Programmation d'applications d'entreprise

Chapitre 2 - Servlet

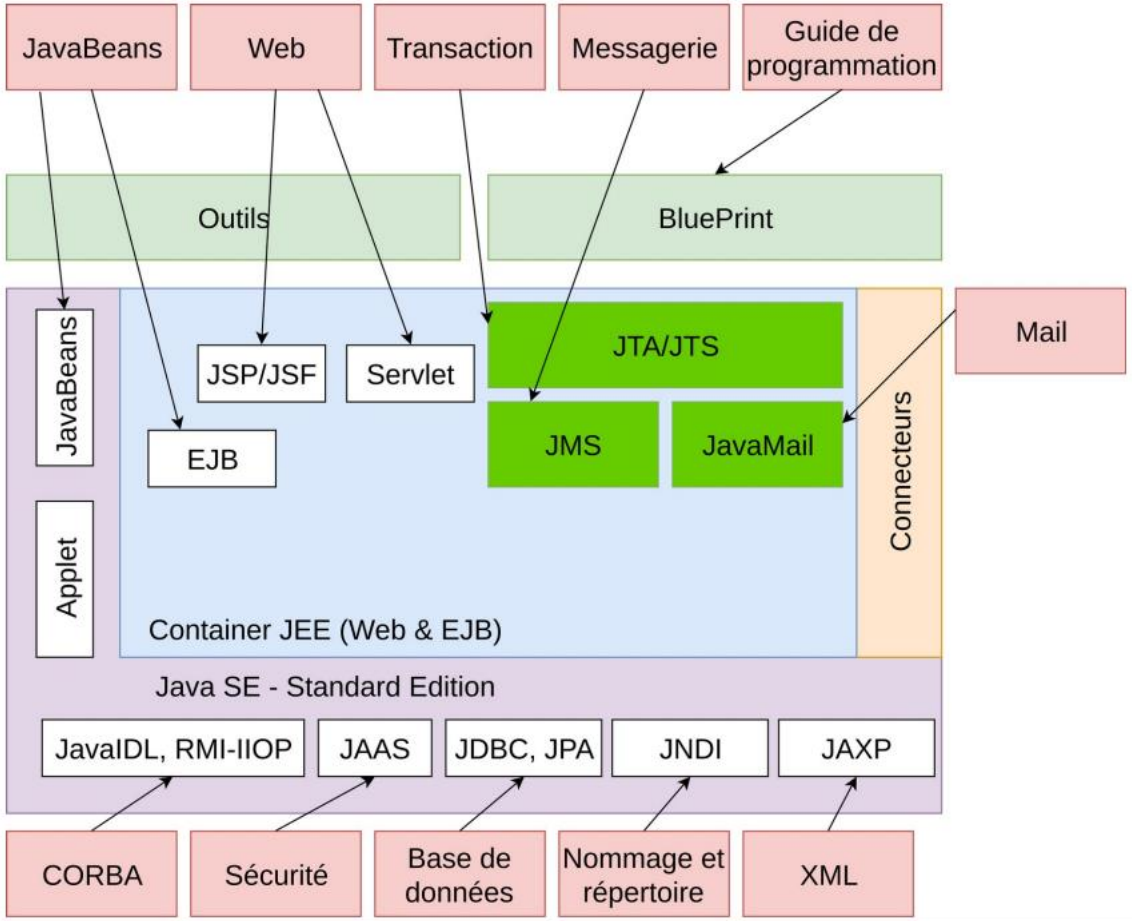
UFR Sciences et Techniques - Antenne de Blois
Sameh.kchaou@univ-tours.fr

Supports de cours adaptés de Anthony Bocci

Plan

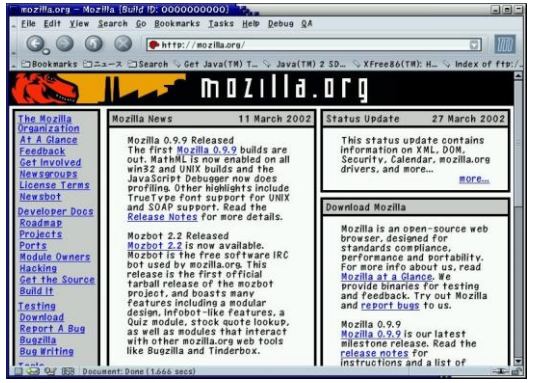
- L'architecture de la plateforme **Java EE**
- Fonctionnement des servlets
- Gestion de l'état
- Gestion des session
- Gestion des cookies

