Using the Rock REST API

Mike Peterson

Spark Development Network

What is REST?

- Representational State Transfer
  - Representation = JSON
  - State Transfer
- Uses basic HTTP Verbs and Resources
  - GET, POST, PUT, DELETE, PATCH
  - Resources = Folders
    - https://www.server.com/api/People
    - https://www.server.com/api/Groups

Using REST to get data

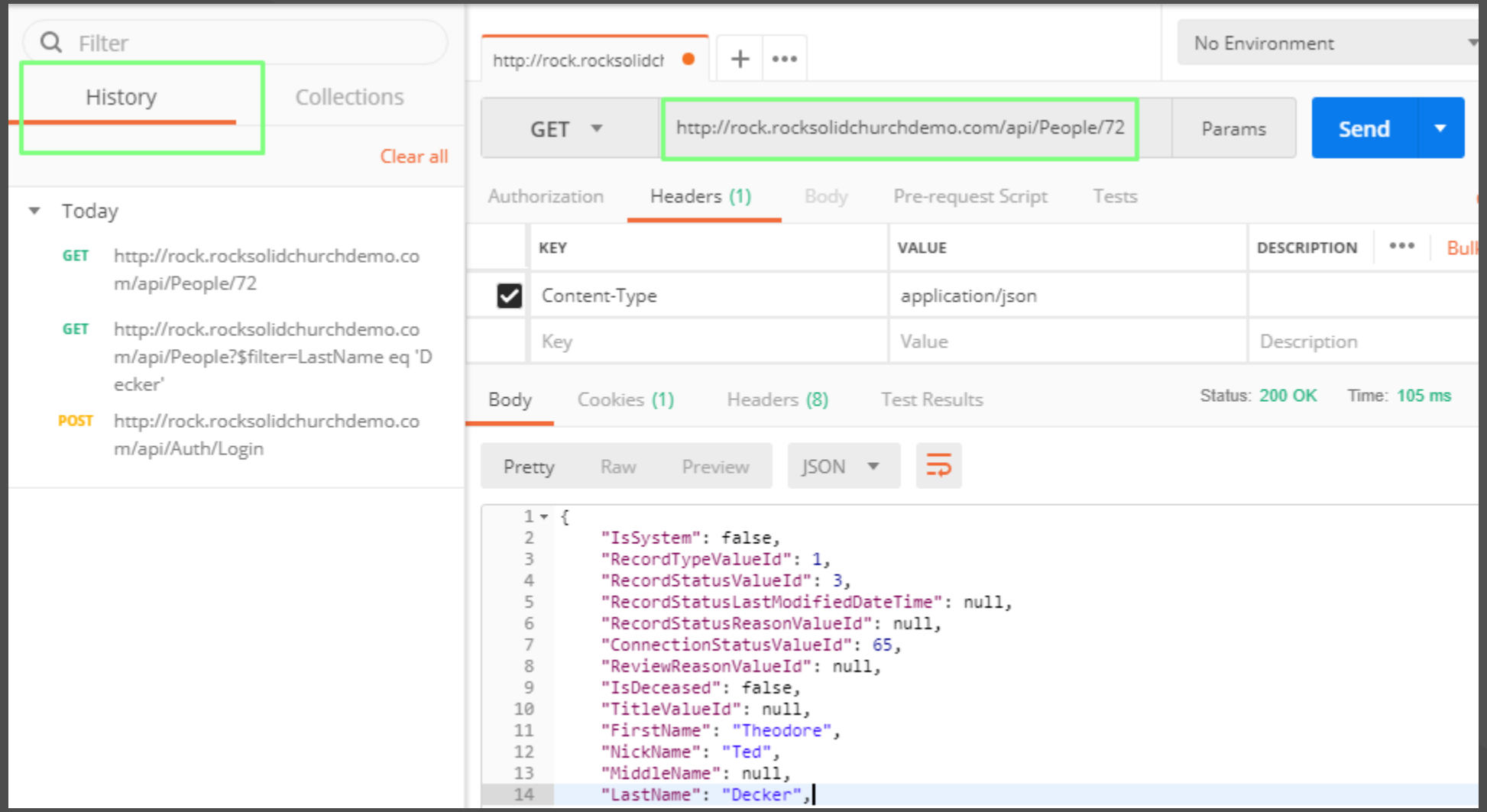
- Using REST to get Cat Facts
  - https://catfact.ninja
  - GET https://catfact.ninja/fact


```
{
  "fact": "Cats spend nearly 1/3 of their waking hours cleaning themselves.",
  "length": 64
}
```

