Servlets

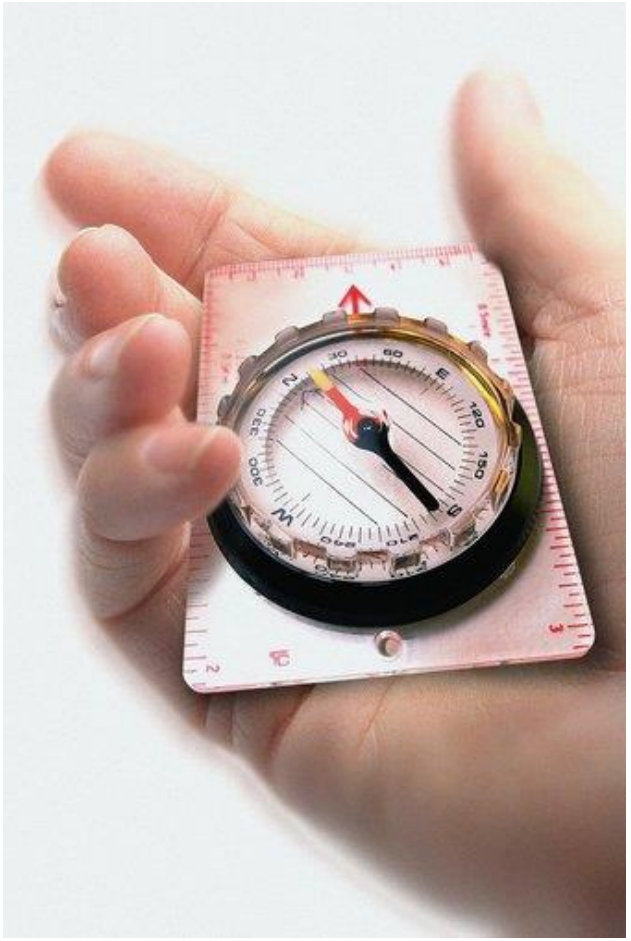
Java Web Applications

Objectifs du cours

En complétant ce cours, vous serez en mesure de:

- Expliquer ce qu'est une servlet
- Décrire l'API Servlet
- Créer des servlets de base et avancés

Plan de cours



- Introduction
- Hiérarchie des servlets
- Traitement des requêtes et des réponses
- Descripteur de déploiement
- Le modèle de conteneur Web

Servlets

OVERVIEW

Presentation

- Classes exécutées sur le serveur
- Traitement dynamique des requêtes
- Production de réponse dynamique
- Généralement utilisé sur un serveur Web

Avantages

- Efficace :
  - Gestion de la mémoire de la JVM
  - Une seule instance par servlet
  - Une requête = Un Thread
  - …
- Utile :
  - Cookies
  - Sessions
  - …

Avantages

- Extensible et flexible :
  - Exécuté sur de nombreuses plates-formes et serveurs HTTP sans changement
  - Amélioré par de nombreux frameworks
- Langage Java ! (puissant, fiable avec beaucoup d'API)

Désavantages

- Ne convient pas pour générer du code HTML et JavaScript
  - Mais JSP le fait (nous le verrons plus tard)
- Nécessite un environnement d'exécution Java sur le serveur
- Nécessite un "Servlet Container" spécial sur le serveur Web
- API de bas niveau
  - Mais de nombreux frameworks sont basés dessus

Processus dynamique

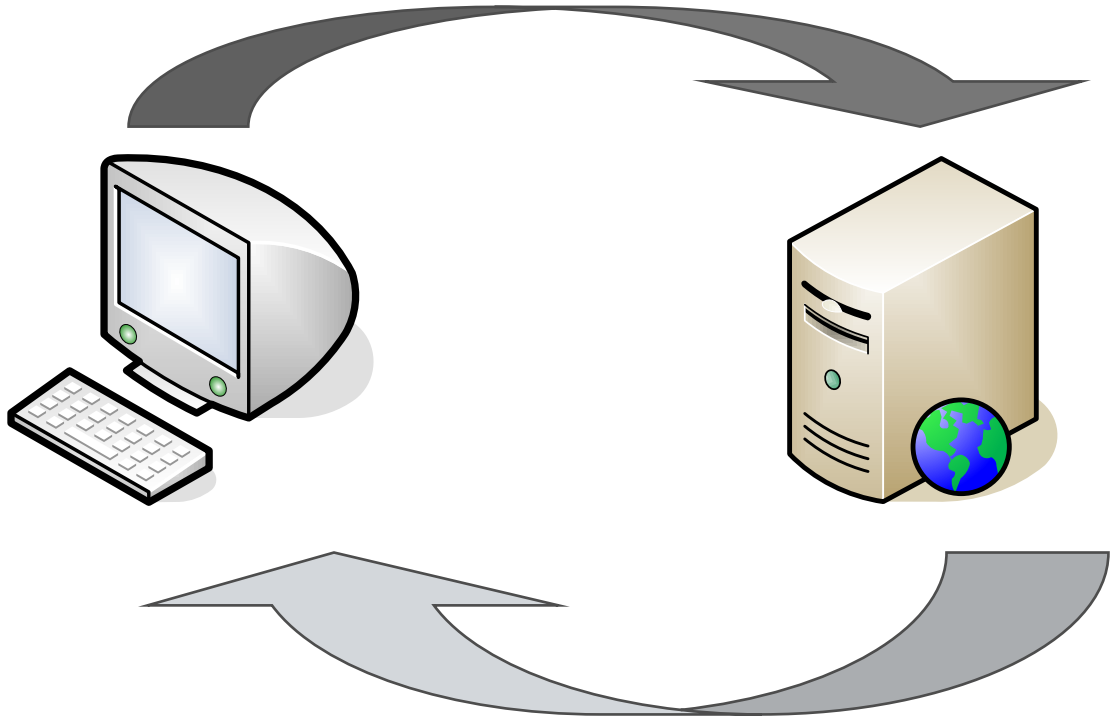
- Traitement basique des requêtes HTTP
  - HTML statique

