Cahier de TPs

JMeter

Pré-requis :
Poste développeur avec accès réseau Internet libre
Linux / Windows 10 / MacOS
Pré-installation de :

- Git
- JDK11
- Optionnellement Docker

Support :
https://github.com/dthibau/jmeter

Table des matières

Atelier 1 : Prise en main de l’interface

1.1 Mise en place de l'application cible

- Récupérer l’application web ***product-service.jar***
- La démarrer avec
  java -jar product-service.jar
- L’application est une application web :
  - qui offre une interface accessible à http://localhost:8080
  - Une API Rest documentée à http://localhost:8080/swagger-ui.html
  - Une API de surveillance disponible à http://localhost:8080/actuator
- A chaque démarrage, la base embarquée H2 est réinitialisée

1.2 Exécution du plan de test

Notre premier plan de test va comporter les éléments suivants :

- Groupe d'unités
- Échantillon HTTP
- Échantillon HTTP par défaut (élément de configuration)
- Un rapport résumé (Récepteur)

Les différentes étapes de l'atelier sont les suivantes :

- Ajouter la configuration par défaut des requêtes HTTP
- Ajouter un groupe d’unité
- Ajouter des échantillons HTTP
- Ajouter un récepteur pour visualiser et stocker les résultats de test
- Sauvegarder et exécuter le plan de test
- Analyser les résultats
- Ré-exécuter le test avec plus d'utilisateurs

1.2.1 Ajouter la configuration par défaut des requêtes HTTP

1. Clic-droit sur l'élément « Plan de test» puis
Ajouter → Configuration → Paramètres HTTP par défaut
2. Renommer éventuellement l'élément
3. Dans le champ serveur, indiquer *localhost*, dans le port http *8080*

1.2.2 Ajouter des utilisateurs

1. Clic-droit sur l'icône du plan de test puis
   Ajouter → Moteur d'utilisateurs → Groupe d'unités
2. Renommer le groupe d'unités en «Application Web» par exemple
3. Pour le moment, nous simulons un seul utilisateur effectuant 5 fois le scénario

1.2.3 Ajouter des requêtes HTTP

Nous allons faire deux requêtes HTTP :
La page d'accueil puis la page des produits

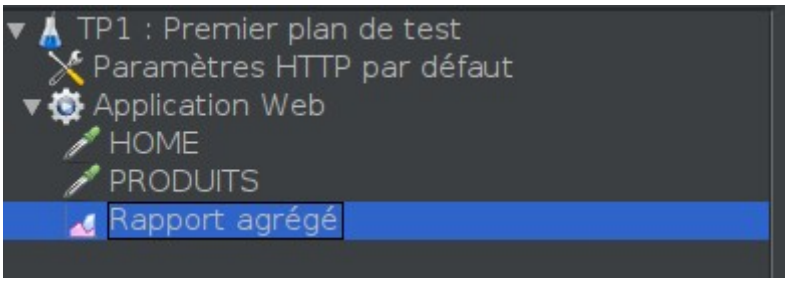
1. Clic-droit sur l'élément « Internautes » puis
   Ajouter → Echantillon → Requête HTTP
2. Renommer le Nom en « Page d'accueil »
   Le champ chemin en « / »
3. Refaire la même chose pour la page /produits

1.2.4 Ajouter un récepteur

Enfin nous ajoutons le récepteur « rapport agrégé »

1. Clic-droit sur l'élément «Application Web» puis
   Ajouter → Récepteur → Rapport agrégé
2. Dans le champ « Écrire les résultats dans un fichier », sélectionner un fichier

Le plan de test doit ressembler à la copie d'écran suivante :



Sauvegarder et exécuter le test.
Visualiser le résultat dans le récepteur et regarder le fichier généré.

1.2.5 Exécution en mode non-gui

Modifier le nombre d'utilisateurs avec l'expression suivante :

- *${__P(USERS,2)}* <=> (La propriété USERS ou 2 si elle n'est pas définie)

Exécuter le test sans GUI :

- en modifiant la propriété USERS
  `./jmeter -n -t TP1.jmx -l result.jtl -JUSERS=100`
- en mode DEBUG
  `./jmeter -n -t ../../Tp1.jmx -l ../../result.jtl -JUSERS=100 -Lorg.apache=DEBUG`
- Génération d'un rapport de test à partir du fichier résultat précédemment créé
  `./jmeter -g result.jtl -o report-batch`

Atelier 2 : Contrôleurs et séquencement de requêtes

Dans ce TP, nous voulons simuler 2 scénarios distincts correspondant à 2 groupes d'unités distincts :

Scénario Rest

1. Appel de la ressource affichant tous les produits
2. Appel de la ressource affichant tous les produits du fournisseur avec l'ID 1
3. Pour 20 % des threads, Ajout d'un produit dans la base avec comme fournisseur l'ID 1

Scénario Web

1. Tous les utilisateurs commencent par la page d'accueil, ils n'y accèdent qu'une seule fois
2. Certains internautes sont chargés des fournisseurs
   Ils accèdent à la liste des fournisseurs puis choisissent d'éditer un fournisseur choisi au hasard parmi 3 et ils reviennent ensuite à la liste des fournisseurs
3. D'autres internautes sont chargés des produits

   Ils accèdent à la liste des produits, filtre la liste avec un fournisseur puis accède au détail d'un produit, modifie le prix et revienne à la liste des produits

Construire le plan de test simulant les 2 scénarios et utiliser un récepteur tableau de résultat pour vérifier la séquence des requêtes

Utiliser un arbre de résultat pour vérifier les réponses HTML et JSON

Atelier 3 : Enregistrement de scénarios

Dans ce TP, nous allons utiliser l'élément Enregistreur de script pour enregistrer des scénarios de test.

