GIANG
ELODIE
3678177

RAPPORT MINI PROJET 3

Le but du mini-projet est de proposer dans une première partie une API pour le site http://dblp.uni-trier.de/db/ht/ sur lequel sont regroupés l'ensemble des publications scientifiques et informatiques, puis dans une seconde partie de réaliser un serveur web utilisant cet API Web développée dans la première partie.
Le premier point majeur qui nous a posé problème dès le début du projet est la taille conséquente du fichier xml mis à notre disposition (3Go) qui nous empêche de relancer l'analyse du fichier à chaque requête. Il était donc nécessaire de réaliser un pré-traitement en codant un programme permettant de générer plusieurs sous-fichiers plus légers. Pour cela, notre code divise le fichier xml principal en sous-fichiers contenant un nombre spécifique de publications. A l'aide des balises contenues dans le fichier xml, nous avons pu les compter et fixer a 500 000 le nombre de publications par fichier. Nous nous sommes donc retrouvées avec 11 fichiers xml de taille avoisinant les 250 Mo. Cependant, nous avions eu un problème sur l'analyse de ses fichiers car nous avions décidé de traiter le mini-projet avant de parser le fichier ce qui a fait que nous avions manqué de temps pour adapter notre code aux fichiers tronqués. Nous avons donc décidé de tester notre code sur le fichier dblp_2020_2020.xml fourni en annexe.

Exercice 1 :

-/publications/{id} : Pour identifier chaque publication, nous avons décidé de choisir leur titre. En itérant sur chaque balise « title », nous avons donc pu identifier la publication ayant le titre « id » passé en paramètre. La fonction renvoie une erreur de type 404 lorsque la publication n'existe pas.

Par la suite, lors de l'intégration du paramètre d'url order dans nos fonctions, nous nous sommes confrontées à un problème qui est que lorsque l'utilisateur souhaitait trier en fonction d'une balise présente à plusieurs reprises dans les informations d'une publication (une publication peut être écrite par plusieurs auteurs par exemple). Il nous a fallu ignorer ce cas et ne traiter que le cas où l'utilisateur souhaiterait trier par rapport à une balise unique pour chaque publication.

-/publications : La fonction utilise un dictionnaire pour trier les publications renvoyées en fonction du paramètre d'url « order ». En effet, à chaque publication parcourue (si la position de la publication rentre dans l'intervalle délimité par les paramètres d'url start et count), celle-ci est stockée dans un dictionnaire associée à la valeur correspondant à la balise « order » précisée. Pour renvoyer la liste des publications et des informations qui lui sont associées, nous avons rajouté une fonction id_publi qui prend en paramètre un titre de publication et renvoie une chaîne de caractères contenant toutes ses informations. Des balises « articles » ont été placées afin de pouvoir compter le nombre de publications (utile pour les tests).
-/authors/{name} : En parcourant toutes les balises filles de la racine, nous pouvons identifier les publications dont le nom d'auteur correspond au paramètre « name » passé en argument. Les publications renseignent alors les auteurs qui ont co-écrits le livre avec l'auteur (qu'on place dans une liste) ainsi le nom de la publication (qu'on place aussi dans une liste). La taille des deux listes ainsi remplies permettent de déterminer le nombre de co-auteurs et de publications de l'auteur « name ».
-/authors/{name}/publications : La fonction associée à la route fonctionne de la même manière que pour la route /publications. En effet, les paramètres d'url start, count et order ont la même fonction que pour cette publication. Ici aussi nous avons utilisé un dictionnaire pour ranger et trier les publications en fonction du paramètre d'url order.

