REST-API für kantonale Geodatenmodelle

Aggregationsinfrastruktur der Kantone

Stefan Ziegler / stefan.ziegler@outlook.com

27. März 2016

Version	Datum	Author	Bemerkungen
0.1	27.03.2016	Stefan Ziegler	Initialversion

Allgemein

Bla bla blanig

Erster «very early draft». Funktionsumfang sicher noch nicht vollständig.

Offene Fragen:

- Umgang mit multi-part Datensätzen. Überhaupt notwendig?
- Mehrsprachigkeit für KGDM?

Kantone (=Mandanten)

`/cantons`

Method	Action	Status code	Formats	Parameters
GET	List all cantons	200	XML, JSON	
POST	Create a new canton	201	XML, JSON	
PUT		405		
DELETE		405		

GET Liefert alle Kantone, die aktiviert sind und welche die Möglichkeit haben, KGDM zu verwalten und anzubieten.

POST Erstellt einen neuen Kanton. Es ist nicht möglich ein beliebiges Objekt zu erstellen, sondern das Objekt unterliegt gewissen Bedingungen/Einschränkungen. Es können nur offizielle Kantone (z.B. «SO», «ZH» etc.) erstellt werden. Dieser Sachverhalt wird beim Erstellen geprüft. Umgesetzt ist es momentan ganz einfach: Die Tabelle enthält bereits sämtliche Kantone. Über das Attribut *activated* wird die Sichtbarkeit gesteuert. (Zuständigkeit: AI-Administrator).

Exceptions

Exception	Status code
POST a non valid canton	406

Beispiele

GET XML-Output:

```
curl -v -XGET
http://localhost:8080/aggregationsinfrastruktur/rest/cantons/
```

resp.

```
curl -v -H "Accept: application/xml" -XGET
http://localhost:8080/aggregationsinfrastruktur/rest/cantons/
```

GET JSON-Output:

```
curl -v -H "Accept: application/json" -XGET
http://localhost:8080/aggregationsinfrastruktur/rest/cantons/
```

POST XML-Input:

```
curl -v -H "Content-type: application/xml" -d
"<canton><code>GL</code><email>foo.bar@gl.ch</email></canton>"
-XPOST
http://localhost:8080/aggregationsinfrastruktur/rest/cantons/
```

`/cantons/<ct>`

Method	Action	Status code	Formats	Parameters
GET	Return canton `<ct>`	200	HTML, XML, JSON	
POST		405		
PUT	Modify canton `<ct>`	200	XML, JSON	
DELETE	Delete canton `<ct>`	200	XML, JSON	recursive

GET Liefert Kanton `<ct>` zurück.

PUT Verändert Kanton `<ct>`. Nur sinnvoll, falls neben offiziellem Kantonskürzel (z.B. «SO», «ZH» etc.) weitere Informationen verwaltet werden, z.B. eine allgemeine Kontaktadresse etc. Ansonsten ist es wenig sinnvoll etwas zu verändern, das sowieso strengen Bedinungen unterliegt.

DELETE Löscht Kanton `<ct>`. Es können keine Kantone gelöscht werden für den bereits KGDM angeboten werden. In diesem Fall muss der *recursive*-Parameter verwendet werden. (Zuständigkeit: Kanton-Administrator).

Exception	Status code
GET for a canton that does not exist.	404
DELETE a non empty canton.	403

Modelle (=KGDM)

`/cantons/<ct>/models`

Method	Action	Status code	Formats	Parameters
GET	List all models	200	XML, JSON	
POST	Create a new model	201	XML, JSON	
PUT		405		
DELETE		405		

GET Liefert alle vorhandenen Modelle für einen bestimmten Kanton <ct>.

POST Erstellt ein neues Modell. (Zuständigkeit: Kanton-Administrator)

`/cantons/<ct>/models/<mdl>`

Method	Action	Status code	Formats	Parameters
GET	Return model `<mdl>`	200	HTML, XML, JSON	
POST		405		
PUT	Modify model `<mdl>`	200	XML, JSON	
DELETE	Delete model `<mdl>`	200	XML, JSON	*recursive*

GET Liefert das Modell `<mdl>` zurück.

PUT Verändert Modell `<mdl>`. Einsatzzweck noch ungewiss. Geht es um Metadaten-Veränderungen (z.B. E-Mailadressen o.ä.) oder wirklich um Modelländerungen? Falls es um Modelländerungen geht, was sind die Auswirkungen auf die unterliegenden Daten? Die passen Dann ja eventuell gar nicht mehr zusammen.

DELETE Löscht Model `<mdl>`. Es können keine Modelle gelöscht werden, falls bereits Daten importiert worden sind. Für diesen Fall muss der *recursive*-Parameter verwendet

werden.

Exception	Status code
GET for a model that does not exist.	404
DELETE a non empty model.	403

Feature types (=Daten/Lieferungen)

`/cantons/<ct>/models/<mdl>/featuretypes`

Method	Action	Status code	Formats	Parameters
GET	List all feature types	200	XML, JSON	
POST	Create a new feature type	201	XML, JSON	
PUT		405		
DELETE		405		

GET Liefert eine Liste sämtlicher *feature types*. Als *feature type* wird nicht direkt und einzig der Datensatz zu einem bestimmten MGDM (`<mdl>`) verstanden, sondern seine Metainformationen. Die Liste enthält z.B. Datum der Lieferung, Name der Datei resp. WFS-URL, eventuell Link zum Datensatz etc.

POST Der Kantons-Administrator löst auf der AI einen Import aus. Noch unklar ist, ob die AI die Daten holt oder die Daten hochgeladen werden.

`/cantons/<ct>/models/<mdl>/featuretypes/<ft>`

Method	Action	Status code	Formats	Parameters
GET	Return feature type `<ft>`	200	XML, JSON	
POST		405		
PUT	Modify feature type `<ft>`	200		
DELETE	Delete feature type `<ft>`	200		

GET Liefert den gewünschten *feature type* `<ft>`. Es wird nicht der Datensatz

zurückgeschickt, sondern die dazugehörigen Metainformationen.

PUT Ändert den *feature type* `<ft>`. Gibt es sinnvolle Anwendungsfälle?

DELETE Löscht den feature type `<ft>`. Gibt es sinnvolle Anwendungsfälle? Eventuell um den letzten (momentan aktuellen) feature type zu löschen. In diesem Fall würde der zweitletzte wieder der online geschaltet. Oder so ähnlich.

Kunden

`/cantons/<ct>/models/<mdl>/costumers`

Method	Action	Status code	Formats	Parameters
GET	List all costumers	200	XML, JSON	
POST	Create a new customer	201	XML, JSON	
PUT		405		
DELETE		405		

GET Liefert eine Liste sämtliche Kunden für das bestimmte KGDM.

PUT Erstellt ein neuen Kunden. (Zuständigkeit: Kanton-Administrator).

`/cantons/<ct>/models/<mdl>/costumers/<cu>`

Method	Action	Status code	Formats	Parameters
GET	Return customer `<cu>`	200	XML, JSON	
POST		405		
PUT	Modify costumer `<cu>`	200		
DELETE	Delete customer `<cu>`	200		

GET Liefert den gewünschten Kunden `<cu>`. Es werden sämtliche Informationen des Kundens zurückgeliefert.

PUT Nimm für den Kunden `<cu>` Veränderungen vor. Vorstellbar z.B. andere geografische Einschränkungen, Änderung des Anfangs- resp. Endtermines der erlaubten

Nutzung etc.

DELETE Löscht den Kunden <cu>.