

xTree REST Anwender-Dokumentation
(draft 0.3, 07.10.2014) zu xTree REST Version 0.81

Aktualisierungen von Version 0.80 nach Version 0.81 (07.10.2014):

- neu: 2.1.i.8. processingStatus (Bearbeitungsstatus)
- neu: 2.1.i.9. lastUpdated (letztes Update)

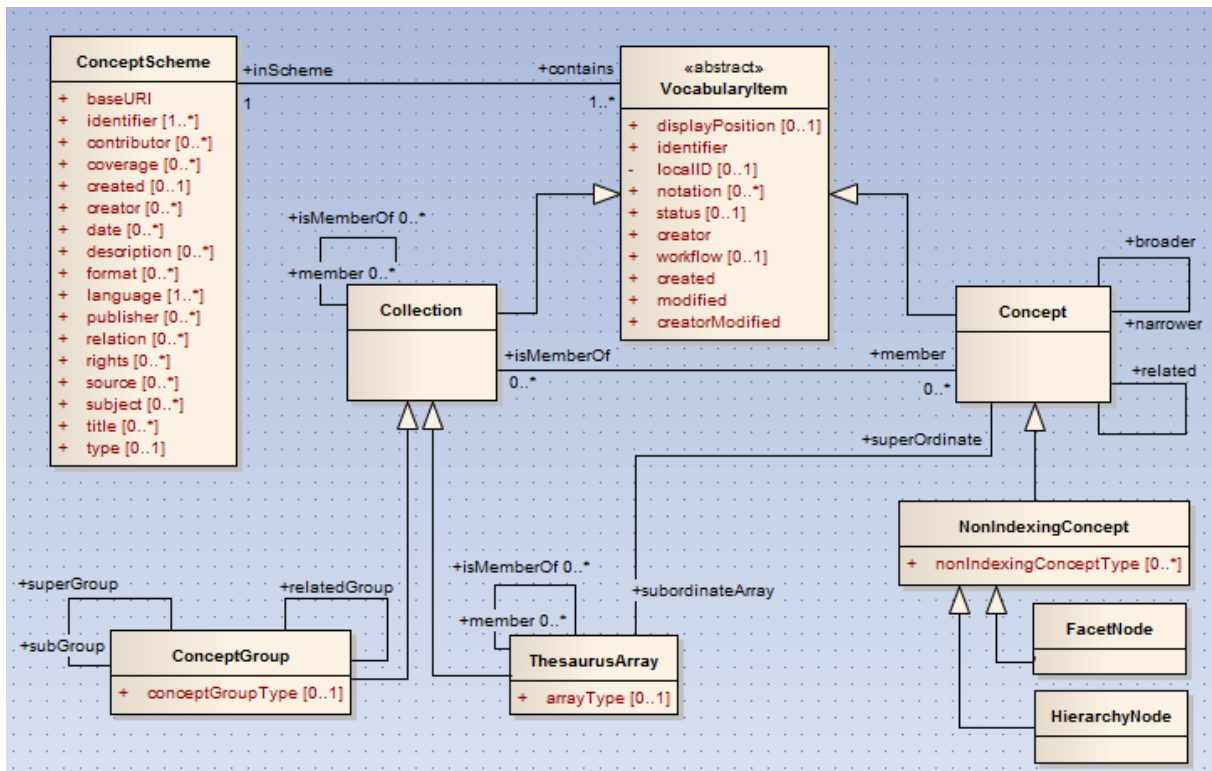
Dieses Dokument beschreibt die Verwendung der RESTful Methoden und das zurückgelieferte JSON Format für die xTree REST API

Die Methoden und das JSON -Format liegen noch nicht in der finalen Version vor. Änderungshinweise werden zukünftig in der Doku mitgeführt.

xTree REST orientiert sich weitestgehend an dem vocnet-Schema in der Version 0.8:

<http://schema.vocnet.org>

UML der Vokabularelement Klassen:



Quelle: Jutta Lindenthal

URLs zu den Anwendungen:

- xTree REST, Version 0.80 (Produktiv):
<http://xtree-rest.digicult-verbund.de>

-

1. REST Methoden

Es gibt momentan 10 lesende REST Methoden. Die Methoden können über den Standard-Port 80 des HTTP Protokolls als GET Anfrage gesendet werden.

