Exercise 1: Making REST Requests

You've been very patient watching videos and taking quizzes. By now, you should have a sense of how REST works. Now, let's get some practice making actual REST requests.

We are going to use a sample API that was created using a technology called Swagger. Swagger is an example of an automated documentation tool. Swagger combines documentation with the ability to try out the API right from within the documentation.

In this example API, you are making requests to modify a database for a pet store. You are going to create, retrieve, update, and delete a user.

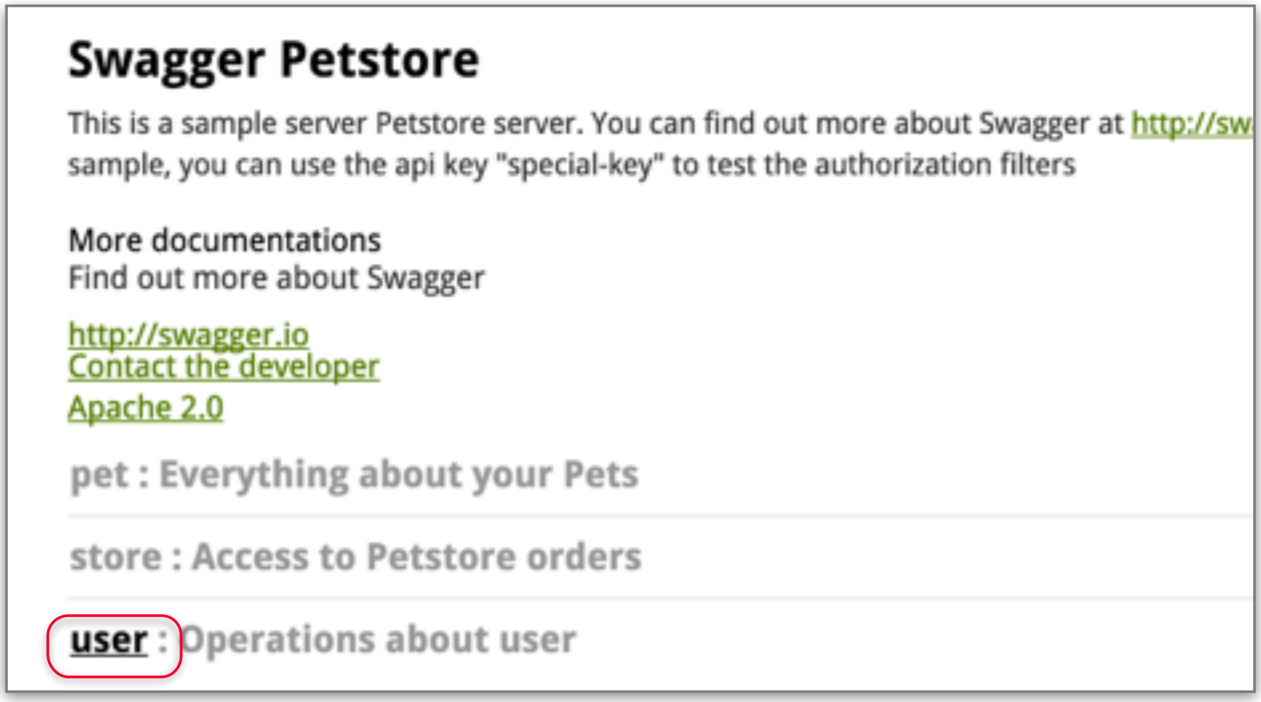
Note: The sample Swagger database can be unreliable. Sometimes you'll create a user and even though it claims to have created it, it's not in the database. Just be patient and try it a few times.

There's a link to the answers that I came up with at the end of this lecture. But don't look until you've really tried to make it work!

Create a User

1. Open a browser window and go to http://petstore.swagger.io/

2. Click on user to open up the user operations.



Swagger Petstore

This is a sample server Petstore server. You can find out more about Swagger at http://sw
sample, you can use the api key "special-key" to test the authorization filters

More documentations
Find out more about Swagger

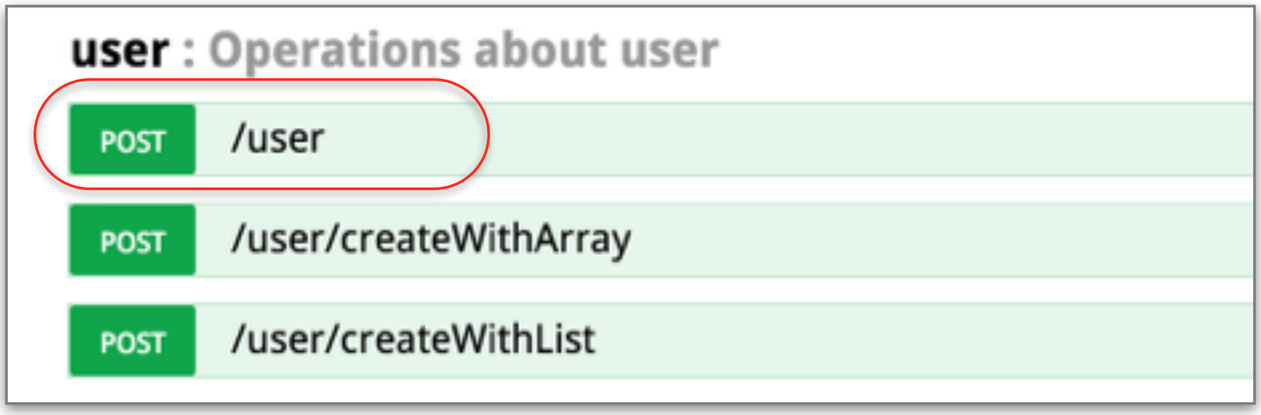
http://swagger.io
Contact the developer
Apache 2.0

