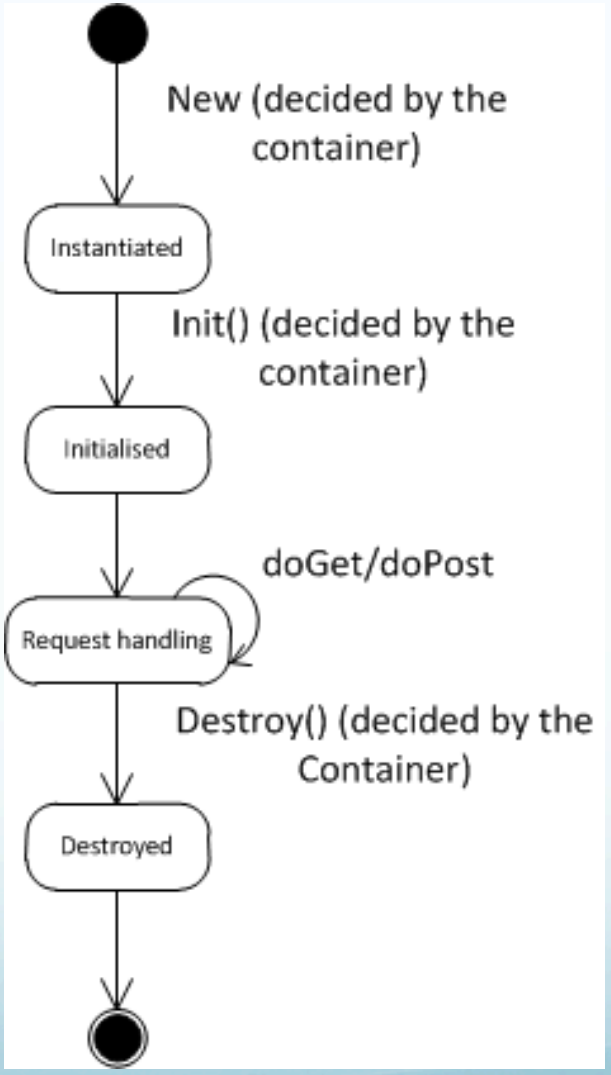
Autre propriétés

- Pour le mapping mime
  - <mime-mapping>
- Pour les fichiers à charger quand un nom de répertoire est donné
  - <welcome-file-list>
- Pour les pages d'erreur
  - <error-page>
  - Pour chaque code d'erreur on peut fixer une page spécifique
- Il y en a d'autres pour
  - La sécurité
  - Les taglibs
  - Les références aux ressources utilisés par les servlets

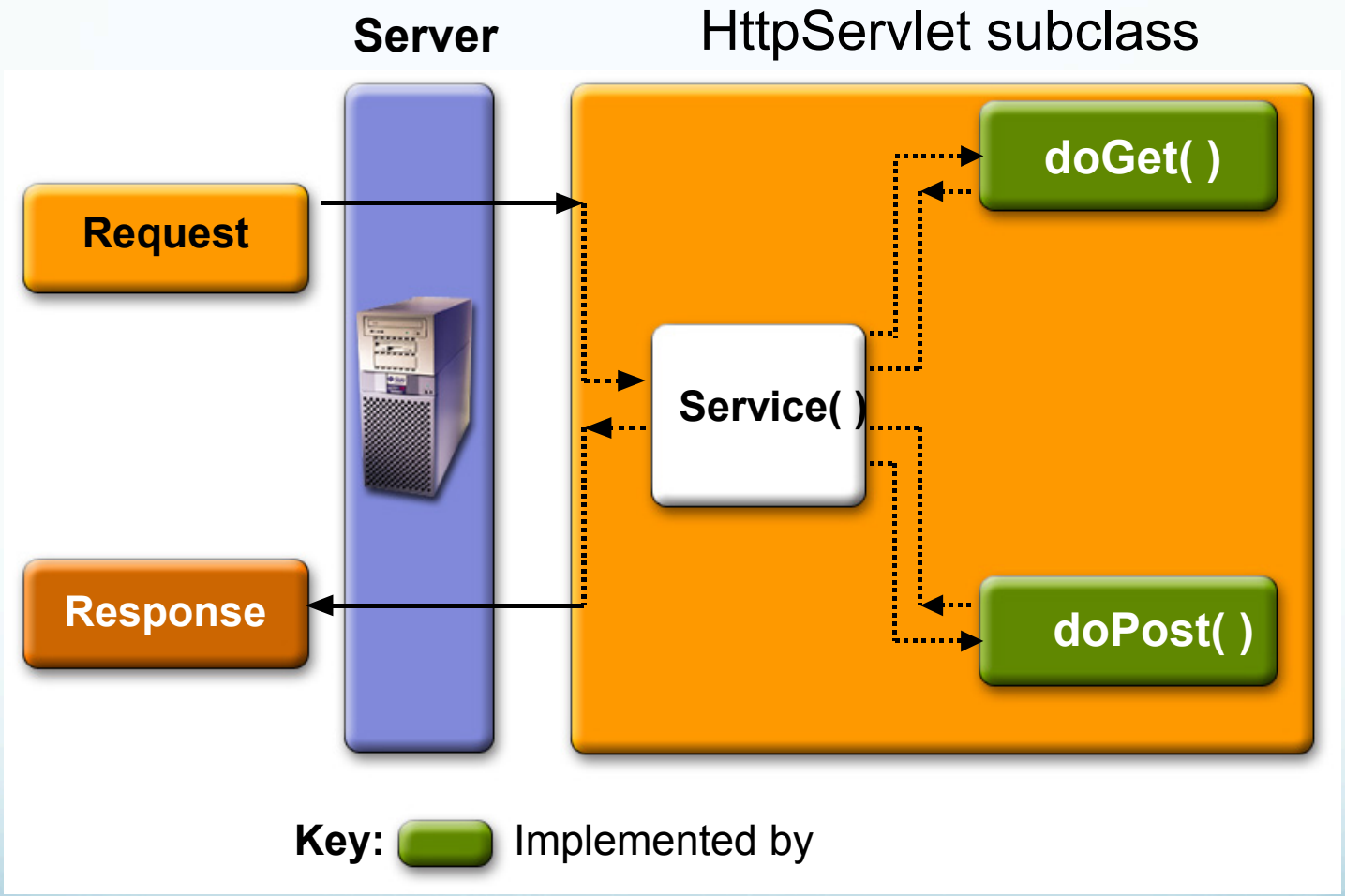
Le cycle de vie d'un servlet

- Tous les servlets ont le même cycle de vie
  - Le container charge le servlet
    - La fonction init() est appelée (initialisation de variables, connection à des bases de données)
  - Le servlet répond aux requêtes des clients
  - Le contaire détruit le servlet
    - La fonction destroy() est appelée (fermeture des connections)

