

# DOSSIER D'ANALYSE

## 1. PorphyryServlet

### 1.1. Rôle de la classe

Cette classe supervise la création et l'affichage de la page web. Pour cela, elle récupère les paramètres `id` (numéros de descripteurs sélectionnés, initialement à '0') et `style` (feuille de style, initialement à 'default') de la requête effectuée par le client, les traite et les stocke dans des structures exploitables, fait appel aux fonctions permettant la génération de la ou des page[s] HTML demandée[s], puis la [les] affiche dans le navigateur.

### 1.2. Principales méthodes

Cette classe est une *Servlet*, c'est-à-dire que le serveur web communique avec elle via une interface basée sur trois principales méthodes :

- `init()`
- `service()`
- `destroy()`

La classe ***HttpServlet***, dont hérite *PorphyryServlet*, fournit une implémentation de la méthode `service()` qui dispatche les commandes http (`GET`, `POST`, etc) vers une ou plusieurs méthodes spécifiques, dont `doGet()`. C'est pourquoi la méthode `service()` n'est pas implémentée dans notre classe.

- `void init(ServletConfig config)`

Paramètre d'appel :

- `config` : objet de configuration utilisé par le moteur de servlets - ici, *Tomcat* - pour passer des informations à la servlet durant l'initialisation.

Cette méthode est lancée automatiquement la première fois que la servlet est invoquée. Celle-ci appelle à son tour la méthode de connexion au serveur de graphes. En cas d'erreur, une re-connexion sera tentée, et si le problème persiste, la servlet affichera un message d'erreur .

- `void doGet(HttpServletRequest req, HttpServletResponse res)`

Paramètres d'appel :

- `req` : objet créé par le conteneur de servlet et comprenant, entre autres, les paramètres inclus dans l'URL d'appel de la servlet.
- `res` : objet de réponse, qui peut être par exemple soit une URL, soit du texte – page HTML ou tout simplement String.

Cette méthode est lancée à chaque nouvelle requête.

Pour récupérer la liste des valeurs passées dans l'URL correspondant à un paramètre donné, on utilise la méthode `getParameter(String name)`. S'il y a plusieurs valeurs, celles-ci sont stockées les unes à la suite des autres dans une seule et unique chaîne de caractères, et séparées par un espace. Il s'agit donc ensuite de les séparer et de les stocker dans une structure adéquate – en l'occurrence ici, une List.

Le mieux aurait été d'utiliser la méthode `getParameterValues(String name)`, qui stocke lesdites valeurs dans un tableau de `String`, mais celle-ci ne fonctionne pas correctement, il a donc fallu se rabattre sur la première méthode, même si cela manque d'élégance...

Pour construire la réponse au client, on utilise, si c'est une URL, la méthode `sendRedirect(String location)`; si c'est une page HTML ou du texte, on utilise `getOutputStream()`, qui renvoie un objet de type `ServletOutputStream` dans lequel on peut "écrire".

- `Object merge(List selection, String stylesheet)`

Paramètres d'appel :

- `selection` : liste des clés de descripteurs préalablement récupérées par la méthode `doGet()`.
- `stylesheet` : feuille de style, récupérée elle aussi dans l'URL.

Cette méthode fusionne les différentes pages HTML générées par les classes *RequestManager* et *DescriptionManager*, si elles ont lieu d'être, c'est-à-dire, par exemple, s'ils y a des documents à afficher. Si c'est non, on se contente de faire appel au *RequestManager*, qui crée la partie supérieure de la page web, contenant les descripteurs sélectionnés, et les descripteurs "sélectionnables". S'il se produit une erreur, la méthode `reconnect()` sera appelée, pour déterminer la nature du problème, et l'adresse de la page d'erreur correspondante sera communiquée à la méthode `doGet()`.

## 2. DescriptionServlet

Cette servlet est utilisée par le client natif, bien qu'elle puisse aussi être appelée via un navigateur web classique. Son rôle est réduit : elle ne reçoit en paramètre qu'un seul numéro d'identifier et une feuille de style, fait appel à la fonction `createPath()` du *DescriptionManager*, et renvoie au client le ou les chemin(s)\* du document.

\* Voir la définition d'un chemin dans le paragraphe 4.1.

## 3. RequestManager

### 3.1. Rôle de la classe

Cette classe crée la partie supérieure de la page web, contenant la liste des descripteurs sélectionnés – que l'on peut enlever de la requête, et celle des descripteurs "sélectionnables" – que l'on peut ajouter à la requête.

La liste des descripteurs sélectionnés est contenue dans l'URL, dans le paramètre `id`. Si aucune valeur n'est contenue dans ce paramètre, la valeur prise par défaut est 0.

