

Brazil-1_1.jar

<http://www.experimentalstuff.com>
douin@cnam.fr

Juillet 2002

http://jfod.cnam.fr/tp_cdi/douin/

Lecture préalable de la présentation de Stephen Uhler
en http://jfod.cnam.fr/tp_cdi/uhler/java-one.pdf

serveur Web de nouvelle génération ?

Ce support accompagne les TPs 11 et 12
en http://jfod.cnam.fr/tp_cdi/

Bibliographie

- [Bra00] www.sun.com/research/brazil
- [Bra01] www.ExperimentalStuff.com, <http://www.brazilhandlers.com:9090>
- [Dig99] R. DiGiorgio. An introduction to the URL programming interface. http://www.javaworld.com/javaworld/jw-08-1999/jw-09-javadev_p.html
- R. DiGiorgio. <http://www.javaworld.com/javaworld/jw-08-2000/jw-0811-javadev.html>
 - Part 1: Learn how to build an application server that can deliver data to clients requiring different protocols(08/2000)
 - Part 2: How to support XML applications with the Brazil project(10/2000)
 - Part 3: Economically sustain PQA, UP.SDK, and J2ME with the Brazil project(01/2001)
 - Part 4: Build multicast-aware apps with JRMS(04/2001)
 - Part 5: Manage users and content with Brazil(08/2001)
 - Part 6: Plug Jini, BeanShell, and JAXM into Brazil (07/2002)
- [DSU00] R. DiGiorgio, C. Stevens, S. Uhler. How Handlers work in Web-accessible home automation. http://www.javaworld.com/javaworld/jw-05-2000/jw-0505-javadev_p.html

et ce livre sur les serveurs Web en Java

- [RS00] P. Rossbach & H. Schreiber. Java Server & Servlets, Building portable Web applications. Addison Wesley 2000. <http://www.webapp.de>

les différents essais ont été réalisés sur la carte TINI

- <http://www.ibutton.com>

HTML

- <http://www.w3.org/Protocols/Specs.html>, <http://www.ietf.org/rfc/rfc2616.txt>

Avec un moteur de recherche voir : UDDI et WSDL

Brazil: An extensible Java Web server

By Rinaldo Di Giorgio

- HTTP servers are the single most important component of the content delivery system for the Web. We have seen three generations of HTTP servers:
 - Initial CERN servers
 - Netscape, Microsoft, Apache
 - Java HTTP servers and Brazil
- We have made few advances in creating one size that fits all HTTP server products. Brazil, from Sun Labs, is a third-generation HTTP server. It uses well-established software engineering principles to support small, modular objects called *handlers*. Handlers implement specific abilities like file services and security services. Over 50 handlers implement a broad spectrum of services that developers can reuse. The Brazil architecture provides an extensible yet understandable framework for a developer looking to build small and possibly application-specific HTTP servers without incurring the high learning curves and administration costs of current servers. The Brazil architecture can scale to meet different requirements because application developers can extend the architecture independently. Brazil prototypes are running on large computers as HTTP servers, on workstations and PCs as personal Web servers, and on embedded computers such as TINI for HTTP-addressable devices and appliances. The goal is for devices like PDAs and cell phones to also service HTTP requests.

Sommaire

- Rappels : serveurs Web en Java, servlets
- Brazil et ses 3 composantes Handler, Filter et Template
- Une orientation : Pattern et Brazil
 - Le pattern Observateur
 - Le Pattern Chaîne de responsabilités
 - Le Pattern Composite et le Pattern Visiteur
 - Et plus si ...

B·R·A·Z·I·L



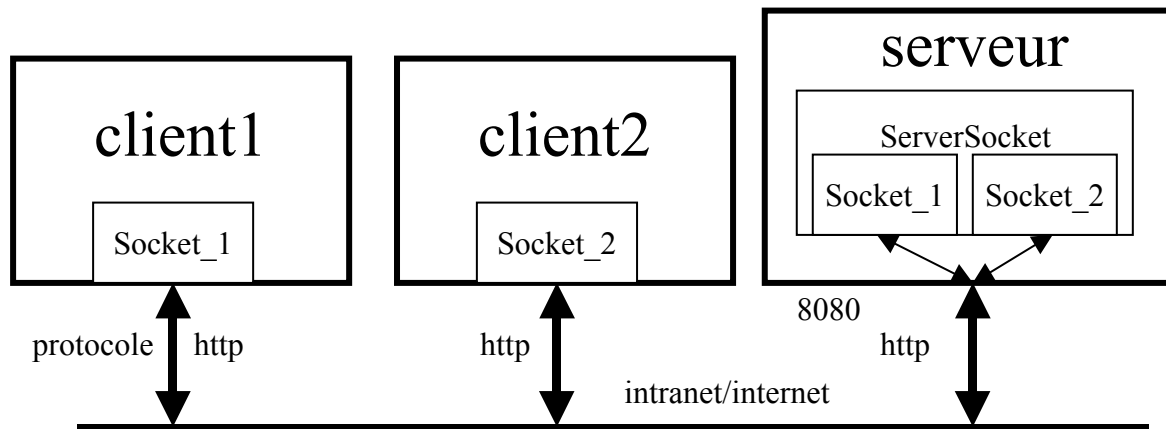
kiosk

TV

Introduction

- Brazil : Objectifs
 - Initialement : serveur web pour cartes à puces
 - Maintenant :
 - boîte à outils pour la construction de serveurs Web
- Trois composantes
 - Handler : serviette légère
 - Filter : filtrage des transactions HTTP
 - Template : interprétation des étiquettes <IMG, <DEBUG, <A, <SQL, <xxx
par le serveur à la lecture du document HTML/XML/...

Serveur Web en Java,(rappel)



```
import java.net.Socket;  
import java.net.ServerSocket;
```

Schéma d'un serveur en Java(1)

```
import java.net.*;
import java.io.*;
public class ServeurWeb_UneSeuleRequete{

    public static void main(String [] args) throws IOException{

        ServerSocket serveur = new ServerSocket(8080);
        Socket s = serveur.accept();

        try{
            PrintStream out;
            out = new PrintStream(s.getOutputStream());
            BufferedReader in;
            in = new BufferedReader(new InputStreamReader(s.getInputStream()));
            String req = in.readLine();

            // traitement de la requête selon le protocole HTTP
            ...
        }
    }
}
```

Schéma d'un serveur en Java(2)

```
public class ServeurWeb{ // un Thread à chaque requête

    public static void main(String [] args) throws IOException{
        ServerSocket serveur = new ServerSocket(8080);
        while(true){
            Connexion session = new Connexion(serveur.accept());
        }
    }

    private class Connexion extends Thread{
        private Socket s;
        public Connexion(Socket s){this.s = s; this.start();}
        public void run(){
            PrintStream out = new PrintStream(s.getOutputStream());
            BufferedReader in;
            in = new BufferedReader(new InputStreamReader(s.getInputStream()));
            String req = in.readLine();

            // traitement de la requête selon le protocole HTTP
```

Traitement d'une requête

```
StringTokenizer st = new StringTokenizer(req);
if((st.countTokens()>=2) && st.nextToken().equals("GET")){
try{String name = st.nextToken();
    if (name.startsWith("/")) name = name.substring(1);
    File f = new File(name);
    DataInputStream fis = new DataInputStream(new FileInputStream(f));
    byte[] data = new byte[(int)f.length()]; fis.read(data);

    out.print("HTTP/1.0 200 OK\r\n");
    out.print("Content-Type: text/html \r\n ");
    out.print("Content-Length: " + f.length() + "\r\n\r\n");
    out.write(data);
}catch(FileNotFoundException e){
    out.print("HTTP/1.0 404 Not Found\r\n");
    out.print("Content-Type: text/html\r\n");
    String str = "<HTML><BODY><H1>" + " Ce fichier n'existe pas !" +
        "</H1></BODY>\r\n\r\n";
    out.print("Content-Length: " + str.length() + "\r\n\r\n");
    out.print(str);
    ...
}
```

Envoi d'une requête (un client)

```
Socket socket = new Socket("lmi92.cnam.fr",8080);
OutputStream out = socket.getOutputStream();

String command = new String("GET " + file + " HTTP/1.0\r\n\r\n");
out.write(command.getBytes());
out.flush();

InputStream in = socket.getInputStream();
int aByte;
// We are writing byte wise, to make it simpler. In a real application,
// this is *not* recommended.

while((aByte = in.read()) != -1){
    System.out.write(aByte);
}

out.close();
in.close();
socket.close();
```

Trace d 'exécution(lmi92.cnam.fr:8080)

HTTP/1.0 200 OK

Last-Modified: Tue, 22 Jan 2002 14:13:12 GMT

Date: Fri, 25 Jan 2002 09:29:59 GMT

Server: Brazil/1.0

Connection: close

Content-Length: 6038

Content-Type: text/html

<HTML>

<HEAD>

<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<META name="GENERATOR" content="Microsoft FrontPage Express 2.0">

<TITLE>JFOD Cnam uv 16472 et uv 16392</TITLE>

</HEAD>

<BODY bgcolor="#E9FEC2">

<P align="center">

Formation Ouverte et à Distance
du Cnam

uv16472 et uv16392

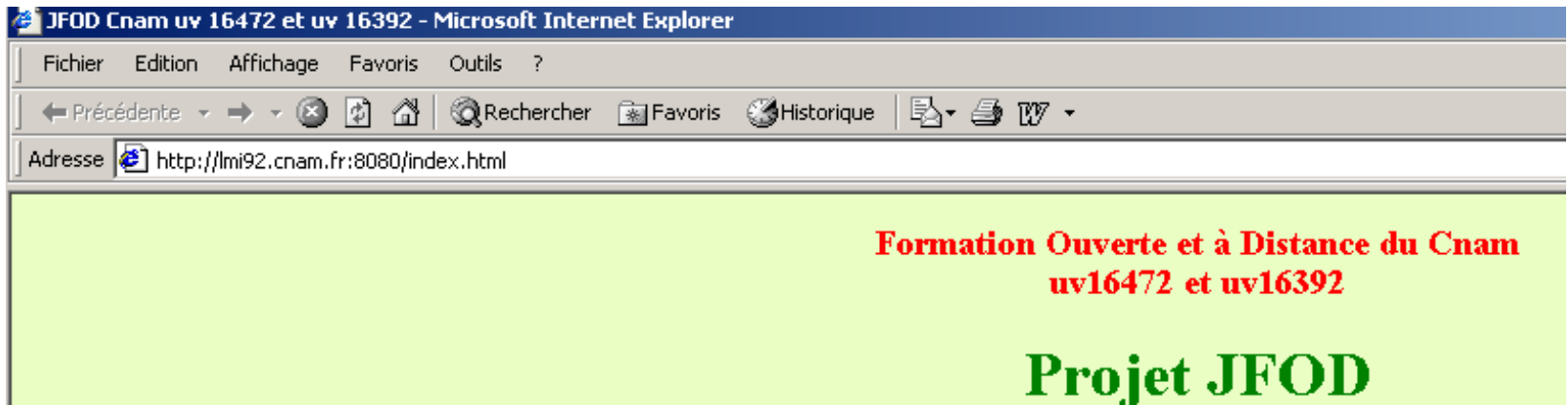
<P align="center">

.....

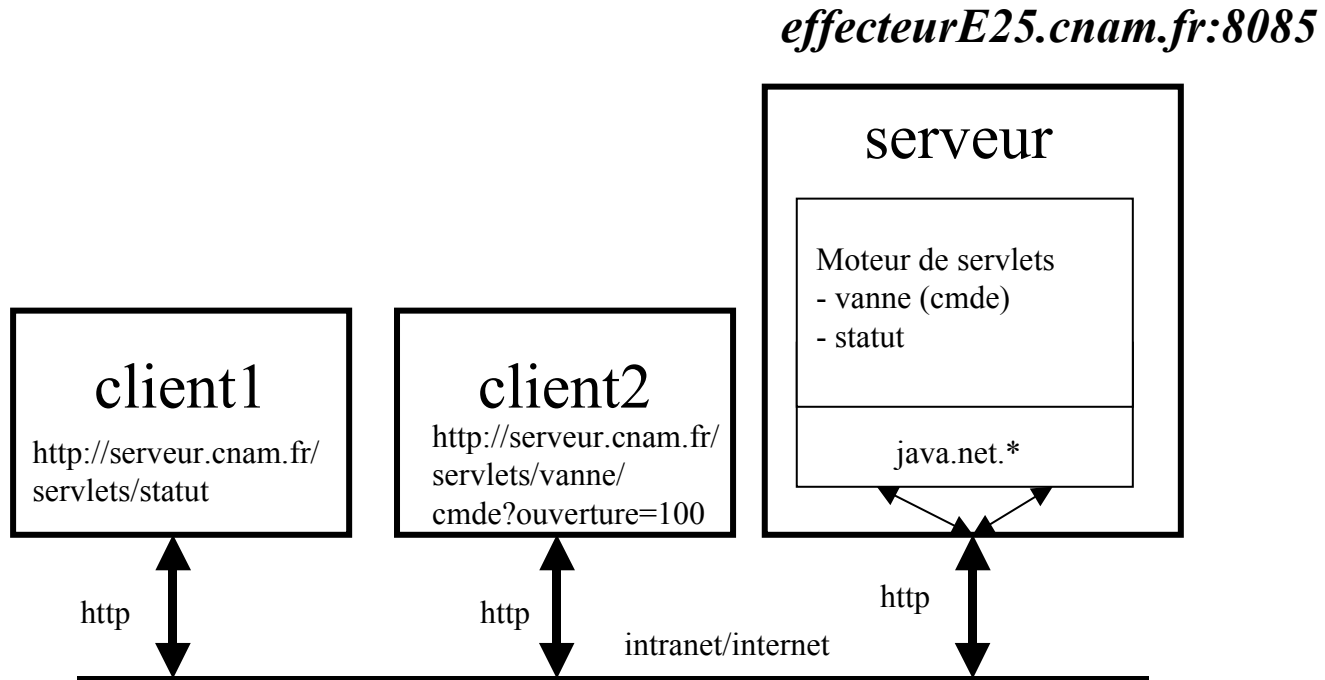
.....