L’architecture de la plateforme Java EE

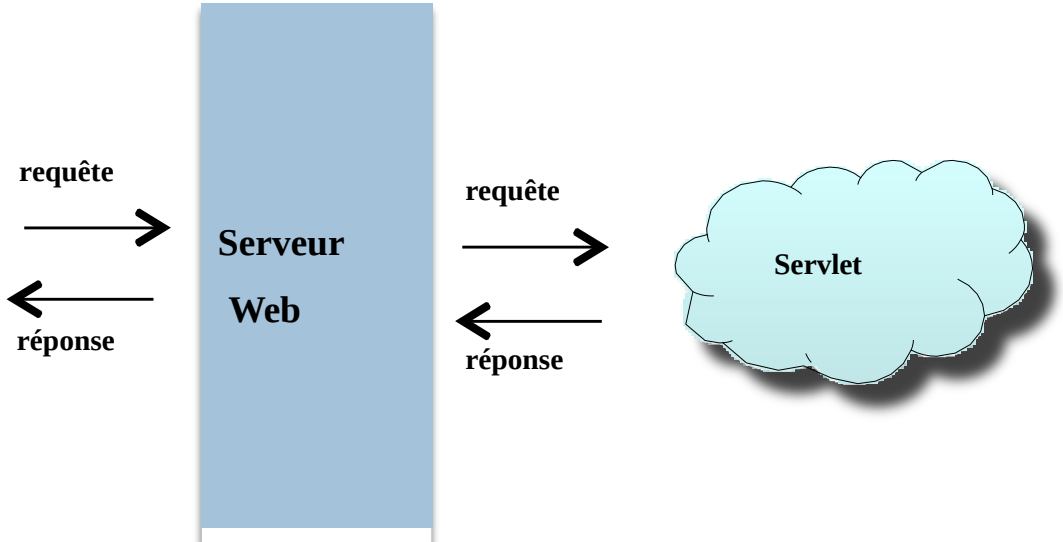


SERVLET

- Permettre la programmation d'**applications web** en Java
  - ✓ servlet web ou servlet (*pour simplifier*)
- Servlet : **Serv**er-side app**let**
  - ✓ Applet est une classe s'exécutant chez le client alors que la servlet s'exécute chez server
- Une classe java s'exécutant coté serveur
  - ✓ reçoit et répond aux requêtes de clients web
  - ✓ généralement via HTTP



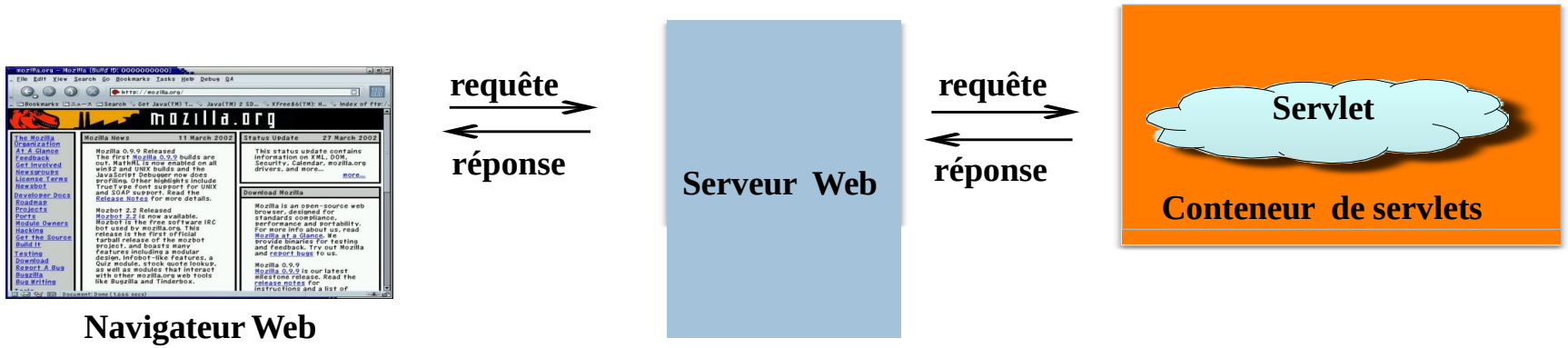
Navigateur Web



SERVLET

❖ ENVIRONNEMENT D'EXÉCUTION

- Les servlets s'exécutent dans un **conteneur de servlets**, appelé aussi **conteneur web**
  - ✓ Conteneur web = serveur web + *moteur* de servlet
  - ✓ Etablit le lien entre le serveur web et la servlet
- Exemples de conteneur de servlets
  - ✓ Tomcat
  - ✓ Jetty
  - ✓ Weblogic ...



SERVLET

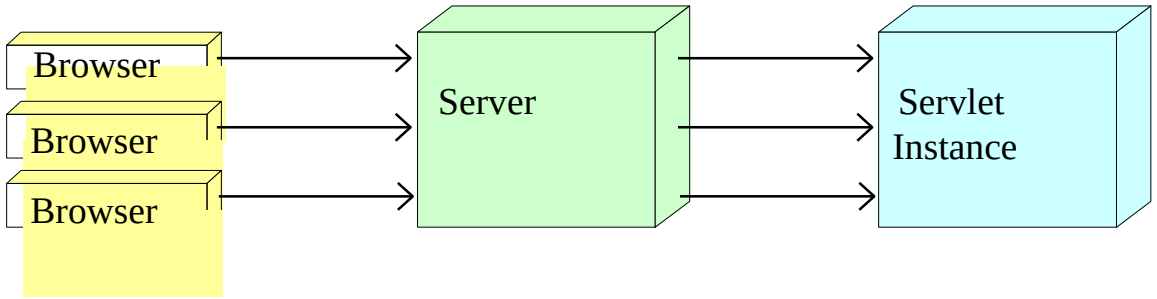
❖ SQUELETTE D'UNE SERVLET

- Une servlet doit implémenter **l'interface** javax.servlet.Servlet
- Cette interface possède les méthodes **init()**, **service()** et **destroy()**

```
Import javax.servlet.*; //l'interface Servlet
public class MyServlet implements Servlet {
    public void init(ServletConfig config) throws ServletException {
        … //appelée par le conteneur de servlets lorsqu'une instance de la servlet est initialisée pour la première fois
    }
    public void service( ServletRequest request, ServletResponse response ) throws ServletException,
            IOException {
        … //reçoit un objet ServletRequest, qui contient les informations de la requête, et un objet ServletResponse, qui permet de formuler la réponse.
    }
    public void destroy() {
        … //pouvez libérer les ressources allouées par la servlet
    }
```

