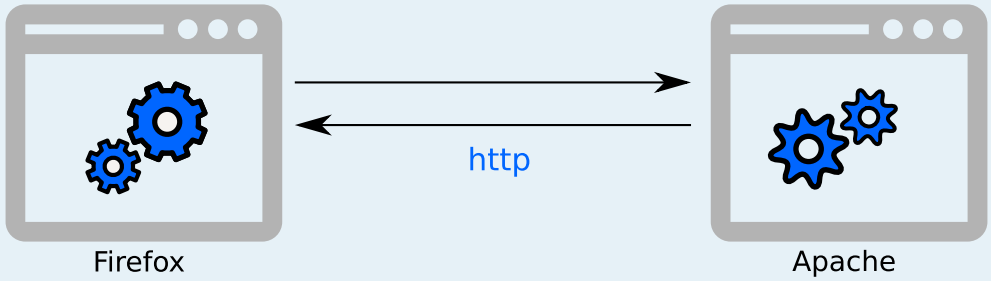
Introduction aux servlets

Java EE

Nicolas Delanoue

Université d'Angers - Polytech Angers

Architecture client serveur en http



Exemple d'échange en utilisant le protocole http

```
$ telnet www.perdu.com 80                                   Connexion au serveur par telnet
Trying 208.97.177.124...
Connected to www.perdu.com.
Escape character is '^]'.

GET / http/1.1                                              Requête HTTP
Host: www.perdu.com

HTTP/1.1 200 OK                                             Réponse du serveur : headers
Date: Sat, 17 Aug 2013 11:59:04 GMT
Server: Apache
Accept-Ranges: bytes
X-Mod-Pagespeed: 1.1.23.1-2169
Vary: Accept-Encoding
Cache-Control: max-age=0, no-cache
Content-Length: 204
Content-Type: text/html

<html><head><title>Vous Etes Perdu ?</title></head><body><h1>Perdu sur l'Interne
t ?</h1><h2>Pas de panique, on va vous aider</h2><strong><pre>    * <----- vous
&ecirc;tes ici</pre></strong></body></html>                 Réponse du serveur : body
```

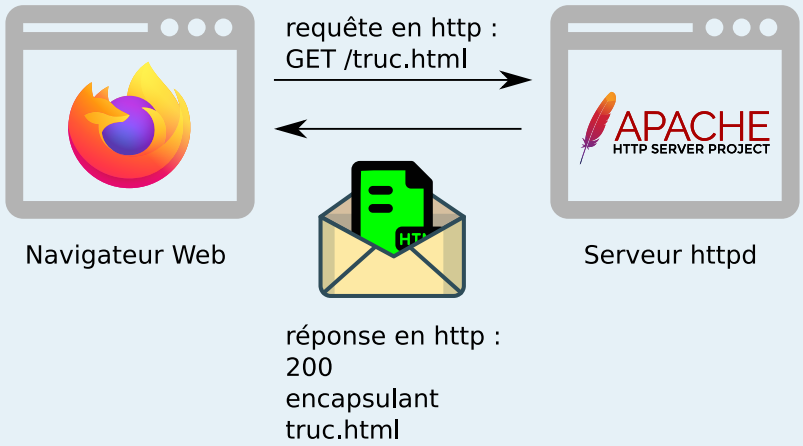
Définition

En HTTP, une *méthode* est une commande spécifiant un type de requête, i.e. elle demande au serveur d'effectuer une action.

Principales commandes en HTTP

- `GET` : méthode la plus courante pour demander une ressource,
- `HEAD` : méthode qui ne demande que des informations sur la ressource (sans demander la ressource elle-même),
- `POST` : méthode utilisée pour transmettre des données en vue d'un traitement à une ressource.

Exemple d'échange en utilisant le protocole http



Définition

En HTTP, le *code code d'état* permet de déterminer le résultat d'une requête.