Etc

Envoi d'une requête depuis un navigateur (methode HTTP/GET)



Serveur Web + moteur de Servlets



`http://effecteurE25.cnam.fr:8085/servlets/Vanne?ouverte=100`

Servlete : principes

Le Scénario

- Le client envoie une requête HTTP au serveur
- Le serveur charge la servlete en mémoire et crée un « Thread » pour celle-ci, la servlete devient résidente jusqu'à ce que le serveur s'arrête
- Le serveur transmet alors la requête à la servlete
- La servlete construit dynamiquement une réponse retournée au serveur
- Le serveur envoie cette réponse au client

- Plusieurs clients en même temps d'une même servlete
- Une servlete peut transmettre la requête à d'autres servletes ou serveurs

Servlete : principes 2

Comment ?

- La servlete est référencée par une URL
- à l'aide du JSDK (Java Servlet Development Kit)
- Java Servlet API
 - `javax.servlet`
 - `javax.servlet.http`
- La servlete hérite de la classe `HttpServlet`
 - les méthodes :

`doGet` HTTP GET

`doPost` HTTP POST

`doPut` HTTP PUT

`doDelete` HTTP DELETE

`init, destroy, getServletInfo, service`

Structure d'une servlet

<http://effecteurE25.cnam.fr:8085/servlets/Vanne?ouverte=100>

```
public class Vanne extends javax.servlet.HttpServlet{  
  
    public void doGet( HttpServletRequest req, HttpServletResponse res){  
        // extraction du type de la requête souhaitée par le client, éventuellement précédée  
        // d'une identification  
        // traitement de cette requête  
        // résultats éventuels envoyés en protocole HTTP vers le client  
    }  
  
    public void doPost( HttpServletRequest req, HttpServletResponse res){  
        // requête HTTP-POST
```

Moteur de servlets et JSP

- JSP : développement de pages hybrides HTML/Java
- Compile et exécute le source comme "servlet" pour tous les fichiers *.jsp
 - le résultat (servlet source et classe) est placée dans un répertoire du système
 - Après chaque mise à jour d'un fichier *.jsp la classe servlet est recompilée

Servlets et JSP : deux références pour en savoir beaucoup plus

<http://tecfa.unige.ch/guides/tie/html/java-jsp/java-jsp.html>

<http://www.research.umbc.edu/~tarr/jst/spr02/jst.html>

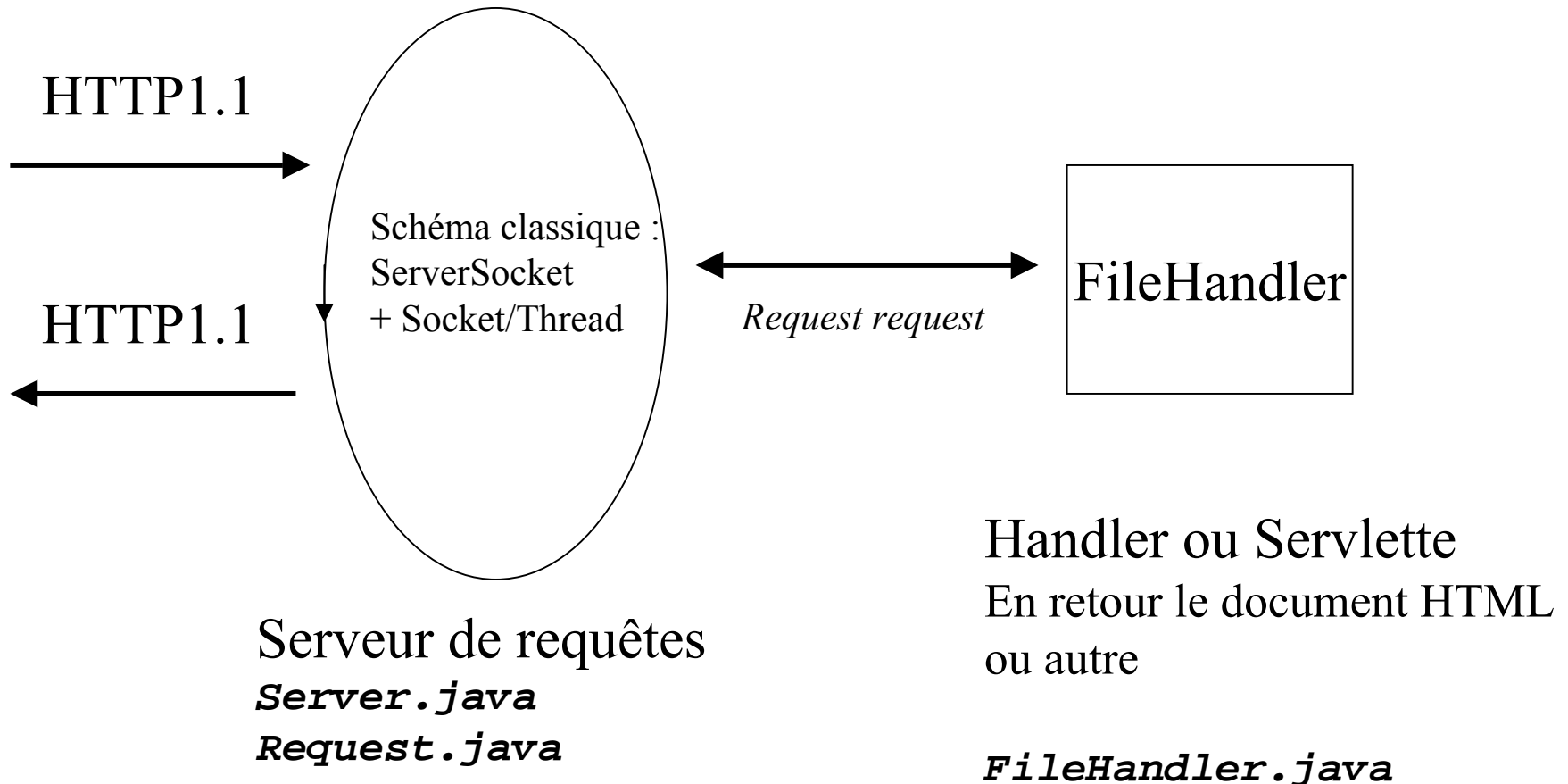
<http://dept25.cnam.fr:8080/PROJET3I/DOC/servlets-JSP-14.11.2001.pdf>

- *donc : développement en JSP plutôt qu'en servlet*
- *alors : peu adapté aux systèmes embarqués/enfouis*

Objectifs de Brazil

- Serveur Web de petite taille
 - Objet,
 - nombreuses fonctionnalités,
 - extensible,
 - sources Java,
 - Porté sur TINI(www.ibutton.com) (45 Ko de code),
- Les mêmes notions de Servlets
 - Mais légères ... (appelées Handler)
 - Avec des filtres ... (sur les transactions)
 - Et des Templates ...(interprétation de balises)

Schéma du serveur minimal Brazil



Le serveur Web de Brazil1.1

- Le paquetage *sunlabs.brazil.server*
- *Main.java* est une implémentation par défaut
FileHandler.java : lecture de documents
 - Le serveur Web, protocole HTTP1.1

```
>java -cp brazil-1_1.jar sunlabs.brazil.server.Main -c config.txt
```

– le fichier de configuration

```
## config.txt
```

```
port=9090
```

```
handler=file
```

```
root=.
```

```
# parametres de FileHandler associés au nom file
```

```
file.class=sunlabs.brazil.server.FileHandler
```

```
file.default=index.html
```

Handler : principes⁽¹⁾

Au démarrage du serveur

- Initialisation du Handler : chargement de la classe à partir du fichier de configuration (appel d'une méthode *init*) une seule fois.
 - Lecture des paramètres globaux du fichier de configuration.

Handler : principes(2)

A chaque requête

- Le client envoie une requête HTTP au serveur Brazil
 - Cette requête contient éventuellement des paramètres locaux,
 - Une instance de la classe Request est créée
- Le serveur transmet cette instance au handler, par un appel de méthode (*boolean respond(Request request)*)
 - Extraction des paramètres, méthodes d'instance de la classe Request

Plusieurs Handlers Brazil

- Les handlers peuvent être chaînés,
 - En utilisant un Handler de chaînage
- Au premier handler de la chaîne retournant la valeur booléenne « true » le parcours est interrompu.
 - C'est une instance du Pattern chaîne de responsabilités

```
import sunlabs.brazil.server.ChainHandler;
```

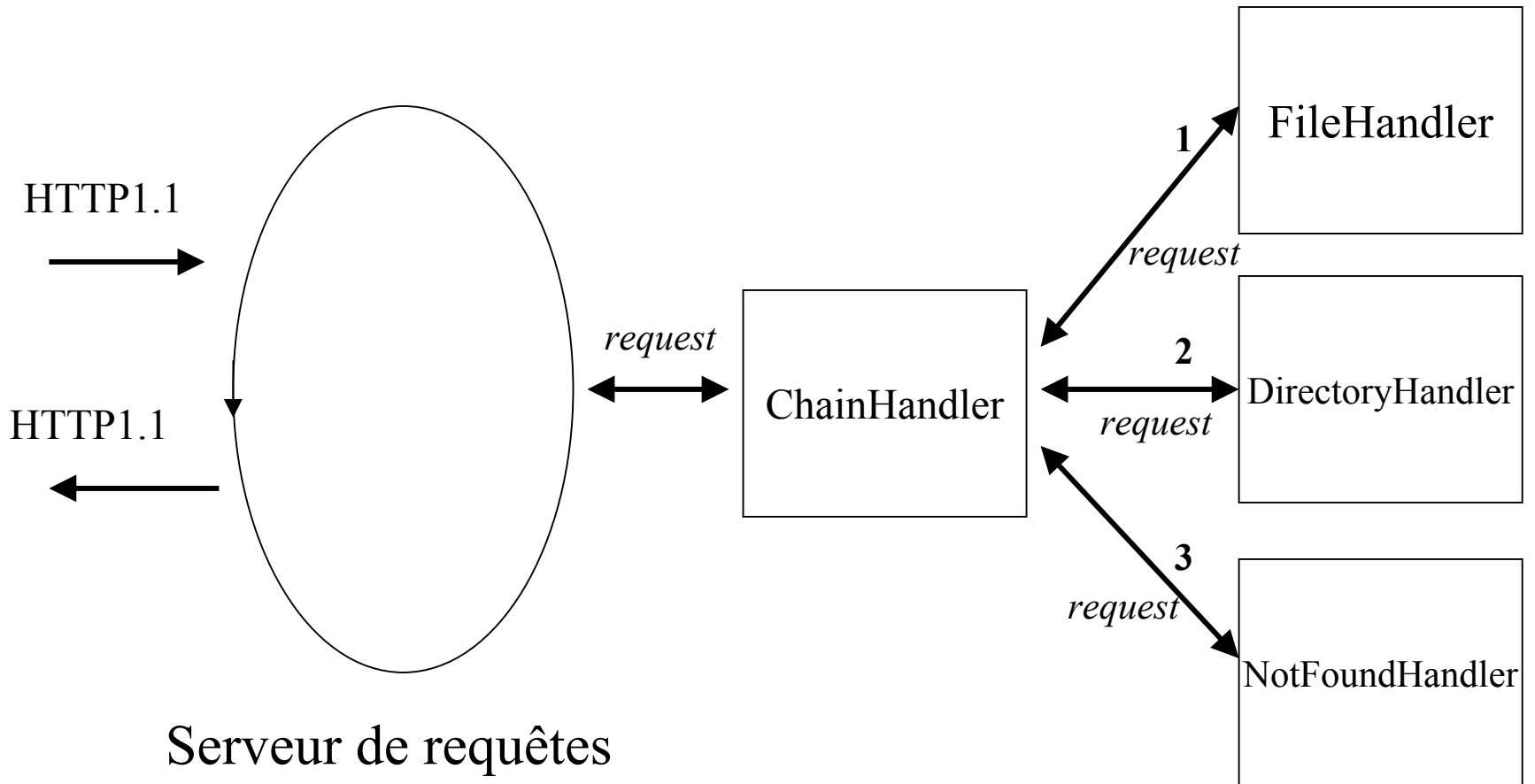
partage de données possible entre deux handlers

Variable globale, Paramètres HTTP, Appels de handler...