1. **getTopClassTC**

gibt alle Begriffe + nonIndexingConcepts + FacetNodes + HierarchyNodes zurück die keine übergeordneten Elemente von dem gleichen vocItem-Typ (vocItem = Vokabular Element) haben.

Parameter:

- i. **vocabulary** (URI, Pflichtfeld)
- ii. **start** (für Blätterfunktionalität, z.B. start= 10: zeige count Einträge ab 10)
- iii. **count** (für Blätterfunktionalität, z.B. count=20: zeige 20 Einträge ab start)
- iv. **jsonfull** (Rückgabemodus)
je nach angefordertem Rückgabemodus enthält das zurückgelieferte JSON die vollständigen vocItems oder nur Teile davon.
0 = standard
1 = full (wie standard + Notes und Semantic Mappings)
- v. **lang**
all oder Sprachcode

2. **getTopCategories**

gibt alle Kategorien zurück, die keine übergeordneten Kategorien haben.

Parameter wie 1.1

3. **getTopConcepts**

gibt alle Begriffe zurück, die keine übergeordneten Begriffe haben.

Parameter wie 1.1

4. **getTopNonIndexingConcepts**

gibt alle nonIndexingConcepts zurück, die keine übergeordneten nonIndexingConcepts haben.

Parameter wie 1.1

5. **getTopFacetNodes**

gibt alle FacetNodes zurück, die keine übergeordneten FacetNodes haben.

Parameter wie 1.1

6. **getTopHierarchyNodes**

gibt alle HierarchyNodes zurück, die keine übergeordneten HierarchyNodes haben.

Parameter wie 1.1

7. **getTopNodeLabels**

gibt alle NodeLabels zurück, die keine übergeordneten NodeLabels haben.

Parameter wie 1.1

8. **getSearchVocItemsByTerm**

Methode für die Suche nach Bezeichnungen im verwendeten Vokabular.

Parameter:

- i. **vocabulary** (URI, Pflichtfeld)
- ii. **searchtermslist**
Durch Leerzeichen getrennte Suchbegriffe. Links- oder Rechtstrunkierung möglich (z.B. „*Burg“, „Burg*)

- iii. **mode** (Pflichtfeld)
UND / ODER Suche
mode = AND (UND Suche)
mode = OR (ODER Suche)
- iv. **start** (s. 1.1.ii)
- v. **count** (s. 1.1.iii)
- vi. **searchfields**
Auswahl über die Rollen der Bezeichnungen
searchfields = all | alt | hidden | pref | quasi-synonym | useBroader | useComb
- vii. **typeofvocitem**
Auswahl der zurückgelieferten Vokabularelemente
- viii. **restrictedto**
Durch Leerzeichen getrennte Liste an Identifikatoren auf die die Suche beschränkt sein soll
- ix. **jsonfull** (s. 1.1.iv)
- x. **lang** (s. 1.1.v)
- xi. **homonymlexicalvalue**
Suchfeld für Homonyme
Vorsicht: Der Parameter "homonymlexicalvalue" wird zukünftig noch in "qualifier" umbenannt

9. **getSearchVocItemsById**

- i. **vocabulary** (URI, Pflichtfeld)
- ii. **searchidslist**
Durch Leerzeichen getrennte Identifikatoren
- iii. **start** (s. 1.1.ii)
- iv. **count** (s. 1.1.iii)
- v. **typeofvocitem** (s.2.8.vii)
- vi. **jsonfull** (s. 1.1.iv)
- vii. **lang** (s. 1.1.v)

10. **getFetchHierarchy**

Methode für die Rückgabe von einem Hierarchiezweig im angegebenen Vokabular.

Parameter:

- i. **vocabulary** (URI, Pflichtfeld)
- ii. **nodeid**
ein vokabularspezifischer Identifikator, Pflichtfeld
- iii. **direction**
up oder down, Pflichtfeld
- iv. **level**
1 oder N, Pflichtfeld
- v. **start** (s. 1.1.ii)
- vi. **count** (s. 1.1.iii)
- vii. **typeofvocitem** (s.2.8.vii)

- viii. **term**
optionaler Suchterm
- ix. **jsonfull** (s. 1.1.iv)
- x. **lang** (s. 1.1.v)

