UŻYCIE SWAGGERA DO PRZYGOTOWANIA RESTFUL API W NODE.JS

INSTALACJA SWAGGERA

Mając zainstalowane NPM i Node.js doinstaluj obsługę http-server:

```
C:\Users\toor\workspace\swagger>npm install -g http-server
C:\Users\toor\AppData\Roaming\npm\http-server -> C:\Users\t
erver
C:\Users\toor\AppData\Roaming\npm\hs -> C:\Users\toor\AppDa
C:\Users\toor\AppData\Roaming\npm
`-- http-server@0.10.0
  +-- colors@1.0.3
  +-- corser@2.0.1
  +-- ecstatic@2.2.1
  | +-- he@1.1.1
  | +-- mime@1.4.1
  | +-- minimist@1.2.0
  | `-- url-join@2.0.2
  +-- http-proxy@1.16.2
  | +-- eventemitter3@1.2.0
  | `-- requires-port@1.0.0
  +-- opener@1.4.3
  +-- optimist@0.6.1
  | +-- minimist@0.0.10
```

Następnie zainstaluj pakiet Swagger nie przejmując się ostrzeżeniami:

```
C:\Users\toor\workspace\swagger>npm install -g swagger
npm WARN deprecated to-iso-string@0.0.2: to-iso-string
npm WARN deprecated jade@0.26.3: Jade has been renamed
e
npm WARN deprecated minimatch@0.3.0: Please update to m
npm WARN deprecated URIjs@1.16.1: package renamed to "u
C:\Users\toor\AppData\Roaming\npm\swagger -> C:\Users\t
C:\Users\toor\AppData\Roaming\npm\swagger-project -> C:
r-project.js
C:\Users\toor\AppData\Roaming\npm
`-- swagger@0.7.5
  +-- async@1.5.2
  +-- commander@2.11.0
  +-- connect@3.6.5
  | +-- finalhandler@1.0.6
  | | +-- on-finished@2.3.0
  | | | `-- ee-first@1.1.1
  | | +-- statuses@1.3.1
  | | `-- unpipe@1.0.0
  | +-- parseurl@1.3.2
```

Stwórz nowy projekt korzystając ze Swaggera wybierając framework **express**:

```
C:\Users\toor\workspace\swagger>swagger project create node_gallery
? Framework? (Use arrow keys)
> connect
  express
  hapi
  restify
  sails
```

```
C:\Users\toor\workspace\swagger>swagger project create node_gallery
? Framework? express
Project node_gallery created in C:\Users\toor\workspace\swagger\node
Running "npm install"...
```

Powinien powstać katalog o nazwie takiej jak tworzony projekt:

```
C:\Users\toor\workspace\swagger>dir
 Volume in drive C has no label.
 Volume Serial Number is 60BC-EEBB

 Directory of C:\Users\toor\workspace\swagger

20.10.2017  08:47    <DIR>          .
20.10.2017  08:47    <DIR>          ..
20.10.2017  08:26    <DIR>          gallery
20.10.2017  08:48    <DIR>          node_gallery
               0 File(s)              0 bytes
               4 Dir(s)  10 785 280 000 bytes free
```

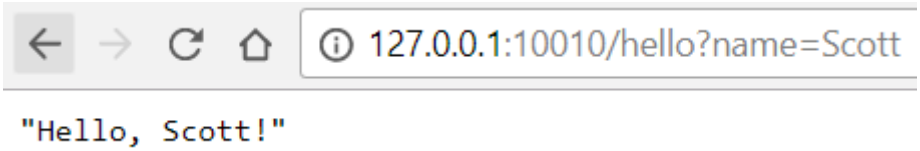
W katalogu tym znajdziemy między innymi plik *app.js* jak i folder *api*:

```
C:\Users\toor\workspace\swagger>dir node_gallery
 Volume in drive C has no label.
 Volume Serial Number is 60BC-EEBB

 Directory of C:\Users\toor\workspace\swagger\node_gallery

20.10.2017  08:48    <DIR>          .
20.10.2017  08:48    <DIR>          ..
20.10.2017  08:47               666 .gitignore
20.10.2017  08:47    <DIR>          api
20.10.2017  08:47               549 app.js
20.10.2017  08:47    <DIR>          config
20.10.2017  08:48    <DIR>          node_modules
20.10.2017  08:47               431 package.json
20.10.2017  08:47                31 README.md
20.10.2017  08:47    <DIR>          test
               4 File(s)          1 677 bytes
               6 Dir(s)  10 780 565 504 bytes free
```