≠

- Gestion des requêtes de servlet
  - Réponse dynamique générée

Requête basique

1. Connexion et demande du client

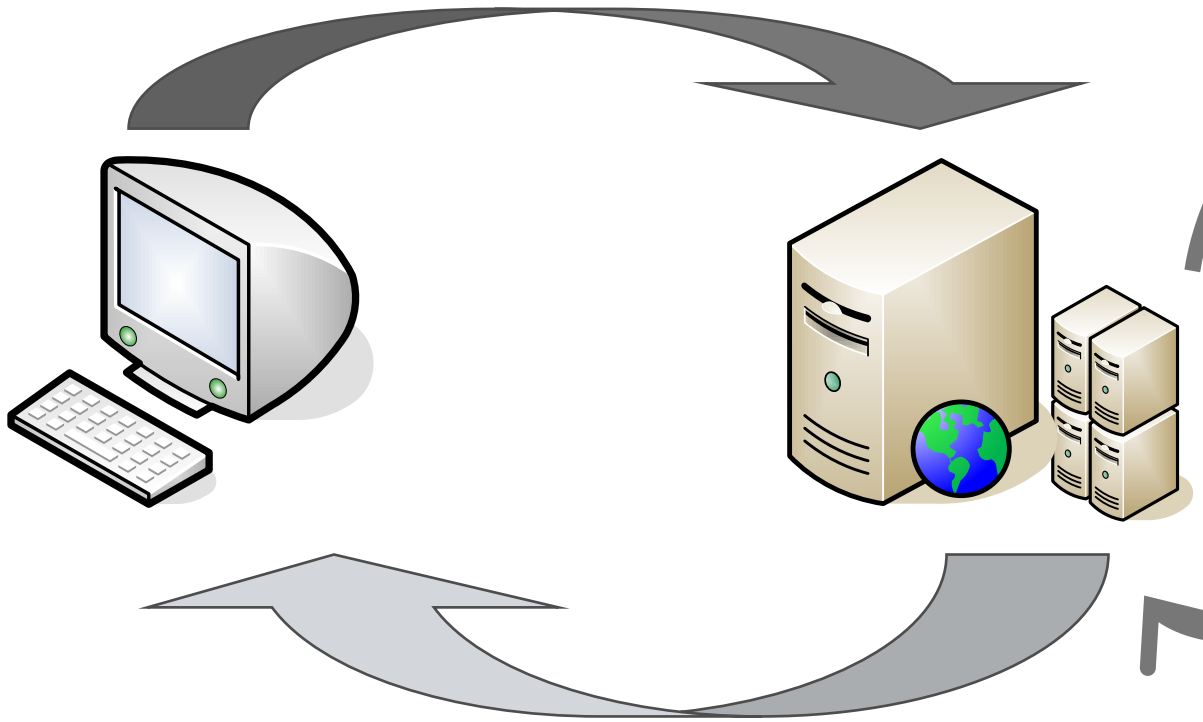


2. Cherche la cible

3. Page transférée au client puis déconnexion

Requête servlet – Premier appel

1. Connexion et demande du client



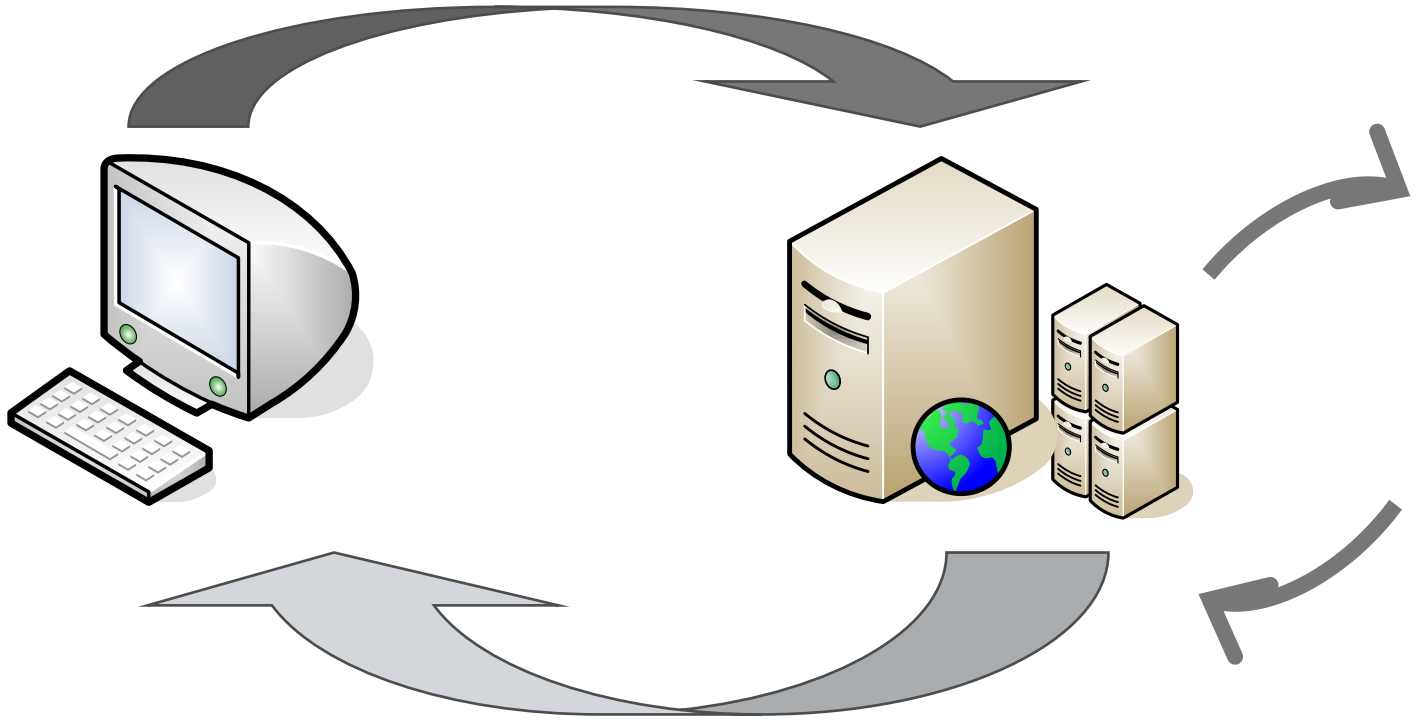
2. La servlet est créée et initialisée par la méthode init()

3. La servlet exécute la méthode service()

4. Réponse transférée puis déconnexion

Requête servlet – Autres appels

1. Connexion et demande du client



2. La servlet exécute la méthode service()

3. Réponse transférée puis déconnexion

Conteneur de servlet

- Le conteneur de servlet implémente le contrat de composant Web de l'architecture Java EE
- Également connu sous le nom de conteneur Web
- Des exemples de conteneurs Servlet sont :
  - Tomcat
  - Jetty
  - Oracle iPlanet Web Server

Frameworks basés sur les servlets

- Peu d'entreprises utilisent la technologie Servlet seule…
  - Mais beaucoup d'entreprises utilisent des frameworks basés sur des servlets !
  - Un grand nombre existe :
    - Struts
    - JSF
    - Spring MVC
    - Wicket
    - Grails

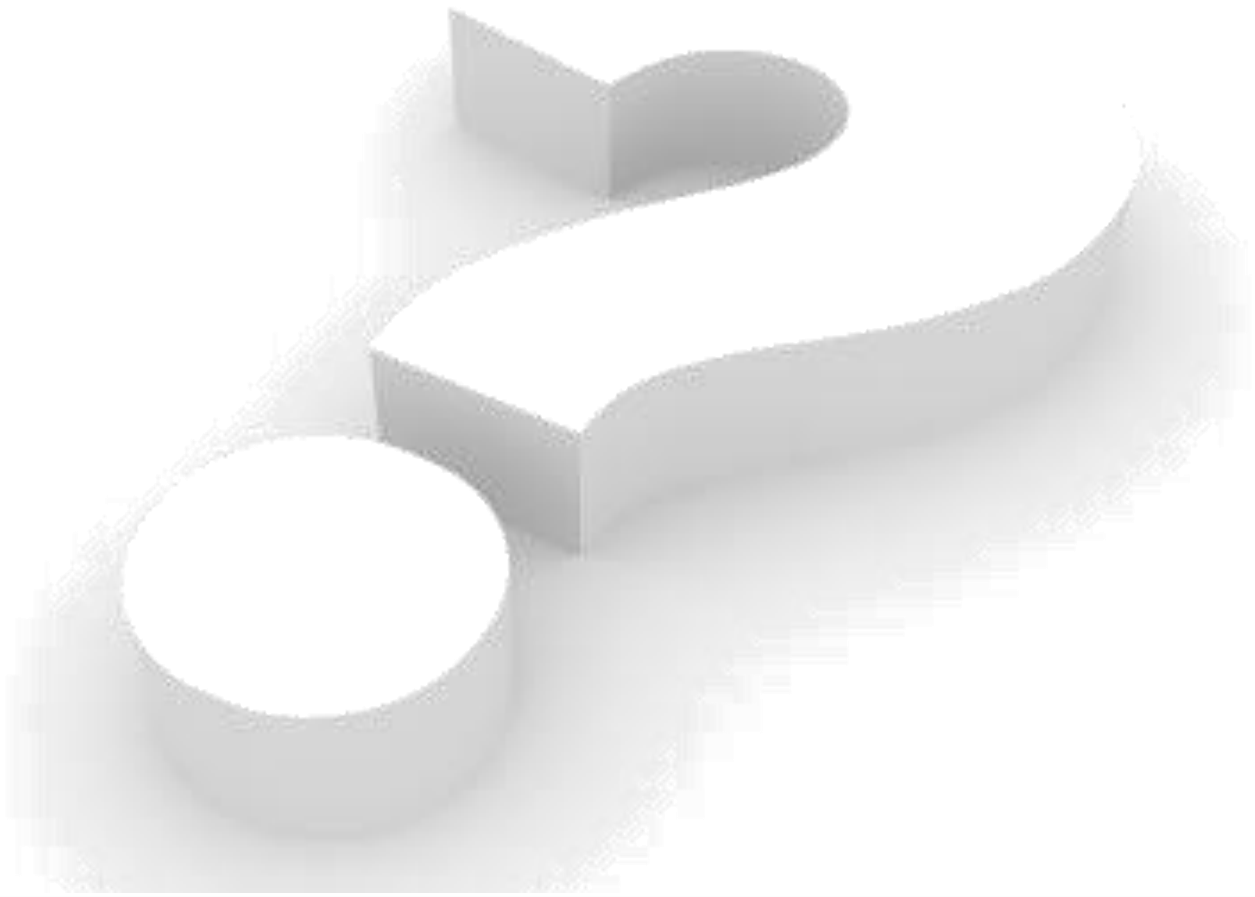
Struts

Spring

APACHE WICKET

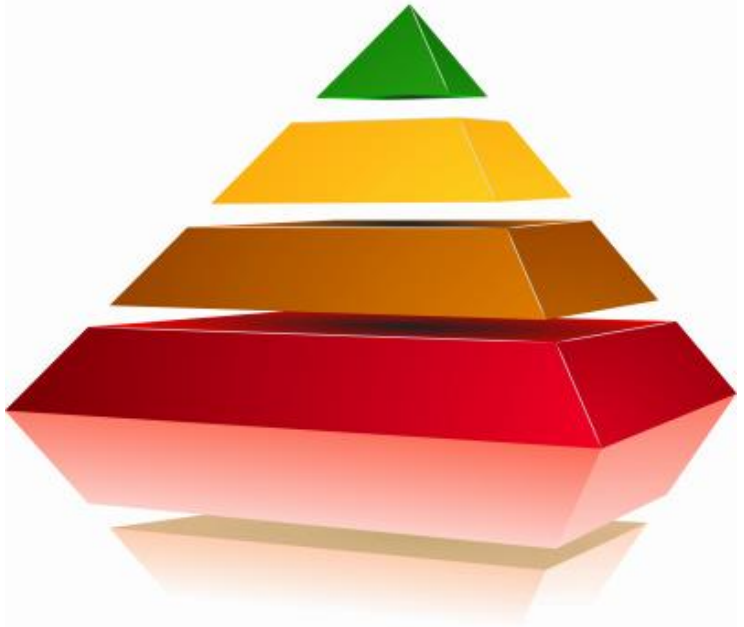
GRAILS

Questions ?



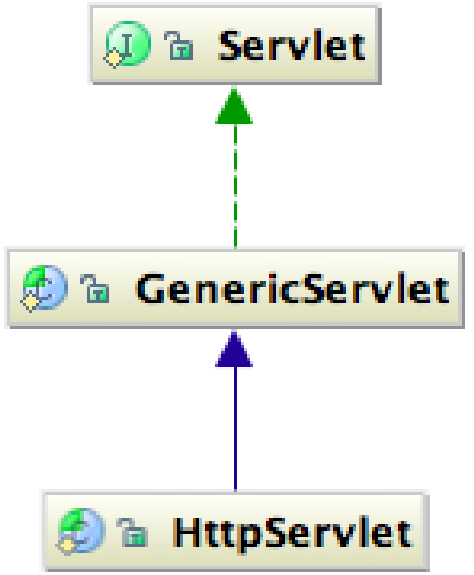
Servlets

HIÉRARCHIE DES SERVLETS



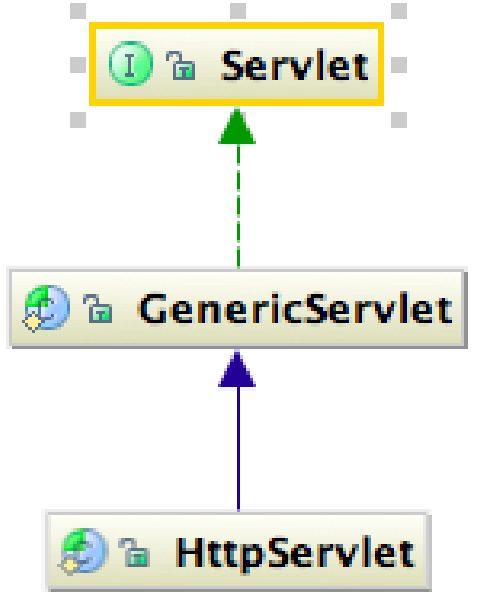
Présentation

- Hiérarchie des servlets est principalement composé de :
  - *Servlet* interface
  - *GenericServlet* abstract class
  - *HttpServlet* abstract class



Servlet interface

- Définit les méthodes qu'une servlet doit implémenter :
  - Comment la servlet est initialisée ?
  - Quels services offre-t-elle ?
  - Comment est-elle détruite ?
  - …



Servlet methods overview

Method	Description
void init(ServletConfig sc)	Appelé par le conteneur de servlet pour rendre le servlet actif
void service(ServletRequest req, ServletResponse res)	Appelé par le conteneur de servlets pour répondre à une requête
void destroy()	Appelé par le conteneur de servlet pour mettre le servlet hors service
ServletConfig getServletConfig()	Renvoie un objet ServletConfig, qui contient les paramètres d'initialisation et de démarrage de ce servlet
String getServletInfo()	Renvoie des informations sur le servlet, telles que l'auteur, la version et le copyright