Les scénarios que nous voulons enregistrer est similaire au moteur d'utilisateur Web du TP précédent.

Le but de cet atelier est donc d'obtenir après enregistrement un fichier *.jmx* le plus proche de celui qui a été élaboré au TP précédent.

3.1 Mise en place du plan de travail

1. Ajouter un élément ***Proxy (Enregistreur de script)*** au plan de travail
2. Préciser le port que vous désirez utiliser
3. Ajouter un élément de configuration « *Requête HTTP par défaut* » et donnez lui les valeurs pour le serveur et le port
4. Créer 2 contrôleurs et les nommer de façon appropriée:
    - Un contrôleur « **1 seule fois** »
    - Un contrôleur « **enregistreur** »

3.2 Configuration du navigateur

Configurer votre navigateur afin qu'il utilise le proxy JMeter.

3.3 Enregistrement de l'entrée dans l'application

Configurer l'enregistreur comme suit :

- Sélectionner le contrôleur « 1 seule fois »
- Mettre chaque groupe dans un contrôleur de transaction

Démarrer l'enregistreur

Accéder à la page d'accueil

Arrêter l'enregistreur

Visualisez les échantillons sauvegardés dans le contrôleur

3.4 Enregistrement séquence : Liste / Visualisation

Changer la configuration de l'enregistreur

- Afin que les fichiers images, css, js soient ignorés
- Ne pas grouper les échantillons

- Enregistrer dans le contrôleur enregistreur

Démarrer l’enregistreur

Exécuter dans le navigateur la séquence :

- Visualisation de liste de produits
- Sélection produits
- Édition
- Retour à la liste

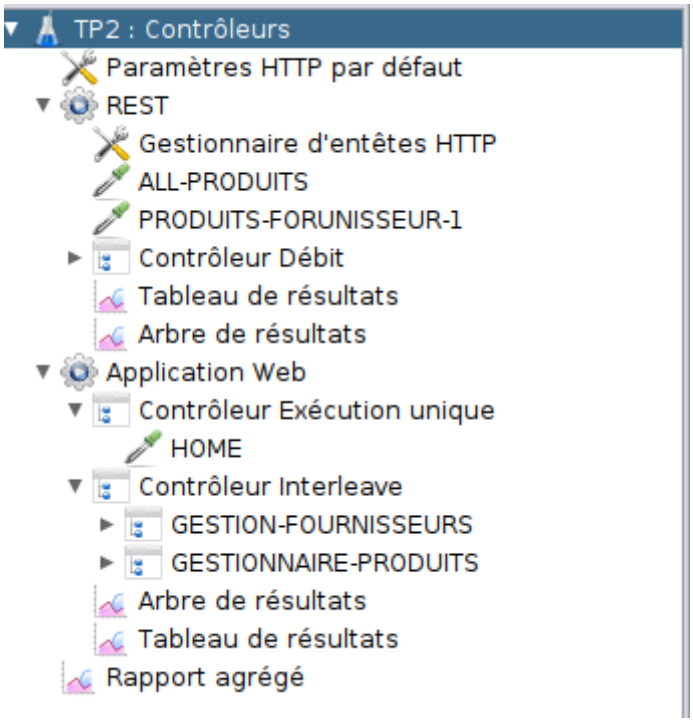
Arrêter l’enregistreur

Visualiser les échantillons enregistrés

Atelier 4 : Variables

L'objectif de ce TP est de faire utiliser les éléments JMeter créant des variables : P*aramètres Utilisateur, Variables prédéfinies, Post-processeur, …*

Nous repartons du TP n°2 :



Nous utilisons une nouvelle version de l'application qui implémente la sécurité :

- Pour la partie web, c'est une sécurité stateful qui stocke dans un cookie de session l'utilisateur loggé
- Pour la partie Rest, c'est une sécurité stateless qui nécessite de passer un jeton à chaque requête. Le jeton peut être demandé à l'adresse /api/authenticate en passant les paramètres login et password

Pour les 2 parties les utilisateurs valides sont fournis au format csv (users.csv)

4.1 Propriétés de configuration

Donner la possibilité de pouvoir spécifier sur la ligne de commande :

- L'adresse du serveur et son port d'écoute
- Le nombre de threads REST et WEB
- Le nombre d'itérations
- Le taux d'ajout de produits

4.2 Utilisation d'une source de données CSV

Utiliser le fichier CSV fourni afin que chaque utilisateur du groupe Web utilise un login/mot de passe différent

4.3 Post-processeur regexp / jQuery

Dans l'application web :

- Utiliser un post-processeur regexp afin qu'une fiche au hasard soit choisie parmi la liste des fournisseurs
- Utiliser un post-processeur CSS et un élément ForEach pour faire une boucle sur tous les fournisseurs.