Postman and Swagger

- Postman
  - https://www.getpostman.com/
- Swagger
  - http://rock.rocksolidchurchdemo.com/api/docs/index
  - Built into Rock
    - Power Tools > API Docs
  - Special Rock features
    - ?loadAttributes=simple
    - ODATA Helpers

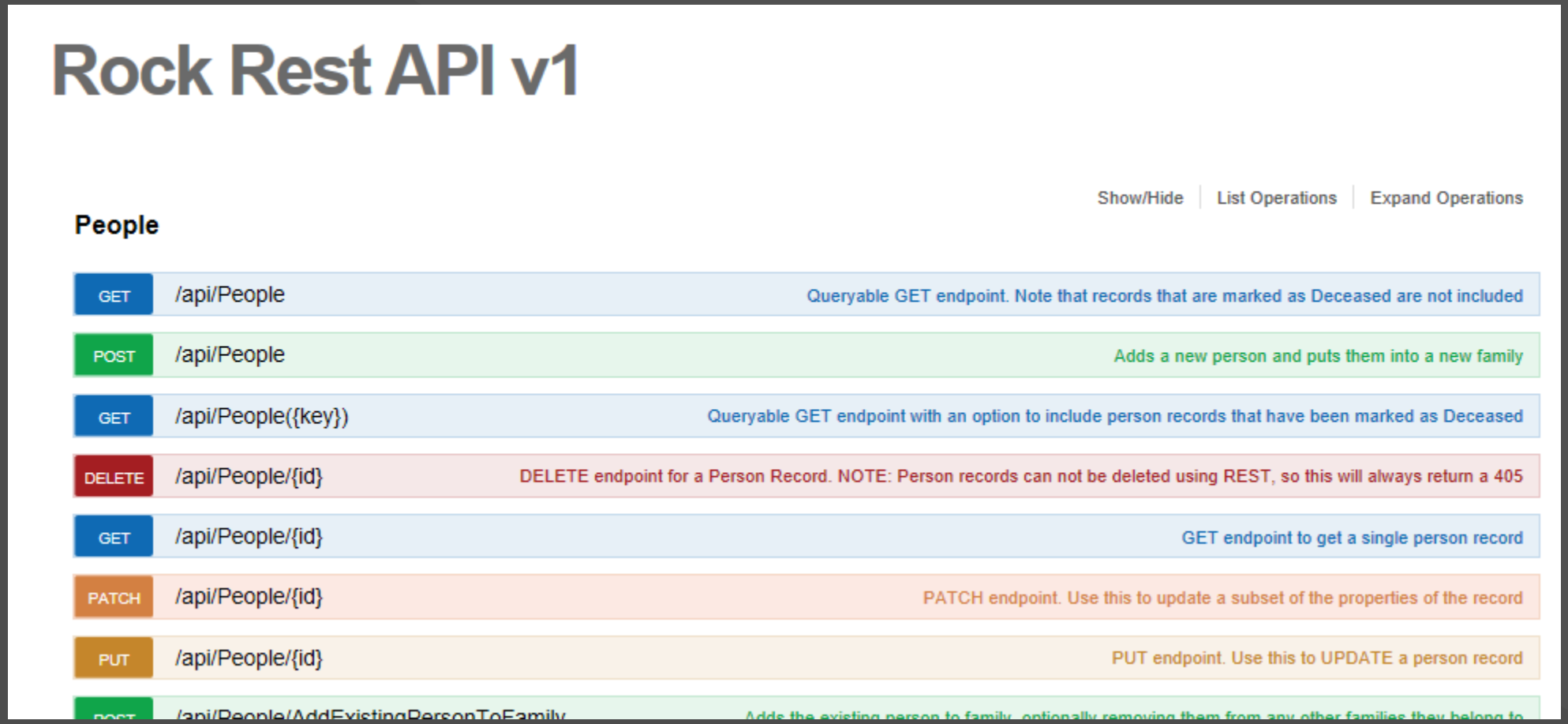
Postman


Filter
History
Collections
Clear all
Today
GET http://rock.rocksolidchurchdemo.com/api/People/72
GET http://rock.rocksolidchurchdemo.com/api/People?$filter=LastName eq 'Decker'
POST http://rock.rocksolidchurchdemo.com/api/Auth/Login
http://rock.rocksolidcl
No Environment
GET
http://rock.rocksolidchurchdemo.com/api/People/72
Params
Send
Authorization
Headers (1)
Body
Pre-request Script
Tests
KEY
VALUE
DESCRIPTION
Content-Type
application/json
Key
Value
Description
Body
Cookies (1)
Headers (8)
Test Results
Status: 200 OK
Time: 105 ms
Pretty
Raw
Preview
JSON
{
"IsSystem": false,
"RecordTypeValueId": 1,
"RecordStatusValueId": 3,
"RecordStatusLastModifiedDateTime": null,
"RecordStatusReasonValueId": null,
"ConnectionStatusValueId": 65,
"ReviewReasonValueId": null,
"IsDeceased": false,
"TitleValueId": null,
"FirstName": "Theodore",
"NickName": "Ted",
"MiddleName": null,
"LastName": "Decker",