Servlet Example 1/2

```
public class MyServlet implements Servlet {

   private ServletConfig config;

 @Override
 public void init(ServletConfig sc) {
     this.config = sc;
 }

 @Override
 public ServletConfig getServletConfig() {
     return this.config;
 }

 ...
```

Servlet Example 2/2

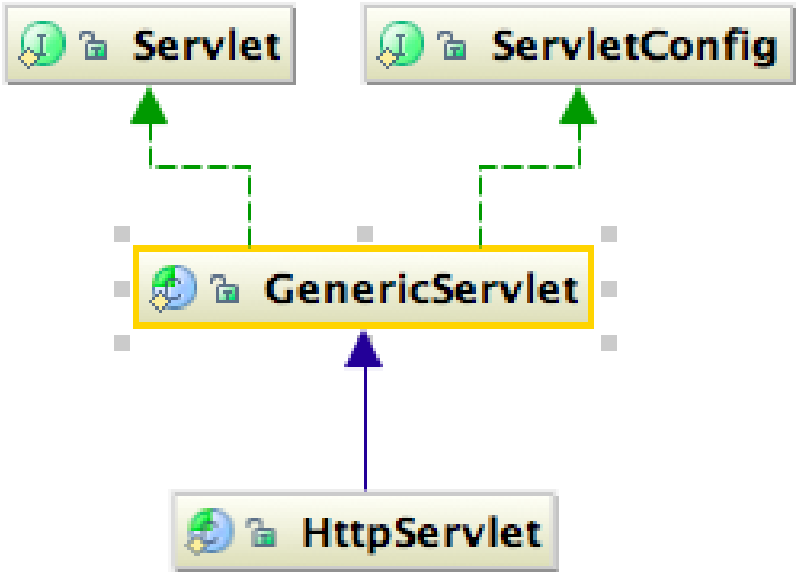
```
...
@Override
public String getServletInfo() {
    return "MyServlet is the best !!";
}

  @Override
  public void service(ServletRequest req,
                      ServletResponse res) {
    res.getWriter().println("Hello world");
  }

  @Override
  public void destroy() { /* ... */ }
}
```

GenericServlet class

- Une servlet à protocole indépendant
  - Implemente :
    - *Servlet*
    - *ServletConfig*



- Les implémentations ont juste besoin de définir la méthode de **service**

GenericServlet class exemple

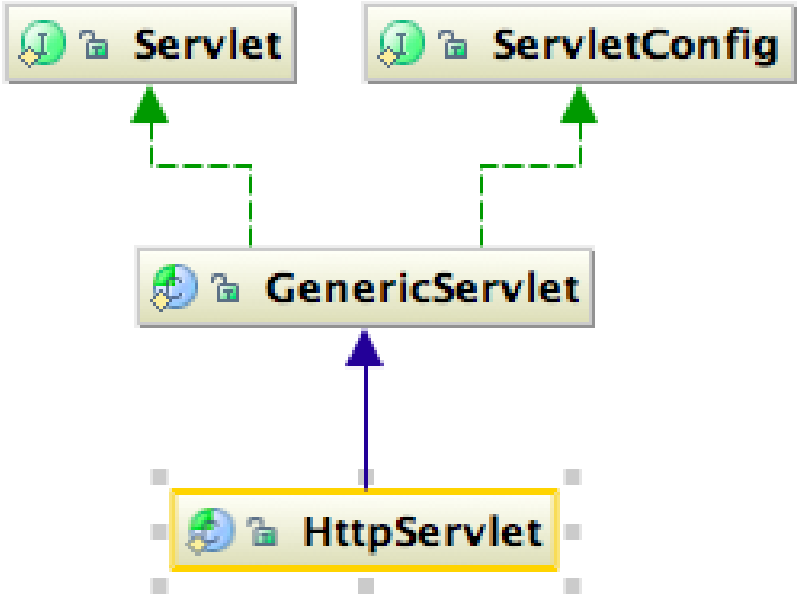
- Hiérarchie des servlets

```
public class MyServlet extends GenericServlet {

@Override
public void service(ServletRequest req,
                                ServletResponse res) {
    // Do whatever you want
}
}
```

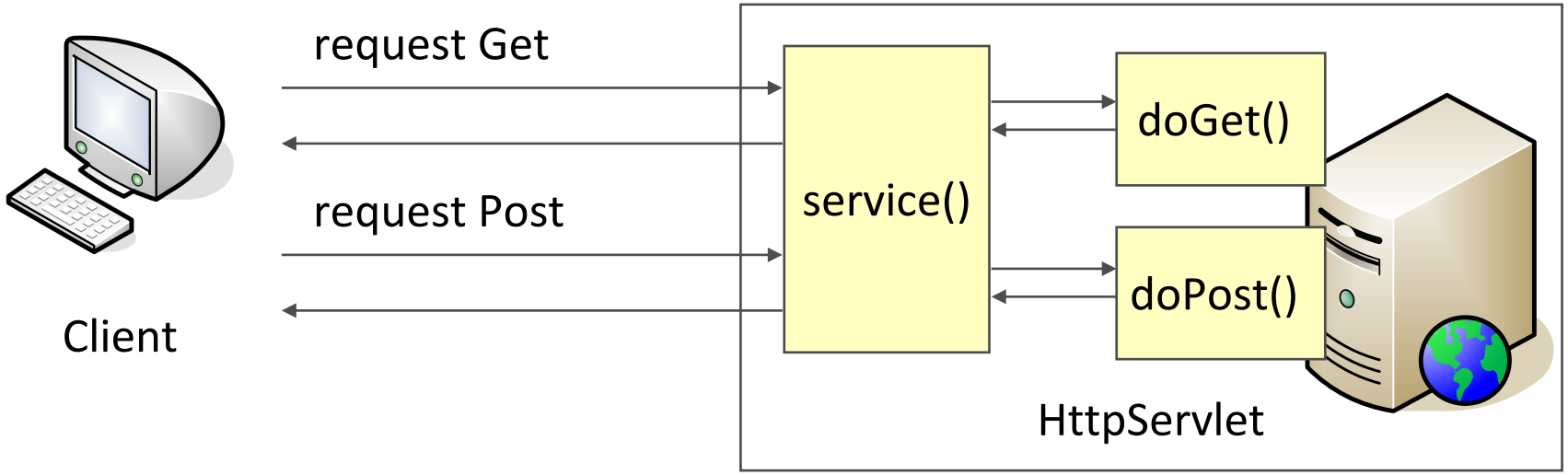
HttpServlet class

- Une Servlet adaptée aux sites Web
- Gère les méthodes HTTP :
  - Requêtes GET avec doGet(…)
  - Requêtes POST avec doPost(…)
  - Requêtes PUT avec doPut(…)
  - Etc…
- Vous pouvez en remplacer un ou plusieurs



HttpServlet class

- Comment ça marche?



- **Attention** : doGet(), doPost() et autres, sont appelés par l'implémentation HttpServlet de la méthode service(). **Si elle est remplacée, les méthodes du gestionnaire HTTP ne seront pas appelées !**

HttpServlet

- Métodes :

Method	Description
void service(HttpServletRequest req, HttpServletResponse res)	Reçoit les requêtes HTTP standard et les distribue aux méthodes doXXX définies dans cette classe
void doGet(HttpServletRequest req, HttpServletResponse res)	Gère les requêtes GET
void doPost(HttpServletRequest req, HttpServletResponse res)	Gère les requêtes POST
…	…

HttpServlet example

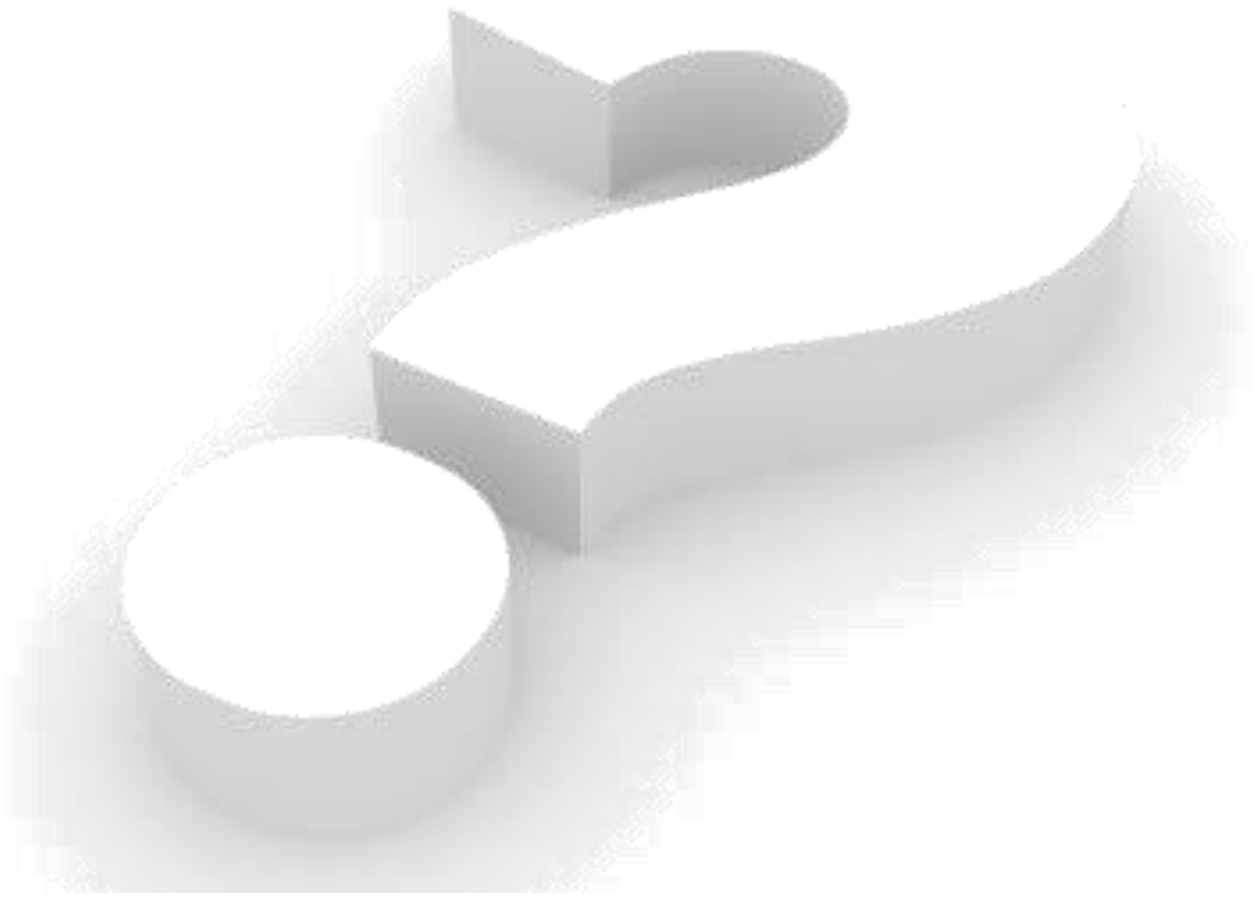
```
public class MyServlet extends HttpServlet {

   @Override
 public void doGet(HttpServletRequest req,
                              HttpServletResponse res) {
    // Do whatever you want
 }

 @Override
 public void doPost(HttpServletRequest req,
                              HttpServletResponse res) {
    // Do whatever you want here too ...
 }

}
```

Questions ?