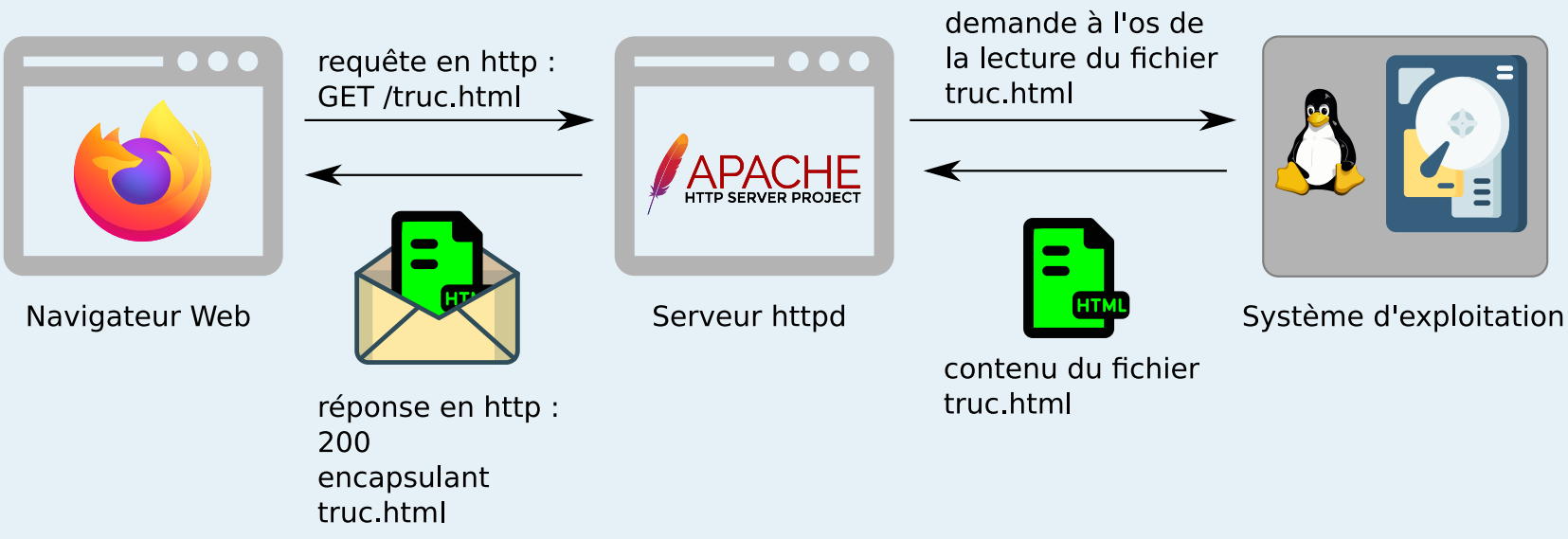
Exemples courants

- 200 : succès de la requête,
- 301 et 302 : redirection, respectivement permanente et temporaire,
- 403 : accès refusé,
- 404 : page non trouvée,
- 500 et 503 : erreur serveur,
- 504 : le serveur n'a pas répondu.

Définition

Une page web statique est une page web dont le contenu ne varie pas en fonction de la demande. i.e. tous les internautes qui demandent la page reçoivent le même contenu.

Exemple

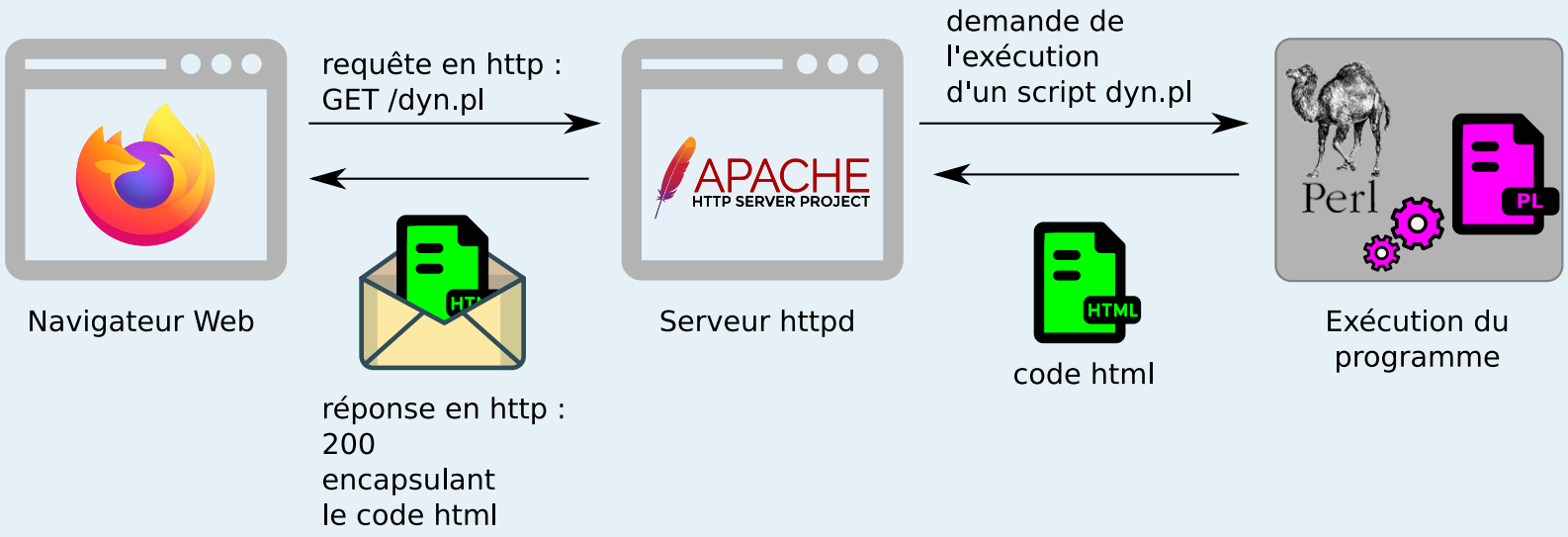


Définition

Une page web dynamique est générée à la demande et son contenu varie en fonction des caractéristiques de la demande.

- heure, adresse IP du demandeur,
- formulaire rempli par le demandeur,
- cookie transmis par le navigateur,
- ...