### 3.2. Principales méthodes

- `boolean updateData(List selectedKeys, GraphServer ref, Integer sessionId)`

Paramètres d'appel :

- `selectedKeys` : liste des clés de descripteurs sélectionnées dans la page web.
- `ref` : référence à la classe *porphyry.server.userManagement.GraphServer*.
- `sessionId` : numéro de session, obtenu lors de la connexion.

Cette méthode doit être appelée avant les autres méthode de la classe, car elle fait appel à la fonction `getGraphView(sessionId, selectedKeys)` de la classe `GraphServer` et met ainsi à jour l'attribut `graphView`, contenant trois listes : descripteurs, identifiants, spécialisations. La liste des descripteurs contient leur clé et leur état (connu, possible, impossible). La liste des identifiants n'est en fait qu'une liste de clés, de type `Integer`.

- `Vector getAllDescriptors(List selectedKeys, GraphServer ref, Integer sessionId)`

Cette méthode récupère les noms des descripteurs connus et possibles, et les stocke dans deux collections différentes, qui seront retournées ensuite. Ces collections sont des dictionnaires contenant des paires clé/label.

Elle met aussi à jour l'attribut `identifiersMap`, contenant des paires clé/URL, et qui sera retourné par la méthode `getAllIdentifiers()`.

- `StringWriter create(List selectedKeys, GraphServer ref, Integer sessionId, String styleSheet)`

Cette méthode crée la structure du fichier XML – les balises, les attributs, la racine du document – et la remplit avec les données récupérées grâce à la méthode précédente.

Elle fait appel à la méthode `render()`, qui transforme le XML en HTML, et retourne la page ainsi générée.

- `StringWriter render(File xmlFile)`

Paramètre d'appel :

- `xmlFile` : fichier XML à transformer en HTML.

Cette méthode applique la feuille de style XSL par défaut – dont le chemin est stocké dans le fichier `porphyry.webserver.Constant` – au fichier XML passé en paramètre. L'adresse de la feuille de style n'a pas lieu d'être paramétrable : si l'on veut modifier la présentation de la page web, il faut soit modifier le XSL, soit en créer un nouveau, et changer l'adresse dans `Constant`.

## 4. DescriptionManager

### 4.1. Rôle de la classe

Cette classe crée la partie inférieure de la page web, contenant le ou les document(s) - image ou texte - dont l'état est *CONNU* ou *POSSIBLE*, et pour chacun d'eux, le(s) chemin(s) y menant.

Un *chemin* est ici la liste ordonnée des descripteurs, du sommet du graphe à l'identifiant correspondant au document. Il peut y avoir plusieurs chemins pour un même document.

### 4.2. Principales méthodes

- `Map getAllTrees(List keysList, GraphServer ref, Integer sessionId)`

Paramètre d'appel :

- `keysList` : liste des clés des identifiants récupérée par la classe `PorphyryServlet` via l'appel à la fonction `RequestManager.getAllIdentifiers()`.

Cette méthode met aussi à jour l'attribut `specialMap`, contenant des paires clé/ensemble, où la clé est celle d'un objet `Specialization.special` – un "fils" en quelque sorte ; et la valeur associée à cette clé est l'ensemble des `Specialization.general` – des "parents" - correspondant à ce "fils".

- `StringWriter create(Map identifiersMap, GraphServer ref, Integer sessionId, String styleSheet)`

Même principe que pour la classe précédente.

- `StringWriter createPath(Integer idKey, GraphServer ref, Integer sessionId, String styleSheet)`

Cette méthode est appelée par la classe `DescriptionManager`. Elle crée le fichier XML correspondant à l'unique clé passée en paramètre, et fait appel à la méthode `render()`, avec sa propre feuille de style – différente de la méthode `create()`.

- `getPath(Integer key, List existingPath)`

Paramètres d'appel :

- `key` : clé de l'identifiant dont on recherche les chemins.
- `existingPath` : liste ordonnée des descripteurs constituant un de ces chemins. Le dernier de la liste est le descripteur précédant immédiatement l'identifiant.

Cette fonction est **récursive** : à chaque nouvel appel, elle ajoute un descripteur au début de la liste existante passée en paramètre, où la clé est le descripteur ajouté à l'appel précédent.

Ce nouveau descripteur est le « parent » du descripteur passé en paramètre. S'il y en a plusieurs, la liste est copiée autant de fois qu'il y a de parents, et à chaque liste est ajouté un parent différent. Puis la fonction se rappelle elle-même pour chacun des nouveaux chemins. Elle s'arrête lorsque le parent est le sommet de l'arbre, car c'est toujours le même et sa valeur est 0.

La collection des fils/parents est l'attribut `specialMap` mis à jour par la méthode `getAllTrees()`.

- `StringWriter render(File xmlFile, String styleSheet)`

Dans cette méthode, l'adresse de la feuille de style est paramétrable : il en existe une par défaut, si la valeur de `style` dans l'URL est à 'default' ou s'il n'y en a pas ; sinon, la nouvelle valeur – supposée valide – est prise en compte.