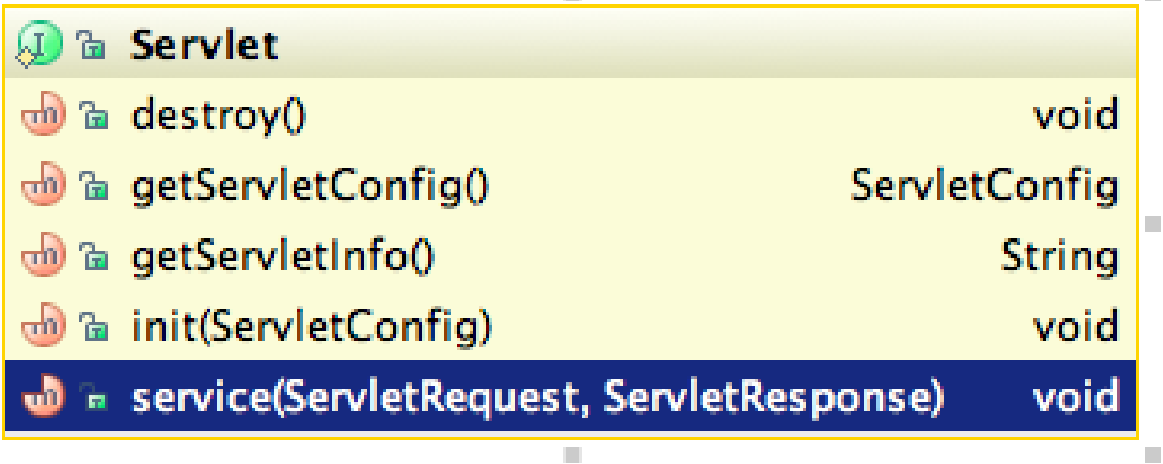
Servlets

TRAITEMENT DES REQUÊTES ET DES RÉPONSES

Introduction

- La méthode Servlet service() possède :
  - Un *ServletRequest* représentant une requête
  - Une *ServletResponse* représentant une réponse

- Vous pouvez les utiliser !



ServletRequest

- Objets définis pour fournir des informations de demande client à un servlet

- Utile pour récupérer :
  - Paramètres
  - Les attributs
  - Nom et port du serveur
  - Protocole
  - …

ServletRequest

- Présentation des méthodes :

Method	Description
String getParameter(String name) **Map<String, String[]> getParameterMap()**	Renvoie le(s) paramètre(s) de la requête
Object getAttribute() **void setAttribute(String name, Object value)**	Récupère/stocke un attribut de/vers la requête
boolean isSecure()	Indique si la demande a été effectuée à l'aide d'un canal sécurisé tel que HTTPS

ServletRequest

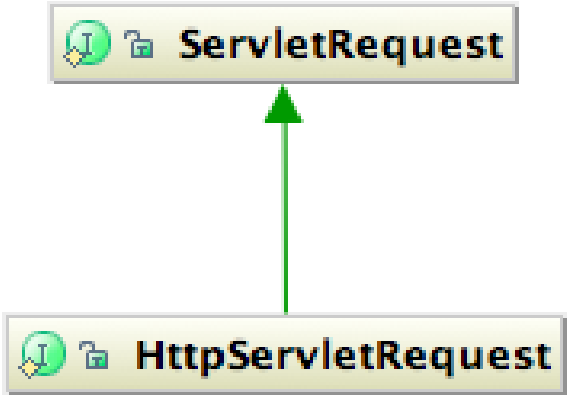
- Présentation des méthodes :

Method	Description
String getServerName() **int getServerPort()**	Obtenir le nom et le port du serveur
String getRemoteAddr() **String getRemoteHost()** **int getRemotePort()**	Obtenir l'adresse, le nom d'hôte et le port du client

HttpServletRequest

- Dédié aux requêtes HTTP

- Utile pour récupérer :
  - La session associée à la demande
  - Les en-têtes de la requête
  - Cookies
  - …



HttpServletRequest

- Présentation des méthodes :

Method	Description
HttpSession getSession() **HttpSession getSession(boolean create)**	Renvoie la session associée à la requête
String getHeader(String name)	Renvoie la valeur de l'en-tête spécifié
Cookie[] getCookies()	Obtient les cookies envoyés avec la demande

ServletResponse

- Objets définis pour aider une servlet à envoyer une réponse au client

- Utile pour récupérer :
  - Le flux de sortie de la réponse
  - Un PrintWriter
  - …

ServletResponse

- Présentation des méthodes :

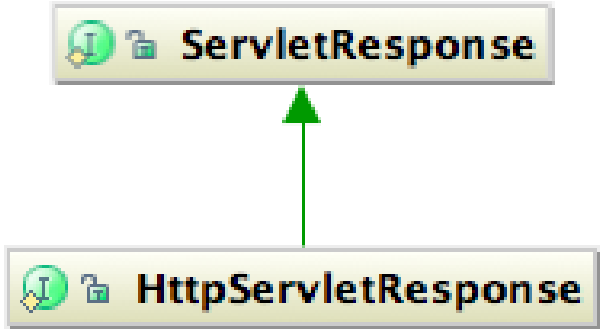
Method	Description
ServletOutputStream getOutputStream()	Renvoie le flux de sortie du servlet. Utile pour les données binaires
PrintWriter getWriter()	Renvoie un writer, utile pour envoyer une réponse textuelle au client
void setContentType(String content)	Définit le type mime du contenu de la réponse, comme "text/html"

Attention : Vous ne pouvez pas utiliser getOutputStream() et getWriter() ensemble.
Appeler les deux dans un servlet leve une **IllegalStateException**.

HttpServletResponse

- Dédié aux réponses HTTP

- Utile pour :
  - Ajouter des cookies
  - Rediriger le client
  - …



HttpServletResponse

- Présentation des méthodes :

Method	Description
void addCookie(Cookie c)	Ajoute un cookie à la réponse
void sendRedirect(String location)	Redirige le client vers l'emplacement spécifié

PrintWriter Example

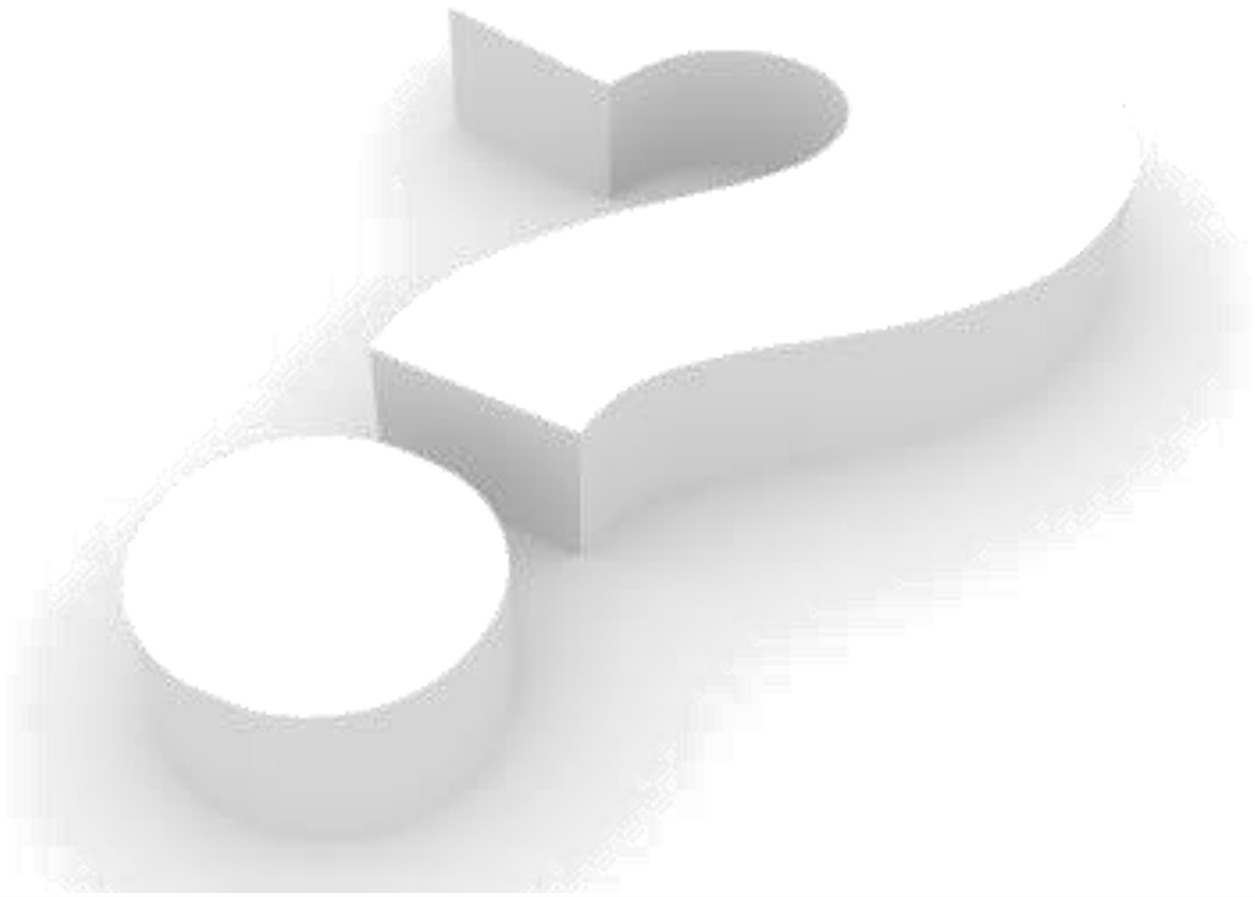
```
public class MyServlet extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req,
                         HttpServletResponse resp) {

      String name = req.getParameter("name");

      resp.setContentType("text/html");
      PrintWriter out = resp.getWriter();
      out.println("<html>");
      out.println("<head>");
      out.println("<title>My Servlet</title>");
      out.println("</head>");
      out.println("<body>");
      out.println("<h1>Hello " + name + " !</h1>");
      out.println("</body>");
      out.println("</html>");
    }
}
```

Questions ?