(en exemple la donnée d 'instance server.props

(voir en fin de présentation l'exemple SqlTemplate))

Schéma du serveur Brazil



Serveur de requêtes
Server.java
Request.java

Les Handlers sont chaînés :1 puis 2 puis 3
sunlabs.brazil.handler.DirectoryHandler
sunlabs.brazil.handler.NotFoundHandler

Les 3 composantes de Brazil

- *L'interface sunlabs.brazil.server.Handler*
- *L'interface sunlabs.brazil.filter.Filter*
- *La classe sunlabs.brazil.template.Template*

L'interface Handler

```
package sunlabs.brazil.server;  
  
public interface Handler{  
    // méthode déclenchée au chargement du Handler  
    // @param server le serveur initiateur  
    // @param prefix le préfixe extrait du fichier de configuration  
    boolean init(Server server, String prefix);  
  
    // méthode déclenchée à chaque requête  
    // @param request une transaction HTTP  
    boolean respond(Request request) throws IOException;  
  
}
```

La classe Request.java, un extrait

```
public class Request{
    . . .
    Request(Server server, Socket sock)    {
        this.server = server;
        this.sock = sock;
        try {
            in = new HttpInputStream(new
                BufferedInputStream(sock.getInputStream()));
            out = new HttpOutputStream(new
                BufferedOutputStream(sock.getOutputStream()));
        } catch (IOException e) {
            }
        ...}
}
```

La classe Request.java, (2)

extraction des paramètres HTTP

- ```
public Hashtable getQueryData(Hashtable table) {
 if (table == null) { table = new Hashtable(); }
 HttpUtil.extractQuery(query, table);
 if (postData != null) {
 String contentType = headers.get("Content-Type");
 if ("application/x-www-form-urlencoded".equals(contentType)) {
 HttpUtil.extractQuery(new String(postData), table);
 }
 }

 return table;
}
```
- ```
public void    sendResponse(String body) throws IOException    {
    sendResponse(body, "text/html", -1);
}
```

La classe Request : utilisation

```
public boolean respond(Request request) ...  
// les paramètres locaux de l'URL http://.../vanne/  
// http://adresseIP:port/vanne/?cmde=ouvrir&valeur=1000  
  
• Hashtable t = request.getQueryData( );  
  // t.toString() == {valeur=1000,cmde=ouvrir}  
  
// en retour  
request.sendResponse("<B>ouverture r&eacute;ussie</B>");  
  
return true; // arrête le chaînage si ChainHandler
```

Lecture des paramètres globaux

- lecture du fichier de configuration
- au format d'une `java.util.Properties`

Voir l'usage : `void load(InputStream inStream);`

– `cle=valeur`

– `cle1=valeur1`

– `xxxx=yyyy`

– `file.class=sunlabs.brazil.server.FileHandler`

– `file.default=index.html`

Le fichier de configuration

- Est affecté au champ props de l'instance server

```
String s = server.props.getProperty(propsPrefix + "name");
```

```
// si propsPrefix = "clock." alors s vaut "/clock/"
```

```
// cela correspond à la sélection du bon handler...
```

```
#un extrait du fichier de configuration
```

```
clock.class=cnam.tpcdi.ClockHandler
```

```
clock.name=/clock/
```

Structure typique d'un Handler

```
public class UnNomDe_Handler implements Handler {
    private String propsPrefix;
    private Server server;
    private final String nomDuParametreConfiguration="name";
    private String name = "/default/";

    // Exécutée une seule fois, lecture des paramètres globaux
    // concernant ce Handler
    // depuis le fichier de configuration

    public boolean init(Server server, String prefix) {
        this.server = server;
        this.propsPrefix = prefix;

        // lecture des paramètres
        name = server.props.getProperty(propsPrefix + nomDuParametreConfiguration);
        ...

        return true;
    }
}
```

Structure typique d'un Handler

```
public class UnNomDe_Handler implements Handler {
    private String propsPrefix;
    private Server server;
    private final String nomDuParametreConfiguration="name";
    private String name = "/default/";

    public boolean init(Server server, String prefix) { ...}

    // Sélection du Handler, et lecture des paramètres locaux
    public boolean respond(Request request) throws IOException {
        String urlPrefix = request.props.getProperty(propsPrefix, name );
        if (!request.url.startsWith(urlPrefix)) {
            log("Not" + name + "prefix: " + request.url);
            return false;}

        Hashtable table = request.getQueryData();
        // traitement de la requête
        String result = .....
        request.sendResponse(result);
        return true;
    }
}
```

Exemple : ClockHandler sur la carte TINI

```
public class ClockHandler implements Handler {
    private String propsPrefix; private Server server;
    private String name = "/clock/";

    public boolean init(Server server, String prefix) {
        this.server = server;
        this.propsPrefix = prefix;
        try {
            // lecture du parametre prefix.name
            String n = server.props.getProperty(propsPrefix + "name");
            if (n != null) name = n;
        } catch (Exception e) {}
        return true;
    }
}
```

Exemple : ClockHandler sur la carte TINI

```
public boolean respond(Request request) throws IOException {

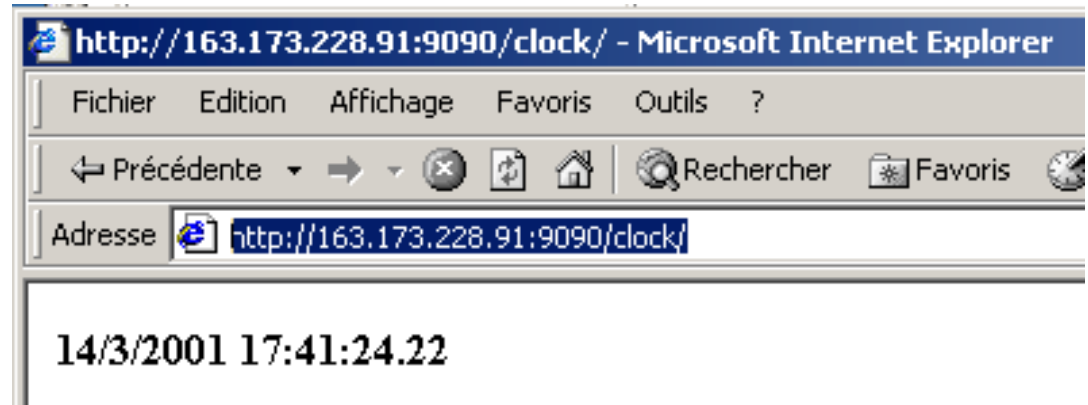
    String urlPrefix = request.props.getProperty(propsPrefix, name);

    if (!request.url.startsWith(urlPrefix)) {
        server.log(Server.LOG_INFORMATIONAL, null, "Not " + name + " prefix: " + request.url);
        return false;
    }

    // la réponse à la requête HTTP
    Clock C = new Clock(); C.getRTC();
    String date = C.getDate() + "/" + C.getMonth() + "/" +
        (C.getYear()+2000) + " " + C.getHour() + ":" +
        C.getMinute() + ":" + C.getSecond() + "." + C.getHundredth();

    String result = "<b>" + date + "</b>";
    request.sendResponse(result);
    return true;
}}
```

Exemple sur TINI suite :



- handler=main
- port=9090
- log=5
- root=.
- #
- main.class=sunlabs.brazil.server.ChainHandler
- main.handlers=**clock file fileNotFound**
- #
- **clock**.class=cnam.tpcdi.ClockHandler
- **clock**.name=/clock/
- #
- **file**.class=sunlabs.brazil.server.FileHandler
- **file**.default=index.html
- #
- **fileNotFound**.class=sunlabs.brazil.handler.NotFoundHandler
- **fileNotFound**.fileName=notFound.html

Structure du ChainHandler

```
public boolean    init(Server server, String prefix)    {
    this.prefix = prefix;
    Properties props = server.props;
    // affectation du vecteur handlers à partir de props
    // extrait du fichier de configuration
```

....

```
public boolean    respond(Request request)    throws IOException    {

    for (int i = 0; i < handlers.length; i++) {
        if (handlers[i].respond(request)) {
            ...
            return true;
        }
    }
    return false;
}
```

Quelques Handlers :

- **SqlHandler** *passerelle* *HTTP <-> SQL*
- **SMTPHandler** *passerelle* *HTTP <-> SMTP*
- **DS2438Handler** *HTTP <-> 1Wire*
- **SNMPHandler** *HTTP <-> SNMP*
- **MODBUSHandler** *HTTP <-> MODBUS/TCP*
- **IntrospectionHandler** *HTTP <-> JVM(java.reflect.*)*
- *******Handler** *HTTP <-> ******
- **utilitaires :**
 - **MultiProxyHandler**
 - **HashtableHandler**

peut remplacer ChainHandler, lorsque les Handler sont indépendants

HTTP <-> SQL

SqlHandler

- Handler d'interrogation de tables d'une base de données
- Ici hypersonicSQL, www.hsqldb.org

SqlHandler : la configuration

#La configuration :

```
sql.class=cnam.sql.SqlHandler
```

```
# sélection du handler, debut de la requete
```

```
sql.name=/hypersonic/
```

```
# paramètres de la base de données (voir JDBC)
```

```
# driver jdbc
```

```
sql.driver=org.hsqldb.jdbcDriver
```

```
# URL de la base de données
```

```
sql.url=jdbc:hsqldb:http://163.173.228.80:8181
```

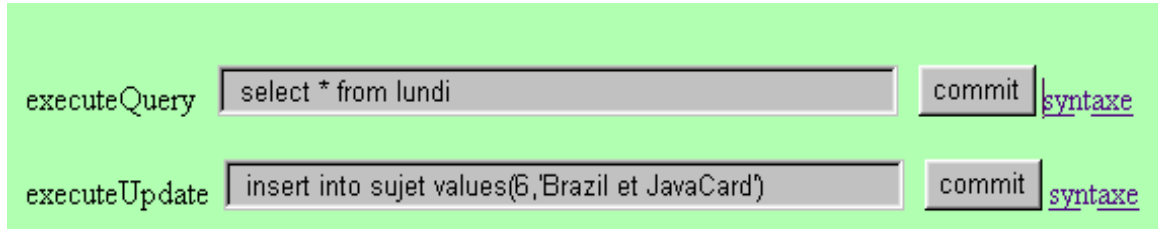
```
# paramètres habituels
```

```
sql.user=sa
```

```
sql.password=
```

SqlHandler : déclenchement

- `http://163.173.228.80:9090/hypersonic/?query=select * from lundi`



executeQuery [syntaxe](#)

executeUpdate [syntaxe](#)

```
<FORM ACTION="http://163.173.228.80:9090/hypersonic/">
  executeQuery
  <INPUT TYPE="text" NAME= "select" VALUE= "select * from lundi "
    SIZE="40">
  <INPUT TYPE=submit VALUE="commit">
  <A HREF="#select">syntaxe</A>
</FORM>
```

SqlHandler : la réponse

```
select * from lundi
```

résultat

NOM	PRENOM	E_MAIL	SUJET
Lambert	michel	mlambert@noos.fr	1
Robleda	bruno	brunodb@club-internet.fr	1
Nguyen	thuan	the_thuan@yahoo.fr	1
Minchella	michel	p_minchella@hotmail.com	1
Derrien	alain	alain_derrien@yahoo.fr	2
Ciret	evariste	eciret@yahoo.fr	2
Lebougre	herve	herve.lebougre@libertysurf.fr	2
Rodriguese	jose	jhrodrigues@wanadoo.fr	2

SqlHandler.java : le squelette(tête)

```
private String name,driver,url,user,password;
```

```
public boolean    init(Server server, String prefix){
```

```
    // Lecture du fichier de configuration
```

```
    } catch (Exception e1) {
```

```
        return false;
```

```
    }
```

```
    // chargement du driver jdbc
```

```
    try {
```

```
        Class.forName(driver);
```

```
    } catch(Exception e) {
```

```
        return false;
```

```
    }
```

SqlHandler.java : le squelette(corps)

```
public boolean respond(Request request) throws IOException{

    if (!request.url.startsWith(urlPrefix)) {return false;}

    Hashtable t = request.getQueryData();
    try{
        con = DriverManager.getConnection(url,user,password);
        stmt = con.createStatement();
        query = (String)t.get("query");
        if (query != null){
            ResultSet rs = stmt.executeQuery(query);
            result = query + formatTable(rs);
        } else {
            String update = (String)t.get("update");
            if( update != null){
                int res = stmt.executeUpdate(update);
                result = update + res;
            } else result = "unknown, only (query or update)";
        } ...
    }
```

SqlHandler

- A essayer avec précautions en
 - http://lmi80.cnam.fr:9090/hypersonic/p?query=select+*+from+lundi
 - Utilisateur: **cnam** mot de passe: **brazil**

SMTPHandler : la configuration

- HTTP <-> SMTP
- La configuration :

```
smtp.class=cnam.ead.SMTPHandler
```

```
# sélection du handler
```

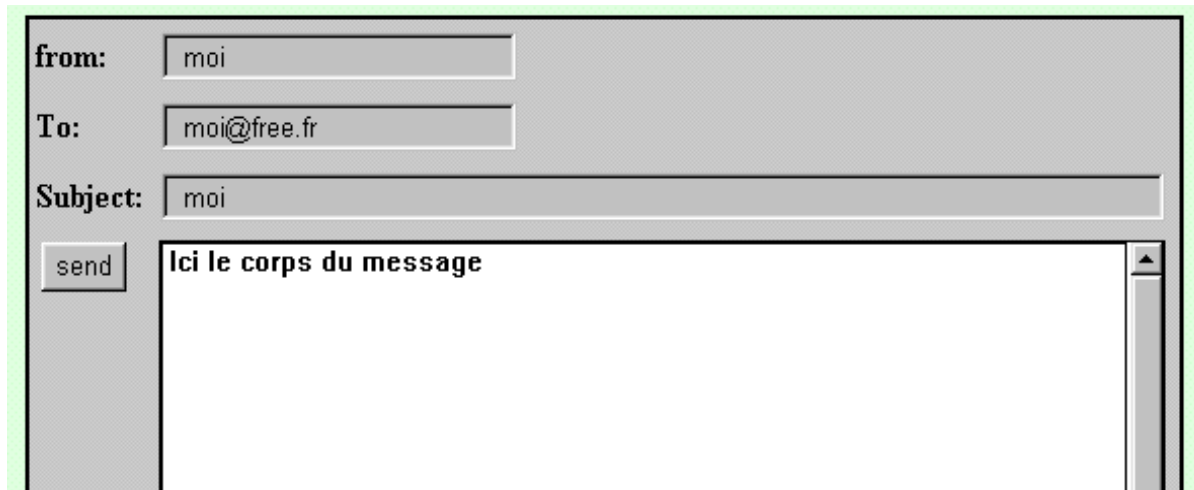
```
smtp.prefix=/smtp/
```

```
# le serveur de messages sortants smtp
```

```
smtp.host=smtp.cnam.fr
```

