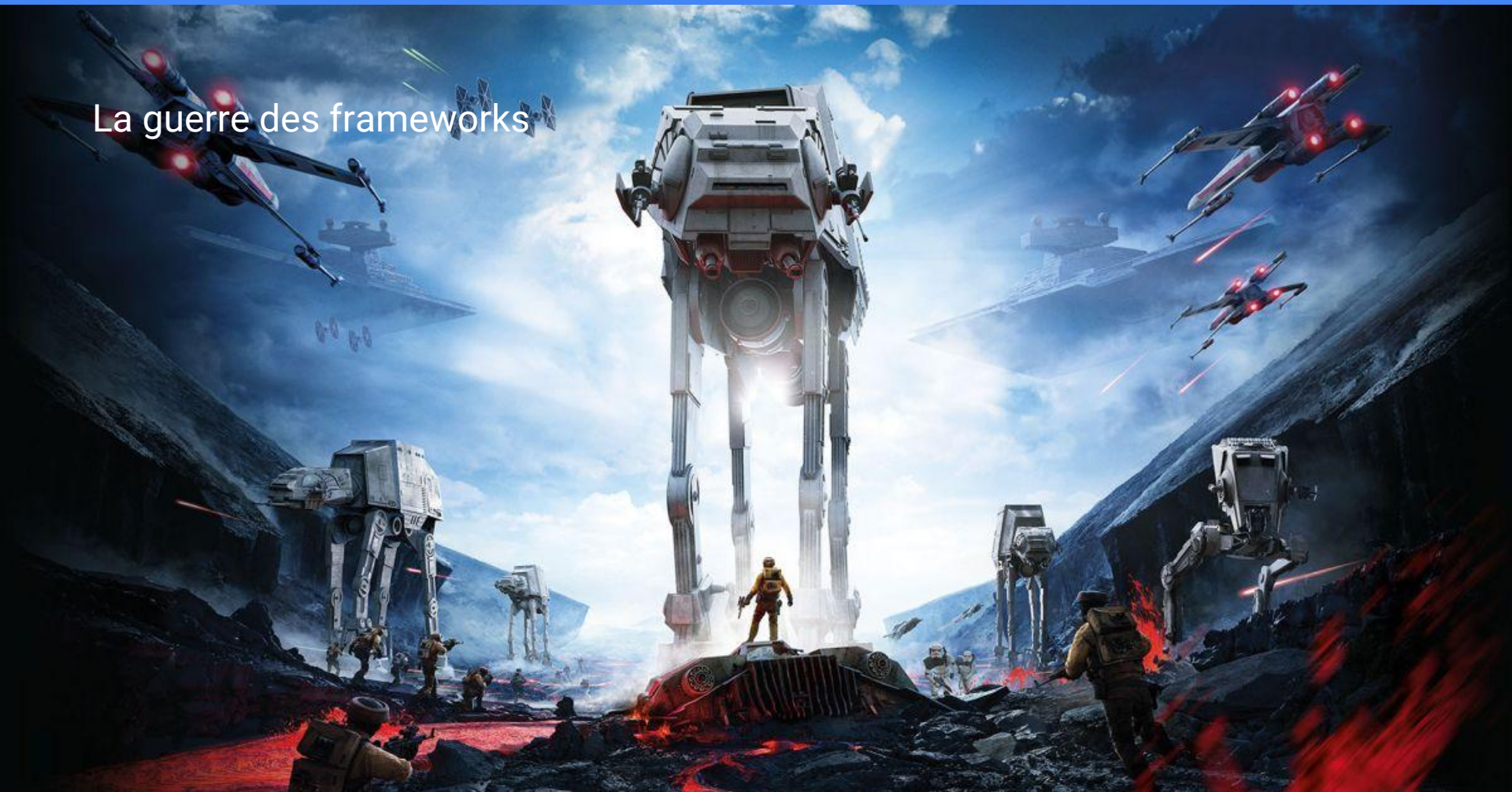
Côté serveur

La guerre des frameworks

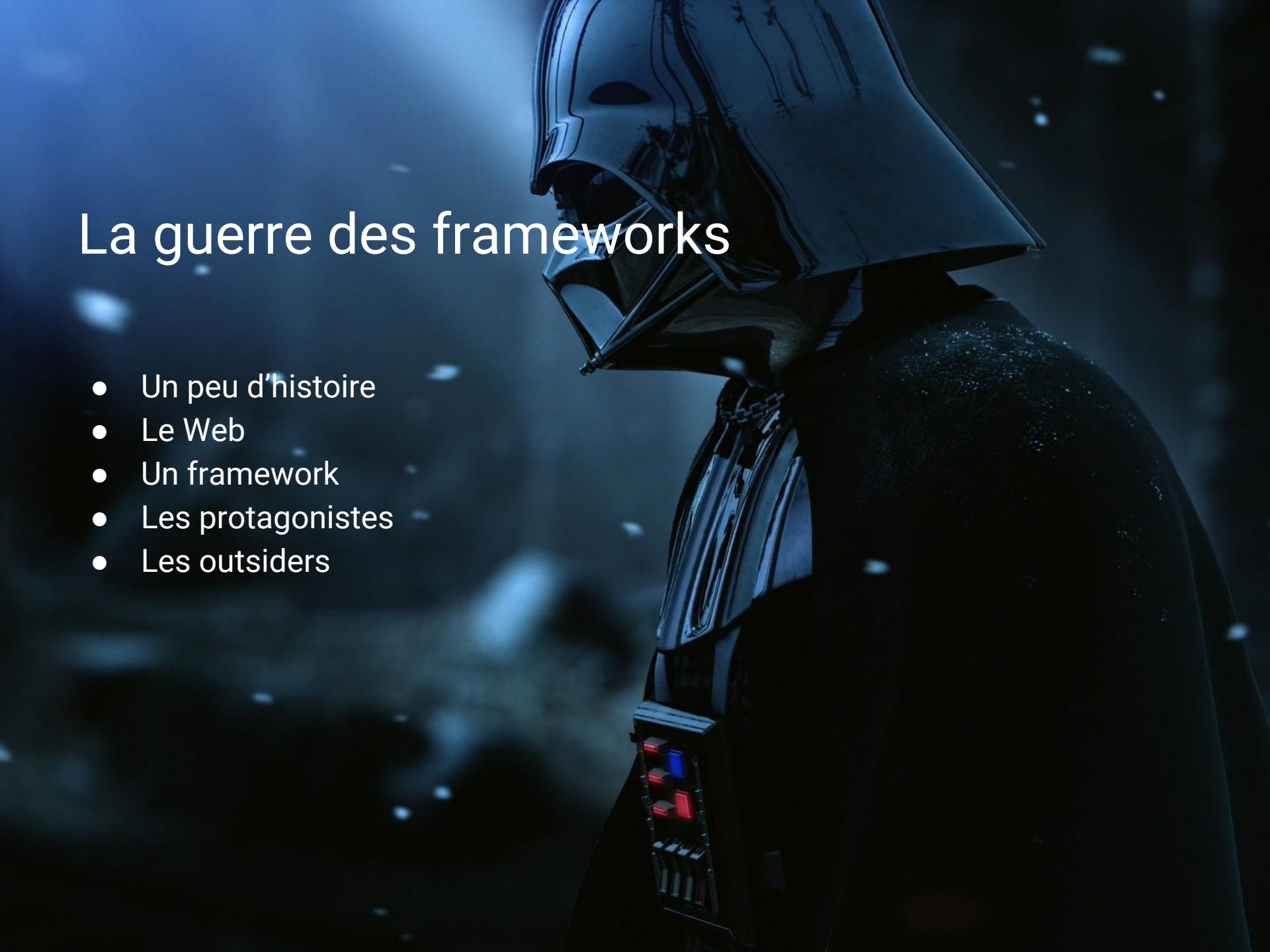


Xavier MARIN

@XavMarin

https://github.com/Giwi

CTO chez @qaobee



La guerre des frameworks

- Un peu d’histoire
- Le Web
- Un framework
- Les protagonistes
- Les outsiders

Un peu d’histoire

La préhistoire

- -3000 : première représentation binaire
- -500 : l'abaque et le boulier
- 1580 : les premiers logarithmes
- 1642 : première machine à calculer de Pascal
- 1728 : métier à tisser avec des cartes perforées
- 1792 : le télégraphe optique
- 1838 : le télégraphe électrique
- 1867 : la machine à écrire
- 1889 : calculatrice de bureau
- 1943 : MARK1 (3 opérations /secondes)

Un peu d'histoire

Les débuts

- 1951 : notion de compilateur
- 1958 : le COBOL
- 1960 : premier multi-tâches et premier micro-ordinateur
- 1963 : la souris
- 1969 : Arpanet (4 nœuds) et premier microprocesseur, UNIX
- 1970 : le C
- 1971 : Arpanet (23 nœuds)
- 1981 : le PC, MSDOS
- 1984 : 1000 nœuds sur Internet
- 1987 : 10000 nœuds sur Internet

