Mkey+

Utilisation des services Mkey en ligne

L'interface Web du service Mkey est disponible à l'adresse :
http://www.identificationkey.fr/mkeyplus/mkey.html
Cette interface permet de procéder à une identification interactive à partir d'une base de connaissances sous forme d'un fichier SDD (xml).

Vous pouvez :

- soit ajouter directement votre fichier sdd sur le serveur. Vous pouvez récupérer celui-ci pour faire vos tests : http://chouettes-hiboux-de-france.identificationkey.org/ChouettesetHibouxdeFrance.sdd.xml
- soit indiquer l'url d'un fichier SDD (celle-ci doit être accessible à partir d'internet). Par exemple : http://chouettes-hiboux-de-france.identificationkey.org/ChouettesetHibouxdeFrance.sdd.xml

Utilisation des API REST "Mkey" et "Xperience"

Les Web Services Mkey sont accessibles à l'url : http://mkey.services.identificationkey.fr

Les Web services retournent des objects JSON de type key : 'value'.

Tester si les services fonctionnent

Exemple : http://mkey.services.identificationkey.fr/identification

Réponse :

Available services: 'getRemainingItemsAndDescriptorsUsingIDs, getDescription, getDescriptiveData, removeSDD, getSimilarityMap, getSimilarityMapForRemainingItem, changeDescriptionHistory'

Test du service '/identification/getDescriptiveData'

Ce service prend au moins 2 parametres. L'url du fichier sdd et le poid global :

- sddURL : http://chouettes-hiboux-de-france.identificationkey.org/ChouettesetHibouxdeFrance.sdd.xml
- withGlobalWeigth : true

Dans un navigateur saisir :

http://mkey.services.identificationkey.fr/identification/getDescriptiveData?sddURL=http://chouettes-hiboux-de-france.identificationkey.org/ChouettesetHibouxdeFrance.sdd.xml&withGlobalWeigth=true

Dans une console type shell utiliser curl :

curl -i
http://mkey.services.identificationkey.fr/identification/getDescriptiveData?sddURL=http://chouettes-hiboux-de-france.identificationkey.org/ChouettesetHibouxdeFrance.sdd.xml&withGlobalWeigth=true

Documentation des Web Services REST / MKey WebServices Details

Webservice function:

The Mkey+ webservice provides 7 functions. These functions can be queried using HTTP requests. They can be reached using URLs created with the webservice address (http://mkey.services.identificationkey.fr)and a path to a webservice function. The seven functions are:

1. */xperience/addrecord*
2. */xperience/getrecord*
3. */xperience/getallitems*
4. */xperience/getfeedbackscsv*
5. */xperience/validate*
6. */xperience/updatesddurl*

For instance, the URL to call the first function is: http://mkey.services.identificationkey.fr/xperience/addrecord. These functions return Json objects, and each element returned has a specific name.

1. addrecord

Send a new record of an identification process with xperience mod

path: /xperience/addrecord

parameter:

- String sddVersion ⇒ name of the sdd file used for this identification (ex: 20150113-135241.sdd.xml)
- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex: 1916916151212-123SDAD32132)
- String jsonItemsSelectedmkeyid ⇒ json with the items given by the user (ex: [123,43])
- String urlImageUser ⇒ (facultatif)

returns:

{ 'status' : 'ok' , 'message' : // , 'spipollsessionid' : 'XXXX-YYYYY' }

error example:

{ 'status' : 'Error' , 'message' : 'Xperiense Data Base not found' ,'spipollsessionid' : 'XXXX-YYYYY' }

2. getrecord

get a history entry

path: /xperience/getrecord

parameter:

- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns: feed the json return with the list of taxa given by the user

{ "status" : "ok" , "value" : {"urlimage":"none","items":["979","1077"],"sessionid":"0123456789-id666"} }

error example:

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id66"} }

3. getallitems

Get list of items in map : Name → Uniqueid

path: /xperience/getallitems

parameter:

- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns:

{ "status" : "ok" , "value" : {"items":{"Les Sphinx du Pin et mauresque (<i>Sphinx maurorum<\/i>, <i>Sphinx pinastri<\/i>)":"1031","Les Clairons brillants (<i>Korynetes<\/i> et autres)":"812","L'Ecaille marbrée (<i>Callimorpha dominula<\/i>)":"402"},"sessionid":"0123456789-id666"} }

error example:

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id66"} }

4. getfeedbackscsv

Get all feedback as CSV in a json return

path: /xperience/getfeedbackscsv

parameter:

- String baseid ⇒ id of the base (ex : 1916916151212)

returns:

{ 'status' : 'ok' , 'value' : 'LES MATRICES' }

error example:

{ "status":"error" ,"value":{"message":"Xperience Base not found", "baseid":"0123456789"} }

LES MATRICES:

- TaxaNames (metadata) : Liste des identifiants des items avec leurs noms
- DescriptorsNames (metadata) : Liste des identifiants des descripteurs avec leurs noms
- StatesNames (metadata) : Liste des identifiants des états avec leurs noms
- TaxaConfusion : Une seule matrice de confusion des taxons
- TaxaDescriptor : Une seule matrice avec en rangé les items et en colonne les descripteurs. Pour chaque descritpeurs, quatres colonnes : nombre d'utilisation, nombre d'erreurs, nombre de doute, nombre d'erreur apres un doute
- StatesConfusionByDescriptor : Un matrice confusion entre les états par descripteur. Avant chaque matrice un ligne avec “DescIDDESCRIPTEUR” (ex : Desc12)
- StatesConfusionByDescriptor : Un matrice confusion entre les états par descripteur et par items. Avant chaque matrice un ligne avec “TaxonIDTAXON;DescIDDESCRIPTEUR” (ex : Taxon43;Desc12)

5. validate

Validate a identification process record

path: /xperience/validate

parameter:

- String item ⇒ uniqueid of the item that will be used for the validation ((ex : 32)
- String sessionid ⇒ sessionid with the form : BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns:

```
{ "status" : "ok" , "value" : {"message":"record validated","sessionid":"0123456789-id666","taxon":"371"} }
```

En cas de taxon non trouvé, on renvoie la liste des taxons possibles, dans “items” :

```
{ "status" : "notfound" , "value" : {"items":{"Les Sphinx du Pin et mauresque (<i>Sphinx maurorum<\/i>, <i>Sphinx pinastri<\/i>)":"1031"},"sessionid":"0123456789-id666","taxon":"1213"} }
```

error example:

```
{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id1","taxon":"1213"} }
```

6. updatesddurl

Update Sdd Url

path: /xperience/updatesddurl

parameter:

- String baseid ⇒ baseid : id of the base (ex : 1916916151212)
- String sddurl ⇒ the news sdd url. See the sddmanger (ex: http://nomdomain.com/messdd/

returns:

```
{ 'status' : 'ok' , 'message' : // }
```

En cas de taxon non trouvé, on renvoie la liste des taxons possibles, dans “items” :

```
{ 'status' : 'error' , 'message' : 'Xperience Base not found' }
```

error example :

```
{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id1","taxon":"1213"} }
```

Le client Mkey

Documentation du client statique “Mkey+ Static Client” réalisé pour xper ⇒ Télécharger
Les services Web étant en utilisation libre, chacun à la possibilité de développer son propre client pour interagir avec Mkey-WebServices.

Xperience

Xperience est un sous projet de Mkey. En appuis sur l'identification (Mkey), l'API xperience est un ensemble de services permettant un retour d'expériences utilisateurs. Les données générées par Xperience sont stockées dans une base de données. Elles pourront ainsi être analysés et permettre aux experts de comprendre les erreurs d'identifications.

Xperience Webservice function:

The Xperience webservice provides 6 functions. These functions can be queried using HTTP requests. They can be reached using URLs created with the webservice address (http://mkey.services.identificationkey.fr) and a path to a webservice function. The seven functions are:

1. */xperience/addrecord*
2. */xperience/getrecord*
3. */xperience/getallitems*
4. */xperience/getfeedbackscsv*
5. */xperience/validate*
6. */xperience/updatesddurl*

For instance, the URL to call the first function is: http://mkey.services.identificationkey.fr/xperience/addrecord. These functions return Json objects, and each element returned has a specific name (see Development API).