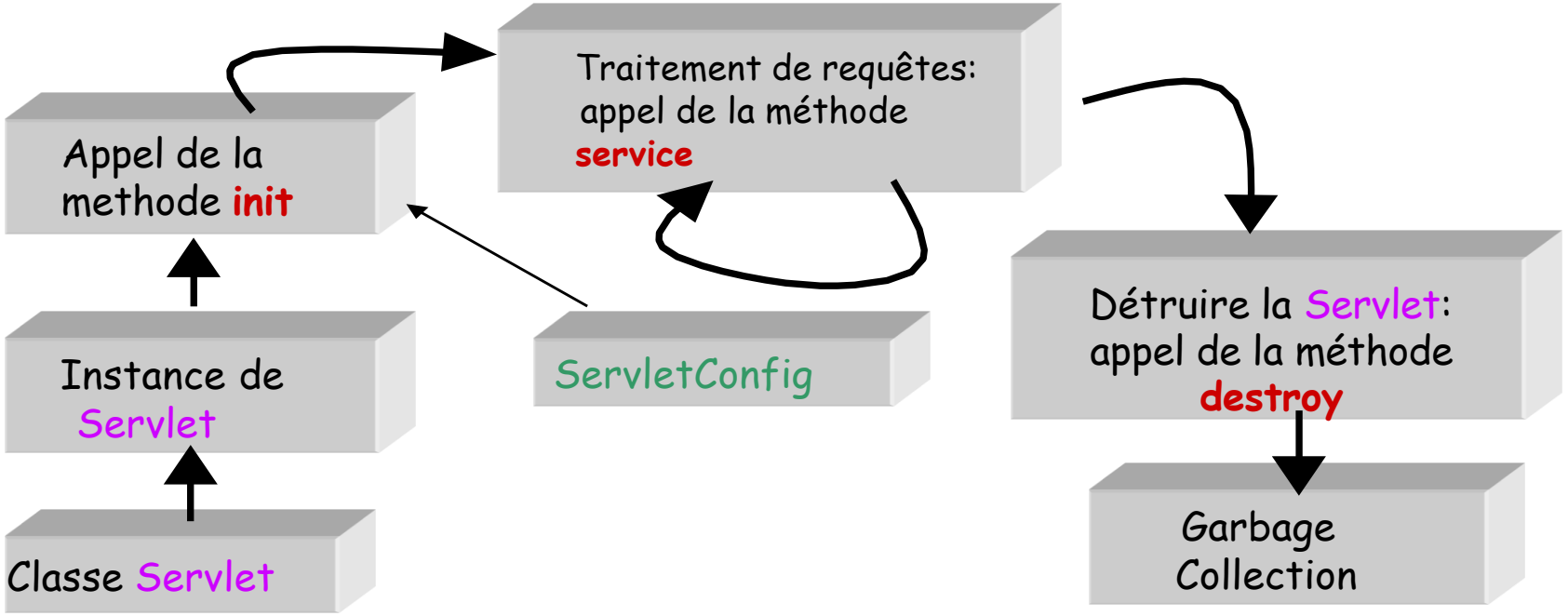
CYCLE DE VIE D'UNE SERVLET

- Le conteneur gère le cycle de vie :
  - Charge la classe de servlet et en initialise une instance
  - Chaque requête est traitée par une instance de servlet dans un **thread séparé**
  - Peut supprimer la servlet
  - Peut garder la servlet pour traiter d'autre requêtes



CYCLE DE VIE D'UNE SERVLET

- Lorsque la servlet est instancié, le conteneur appelle la méthode init() pour initialiser la servlet
- Sur demande du client, le conteneur appelle la méthode service()
- Avant la suppression de la servlet, le conteneur appelle la méthode destroy()



HTTP et Servlets

- HTTP décrit les échanges entre le navigateur et le serveur Web.
- Format d'une **requête HTTP** :
  - Ligne de commande **(commande ou la méthode http** , URL, version de protocole)
  - En-têtes de la requête (des informations supplémentaires au serveur)
  - Corps de la requête
- Format d'une **réponse HTTP** :
  - Ligne de status (version de protocole, code réponse, texte de réponse)
  - En-têtes de la réponse (des informations sur le type de contenu renvoyé)
  - Corps de la réponse

HTTP et Servlets

- En HTTP, le code d'état permet de déterminer le résultat d'une requête
  - Exemples Courants
    - 200 : succès de la reqête
    - 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

HTTP et Servlets

❖ Méthodes http

- En HTTP, une méthode est une commande spécifiant un type de requête, elle demande au serveur d'effectuer une action.
- **Principales commandes en HTTP**
  - *GET* : Méthode la plus courante pour demander une ressource.
  - *POST* : Méthode utilisée pour transmettre des données en vue d'un traitement à une ressource.
  - *HEAD* : Méthode qui ne demande que des informations sur la ressource (sans demander la ressource elle-même)

