OpenReviewIO

Version 1

déc. 21, 2021

Table des matières

Avertissement : OpenReviewIO est actuellement en phase **alpha** ! Il est suscepible de changer durant les prochains mois. Toute contribution est la bienvenue.

OpenReviewIO (ORIO) est un format d'échange d'informations de *review*. Il permet de garantir leur compatibilité entre les outils.

Le format est basé sur le XML.

La description de ses classes est écrite en TOML.

CHAPITRE 1

Versions

Dans l'optique de garantir une interopérabilité maximale entre les outils de *review* qui implémenteront ORIO, la compatibilité est déterminée par un numéro de version.

De cette façon, tous les outils de *review* affichant une compatibilité avec une même version pourront échanger sans difficulté leurs *reviews*.

La numérotation des versions utilise SemVer.

CHAPITRE 2

Description du standard

OpenReviewIO permet de lier des informations dites de *review* à un média.

Note : ORIO a été conçu à l'origine pour la *review* d'animation.

Une *review* contient des notes, une note est composée de *contents*.

Le format OpenReviewIO se présente sous forme d'un dossier dont le nom termine par `.orio`. Il contient un fichier `review.orio` dans lequel toutes les informations de la *review* sont écrites et de dossiers où tous les fichiers utilisés comme contenu sont stockés. Les dossiers sont nommés comme la date de la note qu'ils concernent :

```
review_name.orio
│   review.orio
└── {note.date}
│   │   image.png
│   │   other_img.jpg
```

Pour stocker les informations dans le fichier `review.orio`, le standard XML est choisi car très répandu et facilement lisible.

Note : Les *reviewers* étant rarement très nombreux à travailler en même temps sur un même média, les concurrences en écriture du fichier XML sont négligeables.

2.1 Formats de fichiers

Pour garantir une compatibilité maximale, il est nécessaire de limiter les possibilités accordées aux utilisateurs du format ORIO.

Alors qu'il est possible de lier des fichiers à une note, tous les outils de *review* ne supportent pas tous les types de fichiers. Il est donc nécessaire de spécifier les types acceptés dans les descriptions des objets à chaque nouvelle version d'ORIO.

2.2 Review

Une *review* est liée à un media.

Tous les chemins contenus dans `review.orio` doivent être relatifs si les fichiers correspondants sont placés dans le dossier `.orio` dans des sous-dossiers au nom de la date de la note concernée.

Note : Il est fortement conseillé de nommer le dossier de *review* du nom du média, extension comprise : **media_name.ext.orio**. Bien que le choix du nom soit libre.

2.3 Statut

Le statut est lié à la *review*. Il permet de connaitre l'opération suivante à effectuer sur le media.

ORIO définit un nombre limité de statuts. Les statuts autorisés sont décrits dans chaque version.

OpenReviewIO garde un historique des changements de statuts. Le dernier statut en date étant le statut actuel de la *review*.

2.4 Note

Une note ressemble conceptuellement à un email, elle a une date, un auteur et un contenu qui peut prendre plusieurs formes (texte, image…). Des notes se succèdent dans un fil thématique (la *review*) et une note peut répondre à une note précédemment créée.

Une note est par essence vide, les informations qu'elle contient sont des *contents*.

Avertissement : Dans le cas d'ORIO, il est possible de modifier le contenu d'une note, mais cela est fortement déconseillé pour des raisons pratiques de suivi de production.

2.5 Contenu

Les *contents* sont les données de la note.

Dans la conception, les *contents* suivent la philosophie de la programmation orientée objet, ils héritent d'autres *contents*. Un *content* hérite tous les paramètres de son parent et doit donc les renseigner.

La classe de référence est la classe `Content`. Cette classe est de type `Abstract`, ce qui empêche de l'utiliser telle quelle dans une note.

La définition des *contents* est stockée sous forme de fichier TOML suivant cette nomenclature : `NomAttachment.toml`.