1. addrecord

Send a new record of an identification process with xperience mod

path: /xperience/addrecord

parameter:

- String sddVersion ⇒ name of the sdd file used for this identification (ex: 20150113-135241.sdd.xml)
- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex: 1916916151212-123SDAD32132)
- String jsonItemsSelectedmkeyid ⇒ json with the items given by the user (ex: [123,43])
- String urlImageUser ⇒ (facultatif)

returns:

{ 'status' : 'ok' , 'message' : // , 'spipollsessionid' : 'XXXX-YYYYY' }

error example:

{ 'status' : 'Error' , 'message' : 'Xperiense Data Base not found' ,'spipollsessionid' : 'XXXX-YYYYY' }

2. getrecord

get a history entry

path: /xperience/getrecord

parameter:

- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns: feed the json return with the list of taxa given by the user

{ "status" : "ok" , "value" : {"urlimage":"none","items":["979","1077"],"sessionid":"0123456789-id666"} }

error example:

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id66"} }

3. getallitems

Get list of items in map : Name → Uniqueid

path: /xperience/getallitems

parameter:

- String sessionid ⇒ sessionid with the form: BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns:

{ "status" : "ok" , "value" : {"items":{"Les Sphinx du Pin et mauresque (<i>Sphinx maurorum<\/i>, <i>Sphinx pinastri<\/i>)":"1031","Les Clairons brillants (<i>Korynetes<\/i> et autres)":"812","L'Ecaille marbrée (<i>Callimorpha dominula<\/i>)":"402"},"sessionid":"0123456789-id666"} }

error example:

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id66"} }

4. getfeedbackscsv

Get all feedback as CSV in a json return

path: /xperience/getfeedbackscsv

parameter:

- String baseid ⇒ id of the base (ex : 1916916151212)

returns:

{ 'status' : 'ok' , 'value' : 'LES MATRICES' }

error example:

{ "status":"error" ,"value":{"message":"Xperience Base not found", "baseid":"0123456789"} }

LES MATRICES:

- TaxaNames (metadata) : Liste des identifiants des items avec leurs noms
- DescriptorsNames (metadata) : Liste des identifiants des descripteurs avec leurs noms
- StatesNames (metadata) : Liste des identifiants des états avec leurs noms
- TaxaConfusion : Une seule matrice de confusion des taxons
- TaxaDescriptor : Une seule matrice avec en rangé les items et en colonne les descripteurs. Pour chaque descritpeurs, quatres colonnes : nombre d'utilisation, nombre d'erreurs, nombre de doute, nombre d'erreur apres un doute
- StatesConfusionByDescriptor : Un matrice confusion entre les états par descripteur. Avant chaque matrice un ligne avec “DescIDDESCRIPTEUR” (ex : Desc12)
- StatesConfusionByDescriptor : Un matrice confusion entre les états par descripteur et par items. Avant chaque matrice un ligne avec “TaxonIDTAXON;DescIDDESCRIPTEUR” (ex : Taxon43;Desc12)

5. validate

Validate a identification process record

path: /xperience/validate

parameter:

- String item ⇒ uniqueid of the item that will be used for the validation ((ex : 32)
- String sessionid ⇒ sessionid with the form : BASEID-WHATEVER (ex : 1916916151212-123SDAD32132)

returns:

{ "status" : "ok" , "value" : {"message":"record validated","sessionid":"0123456789-id666","taxon":"371"} }

En cas de taxon non trouvé, on renvoie la liste des taxons possibles, dans “items” :

{ "status" : "notfound" , "value" : {"items":{"Les Sphinx du Pin et mauresque (<i>Sphinx maurorum<\/i>, <i>Sphinx pinastri<\/i>)":"1031"},"sessionid":"0123456789-id666","taxon":"1213"} }

error example:

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id1","taxon":"1213"} }

6. updatesddurl

Update Sdd Url

path: /xperience/updatesddurl

parameter:

- String baseid ⇒ baseid : id of the base (ex : 1916916151212)
- String sddurl ⇒ the news sdd url. See the sddmanger (ex: http://nomdomain.com/messdd/

returns:

{ 'status' : 'ok' , 'message' : // }

En cas de taxon non trouvé, on renvoie la liste des taxons possibles, dans “items” :

{ 'status' : 'error' , 'message' : 'Xperience Base not found' }

error example :

{ "status" : "error" , "value" : {"message":"Record not found","sessionid":"0123456789-id1","taxon":"1213"} }

MKey WebServices Details

Webservice function:

The Mkey+ webservice provides 7 functions. These functions can be queried using HTTP requests. They can be reached using URLs created with the webservice address (http://mkey.services.identificationkey.fr) and a path to a webservice function. **The seven functions are:**

1. */identification/getDescriptiveData*
2. */identification/getRemainingItemsAndDescriptorsUsingIDs*
3. */identification/getDescription*
4. */identification/removeSDD*
5. */identification/changeDescriptionHistory*

(BETA)

6. */identification/getSimilarityMap*
7. */identification/getSimilarityMapForRemainingItem*

For instance, the URL to call the first function is : http://mkey.services.identificationkey.fr/identification/getDescriptiveData

These functions return Json objects, and each element returned has a specific name (see Development API).

