Connexion Documentation

Release 0.5

Zalando SE

November 12, 2015

Contents

Connexion is a framework on top of Flask to automagically handle your REST API requests based on Swagger 2.0 Specification files in YAML.

Contents:

CHAPTER 1

Quickstart

1.1 How to use

Put your API YAML inside a folder in the root path of your application (e.g `swagger\`) and then do

```
import connexion

app = connexion.App(__name__, specification_dir='swagger/')
app.add_api('my_api.yaml')
app.run(port=8080)
```

1.2 Parametrization

Connexion uses Jinja2 to allow the parametrization of specifications.

The specification arguments can be defined globally for the application or for each specific api:

```
app = connexion.App(__name__, specification_dir='swagger/', arguments={'global': 'global_value'})
app.add_api('my_api.yaml', arguments={'api_local': 'local_value'})
app.run(port = 8080)
```

If a value is provided both globally and on the api then the api value will take precedence.

1.3 Routing

Connexion uses the `OperationId` from each Operation Object to identify which function should handle each url.

For example:

```
paths:
  /hello_world:
    post:
      operationId: myapp.api.hello_world
```

If you provided this path in your specification POST requests to `http://MYHOST/hello_world` would be handled by the function `hello_world` in `myapp.api`.

Additionally you can also define a `basePath` on the top level of the API specification, which is useful for versioned APIs. If you wanted to serve the previous endpoint from `http://MYHOST/1.0/hello_world` you could do:

```
basePath: /1.0

paths:
  /hello_world:
    post:
      operationId: myapp.api.hello_world
```

Other alternative if you don't want to include the base path in your specification is provide the base path when adding the API to your application:

```
app.add_api('my_api.yaml', base_path='/1.0')
```

1.4 Request Handling

Connexion validates incoming requests for conformance with the schemas described in swagger specification.

Request parameters will be provided to the handler functions as keyword arguments if they are included in the function's signature, otherwise body parameters can be accessed from `connexion.request.json` and query parameters can be accessed from `connexion.request.args`.

1.5 Response Serialization

By default and if the specification defines that a endpoint produces only json, connexion will automatically serialize the return value for you and set the right content type in the HTTP header. If the endpoint produces a single non json mimetype then connexion will automatically set the right content type in the HTTP header.

1.6 Error Handling

By default connexion error messages are JSON serialized according to Problem Details for HTTP APIs.

Application can return errors using `connexion.problem`.

1.7 Swagger Json

Connexion makes the Swagger specification in json format available from `swagger.json` in the base path of the api.

1.8 Swagger UI

The Swagger UI for an API is available, by default, in `{base_path}/ui/` where `base_path` is the base path of the api.

You can disable the swagger ui either at application level:

```
app = connexion.App(__name__, port = 8080, specification_dir='swagger/', swagger_ui=False)
app.add_api('my_api.yaml')
```

You can also disable it at api level:

```
app = connexion.App(__name__, port = 8080, specification_dir='swagger/')
app.add_api('my_api.yaml', swagger_ui=False)
```

Likewise, you can configure the filesystem and URL paths to the Swagger UI documentation:

```
app = connexion.App(__name__, port = 8080, specification_dir='swagger/')
app.add_api('my_api.yaml', swagger_path='/path/to/swagger-ui', swagger_url='doc')
```

1.9 Server Backend

By default connexion uses the default flask server but you can also use Tornado as the http server, to do so set server to `tornado`:

```
import connexion

app = connexion.App(__name__, port = 8080, specification_dir='swagger/', server='tornado')
```

CHAPTER 2

Security

2.1 Authentication and Authorization

If the specification includes a Oauth2 Security Definition compatible with the Zalando Greendale Team's infrastructure connexion will automatically handle token validation and authorization for operations that have Security Requirements. One main difference between the usual Oauth flow and the one connexion uses is that the API Security Definition **must** include a 'x-tokenInfoUrl' with the url to use to validate and get the token information.

Connexion expects to receive the Oauth token in the `Authorization` header field in the format described in RFC 6750 section 2.1.

For authenticated endpoints connexion will add a `user` and `token_info` properties to `connexion.request` containing the user name and the full token info of the request.