DÉPLOIEMENT DES SERVICES DU MOTEUR DE RENDU DE PARTITIONS GUIDO SUR INTERNET

M. Solomon, D. Fober, Y. Orlarey, S. Letz
Grame
Centre national de création musicale
mike@mikesolomon.org - {fober, orlarey, letz}@grame.fr

Résumé

La librairie Guido embarque un moteur de rendu de partitions musicales et fournit une API de haut niveau pour un ensemble de services liés au rendu de partition. Cette librairie est désormais embarquée dans un serveur HTTP qui permet d'accéder à son interface par le biais d'identifiants de ressource uniformes (URI). Après avoir décrit le style d'architecture *representational state transfer* (REST) sur lequel repose le serveur, l'article montre comment l'API C/C++ de la librairie est rendue accessible sous forme de requêtes HTTP.

1. L'ÉDITION MUSICALE SUR INTERNET

Pendant les dix dernières années, plusieurs outils de gravure musicale ont été développés selon un modèle client-serveur, à destination d'utilisateurs finaux et d'usagers du *cloud computing*. Le serveur GUIDO allie la gravure musicale sur Internet aux principes de l'architecture REST (élaborés dans la section 2) afin d'exposer l'API de la librairie Guido [8] [10] [4] par le biais d'URIs. Après un survol des outils d'édition musicale en ligne, nous dégagerons les thématiques sur lesquelles se basent les services actuellement disponibles et nous situerons le serveur Guido dans le paysage de la gravure musicale sur Internet.

1.1. Editeurs de musique sur Internet

Il y a actuellement trois outils majeurs d'édition musicale sur le web – Noteflight [1], Melodus [2] et Score [3]. Noteflight et Melodus fournissent un environment immersif d'édition musicale sur Internet comparable à Finale ou Sibelius. Scorio est un outil hybride qui utilise des algorithmes de mise en page élémentaires pour sa plate-forme mobile et propose le téléchargement des documents de haute qualité compilés avec GNU LilyPond.

1. http://www.noteflight.com
2. http://www.melod.us
3. https://scorio.com

1.2. Outils de partage de partitions sur Internet

Plusieurs outils musicaux dont Sibelius [4], MuseScore [1], Maestro [5] et Capriccio [6], proposent des services de partage sur Internet des partitions créées à partir de ces logiciels. Ces services permettent le téléchargement, la lecture et parfois le rendu sonore de ces partitions. Capriccio propose une application web écrite en Java qui reproduit les aspects essentiels de son éditeur principal alors que MuseScore, Sibelius et Maestro permettent la synchronisation automatique des partitions avec leurs représentations MIDI.

1.3. Services de compilation JIT sur Internet

WebLily [7], LilyBin [8] et OMET [9] proposent tous la compilation JIT de partitions LilyPond qui sont visualisées en SVG, PDF et/ou canvas HTML 5, selon l'outil. Le "*Guido Note Server*" [11] utilise le moteur Guido pour compiler des documents au format GMN (Guido Music Notation Format) [9] et calculer des images PNG.

1.4. Une alternative RESTful

Tous les outils exposés ci-dessous permettent la création et la visualisation de partitions par le biais de différentes méthodes d'entrée (représentation textuelle, MusicXML, etc.). En revanche, ils ne sont pas adaptés à des échanges d'information dynamiques entre un serveur et un client, du fait qu'ils ne proposent pas d'API publique et ne fournissent pas d'information au-delà d'une représentation graphique de la partition. Le serveur Guido résout ce problème en fournissant une API HTTP, qui agit comme une passerelle entre le client HTTP et l'API C/C++ de la librairie. Le serveur fournit donc aussi bien des représentations graphiques de la partition qu'un ensemble d'information liées à la partition : nombre pages, durée, répartition des éléments musicaux sur la page

4. http://www.sibelius.com
5. http://www.musicaleditor.com
6. http://cdefgabc.com
7. http://weblily.net
8. http://www.lilybin.com
9. http://www.omet.ca

et dans le temps, représentation MIDI... Le serveur Guido comble donc une lacune dans le paysage des outils de gravure musicale sur Internet en fournissnt à la fois des services de rendu de partition et en permettant le déploiement d'applications basées sur ces services.

2. REPRESENTATIONAL STATE TRANSFER

Le serveur Guido est basé sur une architecture de type REST (*REpresentational State Transfer*) [5]. Ce type d'architecture est largement utilisé pour l'élaboration de services Internet et repose sur un modèle client-serveur classique, « orienté resource », c'est-à-dire que le fonctionnement du serveur est optimisé pour la transmission d'informations relatives à des *ressources* [12]. Une des caractéristiques essentielles d'un serveur REST repose sur un fonctionnement *sans état* : toutes les informations requises pour traiter une requête se trouvent dans la requête elle même. Cela signifie, par exemple, qu'une ressource sur le serveur ne peut jamais être modifiée par un client – une modification équivaut au remplacement d'une ressource par une autre.