Note : Pour une meilleure compréhension, une convention de nommage est établie pour le nom des *contents* :
- `{Name}Comment` : Un commentaire concerne tout le media.
- `{Name}Annotation` : Une annotation est relative à un fragment du média.

2.6 Classes

OpenReviewIO décrit également des classes qui lui sont propres. Leurs paramètres sont écrits au format TOML :

- Le nom d'une classe suit la convention CamelCase.
- Les membres principaux de la classe sont à la racine du document.
- Les types des membres sont écrits : `membre = 'Type'`.
  - Dans le cas d'un tableau : `membre = 'Array[Type]'`.
- Les paramètres à utiliser dans les APIs sont dans la table `[parameters]`.
  - Les paramètres optionnels sont indiqués dans une table imbriquée `[parameters.optional]`
  - Les valeurs par défaut des paramètres optionnels sont écrits dans un `array` en seconde position : `membre = [ 'Type', '',]`.

Important : Les objets sont téléchargeables ici.

CHAPITRE 3

Version 1.0

Avertissement : Encore en **alpha** !

3.1 Classes

3.1.1 Status

Classe abstraite

Statuts possibles : `approved`, `rejected`, `waiting review`. Le statut par défaut est `waiting review`.

- type : `String` = Abstract
- parameters :
  - date : `String` -> *Date de création en ISO UTC :* rfc3339
  - author : `String`-> *Auteur de la note.*
  - value : `String`-> *État du statut.*

XML

```
<status date="2020-10-13T10:38:14.173325+00:00" author="Alice">approved</status>
```

Description TOML

Contenu de `Status.toml` :

```
type = "Object"
available = [ "approved", "rejected", "waiting review",]

[parameters]
date = "String"
author = "String"
value = "String"
```

3.1.2 Review

Classe abstraite

- type : `String` = Abstract
- parameters :
  - media_path : `String` -> *Chemin vers le média. Les séquences d'image sont décrites avec un "#" par digit (image_seq-###.exr).*
  - media_signature : `String` -> *Signature MD5 du media pour garantir son authenticité.*
  - status_history : `Dict{Dict}` -> *Tous les changements statuts, le dernier de la liste étant le statut courant.*
  - notes : `Dict{Dict}` -> *Notes associées à la review. La date de création est utilisée pour l'identification.*
  - optional :
  - metadata : *(optionel)* `Dict{AnyType}` = { } -> *Permet de stocker toute donnée supplémentaire relative à la review.*

XML

```
<review media_path="/path/to/other_media.ext" media_signature=
↪"d41d8cd98f00b204e9800998ecf8427e">
  <metadata key="value"/>
  <statuses>
    ...
  </statuses>
  <notes>
    ...
  </notes>
</review>
```

Description TOML

Contenu de `Review.toml` :

```
type = "Object"

[parameters]
media_path = "String"
status_history = "Array[Status]"
notes = "Array[Note]"

[parameters.optional]
metadata = [ "Dict", "None",]
```

3.1.3 Note

Classe abstraite

- type : `String` = Abstract
- parameters :
  - date : `String` -> *Date de création en ISO UTC :* rfc3339
  - author : `String` -> *Auteur de la note.*
  - optional :
  - parent : *(optionel)* `String` = "" -> *Date (id) de la note à laquelle cette note est une réponse.*
  - metadata : *(optionel)* `Dict{AnyType}` = { } -> *Permet de stocker toute donnée supplémentaire relative à la review.*

XML

```xml
<note date="2020-10-13T10:38:14.173325+00:00" author="Michel" parent="">
  <contents>
    ...
  </contents>
  <metadata key="value"/>
</note>
```

Description TOML

Contenu de `Note.toml` :

```toml
type = "Object"

[parameters]
date = "String"
author = "String"
value = "String"

[parameters.optional]
parent = [ "String", "",]
metadata = [ "Dict", "None",]
```

3.1.4 Contents

3.1.4.1 TextComment

Commentaire texte concernant tout le media.