Swagger - Directory

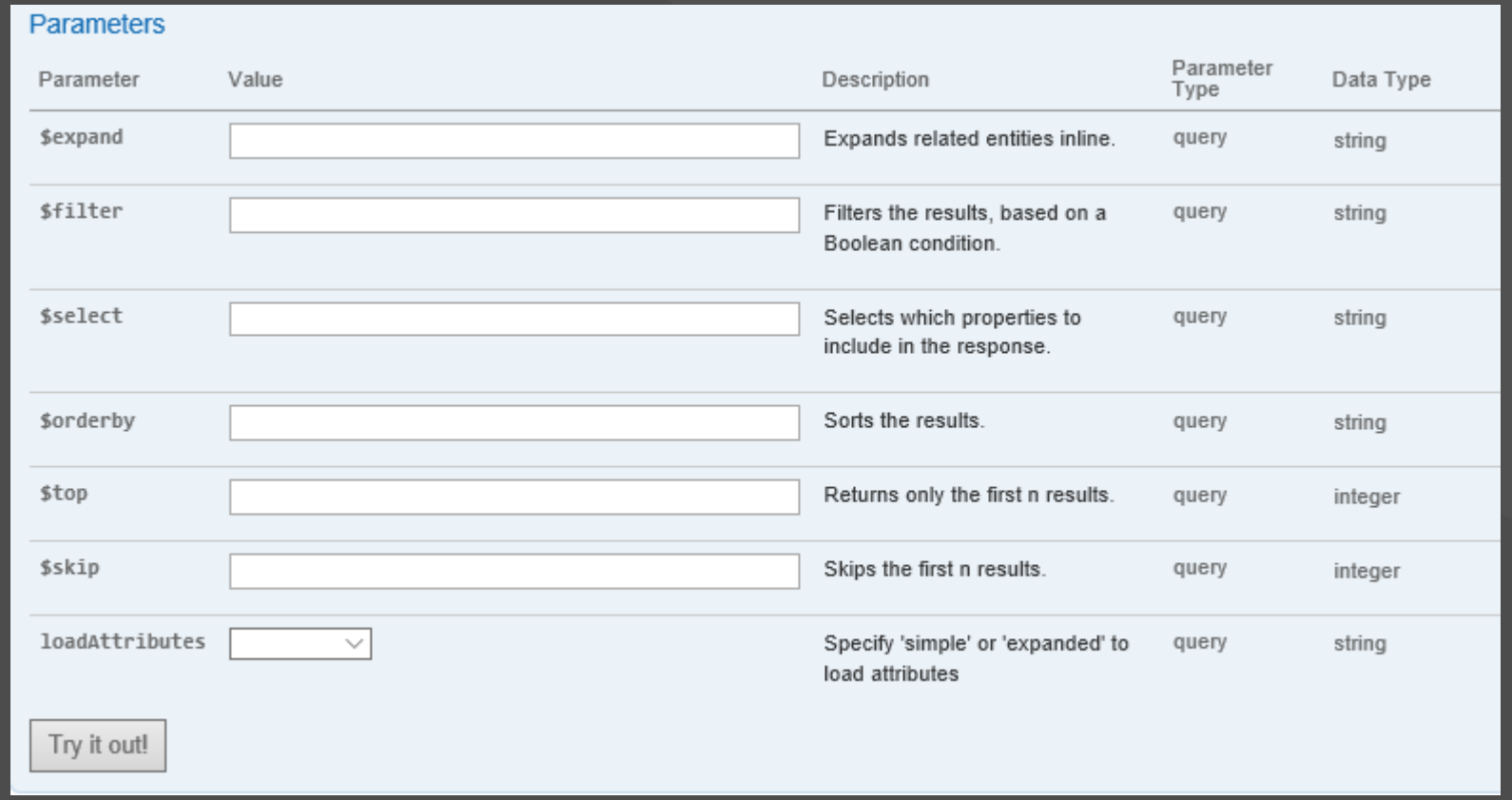
- List of core API Controllers
  - Attendances
  - People
  - FinancialTransactions
  - Groups
  - …
- Also includes plugin APIs

Swagger - Operations



Swagger – Try it out!