SMTPHandler : déclenchement

- `http://lmi80.cnam.fr:9090/smtp/?from=moi&to=moi@free.fr&subject=moi&message=Ici+le+corps+du+message`

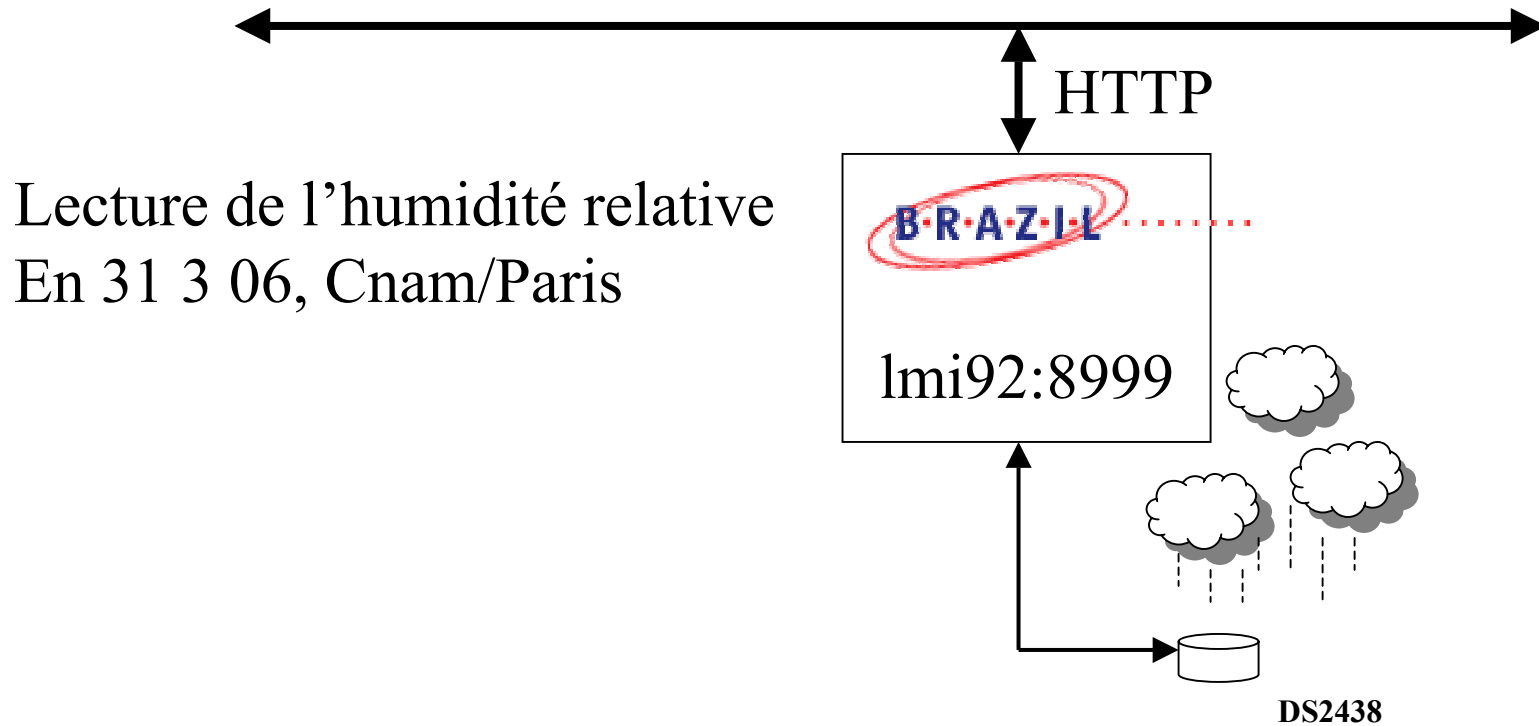


The image shows a web form for sending an email via SMTP. It has a light gray background and a thin black border. The form is divided into several sections:

- from:** A text input field containing the value "moi".
- To:** A text input field containing the value "moi@free.fr".
- Subject:** A text input field containing the value "moi".
- Message Body:** A large text area containing the text "Ici le corps du message". To the left of this area is a "send" button.

The entire form is set against a light green background.

Lecture d'un ibutton : DS2438



à essayer en <http://lmi92.cnam.fr:8999/ds2438/?infoAll=on>

à essayer en <http://lmi92.cnam.fr:8999/ds2438/?infoAll=on&listAll=on>

à lire en

<http://lmi92.cnam.fr:8080/cnam/ibutton/index.html>

<http://lmi92.cnam.fr:8080/cnam/ibutton/DS2438Handler.java>

<http://lmi92.cnam.fr:8080/cnam/ibutton/AppletteDS2438.java>

Un extrait du fichier de configuration

```
#  
handler=main  
port=8999  
log=5  
root=.  
#  
main.class=sunlabs.brazil.server.ChainHandler  
main.handlers= ds2438 file dir  
  
ds2438.class=cnam.ibutton.DS2438Handler  
ds2438.name=/ds2438/
```

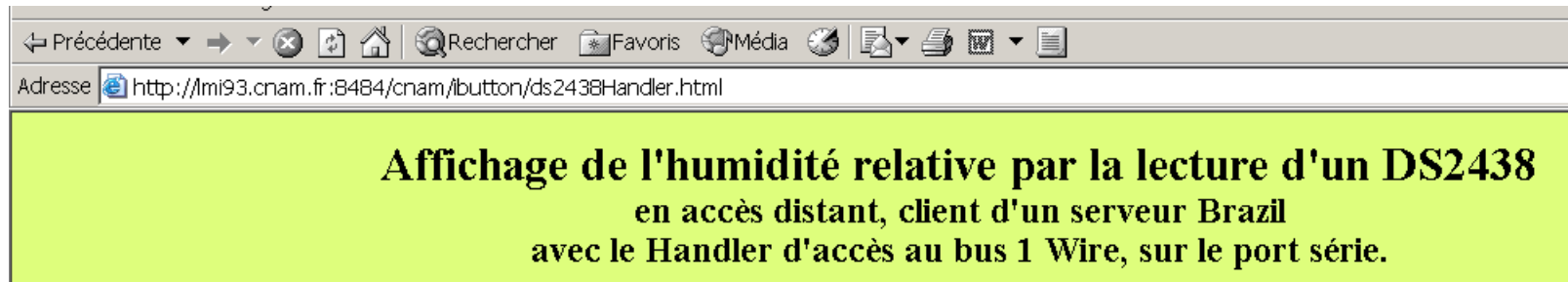
Applet + Requête HTTP cyclique

```
public class AppletteDS2438 extends Applet
    implements ActionListener, Runnable {

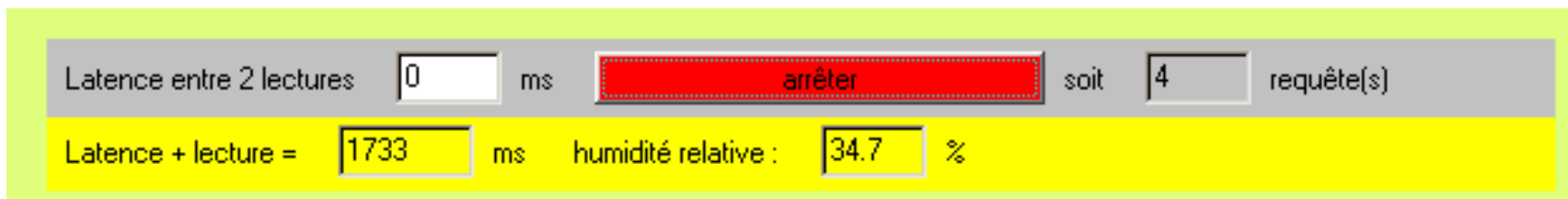
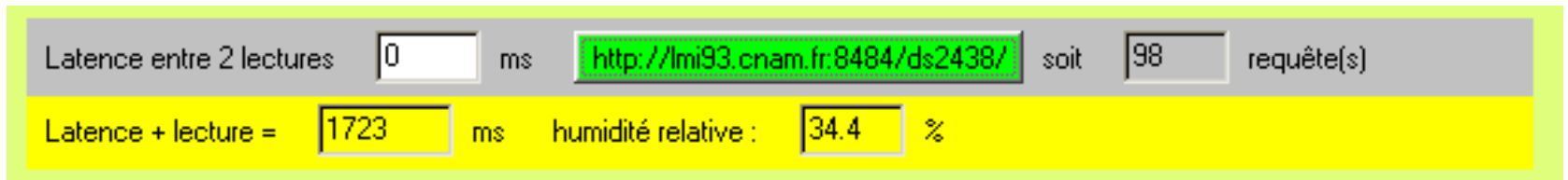
    public void init(){ t = new Thread(this);t.start();}
    public void stop(){t.interrupt();}

    public void run() {
        while(!t.interrupted()){
            long top = System.currentTimeMillis();
            try{
                Thread.sleep(dureeDeLatence);
                DataInputStream is =
                    new DataInputStream(new URL(urlRequest).openStream());
                String str = is.readLine();
                // traiter le retour
                .....
            }
        }
    }
}
```

Trace d'exécution



- Déclenchement cyclique du Handler depuis une applette



ChainHandler vers une critique ?

```
main.class=cnam.brazil.ChainHandler
```

```
main.handlers= clock alarm ds2438 thermo props sql file
```



- Les handlers de la chaîne peuvent se transmettre des paramètres
 - Et/ou modifier des variables du serveur
- Et si les handlers étaient indépendants ?
 - Pourrions éviter leur déclenchement en séquence

HashtableHandler

- Les handlers sont indépendants
- Accès direct au handler approprié
- Sémantique analogue switch/case/default

```
main.class=cnam.brazil.HashtableHandler
```

```
main.handlers= clock alarm ds2438 thermo props sql file
```

```
switch( prefix(url.request) ){  
  case /clock/ : return hTable[/clock/].respond(request);break;  
  case /alarm/ : return hTable[/alarm/].respond(request);break;  
  
  default : return handlers[file].respond(request);  
}
```

HashtableHandler : la configuration

- La configuration est analogue à ChainHandler en ajoutant les valeurs de **hash** de chaque Handler

```
main.class=cnam.tpcdi.HashtableHandler
main.handlers=clock alarm ds2438 thermo props sql file
# valeur de hash pour chaque Handler, sauf file
clock.hash=/horloge/
alarm.hash=/alarme/
ds2438.hash=/ds2438/
thermo.hash=/ds1921/
props.hash=/props/
sql.hash=/sql/

# et ensuite les paramètres habituels des Handlers
clock.class=cnam.brazil.ClockHandler
```

HashtableHandler.java

```
public class HashtableHandler extends ChainHandler{

    private int last;
    private Hashtable hTable;

    public boolean init(Server server, String prefix){
        String pre;
        if ( super.init(server,prefix)){
            hTable = new Hashtable(names.length + names.length/3);
            last = names.length-1;
            for(int i=0;i< last;i++){
                pre = server.props.getProperty(names[i] + ".hash");
                // controles de xxx.prefix : la clé doit être unique
                hTable.put(pre,handlers[i]);
            } ...
        }
    }
}
```

HashtableHandler.java

```
public boolean respond(Request request) throws IOException{
    boolean res;
    String req = request.url;
    String pre = req.substring(0,req.indexOf('/',1)+1);

    Handler h = (Handler)hTable.get(pre);

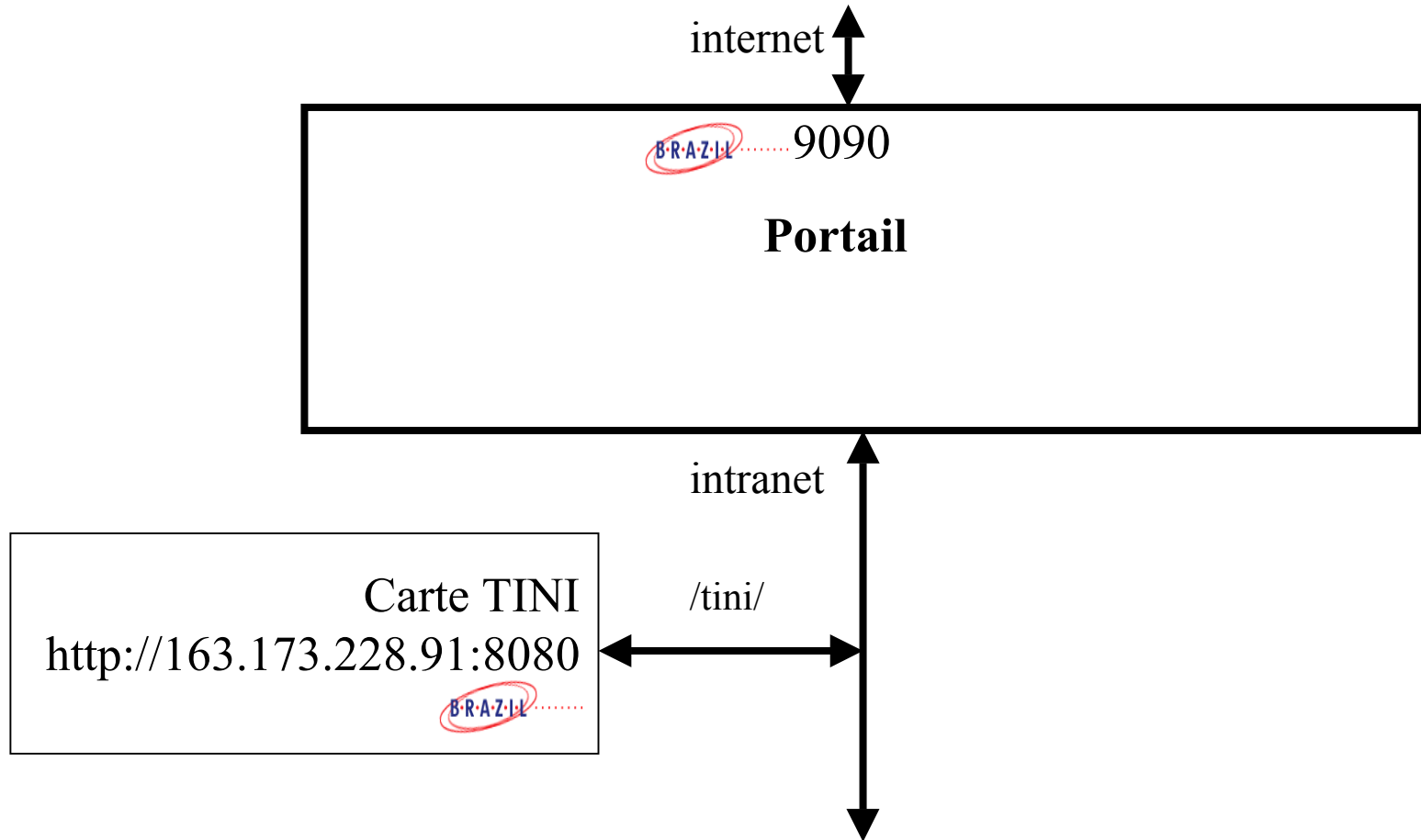
    if(h != null)
        res = h.respond(request);
    else
        res = handlers[last].respond(request);

    return res;
}
}
```

