LES ATTRIBUTS DANS LE CONTENEUR WEB

i. Attribut de portée requête
ii. Attribut de portée session
iii. Les attributs de portée application

Il est souvent utile de partager de l'information dans un conteneur Web entre plusieurs composants. L'information partagée est appelée **un attribut**. Il est défini par sa portée :

Portée de requête (request scope)
L'attribut est disponible pour les servlets qui participent au traitement de la requête cliente. Nous verrons bientôt qu'une même requête peut être prise en charge par une chaîne de plusieurs servlets.

Portée de session (session scope)
L'attribut est disponible pour toutes les requêtes émises par un même client. L'ensemble de ces attributs est appelé une session utilisateur.

Portée de l'application (application scope)
L'attribut est disponible à tout moment à l'ensemble des servlets de l'application Web.

Un attribut est une notion très vague : il s'agit simplement d'une instance d'une classe Java (implémentant si possible l'interface java.io.Serializable).

Le développeur d'application va pouvoir stocker les attributs dont il a besoin dans différents objets fournis par le conteneur (suivant la portée qu'il désire donner à un attribut). La gestion des attributs se fait toujours avec des méthodes aux signatures similaires :

void XXXX.setAttribute(String nom, Object valeur)
Pour positionner un attribut

Object XXXX.getAttribute(String nom)
Pour récupérer un attribut que l'on a préalablement positionné

java.util.Enumeration<String> XXXX.getAttributeNames()
Pour connaître la liste des noms des attributs disponibles

void XXXX.removeAttribute(String nom)
Pour supprimer un attribut

Remarque : Dans la liste ci-dessus XXXX représente l'objet du conteneur stockant les attributs.

> ***Attributs et concurrence***
>
> Nous avons vu au chapitre précédent que la programmation de servlets implique de prendre en considération les problèmes de programmation concurrente (une servlet peut traiter N requêtes simultanément). Ainsi les attributs de portée session et application sont aussi potentiellement accessibles par du code traitant des requêtes simultanées. Il faut donc protéger les modifications concurrentes sur ces attributs.

ATTRIBUT DE PORTÉE REQUÊTE

Les attributs de portée requête sont directement accessibles depuis une instance de `HttpServletRequest`.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {
  User user = new User();
  user.setName(req.getParameter("name"));
  user.setAddress(req.getParameter("address"));
  // on stocke l'utilisateur comme attribut de requête
  req.setAttribute("user", user);
```

```
  // ...
}
```

On peut ensuite récupérer l'attribut à n'importe quelle étape du traitement de la requête

```
User user = User.class.cast(req.getAttribute("user"));
```

Les cas d'usage d'un attribut de portée requête sont limités pour le traitement d'une requête par une seule servlet. Ce type d'attribut devient beaucoup plus utile dès que plusieurs composants Web sont utilisés pour traiter une requête. Nous y reviendrons à plusieurs reprises lorsque nous parlerons des JSP, du modèle MVC et de JSF.

ATTRIBUT DE PORTÉE SESSION

Les attributs de portée session sont disponibles pour toutes les requêtes émises par un même client. Cela signifie que l'application Web mémorise des données relatives au client constituant ainsi une session utilisateur. Le conteneur représente cette session grâce à une instance de la classe HttpSession disponible depuis une instance de HttpServletRequest grâce à la méthode `getSession()`.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {
  User user = new User();
  user.setName(req.getParameter("name"));
  user.setAddress(req.getParameter("address"));
  // on stocke l'utilisateur comme attribut de session
  req.getSession().setAttribute("user", user);

  // ...
}
```

On peut ensuite récupérer l'attribut lors du traitement de n'importe quelle requête reçue de ce client.

```
User user = User.class.cast(req.getSession().getAttribute("user"));
```

Session HTTP : comment cela marche-t-il ?

HTTP est un protocole non connecté et sans état (stateless). Cela signifie que la notion de session n'est tout simplement pas implémentable avec seulement HTTP !

Pour parvenir à créer une notion de session, le conteneur Web associe à une instance de HttpSession un identifiant (le sessionId). Ce sessionId est envoyé au client sous la forme d'un cookie nommé JSESSIONID. Le client renvoie ensuite ce cookie dans l'en-tête de chaque requête envoyée au serveur. Ce dernier retrouve la bonne instance de HttpSession à partir du sessionId et peut l'associer à la requête entrante.

Les limites de la session HTTP

La relative simplicité d'une session HTTP a conduit beaucoup de développeurs à en user (et en abuser). Cependant, ses avantages peuvent très vite être contre-balancés par ses inconvénients.

Il est très difficile pour le conteneur Web de déterminer quand une session doit être supprimée du serveur. En effet, un utilisateur d'un site Web peut très facilement se comporter de façon inattendue. Il peut, par exemple, fermer son navigateur Web ou quitter une page comme bon lui semble sans que le serveur n'en soit informé. Le conteneur considère donc qu'une session est invalidée et peut être détruite s'il ne reçoit plus de requête avec le sessionId pendant un certain laps de temps (généralement configuré à 30 minutes par défaut dans les serveurs d'application). Cela signifie qu'un développeur ne peut jamais être sûr à 100% qu'une session est disponible avec les attributs qu'il espère y trouver. L'utilisation de session peut donc très rapidement favoriser l'apparition de bugs.

Le rattachement d'un client à une session n'est finalement basé que sur l'échange d'un cookie entre un client et un serveur. Un utilisateur mal intentioné peut intercepter une requête HTTP d'un client, extraire le cookie et émettre au serveur sa propre requête avec le cookie volé. C'est ce qu'on appelle un vol de session. L'utilisation de session peut donc très rapidement entraîner des failles de sécurité.

Pour ces raisons (et d'autres encore), il est raisonnable de limiter au maximum l'emploi de la session Web. D'ailleurs, le fait que HTTP soit un protocole sans état n'est ni un hasard ni un oubli mais bien une contrainte voulue par ses auteurs pour garantir la robustesse du Web. Ce sont

les développeurs Web qui ont introduit plus tard la notion de session afin de faciliter la réalisation de certaines fonctionnalités pour leurs applications.

LES ATTRIBUTS DE PORTÉE APPLICATION

Les attributs de portée application sont stockés dans une instance d'un objet ServletContext. Le `ServletContext` représente le contexte d'exécution de l'ensemble des servlets, c'est-à-dire le contexte d'exécution de l'application Web. L'instance de `ServletContext` est accessible depuis n'importe quelle servlet de l'application grâce à la méthode `getServletContext()`.

```
public void init() throws ServletException {
      // on stocke la date de création de la servlet
      this.getServletContext().setAttribute("creationDate", new java.util.Date());
}

@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {
  // on récupère la date de création de la servlet
  java.util.Date creation = java.util.Date.class.cast(this.getServletContext()
                                                          .getAttribute("creationDate"));
  // ...
}
```

Les attributs de portée application sont généralement des données de configuration de l'application créées au lancement ou bien des données mises en cache pour améliorer les performances.