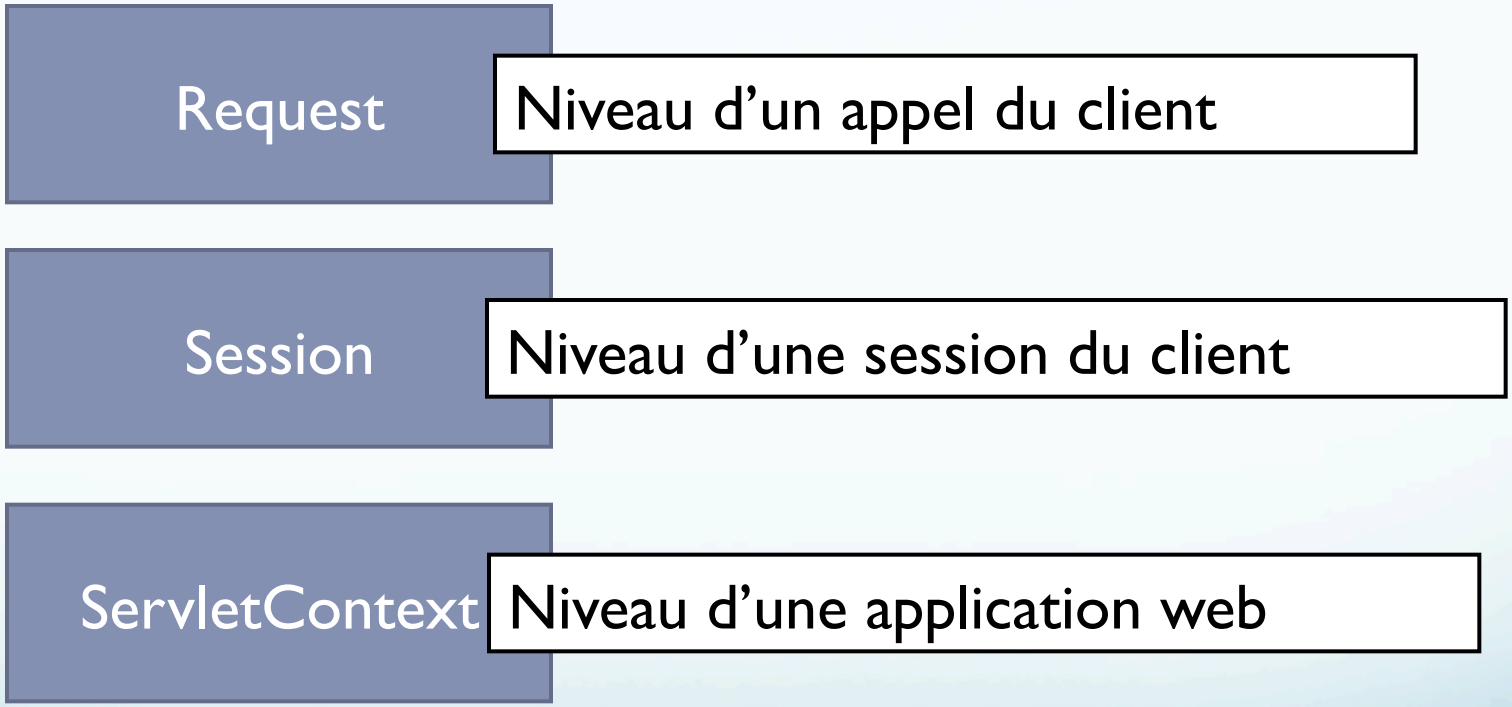
Servlet Life Cycle



doGet et doPost



Visibilité des variables



Les Sessions

- Comment maintenir l'état d'un client au cours d'une série de requêtes d'un même utilisateur pendant un temps donné ?
- HTTP est stateless
- Comment identifier le client ?
- Où mettre les données de la session ?

Servlet et sessions

- La gestion des sessions utilise les techniques classiques
  - URL rewriting
  - Cookies
  - Champs cachés dans les formulaires
- L'API HttpServlet fournit des fonctions pour gérer les sessions
- L'implantation est transparente pour l'utilisateur

Session & HttpServletRequest

- C'est le paramètre request qui maintient les informations sur la session
- Les méthodes sont
  - HttpSession request.getSession(boolean flag)
    - Flag=true : retourne l'objet session courant ou en crée un s'il n'y en a pas.
    - Flag=false : Retourne l'objet session courant ou null s'il n'y en a pas
  - isRequestedSessionIdValid()
    - Vrai si l'id de la session est valide dans le contexte courant
  - isRequestedSessionIdFromCookie()
    - Vrai si l'id de la session vient d'un cookie
  - isRequestedSessionIdFromURL()
    - Vrai si l'id de la session vient d'un URL

Création de la session

```
out.println("<h1>Servlet NewSessionServlet at " + request.getContextPath () + "</h1>");
HttpSession sess=request.getSession();
out.println("Session is new ? "+sess.isNew()+"<br>");
out.println("Session id ? "+sess.getId()+"<br>");
sess.setAttribute("user", "john");
```

Requête dans la même session

```
out.println("<h1>Servlet OldSessionServlet at " + request.getContextPath () + "</h1>");
HttpSession sess=request.getSession();
out.println("Session is new ? "+sess.isNew()+"<br>");
out.println("Session id ? "+sess.getId()+"<br>");
out.println("User ?"+sess.getAttribute("user")+"<br>");
```

