Introduction to the SharePoint REST API

Rob Windsor
rob@robwindsor.com
@robwindsor

About Me

Rob Windsor

.NET/Microsoft 365 developer, trainer, author
Microsoft MVP | Microsoft 365 Development

Twitter: https://twitter.com/robwindsor
LinkedIn: https://www.linkedin.com/in/rwindsor
Blog: https://robwindsor.hashnode.dev
YouTube: https://www.youtube.com/@RobWindsor
GitHub: https://github.com/rob-windsor

SharePoint REST API

API used when building remote applications

What is the REST API in SharePoint

- Data-centric web services **based on** the Open Data Protocol (OData)
  - More on OData later
- Each resource or set of resources is addressable
  - http://<site url>/_api/web
  - http://<site url>/_api/web/lists
  - http://<site url>/_api/web/lists/getByTitle('Customers')
- Operations on resources map to HTTP Verbs
  - GET, PUT, POST, DELETE, …
- Results from service returned in AtomPub (XML) or JavaScript Object Notation (JSON) format

SharePoint REST API History

SharePoint 2010

- Initial REST API added
- /_vti_bin/ListData.svc
- Exposed CRUD operations on list data

SharePoint 2013

- REST API expands and evolves
- ListData.svc deprecated
    - Still available for backwards compatibility
- RESTful operations added to /_vti_bin/Client.svc
- /_api added as an alias for /_vti_bin/Client.svc

SharePoint REST API Coverage

- Sites, Webs, Features, Event Receivers, Site Collections
- Lists, List Items, Fields, Content Types, Views, Forms, IRM
- Files, Folders
- Users, Roles, Groups, User Profiles, Feeds
- Search

OData Queries

Queries represented by query strings added to resource URL

Option	Example
$select	_api/Web/Lists?$select=Title,ItemCount
$filter	_api/Web/Lists?$filter=(Hidden eq false)
$orderby	_api/Web/Lists?$orderby=ItemCount desc
$skip, $top	_api/Web/Lists?$skip=25&$top=10
$expand	_api/Web/Lists?$expand=Fields

Full documentation: http://www.odata.org/documentation/odata-v2-documentation/uri-conventions/#4_Query_String_Options

SharePoint Framework and the REST API

SPFx implements calls to SharePoint REST API via the `SPHttpClient`

Available from the existing context:

- `this.context.spHttpClient.get()`
- `this.context.spHttpClient.post()`

Based on the existing `HttpClient` API

Handles the authentication and default configuration settings:

- Authorization HTTP header
- OData v4
- Minimal metadata returned

Reading List Items

```
private async _getListItems(): Promise<ICountryListItem[]> {
  const response = await this.context.spHttpClient.get(this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items?$select=Id,Title`,
    SPHttpClient.configurations.v1);

  if (!response.ok) {
    const responseText = await response.text();
    throw new Error(responseText);
  }

  const responseJson = await response.json();
  return responseJson.value as ICountryListItem[];
}
```

CRUD Operations with SPFx and the REST API

Use the SharePoint Framework `SPHttpClient`'s `post()` method to write to the SharePoint REST API

Some operations require additional HTTP headers:

- `IF-MATCH`: specify version of the item on the server to be updated / deleted
- `X-HTTP-Method`: specify `MERGE` or `DELETE` in update & delete operations

Some operation require specific data in the payload body

- `@odata.type`: specify the type of data being written to the list when creating

Creating List Items

```
private async _addListItem(): Promise<SPHttpClientResponse> {
  const request: any = {};
  request.body = JSON.stringify({
    Title: new Date().toUTCString()
  });

  const endpoint = this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items`;

  return this.context.spHttpClient.post(
    endpoint,
    SPHttpClient.configurations.v1,
    request);
}
```

Updating List Items

Should specify the type of operation to perform

- Default behavior is to set properties to supplied values
  - Omitted properties are set to `null`
- Override behavior using the `MERGE` method
- Set using the `X-HTTP-Method` header

Specify the version of the item to update

- When updating items, can specify “only update the item on the server if it is version X”
- Ensures you aren’t overwriting someone else’s changes unknowingly
- Enforced with the `IF-MATCH` header & `etag`’s

Updating List Items

```
private async _updateListItem(): Promise<SPHttpClientResponse> {
  const getEndpoint: string = this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items?` +
    `$select=Id,Title&$filter=Title eq 'United States'`;

  const getResponse = await this.context.spHttpClient.get(getEndpoint, SPHttpClient.configurations.v1);
  const responseJson = await getResponse.json();
  const listItem: ICountryListItem = responseJson.value[0];

  listItem.Title = 'USA';
  const request: any = {};
  request.headers = {
    'X-HTTP-Method': 'MERGE',
    'IF-MATCH': (listItem as any)['@odata.etag']
  };
  request.body = JSON.stringify(listItem);

  const postEndpoint: string = this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items(${listItem.Id})`;

  return this.context.spHttpClient.post(postEndpoint, SPHttpClient.configurations.v1, request);
}
```

Deleting List Items

Should specify the type of operation to perform

- Underlying `fetch` API only contains `post()` method; not `delete()`
- Override behavior using the `DELETE` method
- Set using the `X-HTTP-Method` header

Specify the version of the item to delete

- When updating items, can specify “only update the item on the server if it is version X”
- Enforced with the `IF-MATCH` header & `etag`’s
- Decide: does it matter if the version is different?
- If not, use `IF-MATCH = '*'`

Deleting List Items

```
private async _deleteListItem(): Promise<SPHttpClientResponse> {
  const getEndpoint = this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items?` +
    `$select=Id,Title&$orderby=ID desc&$top=1`;

  const getResponse = await this.context.spHttpClient.get(getEndpoint, SPHttpClient.configurations.v1);
  const responseJson = await getResponse.json();
  const listItem: ICountryListItem = responseJson.value[0];

  const request: any = {};
  request.headers = {
    'X-HTTP-Method': 'DELETE',
    'IF-MATCH': '*'
  };
  request.body = JSON.stringify(listItem);

  const postEndpoint = this.context.pageContext.web.absoluteUrl +
    `/_api/web/lists/getbytitle('Countries')/items(${listItem.Id})`;

  return this.context.spHttpClient.post(postEndpoint, SPHttpClient.configurations.v1, request);
}
```

Differences between SharePoint document libraries and SharePoint lists

Document libraries are like SharePoint lists in many ways

Document libraries display contents as files instead of items

When creating a file with the SharePoint REST API, must include the site and folder where the file will be created

Use the `Files` endpoint to work with files in the SharePoint REST API

Uploading files

Remember to:

- Include the file’s contents in the request body
- Include the length of the file’s body in the `content-length` HTTP request header

```
const endpoint = `https://site/_api/web/lists/GetByTitle('Documents')/RootFolder/Files/add(` +
  `overwrite=true, url='${fileName}')`;

const options: ISPHttpClientOptions = {
  headers: { 'CONTENT-LENGTH': fileData.byteLength.toString() }, body:fileData };

const response = await this.context.spHttpClient.post(
  endpoint,
  SPHttpClient.configurations.v1,
  options);
```

Resources

Overview of the SharePoint Framework
https://learn.microsoft.com/sharepoint/dev/spfx/sharepoint-framework-overview

Get to know the SharePoint REST service
https://learn.microsoft.com/sharepoint/dev/sp-add-ins/get-to-know-the-sharepoint-rest-service

Connect to SharePoint APIs
https://learn.microsoft.com/sharepoint/dev/spfx/connect-to-sharepoint