HTTP et Servlets

❖ HTTP GET

- **Principales caractéristiques de GET :**
  - On peut passer des paramètres (clé-valeur) directement dans l'url.
  - La longueur d'une requête GET est limitée.
- Requête HTTP
  - GET http://localhost:8080/Servlet/ParamServlet?id=42&movie=Terminator

HTTP et Servlets

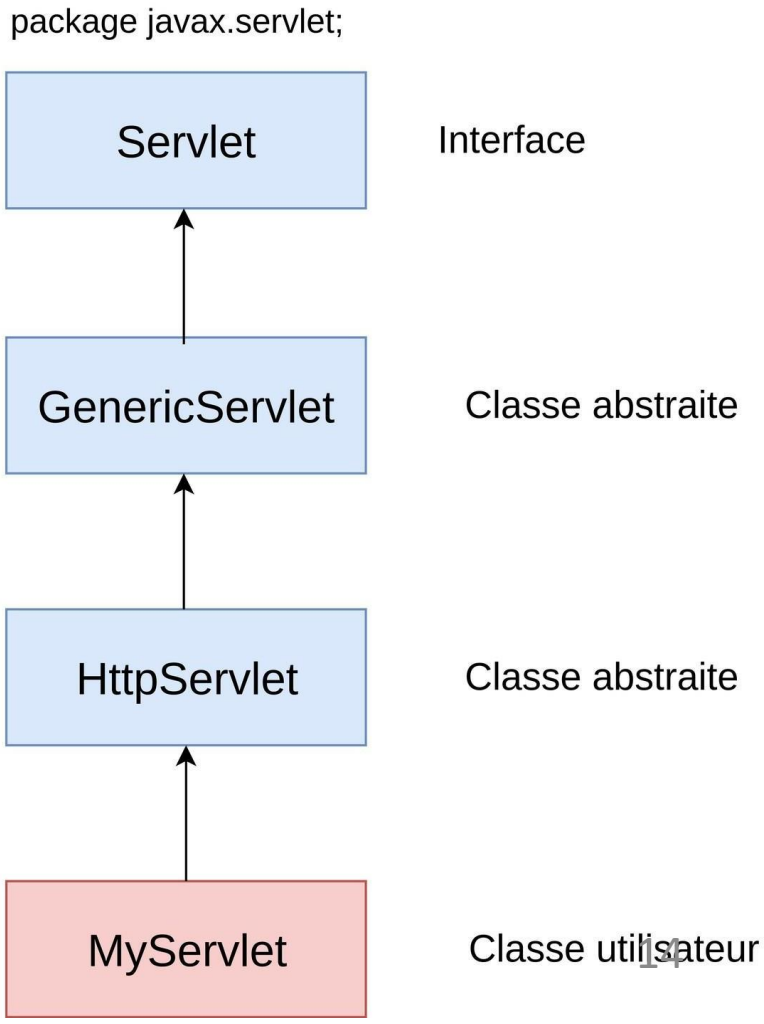
❖ HTTP POST

- Principales caractéristiques de POST
  - Les données transmises ne seront pas visible 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

HTTP et Servlets

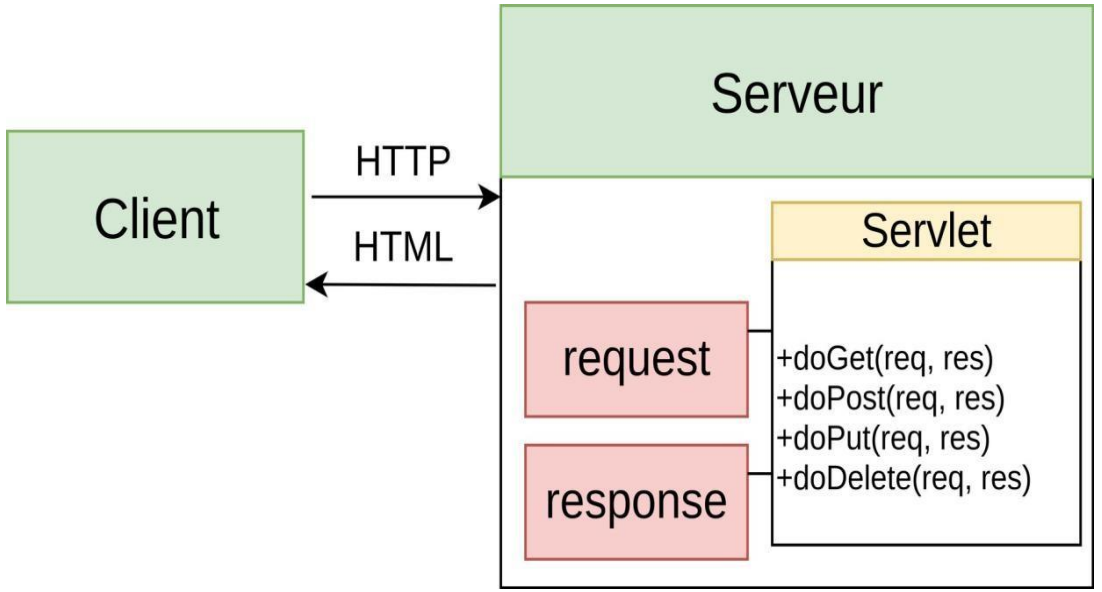
❖ Servlet Web

- Pour faciliter le traitement des requêtes HTTP, la classe de servlet utilisée est javax.servlet.HttpServlet
  - La méthode service() est remplacée par :
    - doGet()
    - doPost()
  - La classe doit implémenter l'une des méthodes redéfinies, selon son fonctionnement (formulaire GET ou POST par exemple)



