

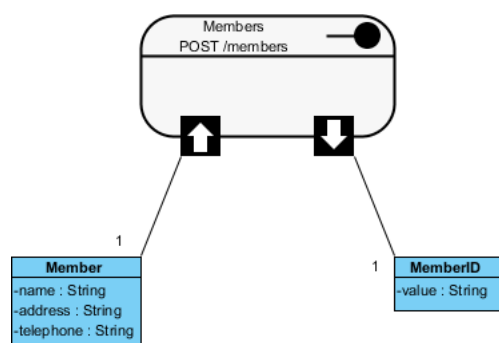


How to Design REST API?

Written Date : March 23, 2015

REpresentational **S**tate **T**ransfer, an architectural style that can be used in building networked applications, is becoming increasingly popular nowadays. Many leading vendors have opened the doors of their services to developers, providing them with restful accesses to different web services.

This tutorial shows how to design REST API with Visual Paradigm. The generation of API (code and library) and API documentation will also be covered.

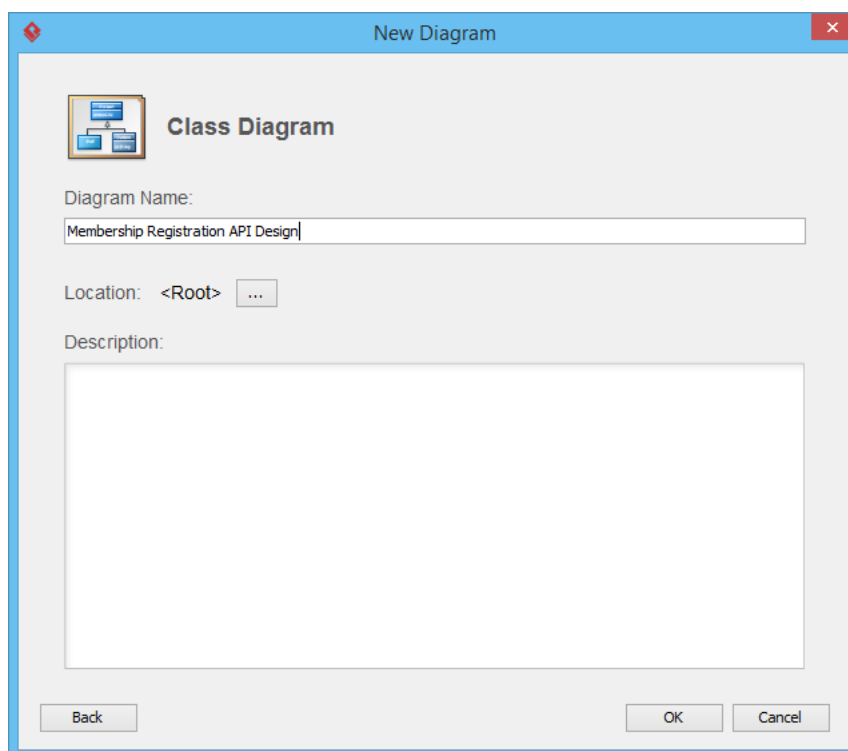


In this tutorial, we will use a simple membership registration service as an example to show how to design the REST API for such a service.

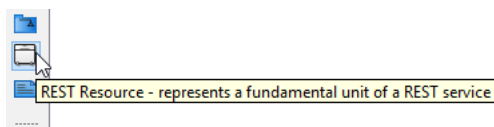
Designing REST API

1. The design of REST API has to done in a Class Diagram. To create a Class Diagram, select **Diagram > New** from the toolbar.
2. In the **New Diagram** window, select **Class Diagram** and click **Next**.

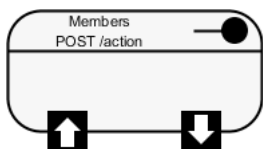
3. Enter *Membership Registration API Design* as diagram name.



4. Click **OK** to confirm.
5. Select **REST Resource** in the diagram toolbar.



6. Click on the diagram to create a REST Resource and name it *Members*.

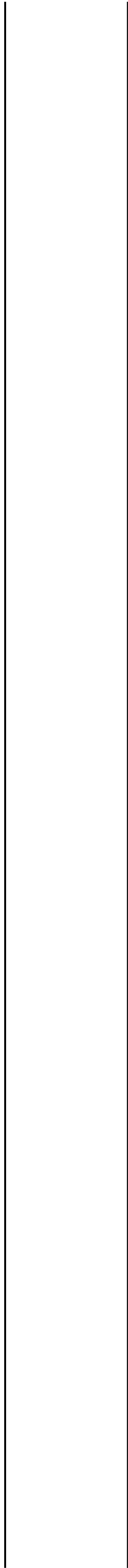


A REST resource is the fundamental unit of an API that conforms to REST, which is what we called a REST API. It is an object with a URI, the http request method, associated parameters and the request/response body. Each of the REST resources represents a specific service available on the path specified by its URI property. Therefore, if you want to model multiple services, please draw multiple REST resources