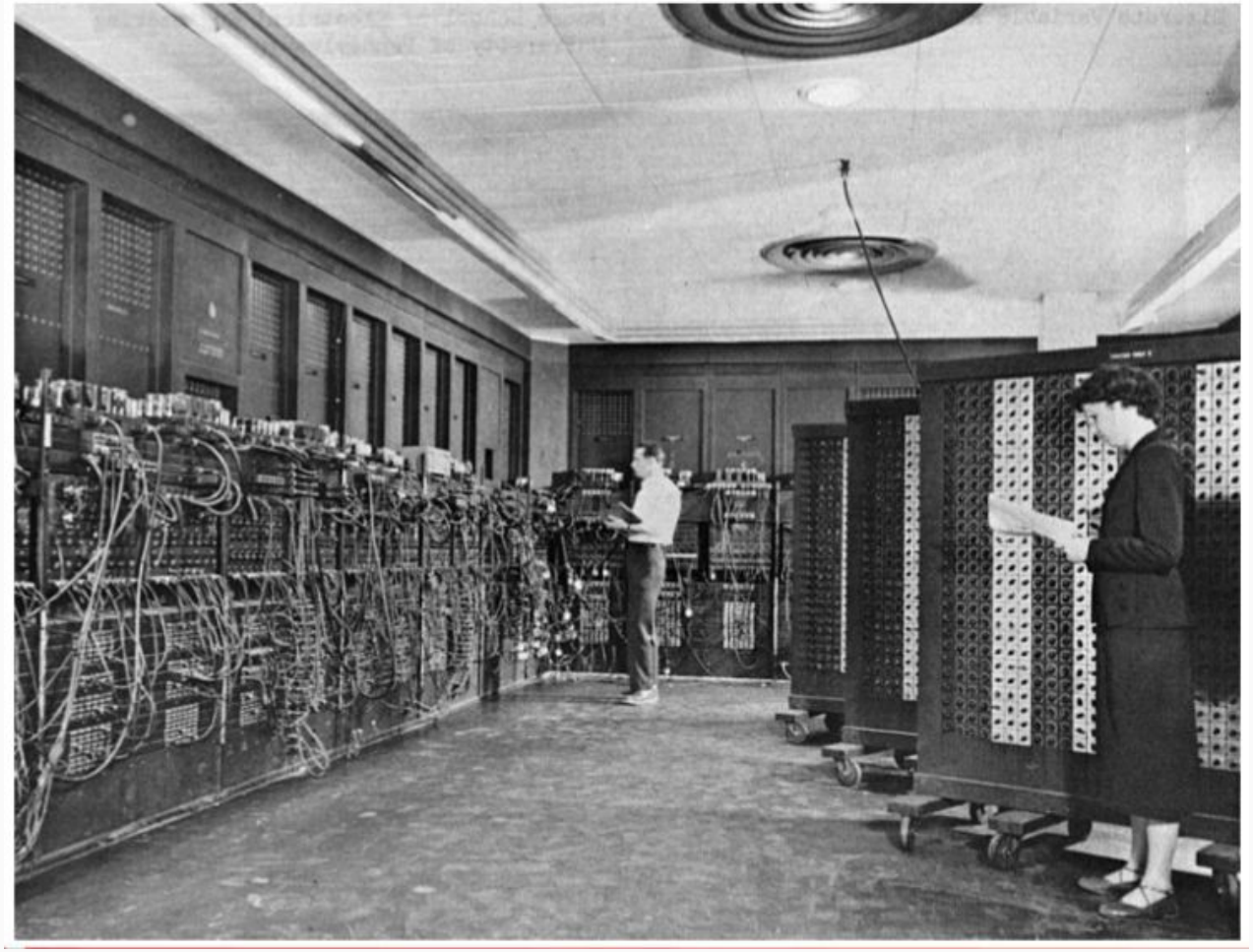
Un peu d’histoire

L’essort

- 1991 : création du protocole HTTP,
  Linux
- 1992 : 1 million d'ordinateurs sur Internet
- 1993 : premier navigateur Internet
- 1995 : Java
- 1996 : 10 millions d'ordinateurs sur Internet