DÉVELOPPEMENT DE SERVLET

- Une servlet Web :
  - Étend *javax.servlet.HttpServlet*
  - Implémentation de la méthode appropriée :
    - doGet() si méthode GET
    - doPost() si méthode POST
  - Reçoit les **requêtes** et renvoie les **réponses** via deux objets :
    - *javax.servlet.ServletRequest* contient les informations de la requête
    - *javax.servlet.ServletResponse* contient les informations de la réponse



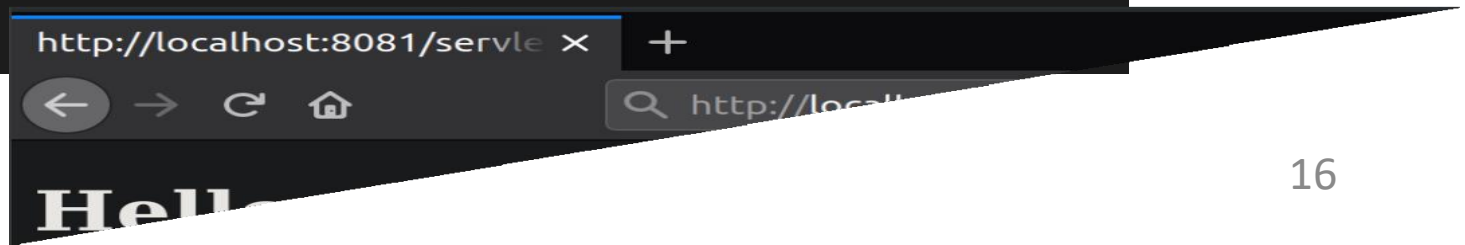
SERVLET WEB : EXEMPLE

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

public class HelloServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException { // <- Imposé par l'API Servlet

        res.setContentType("text/html"); // <- Valeur du header Content-Type
        PrintWriter out = res.getWriter(); // Récupère le flux de sortie
        out.println("<!DOCTYPE html><html><body><h1>Hello, world!</h1></body></html>");

    }
}
```



DÉVELOPPEMENT DE SERVLET

❖ APERÇU DE L'INTERFACE HTTP SERVLET REQUEST

- Principales méthodes :
  - *String getParameter(String name)*
    - Retourne la valeur du paramètre « name » de la requête
  - *Enumeration getParameterNames()*
    - Retourne le nom de tous les paramètres transmis à la requête
  - *String[] getParameterValues()*
    - Retour toutes les valeurs des paramètres transmis à la requête
  - String getMethod()
    - Retourne la méthode HTTP utilisée
  - *String getHeader(String name)*
    - Retourne la valeur du header HTTP « name »
  - *Enumeration getHeaderNames()*
    - Retourne tous les noms des headers HTTP de la requête
  - *String[] getHeaderValues()*
    - Retourne toutes les valeurs des headers HTTP
  - **Exemple**: Une requête donne accès à ses paramètres avec la méthode **getParameter()**.
    https://localhost/Cours?id=42&movie=Terminat