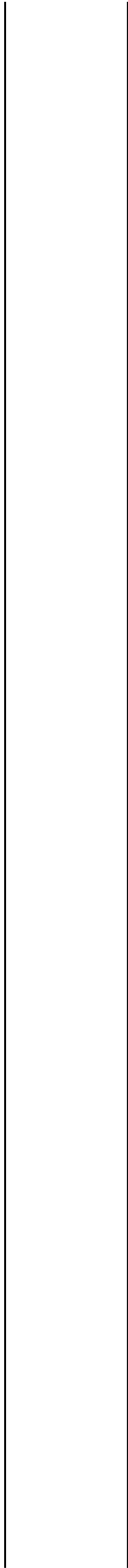
7. Right-click on the *Members* REST Resource and select **Open Specification...** from the popup menu.
8. In the General tab, fill in the following:

Field	Value	Remarks
URI	/members	Each REST Resource has its own URI. Consumers access to URL to access for the REST Resource. Typically, a RESTful URI should refer to a resource that is a thing instead of referring to an action. Therefore when you are deciding the URI, try to use a

		noun instead of a verb.
Method	POST	Specifies the action to act on the resource. GET - A GET method (or GET request) is used to retrieve a represent of a resource. It should be used SOLELY for retrieving data and should not alter. PUT - A PUT method (or PUT



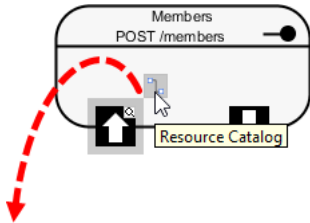
request)
is
used
to
update
a
resource.
For
instance,
if
you
know
that
a
blog
post
resides
at
http://
www.exar
blogs/123
you
can
update
this
specific
post
by
using
the
PUT
method
to
put
a
new
resource
represent
of
the
post.
POST
-
A
POST
method
(or
POST
request)



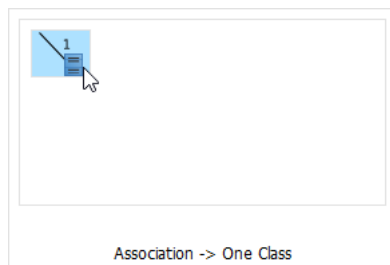
is used to create a resource. For instance, when you want to add a new blog post but have no idea where to store it, you can use the POST method to post it to a URL and let the server decide the URL. **DELETE** - A DELETE method

		(or DELETE request) is used to delete a resource identified by a URI.
Description	Create a new member by providing his/her name, address and telephone number. You will receive an object that holds the name, address, telephone and member ID of the member.	Description of resource that will appear in generated API document. It is recommended to provide clear description of the service, so that the consumer know what the service is and how to operate with it.

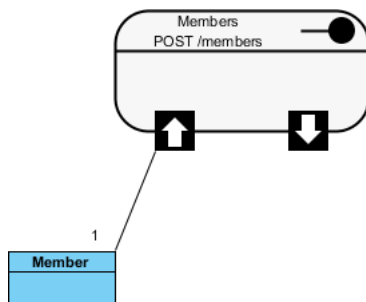
- Click **OK**.
- Let's say membership registration requires users to provide information like name, address and telephone. In API level, these are the parameters required by the service and should be provided by the consumer of the service. Let's represent this in our design. Move your mouse pointer over the **REST Request Body** icon and drag out the **Resource Catalog** button at top right.



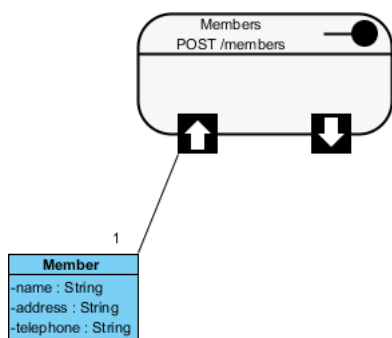
- Release the mouse button and select **Association -> One Class** from Resource Catalog.



- Name the class *Member*.