- ~/api/docs/index#!/People/GETapi_People



Parameters

Parameter	Value	Description	Parameter Type	Data Type
$expand		Expands related entities inline.	query	string
$filter		Filters the results, based on a Boolean condition.	query	string
$select		Selects which properties to include in the response.	query	string
$orderby		Sorts the results.	query	string
$top		Returns only the first n results.	query	integer
$skip		Skips the first n results.	query	integer
loadAttributes		Specify 'simple' or 'expanded' to load attributes	query	string

Try it out!

When to use the Rock REST API

- Non-Web Applications
- JavaScript in Custom Blocks
- Custom Website
- Mobile Apps

How to get authorized

- Three ways
  - Just be logged in
  - Auth Cookie
  - Authorization-Token
- Be aware of CORS (Cross-origin resource sharing)
  - This only comes in to play when client-side script from another domain tries to access the Rock REST API.
  - Rock lets you specify which domains are allowed
    - Add them in Rock at `Home > Security > REST CORS Domains`

.ROCK Auth Cookie

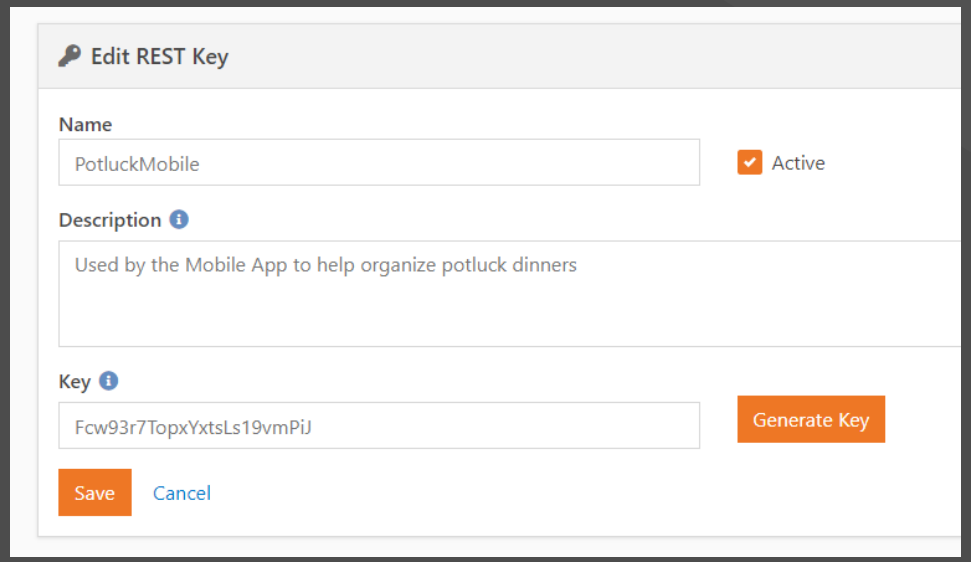
- POST ~/api/Auth/Login

```
{
    "Username" : "tdecker",
    "Password" : “ilovecindy"
}
```

- Response Cookie
  - .ROCK=FA2E7AC97D490A07B571F17F7EA7...

REST Key Authorization-Token

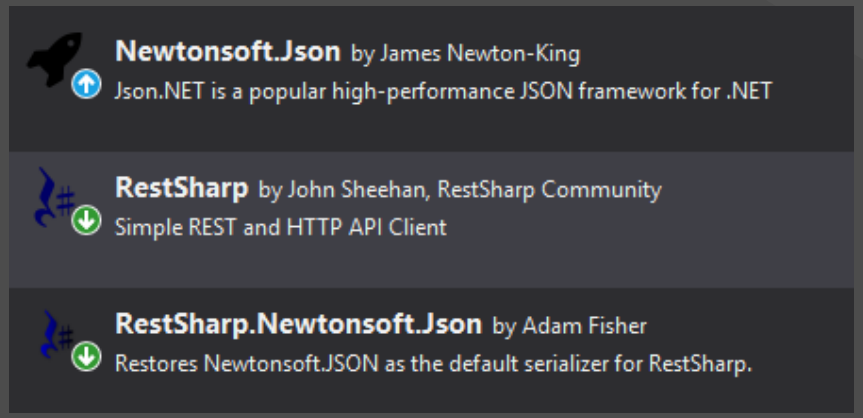
- Generate REST Login at `Home > Security > REST Keys`



- Set security for REST Key (Name)
- Use Key as Authorization-Token (Headers)

Writing a Rock REST client using C#

- Use RestSharp and JSON.NET NuGet Packages



- Rock.Client
  - https://github.com/SparkDevNetwork/Rock/tree/develop/Rock.Client
- Rock.Client.Models.shproj for Xamarin

