

# Compte rendue : Outils pour cycle de vie Logiciel

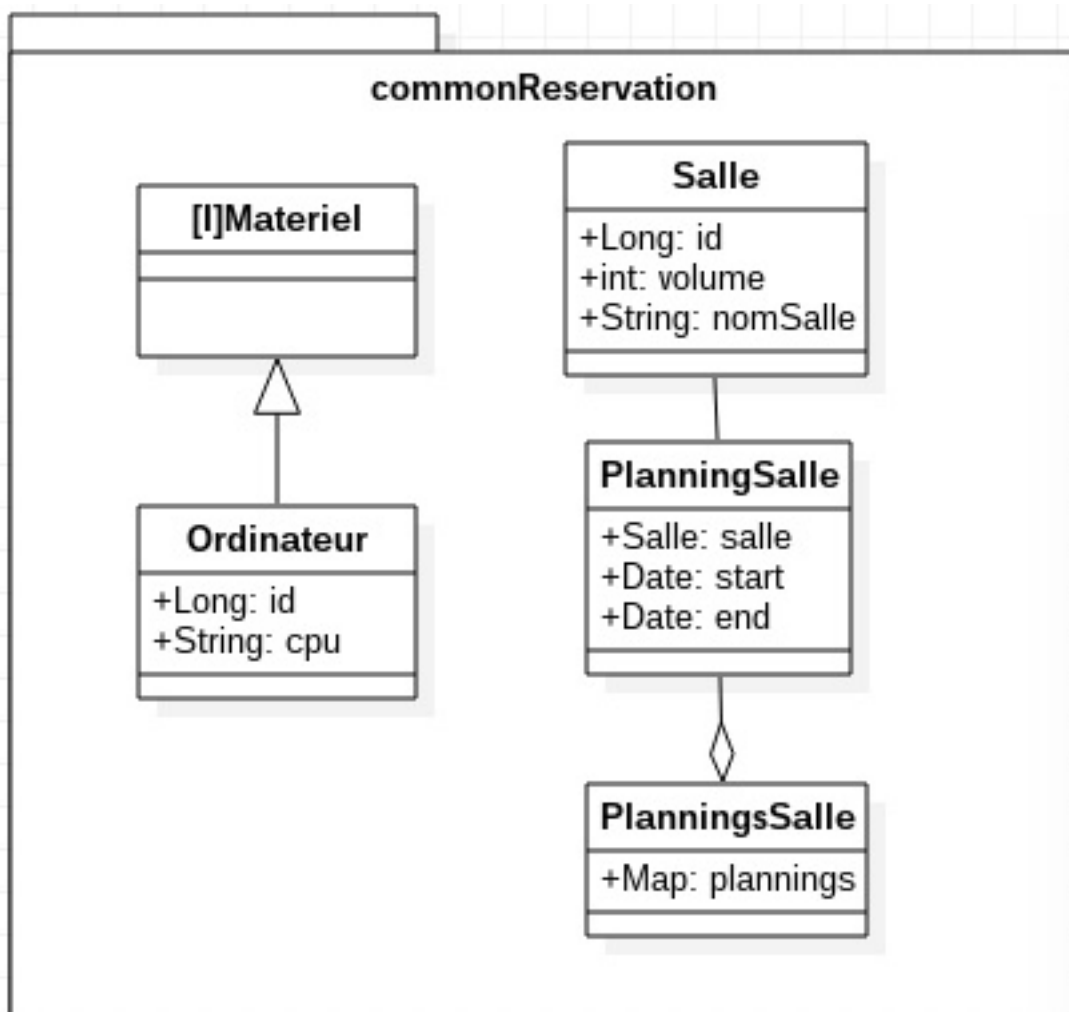
## TP4 & TP5

### Ajouter Planning

L'adresse de Git: [https://github.com/yuxinval/TPGit\\_Outil](https://github.com/yuxinval/TPGit_Outil)

