

Documentum REST Extensibility Tutorial (4): Create Object

Previous: [Documentum REST Extensibility Tutorial \(3\): Explore The Sample Project](#)

This document is for the continuous series for Documentum REST Extensibility tutorials. Previously, we explored each module of the sample project. In this document, we will add a HTTP POST method to create alias sets.

Overview

As the previous tutorial showed, there are two custom resources in the sample project.

- alias sets collection resource
- alias set object resource

As a general REST design principal, object creation is usually implemented as **adding objects to the object collection**. Therefore, we will consider adding a new method onto *AliasSetCollectionController.java*.

And following the HTTP method semantics, we usually use **POST** to create new objects.

Now, please open the existing *acme-rest* project at `<SDK_ROOT>/maven-kit/generated/acme-rest/`. You can use an Java IDE to open the project as a Maven project or open files in the text editor directly.

Implementation

Open the Java class *AliasSetCollectionController.java*, add a new method within the class entity. Below is the complete Java method for the alias set creation. We will explain every piece of the code.

```
@Autowired private SysObjectManager sysObjectManager; @RequestMapping(method = RequestMethod.POST, produ
```

Line 01 ~ 02:

We will use Documentum Core REST existing library to call DFC to create the alias set object. So here we declare a *SysObjectManager* instance with Spring annotation *@Autowired*. We don't need to create the instance for this member. Spring will assign the existing implementation bean for this interface to the controller instance.

Line 04 ~ 12 :

We annotate the controller method with Spring annotation *@RequestMapping*. HTTP method and supported media types should be specified in the request mapping.

Line 13:

We annotate the controller method with Spring annotation *@ResponseBody*. This annotation makes the method return type *AliasSet* serializable as XML or JSON message.

Line 14:

We annotate the controller method with Spring annotation *@ResponseStatus(HttpStatus.CREATED)*. This annotation makes the response status as 201.

Line 15:

We annotate the controller method with Documentum REST annotation *@ResourceViewBinding*. This annotation binds the method return type *AliasSet* to a resource view *AliasSetView*. The view class resolves the link relations of the alias set object representation in the response body.

Line 16 ~ 21:

Here we declare some method parameters with annotations like `@PathVariable`, `@RequestBody`, `@TypedParam` and `@RequestUri`.

- **@PathVariable("repositoryName")** - read path segment value for "repositoryName" from the request URI
- **@RequestBody AliasSet** - deserialize the request body as an AliasSet instance
- **@TypedParam SingleParam** - bind some common Documentum REST URI query parameters into a single Java parameter
- **@RequestUri UriInfo** - read the request URI info

Line 22 ~ 25:

Validate the object type of *dm_alias_set*.

Line 26:

Call *SysObjectManager* to create the alias set object.

Line 27:

Set the parameter for resource rendering. This parameter is common for any object creation. It makes the view implementation know where the object comes from, so that a *Location* response header and *self* link relation are set correctly in the response.

Line 28:

Render the model instance as the output. This will call the parent method to load the annotated resource view (*AliasSetView*) dynamically to build links.

Deploy and Test

Run Maven goal *mvn clean install* to rebuild the project and redeploy it to the web container again.

Open a REST client debugging tool like PostMan or DHC to test the POST method.

Request:

POST http://localhost:8080/acme-rest/repositories/REPO/alias-sets HTTP/1.1 Authorization: Basic ZG1hZG1pbjpwYXNzd29yZA==

Response:

HTTP/1.1 201 Created Location: http://localhost:8080/acme-rest/repositories/REPO/alias-sets/6600000580000515 Content-Type: a

We are done here. In next post, we will add additional operations to update and delete alias sets in the sample project.

Next: [Documentum REST Extensibility Tutorial \(5\): Update and Delete Object](#)

[Learn more about Documentum REST Services >>](#)