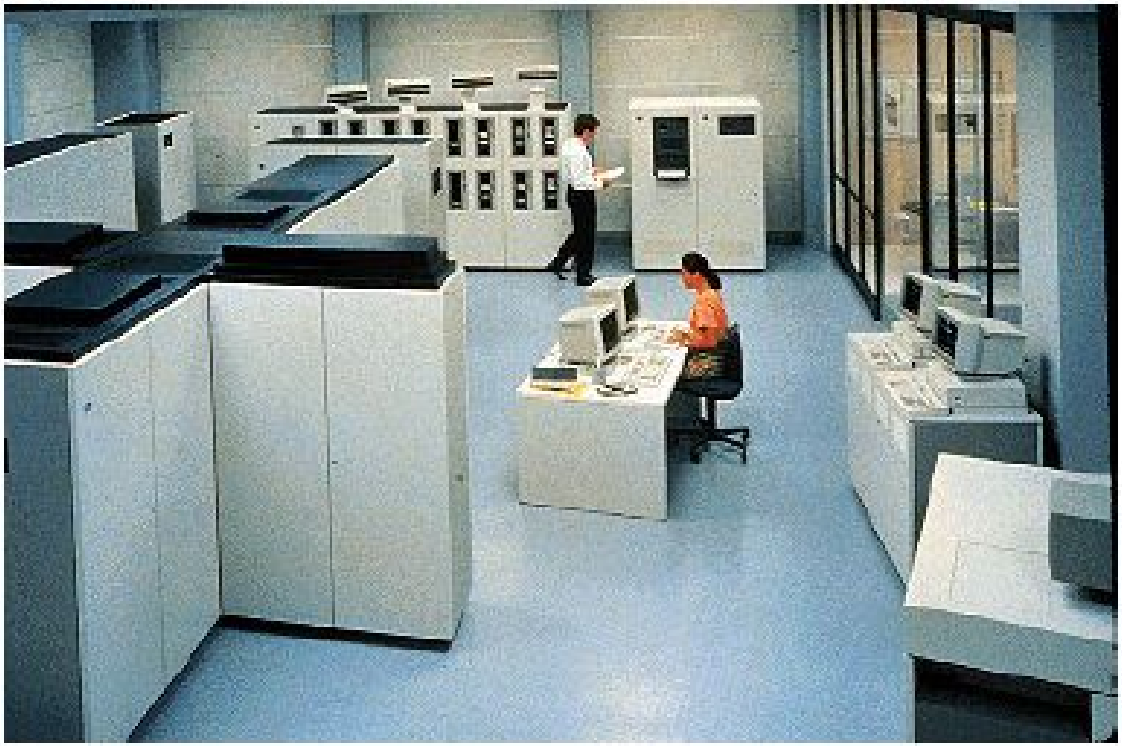
Un peu d'histoire : les architectures

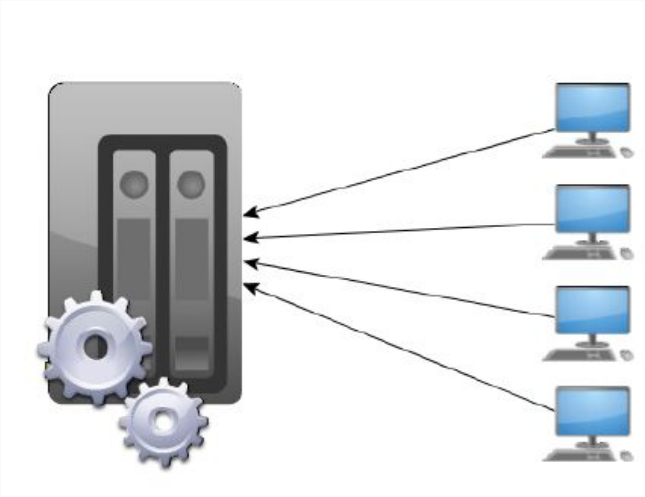
Le ruban perforé



Un peu d'histoire : les architectures

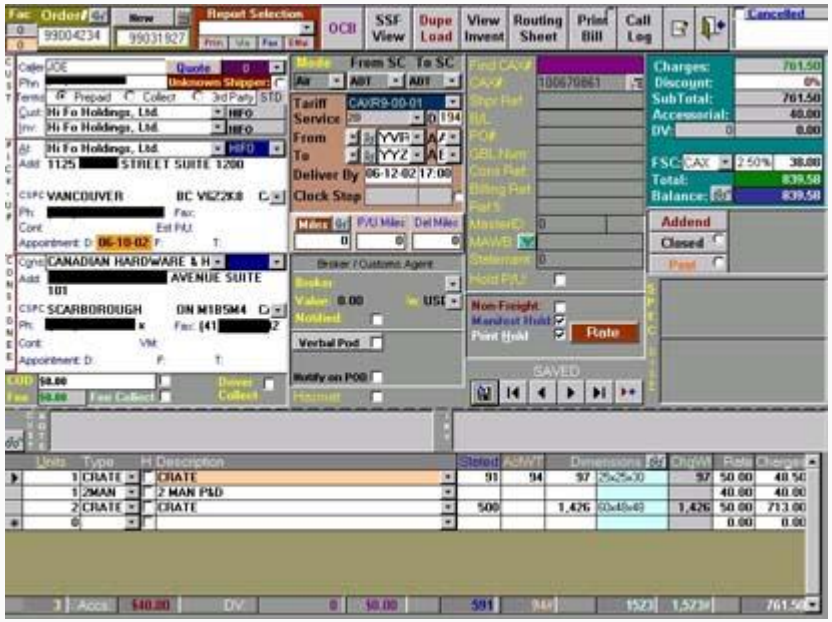
Le terminal

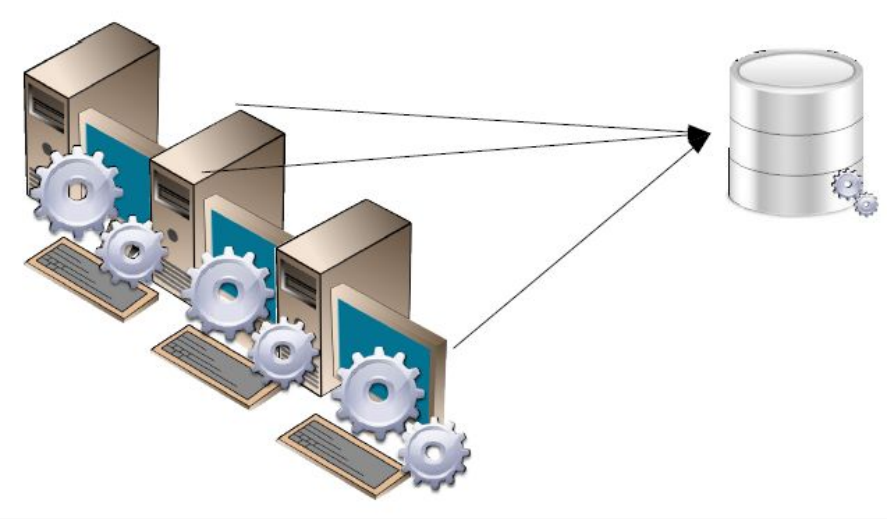




Un peu d'histoire : les architectures

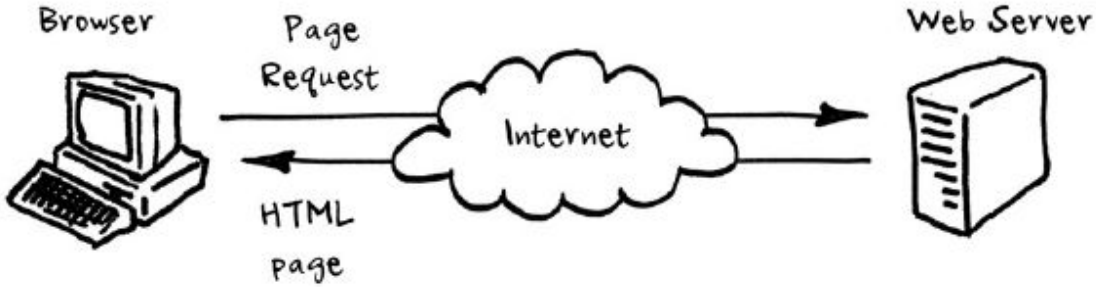
Le client lourd

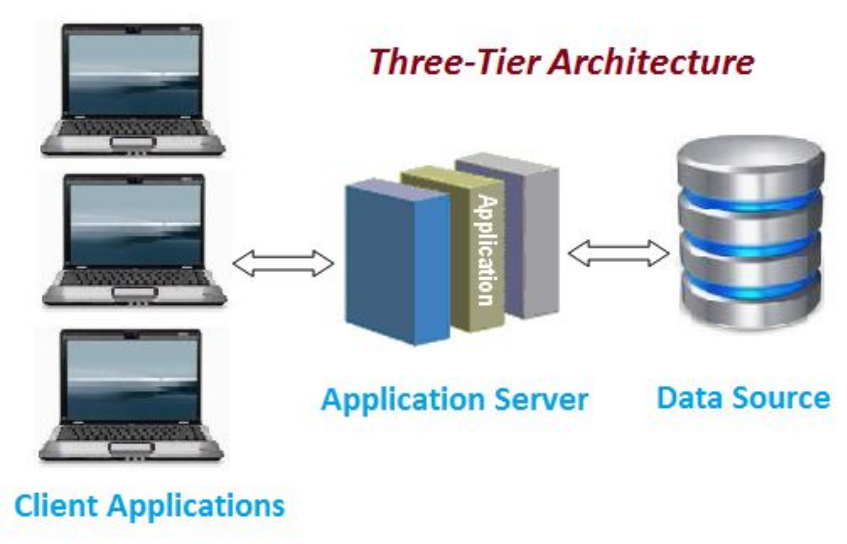




Un peu d'histoire : les architectures

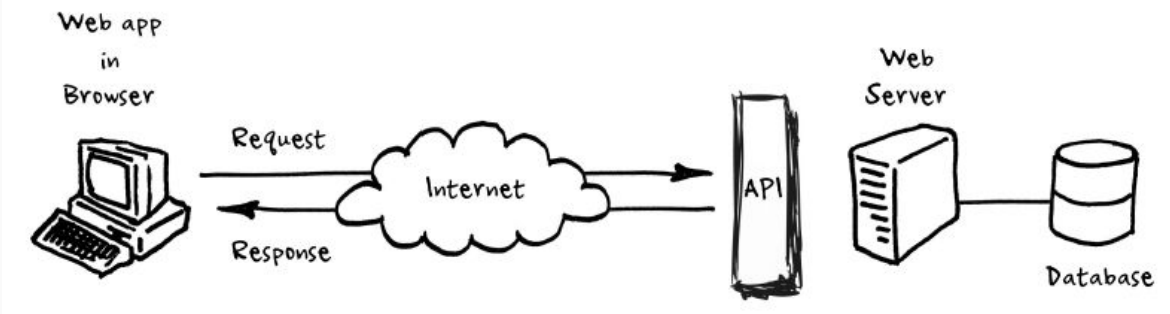
Le client léger / Internet

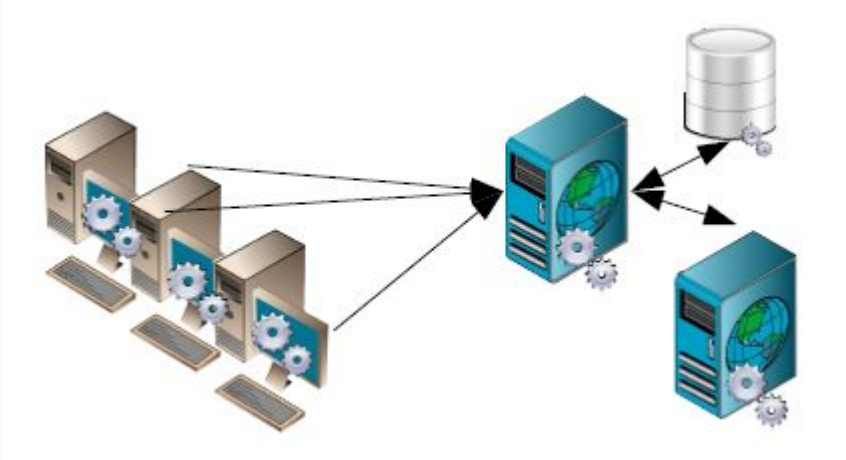




Un peu d'histoire : les architectures

Le client riche





Le Web

1. Un client (navigateur ou machine) émet une requête HTTP
2. Un serveur est à l'écoute de la requête et l'analyse
3. Le serveur envoi au client une réponse HTTP

<scheme>://<user>:<password>@<domain>/<path>?param1=value1&param2=value2#<fragment>

- http://www.w3.org/Protocols/rfc2616/rfc2616-sec5.html
- ftp://gilbert:toto75@ftp.enib.fr/etc/passwd
- https://www.google.fr/search?q=star+wars&client=ubuntu&hs=cKq&channel=fs&source=lnms&tbm=isch&sa=X&ved=0CAgQ_AUoAmoVChMI7PLn1dn8yAIVA7oUCh1VFw8E&biw=1366&bih=639#channel=fs&tbm=isch&q=death+star

Une requête HTTP

Une requête se fait sur une adresse, contient :

- des entêtes
- une méthode
- des paramètres

et éventuellement un corps.

```
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1
Host: www.w3.org
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:42.0) Gecko/20100101 Firefox/42.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
Accept-Encoding: gzip, deflate
Referer: https://www.google.fr
Connection: keep-alive
```

Les méthodes :

- GET
- PUT
- POST
- DELETE
- OPTIONS
- TRACE
- HEAD
- CONNECT

Une réponse HTTP

Une réponse contient des entêtes et éventuellement un corps.

```
Status Code: 200 OK
Cache-Control: max-age=21600
Content-Length: 11464
Content-Type: text/html; charset=iso-8859-1
Date: Fri, 06 Nov 2015 22:22:52 GMT
Expires: Sat, 07 Nov 2015 04:22:52 GMT
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
Server: Apache/2
```

Les status :

- 1xx
  - purement informatif
- 2xx
  - ok, tout va bien navette
- 3xx
  - souvent une redirection
- 4xx
  - ce ne sont pas les droids que vous cherchez
- 5xx
  - le serveur a un souci