```
1String id = request.getParameter("id"); // = "42"
2String movie = request.getParameter("movie"); // = Terminator
```

DÉVELOPPEMENT DE SERVLET

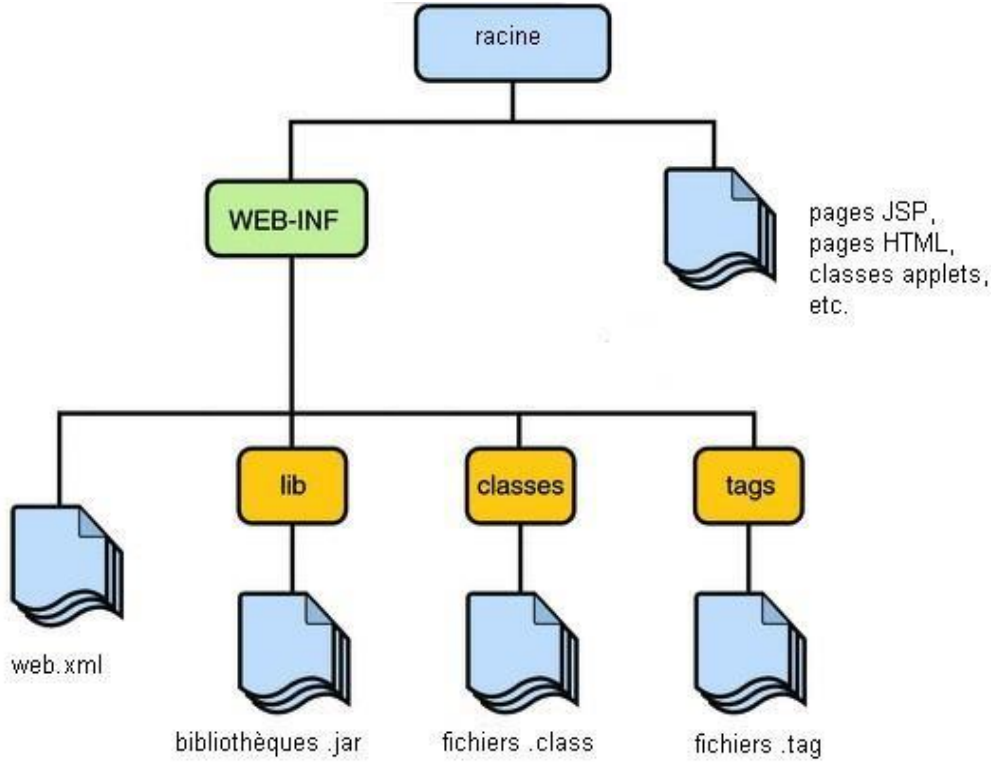
❖ APERÇU DE L'INTERFACE HTTP SERVLET RESPONSE

- Principales méthodes :
  - *PrintWriter getWriter() :* Retourne un PrintWriter qui permet d'envoyer le corps de la réponse au client
  - *void setContentLength(int length) :* définit la longueur du contenu de la réponse HTTP que le serveur enverra au client
  - *void setContentType(String type) :* Définit le header content-type / type MIME
  - *void sendError(int sc, String message) throws java.io.IOException :* Envoie une erreur au client
  - *void setHeader(String name, String value) :* Définit le header « name » avec la valeur « value »

DÉVELOPPEMENT DE SERVLET

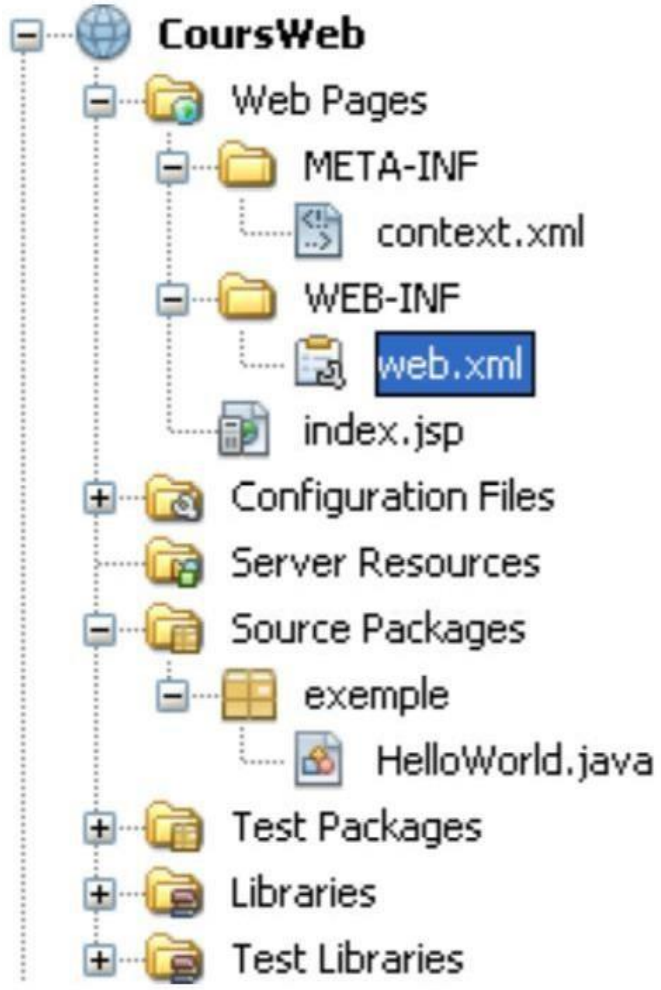
❖ DÉPLOIEMENT D'UNE SERVLET

- Une servlet doit faire partie d'un **module Web** pour être déployée.
  - Un module Web est un ensemble de librairies, fichiers de configuration Java, de code Java, etc.
- Pour déployer une application dans un container, il faut :
  - Tous les composants (classes compilées, ressources, etc) regroupées dans une archive ou module
  - Un descripteur de déploiement contenu dans le module qui précise au conteneur des options pour exécuter l'application.



DÉPLOIEMENT D'UNE APPLICATION WEB

- Une application Web est :
  - Packagée dans une archive **war (.war)**
  - Paramétrée par le fichier **WEB-INF/web.xml**
  - Exécutée par un serveur Web
- Structure :
  - Fichiers Web (HTML, JS, CSS, JSP)
  - Répertoire **META-INF**
    - MANIFEST.MF : informations sur le ZIP
  - Répertoire **WEB-INF/**
    - /classes/ contient des servlets et des classes associées
    - /lib/ contient des fichiers .jar additionnels (JDBC, TagLib etc)
    - /tlds/ contient des fichiers .tld décrivant les TagLibs
    - web.xml est un descripteur de déploiement



DESCRIPTEUR DE DÉPLOIEMENT WEB.XML

- Une application peut contenir plusieurs servlets

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi::schemaLocation="http://java.sun.com/xml/ns/javaee
                         http://java.sun.com/xml/bs/javaee/web-app_2_5.xsd">
    <servlet>
        <!-- Le nom de la servlet-->
        <servlet-name>HelloWorld</servlet-name>
        <!-- Le nom complet de la classe -->
        <servlet-class>exemple.HelloWorld</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloWorld</servlet-name>
        <!-- L'URL d'appel de la servlet -->
        <url-pattern>/hello</url-pattern>
    </servlet-mapping>
</web-app>
```

