Geffrye REST API User Guide

Revision 1.1 BETA– August 2014

System Simulation

http://www.ssl.co.uk

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

TABLE OF CONTENTS

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Document history

Revision	API version	Date	Notes
1.1	v1 BETA	2014-09-02	Base URL updated
1.0	v1 BETA	2014-08-22	First issue

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Introduction

The Geffrye Museum holds a collection of objects relating to the history of the English domestic interior. For examples of objects in the collection, see

http://www.geffrye-museum.org.uk/collections/

A portion of the database is accessible via a URL-based “REST”-like API. This document describes how to access and use that API.

Some familiarity with the HTTP protocol and with JSON is assumed.

The API version at the time of writing is v1 BETA.

If you want to dive straight in with some examples, skip to the section at the end.

Geffrye data model

For the REST API, the data in the Geffrye database is partitioned into resource sets (items of different types). The following sets are available via the API:

- `objects`

  Museum objects from the collection.

- `persons`

  The details of specific individuals referenced from objects, e.g. artists, donors.

- `organisations`

  The details of specific organisations referenced from objects.

- `places`

  The details of specific geographical locations referenced from objects.

API basics

Base URL

The API is hosted at the following URL:

`http://mailgate.geffrye-museum.org.uk:7452/api/rest/beta/`

All your URLs should begin with this.

REST

A REST API uses HTTP requests to interact with a database. The HTTP request methods correspond to “CRUD” database operations:

HTTP method	Database operation
POST	Create
GET	Read
PUT	Update
DELETE	Delete

At the time of writing, the Geffrye REST API is a read-only API and supports only GET requests to retrieve data from the database.

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

If you are unable to phrase your request as a GET request (e.g. because platform/toolkit limitations mean you have to POST) you can include an explicit `method` URL parameter, e.g.

```
?method=GET
```

Some API terminology

Sets (tables, record types) contain ***items*** (rows, records) which have ***elements*** (columns, fields).

API response

Format

By default, API return values are returned as a JSON-encoded object in the body of the HTTP response. However, the API also supports JSON-P and XML encodings.

Use the `format` URL parameter to explicitly request a particular format:

```
?format=json     (the default if omitted)

?format=jsonp&callback=yourFunction

?format=xml
```

The root element for an XML return object is `<return>`.

In XML responses, array values are returned simply as repeating elements. E.g.

JSON:

```
foo: [ "a", "b", "c" ]
```

XML:

```
<foo>a</foo>
<foo>b</foo>
<foo>c</foo>
```

The HTTP Content-Type header will be set appropriately depending on the format returned.

The character set is always UTF-8.

Return values

The response object always includes the following members:

`success`	Boolean (`true` or `false`)	Whether or not your API call has worked.
`result`	Object	The return value of the API call, if successful. The structure will depend on the particular call.

E.g.

Request:

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

```
/objects/O44924
```

Return object:

```
{
    success: true,
    result: {
        uniqueID: "O44924",
        wholeObjectName: "footstool",
        …
    }
}
```

If the `success` member is false, the `result` member will be an object containing the following members:

`errorCode`	Integer	A unique code for each kind of error the API can generate. (See "Error handling" below)
`errorMessage`	String	A description of the error which may reference the particulars of your request

E.g.

Request:

```
/objects/O44924?elements=badger
```

Return object:

```
{
    success: false,
    result: {
        errorCode: 102,
        errorMessage: "Bad element 'badger' for objects"
    }
}
```

Error handling

You should first check the response's HTTP status code. This will be:

HTTP status code	**What it means**
500 Internal Server Error	There was a server error or misconfiguration. The body of the response will be an HTML page.
405 Method Not Allowed	You attempted something other than a GET request.
404 Not Found	Possible causes are that your request URL • was incomplete (e.g. missing the "beta" version number) • was incorrect (e.g. a typo, the wrong version number) • referenced an unknown resource set • referenced a non-existent item ID The body of the response will be an HTML page.