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.
dtd">
<html xmlns='http://www.w3.org/1999/xhtml'>
<head><title>HTTP/1.1: Request</title></head>
<body><address>part of <a rev='Section' href='rfc2616.html'>Hypertext Transfer Protocol -- HTTP/1.1</a><br
/>
RFC 2616 Fielding, et al.</address>
<h2><a id='sec5'>5</a> Request</h2>
<p>A request message from a client to a server includes, within the first line of that message, the method
to be applied to the resource, the identifier of the resource, and the protocol version in use.</p> ...
```

Web Services - Les machines parlent aux machines

- Corba / RMI / Web RPC
  - très, très bas niveau
- JaxB
  - Jurassic Park
  - XML
- SOAP
  - contract-first
  - Inter-opérable, multi langages
  - XML

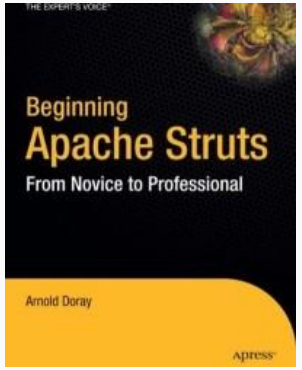
- Thrift
  - contract-first
- Protobuf
  - binaire
- REST
  - JSON
  - Inter-opérable, multi langages
  - on en reparlera

Un framework

- ne pas réinventer la roue, vous la ferez carrée
- masquer la complexité
- se concentrer sur le code métier
- améliorer :
  - la testabilité
  - maintenabilité
- ouvert
  - intégration de modules "prêts à plugger"
  - communautaire
- Réduire le 'time to market'

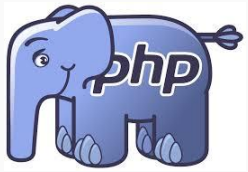
Les protagonistes

- API Servlet
  - Une url = une vue, une page HTTP
  - Session HTTP
- Tomcat
  - Conteneur "léger" (servlets et JSP, pas d'EJB)
- Struts
  - Modèle MVC
  - Configuration XML
- Spring
  - Inversion de contrôle
  - Programmation orientée aspect
  - Couche d’abstraction.



Les protagonistes

- PHP
  - et oui, ça existe encore :/
- PlayFramework
  - Groovy
- NodeJS
  - c10k
  - en javascript
  - on en reparlera
- Go Lang
  - en Go

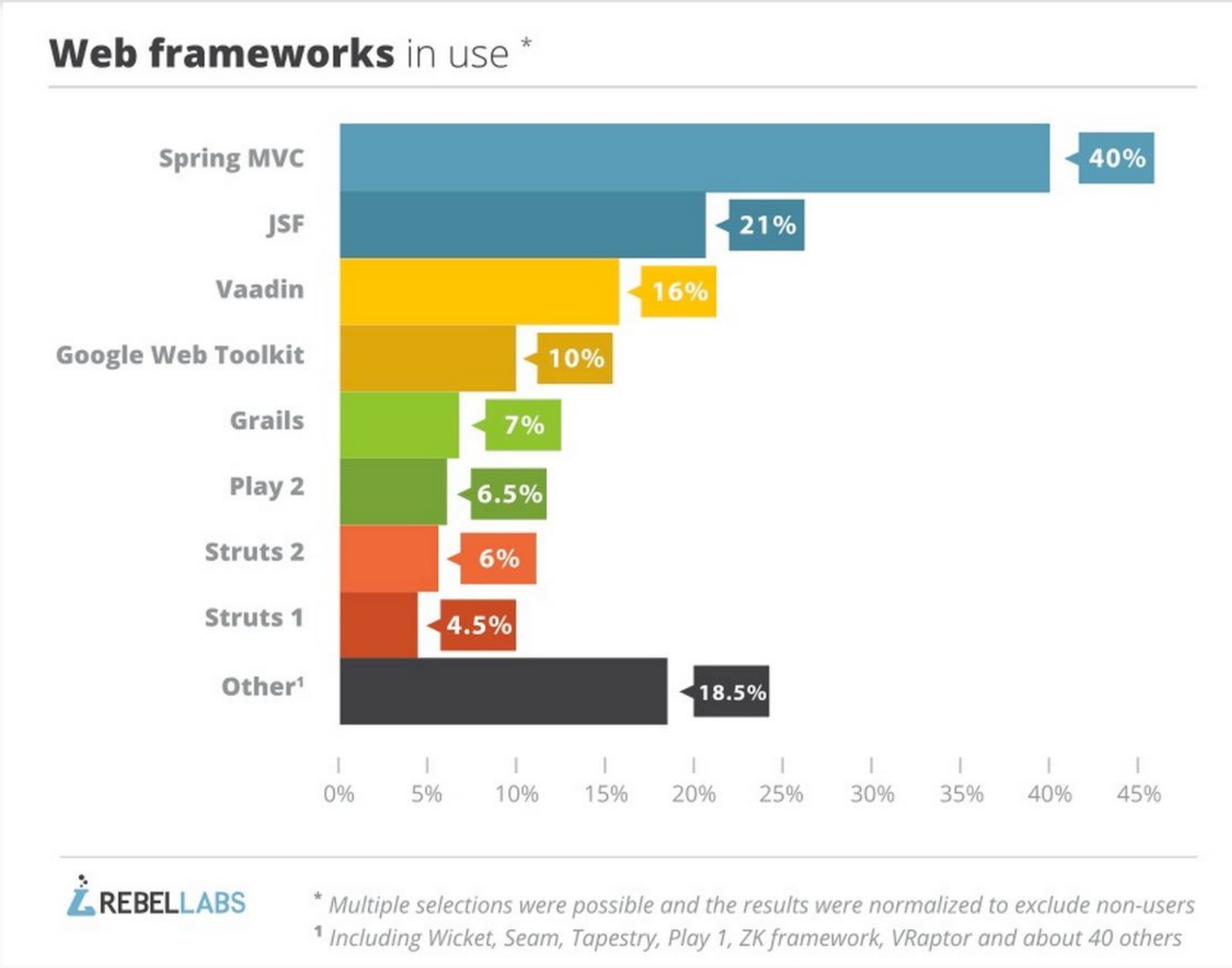


Les outsiders

- c10k
- REST / Web / WebSockets
- ressources statiques / templates
- modulaire
- clusterisable
- robuste
- très, très performant

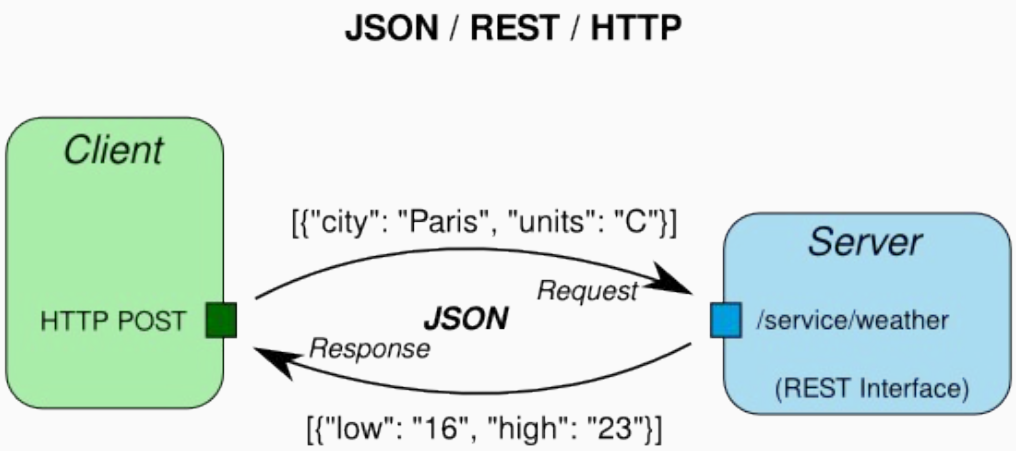
- REST / Web
- ressources statiques / templates
- léger
- simple
- autonome ou embarquable

Les frameworks



REST - Representational State transfert

- GET /tickets