Construit une relation entre une servlet et son URL

DESCRIPTEUR DE DÉPLOIEMENT WEB.XML

INITIALISATION DE SERVLET

- Une servlet peut avoir des paramètres d'initialisation :
  - Définis dans web.xml
  - Changés sans avoir besoin de recompiler l'application
  - Utilisés avec la méthode *getInitParameter()* de l'interface ServletConfig

```
 6     <servlet>
 7         <!-- Le nom de la servlet-->
 8         <servlet-name>HelloWorld</servlet-name>
 9         <!-- Le nom complet de la classe -->
10         <servlet-class>exemple.HelloWorld</servlet-class>
11         <init-param>
12             <param-name>reponse</param-name>
13             <param-value>42</param-value>
14         </init-param>
15     </servlet>
```

CHEMIN DE REQUÊTE

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

https://localhost:8080/CoursWeb/url/request

```
1
2 // Dans une servlet
3 out.println("context path = " + req
4 out.println("servlet
5 out.
```

```
<servlet-mapping>
    <servlet-name>URLServlet</servlet-name>
    <!-- L'URL d'appel de la servlet -->
    <url-pattern>/url/*</url-pattern>
</servlet-mapping>
```

context path = /CoursWeb

Servlet path = /url

Path info = /request

GESTION D'ÉTAT

- HTTP gère uniquement les requêtes et réponses.
- Aucun historique n'est stocké.
- Requêtes indépendantes les unes des autres.
- Une application Web a besoin de savoir
  - Quel utilisateur a émis la requête
  - Les requêtes déjà émises par l'utilisateur
  - L'état général de l'application (nombre de visites, de produits vendus, etc)

  → Il faut ajouter un mécanisme de gestion d'état.

GESTION D'ÉTAT

- Différentes techniques sont utilisées pour gérer l'état :
  - Cookies
  - Sessions grâce à HttpSession
  - Réécriture d'URL (par la méthode encodeURL())
  - Champ de formulaire caché (type="hidden")

COOKIES

❖ Les cookies, qu'est-ce que c'est ?

- Fichier stocké par un serveur Web chez le client
  - Moyen de stocker /récupérer des informations sur le client à chaque requête
  - Éviter à l'utilisateur de retaper sans cesse ses informations
  - L'utilisateur peut interdire leur utilisation dans son navigateur
- Définis dans la classe javax.servlet.http.Cookie

COOKIES

❖ **Comment ça fonctionne ?**

- Créé en donnant un nom et une valeur (string).
  - Cookie uncookie = new Cookie("uniqueid", "42") ;
  - Placé via un objet **response :** response.addCookie(uncookie) ;
- Récupéré via un objet **request**.
  - Cookie[] cookies = request.getCookies() ;
- Méthodes getName() et getValue()

CRÉATION D'UN COOKIE : Exemple

```java
@WebServlet(name="user", urlPatterns={"*.do"})
public class UserServlet extends HttpServlet
{
    protected void doGet(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        try {
            // Création d'un cookie
            Cookie cookie = new Cookie("userid", "42");
            cookie.setMaxAge(60); // Validité du cookie en secondes
            res.addCookie(cookie);
            out.println("Cookie écrit.");
        } finally {
            out.close();
        }
    }
}
```

RÉCUPÉRATION D'UN COOKIE : Exemple

```
 5      protected void doGet(HttpServletRequest request, HttpServletResponse response)
 6          throws ServletException, IOException
 7      {
 8          response.setContentType("text/html;charset=UTF-8");
 9          PrintWriter out = response.getWriter();
10          Cookie[] cookies = request.getCookies();
11          for (int i = 0; i < cookies.length; i++) {
12              Cookie cookie = cookies[i];
13              out.println(cookie.getName() + ": " + cookie.getValue());
14          }
15          // -> Afficherait userid: 42
16      }
17  }
```

SESSION

- Ensemble de requêtes HTTP sur une période données.
- Permet de se souvenir des requêtes d'un utilisateur.
- Suivi de l'activité :
  - Un objet session associé à toutes les requêtes d'un utilisateur
  - Les sessions expirent au bout d'un délai fixé (après X secondes d'inactivité)

SESSION

- L'API HttpServlet permet de gérer les sessions.
- L'objet **request** (HttpServletRequest) maintient les informations de session.
- Accessible via les méthodes :
  - HttpSession session = request.getSession(boolean flag)
    - flag = true pour renvoyer une nouvelle session s'il n'en existe pas encore. False sinon.
  - IsRequestedSessionIdFromCookie() : Vrai si l'identifiant de session provient d'un cookie.
  - IsRequestSessionIdValid() : Vrai si l'identifiant de session est valide dans le contexte courant.
  - IsRequestedSessionIdFromURL() : Vrai si l'identifiant de session provient de l'URL

HTTP SESSION

- Autres méthodes :
  - **void setAttribute(String name, Object value) :** Ajoute un couple clef / valeur à la session
  - **Object getAttribute(String name) :** Retourne la valeur associée à la clef name
  - **String getId() :** Retourne l'identifiant de la session
  - **void removeAttribute(String name) :** Supprime l'attribut name
  - **Enumeration getAttributeNames() :** Retourne tous les noms d'attributs de la session
  - **void invalidate() :** Invalide la session
  - **getLastAccessedTime() :** Retourne l'horodatage (timestamp) du dernier accès à la session en millisecondes.