Log into Rock using RestSharp

```
public RestClient GetRestClient( string userName, string password, Uri rockUrl )
{
    RestClient restClient = new RestClient( rockUrl );

    // specify NewtonsoftJsonSerializer as the serializer
    restClient.AddHandler( "application/json", RestSharp.Serializers.Newtonsoft.Json.NewtonsoftJsonSerializer.Default );

    // create a CookieContainer to keep the .ROCK auth cookie that is returned
    restClient.CookieContainer = new CookieContainer();

    RestRequest restRequest = new RestRequest( Method.POST );
    restRequest.RequestFormat = RestSharp.DataFormat.Json;
    restRequest.Resource = "api/auth/login";
    var loginParameters = new
    {
        UserName = userName,
        Password = password
    };

    restRequest.AddBody( loginParameters );

    // POST Login to Rock.  If it succeeds, the restClient will have a cookie.
    IRestResponse response = restClient.Post( restRequest );

    return restClient;
}
```

Get a Person using Rock's REST API

```
public Rock.Client.Person GetPerson( RestClient restClient, int personId, string attributeKey )
{
    IRestRequest personRequest
        = new RestRequest( $"api/People/{personId}" );

    IRestResponse<Rock.Client.Person> personResponse
        = restClient.Get<Rock.Client.Person>( personRequest );

    Rock.Client.Person person = personResponse.Data;

    return person;
}
```

```
var person = GetPerson( restClient, personId );
```

Get a Person's Attribute using REST

```
public string GetPersonAttribute( RestClient restClient, int personId, string attributeKey )
{
    IRestRequest personRequest
        = new RestRequest( $"api/People/{personId}?loadAttributes=simple" );

    IRestResponse<Rock.Client.Person> personResponse
        = restClient.Get<Rock.Client.Person>( personRequest );

    Rock.Client.Person person = personResponse.Data;

    return person.AttributeValues[attributeKey].Value;
}
```

```
var favoriteColor = GetPersonAttribute( restClient, personId, "FavoriteColor" );
```

Get a Person's Attribute using OData

```
public string GetPersonAttributeUsingOData( RestClient restClient, int personId, string attributeKey )
{
    // make sure also specify the EntityType since the same key could be used for other types of entitys
    var odataFilter = $"Attribute/Key eq '{attributeKey}' and Attribute/EntityType/Guid eq guid'{Rock.Client.SystemGuid.EntityType.PERSON}' and EntityId eq {personId}";

    IRestRequest attributeValueRequest
        = new RestRequest( $"api/AttributeValues?$filter={odataFilter}" );

    // We are using a Queryable endpoint, so the return datatype will be a List<Rock.Client.Attribute>
    IRestResponse<List<Rock.Client.AttributeValue>> attributeValueResponse = restClient.Get<List<Rock.Client.AttributeValue>>( attributeValueRequest );

    List<Rock.Client.AttributeValue> results = attributeValueResponse.Data;

    // There should only be one result (or none if the person doesn't have this attribute), so just fetch the First
    return results.FirstOrDefault()?.Value;
}
```

```
var favoriteColor = GetPersonAttributeUsingOData( restClient, personId, "FavoriteColor" );
```

OData

Rock’s REST API and OData

- Rock supports OData v3
- Lets you create powerful REST queries
- Built upon REST
- Standardized
- Easy to get started with
- Learn as you go

OData Basics

- $filter
- $orderby
- $top and $skip
- $select and $expand

OData vs SQL

OData

- $select=FirstName,LastName
- ~/api/People?
- $filter=LastName eq ‘Decker’
- $orderby=LastName,FirstName
- $skip=20&$top=10

SQL

- SELECT [FirstName],[LastName]
- FROM [Person]
- WHERE [LastName] = ‘Decker’
- ORDER BY [LastName],[FirstName]
- OFFSET 20 ROWS FETCH NEXT 10 ROWS ONLY

OData Filter Expressions

SQL	OData	Description/Example
=, !=	eq, ne	Equal, Not Equal
>, >=, <, <=	gt, ge, lt, le	Great Than, Greater Than or Equal, Less Than, Less Than or Equal
and, or	and, or	Logical operators
LIKE	endswith,startswith	$filter=endswith(LastName, ‘ecker’) and startswith(NickName, ‘Te’)

Parameter DataTypes

Datatype	Value	Example
string	Use single-quotes	`$filter=LastName eq ‘Decker’`
numeric	No quotes	`$filter=Amount gt 5.00`
Guid	guid prefix and quotes	`$filter=Guid eq guid’AA8732FB-2CEA-4C76-8D6D-6AAA2C6A4303’`
Bool	true, false	`$filter=IsDeceased eq true`
Datetime	datetime prefix and quotes	`$filter=TransactionDateTime ge datetime'2014-10-22T14:35:00’`
Date	datetime prefix and quotes	`$filter=BatchDate ge datetime'2014-10-01’`

Use ISO-8601 standard when specifying datetime values.
See http://en.wikipedia.org/wiki/ISO_8601

Applying $filters

GET http://rock.rocksolidchurchdemo.com/api/people?