  - Retrieves a list of tickets
- GET /tickets/12
  - Retrieves a specific ticket
- POST /tickets
  - Creates a new ticket
- PUT /tickets/12
  - Updates ticket #12
- DELETE /tickets/12
  - Deletes ticket #12



La servlet

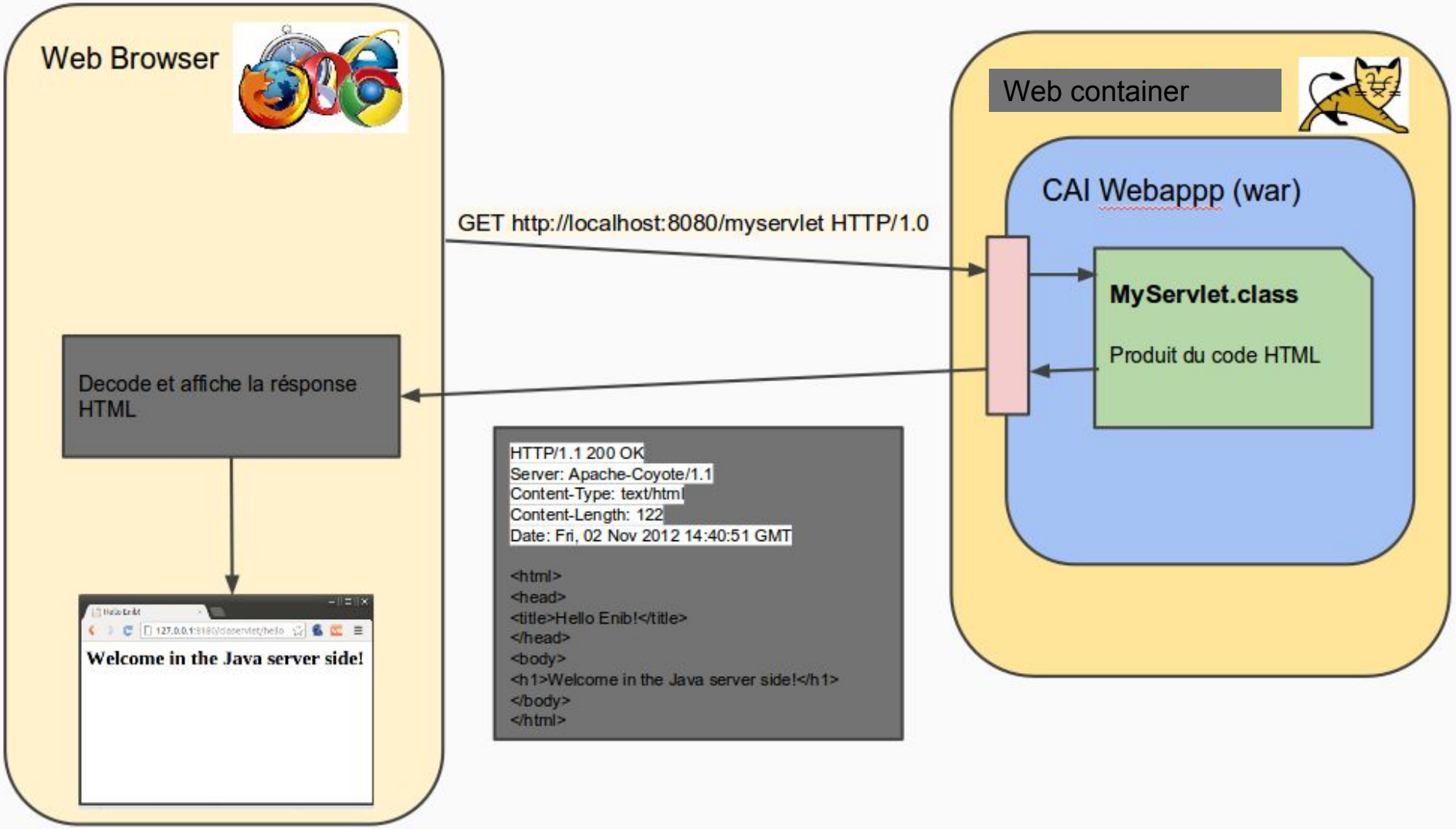
La Servlet est une classe Java permettant de créer dynamiquement des données au sein d'un serveur HTTP.

Une instance unique de la Servlet s'exécute à chaque requête HTTP reçue par le conteneur (Serveur Web).

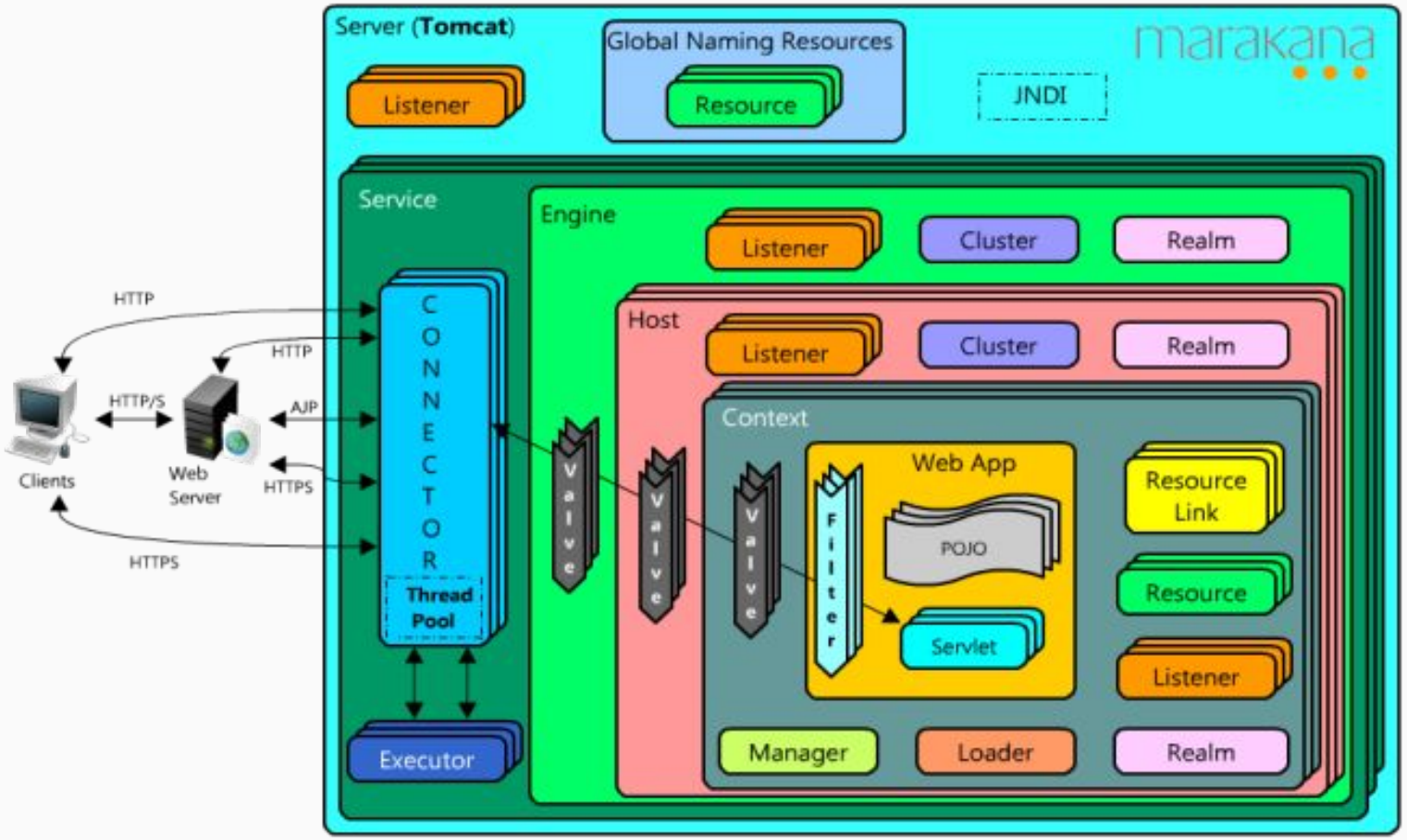
Elle produit du code (HTML, XML, JS...) compréhensible par un navigateur Web.

- Créée en 1997 (v1.0) par Sun Microsystems
- la servlet est un singleton
- interface définie dans le package javax.servlet
- normée par une JSR (Java Specification Requests)
- support natif dans un conteneur web java (Tomcat, Jetty, etc...)

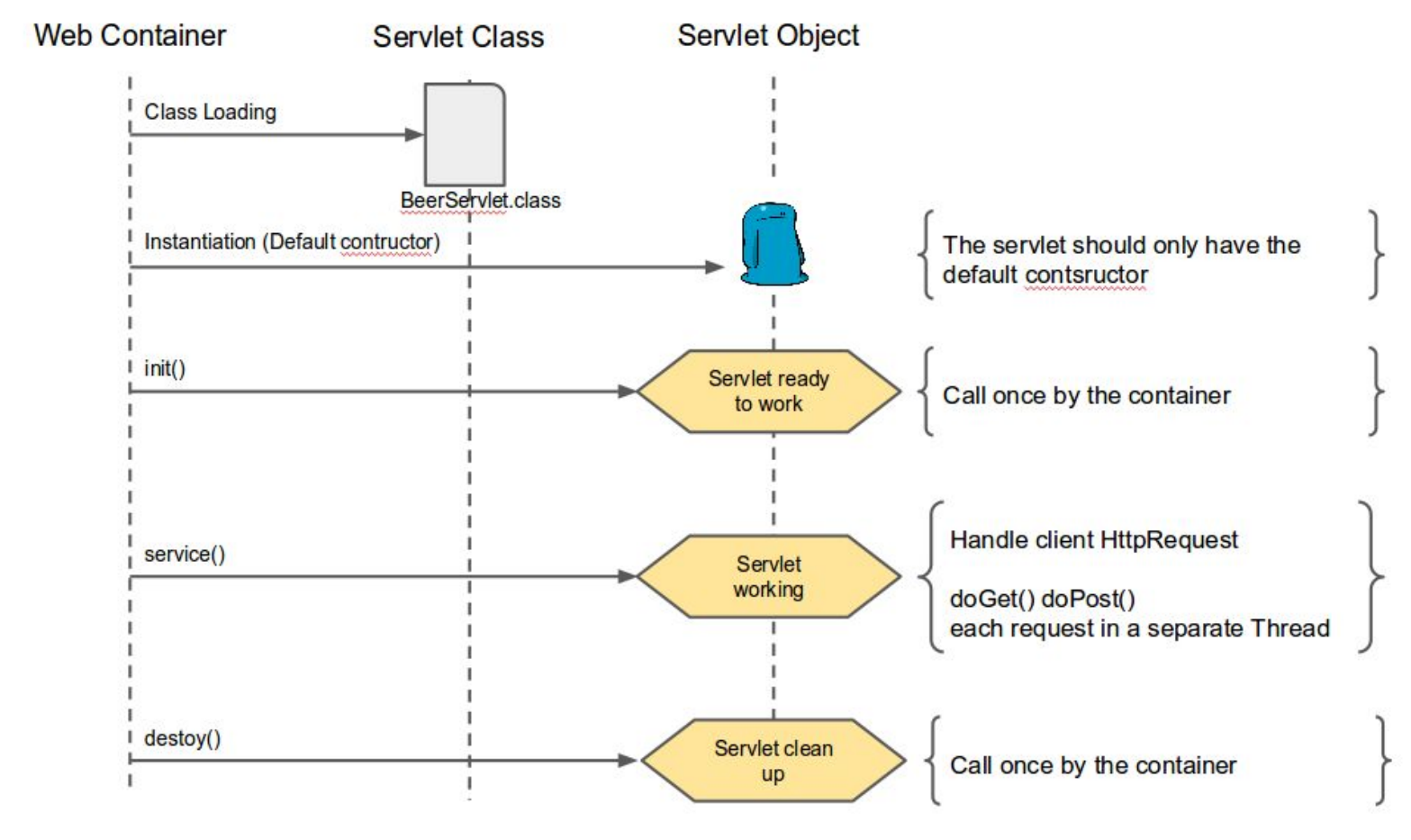
La servlet


Web Browser
Web container
CAI Webapp (war)
GET http://localhost:8080/myservlet HTTP/1.0
MyServlet.class
Produit du code HTML
Decode et affiche la résponse HTML
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: text/html
Content-Length: 122
Date: Fri, 02 Nov 2012 14:40:51 GMT
<html>
<head>
<title>Hello Enib!</title>
</head>
<body>
<h1>Welcome in the Java server side!</h1>
</body>
</html>
Welcome in the Java server side!

La servlet


Server (Tomcat)
Global Naming Resources
Resource
JNDI
marakana
Listener
Service
Engine
Listener
Cluster
Realm
HTTP
HTTP
HTTP/S
AJP
HTTPS
HTTPS
Clients
Web Server
CONNECTOR
Thread Pool
Executor
Valve
Valve
Valve
Host
Listener
Cluster
Realm
Context
Web App
Filter
POJO
Servlet
Resource Link
Resource
Listener
Manager
Loader
Realm

La servlet

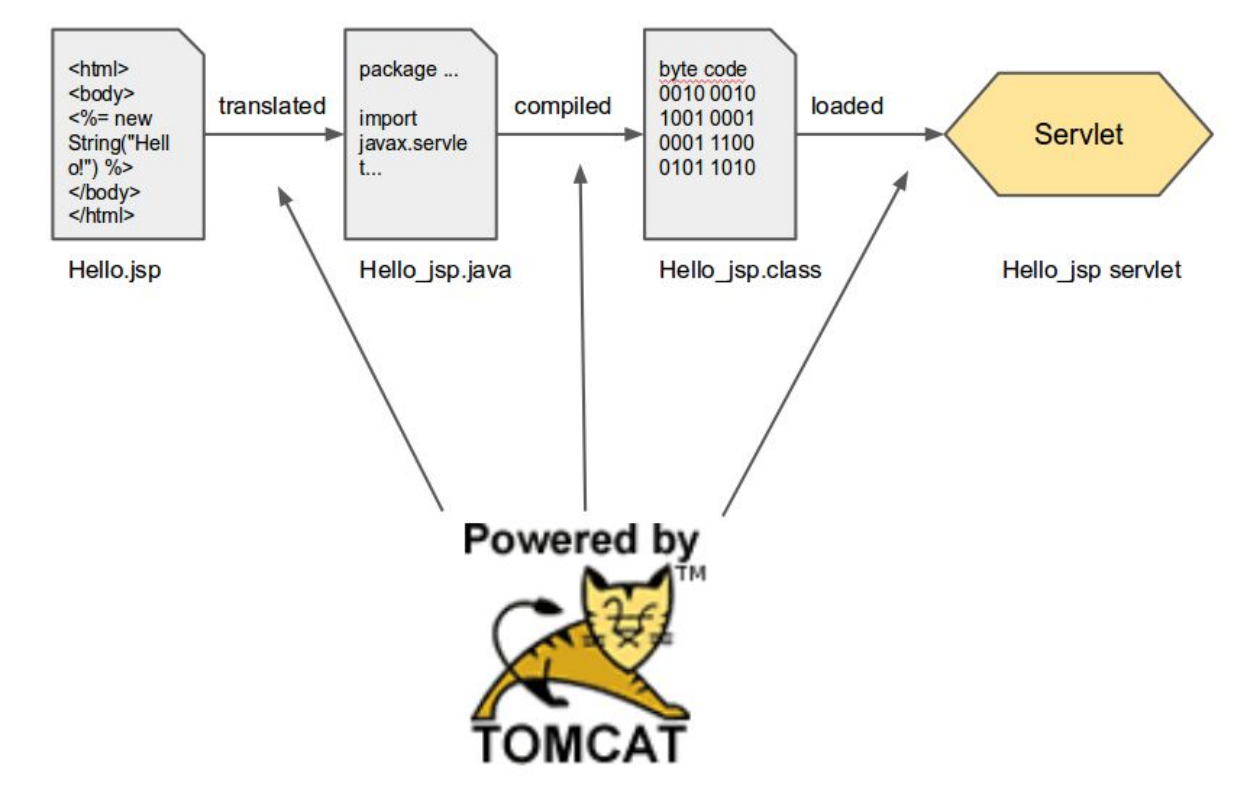

Web Container
Servlet Class
Servlet Object
Class Loading
BeerServlet.class
Instantiation (Default contructor)
The servlet should only have the default contsructor
init()
Servlet ready to work
Call once by the container
service()
Servlet working
Handle client HttpRequest
doGet() doPost()
each request in a separate Thread
destoy()
Servlet clean up
Call once by the container

La servlet

