# JEE – TP3: cookie, session, jersey

L'ensemble des ressources nécessaires au TP peut être retrouvé dans le dépôt GitHub suivant :

| https://github.com/jdassonvil/epsi |
|---|

Le code corrigé du TP se trouve quant à lui dans le dépôt suivant (mis à jour pendant le cours) :

| https://github.com/jdassonvil/epsi-j2ee-tp1 |
|---|

**La mise en œuvre de ces exercices s'appuie sur le travail effectué lors des deux premiers TP. Vous pouvez continuer depuis votre projet précédent TP ou repartir du corrigé.**

Pour reprendre depuis le corrigé voici les instructions :

- Supprimez d'Eclipse une version antérieur du projet si existant
- Mettez à jour le dépôt GIT au niveau de la head ou vers le tag 3.0
- Importez le projet : File > Import > Existing Maven Project

**3.1 Session**

- Mettez à jour la section header du site pour affichez la durée en secondes depuis laquelle l'utilisateur est connecté au site
- Implémentez la classe *LoginServlet* qui doit permettre à l'utilisateur de s'authentifier et de se déconnecter

**3.2 Cookies**

- Modifiez l'application pour rendre la session de l'utilisateur persistante en cas de fermeture du navigateur

**3.3 Jersey**

- Ajoutez la dépendance jersey-client à votre projet :
    - groupId : com.sun.jersey
    - artifactId : jersey-client
    - version : 1.19
- Implémentez l'action d'un paiement à travers un web service tierce :
    - En s'appuyant sur le client Jersey, implémentez une classe *PaymentDao* qui fera l'appel au web service de paiement (attention aux cas d'erreur)
    - Dans le package *epsi.business* implémentez la classe *PaymentService* gérant la logique de paiement
    - Mettez à jour la vue *artist* pour permettre à l'utilisateur de saisir son numéro de carte de bancaire et déclencher l'action de paiement
    - Implémentez un Servlet qui recevra la requête de paiement et la traitera en faisant appel à la classe *PaymentService*

- Informations sur l'interconnexion :
    - Hostname: fakeserver.qlf-waas.aw.atos.net
    - Requête : /sendPayment
    - Méthode: POST
    - Type: json
    - Format du contenu: { "cb": "numero de cb"}
    - Erreurs :
        - Erreur 400
            - Format de l'erreur : {"error": "message"}
    - Succès : code HTTP 200
- Exemple de requête :

```
curl -XPOST --data '{"cb": "4212"}' --header "Content-Type:
application/json" http://fakeserver.qlf-waas.aw.atos.net/sendPayment
```