Faciliter la configuration/customisation du serveur CAS

- Quand une modification est-elle prise en compte ?
- Comment accélérer les tests suite à des modifications de l'overlay CAS ?
    - Désactiver le cache thymeleaf
    - Mettre les templates en dehors du WAR
    - War avec Embedded tomcat
    - Auto-déploiement des modifications dès la sauvegarde des fichiers

Quand une modification est-elle prise en compte ?

- CSS : fichiers statiques servis par tomcat. voir astuces war, auto-déploiement
- html : templates thymeleaf mis en cache à la première utilisation. voir astuces thymeleaf, war, auto-déploiement

- etc/cas/config/log4j2.xml : monitoré par CAS, pris en compte dynamiquement (`monitorInterval` 5 secondes)
- etc/cas/config/cas.properties : monitoré par CAS, plusieurs modifications prises en compte dynamiquement (réf) ou via un POST sur /actuator/refresh
- etc/cas/services/*.json : monitoré par défaut par CAS, modifications/ajouts/suppressions pris en compte dynamiquement (cf cas.service-registry.schedule + cas.service-registry.json.watcher-enabled)
- groovy files : monitoré par CAS, pris en compte dynamiquement (avec cache pour éviter la recompilation)
- message_fr.properties : pris en compte par POST sur /actuator/refresh

- java : build & restart ou parfois hotswap
- pom.xml : build & restart. Un "clean" peut-être nécessaire en cas de chgt de versions de jar

Comment accélérer les tests suite à des modifications de l'overlay CAS ?

Désactiver le cache thymeleaf

cas.properties

```
spring.thymeleaf.cache=false
```

(cf https://fawnoos.com/2021/02/16/cas63-ui-themes/ ou pas un CAS plus vieux https://apereo.github.io/2018/06/10/cas-userinterface-customizations/)

Mettre les templates en dehors du WAR

Comme expliqué sur la page https://fawnoos.com/2021/02/16/cas63-ui-themes/ , il est possible de dire à CAS de prendre les fichiers statiques en dehors du WAR, par exemple dans /etc/cas/templates.

(à valider)

War avec Embedded tomcat

NB : solution alternative aux templates en dehors du WAR, et à "bootrun" (qui marche plus ou moins bien...)

au lieu de

```
java -jar xxx/build/libs/cas.war
# ou avec maven :
java -jar xxx/target/cas.war
```

utiliser

```
gradlew unzipWAR
java -cp "xxx/build/app:xxx/build/app/WEB-INF/classes:xxx/build/app/WEB-INF/lib/*" org.springframework.boot.
loader.WarLauncher
# ou avec maven :
java -cp "xxx/target/cas:xxx/target/cas/classes:xxx/target/cas/WEB-INF/lib/*" org.springframework.boot.loader.
WarLauncher
```

NB : avec maven le répertoire `target/cas` est créé en plus du war.

Auto-déploiement des modifications dès la sauvegarde des fichiers

Voir la page générique expliquant cette problématique du développement Java