HashtableHandler

- Taille du code : 613 octets sur la TINI
- Le comparatif avec ChainHandler ainsi que les mesures en performance restent à faire
- En fonction du nombre de clients, des requêtes effectuées, des handlers sélectionnés
- Effet de bord possible entre deux requêtes

MultiProxyHandler



→ Indépendance d'une architecture

MultiProxyHandler

```
handler=main
port=9090
log=5
root=.
#
main.class=sunlabs.brazil.server.ChainHandler
main.handlers=clock tini file
#
tini.class=sunlabs.brazil.handler.MultiProxyHandler
tini.host=163.173.228.91
tini.prefix=/tini/
tini.port=8080
```

<http://localhost:9090/tini/> est réécrit en **interne** <http://163.173.228.91:8080/>

MultiProxyHandler & JFOD

ED CDL B3	Répondre aux ED CDL	Forum CDL B3
DH CDL B3	Répondre aux DH CDL	Forum CDL B3
Projet CDI	Répondre aux TP CDI	Forum TP CDI
IAGL C3	Répondre aux TP IAGL	Forum IAGL C3
ISTR C2	Répondre aux TP ISTR	Forum ISTR

Réponse et inscription : [mode d'emploi](#)

http://lmi92.cnam.fr:8080/tp_cdi/

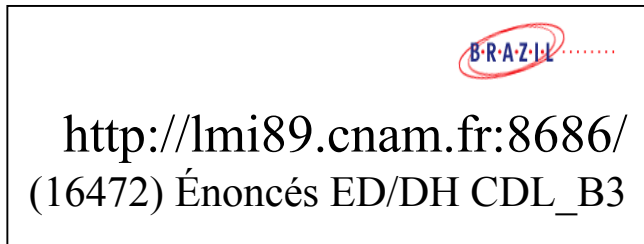
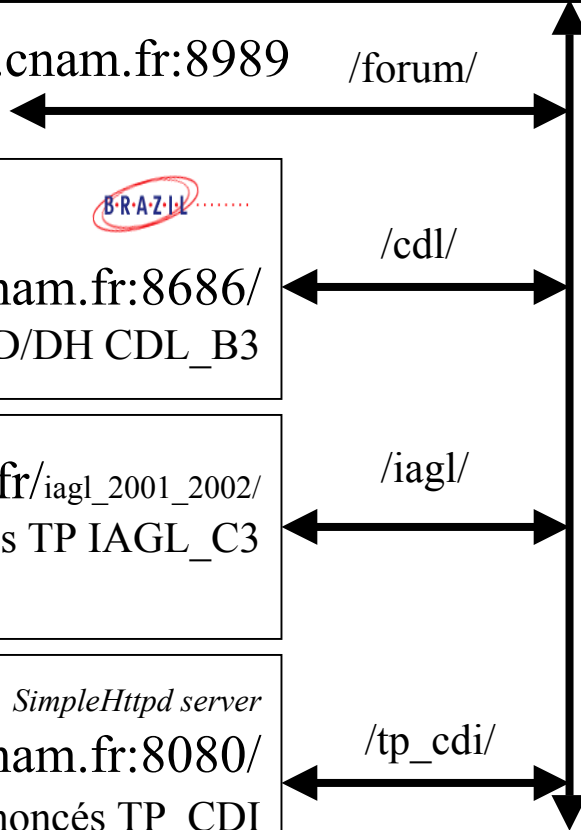
(en interne <http://lmi80.cnam.fr:8080>)

lmi92.cnam.fr:8080 est un portail / 8 machines en tout (pour le moment)

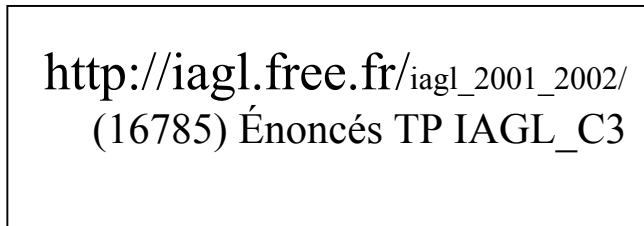
JFOD <http://lmi92.cnam.fr:8080/>



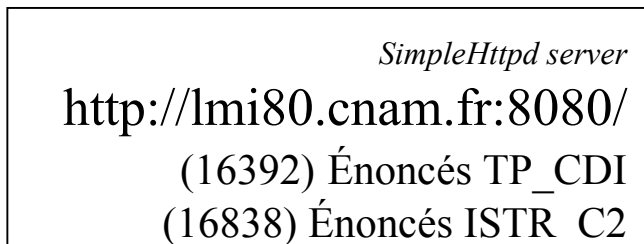
<http://lmi92.cnam.fr:8989> /forum/



/cdl/

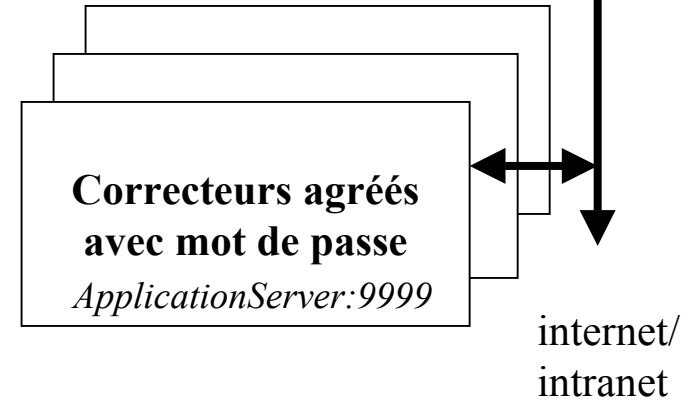


/iagl/



/tp_cdi/

internet/intranet



seuls administrateurs :

@IP : 163.173.228.{90|91|92|93}

MultiProxyHandler & JFOD

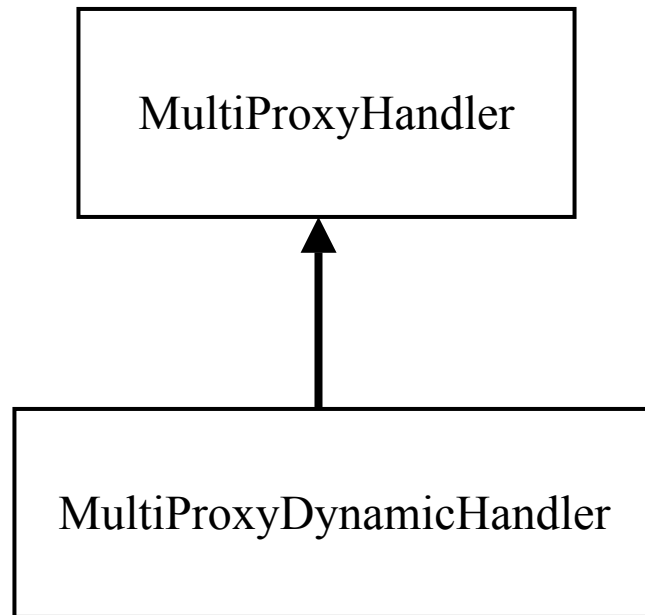
```
handler=main
port=8080
log=3
root=public
#
main.class=sunlabs.brazil.server.ChainHandler
main.handlers= tpcdi cdl iagl forumd file
#
tpcdi.class=sunlabs.brazil.handler.MultiProxyHandler
tpcdi.host=lmi80.cnam.fr
tpcdi.prefix=/tp_cdi/
tpcdi.port=8080
##
forumd.class=sunlabs.brazil.handler.MultiProxyHandler
forumd.host=lmi92.cnam.fr
forumd.prefix=/forum/
forumd.port=8989
##
...
```

À essayer :

- <http://lmi80.cnam.fr:8080>
- <http://lmi92.cnam.fr:8989>

MultiProxyDynamicHandler

- Changement dynamique de proxy ...
- Une solution : hériter de l'existant



Requêtes HTTP

- http://lmi90.cnam.fr:8080/tp_cdi/
 - Redirection en lmi80.cnam.fr:8080
- http://lmi90.cnam.fr:8080/tp_cdi/modify/?host=lmi88.cnam.fr&port=8888
- http://lmi90.cnam.fr:8080/tp_cdi/
 - Redirection en lmi88.cnam.fr:8888

Le code

```
public class MultiProxyDynamicHandler extends MultiProxyHandler{

    public void init(Server s, String prefix){
        // lecture du fichier de configuration
        return super.init(s,prefix);
    }

    public boolean respond(Request request) throws IOException{
        // si l'url commence par /modify/
        // lecture des paramètres HTTP (host et port)
        // recherche avec substitution éventuelle dans la table
        String map = "http://" + host + ":" + port;
        MultiProxyHandler.proxies.put(map,urlPrefix);
        //

        return super.respond(request);
    }
}
```

Quels sont ces Handlers ?

- La plupart reste à découvrir . . .

Le paquetage sunlabs.brazil.handler

```
[AclSwitchHandler , BasicAuthHandler , CgiHandler ,  
ChainHandler ,ChainSawHandler ,ChownHandler ,  
ConfigFileHandler ,CookieSessionHandler ,  
DefaultFileHandler ,DeferredHandler ,DialogHandler ,  
DirectoryHandler , DynamicConfigHandler ,FileHandler ,  
GenericProxyHandler ,HomeDirHandler ,JunkBusterHandler  
LogHandler , MultiHostHandler ,MultiProxyHandler ,  
NotFoundHandler ,PropertiesHandler ,ProxyHandler ,  
ProxyPropertiesHandler ,PublishHandler , PushHandler ,  
ReflectHandler ,ResourceHandler ,RestrictClientHandler ,  
RolesHandler , SMTPHandler , SunNetAuthHandler ,SupplyHandler ,  
TclHandler , TemplateHandler ,UrlMapperHandler ,VirtualHostHandler]  
  
[GenericX10Hanlder , TwoWayHandler]
```

<http://www.brazilhandlers.com:9090/>

```
main.class=sunlabs.brazil.server.ChainHandler
```

```
main.handlers=admin session weather x10 propstext default filter
```

```
admin.class=sunlabs.brazil.admin.AdministrationHandler
```

```
admin.prefix=/admin/
```

```
admin.restrict=127.0.0.1
```

```
session.class=sunlabs.brazil.handler.CookieSessionHandler
```

```
weather.calibrationRate=500
```

```
weather.class=sunlabs.brazil.weather.WeatherHandler
```

```
weather.device=/dev/cua/b
```

```
propstext.class= sunlabs.brazil.handler.PropertiesHandler
```

```
propstext.match=/weather/*
```

```
filter.class=sunlabs.brazil.filter.FilterHandler
```

```
filter.filters=htmlify extract template integrate process
```

```
filter.handler=content
```

Mise en œuvre de Handler

- Faire le TP11 !