DESCRIPTEUR DE DÉPLOIEMENT

J'ai ma servlet, et maintenant ?

Introduction

- Une application Servlet a toujours (ou presque) un Descripteur de déploiement :
- /WEB-INF/web.xml
- Il définit :
  - Les classes de servlets
  - Les mappages de servlets
  - Les classes de filtres
  - Les mappages de filtres

- Les fiches d'accueil
- Les ressources applicatives
- Et quelques autres trucs…

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns=...>

   <display-name>MyApplication</display-name>

   <welcome-file-list>
      <welcome-file>index.html</welcome-file>
      <welcome-file>index.jsp</welcome-file>
   </welcome-file-list>

   <!-- Servlet declarations -->
   <!-- Servlet mappings -->
   <!- Other stuffs -->

</web-app>
```

Déclaration et mappage de servlet

Lorsque vous créez une servlet, pour l'utiliser, vous devez :

1. Le Déclarez dans un bloc servlet
   a. Donnez-lui un nom logique
   b. Donnez le "Fully Qualified Name" du servlet

1. Le mappez à un modèle d'URL dans un bloc de mappage de servlet

web.xml

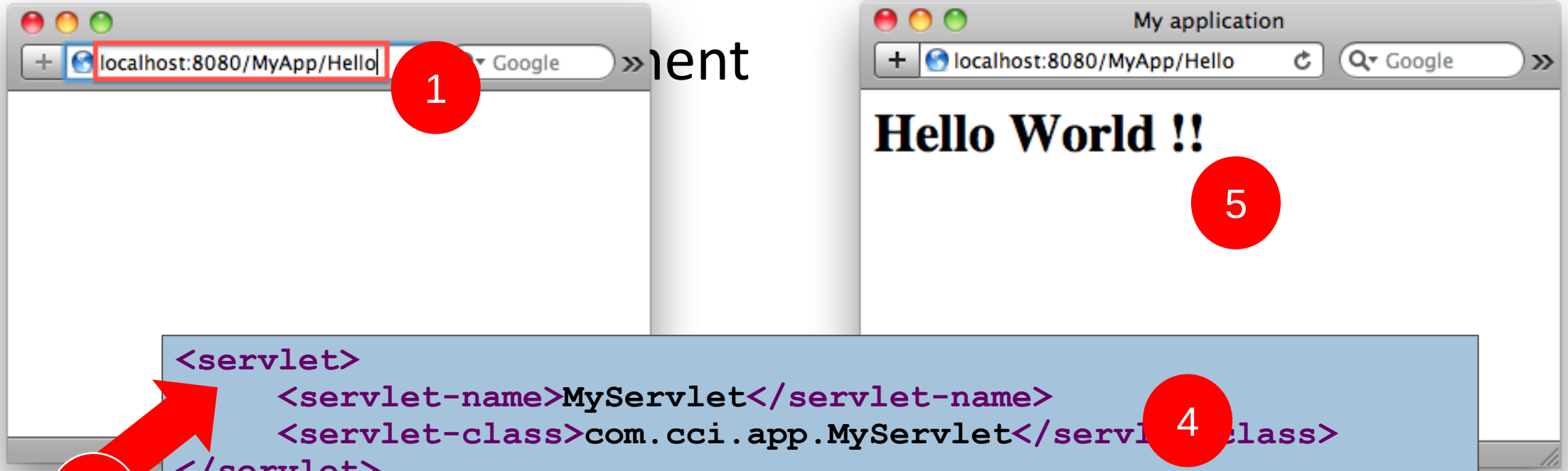
```
<servlet>
   <servlet-name>MyServlet</servlet-name>
   <servlet-class>com.cci.app.MyServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>MyServlet</servlet-name>
   <url-pattern>/Hello</url-pattern>
</servlet-mapping>
```

⇒ Le "servlet-name" dans les deux blocs "servlet" et "servlet-mapping" doit être le même !

How it works?

- ...ent



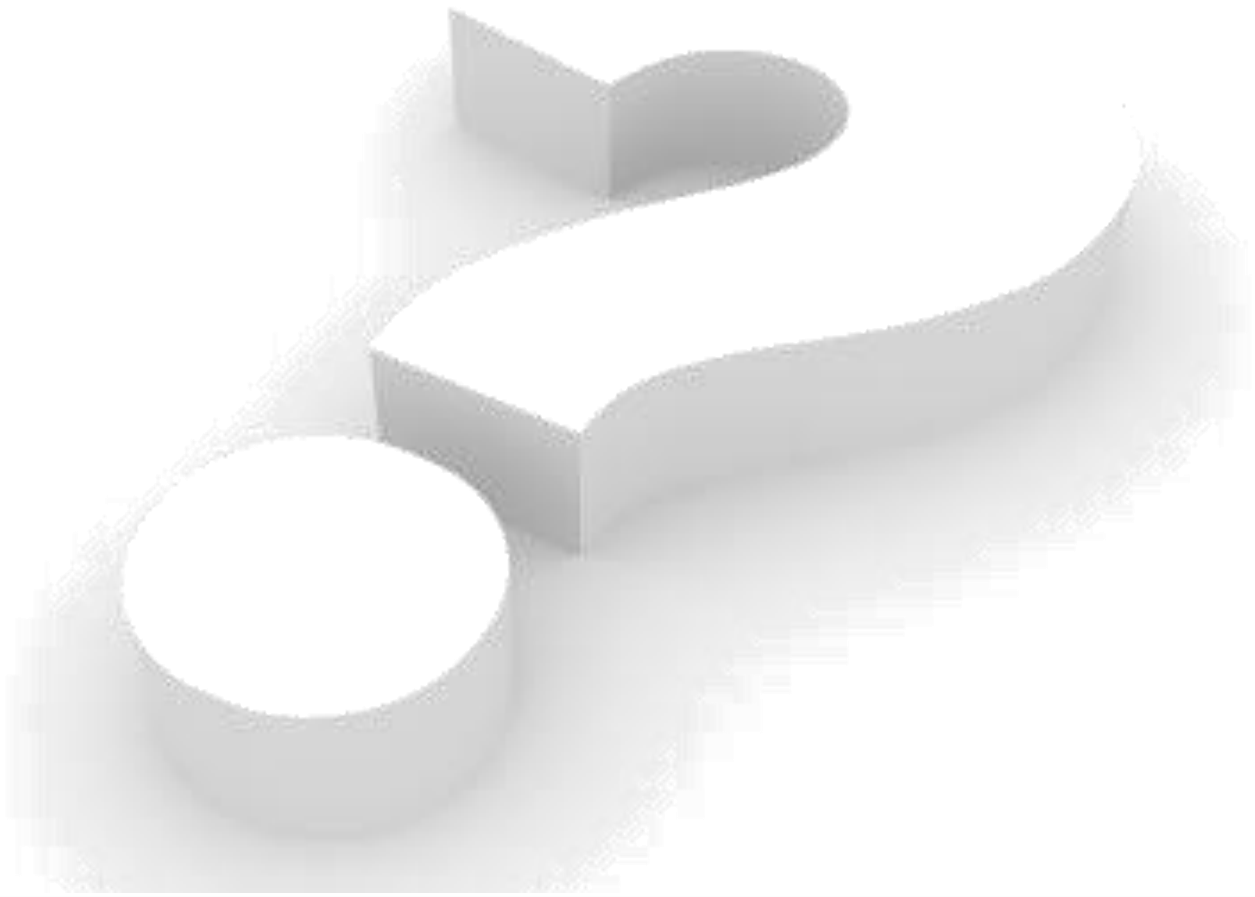

```
<servlet>
        <servlet-name>MyServlet</servlet-name>
        <servlet-class>com.cci.app.MyServlet</servl  lass>
</servlet>

<servlet-mapping>
        <servlet-name>MyServlet</servlet-name>
        <url-pattern>/Hello</url-pattern>
</servlet-mapping>
```

URL Patterns

Règle de correspondance	URL Pattern	Forme d'URL Pattern	URL qui correspondraient
Exact	/something	Toute chaîne avec "/" en debut	/something
Path	/something/*	Chaîne commençant par "/" et se terminant par "/*"	/something/ /something/else /something/index.htm
Extension	*.jsp	Chaîne se terminant par "*.jsp"	/index.jsp /something/index.jsp
Défaut	/	Seulement "/"	N'importe quelle URL sans meilleure correspondance

Questions ?



Exercices (1/3)

- Créez un package nommé :
  - **com.cci.supcommerce.servlet**
- Créer un HttpServlet à l'intérieur
  - Nommez-le **InsertSomeProductServlet**
  - Liez-le à /**basicInsert** (url-pattern)
  - Remplacer la méthode service(…)
    - Créer un nouvel objet **SupProduct** à l'intérieur
    - Définir les attributs de nom, de contenu et de prix de l'objet
    - **Utilisez** la méthode **SupProductDAO.addProduct(SupProduct)** pour le stocker en mémoire

Exercices (2/3)

- Créer un autre HttpServlet
  - Nommez-le **ListProductServlet**
  - Liez-le à /**listProduct** (url-pattern)
- Remplacer la méthode service(…)
  - Utiliser la méthode **SupProductDAO.getAllProducts()**
    - Pour récupérer tous les objets SupProduct ajoutés en mémoire
  - Utilisez un **PrintWriter** obtenu à partir de la **ServletResponse** pour écrire la réponse
- Testez vos deux nouvelles Servlets

Exercices (3/3)

- Créer un autre HttpServlet
  - Nommez-le **ShowProductServlet**
  - Liez-le à /**showProduct** url-pattern
- Remplacer la méthode doGet(…)
  - Rédiger une réponse complète pour le client contenant des informations détaillées sur un produit
    - Où **id** sera passé en tant que paramètre de requête
- Testez-le en saisissant ce type d'URL dans votre navigateur : http://localhost:8080/SupCommerce/showProduct?id=12

Servlets

LE MODÈLE DE CONTENEUR WEB

ServletContext, scopes, filters…

Scopes

- Les applications Web Java ont 3 portées différentes
  - Requête représentée par **ServletRequest**
  - Session représentée par **HttpSession**
  - Application représentée par **ServletContext**

- On a déjà vu **ServletRequest**, on va découvrir les deux autres…

Scopes

- Chacun d'eux peut gérer des attributs
  - Object getAttribute(String name)
  - void setAttribute(String name, Object value)
  - void removeAttribute(String name)

- Ces fonctions sont utiles pour transmettre certaines valeurs le long de la navigation de l'utilisateur, comme le nom d'utilisateur ou le nombre de pages visitées un jour spécifique

ServletContext

- Récupérez-le avec un ServletConfig
  - Ou un HttpServlet
    - (qui implémente un ServletConfig)
- Permet la communication entre les servlets et le conteneur de servlets
  - Une seule instance par application Web par JVM
- Contient des données utiles comme le chemin de contexte, attributs de portée d'application, …

ServletContext

- Ajouter un attribut au **ServletContext**
  - Cet attribut sera accessible dans toute l'application