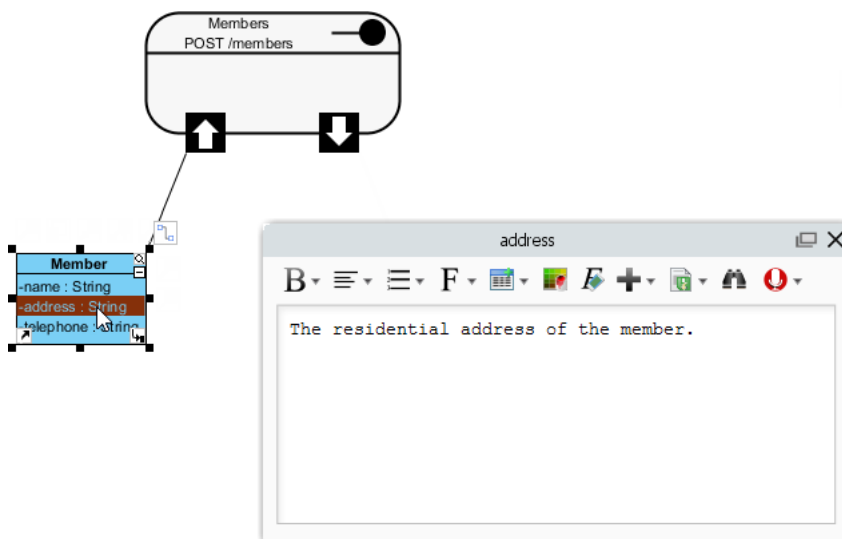


13. Add three String attributes into the *Member* class as parameters - *name*, *address* and *telephone*.



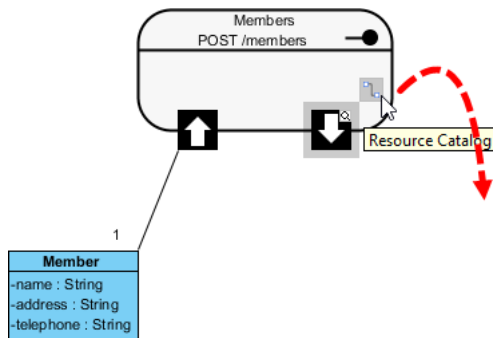
14. Enter the description of the three attributes. You can enter the description in the description editor or in the attribute specification window (Right-click on an attribute and select **Open Specification...** from the popup menu). The description entered will be presented in the API documentation.

Attribute	Description
name	The full name of the member.
address	The residential address of the member.
telephone	The telephone number of the member.

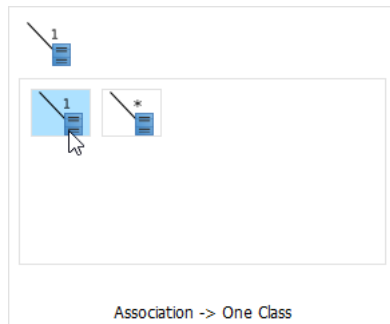


That's all for the request part. You are free to create a more complex structure by creating more associated classes, but normally you don't need to do this. Now, let's move on to the response part.

- Value(s) to be returned by server, if any, is modeled via the **Response Body**. Let's say a member ID will be returned by the server upon the creation of membership. Move your mouse pointer over the **REST Response Body** icon and drag out the **Resource Catalog** button at top right.

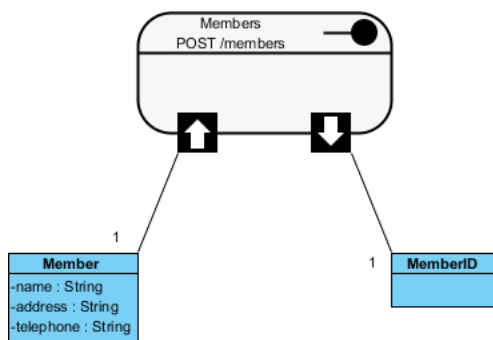


- Release the mouse button and select **Association -> One Class** from Resource Catalog.

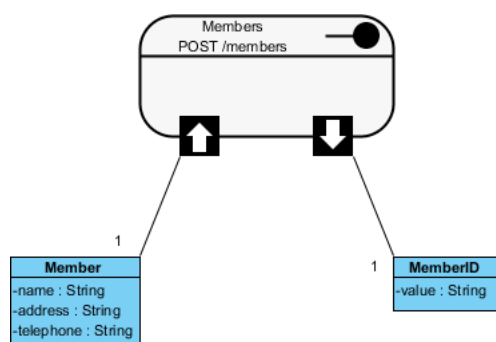


Note: If the service will return an array of objects, select **Association -> Many Class** instead.