-/authors/{name}/coauthors : La fonction associée à la route fonctionne de la même manière que pour la route /publications. En effet, les paramètres d'url start, count et order ont la même fonction que pour cette publication. Ici aussi nous avons utilisé un dictionnaire pour ranger et trier les co-auteurs en fonction du paramètre d'url order.
-/search/authors/{searchString} : Même fonctionnement que pour /authors/{name}/coauthors. Nous avons cependant rajouté une fonction word_in permettant de tester si une chaîne de caractères contenait un mot passé en paramètre (ici en l'occurrence, on testait la présence de « searchString » dans la chaîne de caractères.
-/search/publications/{searchString} : Même fonctionnement que pour /search/authors/{searchString}. Une difficulté supplémentaire est que la fonction associée à cette route devait prendre en compte un paramètre d'url filter de la forme key1:value1,key2:value2,… Pour cela, nous avons décidé d'utiliser json pour pouvoir transformer une chaîne de caractères en un dictionnaire. Il a fallu alors adapter la syntaxe de la chaîne de caractères pour pouvoir appliquer la fonction loads de json. Le dictionnaire pouvait ensuite être manipulé plus facilement.
-/authors/{name_origin}/distance/{name_destination} : La difficulté majeure de la fonction associée à cette route était de trouver un algorithme assez optimisé pour ne pas avoir à parcourir trop de fois le fichier xml afin de trouver une distance minimale. Nous avons essayé d'optimiser le plus possible la fonction mais n'avons pas réussi à l'optimiser assez pour garantir une performance de calcul assez rapide. En effet, lorsque le nombre de co-auteurs d'un auteur est trop important, le temps d'exécution du programme est trop long. Une autre difficulté était de gérer le renvoi de la fonction. En effet, pour afficher le renvoi sur une page Web, il fallait générer une chaîne de caractères ce qui n'était pas possible si on se limitait à une seule fonction. Nous avons donc décidé d'écrire une fonction distance qui nous servirait de fonction récursive. En effet, cette fonction prend en paramètre un auteur d'origine et le même auteur de destination et alimente le chemin qui le sépare de son auteur de destination. Pour cela, la fonction va être appelée récursivement sur les co-auteurs de l'auteur d'origine. La distance augmente dès lors que l'on appelle la fonction sur un co-auteur. Si un auteur n'a plus de co-auteur, c'est que l'on ne peut pas trouver l'auteur de destination à partir de ce chemin, on renvoie donc une distance infinie. Si on le trouve on renvoie la plus petite distance des chemins de chacun de ses co-auteurs ainsi que le chemin correspondant. Pour trouver la plus petite distance d'un ensemble de chemin, nous avons défini la fonction minimal qui prend en paramètre un dictionnaire contenant des chemins et leur distance et renvoie le plus petit chemin et la distance correspondante. Le plus petit chemin trouvé est stocké et chaque parcours de chemin qui est plus grand que le plus petit chemin actuel renvoie immédiatement un chemin de distance infini pour optimiser le temps d'exécution de la fonction. De plus, un auteur ne compte pas un autre auteur comme son co-auteur si cet auteur a déjà parcouru ses co-auteurs. La fonction fonctionne bien pour des petites distances mais pour de longues distances, elle met beaucoup de temps.

Exercice 2 :

-test_1_publication_id : teste si la requête http://{self.server_ip}:{self.server_port}/publications/How to Weigh Values in Value Sensitive Design: A Best Worst Method Approach for the Case of Smart Metering. renvoie le format de réponse souhaité.
-test_2_publication_id: teste si la requête http://{self.server_ip}:{self.server_port}/publications/ renvoie bien 100 publications et non pas une unique publication.

-test_1_publication_100 : compte le nombre de balise <article> pour voir si la requête http://{self.server_ip}:{self.server_port}/publications/ renvoie bien 100 publications.
-test_2_publication : compte le nombre de balise <article> pour voir si la requête http://{self.server_ip}:{self.server_port}/publications/?limit=12 renvoie bien 12 publications.

-test_3_publication : teste si la requête http://{self.server_ip}:{self.server_port}/publications/?start=100&count=2 renvoie bien les 100 premières publications à partir du 2ème élément.
-test_4_publication : teste si la requête http://{self.server_ip}:{self.server_port}/publications/?start=200 renvoie bien 100 publications à partir du 200ème élément.
-test_5_publication : teste si la requête http://{self.server_ip}:{self.server_port}/publications/?count=300 renvoie bien 100 éléments et non 300.
-test_6_publication : teste si la requête http://{self.server_ip}:{self.server_port}/publications/?count=4&order=title renvoie bien 4 publications triées dans l'ordre alphabétique.

-test_1_authors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Louise%20A.%20Dennis renvoie bien la chaîne de caractères demandée.
-test_2_authors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/mj renvoie une erreur lorsque l'auteur n'existe pas.
-test_3_authors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/ renvoie une erreur si l'auteur n'est pas précisé.

-test_1_authors_publications : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Indira%20Nair/publications renvoie bien la chaîne de caractères demandée.
-test_2_authors_publications : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Inconnu/publications renvoie une erreur si l'auteur n'existe pas.
-test_3_authors_publications : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Louise%20A.%20Dennis/publications?start=1 renvoie bien les publications de l'auteur à partir de la publication indexée 1.
-test_4_authors_publications : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Louise%20A.%20Dennis/publications?count=1 renvoie bien une publication de l'auteur.
-test_5_authors_publications : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Louise%20A.%20Dennis/publications?order=journal renvoie bien une liste triée dans l'ordre alphabétique du journal.

-test_1_coauthors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Indira%20Nair/coauthors renvoie bien les co-auteurs de l'auteur.
-test_2_coauthors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Inconnu/coauthors renvoie bien la chaîne de caractère « Pas de co auteurs » pour un auteur qui n'existe pas.
-test_4_coauthors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Alexander%20Guda/coauthors?start=2 renvoie bien la liste de co-auteurs à partir du co-auteur 2.
-test_5_coauthors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Alexander%20Guda/coauthors?count=3 renvoie bien 3 co-auteurs de l'auteur.
-test_6_coauthors : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Alexander%20Guda/coauthors?order=author trie bien la liste de co-auteurs de l'auteur par ordre alphabétique des auteurs.