```
// ...
getServletContext().setAttribute("nbOfConnection”,0);
// ...
```

ServletContext

- Récupérer un attribut du **ServletContext** et le mettre à jour :

```
// ...
ServletContext sc = getServletContext();
Integer nbOfConnection =
        (Integer) sc.getAttribute("nbOfConnection");
sc.setAttribute("nbOfConnection", ++nbOfConnection);
// ...
```

ServletContext

- Supprimer un attribut du ServletContext :
  - Essayer de récupérer une valeur inconnue ou non définie renvoie null

```
// ...
getServletContext().removeAttribute("nbOfConnection");
// ...
```

Session

- Interface : HttpSession
- Récupérez-le avec un HttpServletRequest
  - *request.getSession();*

- Méthodes utiles :
  - *setAttribute(String name, Object value)*
  - *getAttribute(String name)*
  - *removeAttribute(String name)*

Session

- Récupérer une HttpSession et lui ajouter un attribut :
  - Cet attribut sera accessible dans toute la session utilisateur

```
// ...
HttpSession session = request.getSession();
Car myCar = new Car();
session.setAttribute("car", myCar);
// ...
```

Session

- Récupérer un attribut :

```
// ...
Car aCar = (Car) session.getAttribute("car");
// ...
```

Session

- Supprimer un attribut :
  - Essayer de récupérer une valeur inconnue ou non définie renvoie null

```
// ...
session.removeAttribute("car");
// ...
```

Chaining

- Depuis une servlet vous pouvez :
  - inclure le contenu d'une autre ressource dans la réponse
  - transmettre une requête de la servlet à une autre ressource

Chaining

- Vous devez utiliser un **RequestDispatcher**
  - Obtenu avec la demande

```
// Get the dispatcher
RequestDispatcher rd =
    request.getRequestDispatcher("/somewhereElse");
```

Chaining

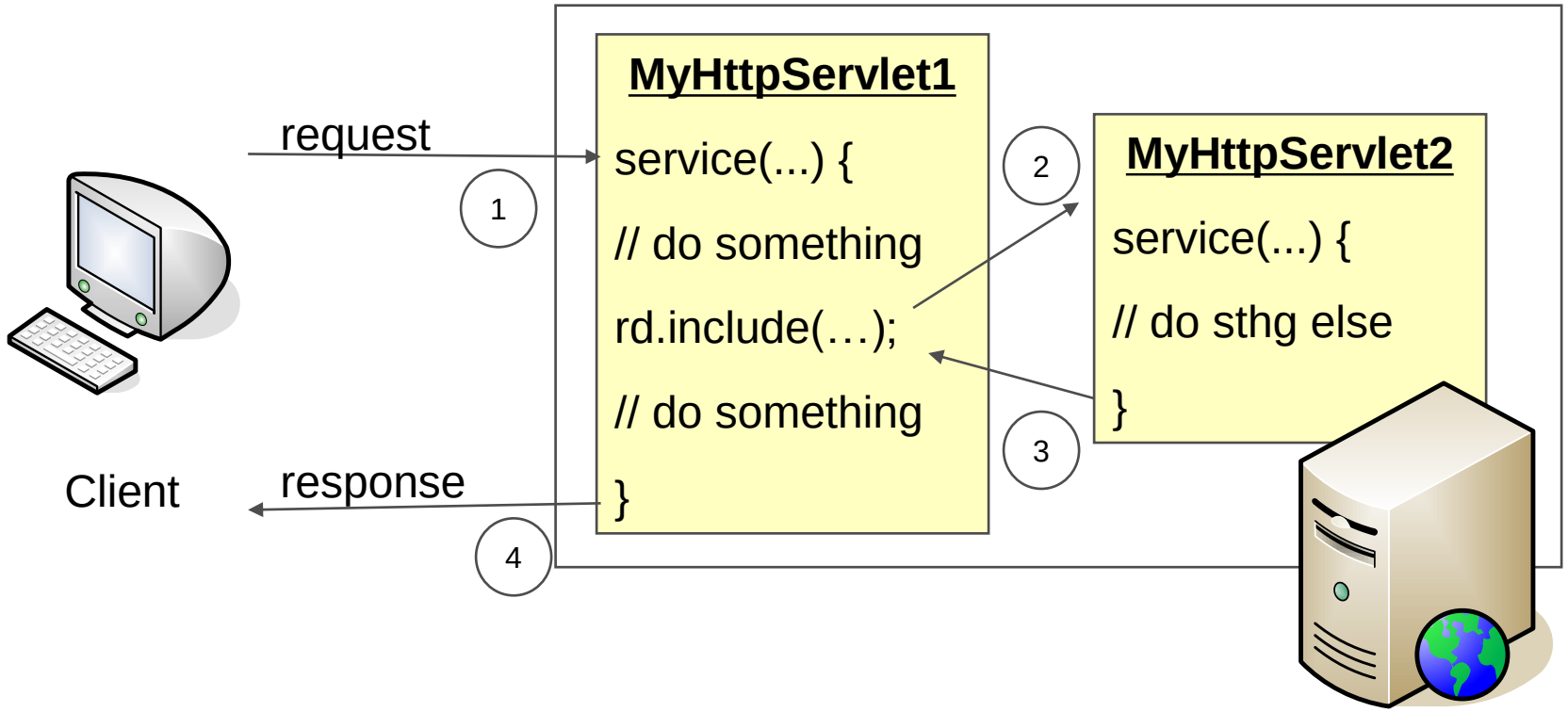
- Ensuite, vous pouvez inclure un autre servlet :

```
rd.include(request, response);
out.println("This print statement will be executed");
```

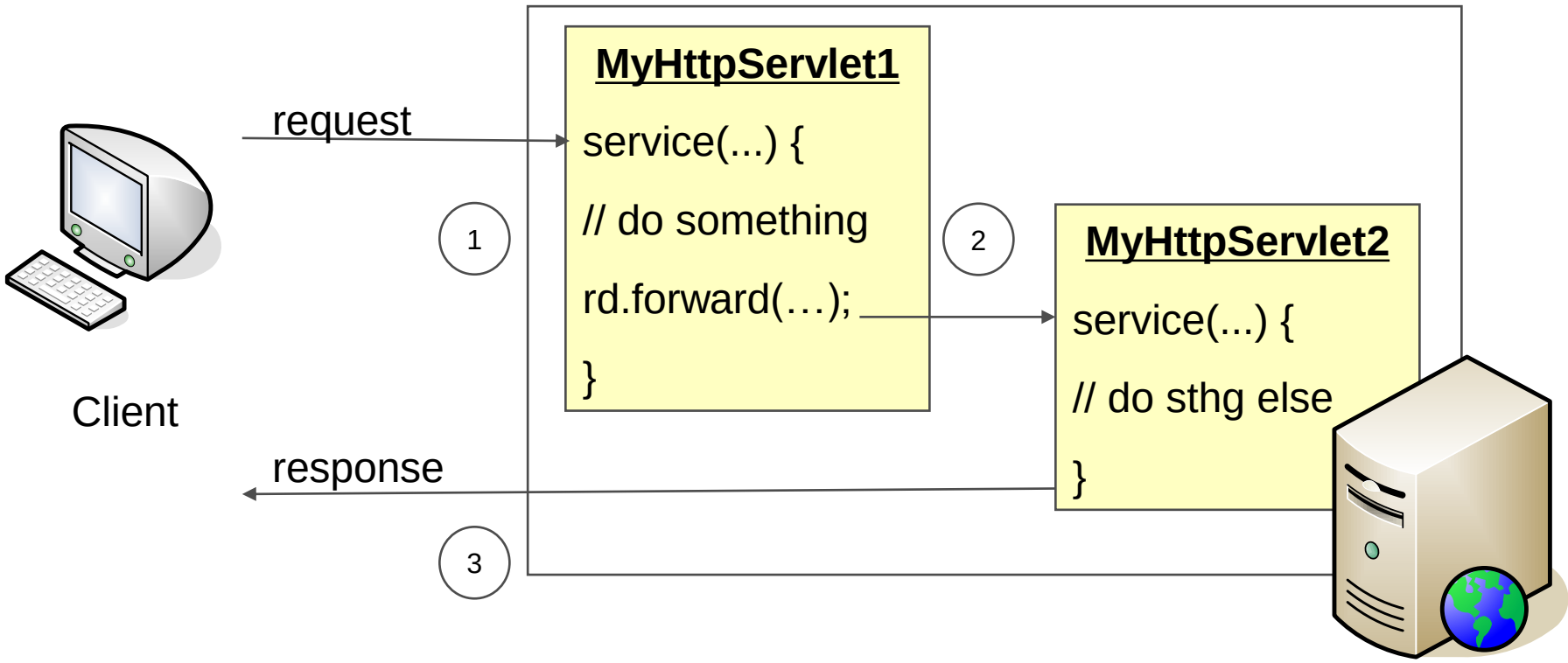
- Ou transférer la requête à un autre servlet :

```
rd.forward(request, response);
out.println("This print statement will not be executed");
```

Chaining – Include



Chaining – Forward



Chaining

- Utiliser les attributs de requête pour transmettre des informations entre les servlets
  - Servlet 1:

```
// Other stuff
Car aCar = new Car("Renault", "Clio");
request.setAttribute("car", aCar);
RequestDispatcher rd =
    request.getRequestDispatcher("/Servlet2");
rd.forward(request, response);
```

Chaining

- Récupérez l'attribut dans un autre servlet :
  - Servlet 2 mappée à **/Servlet2**

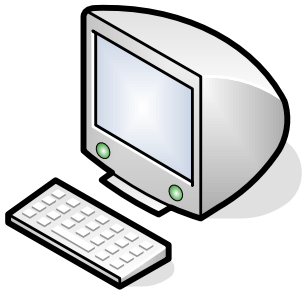
```
// Other stuff
Car aCar = (Car) request.getAttribute("car");
out.println("My car is a " + car.getBrand());
// Outputs "Renault"
```

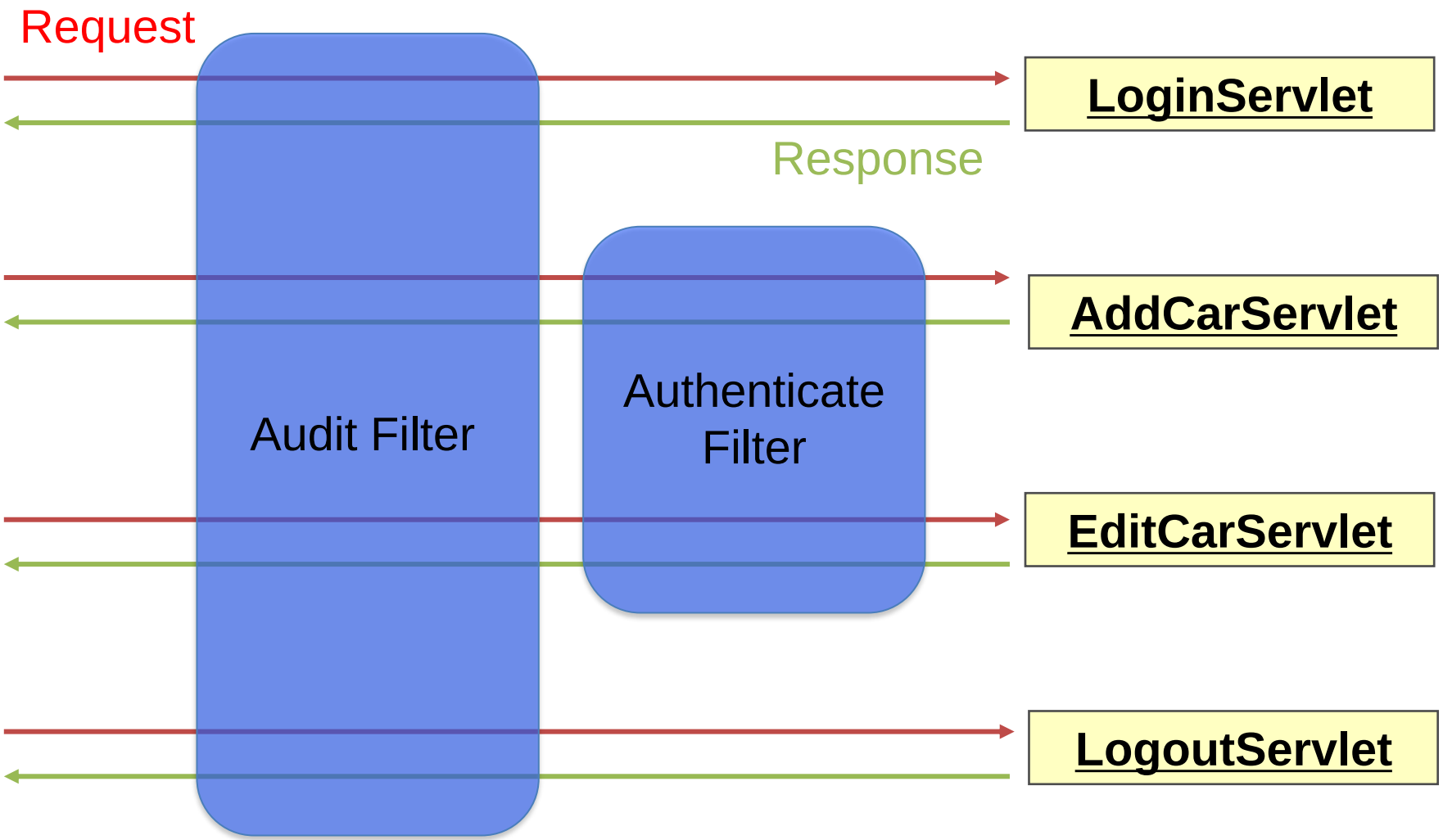
Filter

- Composant qui intercepte dynamiquement les requêtes et les réponses pour les utiliser ou les transformer
- Encapsuler les tâches récurrentes dans des unités réutilisables

- Exemple de fonctions que les filtres peuvent avoir :
  - Authentification - Blocage des demandes en fonction de l'identité de l'utilisateur
  - Journalisation et audit - Suivi des utilisateurs d'une application Web.
  - Compression des données - Rend les téléchargements moins long.

Filter




Request
Response
Audit Filter
Authenticate Filter
LoginServlet
AddCarServlet
EditCarServlet
LogoutServlet

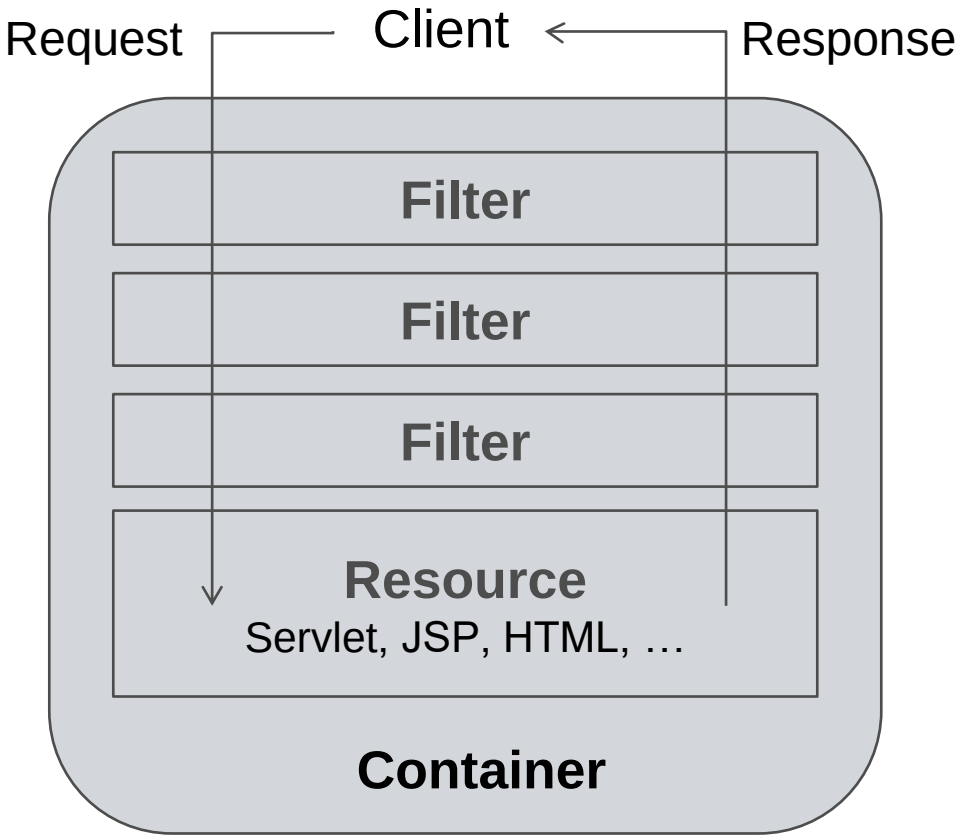
Filter

- Créer une classe implémentant l'interface **javax.servlet.Filter**
- Définir les méthodes :
  - init(FilterConfig)
  - doFilter(ServletResquest, ServletResponse, FilterChain)
  - destroy()

Filter

- Dans la méthode doFilter
  - Utilisez le **doFilter**(…) du **FilterChain** pour appeler l'élément suivant dans la chaîne
  - Les instructions avant cette instruction seront exécutées avant l'élément suivant
  - Les instructions après seront exécutées après l'élément suivant
  - N'appelez pas la méthode doFilter(…) de la FilterChain pour casser la chaîne

Filter


Request
Client
Response
Filter
Filter
Filter
Resource
Servlet, JSP, HTML, …
Container

Filter exemple

```
public final class HitCounterFilter implements Filter {
    //...
    public void doFilter(ServletRequest request,
          ServletResponse response, FilterChain chain) {

       ServletContext sc = filterConfig.getServletContext();
       Counter counter =
            (Counter) sc.getAttribute("hitCounter");

      System.out.println("The number of hits is: "
                                  + counter.incCounter());
       chain.doFilter(request, response);
    }
    //...
}
```

Filter declaration

- Déclarez votre filtre dans le fichier web.xml :

```
<filter>
    <filter-name>MyHitCounterFilter</filter-name>
    <filter-class>
        com.cci.sun.filters.HitCounterFilter
    </filter-class>
</filter>

<filter-mapping>
    <filter-name>MyHitCounterFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

Cookies

- Peut être placé dans un HttpServletResponse
- Peut être récupéré dans un HttpServletRequest
- Constructeur :
  - Cookie(String name, String value)
- Méthodes :
  - String getName()
  - String getValue()

Cookies exemple

- Ajouter un cookie à une réponse :