Servlet OldSessionServlet at /CoursWeb

Session is new ? false
Session id ? 4077AFBCF13A4C2F94B821207A1147D6
User ?john

L'objet HttpSession

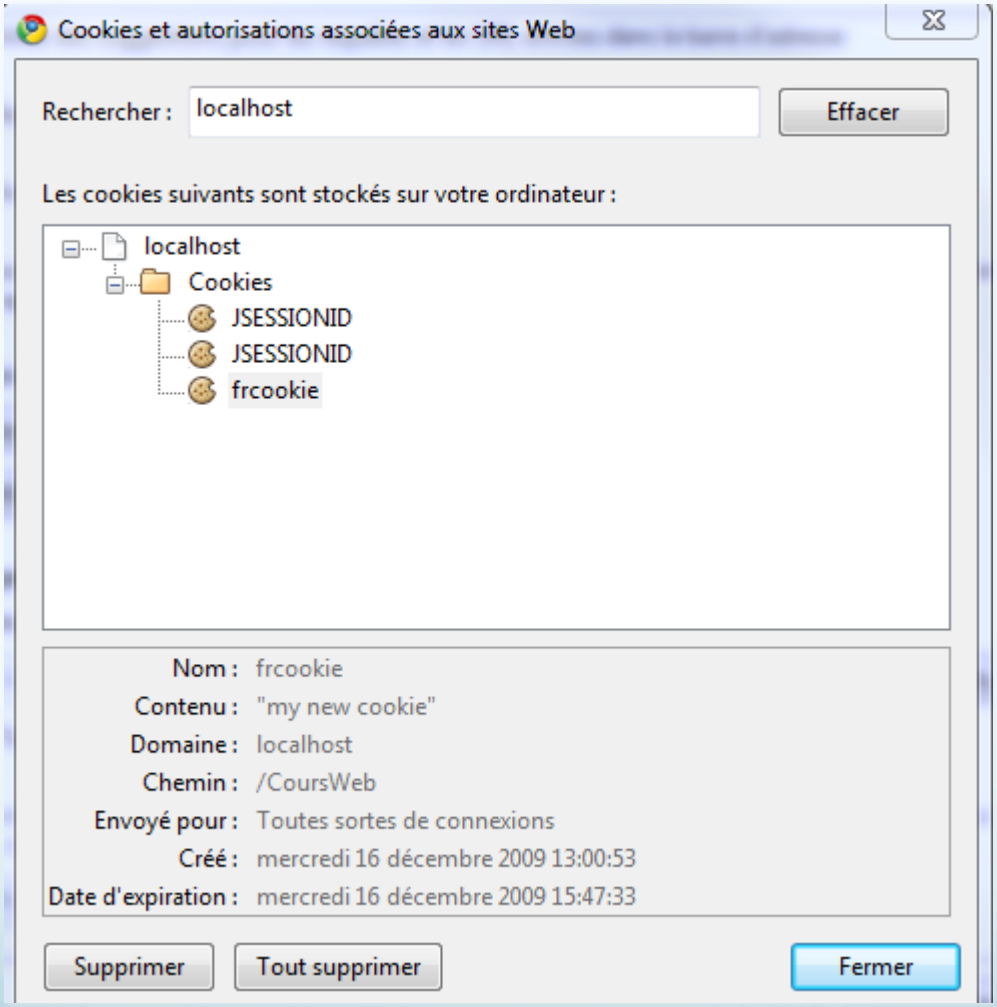
- L'objet HttpSession permet de conserver des valeurs entre des appels d'un client
- Les principales méthodes sont
  - Object getAttribute(String name)
    - Récupère la valeur de l'attribut par son nom
  - Enumeration getAttributeNames()
    - Retourne la liste de tous les attributs
  - Void setAttribute(String name, Object value)
    - Ajoute un attribut à la session
  - String getId()
    - Retourne l'identifiant de la session
  - Void invalidate()
    - Invalide la session
  - Long getCreationTime()
  - Long getLastAccessedTime()
  - Long getMaxInactiveInterval()
  - ServletContext getServletContext()

Cookies

- Cookies can be created and used to store data on the client

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        HttpSession session=request.getSession();
        Cookie cook=new Cookie("frcookie","my new cookie");
        cook.setMaxAge(10000);// in seconds the validity of the cookie
        response.addCookie(cook);
        out.println("cookie written");
    } finally {
        out.close();
    }
}
```

Cookies


Cookies et autorisations associées aux sites Web
Rechercher :
localhost
Effacer
Les cookies suivants sont stockés sur votre ordinateur :
localhost
Cookies
JSESSIONID
JSESSIONID
frcookie
Nom : frcookie
Contenu : "my new cookie"
Domaine : localhost
Chemin : /CoursWeb
Envoyé pour : Toutes sortes de connexions
Créé : mercredi 16 décembre 2009 13:00:53
Date d'expiration : mercredi 16 décembre 2009 15:47:33
Supprimer
Tout supprimer
Fermer

Getting cookies values

```
Cookie[] cookies=request.getCookies();
for (int i = 0; i < cookies.length; i++) {
    Cookie cookie = cookies[i];
    out.println(cookie.getName()+" "+cookie.getValue()+" <br>");
}
```

JSESSIONID 75cb8a814fa566c4d87a1ccb028c
frcookie my new cookie

Configuration des sessions

- Il est possible de fixer la durée d'une session par application (en minutes)
  - <session-config>
    - <session-timeout>30</session-timeout>
  - </session-config>

Terminer une session

```
out.println("<h1>Servlet TerminateSession at " + request.getContextPath () + "</h1>");
out.println("Date de création"+new Date(request.getSession().getCreationTime()));
request.getSession().invalidate();
```

Date de créationTue Feb 05 15:10:31 CET 2008 goodbye

Le Servlet Context

- Correspond au contexte de l'application Web
- Maintien des données pour toute l'application
- Fournit le request dispatcher
- Données d'initialisations pour toute l'application
- Log
- Un ServletContext par application et par JVM

Le ServletContext

- Objet permettant au servlet de communiquer avec le servlet container
- Obtenu avec
  - Servlet.getServletContext()
- Les principales méthodes de ServletContext
  - Object getAttribute(String name)
    - Retourne un attribut du contexte
  - Void setAttribute(String name, Object value)
    - Ajoute ou remplace un objet dans le contexte
  - String getInitParameter(String name)
    - Retourne un paramètre d'initialisation de l'application
  - Void Log(String msg)
    - Ajoute un message dans le log file du servlet container

Le Servlet Context

```
out.println("Context 1 "+getServletContext().getInitParameter("context1"));
out.println("Context 2 "+getServletContext().getInitParameter("context2"));
```

```
<context-param>
    <param-name>context1</param-name>
    <param-value>ma valeur 1</param-value>
</context-param>
<context-param>
    <param-name>context2</param-name>
    <param-value>ma valeur 2</param-value>
</context-param>
```

Servlet WebContextServlet

Context 1 ma valeur 1 Context 2 ma valeur 2

Le RequestDispatcher

- L'objet RequestDispatcher est utilisé pour
  - Transférer la requête à un autre « servlet »
  - Inclure la réponse d'un autre "servlet"
- Pour obtenir le request dispatcher

```
RequestDispatcher rd=getServletContext().getRequestDispatcher("/index.jsp");
rd.forward(request, response);
```

```
String path = "/raisins.jsp?orderno=5";
RequestDispatcher rd = context.getRequestDispatcher(path);
rd.include(request, response);
```

Dispatch : Forward

- La requête peut être transférée
  - rd.forward(request,response);
  - Une utilisation classique consiste à avoir un servlet contrôleur qui transmet les commandes à des servlets spécialisés
  - Attention :
    - le servlet appelant ne doit pas écrire dans la réponse avant de forwarder (exception)
    - Si il tente d'écrire après, ses « outputs » ne sont pas pris en compte.
  - Un « forward est bien un délégation de la requête à un autre « servlet »

Exemple d'utilisation

```
RequestDispatcher rd;
String path=request.getPathInfo();
if (path.equals("/test")) {
    rd=getServletContext().getRequestDispatcher("/index.jsp");
} else {
    rd=getServletContext().getRequestDispatcher("/error.jsp");
}
rd.forward(request, response);
out.println("</body>");
out.println("</html>");
```

Démonstration !

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Servlet1</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet Servlet1 at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");
          out.flush();
        RequestDispatcher rd=this.getServletContext().getRequestDispatcher("/Servlet2");
        rd.forward(request, response);

        out.println("<h1>Servlet Servlet1 at " + request.getContextPath () + "</h1>");
        System.out.println("forwarded");

    } finally {
        out.close();
    }
}
```

Dispatch : include

- La reponse peut être incluse dans la réponse en cours de construction
- rd.include(request,response)
- Ici, on peut implanter un équivalent de server side include.
- Attention : le servlet cible a accès à tous les aspects de l'objet “requête”, mais l'utilisation de

```
String path = "/raisins.jsp?orderno=5";
RequestDispatcher rd = context.getRequestDispatcher(path);
rd.include(request, response);
```