pet : Everything about your Pets

store : Access to Petstore orders

user : Operations about user

3. Remember, there are four types of operations: POST (create), GET (retrieve), PUT (update), and DELETE (remove). Since we want to create a user, we want POST. Click on the first POST, which is used to create a new user. (The other POST calls are used to create multiple users at once.)

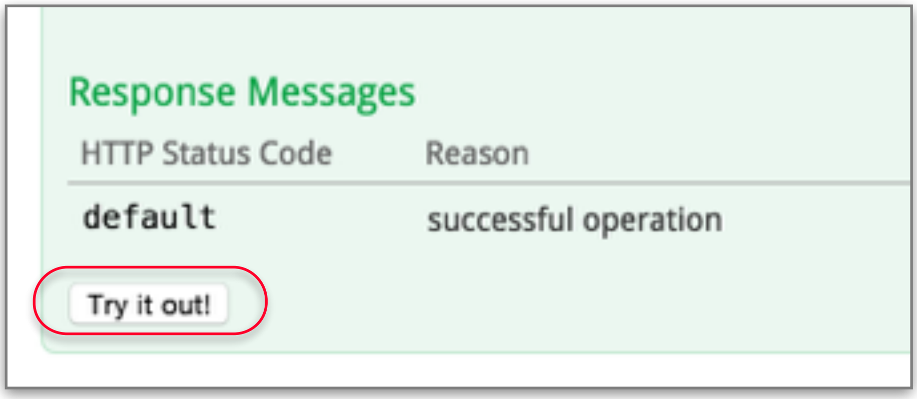

user : Operations about user
POST /user
POST /user/createWithArray
POST /user/createWithList

4. We need to pass it information on the new user we are creating. This is done by setting the body of the POST to be JSON (JavaScript Object Notation). Look under Model | Model Schema, and you will see text that looks like this:

```
{
  "id": 0,
  "firstName": "",
  "username": "",
  "lastName": "",
  "email": "",
  "password": "",
  "phone": "",
  "userStatus": 0
}
```

This describes the format of the JSON that the API is expecting. Click on that text, and you will see it appear in the body section, under Value. Everywhere you see empty "string", add in some fake data, like your user's first name, username, phone number, etc. Remember what you put for username, as you will use it later.

5. Click on **Try it out!** Scroll down to see what the pet store server sent back to you. The most important thing to look at is the Response Code. If you get a value of 200, that means that you have successfully added your user.



HTTP Status Code	Reason
default	successful operation

Retrieve Your User

1. Let's see if your user is in the database. See if you can figure out which call will return user information. Hint: it will be a GET, since GET retrieves information. Click on it.

2. Set the **Response Content Type** to application/json. This will make it return JSON.

3. There is a place to type the username. First, type in a nonsense username that people have probably not chosen already. Click **Try it out!** When you scroll down, you'll see that the response code is 404, which means that the user was not found.

4. Now try it with the username of the user you created. You should get a response code of 200, which means success. (If you are getting a 400 error, that means that the server is having problems -- it's not you. Just wait and try it again later.)

5. Look at the Request URL. It will have your username at the end.

6. Look in the Response Body section. You should see all of the data that you entered when creating the user.

Update Your User

You wouldn't actually use a web page like this to add a user to the database. This is just a way of trying it out. What you would really do was create an application (a web app, a phone app, etc.) that does it for you. The app makes the HTTP call.

For example, let's say someone is using your app and wants to change the password. Figure out which HTTP call you would use to do that. (Hint: it won't say anything directly about a password. Think about the four HTTP operations and which one would be the best one to use.)

Use this HTTP request to change the password to "pass". You'll need to put in all of the information about the user, not just the password.

Once you have successfully made this request, go back to the GET request and click **Try it out!** again. See if the new password is there.

Remove Your User

See if you can figure out how to delete your user. Like before, in the exercise textbook, paste in the method, the Request URL and the Response Code into the exercise text box.

Make the GET request again. You should see a 404 (not found) error.

Answers

If you want to see the answers I came up with, they are here:

http://sdkbridge.com/ud/Exercise1Answers.pdf