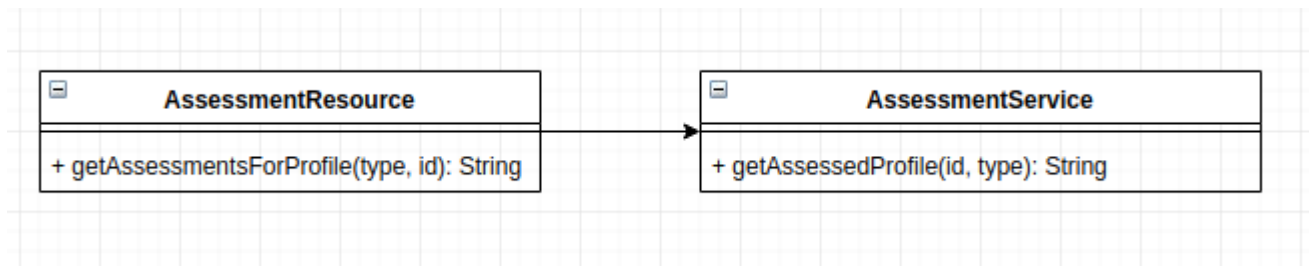


BL-277: Re-scoring profile after adjudication

Problem

Currently we get the current scoring calling a endpoint rest, but we can't know if this is the last version or if the scoring is out of sync (Scoring could be out of sync if assessmentService fail trying to get the adjudications).

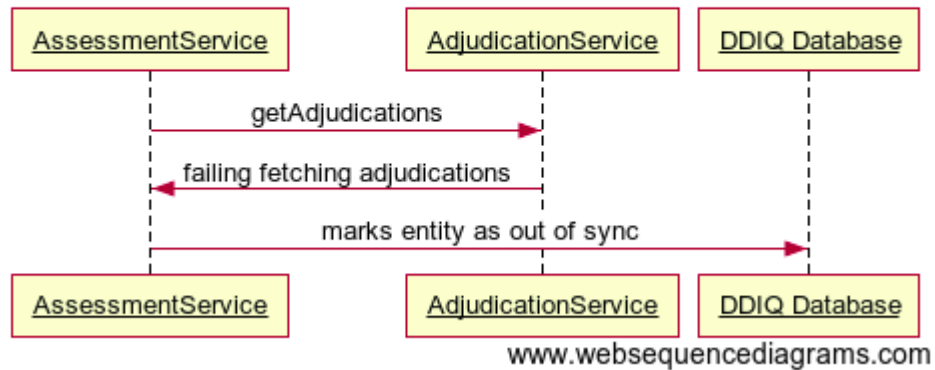


Solution

I propose a solution in two steps, first marking a entity as out of sync and in the second step when we tried to get the scoring of a entity, validate if is out of sync and updating the scoring.

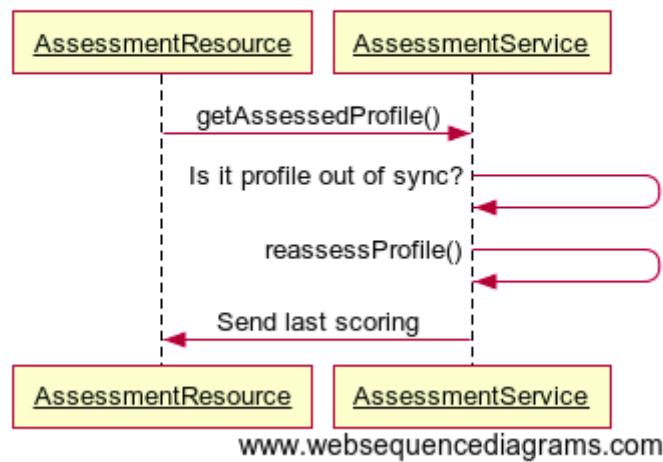
1. We need a new table to save when a re scoring fail for a entity.

Scoring sequence failed.

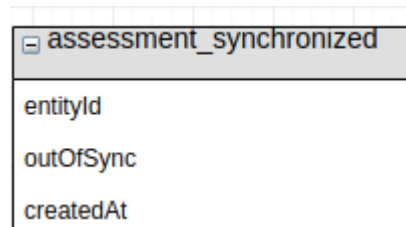


2. In this way we can validate when try the get scoring if this result is out of sync.

Scoring sequence failed.



Implementation



assessment_synchronized
entityId
outOfSync
createdAt

In the following screenshot I show the point where we are going to add the code related assessment synchronized, additionally we need add the DAO and Entities for new table.

AssessmentService:

```
private Map<String, List<Hit>> getAdjudications(Long entityId, String entityType) {  
    Map<String, List<Hit>> adjudications = null;  
    try {  
        adjudications = adjudicationService.getAdjudications(entityId, entityType);  
    } catch (AdjudicationServiceClientException e) {  
        AssessmentSynchronized entity = new AssessmentSynchronized(entityId, true);  
        assesmentSynchronizedDao.save(entity);  
        LOG.error("Error getting the adjudications", e);  
    }  
    return Optional.ofNullable(adjudications).orElse(new HashMap<>());  
}
```

ProfileScoreAssessmentWorker:

```

private Map<String, List<Hit>> getAdjudications(Long entityId, String entityType) {
    Map<String, List<Hit>> adjudications = null;
    try {
        adjudications = adjudicationService.getAdjudications(entityId, entityType);
    } catch (AdjudicationServiceClientException e) {
        AssessmentSynchronized entity = new AssessmentSynchronized(entityId, true);
        assesmentSynchronizedDao.save(entity);
        LOG.error("Error getting the adjudications", e);
    }
    return Optional.ofNullable(adjudications).orElse(new HashMap<>());
}

```

```

@Transactional(readOnly=true)
public String getAssessedProfile(Long id, EntityType type) {
    isNecessarySynchronize(id, type);
    IAdaptable entity = retrieveEntity(id, type);
    if (entity == null) {
        return null;
    }
    return assessmentJsonExporter.export(Exporter.DEFAULT_VIEW, entity.adaptTo(IAssessmentModelAware.class));
}

private void isNecessarySynchronize(Long id, EntityType entityType) {
    if(assessmentDao.isOutOfSync(id)) {
        this.reassessProfile(id, entityType.name());
        assesmentSynchronizedDao.remove(entity);
    }
}

```