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    try {
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Servlet Servlet0</title>");
        out.println("</head>");
        out.println("<body>");
        out.println("<h1>Servlet Servlet0 at " + request.getContextPath () + "</h1>");
        out.println("</body>");
        out.println("</html>");

        this.getServletContext().getRequestDispatcher("/Servlet2").include(request, response);

    } finally {
        out.close();
    }
}
```

Les attributs du contexte

- Valeurs disponibles pour toute l'application

```
Integer i=(Integer) getServletContext().getAttribute("hit");
if (i==null) {
    getServletContext().setAttribute("hit",1);
} else {
    getServletContext().setAttribute("hit",i+1);
}
out.println("HitCount 1 "+getServletContext().getAttribute("hit"));
```

Le contexte

- Il est possible d'avoir des paramètres d'initialisation du contexte
  - <context-param>
    - <param-name>database</param-name>
    - <param-value>testDB</param-value>
    - <description>le nom de la base</description>
  - </context-param>
- Ces paramètres peuvent être utilisés dans le servlet avec la méthode
  - Javax.servlet.ServletContext getInitParameter() et
  - Javax.servlet.ServletContext.getInitParameterNames()

Le ServletConfig

- Objet utilisé par le container de servlet pour passer des valeurs lors de l'initialisation du servlet
- Obtenu avec
  - Servlet.getServletConfig()
- Principales méthodes
  - String getInitParameter(String name)
    - Retourne la valeur d'un paramètre d'initialisation du servlet
  - ServletContext getServletContext()
    - Permet de récupérer l'objet servletcontext
  - String getServletName()
    - Retourne le nom du servlet comme spécifié dans le fichier de description.
- Implanté par HttpServlet

Les listeners

- Les listeners sont des objets dont les méthodes sont invoquées en fonction du cycle de vie d'un servlet ou d'une application
- A la création et à la destruction d'un contexte (une appli)
  - Javax.servlet.ServletContextListener
- Quand on modifie les attributs du contexte
  - Javax.servlet.ServletContextAttributeListener
- A la création, la suppression, le timeout d'une session
  - Javax.servlet.HttpSessionListener
- A la création, modification, suppression d'un attribut de session
  - Javax.servlet.HttpSessionAttributeListener

Pourquoi des listeners ?

- Pour une application Web
  - Pas de point d'entrée (main)
  - Pas d'exit()
  - Comment faire pour initialiser l'application ?
- Pour une session
  - Possibilité de démarrer à plusieurs adresses (bookmarks, recherche)
  - Comment initialiser la session ?
  - Pas de fin de session explicite.
  - Comment terminer la session ?

Les listeners du contexte

ServletContextListener

Void **contextDestroyed**(ServletContextEvent sce)
Notification that the servlet context is about to be shut down.

void **contextInitialized**(ServletContextEvent sce)
Notification that the web application is ready to process requests.

ServletContextAttributeListener

void **attributeAdded**(ServletContextAttributeEvent scab)
Notification that a new attribute was added to the servlet context.

void **attributeRemoved**(ServletContextAttributeEvent scab)
Notification that an existing attribute has been remved from the servlet context.

void **attributeReplaced**(ServletContextAttributeEvent scab)
Notification that an attribute on the servlet context has been replaced.

Les listeners de session

HttpSessionListener

Void **sessionCreated**(HttpSessionEvent se)

Notification that a session was created.

void **sessionDestroyed**(HttpSessionEvent se)

Notification that a session was invalidated.

HttpSessionAttributeListener

void **attributeAdded**(HttpSessionBindingEvent se)

Notification that an attribute has been added to a session.

void **attributeRemoved**(HttpSessionBindingEvent se)

Notification that an attribute has been removed from a session.

void **attributeReplaced**(HttpSessionBindingEvent se)

Notification that an attribute has been replaced in a session.

La déclaration d'un listener

- <listener>
- <listener-class>
- cours.exemple1.SessionCounter
- </listener-class>
- </listener>

- Il n'est pas nécessaire de préciser le type du Listener (pourquoi ?)

```
<listener>
    <description>HttpSessionListener, HttpSessionAttributeListener</description>
    <listener-class>exemple.MyListener</listener-class>
</listener>
```

```java
@WebListener()
public class MyListener implements ServletContextListener {

    public void contextInitialized(ServletContextEvent sce) {
        System.out.println(sce.getServletContext().getContextPath()+":activated");
    }

    public void contextDestroyed(ServletContextEvent sce) {
            System.out.println(sce.getServletContext().getContextPath()+":Destroyed");
    }
}
```

```
INFO: /TestListenersAndFilters:activated
INFO: Loading application TestListenersAndFilters at /TestListenersAndFilters
INFO: TestListenersAndFilters was successfully deployed in 259 milliseconds.
INFO: /TestListenersAndFilters:Destroyed
```

Exemple

```
public class MyListener implements HttpSessionListener, HttpSessionAttributeListener {

    public void sessionCreated(HttpSessionEvent arg0) {
        System.out.println("Session Created"+arg0.getSession());
        arg0.getSession().setAttribute("newsession", "ma valeur");
    }

    public void sessionDestroyed(HttpSessionEvent arg0) {
        System.out.println("Session Destroyed"+arg0.getSession());
    }

    public void attributeAdded(HttpSessionBindingEvent arg0) {
        System.out.println("Session Attrbute Added"+arg0.getName());
        System.out.println("Session Attrbute Added"+arg0.getSession().getAttribute(arg0.getName()))
    }

    public void attributeRemoved(HttpSessionBindingEvent arg0) {
        throw new UnsupportedOperationException("Not supported yet.");
    }

    public void attributeReplaced(HttpSessionBindingEvent arg0) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
}
```

Un autre exemple

```
public class SessionCounterListener implements HttpSessionListener {

    public void sessionCreated(HttpSessionEvent arg0) {
        Integer i=(Integer) arg0.getSession().getServletContext().getAttribute("count");
        arg0.getSession().getServletContext().setAttribute("count",i+1);
        System.out.println("Nombre de sessions = "+(i+1));
    }

    public void sessionDestroyed(HttpSessionEvent arg0) {
        Integer i=(Integer) arg0.getSession().getServletContext().getAttribute("count");
        arg0.getSession().getServletContext().setAttribute("count",i-1);
        System.out.println("Nombre de sessions = "+(i-1));
    }
}
```

```
Application déployée
Session Created AB4130236FA5B9E054232D468ED9F070
Session Attribute Added newsession
Session Attribute Value ma valeur
Nombre de sessions = 1
Session Attribute Added user
Session Attribute Value john
Nombre de sessions = 0
Session Destroyed AB4130236FA5B9E054232D468ED9F070
```

Les filtres

- Un filtre est un morceau de code exécuté entre la requête et le « endpoint »
- Permettent de faire du pre et post-processing sur une requête
  - Lire la requête, modifier la requête modifier la réponse, retourner des erreurs au client
- Même cycle de vie qu'un servlet
  - Init / doFilter / destroy

Intérêt des filtres

- Permettent d'étendre l'application sans changer les servlets
- Services transversaux en entrée et en sortie
- En entrée
  - Log des appels
  - Sécurité
  - Contrôle des paramètres
- En sortie
  - Post processing du flux (XSLT)
  - Compression (gzip)
- Fonctionnent comme un pipe & filter

Coder un filtre

- Implanter la méthode doFilter()
- Déclarer le filtre dans web.xml
  - Filter
  - Filter-mapping
- Même fonctionnement que pour un servlet
- Il faut implanter la fonction doFilter
- Le transfert à la suite de la chaine se fait par la fonction chain.doFilter()
  - Transfert à un autre filtre ou à un servlet ou une page HTML ou une page JSP
  - La suite s'exécute au retour de doFilter

Exemple (généré par NetBeans)