Exemple avec un script cgi

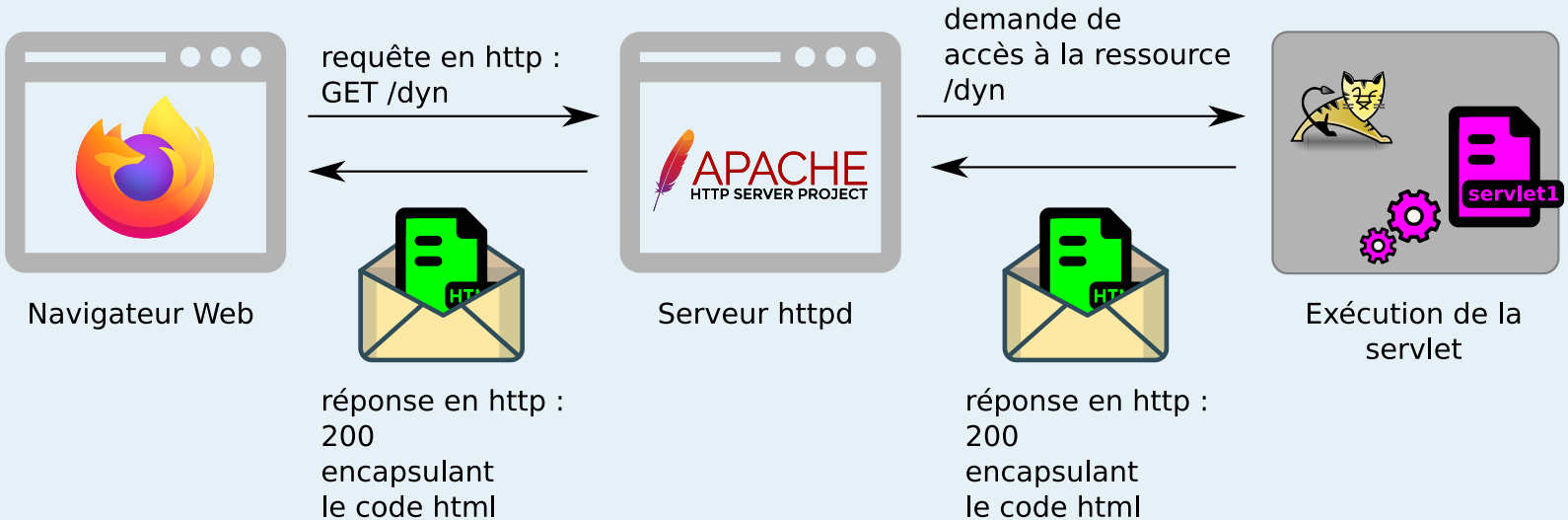


Définition

Une page web dynamique est générée à la demande et son contenu varie en fonction des caractéristiques de la demande.

- heure, adresse IP du demandeur,
- formulaire rempli par le demandeur,
- cookie transmis par le navigateur,
- ...

Exemple avec un serveur d'applications



Spécifications de Java EE

Contrat entre le conteneur Web et les composants :

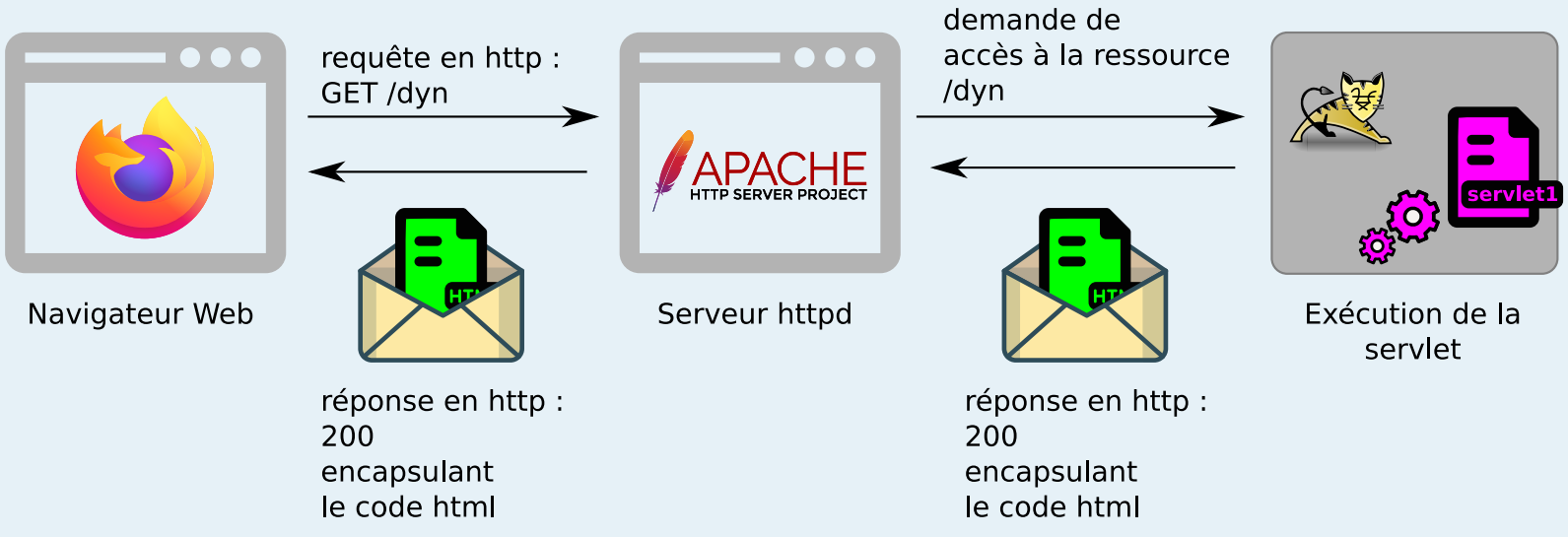
- pour définir le cycle de vie des composants,
- pour définir le comportement des composants,
- pour définir les services que le serveur offre aux composants.

Pages dynamiques avec Java EE

Deux types de composants Web dans la spécification Java EE :

- les Servlets,
- les JSP (Java Server Pages).