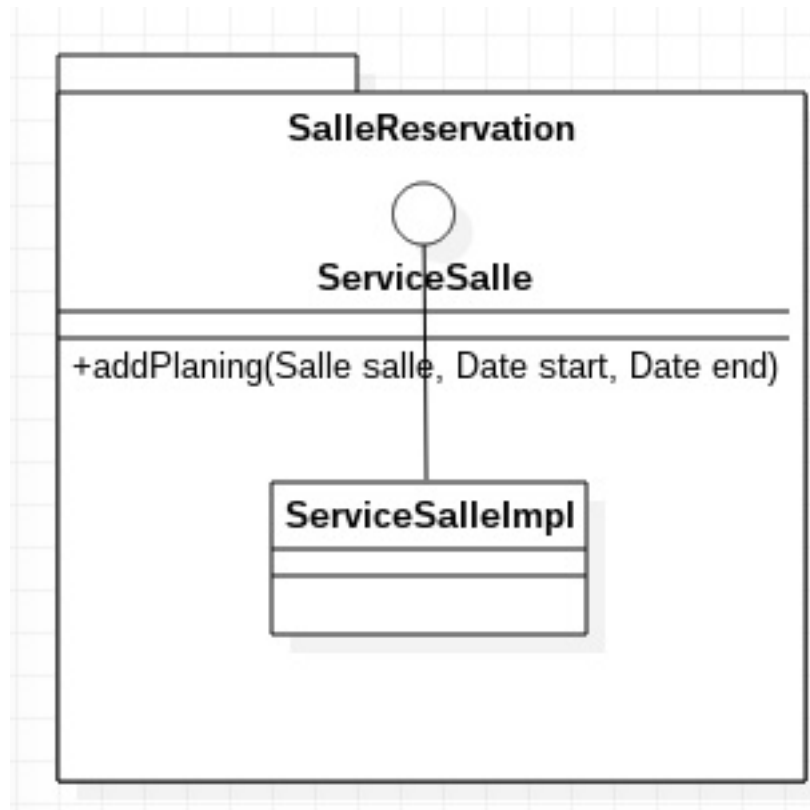
Dans le commonReservation, il y a les entités du Matériel et de la Salle, j'ai ajouté aussi deux class: Planning et Plannings.

Ceci un schémas de module commonReservation:



J'ai fait juste le planning pour la salle, PlanningSalle contient juste un planning pour une salle, et PlanningsSalle contient une collections de PlanningSalle, j'utilise Map pour faciliter la recherche, et j'utilise "Standalone" pour rassurer qu'il y a qu'une plannings pour les salles dans le processus ainsi faciliter d'ajouter les plannings sans conflit du temps pour chaque salle.

Ceci un schémas de module reservationSalle:



Il y a qu'une service selon tp, addPlaning retourne un boolean si on a réussi à ajouter un planing pour une salle.

Pour le module serveurReservation, semble qu'on doit utiliser le mode Factory pour générer les réponses des requête du client, mais je ne vois pas comment faire ça. Ce module dépend de commonReservation et reservationSalle, dans la classe ReservationFactory, ServiceSalle devrait être instancier, mais je comprend pas trop. Ou ça veut dire que c'est la partie du client?

A mon avis, c'est la classe ServiceReservation qui va instancier une service addPlaning, et le ReservationFactory fabrique juste le code de HTTP si le service a bien effectué.

Ceci le code de ReservationService:

```
@RequestMapping(method = RequestMethod.GET)

public @ResponseBody ReservationFactory addReservation(@RequestBody Planning planning) {
    ServiceSalle serviceSalle = new ServiceSalleImpl();
    boolean bool = serviceSalle.addPlanning(planning);
    return new ReservationFactory(bool);
}
```

Ceci le code de ReservationFactory:

```
public class ReservationFactory {

    public ReservationFactory(boolean bool){
        if(bool){
            success();
        } else {
            fail();
        }
    }

    public ResponseEntity<HttpStatus> success(){
        return ResponseEntity.ok(HttpStatus.OK);
    }

    public ResponseEntity<HttpStatus> fail(){
        return ResponseEntity.ok(HttpStatus.CONFLICT);
    }
}
```

## Git

---

### Usage basique

Ce projet je le fais toute seule, donc je ne peux que faire une branche et en merger avec la branche principale.

```
# create a new branch
git checkout -b yuxin

# merge with master
git merge master

# push to remote repository
git push remote yuxin
```

## Conflit

### Manuel

Normalement quand deux personne sur un même projet de Git modifient un même fichier, il y aura un conflit si on pull et merger, si on veut garder ma version, il faut supprimer l'index de conflit manuellement dans ce fichier.