Sprawdzamy czy projekt się uruchamia wywołując polecenie *swagger project start* ***nazwa_projektu***:

```
C:\Users\toor\workspace\swagger>swagger project start node_gallery
Starting: C:\Users\toor\workspace\swagger\node_gallery\app.js...
  project started here: http://localhost:10010/
  project will restart on changes.
  to restart at any time, enter `rs`
try this:
curl http://127.0.0.1:10010/hello?name=Scott
```

Możemy przejść pod wskazany adres w celu weryfikacji wyniku:

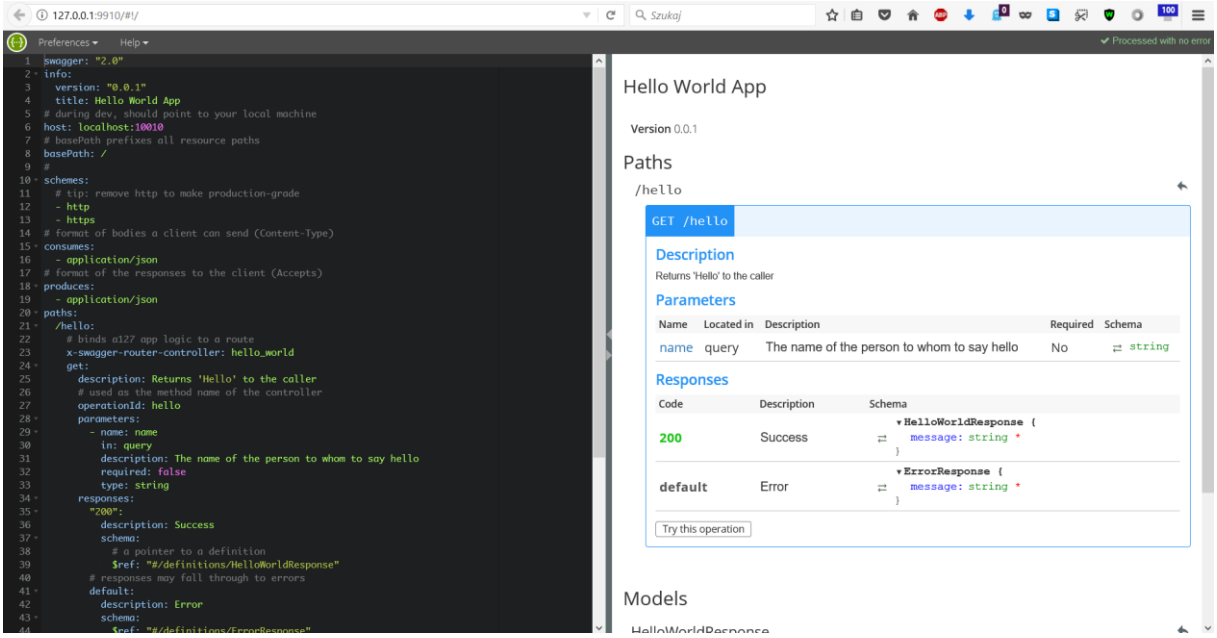


Przerywamy działanie aplikacji i przechodzimy do zaplanowania API naszego projektu.

Przechodzimy w tym celu do edytora projektu:

```
C:\Users\toor\workspace\swagger>swagger project edit node_gallery
Starting Swagger Editor.
Opening browser to: http://127.0.0.1:9910/#/edit
Do not terminate this process or close this window until finished editing.
```

Startowo będzie tam widoczna konfiguracja przykładowa wygenerowana wraz z naszym projektem.



Jeżeli projekt będzie uruchomiony to możliwe jest bezpośrednio z edytora sprawdzenie stworzonych ścieżek REST, w przeciwnym razie możliwe jest tylko ich edytowanie tak jak konfiguracji projektu.