4.4 Post-processeur Json et authentification via JWT

Utiliser un post-processeur Json pour extraire le jeton JWT

Utiliser le jeton dans les requêtes REST dans l'entête ***Authorization*** avec comme valeur *Bearer ${TOKEN}*

4.5 Paramètres utilisateur et contrôleur If

Utiliser les éléments *Paramètres Utilisateur* et *Contrôleur If* afin de facilement pouvoir changer la proportion entre les profils *Gestionnaire Fournisseurs* et *Gestionnaire produits*

4.6 Fonctions aléatoires

Utiliser des valeurs aléatoires pour les valeurs des formulaires de mise à jour d'un produit

Atelier 5 : Test de charge

Dans ce TP, nous allons exécuter toutes les étapes d'un test de charge à partir des scénarios précédemment enregistrés.

L’application a été mise à jour en particulier une ressources REST pour le monitoring est disponible : ***http://localhost:8080/actuator***

5.1 Ajout de compteur

Ajouter un ou plusieurs compteurs gaussien pour simuler le think time des utilisateurs dans la partie Web

5.2 Validation du test

La validation du test consiste :

- Validation de la séquence des requêtes en mono-utilisateur avec un tableau ou arbre de résultats
- Validation du test en modifiant le temps de pause
- Diversifier les données du formulaire d’inscription
- Validation que les URLs d'inscription inscrivent bien des nouvelles données en base
- Valider que les bons métriques sont récupérés (en particulier, on essaiera d'obtenir dans les récepteurs les calculs de temps concernant une URL et les sous-requêtes associées)

Une fois le test validé, faites une sauvegarde de votre plan de test sous un nom différent et passer à l'étape suivante

5.3 Paramétrer le test

Mettre en place des fichiers *.properties* permettant de fixer certaines valeurs pour les variables prédéfinies

5.4 Récepteurs et pour enregistrement des erreurs éventuelles

Désactiver tous les récepteurs utilisés pour la validation

Ajouter un enregistreur pour chaque groupe d’unité. Ces enregistreurs stockent toutes les informations disponibles en cas d’erreur uniquement

5.5 Exécution du test de charge et génération du rapport standard

Exécuter les tests en mode batch en augmentant progressivement le nombre d'utilisateurs et en générant le tableau de bord de performance.

Surveiller les fichiers d'erreurs ainsi que le comportement de l’application

Entre chaque test, sauvegarder les fichiers résultats.

5.6 Visualiser les différents résultats

Les seuils APDEX sont-ils toujours bons ?

5.7 Personnalisation du rapport

Personnaliser :

- Le titre du rapport
- Les seuils APDEX de certaines URLs
- Le format d’affichage de la date
- Filtrer certaines URLs

Optionnellement :

- Modifier les gabarits HTML
- Créer un graphique personnalisé qui affiche le nombre de connexions actives base de données

Atelier 6 : Tests fonctionnels

Dans ce TP, nous allons utiliser les différents éléments assertions proposés par JMeter

Les cas de tests suivants s’appliquent à la partie Web:

- TC1 : Toutes les URLs ont une taille inférieure à 100ko
- TC2 : Toutes les URLs ont bien une balise Title
- TC3 : Lorsque l'utilisateur saisit un mauvais mot de passe, il est redirigé vers la page de login avec un message d'erreur
- TC4 : Authentification réussie : Lorsque l'utilisateur saisit un bon login/mot de passe, il est redirigé vers la page d’accueil .
- TC5 : Toutes les réponses sont encodées en UTF-8
- TC6 : L’édition d’un produit contient un champ caché contenant l’ID

Les cas de tests suivants s’appliquent à la partie Rest :

- TC1 : Si le jeton de n’est pas valide, le code retour est 403
- TC2 : Lors de l’ajout d’un nouveau produit, l’id est retourné
- TC3 : Le nom du produit est correctement retourné

Créer un nouveau plan de test qui permettra de valider ces différents cas de tests.

Utiliser les bons récepteurs

Atelier 7 : Script

Dans ce TP, nous utilisons des éléments de scripting de JMeter

Test fonctionnel

Post-processeur + Exécution de script
Reprendre le test fonctionnel du TP précédent et implémenter le cas de test suivant :

- TC8 : Lors de l'ajout d'un produit, le nombre de membres est augmenté de 1

Pour obtenir ce résultat, vous pouvez utiliser des post-processeur permettant de récupérer le nombre de produits puis une ***assertion Script*** permettant que le nombre de membres a augmenté de 1

URLs aléatoires

Créer un fichier texte contenant un ensemble d'URLs.

Écrire un script Groovy mettant à jour une variable JMeter à partir d'une ligne lue au hasard dans le fichier d'URLs

Atelier 8 : Tir de charge distribué

Dans ce TP, nous allons lancer le test de charge en mode distribué avec 1 maître et 2 esclaves.

Les stagiaires pourront s'organiser en binôme.