- type : `String` = Content. -> *Type de la classe dont sont héritées les propriétés.*
- parameters :
  - body : `String` -> *Contenu textuel du commentaire.*

XML

```xml
<TextComment body="Text content body"/>
```

Description TOML

Contenu de `TextComment.toml` :

```toml
type = "Content"

[parameters]
body = "String"
```

3.1.4.2 TextAnnotation

Commentaire texte relatif à un fragment temporel d'un média. Composé d'une image de départ et d'une durée.

- type : `String` = *TextComment*. -> *Type de la classe dont sont héritées les propriétés.*
- parameters :
  - frame : `Integer` = 0 -> *Image à laquelle l'annotation commence.*
  - duration : `Integer` = 1 -> *Durée de l'annotation en nombre d'images.*

XML

```xml
<TextAnnotation body="Text annotation body" duration="7" frame="10"/>
```

Description TOML

Contenu de `TextAnnotation.toml` :

```
type = "TextComment"

[parameters]
frame = "Integer"
duration = "Integer"
```

3.1.4.3 Image

Classe abstraite. Les fichiers image acceptés sont les fichiers aux extensions JPEG, JPG et PNG.

- type : `String` = Abstract
- mime : `Array[String]` -> *MIME_ autorisé pour les images : JPG, JPEG or PNG.*
- parameters :
  - path_to_image : `String` -> *Chemin vers le fichier image.*

XML

```
<Image path_to_image="/path/to/image.png"/>
```

Description TOML

Contenu de `Image.toml` :

```
type = "Content"
mime = [ "jpeg", "jpg", "png",]

[parameters]
path_to_image = "String"
```

3.1.4.4 ImageAnnotation

Annotation par image concernant une partie du media.

- type : `String` = Content. -> *Type de la classe dont sont héritées les propriétés.*
- paramètres :
  - image : `Image`
  - frame : `Integer` = 0 -> *Image à laquelle l'annotation commence.*
  - duration : `Integer` = 1 -> *Durée de l'annotation en nombre d'images.*
  - optional :
  - reference_image : *(optionel)* `String` = "" -> *Chemin vers le fichier image utilisé comme référence de l'annotation.*

XML

```
<ImageAnnotation duration="10" frame="3" path_to_image="/path/to/image_annotation.png
↪" reference_image="/path/to/reference_image.png"/>
```

Description TOML

Contenu de `ImageAnnotation.toml` :

```
type = "Image"

[parameters]
frame = "Integer"
```

(suite sur la page suivante)

(suite de la page précédente)

```
duration = "Integer"

[parameters.optional]
reference_image = [ "Image", "",]
```

3.1.5 Fichier de démonstration

```
<?xml version='1.0' encoding='UTF-8'?>
<review media_path="/path/to/other_media.ext" media_signature=
↪"d41d8cd98f00b204e9800998ecf8427e">
  <metadata/>
  <statuses>
    <status date="2020-10-13T10:38:14.173325+00:00" author="Michel">rejected</status>
    <status date="2020-10-13T10:38:14.176828+00:00" author="Alice">approved</status>
  </statuses>
  <notes>
    <note date="2020-10-13T10:38:14.173325+00:00" author="Michel" parent="">
      <contents>
        <TextComment body="Text content body"/>
        <TextAnnotation body="Text annotation body" duration="7" frame="10"/>
        <Image path_to_image="/path/to/image.png"/>
        <ImageAnnotation duration="10" frame="3" path_to_image="/path/to/image_
↪annotation.png" reference_image="/path/to/reference_image.png"/>
      </contents>
      <metadata key="value"/>
    </note>
  </notes>
</review>
```

CHAPITRE 4

Intégrations d'OpenReviewIO

4.1 API

— OpenReviewIO Python Module

4.2 Logiciels

Important : Dernière version en date : *Version 1.0*.

Les objets sont téléchargeables ici.

CHAPITRE 5

Remerciements

Mille mercis à Jean-François Sarazin et Élie Michel pour leur aide et leur relecture.