Exemple avec un serveur d'applications



Définition

Une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur (souvent HTTP).

Remarques

- Les servlets utilisent l'API Java Servlet (package javax.servlet).
- Une servlet est un fichier .class sur le serveur,
- Elle s'exécute sur le serveur.
- Réaliser une servlet, c'est écrire une classe java.

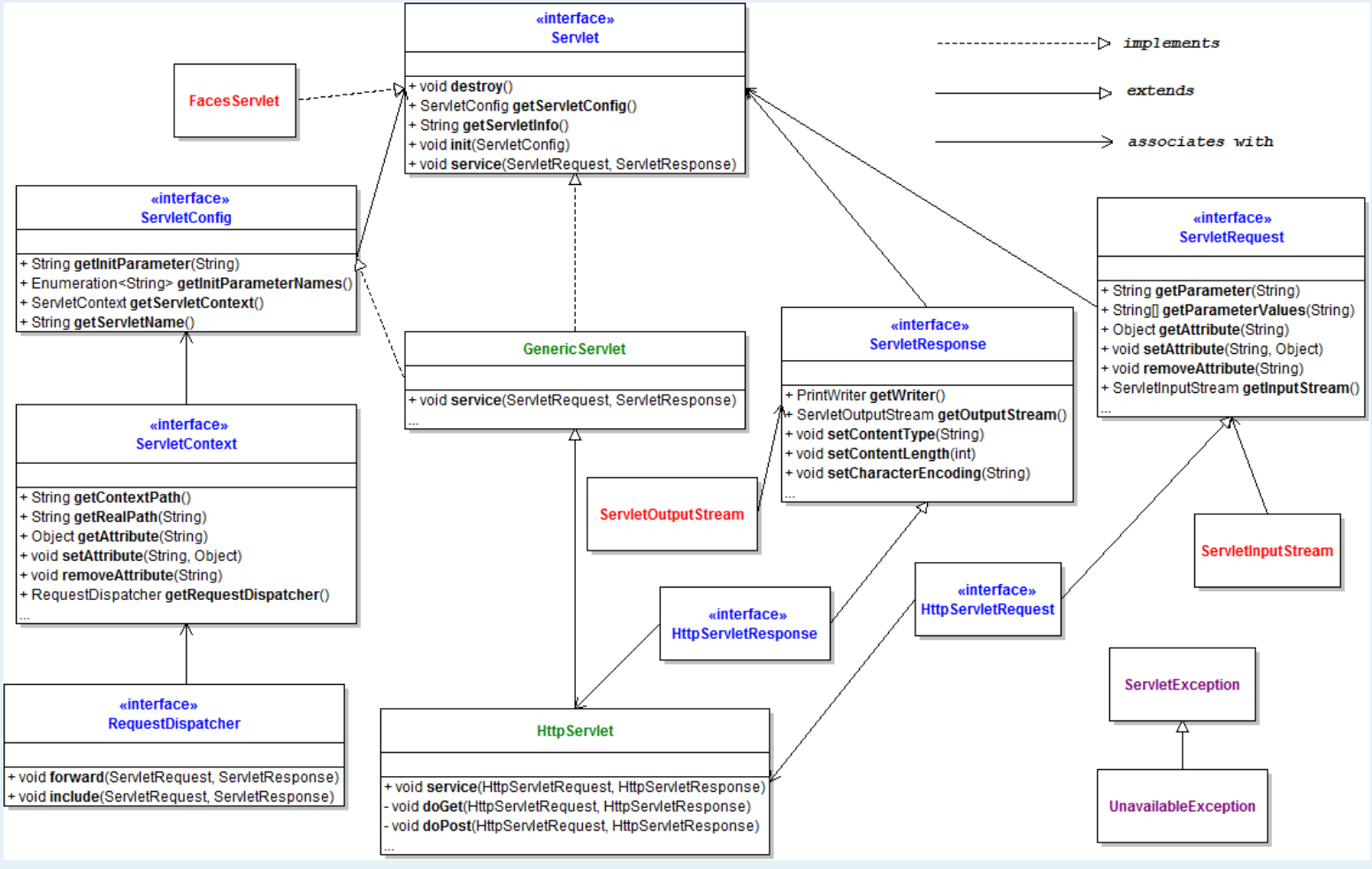
Réalisation d'une servlet générique "Hello world !"

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

public class HelloServlet extends GenericServlet
{
 @Override
 public void service(ServletRequest req, ServletResponse res)
   throws ServletException, IOException {
    try
     {
     PrintWriter out = res.getWriter() ;
     out.println ("<!DOCTYPE html>") ;
     out.println ("<title>Bonjour tout le monde !</title>") ;
     out.println ("<p>Hello world!</p>") ;
     out.close();
     }
   catch (IOException e) { e.printStackTrace() ; }                    }
}
```

Sortie générée par cette servlet :