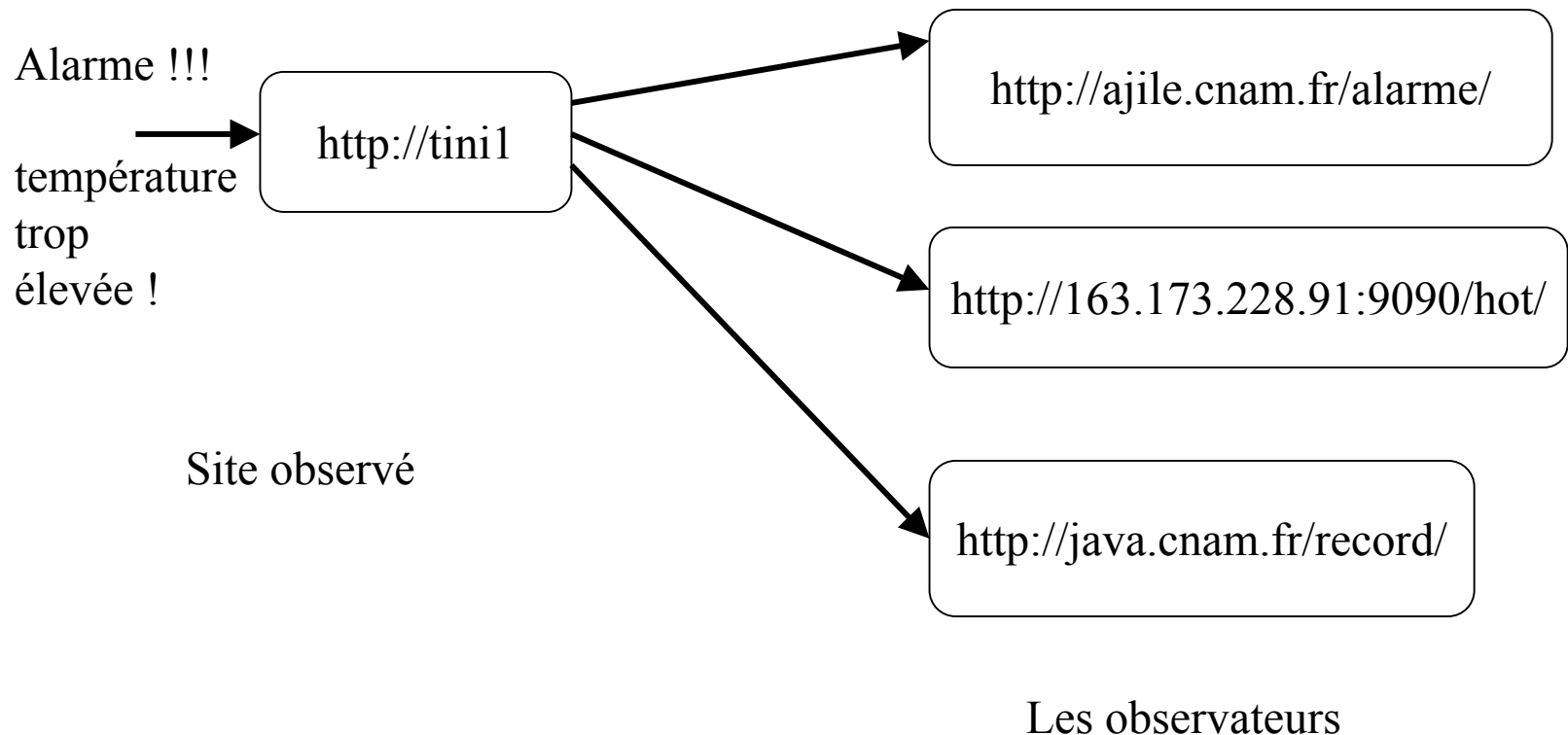
Petits Pattern entre nous

- Observateur ?
- Chain of Responsibility ?
- Composite et Visiteur ?

- Et bien d'autres encore

Le Pattern Observateur et Brazil

- Gestion des évènements asynchrones ?
- Inscription d'observateurs/serveurs Brazil ou autres



Le Pattern Observateur en Java

- Implémentation : usage de
 - La classe `java.util.Observable`
 - L'interface `java.util.Observer`
- Soit l'implémentation des méthodes
 - `addObserver`
 - `notifyObservers`
 - `deleteObservers`
- HTTP <--> Appel de méthodes Java
- Le site gestionnaire peut être externe à l'observé

Implémentation de méthodes

- Appel de méthodes en URL

```
addObserver("http://java.cnam.fr");
```

en HTTP

```
http://un_serveur_Brazil/alarm/
```

```
addObserver/param?url=http://java.cnam.fr
```

```
notifyObservers()
```

en HTTP

```
http://un_serveur_Brazil/alarm/notifyObserver
```

Implémentation du Handler alarm

```
public class AlarmHandler extends Observable implements
    Handler{

    public void init( ...)

    public void respond(Request request){
        // Contrôle du préfixe alarm
        if( request.url.startwith("addObserver") ){
            Hashtable t = request.getQueryData();
            addObserver( new ServerObserver((String)t.get("url")) )
        }
    }

    else if ...
}
```

La classe des Observés

```
private static class ServerObserver implements Observer{
    private String urlString;

    public void update(Observable o, Object arg){
        try{
            URL url = new URL(this.urlString);
            url.openStream();
        }catch(IOException ioe){
            System.out.println("IOException : " + ioe);
        }
    }

    public ServerObserver(String urlString){
        this.urlString = urlString;
    }
}
```

Un exemple d'utilisation

- **Inscription des Observateurs**

le serveur **java.cnam.fr** est un observateur, il sera prévenu lors d'une notification à la suite d'une alarme

<http://lmi80.cnam.fr:9090/alarm/addObserver/param?url=http://java.cnam.fr/>

- le serveur **Brazil** installé sur la carte **TINI** mémorise par le Handler "record" la valeur reçue lors de la notification

<http://lmi80.cnam.fr:9090/alarm/addObserver/param?url=http://163.173.228.92:8222/record/>

le site **Brazil** (en californie ?) est un observateur de nos TP(enfin presque)

<http://lmi80.cnam.fr:9090/alarm/addObserver/param?url=http://www.experimentalstuff.com/>

une carte **TINI** de l'esiee (à Noisy le grand) est également un observateur...

<http://lmi80.cnam.fr:9090/alarm/addObserver/param?url=http://145.147.215.50.197:8080/>

2) Deux réveils de tous les observateurs

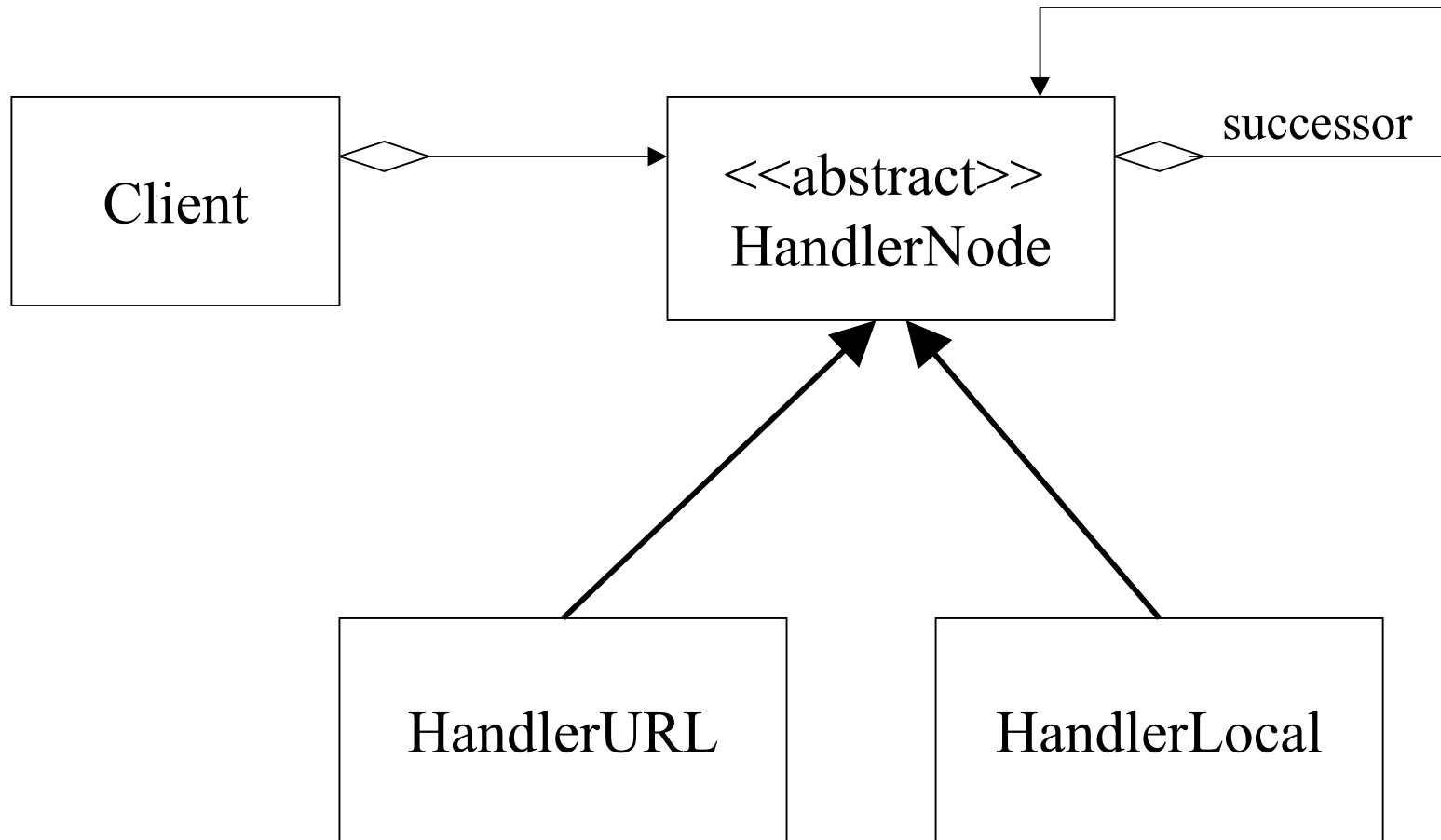
2.1) Alarme !! : la valeur est de 30

<http://lmi80.cnam.fr:9090/alarm/notifyObservers/param?valeur=30>

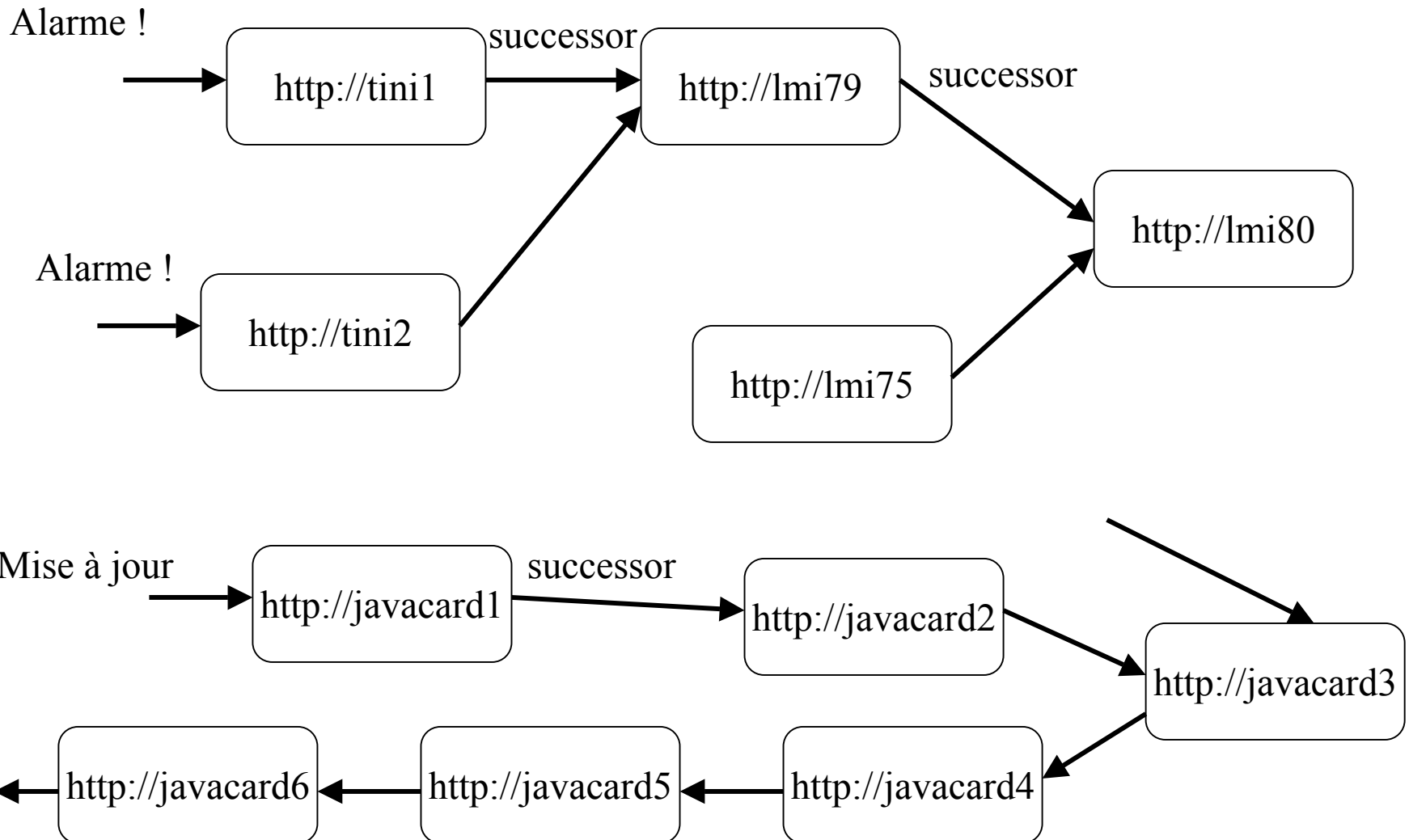
2.2) Alarme !! : la valeur est de 35

<http://lmi80.cnam.fr:9090/alarm/notifyObservers/param?valeur=35>

Le pattern : Chain of Responsibility



Chain of responsibility : instances



HandlerNode.java

```
public abstract class HandlerNode{
    protected HandlerNode successor;

    public HandlerNode(){ this.successor = null;}

    public HandlerNode(HandlerNode successor){ ... }

    public void setSuccessor(HandlerNode successor){
        this.successor = successor;
    }

    public boolean handle(Object arg){
        if(successor != null) return successor.handle(arg);
        return false;
    }
}
```