- Name the class *MemberID*.



18. Add a String attribute *value*.



19. Enter the description of the value attribute: The member ID.

Specifying the Request and Response Header and Example

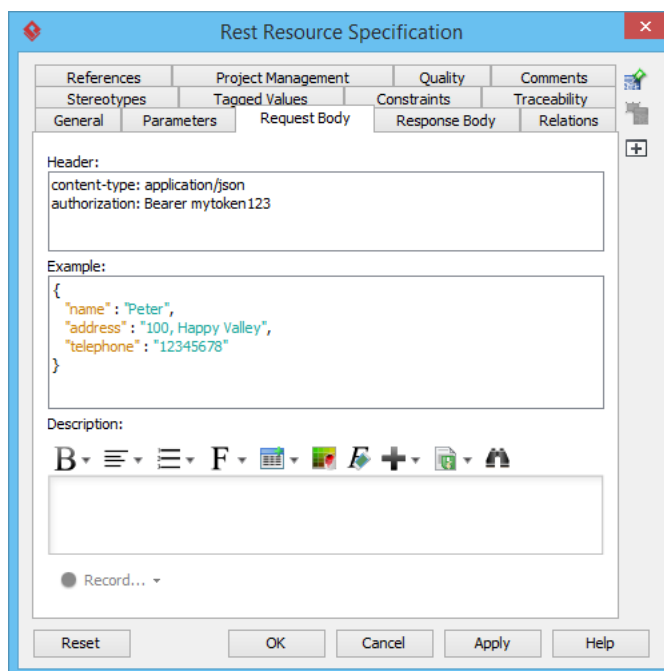
A HTTP message consists of a HTTP request line, a collection of header fields and an optional body. In order for consumers to access a REST Resource, you have to specify the request headers and request (body) example. The request header and example specified will be presented in the generated API documentation. Consumer can then follow the documentation in using the API.

1. Right-click on the *Members* REST Resource and select **Open Specification...** from the popup menu.
2. Open the **Request Body** tab.
3. Enter the **Header**:

```
content-type: application/
json
authorization: Bearer
mytoken123
```

4. Enter the **Example** in JSON:

```
{
  "name" : "Peter",
  "address" : "100, Happy Valley",
  "telephone" : "12345678"
}
```



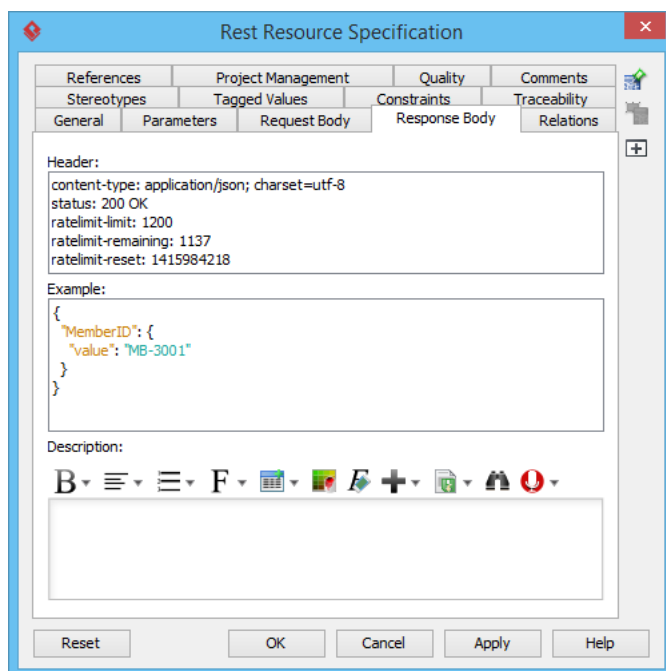
5. Open the **Response Body** tab.

6. Enter the **Header**:

```
content-type: application/json;  
charset=utf-8  
status: 200 OK  
ratelimit-limit: 1200  
ratelimit-remaining: 1137  
ratelimit-reset: 1415984218
```

7. Enter the Example in JSON:

```
{  
  "MemberID": {  
    "value": "MB-3001"  
  }  
}
```



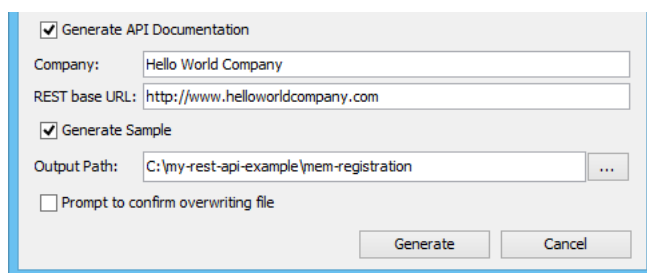
8. Click **OK** to confirm the changes.