```
@WebFilter(filterName="MyFilter", urlPatterns={"/test"},
           dispatcherTypes={DispatcherType.REQUEST})
public class MyFilter implements Filter {
```

```
public void doFilter(ServletRequest request, ServletResponse response,
                     FilterChain chain)
    throws IOException, ServletException {

    doBeforeProcessing(request, response);

    chain.doFilter(request, response);

    doAfterProcessing(request, response);
}
```

PreProcessing

- Affichage des paramètres

```
private void doBeforeProcessing(ServletRequest request, ServletResponse response)
    throws IOException, ServletException {
    if (debug) log("MyFilter:DoBeforeProcessing");

    for (Enumeration en = request.getParameterNames(); en.hasMoreElements(); ) {
        String name = (String)en.nextElement();
        String values[] = request.getParameterValues(name);
        int n = values.length;
        StringBuffer buf = new StringBuffer();
        buf.append(name);
        buf.append("=");
        for(int i=0; i < n; i++) {
            buf.append(values[i]);
            if (i < n-1)
                buf.append(",");
        }
        log(buf.toString());
    }
}
```

PotsProcessing

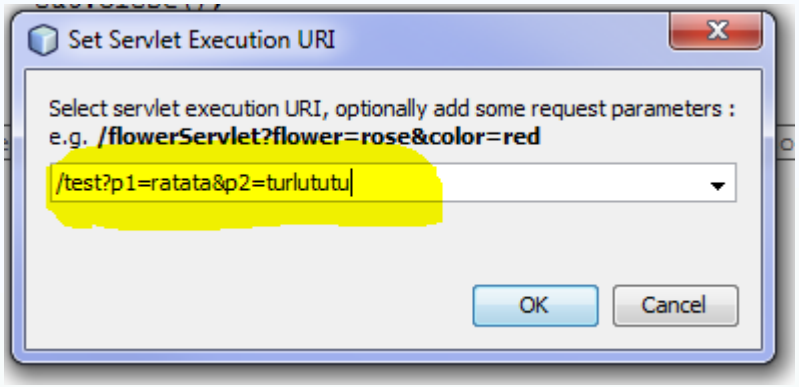
- Ajout de texte dans la sortie

```
private void doAfterProcessing(ServletRequest request, ServletResponse response)
    throws IOException, ServletException {
    if (debug) log("MyFilter:DoAfterProcessing");

    for (Enumeration en = request.getAttributeNames(); en.hasMoreElements(); ) {
        String name = (String)en.nextElement();
        Object value = request.getAttribute(name);
        log("attribute: " + name + "=" + value.toString());

    }
    PrintWriter respOut = new PrintWriter(response.getWriter());
    respOut.println("<P><B>This has been appended by an intrusive filter.</B>");
}
```

Le servlet filtré

```
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    PrintWriter out = response.getWriter();
    request.setAttribute("kiki", "kikou");
        out.println("Servlet FilterTestServlet at " + request.getContextPath () + "<br>");
        out.println("I have been filtered");
}
```

L'exécution



Servlet FilterTestServlet at /CoursWeb
I have been filtered

This has been appended by an intrusive filter.

```
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():MyFilter:Initializing filter
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():MyFilter:DoBeforeProcessing
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():p1=ratata
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():p2=turlututu
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():MyFilter:DoAfterProcessing
INFO: PWC1412: WebModule[/CoursWeb] ServletContext.log():attribute: kiki=kikou
```

Sur le serveur

Wrapping

- Principe : encapsuler l'objet requête ou l'objet réponse pour modifier son comportement
- Encapsulation de request : surcharge HttpServletRequestWrapper
- Encapsulation de reponse : surcharge de HttpServletRequestResponse
- Ensuite il suffit de remplacer l'objet wrappé.
- Le Wrapping peut aussi être utilisé dans des Servlets normaux.

Le Wrapping des réponses

- Plus compliqué que pour les requêtes
- Nécessite de
  - récupérer la réponse d'abord
  - Ou d'écrire un Writer spécialisé
- Utilisations classiques
  - Caching des réponses
  - Compression des réponses

Exemple de filtre

```
public void doFilter(ServletRequest request, ServletResponse response,
                     FilterChain chain)
    throws IOException, ServletException {

    System.out.println("Execution du filtre (before proceed)");
    chain.doFilter(request, response);
    System.out.println("Execution du filtre (after proceed)");
}
```

La gestion des erreurs

- Il est possible de définir les pages à afficher
  - En fonction d'erreurs http
  - En fonction d'exceptions java

Dans web.xml

Pour une exception java

```
<error-page>
    <exception-type>
        cours.event.EventException
    </exception-type>
    <location>/erreur.html</location>
</error-page>
```

Pour une erreur Http

```
<error-page>
   <error-code>404</error-code>
   <location>/404.html</location>
</error-page>
```

Servlet et sécurité (from servlet spec)

- A web application contains resources that can be accessed by many users.
- These resources often traverse unprotected, open networks such as the Internet.
- In such anenvironment, a substantial number of web applications will have security requirements.
  - Authentication
  - Access Control
  - Data Integrity

Sécurité

- The Servlet Container support declarative security and programmatic security.
  - Declarative : Security policies are declared in deployement descriptor (web.xml) or as annotation
  - Programmatic security allow servlet developers to directly interact with security API of the container.
    - Use it carefully (only if declarative security is not sufficient)

Declarative security

```
<security-constraint>

    <web-resource-collection>
     <web-resource-name>precluded methods</web-resource-name>
     <url-pattern>/*</url-pattern>
     <url-pattern>/acme/wholesale/*</url-pattern>
     <url-pattern>/acme/retail/*</url-pattern>
     <http-method-exception>GET</http-method-exception>
     <http-method-exception>POST</http-method-exception>
    </web-resource-collection>

    <auth-constraint/>

</security-constraint>

<security-constraint>

    <web-resource-collection>
     <web-resource-name>wholesale</web-resource-name>
     <url-pattern>/acme/wholesale/*</url-pattern>
     <http-method>GET</http-method>
     <http-method>PUT</http-method>
    </web-resource-collection>