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

400 Bad Request	You did something wrong in your API call. The `success` member of the response object should be false, and the `result` member will be an object containing error information as described in “Return values” above.
200 OK	Your API call completed. The `success` member of the response object should be true.

Ideally you should cope with redirection codes (302, 303) in case the API moves in the future. All other status codes should be treated as errors.

API error codes

Code	What it means, and troubleshooting
102	*Unknown data element* You've asked for a data element that doesn't exist. Check for typos, check you're querying the right set, and check against the list of elements in “Set element reference”. Note that element names are case-sensitive.
103	*Unsupported response format* You've asked for a `format` other than `json`, `jsonp`, or `xml`.
104	*Missing JSON-P callback* If you've specified `format=jsonp`, make sure you also specify a `callback`.
105	*Bad query element* You're trying to query into an unknown query element, or you're using an invalid operator for that query element. Check for typos, check you're querying the right set, make sure it's in the list in “Set element reference” and that it supports the operator you're using.
106	*Bad argument to query element* In practice, this means you didn't supply a valid range to the `range` operator. Make sure you give a comma-separated pair of values.
107	*Attempt to query an item* If you've made a unique item request (e.g. /objects/O44924), you can't include query parameters in the URL.
108	*Invalid result offset* You've supplied a negative `offset`, or you've exceeded the number of items in the query result. (Note that this could happen even if your maths is correct, because the size of the result could conceivably change across requests.)
109	*Invalid result limit* You've specified a negative `limit`. (Note that a limit of zero is ok, but will of course always yield no matches.)
110	*Bad sort* Either the specified sort element is not recognised, or it requires a value that hasn't been specified (e.g. a coordinate point, when sorting by distance)

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Fetching data – GET requests

Fetching all the items in a set

The URL to fetch all the items in a set is

/{set}/

(Include the trailing slash – it's important.)

For example

http://gmdb.gmdb.org.uk/api/rest/beta/objects/

will fetch all the collection objects in the database. Note that items are returned in batches (see “Traversing the set” below).

The response object's `result` member will be an object with the following members:

`found`	Integer	The total number of items in the set.
`items`	Array	The list of items in the batch. Each item is an object whose elements depend on the type of item – see “Set element reference” for reference.

Traversing the set

Set items are returned in batches (of 10 by default). You can control the batch size, and the starting point in the set, using the following URL parameters, both of which are optional:

`offset`	Integer	The zero-based index into the set of the first item to be returned.	Default: 0 (i.e. start at first item)
`limit`	Integer	The maximum number of items to be returned.	Default: 10

E.g.

/objects/?**offset=100**&**limit=50**

(Note the / before the ? - it's important.)

Limiting the item data returned

By default, the API will include all the available elements for each item returned. You may only be interested in certain elements, in which case you can use the `elements` URL parameter:

`elements`	Comma-separated list	Limits the elements returned for each item to those specified in the list

E.g.

Request:

/objects/?**elements=uniqueID,title,URL**

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Item in response:

```
{
    uniqueID: "O44923",
    title: "chair",
    URL: "http://www.geffrye-
museum.org.uk/collections/explore-our-collections/item-
detail/?id=O44923"
}
```

See "Set element reference" for the full set of elements available for each set.

Including elements from related items

Items in one set may be related to items in another set. For example, objects can have a place of production. When specifying the elements to return, you can pull in data from across such a relationship by using dot notation.

E.g.

Request:

/objects/?elements=uniqueID,**productionPlaces.name**

Item in response:

```
{
    uniqueID: "O44422",
    productionPlaces: [
        {
            name: "Hackney"
        }
    ]
}
```

This mechanism helps you avoid making multiple API requests to gather data from multiple sets.

Link elements (such as "productionPlaces" in the above example) are indicated in the "Set element reference" section of this document.

You can use the link element directly in your list of elements (`elements=productionPlaces`), in which case the API will return all the elements from the linked item.

Sorting

By default, objects are sorted by production date (the date the object was made). Currently, the API does not support other sort orders.

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Querying sets (searching)

You can filter sets by including query parameters in your request URL. The response object and traversal logic is the same as for a plain set request, except of course that you'll only be seeing the matching subset.

