Article
Sarah Schlegel · Sept 6, 2022 8m de lecture

Présentation des webservices REST JSON

Bonjour la communauté !

Cet article vise à donner un aperçu des webservices REST JSON développés pour TrakCare.

Ces webservices ont été développés dans le but de permettre aux utilisateurs d ' accéder aux données de TrakCare depuis l ' extérieur, notamment via des applications externes.

Ils sont développés en REST avec ObjectScript, et permettent d ' accéder aux données via quatre modes :

- **List** : qui permet de récupérer un ensemble de données à partir d ' un filtre (par exemple la liste d ' épisodes pour un certain patient) ;
- **Open** : qui renvoie une donnée spécifique récupérée via son ID interne ou une clé unique (par exemple le numéro d ' épisode pour un épisode donné) ;
- **Save** : qui enregistre une nouvelle entrée dans la table concernée et retourne son ID interne ;
- **Update** : qui met à jour une entrée existante à partir de son ID.

Accès aux webservices

L ' URL d ' appel générique de l ' application des webservices est :

```
http://<adresse IP ou alias du serveur TrakCare>:57772/trakRestWS/tcrest/call
```

On passe en paramètre un JSON contenant les données d ' authentification de l ' utilisateur, les informations des webservices concernés (classe et méthode) à appeler, ainsi que les paramètres d ' appel de la méthode. Ces paramètres sont, la plupart du temps, les suivants :

- pour les webservices List, le numéro d ' épisode ;
- pour les webservices Open, l ' ID interne de l ' élément concerné ;
- pour les webservices Save, les différents champs obligatoires à la sauvegarde d ' un élément, ainsi qu ' éventuellement les contexte et action ACN associés, et les éventuels paramètres facultatifs ;
- pour les webservices Update, l ' ID interne de l ' élément à modifier, ainsi que les champs à mettre à jour.

La structure du JSON associé est donc la suivante :

```
{
    "ClassName": "<Nom complet de la classe>",
    "MethodName":
"<Nom de la méthode, généralement List, Open, Save, SaveWithACN ou Update>",
    "AppName": "<Nom de l'application qui appelle le webservice>",
    "Request": {
        "UserName": "<Nom d'utilisateur TrakCare>",
        "Password": "<Mot de passe encodé>",
```

```
        "Encoder": "Base64",
        "Params": {
            <Paramètres>
        }
    }
}
```

Le paramètre Encoder permet de définir l ' encodage du mot de passe, ce qui évite d ' envoyer le mot de passe de l ' utilisateur en clair dans le JSON. Si le paramètre Encoder n ' est pas passé, le paramètre Password prendra en entrée le mot de passe de l ' utilisateur non encodé.

Filtres

Les webservices List et Open retournent bien souvent des blocs de données volumineux, qui peuvent causer des ralentissements pour l ' application cliente. C ' est pourquoi un système de filtrage des champs retournés est possible si l ' on ajoute un champ additionnel dans le JSON.

Il est possible de filtrer en ne récupérant que les champs concernés avec le paramètre **FilterInclude**, ou en excluant les champs non-désirés avec **FilterExclude**. Le JSON précédemment montré peut donc prendre l ' une des deux formes suivantes :

```
{
    "ClassName": "<Nom complet de la classe>",
    "MethodName":
"<Nom de la méthode, généralement List, Open, Save, SaveWithACN ou Update>",
    "AppName": "<Nom de l'application qui appelle le webservice>",
    "Request": {
        "UserName": "<Nom d'utilisateur TrakCare>",
        "Password": "<Mot de passe encodé>",
        "Encoder": "Base64",
        "Params": {
            <Paramètres>
        },
        "FilterInclude": "<Champs à inclure séparés par une virgule>"
    }
}
```

OU

```
{
    "ClassName": "<Nom complet de la classe>",
    "MethodName":
"<Nom de la méthode, généralement List, Open, Save, SaveWithACN ou Update>",
    "AppName": "<Nom de l'application qui appelle le webservice>",
    "Request": {
        "UserName": "<Nom d'utilisateur TrakCare>",
        "Password": "<Mot de passe encodé>",
        "Encoder": "Base64",
        "Params": {
            <Paramètres>
        },
        "FilterExclude": "<Champs à exclure séparés par une virgule>"
```

```
    }
}
```

Code tables / Thesauri

Certaines données de TrakCare sont stockées dans des code tables, ou thesauri. Ce sont des dictionnaires de données qui sont référencés par d ’ autres tables de TrakCare via leur code ou description, et sont donc nécessaires pour le bon fonctionnement des webservices.

Liste des code tables

La liste des code tables accessibles à l ’ utilisateur peut être récupérée grâce au JSON suivant :

```
{
    "ClassName": "Region.FRXX.WebServices.REST.CodeTables.GenericCodeTables",
    "MethodName": "List",
    "AppName": "<Nom de l’application qui appelle le webservice>",
    "Request": {
        "UserName": "<Nom d’utilisateur TrakCare>",
        "Password": "<Mot de passe>",
        "Params": {
            "CT": "*"
        }
    }
}
```

Et le JSON de retour contiendra la liste des thesauri auxquels il est possible d ’ accéder.

Contenu d ’ une code table

Pour accéder au contenu d ’ une code table spécifique, on remplace le contenu du paramètre CT dans le JSON par le nom de la code table à laquelle on souhaite accéder. Deux paramètres additionnels, **Filter** et **FilterActif**, permettent de spécifier la requête.