2. JSON Format

ein in <http://jsonviewer.net> decodierter JSON String aus xTree REST:

```

$json (
  vocItemCount = 1
  VocabularyItem => Array (1)
  (
    ['0'] (
      Concept (
        id = "601.78"
        status = "approved"
        displayPosition = "10"
        inScheme = "http://digicult.vocnet.org/stil"
        Term => Array (3)
        (
          ['0'] (
            labelRole = "prefLabel"
            lang = "de"
            Term = "Pointillismus"
          )
          ['1'] (
            labelRole = "prefLabel"
            lang = "en"
            Term = "Pointillism"
          )
          ['3'] (
            labelRole = "altLabel"
            lang = "de"
            Term = "Chromoluminarismus"
          )
        )
      )
      broader => Array (1)
      (
        ['0'] (
          Concept (
            id = "601.2"
            status = "approved"
            inScheme = "http://digicult.vocnet.org/stil"
            Term => Array (1)
            (
              ['0'] (
                labelRole = "prefLabel"
                lang = "de"
                Term = "Kunst des 19. Jhdt./Frühe Moderne"
              )
            )
          )
        )
      )
    )
  )
)

```

Der Aufbau des JSON Formats:

1. **VocabularyItem** => Array
 - i. **Concept | ConceptGroup | ThesaurusArray | NonIndexingConcept | FacetNode | HierarchyNode**

unterschiedliche Benennungen für vocItems:

- Concept = Begriff
- ConceptGroup = Kategorie = Begriffsgruppe
- ThesaurusArray = Node Label = Gruppierung nach Merkmal
- NonIndexingConcept = Begriff als Nichtdeskriptor
- FacetNode = Facette
- HierarchyNode = Hierachieknoten

1. **id**
lokaler xTree Identifikator
2. **status**
3. **notation**
4. **displayPosition**
5. **earliestDate**
6. **latestDate**
7. **pk**
Primary Key
8. **processingStatus**
Bearbeitungsstatus
9. **lastUpdated**
letztes Update
10. **inScheme**
URI Vokabular
11. **Term** => Array
 - a. **labelRole**
 - b. **lang**
 - c. **grammaticalNumber**
 - d. **pk**
Primary Key
 - e. **Term**
 - f. **qualifier**
Homonymzusatz, neu in REST Version 0.79
12. hierarchische Relationen
 - a. für Concept:
 - i. **broader** => Array
übergeordnete vocItems: Concept,
NonIndexingConcept, FacetNode, HierarchyNode
 - ii. **narrower** => Array
untergeordnete vocItems: Concept,
NonIndexingConcept, FacetNode, HierarchyNode

- iii. **isMemberOf** => Array
übergeordnet: ThesaurusArray, ConceptGroup
 - iv. **subordinateArray** => Array
untergeordnet: ThesaurusArray
 - b. für ConceptGroup
 - i. **superGroup** => Array
übergeordnet: ConceptGroup
 - ii. **subGroup** => Array
untergeordnet: ConceptGroup
 - iii. **member** => Array
zugeordnet: Concept, NonIndexingConcept, FacetNode, HierarchyNode
 - c. für ThesaurusArray
 - i. **superOrdinate** => Array
übergeordnete vocItems: Concept, NonIndexingConcept, FacetNode, HierarchyNode
 - ii. **member => Array**
untergeordnete vocItems: Concept, NonIndexingConcept, FacetNode, HierarchyNode, ThesaurusArray
 - iii. **isMemberOf**
übergeordnet: ThesaurusArray
 - d. für NonIndexingConcept
wie Concept
 - e. für FacetNode
wie Concept
 - f. für HierarchyNode
wie Concept

13. Note => Array

- a. **lang**
- b. **source**
- c. Anmerkungstypen
 - i. **changeNote**
 - ii. **definition**
 - iii. **editorialNote**
 - iv. **etymology**
 - v. **example**
 - vi. **explanation**
 - vii. **extDefinition**
 - viii. **glossaryEntry**
 - ix. **historyNote**
 - x. **miscNote**
 - xi. **note**
 - xii. **scopeNote**

14. MapItem => Array

a. mappingRelation

closeMatch | exactMatch | broadMatch | narrowMatch | relatedMatch

s.a. <http://www.w3.org/TR/skos-reference/#mapping>

b. mapItemID

c. mapItemIDType

uri | url | hdl | local | urn | doi

d. mapItemLabel

e. mapItemSource

15. related => Array

assoziative Beziehung zwischen zwei Begriffen