```
<!DOCTYPE html>
<title>Bonjour tout le monde !</title>
<p>Hello world!</p>
```

Diagramme de classes UML



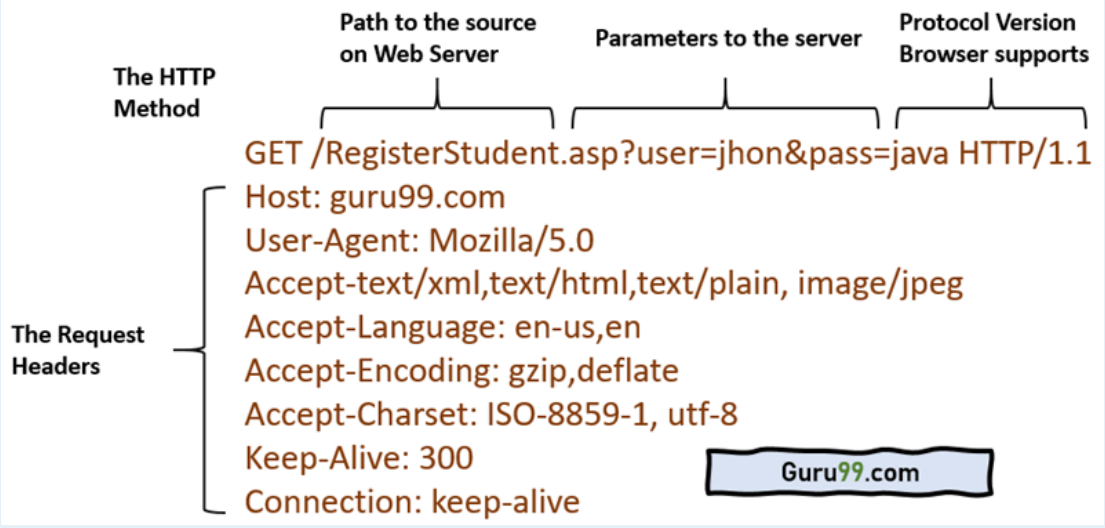
La classe HttpServlet implémente (en autre) deux méthodes : `doGet()` et `doPost()`

Principales caractéristiques de GET

- On peut passer des paramètres directement dans l'url.
- La longueur de la méthode GET est limitée.

Requête http

```
GET /RegisterStudent.asp?user=jhon&pass=java
```

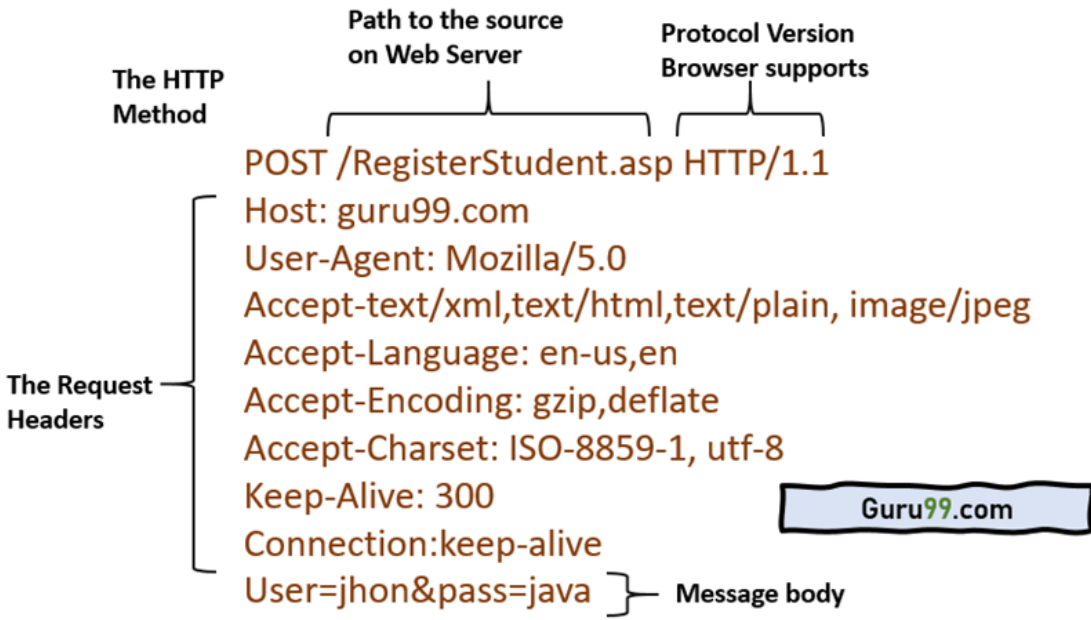


Principales caractéristiques de POST

- Les données transmises ne seront pas visibles dans l'URL,
- les paramètres ne sont pas enregistrés dans l'historique du navigateur,
- aucune restriction concernant la longueur des données,
- sécurité des informations sensibles et confidentielles.

Requête http

```
POST /RegisterStudent.asp HTTP/1.1
Host: www.guru99.com
user=value1&pass=value2
```



Exemple de servlet HttpServlet "Hello world !"

```java
// Ajout des librairies java nécessaires
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;

// Etend la classe HttpServlet
public class HelloWorld extends HttpServlet {

   // Attribut de la classe :
   private String message;

   // Initialisation: (cette méthode n'est exécutée qu'une fois pour toutes)
   public void init() throws ServletException {
      message = "Hello World";
   }

   // Traitement d'un http get: (cette méthode est exécutée pour chaque requête)
   public void doGet(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
      // Précise de quel type va être la réponse
      response.setContentType("text/html");
      // La partie ``dynamique'' apparait ici :
      PrintWriter out = response.getWriter();
      out.println("<h1>" + message + "</h1>");
   }

   // Méthode exécutée juste avant la destruction de l'objet.
   public void destroy() {
   }
}
```

La méthode `doGet()` est exécutée lors d'une requête http `GET`.

Exemple de méthode doGet()

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException {
    // Précise de quel type va être la réponse
    resp.setContentType("text/html");
    // La partie ``dynamique'' apparait ici :
    PrintWriter out = response.getWriter();
    out.println("Aujourd'hui, nous sommes le " + new Date() + ".");
    out.close();
    }
```

Rôle des objets request et response

- L'objet de la classe `HttpServletRequest` contient les données transmises par le client (automatiquement formatées par le conteneur Web).
- L'objet de la classe `HttpServletResponse` contiendra la réponse html transmise au client.