Performing an Interactive Identification using Mkey+ and a custom client

Development API

The parameters in entry of the Mkey+ API functions only use the stringified* version of the Mkey+ JSON objects.

The webservice listen to GET query methods and the dataType must be Jsonp.

Here is an example of a JQuery javascript code snippet that calls one of MKey+ function:

```
var webserviceURL = http://mkey.services.identificationkey.fr
var sddFileURL = http://yourServer.com/yourSddFile.xml
$.ajax({
url : webserviceURL + '/identification/getDescriptiveData',
data : {
sddURL : sddFileURL,
withGlobalWeigth : true
},
method : 'GET',
dataType : 'jsonp'

}).done(function(data) {
… response management …
}
```

- using the standard javascript function JSON.stringify()

0. Basic Json objects used and returned by Mkey+

Item: This object represents an item which can be described in a knowledge base. For taxonomists, it usually is a taxon.

```
{
name : String, an item's name
alternativeName : String, an item's alternative name
detail : String, an item's detailed description
resourceIds : [long], the IDs of the resources associated to an item
id : long, an item's id, unique identifier generated sequentially by Mkey+
}
```

Descriptor: This object is a tool that serves to describe Items, essentially a Character for taxonomists. A Descriptor can be described with States (if it is a categorical Descriptor), or a Quantitative Measure (If it is a quantitative Descriptor).

```
{
name : String, a Descriptor's name
detail : String, a Descriptor's detailed description
resourceIds : [long], the IDs of the resources associated to a Descriptor
stateIds : [long], the IDs of the states associated to a Descriptor
inapplicableState = [long], the states of a parent Descriptor
for which a Descriptor is inapplicable
isCategoricalType = Boolean, true if the Descriptor is a categorical
Descriptor, false otherwise
isQuantitativeType = Boolean, true if the Descriptor is a quantitative
Descriptor, false otherwise
isCalculatedType = Boolean, true if the Descriptor is a calculated
Descriptor, false otherwise
id : long, this Descriptor's id, an unique identifier generated sequentially
by Mkey+
}
```

State: this object is a component of categorical Descriptors, e.g. for a Descriptor named “Color of the eye”, its States could be “Blue”, “Black”, etc…

{
name : String, a state's name
detail : String, a state's detailed description
resourceIds : [long], the IDs of the resources associated to a state
id : long, a state's id, an unique identifier generated sequentially by Mkey+
}

Resource: this object is a storage object, and is used to store media resources, which can be associated to several objects, such as Items, Descriptors, States, etc…

{
name : String, a resource's name
author : String, a resource's author
type : String, a resource's media type ("video","image","sound")
url : String, a resource's url
legend : String, a resource's legend
keywords : String, a resource's keywords
id : long, a resource's id, an unique identifier generated sequentially by Mkey+
}

DescriptionElement: This object stores the description of an Item, according to a single Descriptor, it represents the content of a single cell of the taxa / characters matrix. It may contain the list of selected States if the Descriptor is a categorical Descriptor, or a QuantitativeMeasure object, if the Descriptor is a quantitative Descriptor.

{
calculatedStates : [state], the calculated states representing an item's description
contextualWeight : int, the weigth of this description element
quantitativeMeasure : quantitativeMeasure, the quantitative measure re--prensenting an item description
states : [state], the states representing an item's description
unknown : boolean, true if this description element is unkwnow
}

QuantitativeMeasure: This object is associated to a DescriptionElement object, for a given quantitative Descriptor and Item, it contains the quantitative measures used to describe a specific Item for a given quantitative Descriptor.