Plusieurs conventions sont proposés pour structurer les requêtes en forme d'URI afin qu'elles soient plus faciles à construire et décoder. Pour accélérer les interactions entre le serveur et le client, l'architecture REST permet la mise en cache de certains éléments, tant du côté du client que serveur. Le serveur doit proposer une interface uniforme en harmonisant sa syntaxe d'accès et en proposant les mêmes services à tous les clients qui l'utilisent.

Un serveur qui met en œuvre les recommandations REST s'appelle un serveur RESTful.

3. LE SERVEUR GUIDO HTTP

Le serveur Guido est un serveur RESTful qui compile des documents au format GMN (Guido Music Notation) et renvoie différentes représentations de la partition calculée à partir du code GMN. Il accepte des requêtes par le biais de deux méthodes du protocole HTTP : POST pour envoyer des partitions au format GMN au serveur, et GET pour interroger ces partitions ou pour en demander différentes représentations.

3.1. La méthode POST

L'implémentation de POST dans le serveur Guido est RESTful dans la mesure où il ne garde pas d'information par rapport aux clients et ne stocke que les documents GMN qu'on lui envoie. Supposons qu'un serveur Guido HTTP fonctionne sur le site `http://guido.grame.fr` sur le port 8000, une requête POST avec le code GMN `[a b c d]` pourrait être envoyée de la manière suivante :

```
curl -d"data=[a b c d]" http://guido.grame.fr:8000
```

Pour un code GMN valide, une réponse est renvoyée au format JSON [3], avec un identifiant unique qui correspond au code GMN.

```
{
  "ID": "07a21ccbfe7fe453462fee9a86bc806c8950423f"
}
```

Cet identifiant est généré via l'algorithme de hachage cryptographique SHA-1 [6] qui transforme des documents numérisés en clé de 160 bits. La probabilité de collision de deux clés est très faible ($\frac{1}{2^{63}}$), et l'on peut donc considérer que cette clé constitue un identificateur unique. Ce type d'identification est, par exemple, utilisé par Git [2], le système de gestion de versions distribué.

La clé SHA-1 est la représentation interne du code GMN qui est utilisée pour toutes les requêtes ultérieure. Dans l'exemple suivant, on accède à une partition en faisant référence à sa clé SHA-1. Cette clé sera désormais abrégée en `<key>` pour faciliter la lecture.

`http://guido.grame.fr:8000/<key>`

Sans autre information dans l'URL, l'objet de retour par défaut est illustré par la Figure 1.

Figure 1. Score with SHA-1 tag `<key>`.

Pour un fonctionnement totalement RESTful, le code GMN devrait être fournit avec toute requête le concernant. Dans la pratique, on peut considérer que la clé SHA-1 remplace ce code sans effet de bord. Son utilisation minimise la déviation du style RESTful tout en proposant un point d'accès unique à la partition.

3.2. La méthode GET

Une requête GET permet d'obtenir des informations sur une partition identifiée sur le serveur par une clé SHA-1 ou différentes représentations de cette partition. Selon la requête, les données renvoyées par le serveur sont de type :

- image (jpeg, png ou svg) pour une représentation graphique de la partition,
- MIDI pour une représentation MIDI de la partition,
- JSON pour toute autre demande d'information.

4. L'API HTTP DE LA LIBRAIRIE GUIDO

Le serveur Guido expose l'API de la librairie Guido en créant des équivalences une à une entre l'API HTTP et les fonctions C/C++ correspondantes. Les arguments des fonctions sont passés par le biais de paires *clé-valeur* dans la partie *query* de l'URI transmise au serveur. Une présentation complète de l'API se trouve dans la documentation du serveur [7].

Cette section décrit les conventions utilisées pour passer de l'API C/C++ à l'API HTTP. Elle est suivie de plusieurs exemples concrets d'interactions avec le serveur.

4.1. Clé SHA-1 comme représentation de partition

La section 3.1 a décrit comment la clé SHA-1 remplace du code GMN dans les URIs envoyés au serveur. Cette clé peut-être vue comme l'équivalent d'un pointeur sur une partition, en l'occurrence un *handler* sur une représentation abstraite (GAR) de la partition pour la librairie Guido.