Free text search

The URL parameter `q` will perform a free text search across all the elements in the set and return only matching items.

For example:

/objects/?**q=chair**

In most cases, text searches support wildcards using an asterisk, e.g.

/object/?**q=chair***

Element-specific search

For each set there are a range of *query elements* which let you search into specific data. To use a query element, include it in your request URL with a prefix of "`q.`".

E.g. to search for the object with unique ID O44924

/objects/?**q.uniqueID=O44924**

The list of available query elements is given in the "Set element reference" section of this document. (Note that although these often share the same name as item elements, they aren't the same, and you can't therefore search into every item element.)

Some query elements support multiple *operators* which control how the query is interpreted. For example, the object `wholeObjectName` query element supports both "exact" searching (for exact matches) and "text" searching (free text and potentially partial matches). The operator is appended to the query element in your request:

/objects/q.wholeObjectName**.exact=footstool**

The full set of operators is as follows (although consult the "Set element reference" section to see which query elements support which operators)

`text`	Perform a free text search into the element. For example, "york" would match "new york"
`exact`	Return only items where the value is found exactly. For example, "york" would match "York" but not "New York"
`range`	Return only items where the value is within the range specified (as a comma-separated pair). You can also give just a single value, in which case it's treated as a lower bound. To specify only an upper bound, prefix your value with a comma (i.e. supply only the second half of the comma-separated pair).
`branch`	The values in some data elements are taken from a hierarchical authority. The branch operator requests that not only items with the query term be found, but also any items using terms below the query term in the hierarchy. For example, "London" sits below "England" in the tagging hierarchy, so while `q.contentTag.exact=England` would only find items tagged with "England", `q.contentTag.branch=England` would also find items tagged with "London".

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

The default operator for a query element (i.e. if you don't specify one explicitly) is usually “text”, but consult the element reference for specific elements.

Fetching an item by unique ID

As well as querying a set to retrieve an item by its ID (using the `uniqueID` query element as in previous examples), you can use a plain URL to fetch the item:

/{set}/{id}

For example

http://gmdb.gmdb.org.uk/api/rest/beta/objects/O44924

IDs are not case sensitive in this usage; however we'd recommend you always use the correct case.

As with set requests and queries, you can also use the `elements` URL parameter to control what item elements are returned (see “Limiting the item data returned”).

If no item exists in the database for the given ID, or the ID corresponds to an item that's in a different set, a 404 Not Found response is returned. The body of the response is an HTML page, so don't attempt to parse it.

If the ID does exist, the `result` member of the response object will be the item, returned as an object.

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Examples and common recipes

Note: you'll need to append your API key to all of these examples (add &key=yourkey to the end).

Fetching an obejct by ID as a JSON object

http://gmdb.gmdb.org.uk/api/rest/beta/objects/O44924

Returning XML rather than JSON

http://gmdb.gmdb.org.uk/api/rest/beta/objects/O44924?format=xml

Returning JSON-P rather than JSON

http://gmdb.gmdb.org.uk/api/rest/beta/objects/O44924?format=jsonp&callback=processObject

Returning specific object information rather than everything

http://gmdb.gmdb.org.uk/api/rest/beta/objects/O44924?elements=wholeObjectName,URL

Fetching all the objects (first batch of 10 obejcts)

http://gmdb.gmdb.org.uk/api/rest/beta/objects/

Fetching the next batch of 10 objects

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?offset=10

Fetching all the objects (first batch of 50 objects)

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?limit=50

Fetching the next batch of 50 objects

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?limit=50&offset=50

Find objects containing the word “Tudor”

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?q=Tudor

Find objects containing the word “Tudor” in the name

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?q.wholeObjectName=Tudor

Find objects created in the 16th century

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?q.productionDates.range=1500,1599

Include the production place name from the places authority in object search results

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?elements=uniqueID,title,productionPlaces.name