-test_1_search_authors : teste si la requête http://{self.server_ip}:{self.server_port}/search/authors/alex renvoie bien 100 auteurs contenant la chaîne de caractères « alex »
-test_2_search_authors(self): teste si http://{self.server_ip}:{self.server_port}/search/authors/Tsalidis renvoie la bonne chaîne de caractères.
-test_3_search_authors : teste si http://{self.server_ip}:{self.server_port}/search/authors/Chistos renvoie : « Aucun auteur trouvé »

-test_4_search_authors : teste si la requête http://{self.server_ip}:{self.server_port}/search/authors/Pauline?start=15 renvoie les 15 premiers éléments de la liste.
-test_5_search_authors : teste si la requête http://{self.server_ip}:{self.server_port}/search/authors/Pauline?count=2 renvoie 2 éléments.
-test_6_search_authors : teste si la requête : http://{self.server_ip}:{self.server_port}/search/authors/Pauline?order=author renvoie bien une liste triée par auteur.
-test_7_search_authors(self): teste si la requête http://{self.server_ip}:{self.server_port}/search/authors/Pauline?start=50 renvoie une erreur.

-test_1_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/moral renvoie la bonne chaîne de caractères.
-test_2_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/smsmsms renvoie bien « Aucune publication trouvée ».
-test_3_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/moral?count=10 renvoie 10 publications qui contiennent « moral ».
-test_4_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/moral?start=100&count=3 renvoie bien 3 publications contenant « moral » à partir de la 100^{e} publication.
-test_5_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/morali?order=author renvoie une liste triée dans l'ordre alphabétique des auteurs.
-test_6_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/moral?filter=number:2,year:2 renvoie bien la bonne liste pour un filtre : number:2,year:2
-test_7_search_publications : teste si la requête http://{self.server_ip}:{self.server_port}/search/publications/morali?start=500 renvoie bien une erreur.

-test_1_distance : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Ming Shan/distance/Amos Darko renvoie la bonne chaîne de caractères et la bonne valeur.
-test_2_distance : teste si la requête http://{self.server_ip}:{self.server_port}/authors/Ming %20Shan/distance/Peter%20Rasche renvoie bien qu'il n'y a pas de chemin entre les deux auteurs.

-test_1_error : teste si les requêtes http://{self.server_ip}:{self.server_port}/authors/ et http://{self.server_ip}:{self.server_port}/publications/mj ont le même format.
-test_2_error : teste si les requêtes http://{self.server_ip}:{self.server_port}/search/authors/smsmsms, http://{self.server_ip}:{self.server_port}/search/publications/smsmsms et http://{self.server_ip}:{self.server_port}/authors/Ming%20Shan/distance/Peter%20Rasche ont une valeur de retour de même format.

Exercice 3 :

Le but de l’exercice est de créer un serveur Web à partir de l’API Web développée dans l’exercice 1.
-/@route("/author") : La fonction associée à cette route permet de chercher la liste complète des publications et des co-auteurs d’un auteur. Pour cela elle propose soit de chercher un auteur à partir d’une sous -séquence de caractères (et renvoie donc à @route("/author", method='POST')) filtré par un paramètre order ou non ou alors de chercher directement la liste des publications et des co-auteurs directement à partir du nom d’un auteur (renvoie à la route @route("/doauthor", method='POST')).
-@route("/author", method='POST') : La fonction do_input associée à cette fonction permet d’effectuer une requête web permettant d’exécuter la fonction associée à la route search/authors/{searchString} définie dans l’exercice 1. La réponse de cette fonction renvoie de même à un formulaire permettant d’accéder à la route @route("/doauthor", method='POST').
-@route("/doauthor", method='POST') : La fonction associée à cette route permet d’effectuer une requête web permettant d’exécuter les fonctions associées aux routes authors/{name}/publications et /authors/{name}/coauthors définies dans l’exercice 1.

-@route("/dist") : La fonction associée à cette route permet de chercher la distance entre deux auteurs. Pour cela elle propose soit de chercher deux auteurs à partir de sous -séquences de caractères (et renvoie donc à @route("/dist", method='POST')) ou alors de chercher directement la distance entre les deux auteurs à partir du nom des auteurs (renvoie à la route @route("/doinputdist", method='POST')).
-@route("/dist", method='POST') : La fonction do_input_dist associée à cette fonction permet d’effectuer une requête web permettant d’exécuter la fonction associée à la route search/authors/{searchString} définie dans l’exercice 1. La réponse de cette fonction renvoie de même à un formulaire permettant d’accéder à la route @route("/doinputdist", method='POST').
-@route("/doinputdist", method='POST') : La fonction associée à cette route permet d’effectuer une requête web permettant d’exécuter la fonctions associé à la route @route('/authors/<name_origin>/distance/<name_destination>') définie dans l’exercice 1.