- $filter=BirthDate lt datetime'1971-11-05'
- $filter=Guid eq guid'1ea811bb-3118-42d1-b020-32a82bc8081a'
- $filter=IsEmailActive eq true
- $filter=TitleValueId ne null
- $filter=RecordStatusValue/Value eq 'Active'
- $filter=Photo/ModifiedDateTime ge datetime'2014-10-22’

GET http://rock.rocksolidchurchdemo.com/api/groupmembers?

- $filter=GroupRole/Name eq 'Adult' and Person/LastName eq 'Decker'

$filters with lambdas (all, any)

- `GET ~/api/books?$filter=Versions/any(version: version/VersionDefinedValueId eq 205)`
- `GET ~/api/people?$filter=PhoneNumbers/any() and PhoneNumbers/all(a:startswith(a/Number, '602'))`

$orderby

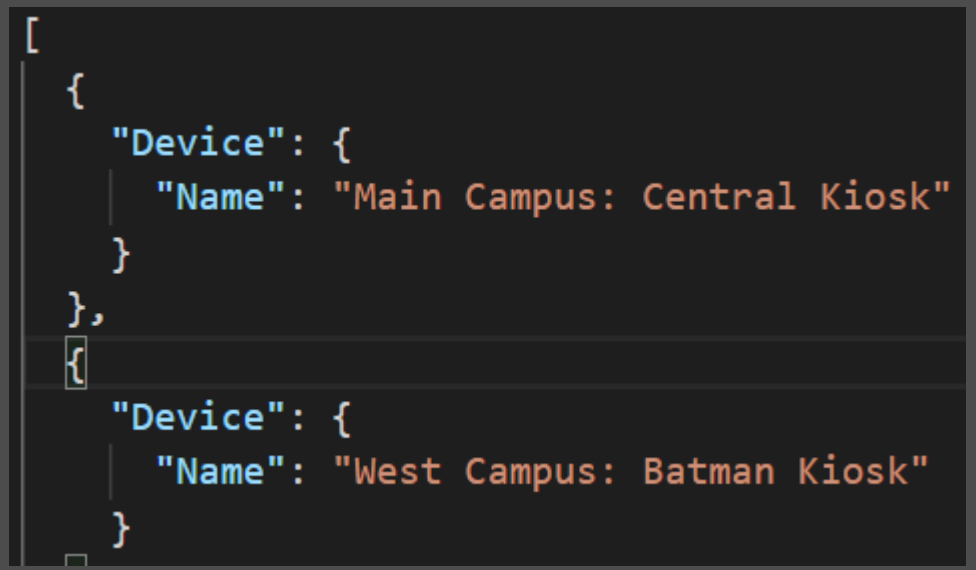
- `/api/people?$orderby=LastName`
- `/api/people?$orderby=LastName,NickName`
- `/api/people?$orderby=BirthDate desc,LastName`

$top and $skip

- /api/Attendances?$top=10
- /api/Attendances?$skip=20&$top=10
- /api/Attendances?$orderby=StartDateTime&$top=10

$select and $expand

- api/Attendances?$select=CampusId
- api/Attendances?$select=CampusId,ScheduleId
- api/Attendances?$expand=Device&$select=Device/Name



$filter examples

- $filter=LastName eq 'Decker'
- $filter=endswith(LastName, ‘ecker’) and startswith(NickName, ‘Te’)
- $filter=Birthyear gt 1971 and Birthyear lt 1981
- $filter=Guid eq guid’1ea811bb-3118-42d1-b020-32a82bc8081a’
- $filter=TransactionDateTime ge datetime'2014-10-22T14:35:00’
- $filter=Amount gt 5.00
- $filter=RecordStatusValue/Value eq ‘Active’
- $filter=Photo/ModifiedDateTime ge datetime’2014-10-22‘
- $filter=PhoneNumbers/any() and PhoneNumbers/all(a:startswith(a/Number, '602'))