Konfiguracja poszczególnych wywołań ma miejsce w oparciu o format YAML (ważne są wcięcia!):

```yaml
paths:
  /hello:
    # binds a127 app logic to a route
    x-swagger-router-controller: hello_world
    get:
      description: Returns 'Hello' to the caller
      # used as the method name of the controller
      operationId: hello
      parameters:
        - name: name
          in: query
          description: The name of the person to whom to say hello
          required: false
          type: string
      responses:
        "200":
          description: Success
          schema:
            # a pointer to a definition
            $ref: "#/definitions/HelloWorldResponse"
        # responses may fall through to errors
        default:
          description: Error
          schema:
            $ref: "#/definitions/ErrorResponse"
```

Kolejno w sekcji *paths:* podajemy definicję ścieżki wraz ze wskazaniem metody (*get:, post:, delete:, put:*), funkcji kontrolera na poziomie logiki businessowej (*operationId:*), przekazywanymi parametrami (*parameters:*) oraz możliwymi odpowiedziami.

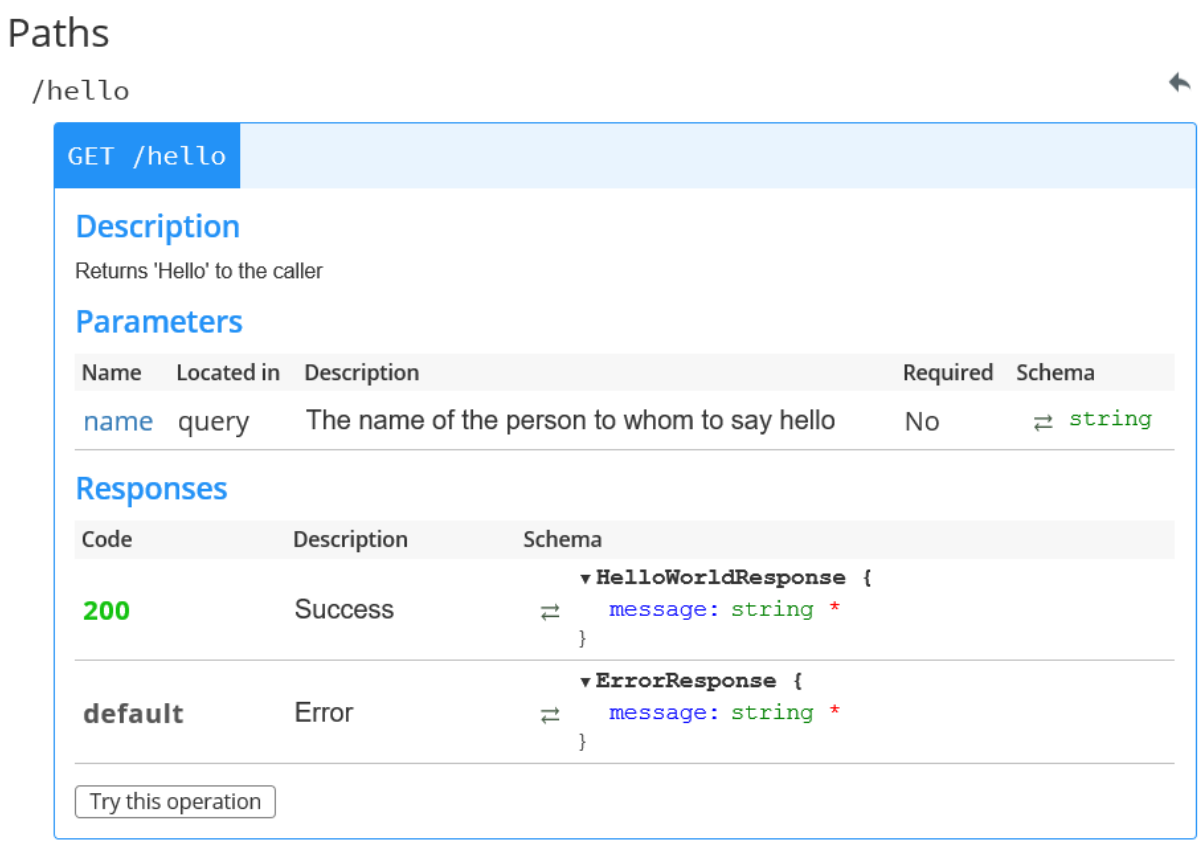
Jeżeli chcemy aby parametr był przekazywany pozycyjnie w ścieżce zamiast w zapytaniu zmieniamy pole *in:* z *query* na *path*, a w definicji ścieżki dodajemy stosowny placehoder (w powyższym przypadku byłoby to zastąpienie */hello* przez */hello/{name}*).