```
public class HtmlServlet extends HttpServlet {
     private static final long serialVersionUID = 1L;

     public HtmlServlet() {
               super();
     }

     protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
          doPost(request, response);
     }

     protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
          response.setContentType("text/html") ;
          PrintWriter out = response.getWriter() ;
          out.println("<html>") ;
          out.println("<head>") ;
          out.println("<title>Hello Enib!</title>") ;
          out.println("</head>") ;
          out.println("<body>") ;
          out.println("<h1>Welcome in the Java server side!</h1>") ;
          out.println("</body>") ;
          out.println("</html>") ;
    }
}
```

JSP

- JSP = JavaServer Pages
- Technologie Java crée par Sun
- Permettent la génération de pages web dynamiques.
- Mélange de Java côté serveur et d'HTML côté client.
- Adore se transformer !
- Adepte du ménage à 3 (la JSP n'existe pas sans container ni Servlet)



UglyScriptlet.jsp

```
<html>
    <body>
      Luke <br>
      <% out.println(plain.old.java.Reason.getText() );  %>
    </body>
</html>
```

Reason.java

```
package plain.old.java;

pubic class Reason {
    public static synchronized String getText() {
        return new String("Je suis ton père.");
    }
}
```

UglyScriptlet.jsp

```
<%@ page import="plain.old.java.*" %>
<html>
    <body>
        Luke<br>
        <%= (Reason.getText() );  %>
    </body>
</html>
```

Reason.java

```
package plain.old.java;

pubic class Reason {
    public static synchronized String getText() {
        return new String("Je suis ton père.");
    }
}
```

JSP + EL

UglyScriptlet.jsp

```
<html>
    <body>
        Luke<br>
        ${ugly.reason}
    </body>
</html>
```

Expression trop longue

```
<%= ((fr.enib.cai.core.Beer) request.getAttribute("beers").get(0).getName()
%>
```

C'est mieux, non ?

```
${beers[0].name}
```

Attention les objets implicites d'EL sont des Maps - pair(name / value)

```
pageScope - requestScope - sessionScope - applicationScope
param - paramValues
header - headerValue
cookie
initParam
```

JSP + JSTL

Java Standard Tag Library :

- 4 bibliothèques de tags utilisables dans les JSP
- Objectif : faciliter le développement des JSP
- Anéantir les scriptlets

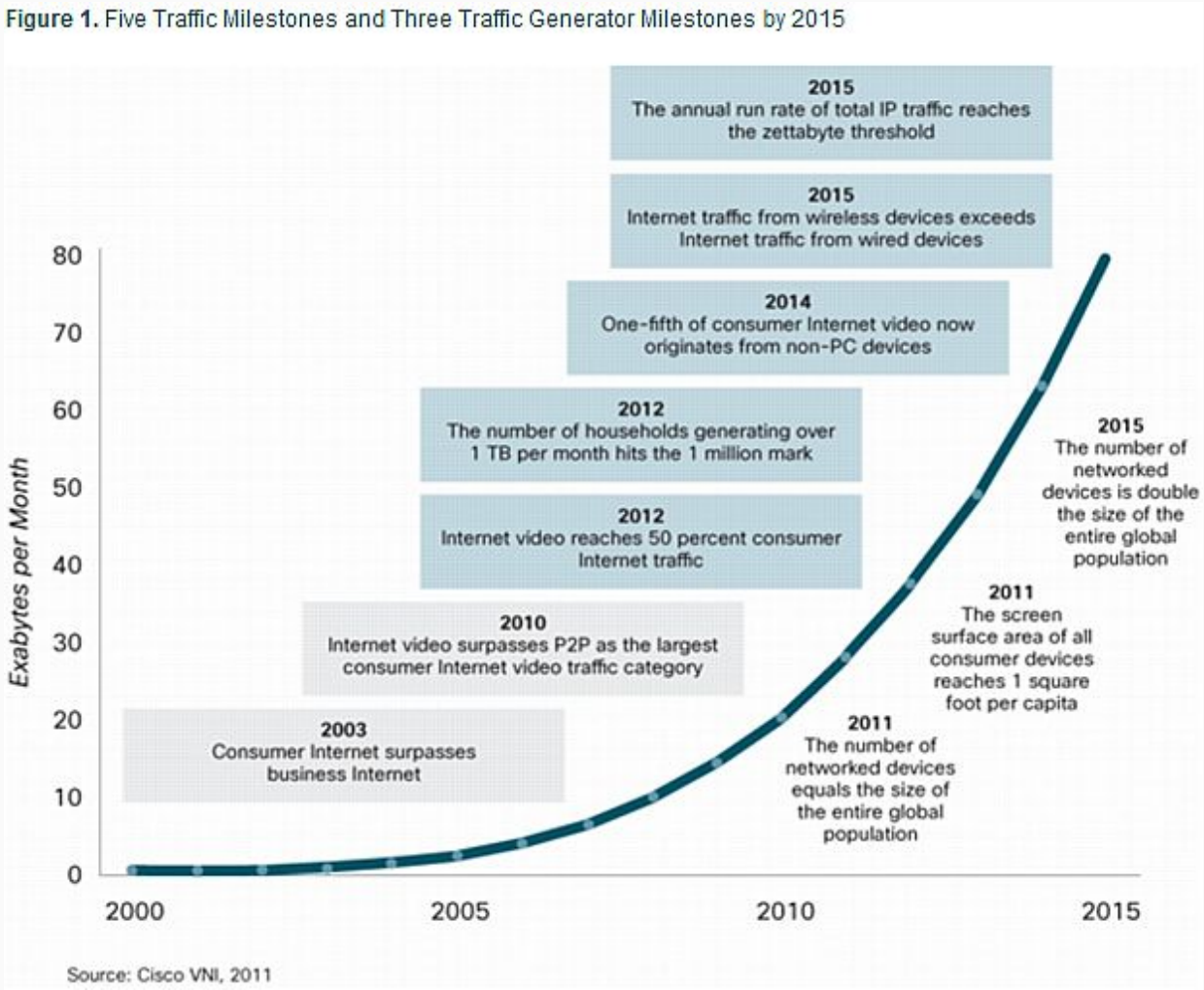
```
<c:out value='${user}' default='invité' />

<c:forEach var="beer" items="${beers}" >
     Beer name : ${beer.name}
</c:forEach>

<c:if test="${beer.name eq 'Karmeliet'}">
     Yes ${beer.name} have an incredible flavour !
</c:if>