HandlerURL.java

```
public class HandlerURL extends HandlerNode{
    private String urlString;

    public HandlerURL(String urlString){
        this.urlString = urlString;
    }

    public boolean handle(Object arg){
        URL url = new URL(urlString + arg);
        // connexion puis lecture de la réponse,
        // si elle n'est pas satisfaisante alors
        return successor.handle(arg); //super.handle(arg);

        return true;
    }
}
```

Brazil Handler <http://serveur/prefix/>

addNode (site, site2);

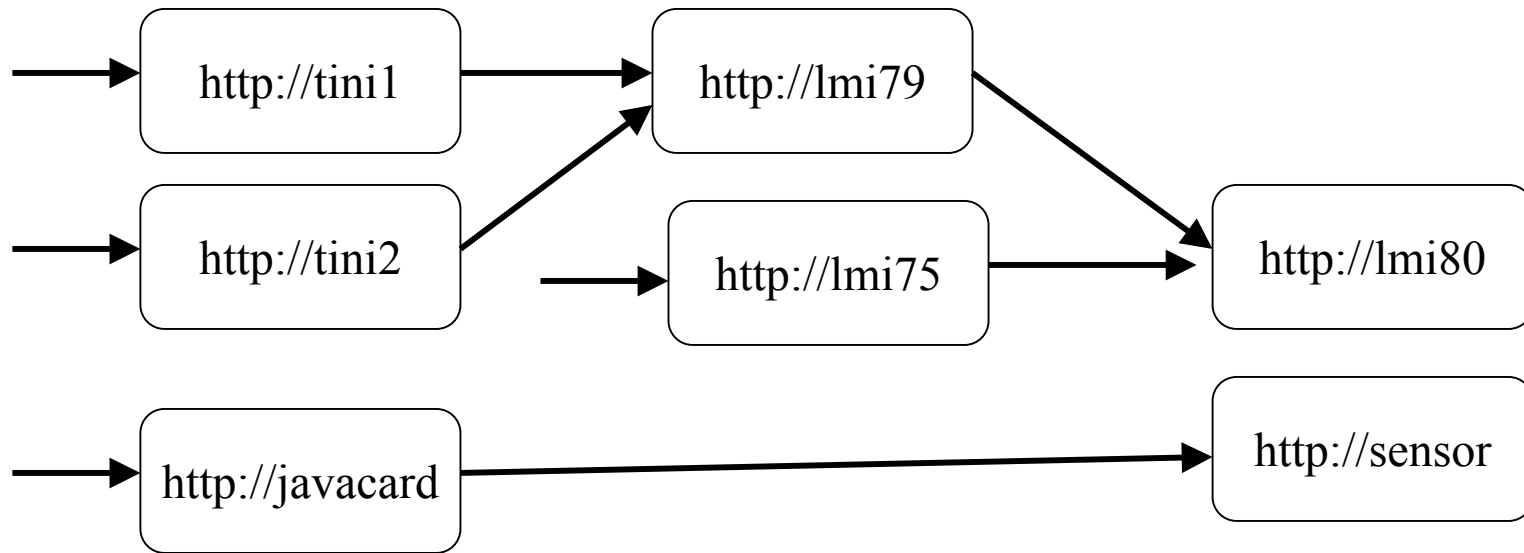
site2 = site.successor

- `addNode/param?url=http://site&succ=http://site2`

notifyNode(site,parametre);

- `notifyNode/param?url=http://site¶metre=100`

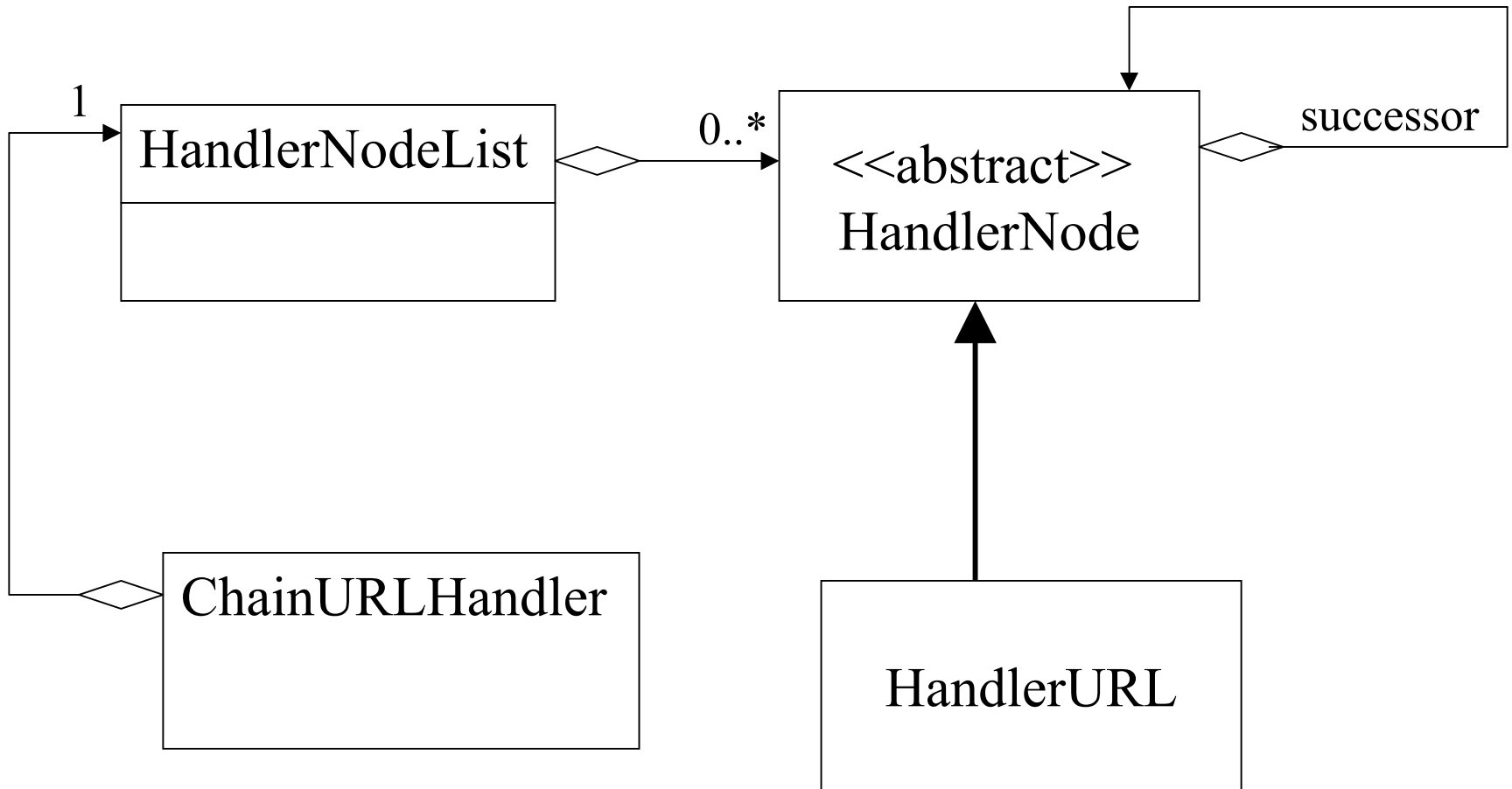
Chain of responsibility : instances



- `http://...addNode/param?url=http://lmi79&succ=http://lmi80`
- `http://...addNode/param?url=http://lmi75&succ=http://lmi80`
- `http://...addNode/param?url=http://tini1&succ=http://lmi79`
- `http://...addNode/param?url=http://tini2&succ=http://lmi79`
- `http://...addNode/param?url=http://javacard&succ=http://sensor`

- `http://...notifyNode/param?url=http://tini2&valeur=100`

ChainURLHandler, Brazil



La classe HandlerNodeList

- La classe liste des Nœuds utilise par délégation une Hashtable :
 - Clé (URL String, le site) ->
 - Valeur (HandlerNode, le nœud associé)
- 1 ou plusieurs URL ont un seul successeur
- Attention aux boucles ...

HandlerNodeList .java

```
public class HandlerNodeList{  
    private Hashtable table = new Hashtable();  
  
    public boolean isEmpty(){  
        return table.isEmpty();  
    }  
  
    public String toString(){  
        return table.toString();  
    }  
  
    public HandlerNode get(String url){  
        return (HandlerNode)table.get(url);  
    }  
}
```

prefix/addNode

```
public void add(String url, String succ){ // addNode en HTTP
    HandlerNode succHandler, urlHandler;
    url = url.intern(); succ = succ.intern();
    succHandler = (HandlerNode)table.get(succ);
    if(succHandler == null){
        succHandler = new HandlerURL(succ);
        table.put(succ, succHandler);
    }
    urlHandler = (HandlerNode)table.get(url);
    if(urlHandler == null){
        urlHandler = new HandlerURL(url);
        table.put(url, urlHandler);
    }
urlHandler.setSuccessor(succHandler);
}
```

prefix/notifyNode/

```
http://...notifyNode/param?url=http://tini1&valeur=100
```

```
http://...notifyNode/param?url=http://tini2&valeur=100
```

```
public boolean respond(Request request){
```

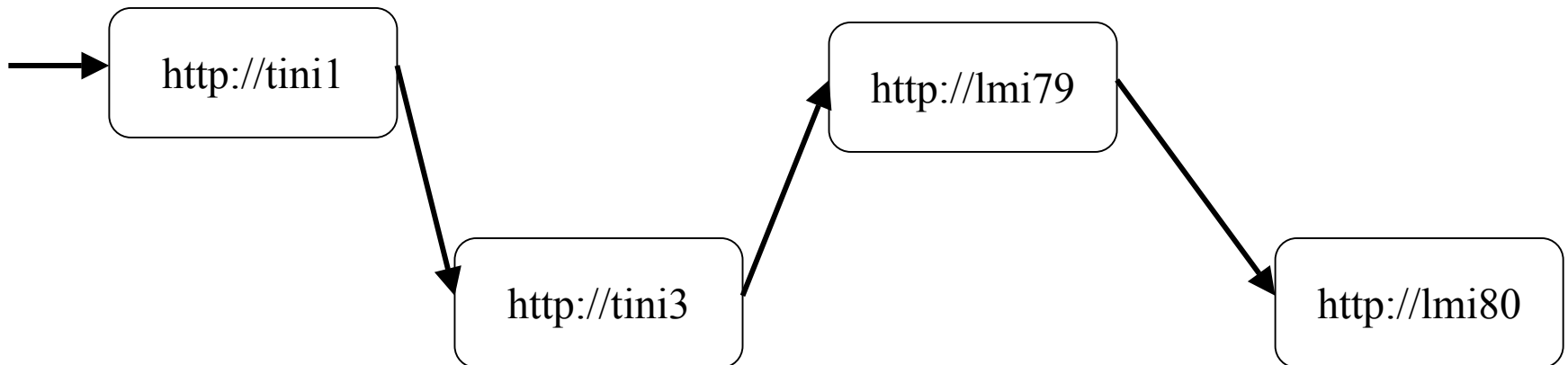
```
    handler = list.get("http://tini1");
```

```
    if (handler.handle( request.query)) { .....
```



Ajout d'un nœud

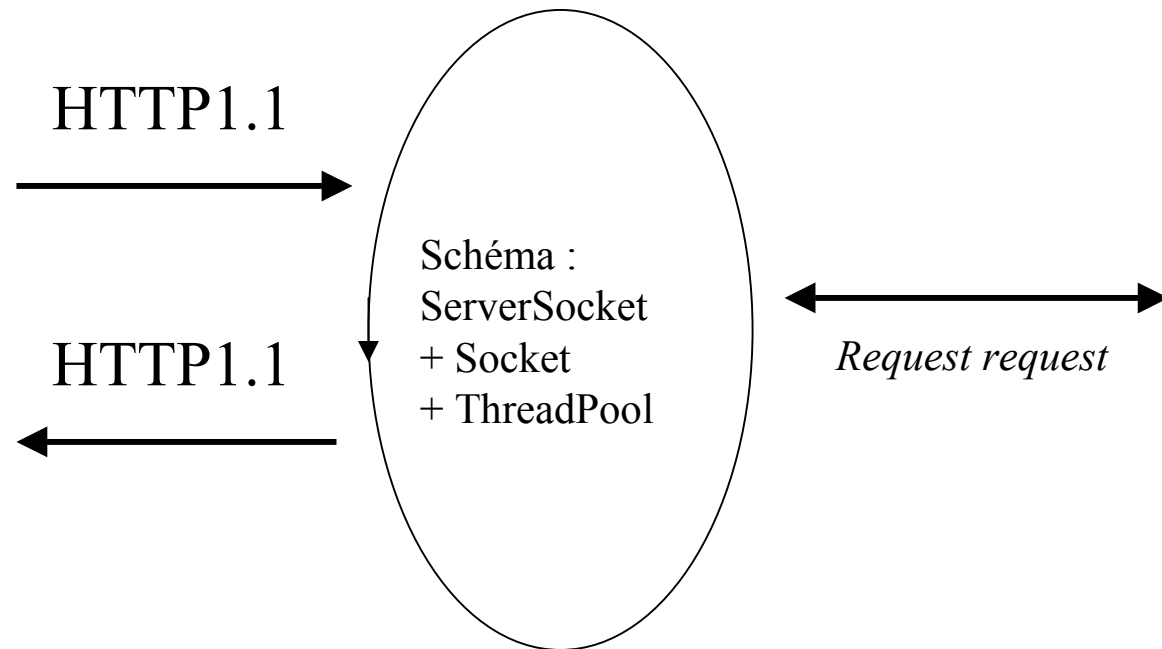
`http://...addNode/param?url=http://tini3&succ=http://lmi79`
`http://...addNode/param?url=http://tini1&succ=http://tini3`



Idées de développement

- D'autres patterns :
 - Le Pattern Proxy
 - Le Pattern Composite et ses Visiteurs
 - ...
- D'autres Handler
 - Introspection
 - URLClassLoader
 - URLObjectLoader
 - DynamicHandler
 - ...