W parametrach jest również zawarte wskazanie *$ref* do zdefiniowanych struktur danych umieszczonych w końcowej sekcji dokumentu i wyglądających następująco:

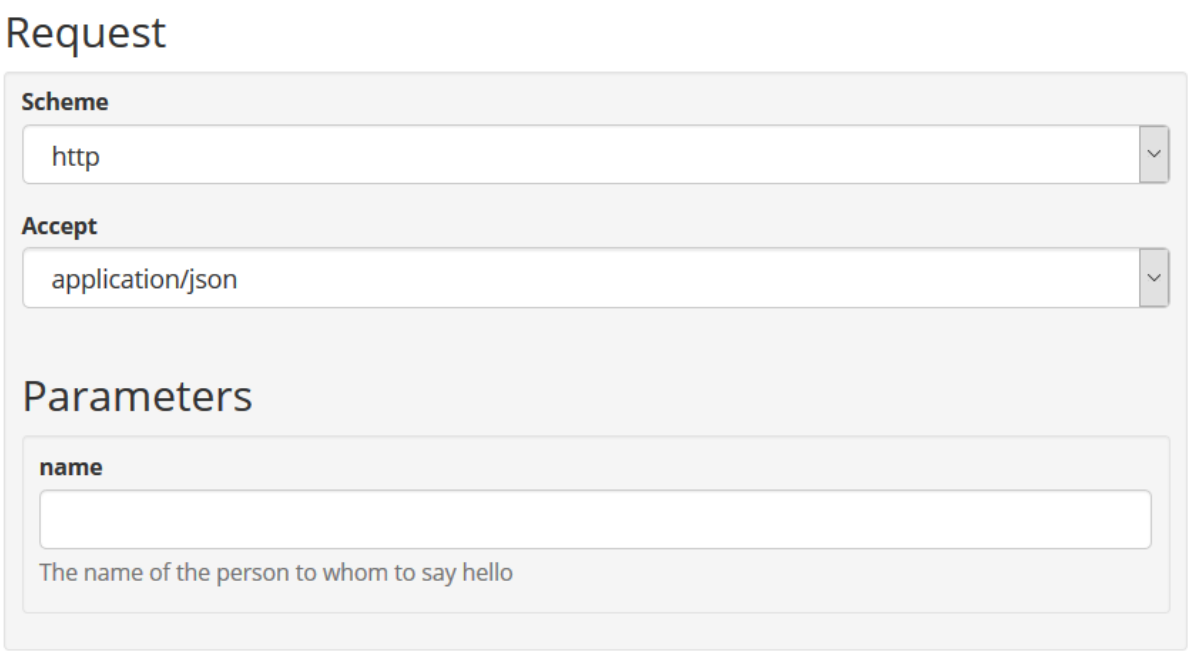
```
# complex objects have schema definitions
definitions:
  HelloWorldResponse:
    required:
      - message
    properties:
      message:
        type: string
  ErrorResponse:
    required:
      - message
    properties:
      message:
        type: string
```

Wskazujemy też przyjmowane i zwracane formaty danych (*consumes:* i *produces:*).

Cała konfiguracja jest zwizualizowana co pozwala na łatwe sprawdzenie jej poprawności:



Możemy też przetestować jej działanie wybierając ***Try this operation***:



Następnie ***Send request*:**

Jeżeli zależy nam na tym aby zmiany w naszym projekcie realizowanych za pośrednictwem edytora Swaggera miały bezpośrednie przełożenie na kod uruchamiamy projekt w trybie ***MOCK***, czyli z dodaną flagą -m:

```
C:\Users\toor\workspace\swagger>swagger project start node_gallery -m
Starting: C:\Users\toor\workspace\swagger\node_gallery\app.js...
  project started here: http://localhost:10010/
  project will restart on changes.
  to restart at any time, enter `rs`
try this:
curl http://127.0.0.1:10010/hello?name=Scott
Project restarted. Files changed:  [ 'c:\\Users\\toor\\workspace\\swagger\\node_gallery\\api\\swagger\\swagger.yaml' ]
  project started here: http://localhost:10010/
```

ZADANIE:

Korzystając ze Swaggera zaprojektuj RESTful obejmujący pełny CRUD dla galerii internetowej pozwalający na obsługę zdjęć, galerii i użytkowników. Wstępnie wypisz możliwe operacje lub przygotuj diagram przypadków użycia.