REST Base Service

We have a base REST API service that we can leverage in the entire app stack.

It encapsulates the ceremony of dealing with backend calls, and provides us with a clean set of generic, async methods.

It's located in: lib-oga-webui-sharedkernel.

To use in a component or service, import it with this:

```
import { RestBaseService } from 'lib-oga-webui-sharedkernel';
```

Add it to your constructor or a private injection:

```
constructor(private _restsvc: RestBaseService)
{
  ...
}
```

Now, you can use its various calls.

Each call has these properties:

- Uses one of the four verbs: GET, POST, PUT, DELETE.
- Accepts the endpoint path fragment as string.
  Composes the url from the app origin and the api_BasePath property from environment.ts.
- Accepts an optional header instance, of type HttpHeaders.
- Returns a Promise.
  Use await to call the method.
- Returns a generic web response object. See this for details: Generic Web Reply).
  The concrete type must be specified when calling the method.

API URL

Each method composes the url based on the app's origin and the api_BasePath property from environment.ts.

The url is composed of this form:

`${origin}${this.env.api_BasePath}/${urlpath}`

The origin will be the app's origin, unless overridden.

> You can override the origin used by setting the public property, OriginOverride, of the REST service instance before making the call.

The api_BasePath is defined in your app's environment.ts file.

The urlpath is what your calling method passes in.
It should include any query parameters, already urlencoded.

Example Usage

> NOTE: The endpoint passed to the method does NOT need to include a leading '/'. This is appended for you.

GET - Returning a void:

```
let res = await this._restsvc.GetRequest<void>("VoidTests_V1/Get/AcceptVoid/ReturnVoid");
if(!res || res === null || !res.IsReturnGood())
{
  // Call failed.
}
// Call returned success.
```

GET - Returning a type:

```
let res = await this._restsvc.GetRequest<sometype>("API_endpoint/Get/Somecall");
if(!res || res === null || !res.IsReturnGood())
{
  // Call returned an error or failed.
}
// Call returned success.
```

POST - Accepting a type and returning a type:

```
let res = await this._restsvc.PostRequest<requesttype,responsetype>("API/ENDPOINT", requestdata);
if(!res || res === null || !res.IsReturnGoodDataExists())
{
  // The call failed, or returned no data.
```

```
}
// It succeeded, and we have data.
```

PUT - returning a type:

```
let res = await this._restsvc.PutRequest<requesttype,responsetype>("API/ENDPOINT", requestdata);
if(!res || res === null || !res.IsReturnGoodDataExists())
{
  // The call failed, or returned no data.
}
// It succeeded, and we have data.
```

DELETE - returning a void:

```
let res = await this._restsvc.DeleteRequest<responsetype>("API/ENDPOINT");
if(!res || res === null || !res.IsReturnGood())
{
  // The call failed, or returned no data.
}
// It succeeded, and we have data.
```

Revision #1
Created 30 March 2025 19:21:01 by glwhite
Updated 30 March 2025 20:33:35 by glwhite