Idées de développement avec le ThreadPool voir le Tp10



Serveur de requêtes
Server.java
Request.java

Encore un Handler !

- **public class FilterHandler implements Handler**

The **FilterHandler** captures the output of some **Handler** and then applies an number of **Filters** to change that output before it is returned to the client.

This handler provides one of the core services now associated with the Brazil Server: the ability to dynamically rewrite web content obtained from an arbitrary source.

For instance, the **FilterHandler** can be used as a proxy for a PDA. The wrapped **Handler** would go to the web to obtain the requested pages on behalf of the PDA. Then, a **Filter** would examine all "text/html" pages and rewrite the pages so they fit into the PDA's 200 pixel wide screen. Another **Filter** would examine all requested images and dynamically dither them to reduce the wireless bandwidth consumed by the PDA.

Les 3 composantes de Brazil

- *L'interface sunlabs.brazil.server.Handler*
- *L'interface sunlabs.brazil.filter.Filter*
- *La classe sunlabs.brazil.template.Template*

L'interface Filter

```
package sunlabs.brazil.filter;  
import sunlabs.brazil.server.Handler;  
import sunlabs.brazil.server.Request;  
import sunlabs.brazil.util.http.MimeHeaders;  
  
public interface Filter extends Handler{  
    public boolean shouldFilter(Request request, MimeHeaders headers);  
  
    public byte[] filter(Request request, MimeHeaders headers, byte[] content);  
  
}
```

Un exemple de filtre : PlainFilter

```
public class PlainFilter implements Filter {

    String prefix = "<title>text document</title><body bgcolor=white><pre>";
    String suffixe = "</pre></body>";

    public boolean init(Server server, String prefix) {return true;}
    public boolean respond(Request request) { return false; }

    public boolean shouldFilter(Request r, MimeHeaders headers) {
        String type = headers.get("content-type");
        return (type != null && type.startsWith("text/plain"));
    }
}
```

Un exemple de filtre suite

```
public byte[] filter(Request request, MimeHeaders headers,  
    byte[] content) {  
    String result = prefixe +  
                    htmlIfy(new String(content)) +  
                    suffixe;  
    headers.put("content-type", "text/html");  
    return result.getBytes();}
```

Le fichier de configuration

```
#
handler=main
port=9090
log=5
root=.
#
main.class=sunlabs.brazil.server.ChainHandler
main.handlers=clock tini reflect filter dir dirOrFileNotFound

#
filter.class=sunlabs.brazil.filter.FilterHandler
filter.handler=file
filter.filters=plain
#
plain.class=sunlabs.brazil.filter.PlainFilter
plain.template= <b> c'est un document de type mime text/plain</b><br>\
                  filtre : PlainFilter<br> \
                  <pre><b><i>%</i></b></pre>
#
#          le texte est associé au %
```

Le résultat

c'est un document de type mime text/plain

filtre : PlainFilter

```
#  
handler=main  
port=9090  
log=5  
root=.  
#  
main.class=sunlabs.brazil.server.ChainHandler  
main.handlers=clock alarm tini reflect props filter dir dirOrFileNotFound  
#  
filter.class=sunlabs.brazil.filter.FilterHandler  
filter.handler=file  
filter.filters=plain  
#  
plain.class=sunlabs.brazil.filter.PlainFilter  
plain.template=<b> c'est un document de type mime text/plain</b><br> |  
    filtre : PlainFilter<br> |  
<pre><b><i>%</i></b></pre>  
#  
mime.java=text/plain  
mime.txt=text/plain
```

Le paquetage sunlabs.brazil.filter

[CookieFilter , CookieSessionHandler , CopyContentFilter,
HistoryFilter , ReplaceFilter , SessionFilter ,TclFilter ,
TemplateFilter , UrlSessionFilter]

Quelques filtres !

- **public class CopyContentFilter implements Filter**

Filter to save content (of an entire site) to a disk file. This is used to "steal" other sites. It is expected to be used in conjunction with a **GenericProxyHandler**. Only files that don't already exist on the local file system are saved.

- **public class TemplateFilter implements Filter**

The **TemplateFilter** sends HTML content through an Html/XML parser to a set of **Templates**. Each Html/XML tag may dynamically invoke a Java method present in the **Templates**. The dynamically-generated content from evaluating the Html/XML tags is returned to the caller.

Les 3 composantes de Brazil

- *L'interface sunlabs.brazil.server.Handler*
- *L'interface sunlabs.brazil.filter.Filter*
- ***La classe sunlabs.brazil.template.Template***

La classe Template

```
package sunlabs.brazil.template;

public class Template{

/**
 * Called before this template processes any tags.
 */

public boolean    init(RewriteContext hr)    {

return true;

}

/**
 * Called after all tags have been processed, one final chance.
 */

public boolean    done(RewriteContext hr)    {

    return true;

}

}
```

La classe CountTemplate

```
package sunlabs.brazil.template;

import sunlabs.brazil.server.Server;

public class CountTemplate extends Template{

    private int images = 0;        // the various counts
    private int links = 0;

    public void tag_img(RewriteContext hr){

        images++;

    }

    public void tag_a(RewriteContext hr) {

        links++;

    }

    public boolean done(RewriteContext hr){

        hr.request.props.put("Links", "" + links);

        hr.request.props.put("Images", "" + images);

        return false;}}}
```

Usage de BSL

- `<table border>`
- `<foreach name=a glob=*> <tr>`
- `<td><property a.name></td>`
- `<td><property a.value></td>`
- `<tr>`
- `</foreach>`
- `</table>`

Une table HTML de `server.props`

- BSL :
- `<if> ...<elseif> ... <else> ... </if>`
- `<foreach> ... </foreach>`
- `<property name=xxx>` est remplacé par `xxx`

Usage de BSL : le résultat

Adresse  <http://lmi93.cnam.fr:9999/TestSql.html>

Test SqlTemplate, serveur Brazil, le 22 Mai 2001

main.handlers	sql file
mime.html	text/html
handler	main
sql.url	jdbc:HypersonicSQL:http://163.173.228.80:8181
mime.css	text/x-css-styleSheet
sql.class	sunlabs.brazil.template.TemplateHandler
mime.txt	text/plain
mime.gif	image/gif
mime.jpg	image/jpeg

SqlTemplate

- balise HTML sql

```
<sql prefix=inscrit> select * from lundi</sql>
```

#le fichier de configuration

```
sql.class=sunlabs.brazil.template.TemplateHandler  
sql.templates=sunlabs.brazil.template.PropsTemplate \  
    SqlTemplate \  
    sunlabs.brazil.template.BSLTemplate
```

```
sql.driver=org.hsqldb.jdbcDriver  
sql.url=jdbc:hsqldb:http://163.173.228.80:8181  
sql.sqlPrefix=param  
param.user=sa  
param.password=
```

SqlTemplate + BSLTemplate

```
<sql prefix=inscrit> SELECT * FROM lundi</sql>
```

```
<TABLE border=2>
```

```
<foreach name=index property=inscrit.rows >
```

```
<tr>
```

```
<foreach name=field list="NOM PRENOM SUJET E_MAIL">
```

```
<td>
```

```
<property name=inscrit.LUNDI.${field}.${index}>
```

```
</td>
```

```
</foreach>
```

```
</tr>
```

```
</foreach>
```

```
</TABLE>
```

SqlTemplate : le résultat



Test SqlTemplate, serveur Brazil, le 22 Mai 2001

Lambert	michel	1	mlambert@noos.fr
Robleda	bruno	1	brunodb@club-internet.fr
Nguyen	thuan	1	the_thuan@yahoo.fr
Minchella	michel	1	p_minchella@hotmail.com
Derrien	alain	2	alain_derrien@yahoo.fr
Ciret	evariste	2	eciret@yahoo.fr
Lebougre	herve	2	herve.lebougre@libertysurf.fr
Rodriguese	jose	2	jhrodrigues@wanadoo.fr
Routaboul	joel	2	jroutab@club-internet.fr
Abo	robert	3	Robert.ABO@sema.fr
Martial	eric	4	emartial@club-internet.fr

LDAPTemplate

- balise HTML ldap
 - Même principe que SqlTemplate

```
<ldap host="lmi80.cnam.fr" dn="cn=nomReseau,ou=network,ou=materiel,  
ou=deptinfo,o=cnam.fr" prefix=LDAP>
```

```
<ldap host="lmi80.cnam.fr" base="o=cnam.fr" search="(sn=*)" "  
prefix=PERSONNEL>
```

Note : lmi80.cnam.fr:389 il y a un serveur LDAP, www.openldap.org
dn et search sont exclusifs

#le fichier de configuration

```
tpl.class=sunlabs.brazil.template.TemplateHandler  
tpl.templates= sunlabs.brazil.template.PropsTemplate \  
sunlabs.brazil.ldap.LDAPTemplate \  
sunlabs.brazil.template.BSLTemplate
```

#La compilation de LDAPTemplate.java nécessite ldap40.jar (bibliothèque de Netscape)

BSL encore...

```
<table border=2>
<foreach name=index property=PERSONNEL.rows sort>
  <tr>
    <foreach name=field list="sn telephonenumber">
      <td>
        <property name=PERSONNEL.${index}.${field}>
      </td>
    </foreach>
  </tr>
</foreach>
</table>
```

Le paquetage sunlabs.brazil.template

[AddHeaderTemplate, ChangedTemplate , ContentTemplate , DirectoryTemplate ,
FormClientTemplate , FormTemplate , HighlightTemplate ,
IncludeTemplate , LDAPTemplate , ModifiedTemplate ,
NoImageTemplate , PropsTemplate ,
PythonServerTemplate ,
TclServerTemplate ,
UrlNavBarTemplate]

Quelques templates !

- **public class NoImageTemplate extends Template**

SAMPLE Template class for removing all images from a web page, and replacing them with their alt strings This class is used by the TemplateHandler Each image is replaced by a text string defined by the server property "template", which the first "%" replaced by the contents of the alt attribute.

- **public class IncludeTemplate extends Template**

Template class for substituting html pages into an html page. This class is used by the TemplateHandler. This does not perform traditional server-side include processing, whose normal purpose is to include standard headers and footers. That functionality is provided instead by including the content into the template, and not the other way around.

This include incorporates entire pages from other sites.

- **public class ContentTemplate extends Template**

Template class for extracting content out of remote html page This class is used by the TemplateHandler, for extracting the "content" out of html documents for later integration with a look-and-feel template using one or more of: PropsTemplate, BSLTemplate, or ReplaceFilter, The plan is to snag the title and the content, and put them into request properties. The resultant processed output will be discarded. The following properties are gathered:

À compléter

- N'hésitez pas !

Vers des Fédérations de serveurs Web :

- MultiProxyHandler

+

- Ajout/retrait dynamique de liens (serveurs)

+

- Persistence avec une base de données

+

- utilisation de BSL ...

+

- HashtableHandler/ChainHandler(pour les portails)

➔ *voir également UDDI et WDSL*

Universal Description, Discovery, and Integration

Web Services Description Language

Premières conclusions

- en taille de code sur la TINI Brazil : 45 Ko
- en taille de données :

- améliorations à faire
 - Thread/connexion pré-alloué,
 - ThreadPool.java voir le tp10
 - Pattern ThreadPool, mark Grand volume 3

À suivre ..

Une annexe : les paquetages

sunlabs.brazil.filter
sunlabs.brazil.handler
sunlabs.brazil.ldap
sunlabs.brazil.proxy
sunlabs.brazil.python
sunlabs.brazil.server
sunlabs.brazil.session
sunlabs.brazil.tcl
sunlabs.brazil.template
sunlabs.brazil.util
sunlabs.brazil.util.http
sunlabs.brazil.util.regex

Brazil project framework Developer Documentation

Welcome to the Brazil server developer documentation.

See:

[Description](#)

[Handler option summaries](#)

Packages

sunlabs.brazil.filter	Filters are a type of handler, used by the filterHandler that can modify content after it has been obtained by another handler.
sunlabs.brazil.handler	This package contains a collection classes that implement the Handler interface for use with the Server package, along with several support classes.
sunlabs.brazil.ldap	Provide ways of integrating LDAP into the Brazil project server.
sunlabs.brazil.proxy	Handlers, filters, and utilities for using the Brazil project framework as an HTTP proxy.
sunlabs.brazil.python	Provide ways of integrating the Python scripting language into the Brazil project server.
sunlabs.brazil.server	Generic http protocol stack, essential handlers and drivers.
sunlabs.brazil.session	A generic, extensible mechanism for managing session state.
sunlabs.brazil.tcl	Provide ways of integrating the TCL scripting language into the Brazil project server.
sunlabs.brazil.template	Template classes for use with TemplateHandler or TemplateFilter for filtering HTML and XML content.
sunlabs.brazil.util	Utility classes that are generically useful in Java language programs.
sunlabs.brazil.util.http	Utility classes for dealing with the HTTP protocol.
sunlabs.brazil.util.regex	This package contains a converted-to-Java-language version of Henry Spencer's regular expression package contained in TCL version 8.0.