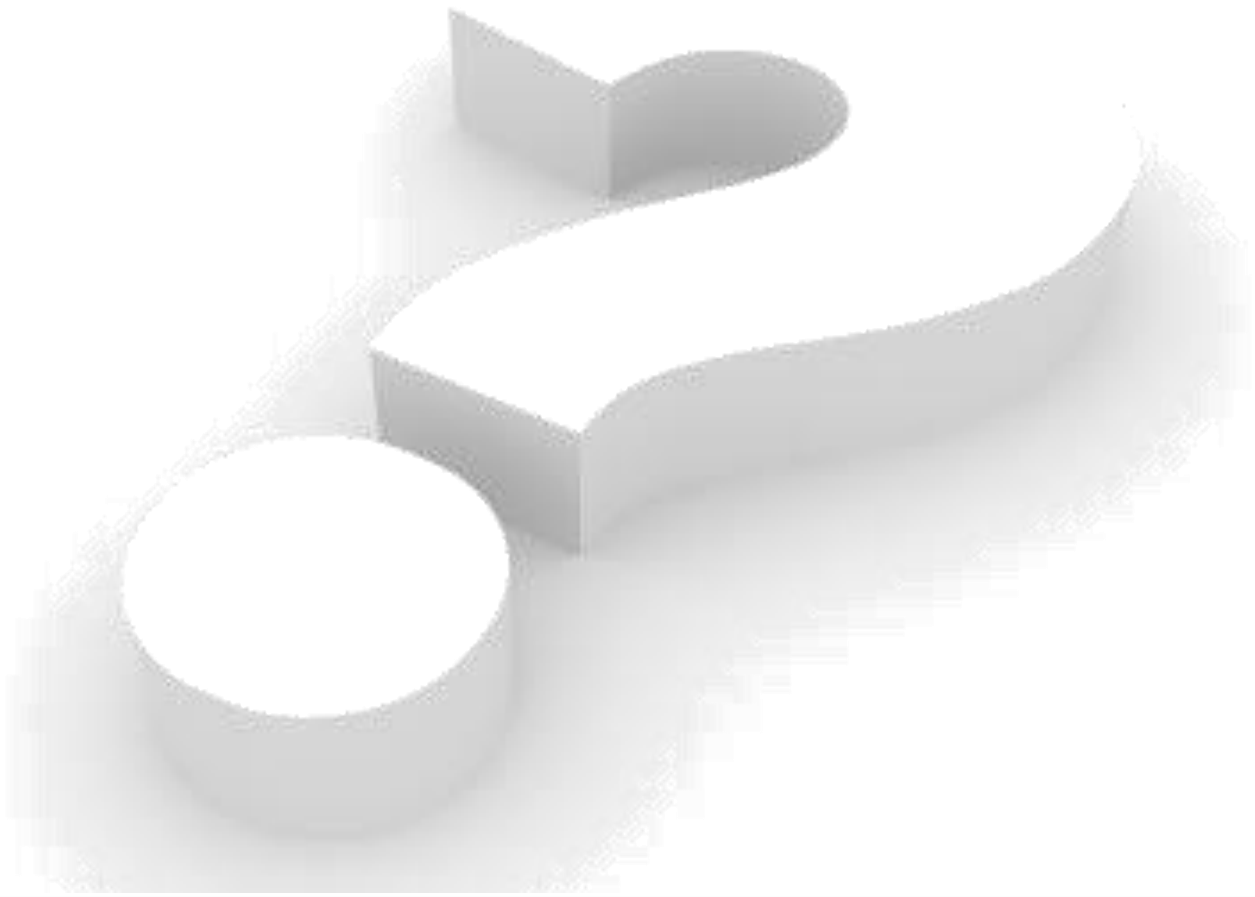
```
// ...
Cookie myCookie = new Cookie("MySuperCookie",
                             "This is my cookie :)");
response.addCookie(myCookie);
// ...
```

Cookies exemple

- Récupérer les cookies de la requête :

```
// ...
Cookie[] list = request.getCookies();
for(Cookie c : list) {
System.out.println("Name: " + c.getName()
                   + " Value: " + c.getValue());
}
// ...
```

Questions ?



Exercices (1/3)

- Créer un nouveau HttpServlet
  - Nommez-le **LoginServlet**
  - Liez-le à **/login** (url-pattern)
- Remplacer la méthode doPost(…)
  - Récupérer le nom d'utilisateur passé dans la requête
  - Ajoutez-le comme attribut de session
  - Transférer la requête au servlet de listing

Exercices (2/3)

- Créer une nouvelle classe de filtre
  - Nommé **AuthenticateFilter**
  - Lier à **/auth/*** (url-pattern)
- Dans la méthode doFilter(…)
  - Vérifiez si l'attribut de session de nom d'utilisateur existe
  - Si c'est le cas, appelez simplement l'élément suivant de la chaîne
  - Sinon, redirigez l'utilisateur vers le formulaire de connexion

Exercices (3/3)

- Créez un fichier nommé **login.html** dans le dossier WebContent
  - Définissez un formulaire à l'intérieur avec un champ de texte nommé username
  - Définissez le modèle d'URL du servlet de connexion en tant qu'action dans le formulaire html
- Ajoutez **/auth** au début du modèle d'url de
  - **InsertSomeProductServlet**
  - Testez que votre AuthenticateFilter et votre LoginServlet fonctionnent !

GAGNER DU TEMPS ?

Annotations, Web fragments, …

Gagner du temps ?

Configuration par annotations

- Les annotations sont de plus en plus utilisées dans le développement Java :
  - Pour mapper des classes avec des tables dans la base de données (JPA)
  - Pour définir des règles de validation (Bean Validation)
  - Et aussi pour définir les Servlets et les Filtres sans les déclarer dans le Descripteur de déploiement !

Gagner du temps ?

Principales annotations

- **@WebServlet** : Marquer une classe comme Servlet (doit toujours étendre HttpServlet)

- Certains attributs de cette annotation sont :
  - **name** (optionnel) :
    - Le nom de la servlet
  - **urlPatterns** :
    - Les modèles d'URL auxquels s'applique le servlet

Gagner du temps ?

Principales annotations

- **@WebFilter** : marquer une classe comme filtre

- Certains attributs de cette annotation sont :
  - **filterName** (optionnel) :
    - Le nom du filtre
  - **servletNames** (optionnel si urlPatterns) :
    - Les noms des servlets auxquels s'applique le filtre
  - **urlPatterns** (optionnel si servletNames) :
    - Les modèles d'URL auxquels le filtre s'applique

Exemples

```
@WebServlet(urlPatterns="/myservlet")
public class SimpleServlet extends HttpServlet {
  ...
}
```

```
@WebFilter(urlPatterns={"/myfilter","/simplefilter"})
public class SimpleFilter implements Filter {
  ...
}
```

Gagner du temps ?

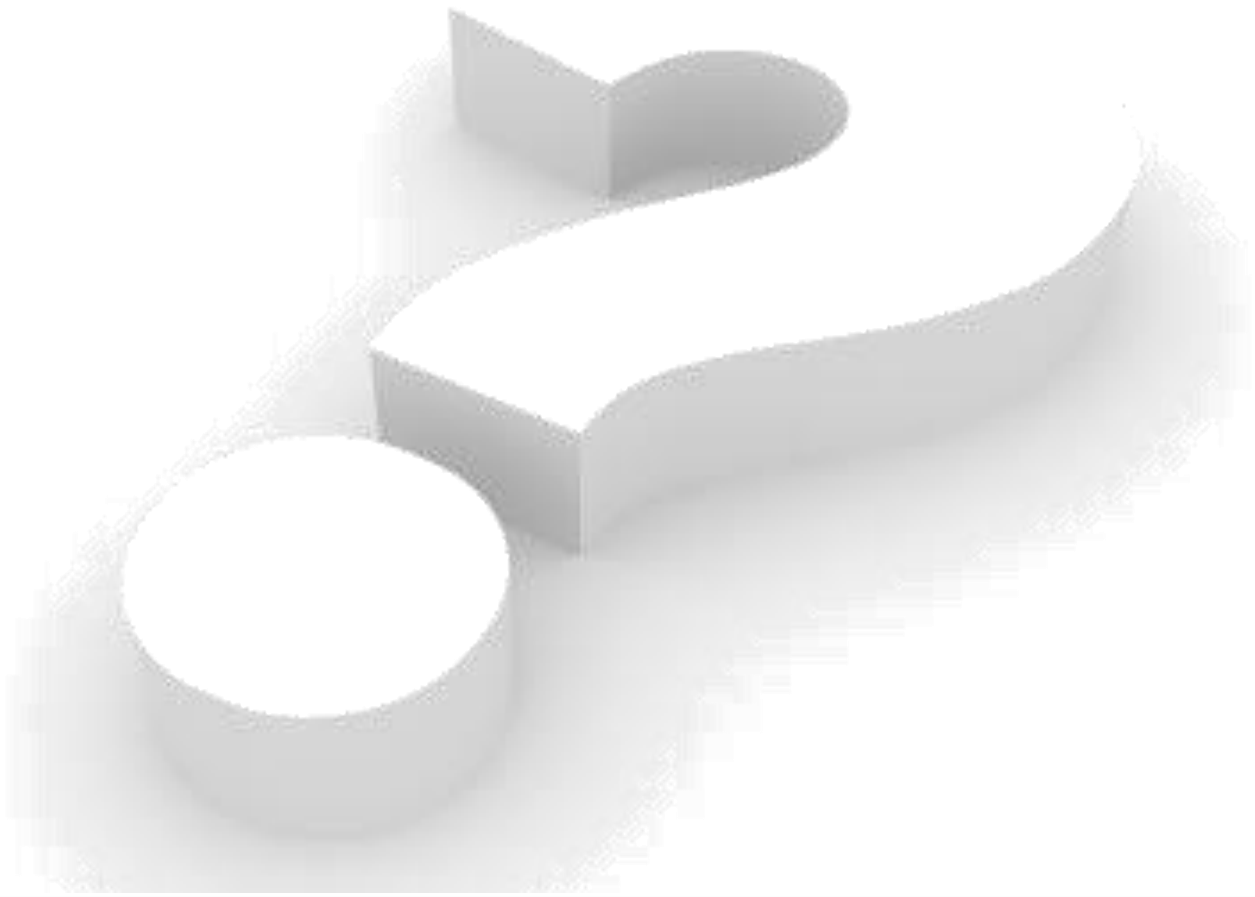
Fragments Web

- Système permettant de partitionner le Descripteur de déploiement
  - Utile pour les bibliothèques tierces pour inclure une configuration Web par défaut
- Un Fragment Web, en tant que Descripteur de déploiement, est un fichier XML :
  - Nommé **web-fragment.xml**
  - Placé dans le dossier **META-INF** de la bibliothèque
  - Avec **<web-fragment>** comme élément racine

Web Fragments Example

```
<!-- Example of web-fragment.xml -->
<web-fragment>
    <name>MyFragment</name>
    <listener>
        <listener-class>
            com.enterprise.project.module.MyListener
        </listener-class>
    </listener>
    <servlet>
        <servlet-name>MyServletFragment</servlet-name>
        <servlet-class>
            com.enterprise.project.module.MyServlet
        </servlet-class>
    </servlet>
</web-fragment>
```

Questions ?



Exercices

- Il est temps d'être moderne !
  - Supprimez le Descripteur de Déployeur de votre projet
  - Mettez à jour vos servlets et votre filtre pour utiliser les annotations

Fin

Merci de votre attention