## Reset et Revert

Ceci mon git log:

```
commit b8edcf790374eff1ddb68fa084ac3b2a8935e4c9
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 13:40:11 2017 +0100
    add comments
    reservationArtifact [TPGit_Outil yuxin]
    reservationMateriel [TPGit_Outil yuxin]
commit 6e97ac4e9710e4f549ac9ef067e2b5234fd29c9d
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 13:37:21 2017 +0100
    maybe factory means responseBody
    org.reservationSalle
    org.reservationSalle.service
commit 1e16cd1cfef674f1c6fcea27a19f4ef5fc0e9fc7
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 13:17:08 2017 +0100
    add factory a part
    Maven Dependencies
commit 15ff409c263599cabe05d068b65cce5070b28291
Author: yuxinShi02 <syx1026@gmail.com>
Date: Thu Nov 30 22:42:08 2017 +0100
    add a part
    pom.xml
    .mvn
```

Je voudrais retourner à commit "6e97ac" et mettre le dernier commit dans la poubelle on peut utiliser mais sans changer le fichier actuelle: `git reset --soft HEAD^`, maintenant le commit "b8edcf" n'existe plus:

```

commit 6e97ac4e9710e4f549ac9ef067e2b5234fd29c9d
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 13:37:21 2017 +0100

    maybe factory means responsebody

commit 1e16cd1cfef674f1c6fcea27a19f4ef5fc0e9fc7
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 13:17:08 2017 +0100

    add factory a part
# merge with master

commit 15ff409c263599cabe05d068b65cce5070b28291
Author: yuxinShi02 <syx1026@gmail.com>
Date: Thu Nov 30 22:42:08 2017 +0100
# push to remote repository

git add a part yuxin
...

commit 0af0bd003f2d3f8ecf40a36a3b33b4160dab13c7
Merge: da35f07 00eb6d4
Author: yuxinShi02 <syx1026@gmail.com>
Date: Wed Nov 29 21:44:26 2017 +0100

```

Si on ne veut pas perdre le dernier commit en retournant HEAD^, on peut aussi utiliser `git revert HEAD^`, git va créer un nouveau commit dans cette branche qui correspond avec HEAD^.

## Tag

Ajouter un tag pour cette version:

```

tearsyu@Dog:~/workspace/TPGit_Outil$ git tag
tearsyu@Dog:~/workspace/TPGit_Outil$ git tag -a snapV1 -m "my first version"
tearsyu@Dog:~/workspace/TPGit_Outil$ git tag
snapV1
tearsyu@Dog:~/workspace/TPGit_Outil$ git show snapV1
tag snapV1
Tagger: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 14:25:58 2017 +0100

my first version

commit 397aa346e6f2d9ee2910b6832a1f5c9cb676d62d
Merge: 0a36bae b19488a
Author: Tearsyu <syx1026@gmail.com>
Date: Tue Dec 5 14:21:14 2017 +0100

    conflit

diff --cc reservationArtifact/bin/pom.xml
index 28e1e9f,4aa140b..0000000
deleted file mode 100644,100644
+++ a/reservationArtifact/bin/pom.xml

```

annotated tags  
 Creating an annotated tag in local storage  
 Command:  
 \$ git tag -a v1.4 -m "my version"  
 \$ git tag  
 v0.1  
 v1.3  
 v1.4  
 The -m specifies a tagging message  
 annotated tag, Git launches your editor  
 You can see the tag data along with the commit data  
 \$ git show v1.4  
 tag v1.4  
 Tagger: Ben Straub <ben@straub.com>  
 Date: Sat May 3 20:19:12 2014 +0200

## Rebase

Rebase fonctionne un peu comme merge, mais il va ajouter les commits souhaités de la branche que l'on veut merger dans la branche curante. On peut l'utiliser pour notre branche personnelle pour aligner et nettoyer l'histoire de git log, mais ce n'est pas une bonne idée de l'utiliser dans une branche publique, ça peut être risqué parce que on est obligé de merger encore une fois pour notre master et celui des autres du groupe.