Méthodes de la classe HttpServletRequest

- `String getParameter(String param)` permet de récupérer la valeur associée au paramètre `param`
- `Enumeration getParameterNames()` retourne l'ensemble des noms des paramètres.
- ...

Exemple d'utilisation

Pour le traitement de la requête http `GET /MaServlet?nom=nicolas`, on écrit :

```
public void doGet(HttpServletRequest req, HttpServletResponse resp)
  throws ServletException, IOException {
    // Précise de quel type va être réponse
    String name = req.getParameter("nom");
    PrintWriter out = resp.getWriter();
    out.println("Bonjour " + name.toUpperCase() + ".");
    out.close();
  }
```

Code html obtenu : `Bonjour NICOLAS`

Méthodes de la classe `HttpServletResponse`

- `void setContentType(String type)` définit le type MIME du document généré par l'exécution de la servlet.
- `PrintWriter getWriter()` renvoie le flux de sortie permettant à la servlet de générer son résultat.
- ...

Environnement d'exécution des servlets

- Une servlet est exécutée dans un contexte particulier mis en place par le moteur de servlets.
- Elle peut obtenir des informations à partir :
  - de la requête http du client (get, post, cookie, ...)
  - des données stockées lors d'une session précédemment créée,
  - du contexte de l'application (données partagées par tous les clients).

Exécution d'une servlet

- Lors du *chargement initial*, le conteneur Web instancie et initialise la servlet et exécute de sa méthode `Init()`.
- Lors des *appels suivants*, le conteneur Web crée un thread pour exécuter la code adhoc de la servlet ,

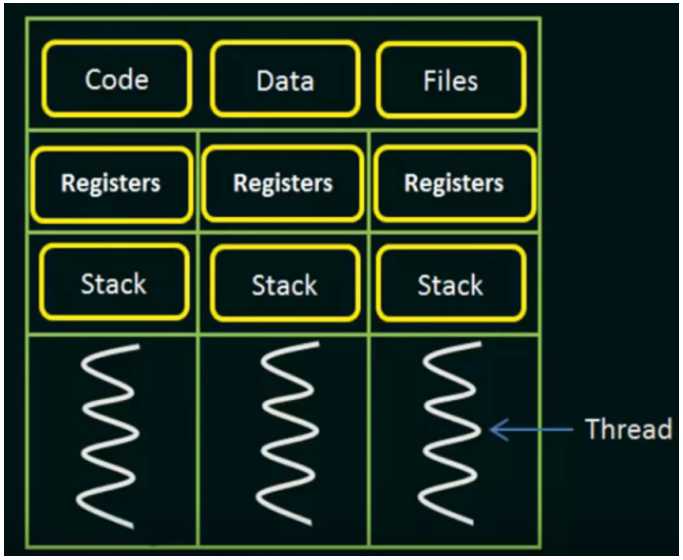


FIGURE – Processus avec 1 thread.

Exécution d'une servlet

- Lors du *chargement initial*, le conteneur Web instancie et initialise la servlet et exécute de sa méthode `Init()`.
- Lors des *appels suivants*, le conteneur Web crée un thread pour exécuter la code adhoc de la servlet ,

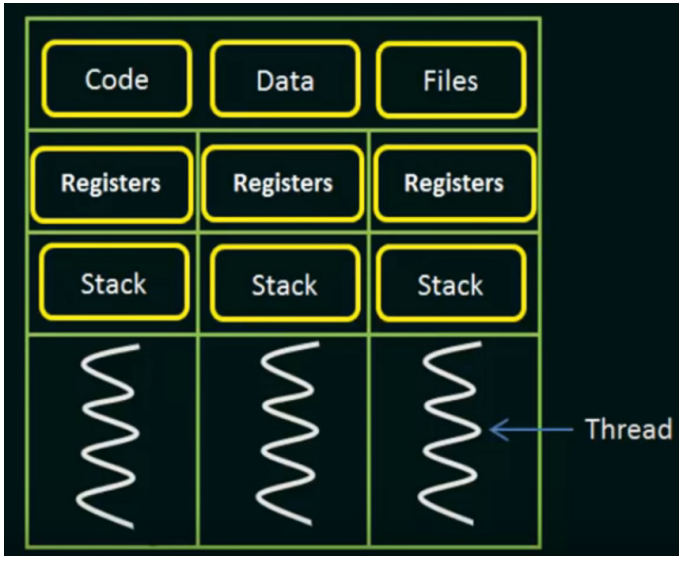


FIGURE – Processus avec 1 thread.