    <auth-constraint>
     <role-name>SALESCLERK</role-name>
    </auth-constraint>

</security-constraint>
```

Security constraints

TABLE 13-4 Security Constraint Table

url-pattern	http-method	permitted roles	supported connection types
/*	all methods except GET, POST	access precluded	not constrained
/acme/wholesale/*	all methods except GET, POST	access precluded	not constrained

Security model

- A security role is a logical grouping of users defined by the **Application Developer** or **Assembler**.
- When the application is **deployed**, roles are mapped by a **Deployer** to **principals or groups in the runtime environment**.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="
    <servlet>
        <servlet-name>Hello</servlet-name>
        <servlet-class>Hello</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Hello</servlet-name>
        <url-pattern>/Hello</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <security-constraint>
        <display-name>Constraint1</display-name>
        <web-resource-collection>
            <web-resource-name>World</web-resource-name>
            <description/>
            <url-pattern>/World</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description/>
            <role-name>pouet</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
    </login-config>
    <security-role>
        <description>c'est le group qui fait des pouet pouets</description>
        <role-name>pouet</role-name>
    </security-role>
</web-app>
```

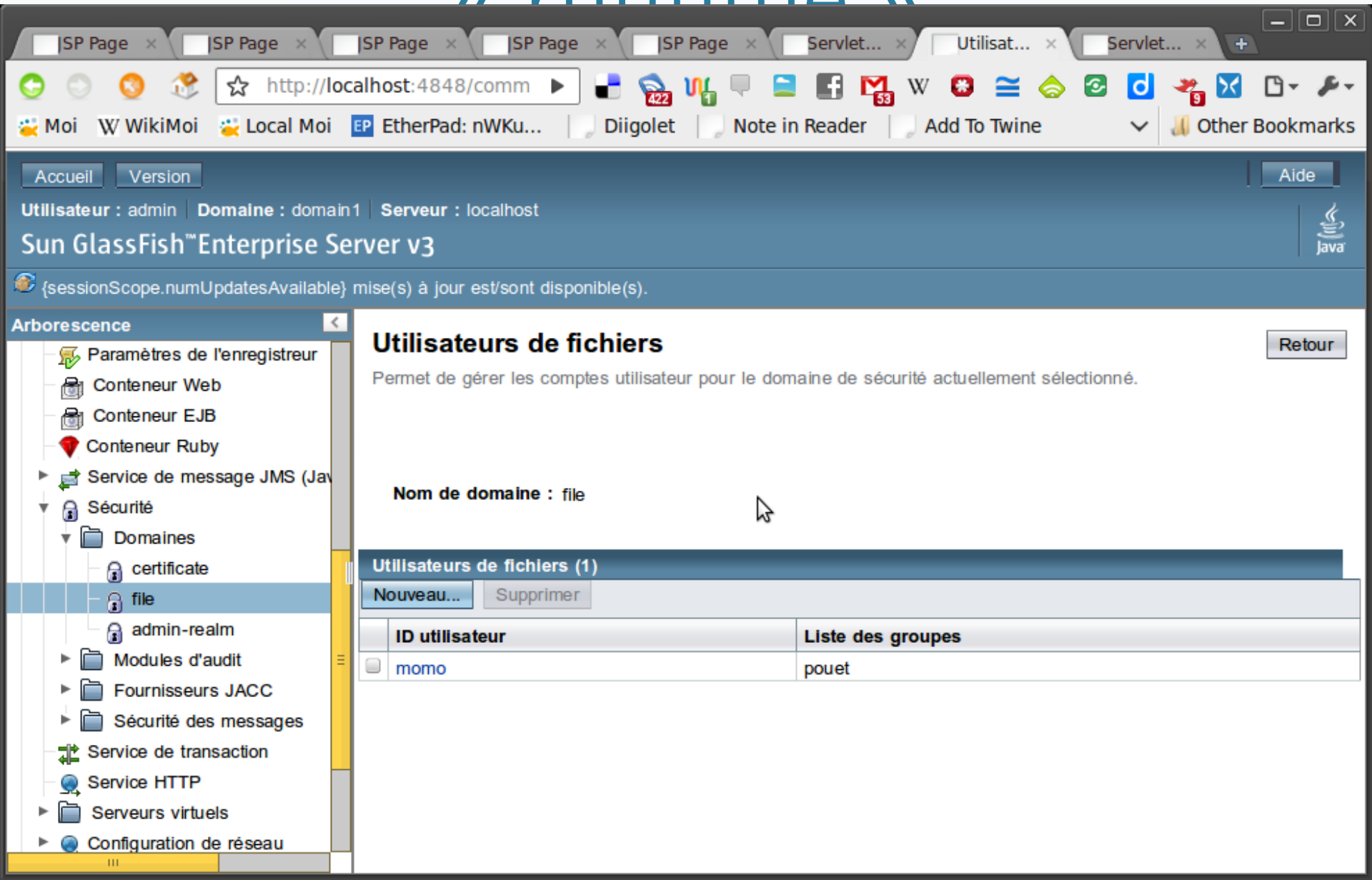
Les users ayant endossé le rôle « pouet » ont le droit de requéter /World

Peut être changé par le « deployer »