Swagger and OData

People

GET /api/People

Parameters

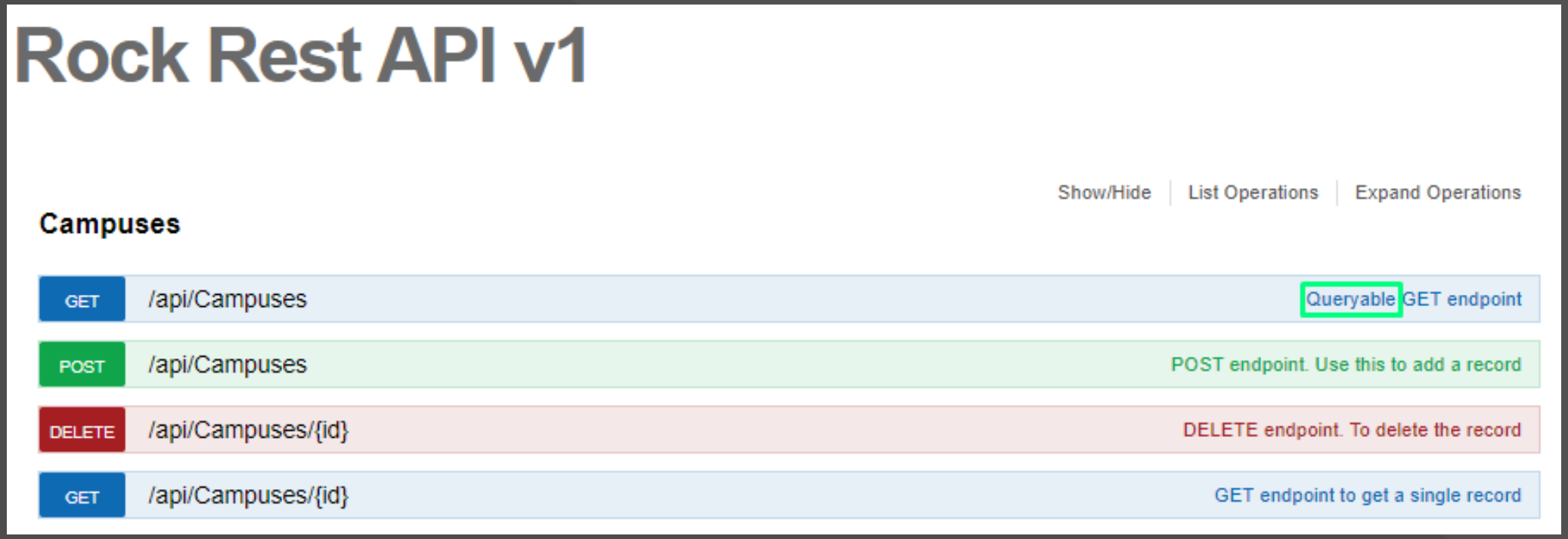
Parameter	Value
$expand	ConnectionStatusValue
$filter	
$select	Id, FirstName, LastName, ConnectionStatusValue/Value

Response Body


```
},
{
  "LastName": "Decker",
  "FirstName": "Theodore",
  "Id": 72,
  "ConnectionStatusValue": {
    "Value": "Member"
  }
},
{
  "LastName": "Decker",
  "FirstName": "Cynthia",
  "Id": 73,
  "ConnectionStatusValue": {
    "Value": "Member"
  }
}
```

OData Gotchas

- OData only works on Queryable Endpoints



OData Gotchas

- Non-Database fields don’t work

```
public partial class Person : Model<Person>, IRockIndexable
{
```

```
[DataMember]
[NotMapped]
public virtual string FullName
{
    get
    {
        // Use the SuffixValueId and DefinedValue cache instead of referencing SuffixValue property so
        // that if FullName is used in datagrid, the SuffixValue is not lazy-loaded for each row
        return FormatFullName( NickName, LastName, SuffixValueId );
    }

    private set
    {
        // intentionally blank
    }
}
```

OData and Model Attributes

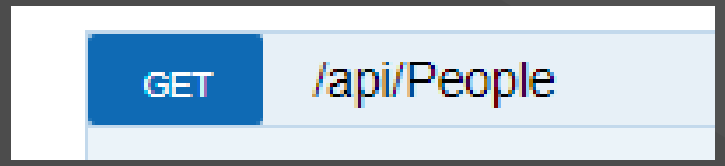
- Attributes don’t support OData. They aren’t database fields.