<c:choose>
     <c:when test="${beer.name eq 'Karmeliet'}">
          Yes ${beer.name} have an incredible flavour !
     </c:when>
     <c:when test="${beer.name eq 'Rochefor 8'}">
          ${beer.name} is another best choice !
     </c:when>
     <c:otherwise>
          Dammed give me a Kro !
     </c:otherwise>
</c:choose>
```

Séparation de la logique métier (Servlet en Java) de la partie présentation dévolue aux Web-designers.


A Brain-Friendly Guide
2nd Edition
New Mock Exam Included
Head First
Servlets & JSP™
Passing the Sun Certified Web Component Developer Exam
Test yourself with more than 200 realistic exam questions
Updated to cover the latest version of the SCWCD exam for J2EE 1.5
BANG!
Watch it!
Avoid deadly traps & gotchas on the 1.5 exam
Use c:out to get your message to the world
Learn how Ted improved his appeal with dynamic attributes
Fool around in the Custom Tag Library
O'REILLY®
Bryan Basham, Kathy Sierra & Bert Bates

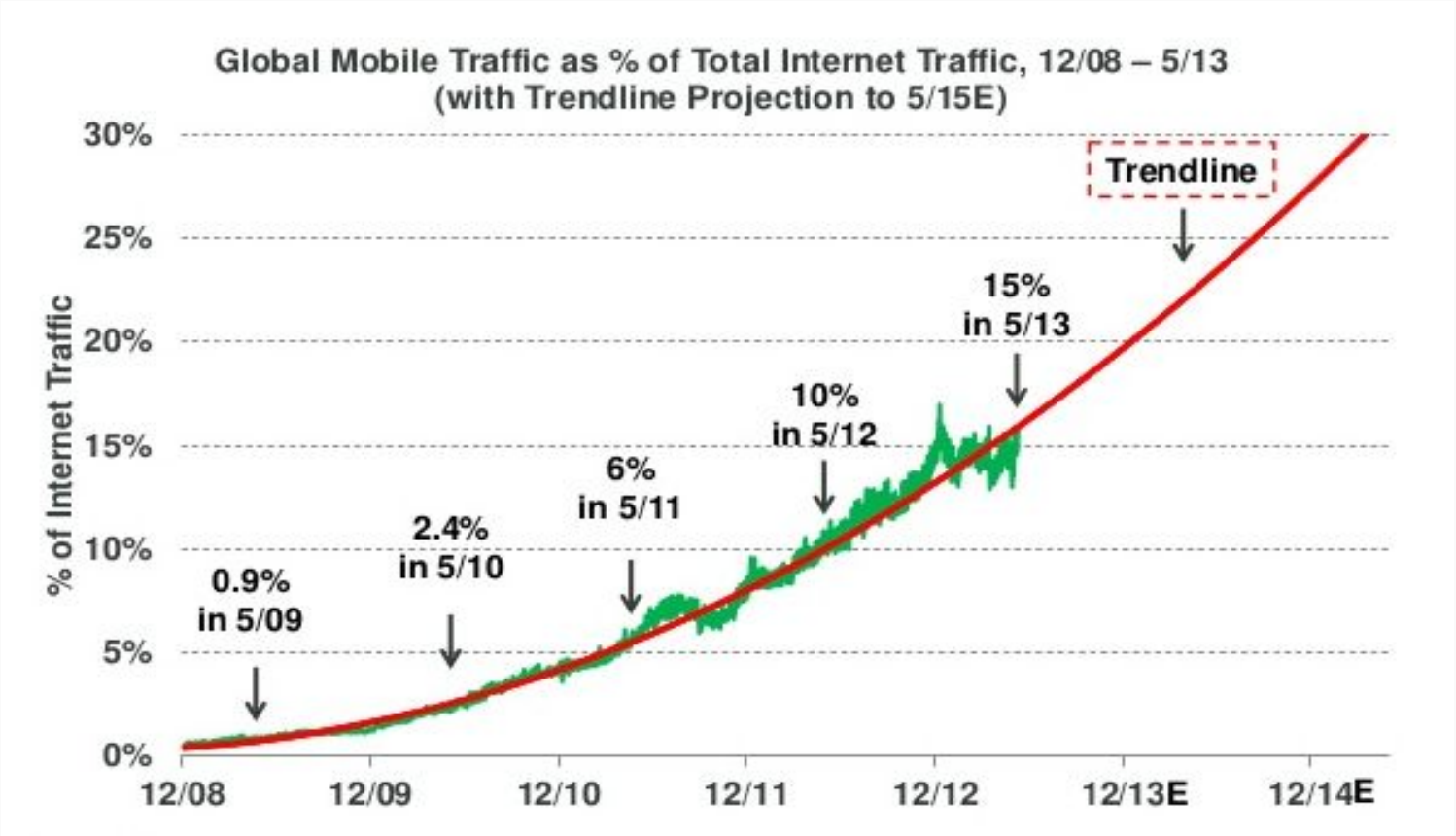
C10k

- Répondre à un grand nombre de clients en même temps
- C10k = 10 000 connexions simultanées



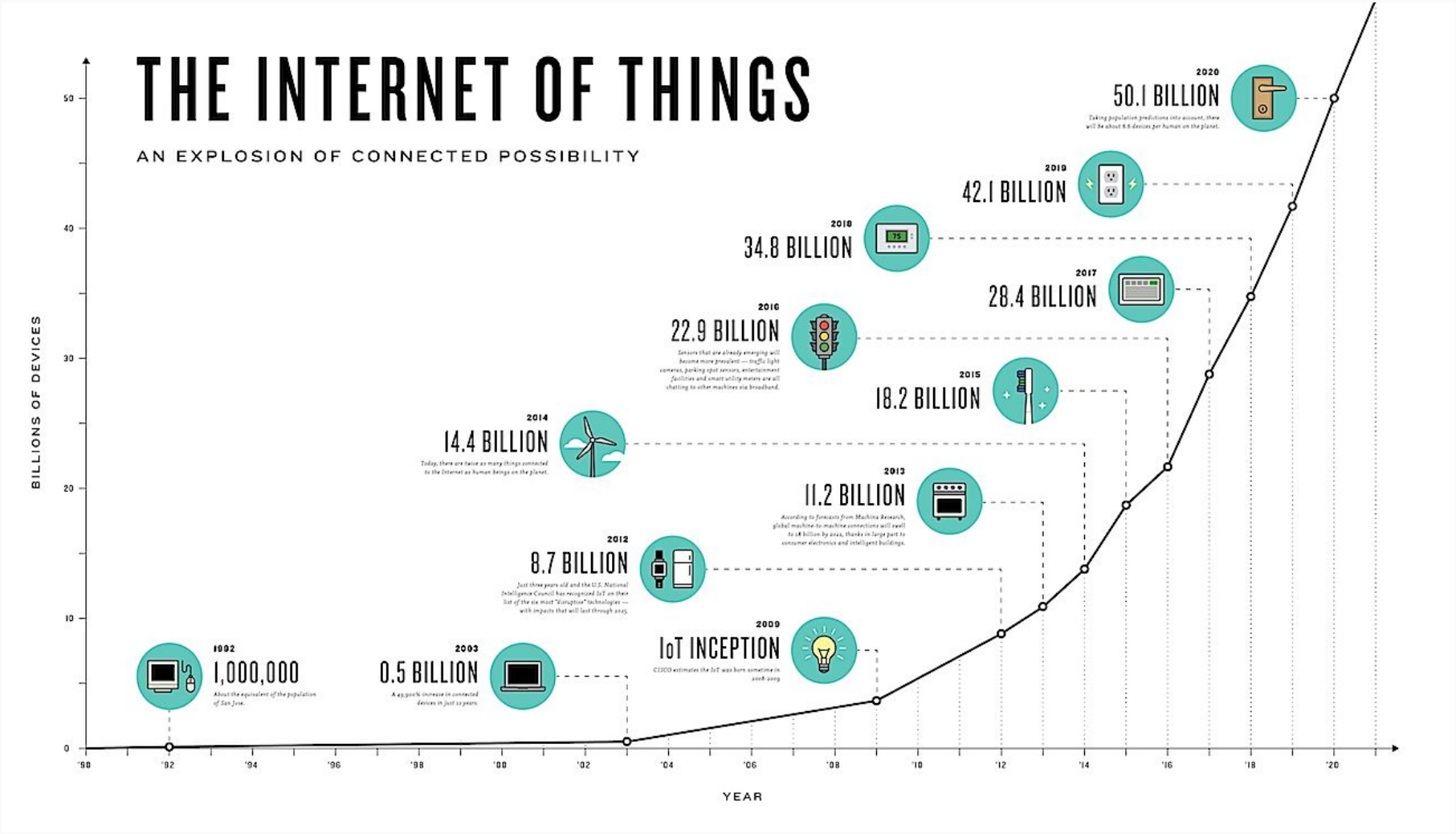
C10k

- Répondre à un grand nombre de clients en même temps
- C10k = 10 000 connexions simultanées



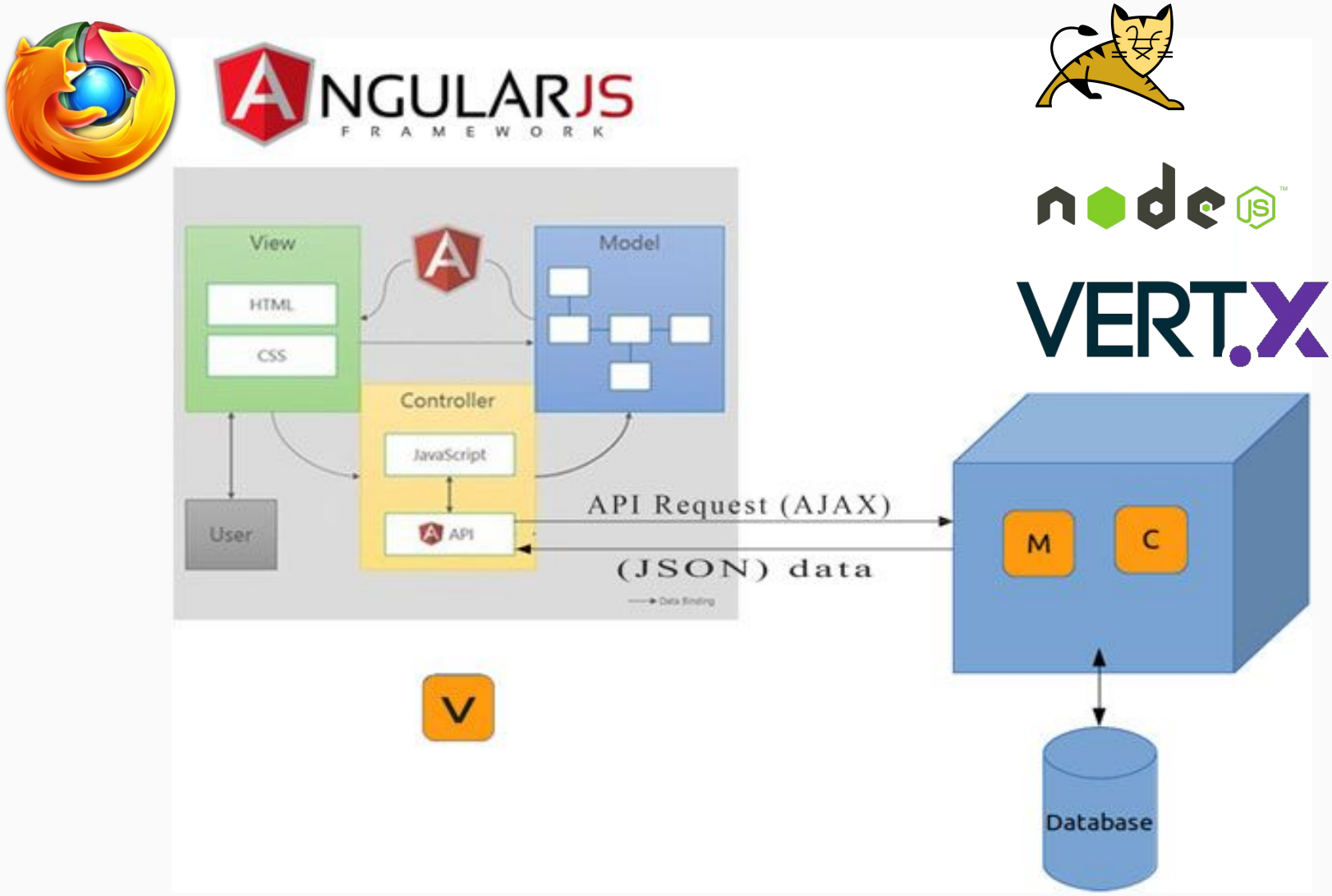
C10k

- Répondre à un grand nombre de clients en même temps
- C10k = 10 000 connexions simultanées



C10k

- Changement de paradigme



C10k

Vert.X

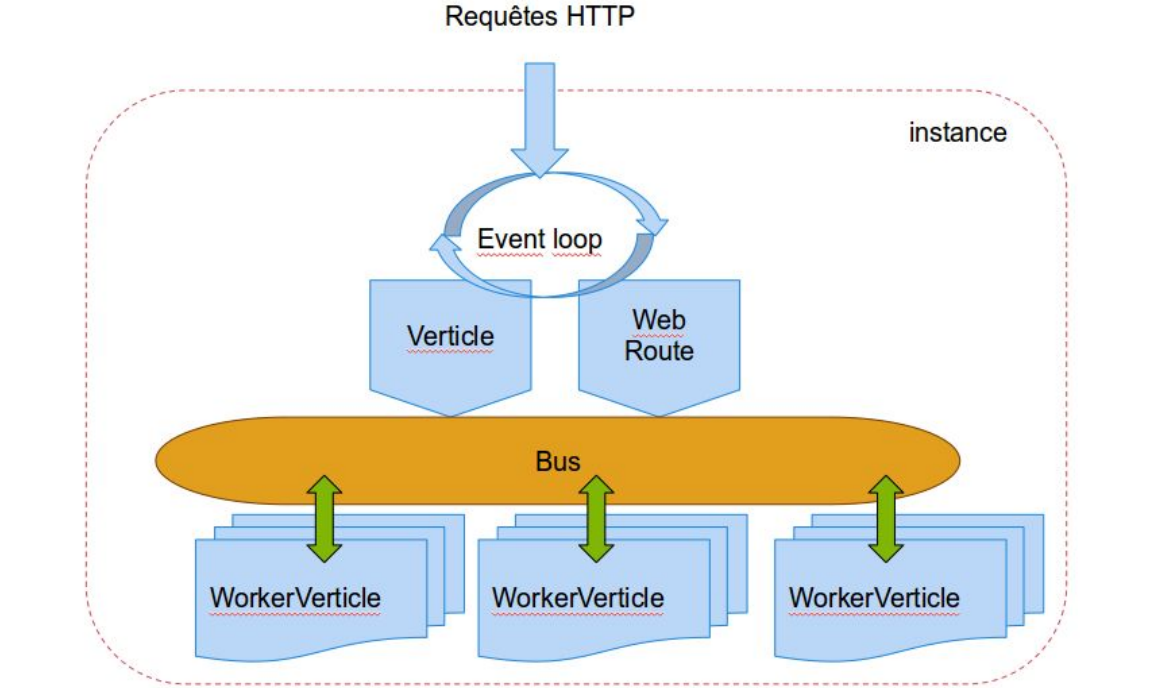
Simple threadSafe concurrent asynchronous eventDriven reactive eventBus over a scalable polyglot embeddable toolkit plateform

- Créé par Tim Fox @timfox
- en 2011 chez VMWare
- sous le nom de Node.x
- 2012 : devient Vert.X
- 2013 : passe dans la fondation Eclipse
- 24/06/2015 sortie de la v3
- environ 180 contributeurs

http://vertx.io/

Vert.X

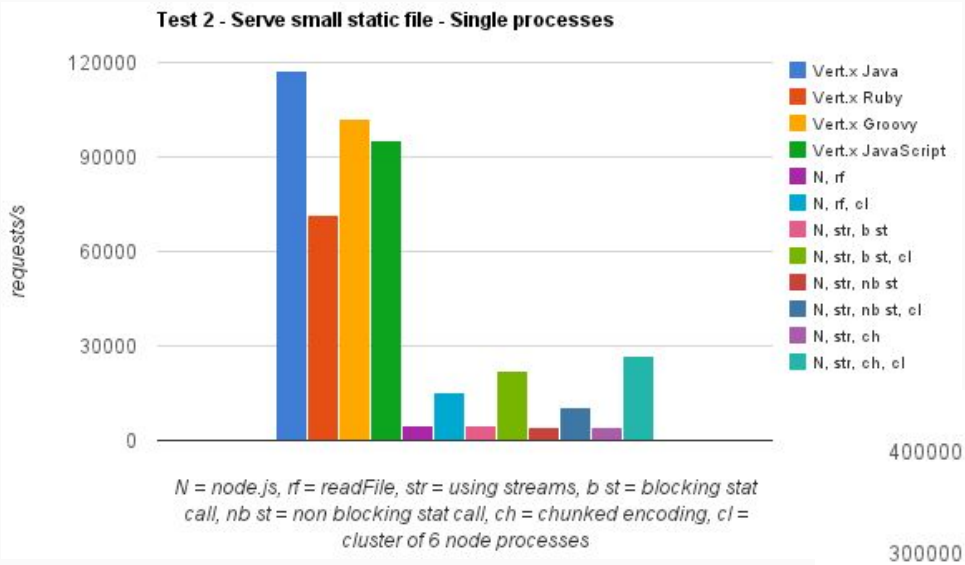
- Scalable verticalement et horizontalement
- Distribué
- Événementiel et asynchrone
- I/O non bloquantes
- Tolérant à la panne
- Extensible



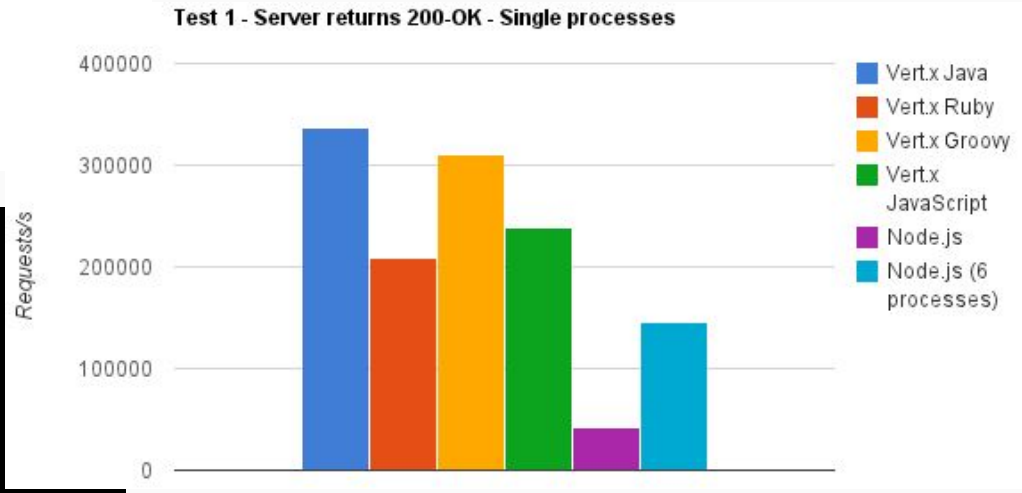
Vert.X

- Vert.x core
  - Clients / Serveurs TCP/SSL
  - Clients / Serveurs HTTP/HTTPS
  - REST et Multipart
  - Websockets et Sock.js
  - Accès distribué de l'Event-bus
  - Maps et Sets partagés
  - Logging
- Vert.x Web
  - Routages et sous-routages
  - Sessions
  - Cookies
  - Sécurité (basic auth, shiro, JWT, …)
  - Templates (Handlebars, Jade, MVEL, Thymeleaf)
  - SockJS
  - Static files
  - …
- Accès aux données
  - MongoDB, JDBC, Redis, SQL Common
- Intégration
  - Mail et JCA
- Sécurité
  - basic auth, shiro, JWT, JDBC Auth
- Reactive
  - Vert.X Rx, Reactive streams
- Metrics
- Vert.X unit

Vert.X


Test 2 - Serve small static file - Single processes
120000
90000
60000
30000
0
requests/s
Vert.x Java
Vert.x Ruby
Vert.x Groovy
Vert.x JavaScript
N, rf
N, rf, cl
N, str, b st
N, str, b st, cl
N, str, nb st
N, str, nb st, cl
N, str, ch
N, str, ch, cl
N = node.js, rf = readFile, str = using streams, b st = blocking stat call, nb st = non blocking stat call, ch = chunked encoding, cl = cluster of 6 node processes


DARK
SIDE


Test 1 - Server returns 200-OK - Single processes
400000
300000
200000
100000
0
Requests/s
Vert.x Java
Vert.x Ruby
Vert.x Groovy
Vert.x JavaScript
Node.js
Node.js (6 processes)

SparkJava

SparkJava

build.gradle

