TP345

Développement d'un robot Web

1 Objectif

L'objectif est de faire un programme qui, étant donné une requête de recherche lue au clavier, est capable de produire une page web contenant des liens sur des pages satisfaisant cette requête. La page web produite contiendra la liste des titres des pages qui seront des liens cliquables vers les pages en question. Pour illustrer ce TP, comme **site de référence** on se servira du site "Vivastreet". Par exemple, en supposant que l'on donne comme requête `(peugeot and rennes) and (307 or 308)`, l'objectif est de produire une page web contenant des liens sur des annonces du site de référence dont le titre, mots clés ou texte de l'annonce contiennent les mots "peugeot", "rennes" et soit "307" soit "308". Dans les requêtes, on acceptera n'importe quelle combinaison de mots clés combinés avec `and` et `or`.

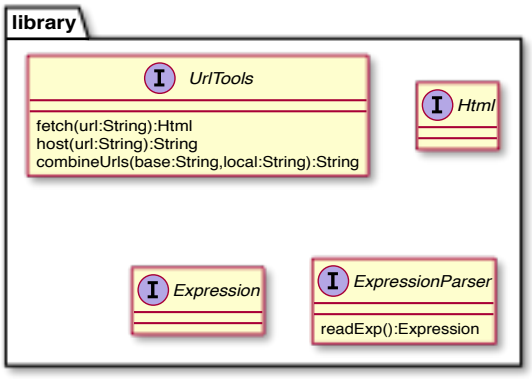
2 Préambule

Dans EclipseGen, importez le projet se trouvant dans `/share/l2ie/gen/TP345/TP345.zip` de la façon suivante :

File>Import>Existing Projects into Workspace>Select archive file

Le projet installé est composé de trois répertoires : un répertoire `src` qui contiendra le source du projet, un répertoire `lib` qui contient trois librairies nécessaires au projet (lecture et mise en forme de documents HTML). Dans `src` se trouve un package `library` dans lequel vous trouverez 4 fichiers Scala :

- `Expression` qui contient le type des expressions utilisées pour les requêtes ainsi que la méthode `readExp` capable de lire au clavier une chaîne de caractères et de la transformer en une expression de ce type ;
- `HtmlStructure` qui contient le type `Html` utilisé pour représenter les documents HTML dans le projet ;
- `UrlTools` qui contient le trait du même nom ainsi qu'un objet implantant ce trait. Cet outil vous fournit 3 outils essentiels pour analyser des URLs ;
- `Decoupage` qui contient les traits définis plus bas.

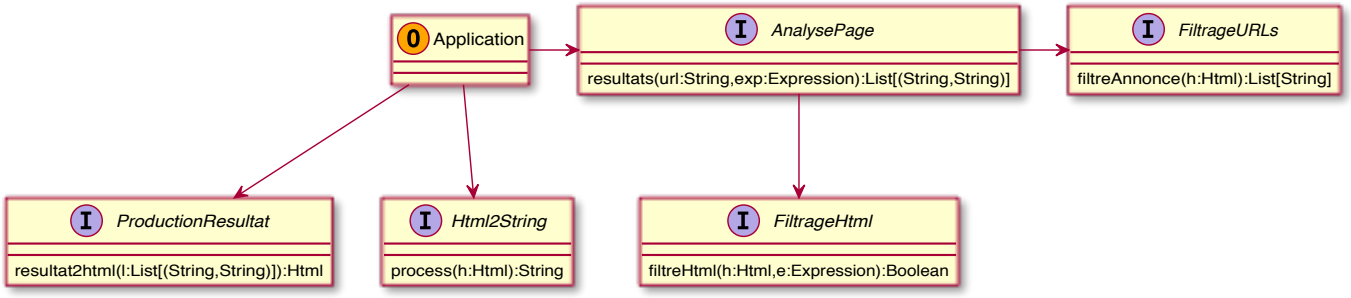


Vous aurez également besoin d'écrire vos résultats dans un fichier. Une façon correcte de faire cela en Scala est :

```
import java.io.FileWriter
val file= new FileWriter("monFichier.txt")
try{
   file.write("Ceci est le contenu du fichier créé")
} finally file.close()
```

3 Découpage de l'application

On propose de découper le travail des groupes de TP en 6 équipes de la façon suivante :



3.1 Application globale

1. Lit la requête
2. Construit l'URL à passer au site de référence
3. Utilise `AnalysePage.resultats` (équipe 3.2) avec cette URL pour obtenir la liste de couples résultats (Titre,URL) qui satisfont la requête
4. Utilise `ProductionResultat.resultat2html` (équipe 3.5) pour obtenir le document `Html` rassemblant la liste de couples résultats
5. Utilise `Html2String.process` (équipe 3.6) pour obtenir la chaîne de caractères correspondant au document `Html` précédent
6. Ecrit le résultat dans un fichier.

3.2 Analyse de la page

```
trait AnalysePage{
/** A partir d'une URL de requête sur le site de référence et d'une expression exp,
    retourne une liste de pages issues de la requête et satisfaisant l'expression.

    @param url l'URL de la requête sur le site de référence
    @param exp l'expression à vérifier sur les pages trouvées
    @return la liste des couples (titre,ref) où ref est l'URL d'une page
            satisfaisant l'expression et titre est son titre. */

  def resultats(url:String,exp:Expression):List[(String,String)]
}
```

1. Récupère le document `Html` associé à la page `url`
2. Utilise `FiltrageURLs.filtreAnnonce` (équipe 3.3) pour extraire du document `Html` la liste `lURLs` des URLs correspondant à des annonces
3. Récupère la liste des documents `Html` associés à la liste `lURLs`
4. Utilise `FiltrageHtml.filtreHtml` (équipe 3.4) pour construire une liste de couples (URL,document `Html`) qui satisfont la requête
5. Extrait le titre de chaque page solution, pour renvoyer ainsi une liste de couples (Titre,URL)

3.3 Filtrage des URLs

```
trait FiltrageURLs{
/** A partir d'un document Html h, rend la liste des URLs accessibles à partir
    de h (ces URLs sont des hyperliens h) tels que ces URLs sont tous des URLs
    d'annonces du site de référence

    @param h le document Html
    @return la liste des URLs d'annonces contenues dans h
 */

  def filtreAnnonce(h:Html):List[String]
}
```

1. Extrait la liste des URLs incluses dans un document `Html`
2. Retire de cette liste les URLs qui ne sont pas des URLs d'annonces du site de référence

3.4 Filtrage des documents Html pour une requête donnée

```
trait FiltrageHtml{
/** A partir d'un document Html h et d'une requête e, dit si le document
    satisfait l'expression e

    @param h le document Html
    @param e l'expression
    @return true si le document satisfait l'expression e
*/

  def filtreHtml(h:Html,e:Expression):Boolean
}
```

1. Définit si un document `Html` contient un mot clé
2. Définit cette vérification pour des requêtes avec `And` et `Or`

3.5 Production de Pages Html

```
trait ProductionResultat{
/** A partir d'une liste de couples (titre,URL), produit un document Html, qui
    liste les solutions sous la forme de liens cliquables

    @param l la liste des couples solution (titre,URL)
    @return le document Html listant les solutions
 */

  def resultat2html(l:List[(String,String)]):Html
}
```

3.6 Html vers String

```
trait Html2String{
/** Produit la chaîne de caractères correspondant à un document Html

    @param h le document Html
    @return la chaîne de caractères représentant h
 */

  def process(h:Html):String

}
```