(Get) /admin/exportConfig/:configName

(post) /admin/importConfig

(Get) /admin/getLayersFromCapabilities

====================================> config <=====================================

(Get) /admin/config

Returns a list of all configs in the database.

(Get) /admin/config?name=someName

Returns a list of all configs in the database that has a specific name

(Get) /admin/config?search=searchStr

Returns a list of all configs in the database that their name matches the search string

(Get) /admin/config?name=someName&search=searchStr

Returns a list of all configs in the database that has a specific name. In this case search has no effect and name query takes precedence.

(Post) /admin/config

Creates a new config, an object is expected in the body:

{

"name": "conf50",

"projection\_code": "EPSG:3006",

"projection\_extent": "487000,6803000,773720,7376440",

"extent": "487000,6803000,773720,7376440",

"center": "617820,6920585",

"zoom": 3,

"resolutions": "280,140,70,28,14,7,4.2,2.8,1.4,0.56,0.28,0.14,0.112,0.056",

"featureinfo\_options": null,

"proj4defs\_id": 1

}

Including a new id in the body will throw an SQLite error, a new id will be set automatically and returned. Excluding a property results in “null” value for that property. If no object is provided in the body an object will be created that has “null” value for all of its properties except for id. Including properties other than those specified in the database has no effect. For example if there is a typo in the name:

{

"namee": "conf50",

}

This will create a new object with (“name” : null) and above “namee” will be ignored.

(Get) /admin/config/:id

Returns a config with a specific id

(Put) /admin/config/:id

Updates a config with a specific id, an object is expected in the body that has the desired modifications in it:

{

"name": "new\_name",

"projection\_code": "new\_projection\_code"

}

Including a new id in the body has no effect, id cannot be changed. There is no need to include all the properties in the body, it suffices to have those to be changed only. For example above object only updates name and projection\_code and leaves the rest of the object unchanged. If no object is provided in the body object in the database remains unchanged.

(Del) /admin/config/:id

Deletes a config with a specific id. Note that removing a config means that other objects belonging to that config like layers and groups remain in the database but unconnected and useless! Adding a config with the same name has no effect either because configs have a unique id beside their unique name, so the new config will have a different id that cannot match and find other objects although it has the same unique name as the deleted config.

=====================================> group <=====================================

(Get) /admin/group

(Get) /admin/group?name=someName

(Get) /admin/group?search=searchStr

(Get) /admin/group?name=someName& search=searchStr

(Get) /admin/group/:id

There is a method available on group objects: getConfig(). This is a handy method that reads the config id from a layer that belongs to this group so we know which config the group belongs to.

(Put) /admin/group/:id

(Post) /admin/group/:id

(Del) /admin/group/:id

=====================================> layer <=====================================

(Get) /admin/layer

(Get) /admin/layer?name=someName

(Get) /admin/layer?search=searchStr

(Get) /admin/layer?configId=id

To retrieve all layers that belong to a specific config

(Get) /admin/layer?name=someName& search=searchStr

(Get) /admin/layer/:id

(Put) /admin/layer/:id

(Post) /admin/layer/:id

(Del) /admin/group/:id

=====================================> source <=====================================

(Get) /admin/source

(Get) /admin/source?name=someName

(Get) /admin/source?search=searchStr

(Get) /admin/source?name=someName& search=searchStr

(Get) /admin/source/:id

(Put) /admin/source/:id

(Post) /admin/source/:id

(Del) /admin/source/:id

=====================================> attribute <===================================

(Get) /admin/attribute

Returns a list of all attributes in the database.

(Get) /admin/attribute?layerId=id

To retrieve all attributes that belong to a specific layer

(Get) /admin/attribute/:id

Returns an attribute with a specific id

(Put) /admin/attribute/:id

(Del) /admin/attribute/:id

(Post) /admin/attribute

Creates several new attributes, an array of objects is expected in the body:

[{

"name": "DESCRIPTO",

"title": "Comment",

"url": "",

"url\_title": "",

"layer\_id": 147

},

{

"name": "MAX",

"title": "max",

"url": "",

"url\_title": "",

"layer\_id": 147

}]

=====================================> domain <===================================

admin/domain/fetchLayersAndGroups/:configId

returns an object with a list of all layers and all groups for a config id. Groups are also available in each layer object. This is how orm works, if we find a layer from db, and then find its group using getGroup(), the layer object becomes like:

{

"name": "Lantmateriet:Ortofoto50\_25cmz4\n",

"name\_id": null,

"title": "Flygfoto",

"format": "image/jpeg",

"queryable": false,

"visible": false,

"type": "WMS",

"attribution": "&copy Lantmäteriet Geodatasamverkan",

"order\_number": 1,

"id": 5,

"config\_id": 4,

"group\_id": 1,

"source\_id": 1,

"style\_id": 1,

"group": {

"name": "backgrdfsfsfound",

"title": "Bakgrundskartor\_edited",

"parent": null,

"expanded": true,

"order\_number": 1,

"id": 1

}

}

meaning that both group\_id and group become available. Note that some layer may not have any group, but that is allowed only to be handy for testing and developing. Each layer MUST have a group for the client code to run properly and it will crash otherwise.

admin/domain/fetchConfigGroups/:configId

Returns all groups that belong to a config, by reading all layers of that config first and then corresponding group of each layer.