Generating REST API and API Documentation

Once you have finished the design of your REST APIs, you can generate the API and the API documentation.

1. Select **Tools > Code > Generate REST API...** from the toolbar.
2. In the **REST API** window, keep **Provider** selected for **API Type**. By doing so, you will be able to generate API documentation as well as the server sample code that guides you in programming your service (logic).
3. Check the *Members* resource to generate API.
4. Check **Generate API Documentation** to generate the HTML files that shows how to use the selected REST Resource(s). Supposedly, you will publish the generated API documentation in your website, so that the consumers of your service can read through it to know how to access use your APIs.
5. Enter your company name, which will be presented in the API documentation.
6. Enter the base URL of your services.
7. Check **Generate Sample** to generate the source code that teaches you how to program your API. The sample code is rich and informative. Therefore, instead of programming from scratch, we strongly recommend you to generate the sample code and modify its content to fit your needs.

8. Enter the output path of the code.



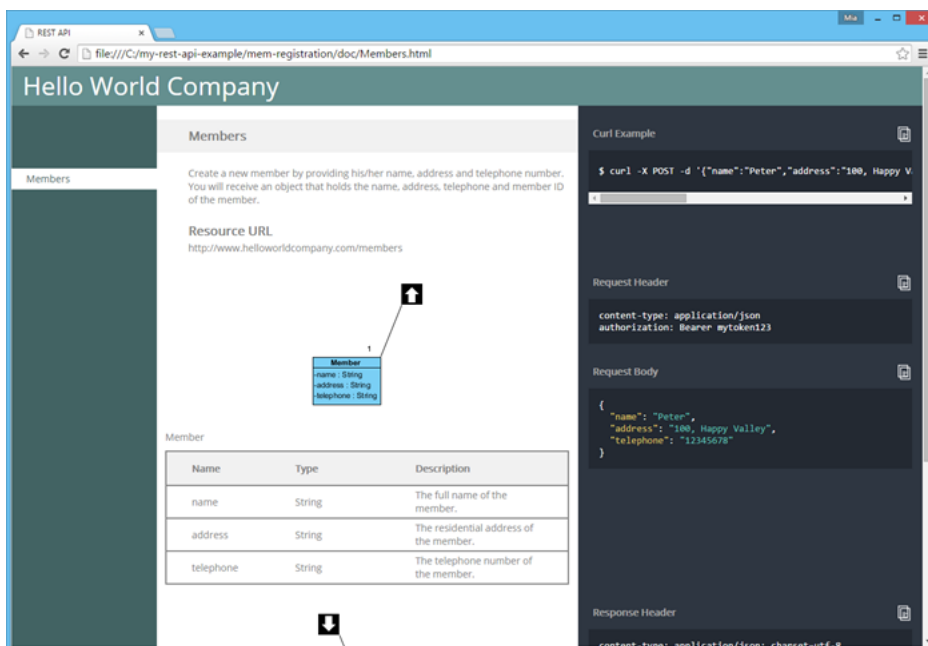
The screenshot shows a dialog box with the following fields and options:

- Generate API Documentation
- Company: Hello World Company
- REST base URL: http://www.helloworldcompany.com
- Generate Sample
- Output Path: C:\my-rest-api-example\mem-registration
- Prompt to confirm overwriting file
- Buttons: Generate, Cancel

9. Click Generate. The following folders are generated in the output directory.

Folder	Description
doc	The API documentation. You should publish the API documentation in your website, so that the consumers of your service can check the documentation to learn the API.
lib	In order for the generated code to work, the Google Gson library must be presented in your class path. To avoid any compatibility issues, please download library manually: https://code.google.com/p/google-gson/ and then place the file in the lib folder.
sample_src	The sample code of client and servlet. It shows you how to access as client and how to react to a request as provider. We strongly recommend you to copy the code and modify it by filling in your own service logic.
src	The source code of the communication model. Do not modify the file content or else the code may not be able to function properly.

10. Open the generated API documentation and take a look. The design (image), description of parameters, request and response header and example are presented in the document.



Resources

1. [Download REST API Example - Simple Registration Service.vpp](#)



Visual Paradigm home page
(<https://www.visual-paradigm.com/>)

Visual Paradigm tutorials
(<https://www.visual-paradigm.com/tutorials/>)