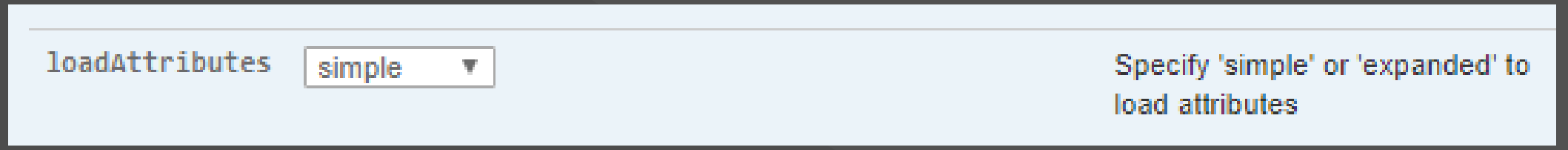
```
/// <summary>
/// List of attributes associated with the object. This property will not include the attribute values.
/// The <see cref="AttributeValues"/> property should be used to get attribute values. Dictionary key
/// is the attribute key, and value is the cached attribute
/// </summary>
/// <value>
/// The attributes.
/// </value>
[NotMapped]
[DataMember]
[LavaIgnore]
public virtual Dictionary<string, AttributeCache> Attributes { get; set; }

/// <summary>
/// Dictionary of all attributes and their value. Key is the attribute key, and value is the associated attribute value
/// </summary>
/// <value>
/// The attribute values.
/// </value>
[NotMapped]
[DataMember]
[LavaIgnore]
public virtual Dictionary<string, AttributeValueCache> AttributeValues { get; set; }
```

REST and Model Attributes

- The Rock REST API does have some support for Attributes
- Attributes can be a little expensive (especially if loading more than one record) and are not included by default
- ?loadAttributes=simple
  - Less expensive, usually all you need
- ?loadAttributes=expanded
  - Includes all fields, usually much more info than you need
- ?loadAttributes=true is the same as simple
- ~/api/AttributeValues supports OData, but can be tricky

Attributes and Swagger

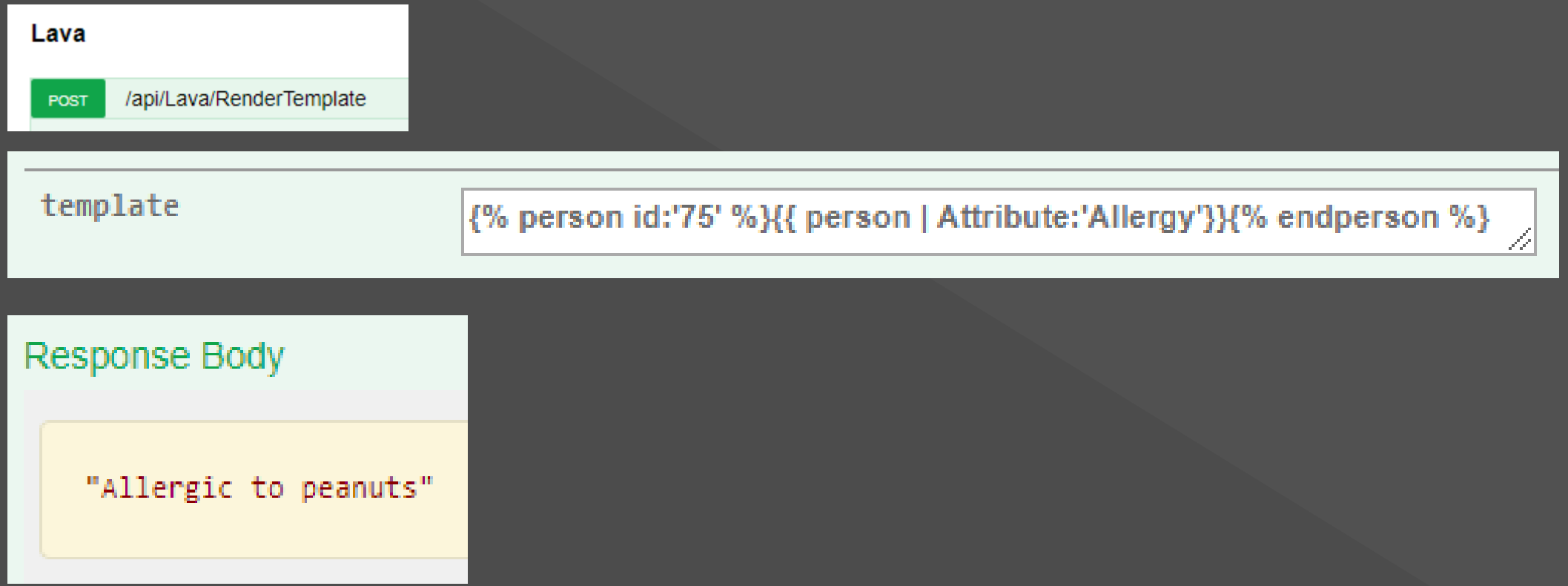




Response Body


```
"AttributeValues": {
  "Allergy": {
    "AttributeId": 676,
    "EntityId": 75,
    "Value": "Allergic to peanuts"
  },
  "LegalNotes": {
```

Attributes, Swagger and Lava



https://www.rockrms.com/lava/remote-lava