Filter

Le paramètre **Filter** permet de filtrer sur le code et la description des éléments contenus dans la code table. Il se base sur un système de pattern matching semblable au SQL, en utilisant % pour remplacer les caractères inconnus.

Le filtrage peut s'effectuer des manières suivantes :

- Filtrage strict avec une chaîne de caractères exacte : "Bilatéral" ne renverra que les éléments dont le code ou la description contient exactement la chaîne de caractères "Bilatéral", indépendant de la casse.
- Filtrage souple à l'aide du Wildcards % : le % définit une chaîne de caractères inconnue. Ainsi,
    - "%al" renverra tous les éléments dont le code ou la description se termine par "al" (ou "AL" ou "Al" ou "aL")
    - "Bi%" renverra tous les éléments dont le code ou la description commence par "Bi" (ou "BI" ou "bi" ou "bI").
    - "%la%" renverra tous les éléments dont le code ou la description contient la chaîne de caractères

"al" quelque part (indépendamment de la casse).
    - dans ce cas, les % peuvent être combinés pour récupérer des éléments qui contiennent plusieurs chaînes de caractères, par exemple "%bi%al%"
  - "%" renverra tous les éléments. Attention cependant, certaines code tables peuvent contenir un grand nombre de données susceptible de faire planter une application cliente.

Si le paramètre Filter n'est pas envoyé ou vide, le webservice renvoie par défaut une liste vide.

FilterActif

Par défaut, le webservice ne renvoie que les éléments actifs du thésaurus. Pour filtrer sur les éléments actifs et inactifs, le paramètre **FilterActif** avec la valeur "ALL" doit être ajouté.

Un exemple complet de requête pour récupérer la liste des professionnels de santé (thésaurus CTCareProv) est donc :

```
{
    "ClassName": "Region.FRXX.WebServices.REST.CodeTables.GenericCodeTables",
    "MethodName": "List",
    "AppName": "Postman",
    "Request": {
        "UserName": "userWS",
        "Password": "****",
        "Params": {
            "CT": "CTCareProv",
            "Filter": "%",
            "FilterActif": "ALL"
        }
    }
}
```

ACN

La méthode Save est une méthode générique disponible pour tous les webservices de sauvegarde d ' une nouvelle entrée. Néanmoins, un Save ne sauvegarde l ' entrée que dans la base de données TrakCare, et pas dans le DPI.

Pour sauvegarder l ' entrée dans la base de données et le DPI, il faut utiliser la méthode abstraite **SaveWithACN**, qui est implémentée par tous les webservices Save. Cette méthode implique d ' envoyer dans les params deux paramètres supplémentaires : **ACNContext** et **ACNAction**. Il faut aussi que le numéro d ' épisode (**Episode**) soit présent, même s ' il n ' est pas requis pour l ' opération de sauvegarde normale.

La liste des contextes est disponible dans le thésaurus **MRCEncEntryType**. Ensuite, il est possible de récupérer l ' action associée à une classe donnée en interrogeant le webservice générique Save et sa méthode **GetActions**.

Exemples finaux

Prenons l ' exemple d ' une l ' utilisation des webservices sur le antécédents familiaux. Une requête pour lister les antécédents familiaux d ' un patients aura la forme suivante :

```
{
```

```
    "ClassName": "Region.FRXX.WebServices.REST.PAFamily.List",
    "MethodName": "List",
    "AppName": "SoapUI",
    "Request": {
        "UserName": "userWS",
        "Password": "******",
        "Params": {
            "IPP": "21002205"
        }
    }
}
```

Pour récupérer ensuite l ' un de ces antécédents familiaux spécifiques, on aura :

```
{
    "ClassName": "Region.FRXX.WebServices.REST.PAFamily.Open",
    "MethodName": "Open",
    "AppName": "SoapUI",
    "Request": {
        "UserName": "userWS",
        "Password": "demo",
        "Params": {
            "ID": "679063||2"
        }
    }
}
```

Pour sauvegarder un nouvel antécédent familial en prenant en compte la sauvegarde avec ACN, il faudra envoyer le JSON suivant :

```
{
    "ClassName": "Region.FRXX.WebServices.REST.PAFamily.Save",
    "MethodName": "SaveWithACN",
    "AppName": "SoapUI",
    "Request": {
        "UserName": "userWS",
        "Password": "demo",
        "Params": {
            "Episode": "196001820",
            "IPP": "21002205",
            "ACNContext": "Toutes les actions",
            "ACNAction": "TC.FMHIS",
            "MRCIDCodeOrDesc": "C254",
            "FAMRelationCodeOrDesc": "10",
            "FAMDesc": "Antécédent familial créé avec un webservice"
        }
    }
}
```

Et enfin pour mettre à jour l ' antécédent, on n ' enverra que les champs concernés par la modification :

```
{
    "ClassName": "Region.FRXX.WebServices.REST.PAFamily.Save",
```

```
    "MethodName": "Update",
    "AppName": "SoapUI",
    "Request": {
        "UserName": "userWS",
        "Password": "******",
        "Params": {
            "ID": "679063||1",
            "FAMDesc": "Antécédent familial modifié avec un webservice"
        }
    }
}
```


Conclusion

Les webservices REST JSON développés permettent donc d ’ accéder et de mettre à jour les données de TrakCare à partir d ’ applications tierces.

L ’ objectif futur est de faire une surcouche de l ’ Engine de redirection pour permettre aux utilisateurs d ’ accéder aux données selon les standards REST (séparation des ressources par URL, utilisation des verbes HTTP, etc.)

#JSON #ObjectScript #REST API #TrakCare