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Serv
<sun-web-app error-url="">
  <context-root>/MyHello</context-root>
  <security-role-mapping>
    <role-name>pouet</role-name>
    <group-name>pouet</group-name>
  </security-role-mapping>
  <class-loader delegate="true"/>
  <jsp-config>
    <property name="keepgenerated" value="true">
      <description>Keep a copy of the generated servlet class' java code.</description>
    </property>
  </jsp-config>
</sun-web-app>
```

Le rôle « pouet » est associé au groupe « pouet » déclaré sur glassfish, pour l'application web MyHello

Déclarer les users dans le « runtime »



Authentication

- A web client can authenticate a user to a web server using one of the following mechanisms:
  - HTTP Basic Authentication
  - HTTP Digest Authentication
  - HTTPS Client Authentication
  - Form Based Authentication

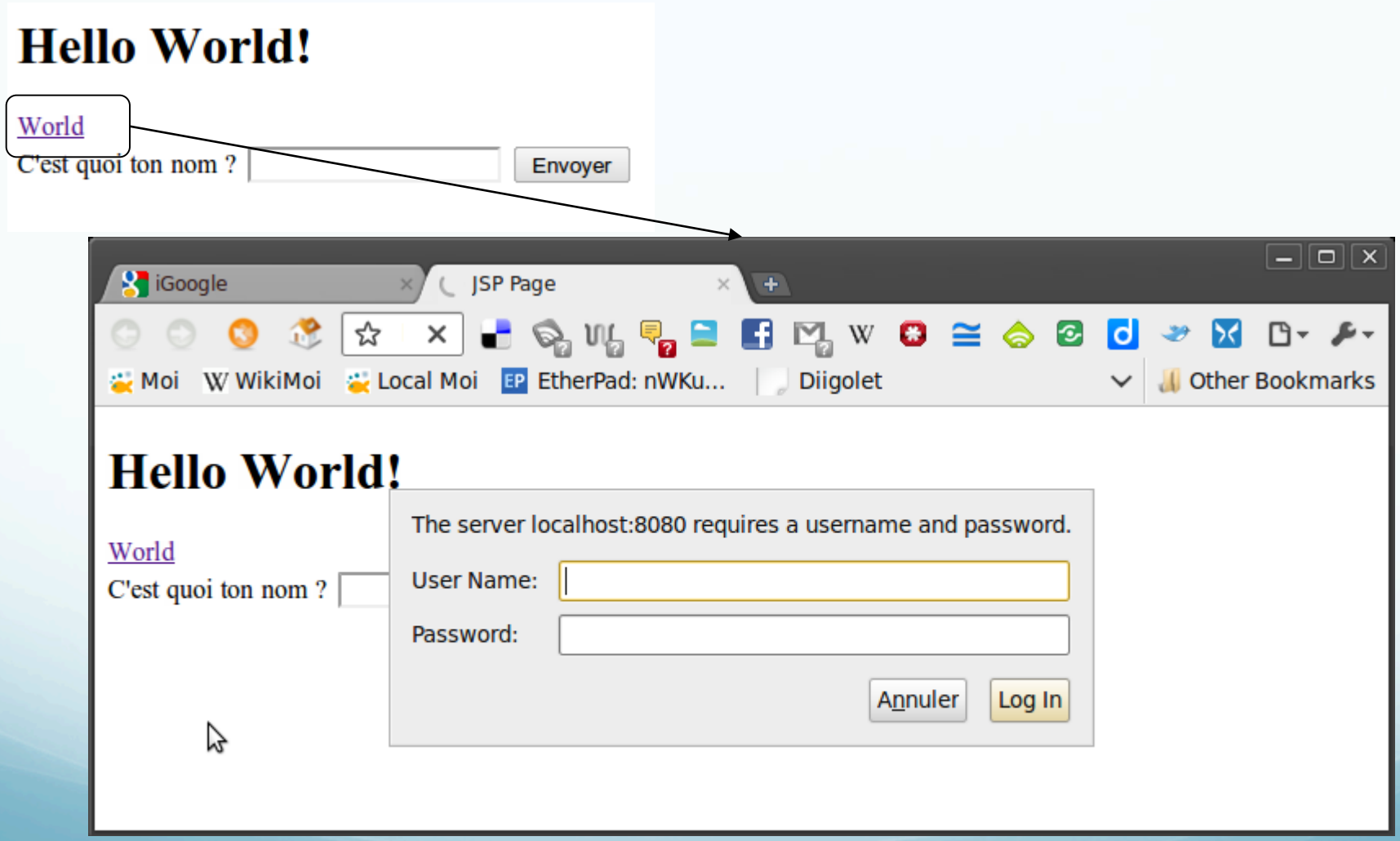
Authentication

- HTTP Basic Authentication, which is based on a username and password, is the authentication mechanism defined in the HTTP/1.0 specification
- Basic Authentication is not a secure authentication protocol.
  - User passwords are sentin simple base64 encoding
  - the target server is not authenticated
- Require HTTPS or IPSEC/VPN to be more secure

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="3.0" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="
    <servlet>
        <servlet-name>Hello</servlet-name>
        <servlet-class>Hello</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>Hello</servlet-name>
        <url-pattern>/Hello</url-pattern>
    </servlet-mapping>
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>
    <security-constraint>
        <display-name>Constraint1</display-name>
        <web-resource-collection>
            <web-resource-name>World</web-resource-name>
            <description/>
            <url-pattern>/World</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <description/>
            <role-name>pouet</role-name>
        </auth-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
    </login-config>
    <security-role>
        <description>c'est le group qui fait des pouet pouets</description>
        <role-name>pouet</role-name>
    </security-role>
</web-app>
```

Authentication
« BASIC »

Secured access to « World » servlet



Authentication : Digest

- HTTP Digest Authentication authenticates a user based on a username and a password.
- HTTP Digest Authentication does not send user passwords over the network
- the client sends a one-way cryptographic hash of the password (and additional data).
- HTTP Digest authentication requires that clear text password equivalents be available to the authenticating container so that it can validate received authenticators by calculating the expected digest.

Authentication « form login »

The web application deployment descriptor contains entries for a login form and error page. The login form must contain fields for entering a username and a password. These fields must be named `j_username` and `j_password`, respectively.

When a user attempts to access a protected web resource, the container checks the user's authentication. If the user is authenticated and possesses authority to access the resource, the requested web resource is activated and a reference to it is returned. If the user is not authenticated, all of the following steps occur:

Login Form

1. The login form associated with the security constraint is sent to the client and the URL path triggering the authentication is stored by the container.
2. The user is asked to fill out the form, including the username and password fields.
3. The client posts the form back to the server.
4. The container attempts to authenticate the user using the information from the form.
5. If authentication fails, the error page is returned using either a forward or a redirect, and the status code of the response is set to 200.
6. If authentication succeeds, the authenticated user's principal is checked to see if it is in an authorized role for accessing the resource.
7. If the user is authorized, the client is redirected to the resource using the stored URL path.

   The error page sent to a user that is not authenticated contains information about the failure.

Login form authentication

Form Based Authentication has the same lack of security as Basic Authentication since the user password is transmitted as plain text and the target server is not authenticated. Again additional protection can alleviate some of these concerns: a secure transport mechanism (HTTPS), or security at the network level (such as the IPSEC protocol or VPN strategies) is applied in some deployment scenarios.

The authenticate method of the HttpServletRequest interface provides an alternative means for an application to control the look and feel of it's login screens.

Deployment for form authentication (David Geary, Nov 30, 2001)

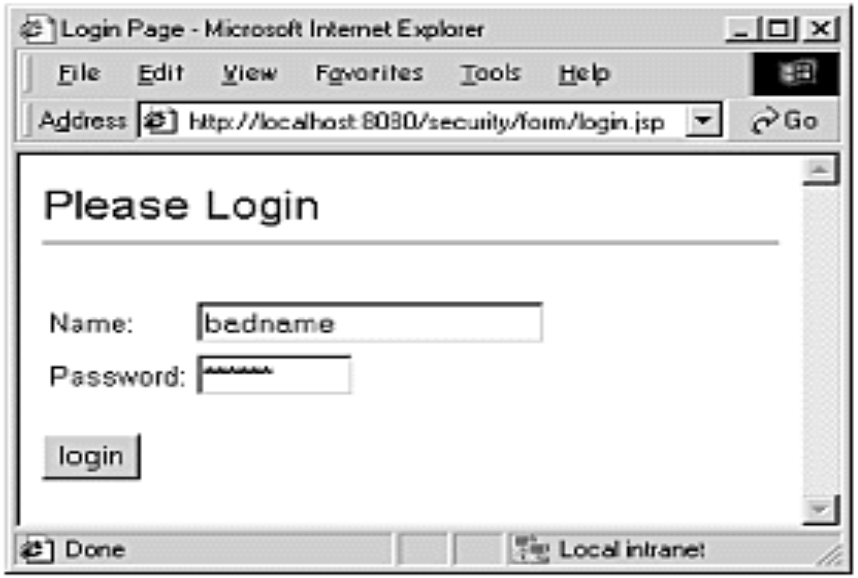
```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!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>
  <security-constraint>
    <web-resource-collection>
     <web-resource-name>A Protected Page</web-resource-name>
     <url-pattern>/protected-page.jsp</url-pattern>
   </web-resource-collection>

   <auth-constraint>
     <role-name>tomcat</role-name>
   </auth-constraint>
  </security-constraint>

  <login-config>
   <auth-method>FORM</auth-method>
   <form-login-config>
     <form-login-page>/login.jsp</form-login-page>
     <form-error-page>/error.jsp</form-error-page>
   </form-login-config>
  </login-config>
</web-app>
```