EXEMPLE

```
5     protected void doGet(HttpServletRequest request, HttpServletResponse response)
6         throws ServletException, IOException
7     {
8         PrintWriter out = response.getWriter();
9         HttpSession session = request.getSession();
10        // La session est-elle nouvelle ?
11        out.println("Nouvelle session ? " + session.isNew() + "<br />");
12        // L'identifiant de session (valeur du cookie JSESSIONID)
13        out.println("Session id : " + session.getId() + "<br />");
14        // On ajoute une valeur à la session
15        session.setAttribute("user", "Chuck");
16        // L'attribut user est dispo dès maintenant et pour les
17        // prochaines requêtes HTTP
18        out.println("Bienvenue " + session.getAttribute("user"));
19    }
```

CONFIGURATION DE SESSION

- Définition de la durée d'une session :
  - Dans le fichier **web.xml** (en minutes).

```
<session-config>
  <session-timeout>30
</session
```

  - Avec la méthode ***void* setMaxInactiveInterval *(int seconds) ;***
- Terminaison de session
  - Automatiquement après l'expiration du timeout
  - Manuellement avec la méthode ***void setMaxInactiveInterval(int seconds);***
  - Manuellement avec la méthode ***void invalidate() de l'objet HttpSession***

SERVLET CONTEXT

- Correspond au contexte de l'application Web
- Maintient des données pour toute l'application
- Données d'initialisation pour toute l'application
- Un ServletContext par application et par JVM
- Ensemble de couples (name, value) partagés par toutes les servlets instanciées
- Objet permettant au servlet de communiquer avec le conteneur de servlet

SERVLET CONTEXT

- Obtention de l'objet ServletContext
  - ✓ Méthode ServletContext getServletContext() héritée de HttpServlet
- Méthodes de l'objet ServletContext
  - ✓ Gestion des attributs , même principe que pour la session
    - ➢ void setAttribute(String key, Object value)
      déposer un attribut
    - ➢ Object getAttribute(String key)
      récupérer un attribut
  - ✓ Accès aux paramètres de configuration de l'application
    - ➢ String getInitParameter(String name)
    - ➢ Paramètres déclarés dans le fichier de déploiement (web.xml)
    - ➢

```
<context-param>
      <param-name>admin-mail</param-name>
      <param-value>Joe.Moose@yukon.org</param-value>
</context-param>
```

SERVLET CONTEXT

- **Exemple :**

```
<context-param>
  <param-name>contextParam1</param-name>
  <param-value>valeur 1</param-value>
</context-param>
 <context-param>
  <param-name>contextParam2</param-name>
  <param-value>valeur 2</param-value>
</context-param>
```

```
PrintWriter out = response.getWriter();
String param1 = getServletContext().getInitParameter("contextParam1");
String param2 = getServletContext().getInitParameter("contextParam2");
out.println("Contexte param 1 : " + param1 + "<br />");
out.println("Contexte param 2 : " + param2 + "<br />");
```

CONCLUSION

- Les servlets étendent le comportement des serveurs Web avec des programmes Java.
- Avantages :
  - Portabilité
  - Facilité d'écriture
  - Définition du code, du paquetage, du déploiement

CONCLUSION

- **Limites**
  - Difficile d'écrire du code HTML dans du code Java
  - Introduction de la technologie Java Server Pages (JSP)
  - Pas de mécanisme intégré de distribution
  - Introduction de la technologie Enterprise Java Beans (EJB)

Quiz !

1. **Qu'est-ce qu'une servlet en Java ?**

   a) Une application autonome
   b) Un composant côté serveur qui génère du contenu dynamique
   c) Un outil de débogage Java
   d) Un type de base de données

Quiz !

Quelle méthode d'une servlet est appelée lors de sa première initialisation ?

a) init()
b) service()
c) doGet()
d) destroy()

Quiz !

Quelle interface doit être implémentée pour créer une servlet ?

a) Servlet
b) HttpServlet
c) GenericServlet
d) Runnable

Quiz !

Quelle méthode est utilisée pour traiter les requêtes GET ?

a) doPost()
b) doGet()
c) service()
d) init()

Quiz !

Quel code HTTP est généralement renvoyé par un Servlet en cas de réussite d'une requête doPost() ?

a) 404
b) 200
c) 500
d) 302

Quiz !

Comment une servlet peut-elle récupérer un paramètre de requête nommé "user" ?

a) request.getParameter("user")
b) response.getParameter("user")
c) request.getAttribute("user")
d) response.getAttribute("user")

Quiz !

Quelle est la principale différence entre doGet() et doPost() ?

a) doGet() est utilisé pour les requêtes HTTP GET, doPost() pour les requêtes HTTP POST
b) doGet() est plus sécurisé que doPost()
c) doPost() est plus rapide que doGet()
d) Il n'y a aucune différence

Quiz !

Combien de fois un objet Servlet est-il instancié pendant la durée de vie d'une application ?

a) Une fois pour chaque requête
b) Une fois par session utilisateur
c) Une seule fois
d) Autant de fois que nécessaire

Quiz !

Où sont stockées les informations de session pour une servlet ?

a) Dans des variables locales
b) Dans l'objet HttpSession
c) Dans l'objet ServletContext
d) Uniquement dans des cookies

Quiz !

Écrivez un exemple de code pour une servlet qui affiche "Bonjour, [Nom]" où le nom est récupéré via un paramètre de requête HTTP !

MERCI