{
min : long, this QuantitativeMeasure's minimum value
max : long, this QuantitativeMeasure's maximum
mean : long, this QuantitativeMeasure's mean
}

1. Service : getDescriptiveData

First function to be called, *getDescriptiveData* initializes the webservice both on the client and server side. It returns every element used in the identification process, parsed from the SDD file.

path: /identification/getDescriptiveData(String SDDurl, Boolean withGlobalWeigth)

parameter (in the javascript before stringify):

** SDDurl = String* ** withGlobalWeigth = Boolean*

returns:

{descriptorsScoreMap,Items,Descriptors,States,Resources,DescriptorRootId,Depend--ancyTable,InverteddependencyTable}

- Items = [item], every item contained in the SDD
- Descriptors = [Descriptor], every Descriptor contained in the SDD
- States = [state], every states contained in the SDD
- Resources = [resource], every resources contained in the SDD
- descriptorsScoreMap = {Descriptor,float}, an associative map, which associates

to each Descriptor its discriminant power.

- DescriptorRootId = [int], ids of the node roots (no dependency)
- InvertedDependencyTable = {long,long}, the first long is the ID of the descriptor which parent is the second long.

2. Service : getJSONRemainingItemsAndRemainingDescriptorsScoreUsingIds

Main function, used to retrieve the remaining (i.e. non-discarded) Items based on a submitted description. It also returns the remaining Descriptors, along with their new discriminant power.

path: /identification/getJSONRemainingItemsAndRemainingDescriptorsScoreUsingIds(String SDDurl, String descriptions, String remainingItemsIDs, String discardedDescrip- -torsIDs, Boolean withScoreMap, Boolean withGlobalWeigth)

parameter (in the javascript before stringify):

SDDurl = String *description = {* *selectedStatesNames : [int]* *quantitativeMeasure : {min=int,max=int,mean=int}* *}* *remainingItemsID = [int]* *discardedDescriptorsID = [int]* *withScoreMap = boolean* *withGlobalWeigth = boolean*

returns:

{discardedDescriptorsInIteration,remainingItems,descriptorScoreMap}

- remainingItems = item, array of remaining items
- discardedDescriptorsInIteration = Descriptor, array of discarded descriptor, child descriptor can be discarded with its parents.
- descriptorsScoreMap = {Descriptor,float}, an associative map, which associates to each Descriptor its discriminant power.

3. Service : getDescription

Returns the description (i.e. the description of all the Descriptors) for an Item whose name has been passed as an argument. Used to display the item description window.

path: /identification/getDescription(String itemName, String SDDurl)

parameter (in the javascript before stringify):

- itemName : String
- SDDurl : String

returns:

{description,innapDescriptorId}

- description = [descriptionElement], array of description elements describing this item.
- innapDescriptorId = [int], array of inapplicable descriptor IDs

4. Service : removeSDD

Delete the sdd from the Mkey+ server memory.

path: /identification/removeSDD(String SDDurl)

parameter (in the javascript before stringify):

- SDDurl : String

returns:

nothing

5. Service : changeDescriptionHistory

Compute identification based on the description given in parameter. This function is used to modify at all time user's identification.
path: /identification/changeDescriptionHistory

parameter (in the javascript before stringify):

- SDDurl : String
- descriptions : [{selectedStatesNames : [int], quantitativeMeasure : {min=int,max=int,mean=int}}]

returns:

{discardedDescriptors,remainingItems,descriptorScoreMap,descriptions}

- remainingItems = item, array of remaining items
- discardedDescriptorsInIteration = Descriptor, array of discarded descriptor, child descriptor can be discarded with its parents.
- descriptorsScoreMap = {Descriptor,float}, an associative map, which associates to each Descriptor its discriminant power.
- descriptions = [{selectedStatesNames: [int],quantitativeMeasure : {min=int,max=int,mean=int}}], the new description computed by the function.

6. Service : getSimilarityMap

Return a map which link items with a float representing a similarity score. This score is computed base on the descriptions submitted by the user. Items which have few difference with the descriptions given in parameter will have a high score.

path: /identification/getSimilarityMap

returns:

{similarityMap}

- similarityMap = [int,float], int is the item ID and float the corresponding similarity score.

7. Service : getSimilarityMapForRemainingItem

Return a map which link items with a float representing a similarity score. This score is computed base on a global description regrouping every remaining items description elements. Items which have few difference with the descrip- tions given in parameter will have a high score.

path: /identification/getSimilarityMapForRemainingItem

returns: {similarityMap}

- similarityMap = [int,float], int is the item ID and float the corresponding similarity score.