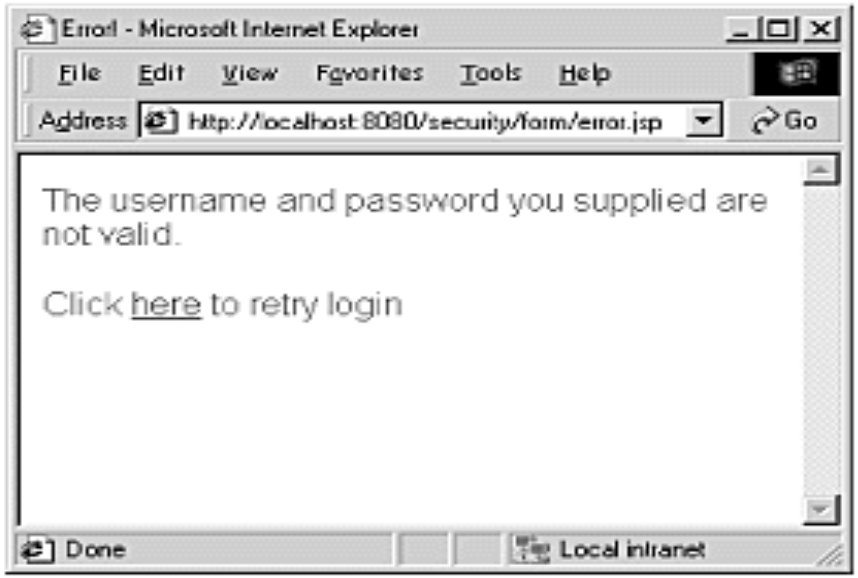
Login form (David Geary Nov 30, 2001)

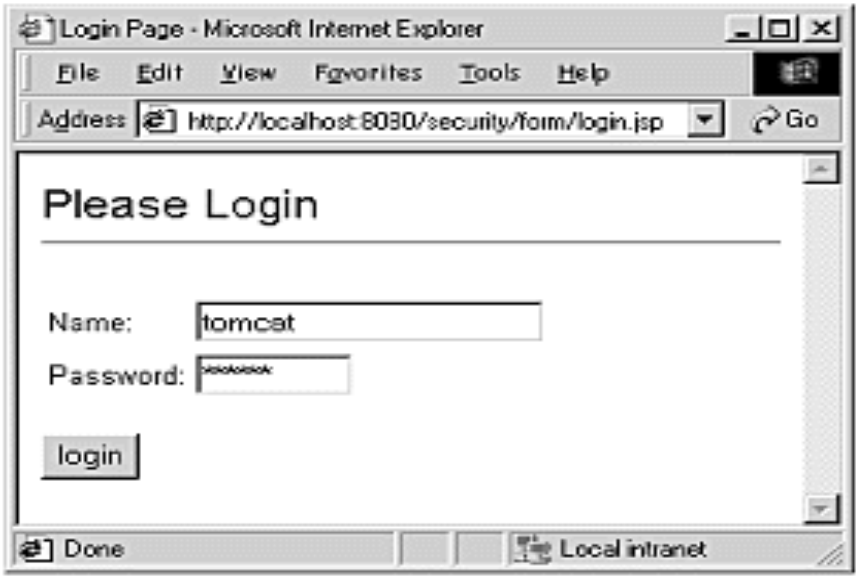
```
<html><head><title>Login Page</title></head>
<body>
<font size='5' color='blue'>Please Login</font><hr>

<form action='j_security_check' method='post'>
<table>
 <tr><td>Name:</td>
   <td><input type='text' name='j_username'></td></tr>
 <tr><td>Password:</td>
   <td><input type='password' name='j_password' size='8'></td>
 </tr>
</table>
<br>
  <input type='submit' value='login'>
</form></body>
 </html>
```

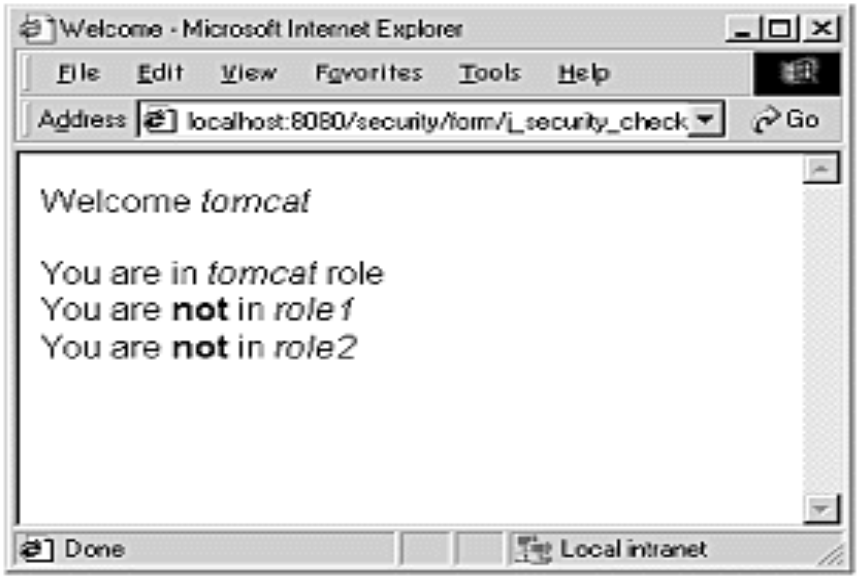
David Geary

Nov 30, 2001









Declarative security and annotation

- @ServletSecurity(@HttpConstraint(transportGuarantee =
- TransportGuarantee.CONFIDENTIAL))
- public class Example2 extends HttpServlet {
- ...
- }

- @ServletSecurity(@HttpConstraint(EmptyRoleSemantic.DENY))
- public class Example3 extends HttpServlet {
- }

- @ServletSecurity(@HttpConstraint(EmptyRoleSemantic.DENY))
- public class Example3 extends HttpServlet {
- }

Servlet Security

- @ServletSecurity((httpMethodConstraints = {
- @HttpMethodConstraint(value = "GET", rolesAllowed = "R1"),
- @HttpMethodConstraint(value = "POST", rolesAllowed = "R1",
- transportGuarantee = TransportGuarantee.CONFIDENTIAL)
- })
- public class Example5 extends HttpServlet {
- }

Conclusion

- The servlet model is a component model
- Servlet developpers have to conform to the model (implements HttpServlet class)
- The Servlet container offers:
  - Web application lifecycle management
  - Distributed container
  - Security management
  - Servlet composition (by dispatching)
  - Interception (through filters)
  - Session management facilities

What we have not seen

- Distributed Servlet container over a cluster of computers
- Asynchronous servlet (Servlet 3.0)
- Web Fragment :
  - Make Web application by composition web application fragment...

Servlet limitations

- Request/Response model can decrease maintainability...
  - The business logic of application are in servlets
  - HTML code or XHTML code is embedded in servlet...
- Need an higher abstraction level
  - to leverage maintainability problems
  - To speed up coding stage (RAD, rapid application development)...