4.2. Fonction comme segment d'URI

Un segment d'URI dans l'API HTTP correspond à une fonction dans l'API C/C++. Par exemple, la fonction `GuidoGetPageCount` dans l'API C/C++ correspond au segment `pagescount` dans une URI.

Les fonctions de l'API Guido peuvent être classées en deux catégories :

- fonctions qui fournissent de l'information sur la librairie.
- fonctions qui fournissent de l'information sur une partition.

Pour l'API C/C++ de la librairie, les fonctions qui s'adressent à une partition prennent comme argument une *handler* sur la partition, qui peut être vu comme un pointeur sur la représentation interne de la partition. Avec l'API HTTP, la clé SHA-1 remplace ce *handler* dans l'URI et permet de définir la partition cible de la requête :

- les requêtes adressées à une partition sont préfixées par la clé SHA-1,
- les requêtes adressées au moteur de rendu ne sont pas préfixées.

Par exemple,

```
http://guido.grame.fr:8000/version
```

renvoie la version de la librairie.
Par ailleurs, l'URI :

```
http://guido.grame.fr:8000/<key>/voicescount
       ou <key> est la clé SHA-1
```

expose la fonction `GuidoCountVoices` par le biais du segment `voicescount`, adressé à la partition identifiée par `<key>`, et renvoie le nombre de voix de la partition.

La table 1 présente une liste de fonctions de l'API Guido et les segments d'URI correspondants dans les requêtes au serveur.

4.3. Arguments comme paires clé-valeur

Pour les fonctions qui prennent des arguments, ceux-ci sont déclinés en paires clé-valeur dans l'URI. Par exemple, la fonction `GuidoGetStaffMap` prend un argument `staff` qui permet de préciser la *staff* (portée) cible de la requête.

C/C++ API	URI segment	cible
GuidoGetPageCount	pagescount	partition
GuidoGetVoiceCount	voicescount	partition
GuidoDuration	duration	partition
GuidoFindPageAt	pageat	partition
GuidoGetPageDate	pagedate	partition
GuidoGetPageMap	pagemap	partition
GuidoGetSystemMap	systemmap	partition
GuidoGetStaffMap	staffmap	partition
GuidoGetVoiceMap	voicemap	partition
GuidoGetTimeMap	timemap	partition
GuidoAR2MIDIFile	midi	partition
GuidoGetVersionStr	version	moteur
GuidoGetLineSpace	linespace	moteur

Table 1. Représentation des fonctions de l'API Guido comme segments d'URI.

```
http://guido.grame.fr:8000/<key>/staffmap?staff=1
```

Pour permettre aux URIs incomplets ou mal-formés d'être traités, tous les arguments passés aux fonctions ont des valeurs par défaut pour le serveur. Ces valeurs sont parfois indiquées dans l'API et sinon correspondent à des valeurs qui auraient du sens pour la plupart de partitions.

4.4. Options de mise en page et formatage

Le serveur Guido permet le contrôle de la mise en page et le formattage de la partition grâce à des paires clé-valeur indiqués de manière similaire aux arguments des fonctions décris ci-dessus. Ces paramètres sont utilisés de différentes manières selon leur fonction dans l'API C/C++ Guido . Certains paramètres, comme `topmargin`, font partie d'une structure `GuidoPageFormat` qui est utilisée pour contrôler la taille globale d'une page. D'autres, comme `resize`, déclenchent un appel à une fonction. Le paramètre `width` est utilisé plusieurs fois dans le processus de compilation selon le format de sortie choisi.

Toutes les options de mise en page et de formatage sont spécifiées dans l'URI, afin de fournir toutes les informations nécessaires au rendu de la partition sans gérer d'états et de satisfaire ainsi aux recommendations RESTful.

4.5. Valeurs de retour

Les paquets renvoyés par le serveur sont constitués d'une enveloppe et d'un contenu. L'enveloppe indique un code de retour et un type MIME pour le contenu associé. Les types renvoyés sont parmi :

- `image/[jpeg | png | svg+xml]` pour une représentation graphique de la partition au format jpeg, png ou svg,
- `audio/midi` pour une représentation MIDI de la partition,

– `application/json` pour des informations structurées sur la partition.

JSON est utilisé de manière consistante pour toutes les demandes informations. Pour les fonctions qui renvoient des structures de données, les données renvoyées par le serveur sont typées et structurées de manière équivalente en JSON. Par exemple, la structure `Time2GraphicMap` contient une liste de paires où chaque paire contient la durée d'un événement (`TimeSegment`) et les coordonnées du rectangle englobant l'événement dans la la page (`FloatRect`). `TimeSegment` et `FloatRect` sont elles mêmes des structures qui contiennent d'autres données. Ces structures peuvent être articulées comme une hiérarchie de dictionnaires JSON, comme dans l'exemple fourni dans la section 4.6.3 où `time` correspond à un `TimeSegment` et `graph` correspond à un `FloatRect`.