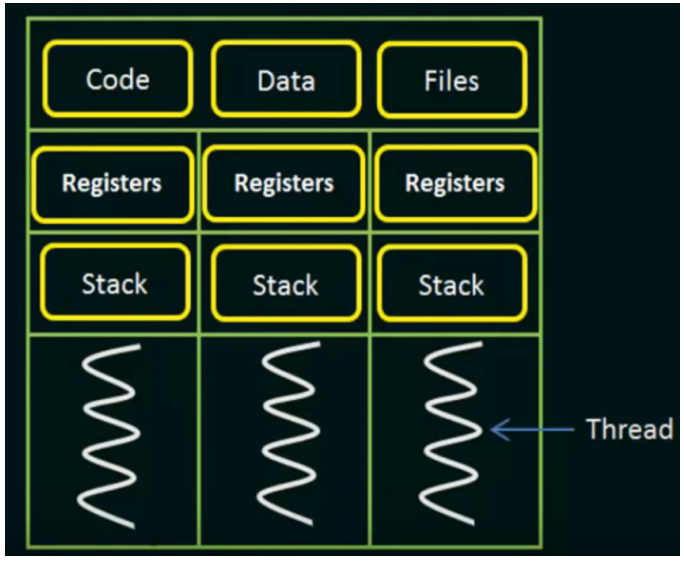
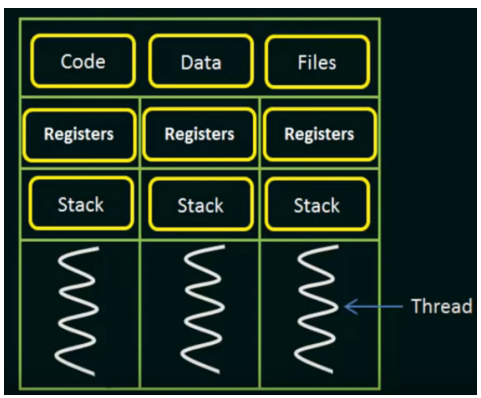


FIGURE – Processus avec 3 threads.

Illustration

```
public class CompteurServlet extends HttpServlet {
  // Cet attribut est commun à tous les threads
  public int compteur = 0;
  public void doGet(HttpServletRequest req, HttpServletResponse resp)
     throws ServletException, IOException {
       // Chaque thread incrémente cette variable
       compteur++;
       resp.setContentType( "text/html" );
       PrintWriter out = resp.getWriter();
       out.println( "<html><body>" );
       out.println( "<h1> "+ compteur  + "</h1>" );
       out.println( "</body></html>" );
     }
  }
```



Une page dynamique

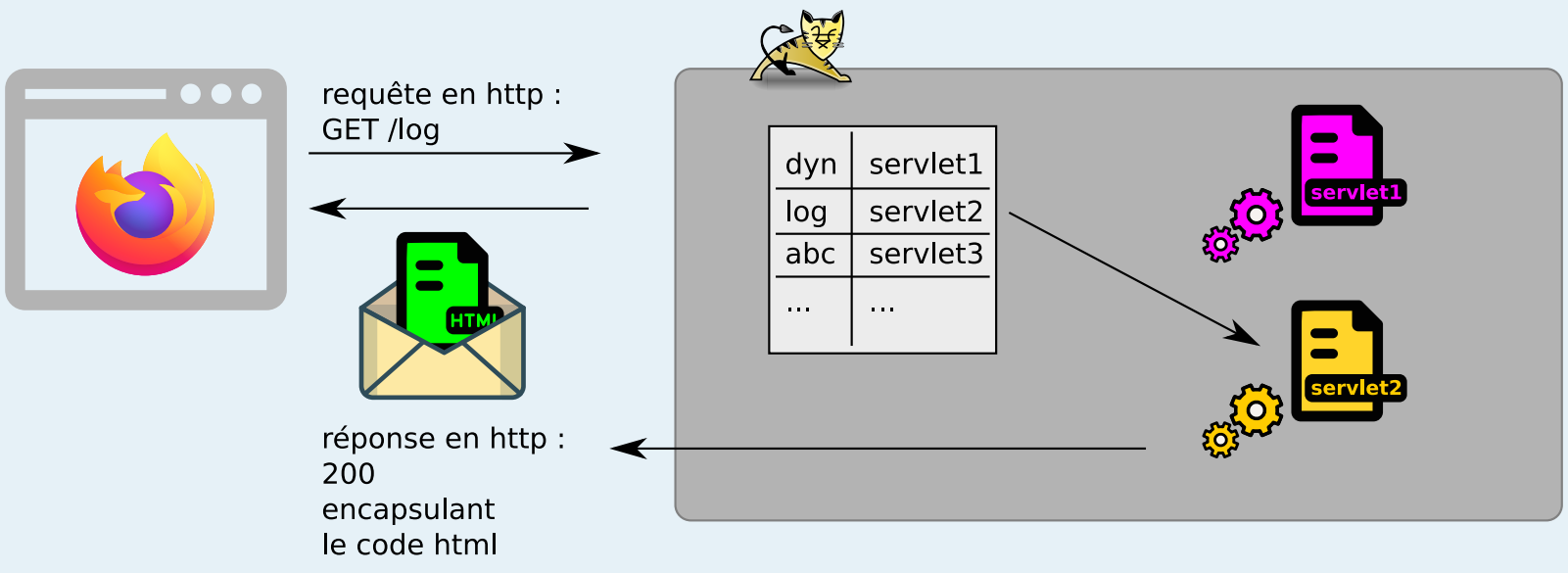
```
public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
    // Précise de quel type va être réponse
    res.setContentType("text/html");
    // La partie ``dynamique'' apparait ici :
    PrintWriter out = res.getWriter();
    out.println("Aujourd'hui, nous sommes le " + new Date() + "<br>");
    out.close();
}
```

Remarques

- Chaque client verra l'heure s'affichée en fonction de l'heure à laquelle il a provoqué l'exécution du thread.
- Les objets "`newisés`" sont dans la pile.