Find objects made of paper

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?q.materials=paper

Find objects keyworded with any kind of Lighting keyword (gas lighting, oil lighting etc)

http://gmdb.gmdb.org.uk/api/rest/beta/objects/?q.webKeywords.branch=Lighting

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Set element reference

`objects`

Data elements

Element	List?	Type	Notes
`uniqueID`	No	string	The Geffrye's unique ID for the object.
`publicationDate`	No	date (string)	The date the object was added to the Geffrye database.
`modificationDate`	No	date (string)	The date the object data was last modified.
`wholeObjectName`	No	string	The broad type of object, e.g. chair, door.
`collectionCode`	No	string	A code for the responsible department.
`objectNumber`	No	string	The unique identifier assigned to the physical object by the Museum.
`briefDescription`	No	string	Free text short description.
`title`	Yes	string	The titles by which the object is known.
`materials`	Yes	string	
`techniques`	Yes	string	
`dimensions`	Yes	string	A list of dimensions of the form “{dimension}: {value}{unit}”, e.g. “Height: 27cm”
`stylePeriods`	Yes	string	
`productionDates`	Yes	string	
`associatedDates`	Yes	string	
`labelText`	No	string	
`webKeywords`	Yes	string	
`URL`	No	URL (string)	The URL of this object on the Geffrye website, if published.
`thumbURL`	No	URL (string)	The URL of a 96x96 pixel thumbnail for the object.
`parts`	Yes	object: `description` `onDisplay`	A list of the parts of the object, including a description of each part, and whether the part is on display (“Y” if so)

Link elements

Element	Relation	Links to (set)
`productionPlaces`	1:N	`places`
`productionPersons`	1:N	`persons`
`productionOrganisations`	1:N	`organisations`
`associatedPlaces`	1:N	`places`
`associatedPersons`	1:N	`persons`

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

Query elements

Unless otherwise specified, query elements search into the data elements of the same name.

The first operator listed is the default operator if no operator is explicitly specified.

Element	Supported operators	Notes
uniqueID	exact	Case insensitive
publicationDate	exact, range	
modificationDate	exact, range	
wholeObjectName	text, exact	
materials	text	
techniques	text	
stylePeriods	text	
productionDates	range	
webKeywords	branch	

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

persons

Data elements

Element	List?	Type	Notes
uniqueID	No	string	The Geffrye's unique ID for the object.
publicationDate	No	date (string)	The date the person was added to the Geffrye database.
modificationDate	No	date (string)	The date the person data was last modified.
name	No	string	Person's full name
occupation	No	string	

Query elements

Unless otherwise specified, query elements search into the data elements of the same name.

The first operator listed is the default operator if no operator is explicitly specified.

Element	Supported operators	Notes
uniqueID	exact	Case insensitive
publicationDate	exact, range	
modificationDate	exact, range	

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

organisations

Data elements

Element	List?	Type	Notes
uniqueID	No	string	The Geffrye's unique ID for the object.
publicationDate	No	date (string)	The date the organisation was added to the Geffrye database.
modificationDate	No	date (string)	The date the organisationdata was last modified.
name	No	string	

Query elements

Unless otherwise specified, query elements search into the data elements of the same name.

The first operator listed is the default operator if no operator is explicitly specified.

Element	Supported operators	Notes
uniqueID	exact	Case insensitive
publicationDate	exact, range	
modificationDate	exact, range	

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission

places

Data elements

Element	List?	Type	Notes
uniqueID	No	string	The Geffrye's unique ID for the object.
publicationDate	No	date (string)	The date the place was added to the Geffrye database.
modificationDate	No	date (string)	The date the place data was last modified.
name	No	string	
latitude	No	string	E.g. N 52° 59' 53"
longitude	No	string	E.g. W 0° 24' 33"

Query elements

Unless otherwise specified, query elements search into the data elements of the same name.

The first operator listed is the default operator if no operator is explicitly specified.

Element	Supported operators	Notes
uniqueID	exact	Case insensitive
publicationDate	exact, range	
modificationDate	exact, range	

© 2014 System Simulation Ltd. All rights reserved
No reproduction without permission