4.6. Exemples

4.6.1. voicescount

La requête `voicescount` demande le nombre de voix dans une partition. Elle correspond à la fonction `GuidoCountVoices` de l'API Guido . L'URI :

```
http://guido.grame.fr:8000/<key>/voicescount
```

produit la réponse suivante du serveur :

```
{
  "<key>": {
    "voicescount": 1
  }
}
```

où <key> est la clé SHA-1.

4.6.2. pageat

La requête `pageat` demande la page contenant la date passée en argument, où « date » correspond à une date de la partition. Elle correspond à la fonction `GuidoFindPageAt` de l'API Guido . L'URI :

```
http://guido.grame.fr:8000/<key>/pageat?date=1/4
```

produit la réponse suivante du serveur :

```
{
  "<key>": {
    "page": 1,
    "date": "1/4"
  }
}
```

4.6.3. staffmap

La requête `staffmap` demande la description des relations entre espace graphique et temporel de la partition. Elle renvoie une liste de paires associant un espace graphique décrit comme un rectangle, et un espace temporel décrit comme un intervalle borné par deux dates. Les dates sont indiquées par des rationels exprimant du temps musical (i.e. relatif à un tempo) où 1 représente la ronde. La requête correspond à la fonction `GuidoGetStaffMap` de l'API Guido.
L'URI :

```
http://guido.grame.fr:8000/<key>/staffmap?staff=1
```

produit la réponse suivante du serveur (abrégée pour des raison de commodité) :

```
{
  "<key>": {
    "staffmap": [
      {
        "graph": {
          "left": 916.18,
          "top": 497.803,
          "right": 1323.23,
          "bottom": 838.64
        },
        "time": {
          "start": "0/1",
          "end": "1/4"
        }
      },
      .
      .
      .
      {
        "graph": {
          "left": 2137.33,
          "top": 497.803,
          "right": 2595.51,
          "bottom": 838.64
        },
        "time": {
          "start": "3/4",
          "end": "1/1"
        }
      }
    ]
  }
}
```

5. CONCLUSION

Le serveur Guido repose sur une architecture architecture RESTful afin d'exposer l'API C/C++ de la librairie Guido à travers une API HTTP. La spécification cette API permet de traiter les requêtes sans gérer d'états successifs, dans la mesure où toutes les informations nécessaires à une requête sont contenues dans son URI. La disponibilité de ce service sur Internet ouvre de nouvelles perspectives dans le développement d'applications Web qui souhaitent incorporer la notation musicale symbolique, que ce soit pour visualiser des partitions ou encore pour exploiter des données sur ces partitions.

Le projet Guido est un projet *open source* hébergé sur sourceforge (`http://guidolib.sf.net`). Le serveur est actuellement accessible à l'adresse `http://guidoservice.grame.fr/`.

Références

[1] T. Bonte. MuseScore : Open source music notation and composition software. Technical report, Free and Open source Software Developers' European Meeting, 2009. http://www.slideshare.net/thomasbonte/musescore-at-fosdem-2009.

[2] Scott Chacon. *Pro Git*. Books for professionals by professionals. Apress, 2009.

[3] D. Crockford. The json data interchange format. Technical report, ECMA International, October 2013.

[4] C. Daudin, D. Fober, S. Letz, and Y. Orlarey. The Guido Engine - a toolbox for music scores rendering. In *Proceedings of the Linux Audio Conference 2009*, pages 105–111, 2009.

[5] R. Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[6] P. Gallagher. Secure hash standard (shs). Technical report, National Institute of Standards and Technology, March 2012.

[7] Grame. *Guido Engine Web API Documentation v.0.50*, 2014.

[8] Grame. *GuidoLib v.1.52*, 2014.

[9] H. Hoos, K. Hamel, K. Renz, and J. Kilian. The GUIDO Music Notation Format - a Novel Approach for Adequately Representing Score-level Music. In *Proceedings of the International Computer Music Conference*, pages 451–454. ICMA, 1998.

[10] H. H. Hoos and K. A. Hamel. The GUIDO Music Notation Format Specification - version 1.0, part 1 : Basic GUIDO. Technical report TI 20/97, Technische Universitat Darmstadt, 1997.

[11] Renz K. and H. Hoos. A Web-based Approach to Music Notation Using GUIDO. In *Proceedings of the International Computer Music Conference*, pages 455–458. ICMA, 1998.

[12] L. Richardson and S. Ruby. *RESTful Web Services*. O'Reilly Media, 2008.