Classiquement, l'accès à une servlet suit les étapes suivantes :

1. via l'url, le navigateur crée une requête http,
2. le serveur http réceptionne cette requête et sollicite une servlet,
3. la servlet s'exécute.



Réalisation

Cette table de correspondance est construite à partir des annotations `@WebServlet`.

Exemple de servlet annotée

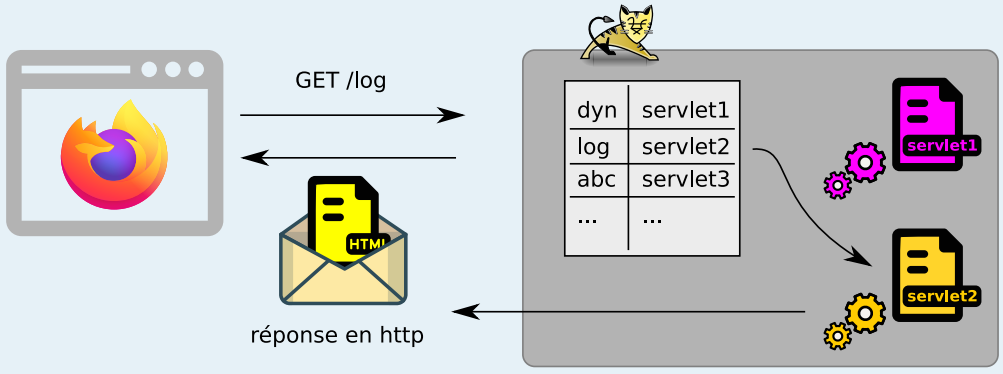
```
@WebServlet("/authentification")
public class servlet2 extends HttpServlet {

  protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
            res.getWriter().append("Ici, on se logue ;-)");
        }
```

Plusieurs motifs d'URL pointant vers la même servlet :

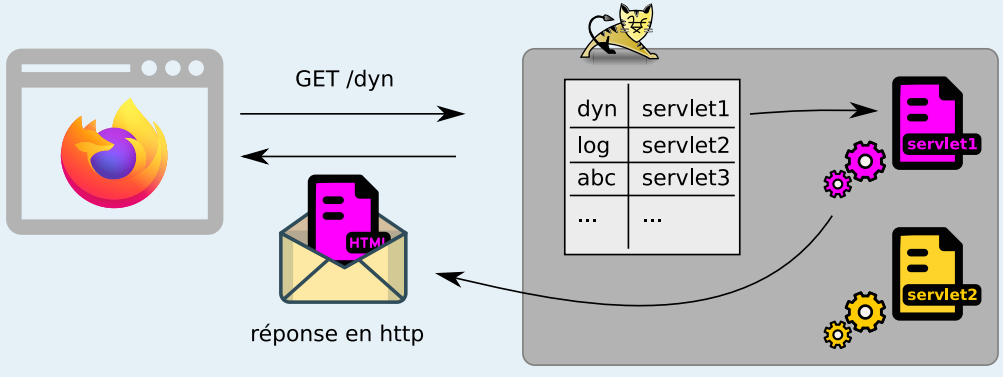
```
@WebServlet({"/hello", "/bonjour"})
```

```
@WebServlet(
  urlPatterns = "/fichiers/*",
  initParams = @WebInitParam(name = "chemin", value = "/fichiers/"))
```

Client 1, accès à 10h00



Client 1, accès à 10h01



Remarque

En http, d'une requête à une autre, rien ne permet de reconnaitre nativement un client.

http est "sans état"

En http, d'une requête à une autre, rien ne permet de reconnaitre nativement un client.

ftp est "avec état"

En ftp et par exemple, la même commande n'aura pas les mêmes conséquences en fonction de l'état ...

```
cd mes_documents
put fichier1.mp3
```

```
cd ma_musique
put fichier1.mp3
```

Quelques techniques classiques de suivi de session

1. réécriture d'url,
2. cookies (et sessions),
3. champs masqués dans les formulaires,
4. ...

Implémentation d'une session - écriture

```
@WebServlet("/page1")
public class page1 extends HttpServlet {
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

      HttpSession session = req.getSession();
      session.setAttribute("jeu", "the incredible machine");

    }
  }
```

Implémentation d'une session - lecture

```
@WebServlet("/page1")
public class page2 extends HttpServlet {
  protected void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {

      HttpSession session = req.getSession();
      String mon_jeu = (String) session.getAttribute("jeu");

    }
  }
```

Remarque

- Le cast est obligatoire car le prototype de la méthode getAttribute est

  ```
  Public Object getAttribute(String nom);
  ```

- Où sont stockés ces données ?

Méthodes disponibles sur un objet de classe HttpSession

- `void setAttribute(String name, Object value)`
- `Object getAttribute(String name)`
- `Enumeration getAttributeNames()`
- `void removeAttribute(String name)`
- `void setMaxIntervalTime(int seconds)`
- ...

Conclusion

Une servlet est un programme java, qui

- s'exécute sur le serveur (sous forme de threads),
- peut être invoqué via un url,
- génère du code html.

Cas d'utilisation classique

Traitement contenant majoritairement du code java (i.e. peu d'html)

Lors du TP, nous en apprendrons encore plus sur :

1. les cookies,
2. le contexte,
3. le chaînage de servlets,
4. l'annotation `@Webfilter` et l'interface `Filter`.