```
dependencies {
    compile 'com.sparkjava:spark-core:2.3'
    testCompile group: 'junit', name: 'junit', version: '4.+'
}
```

HelloWorld.java

```
import static spark.Spark.*;

public class HelloWorld {
     public static void main(String[] args) {
          get("/hello", (req, res) -> "Hello World");
     }
}
```

Lambda Java 8

$ java org.giwi.HelloWorld

http://localhost:4567/hello

SparkJava

Une route Spark =

- un verbe
- un path
- un callback

```
get("/", (request, response) -> {
    // .. Show something ..
});

post("/", (request, response) -> {
    // .. Create something ..
});

put("/", (request, response) -> {
    // .. Update something ..
});

delete("/", (request, response) -> {
    // .. annihilate something ..
});

options("/", (request, response) -> {
    // .. appease something ..
});
```

```
// matches "GET /hello/foo" and "GET /hello/bar"
// request.params(":name") is 'foo' or 'bar'
get("/hello/:name", (request, response) -> {
    return "Hello: " + request.params(":name");
});
```

```
// matches "GET /say/hello/to/world"
// request.splat()[0] is 'hello'
// and request.splat()[1] 'world'
get("/say/*/to/*", (request, response) -> {
    return "Number of splat parameters: "
     + request.splat().length;
});
```

Les routes sont testées/exécutées dans leur ordre de déclaration

SparkJava

- Requêtes

```
request.attributes();
request.attribute("foo");
request.attribute("A", "V");
request.body();
request.bodyAsBytes();
request.contentLength();
request.contentType();
request.contextPath();
request.cookies();
request.headers();
request.headers("BAR");
request.host();
request.ip();
request.params("foo");
request.params();
request.pathInfo();
request.port();
request.protocol();
request.queryMap();
```

```
request.queryMap("foo");
request.queryParams();
request.queryParams("FOO");
request.queryParamsValues("FOO")
request.raw();
request.requestMethod();
request.scheme();
request.servletPath();
request.session();
request.splat();
request.uri();
request.url();
request.userAgent();
```

http://sparkjava.com/documentation.html#routes

SparkJava

- Réponses

```
response.body("Hello");
response.header("FOO", "bar");
response.raw();
response.redirect("/example");
response.status(401);
response.type("text/xml");
```

- Sessions

```
request.session(true)
request.session().attribute("user")
request.session().attribute("user", "foo")
request.session().removeAttribute("user")
request.session().attributes()
request.session().id()      request.
session().isNew()
request.session().raw()
```

- Filtres

```
before((request, response) -> {
    boolean authenticated;
    // ... check if authenticated
    if (!authenticated) {
        halt(401, "You are not welcome here");
    }
});

before("/protected/*", (request, response) -> {
    // ... check if authenticated
    halt(401, "Go Away!");
});

after((request, response) -> {
    response.header("foo",
     "set by after filter");
});
```

http://sparkjava.com/documentation.html#filters

SparkJava

- Exceptions

```
get("/throwexception", (request, response) -> {
    throw new NotFoundException();
});

exception(NotFoundException.class, (e, request,
response) -> {
    response.status(404);
    response.body("Resource not found");
});
```

- Fichiers statiques

```
staticFileLocation("/public");
externalStaticFileLocation("/var/www/public");
```

SparkJava

- Templates : http://sparkjava.com/documentation.html#views-templates
  - Freemarker
  - Mustache
  - Velocity
  - Thymeleaf
  - Handlebars
  - Jade
  - Jetbrick
  - Pebble
  - Water

```
public class FreeMarkerExample {
    public static void main(String args[]) {
        get("/hello", (request, response) -> {
            Map<String, Object> attributes = new HashMap<>();
            attributes.put("message", "Hello World!");
            // The hello.ftl file is located in directory:
            // src/test/resources/spark/template/freemarker
            return new ModelAndView(attributes, "hello.ftl");
        }, new FreeMarkerEngine());
    }
}
```

« Si vous avez compris ce que je viens de vous dire, c'est que je me suis probablement mal exprimé »

A. Greenspan

https://github.com/Giwi/Sparkjava-Beers
https://github.com/Giwi/vertx3-angular-beers