Web Services With Apogee

Server Version 1.0

You can easily make a web service using an Apogee workspace. To do this you will need the Apogee server, on which you can deploy the workspace as a web service.

NOTE: This server is currently meant for experimentation and is not production ready.

Contents

Apogee Server and Development Environment Installation

The apogee server runs on Node.js. It is recommended that you use the ApogeeNodeElectron version of the Apogee application for your development environment to create workspaces, since this version uses the same Node.js environment that the server uses.

The two of these can be downloaded as zip packages from the following web page:

- https://www.apogeejs.com/web/downloads.html

For the following installation we will assume you have npm installed on your machine. (Apologies for the sparse instructions.)

Apogee Server Installation

1. Download the zip file for the Apogee Server.

2. Unpack the zip file into the directory where you would like to install it.

3. Open a terminal window in the directory.

4. Run the following command to install the application:

```
npm install
```

5. To configure the port on which the server is listening, open the *package.json* file and edit the entry for the port.

6. Run the following command to start the server:

```
npm start
```

Apogee "Node.js" Electron Application Installation

1. Download the zip file for the Apogee "Node.js" electron application.

2. Unpack the zip file into the directory where you would like to install it.

3. Open a terminal window in the directory.

4. Run the falling command to install the application:

```
npm install
```

5. Run the following command to start the appilcation.

```
npm start
```

Notes on Apogee Electron Apps

- There are two versions of Apogee running on electron, for now at least.
    - Apogee "Web" Electron Application - This is recommended for any use outside of writing web services. This version of the app is identical to the web version as far as the workspaces it runs. From it you can link to any libraries or modules from the web. You do *not* have access to Node.js functions. There is however the ability to saved and open workspaces from the local file system.
    - Apogee "Node.js" Electron Application - This is recommended only for creating web services. This provides full access to Node.js functions, just as there is in the web server. Here you can not link to libraries and modules from the web. Instead you access NPM modules from your local machine.

Adding References

References are installed as NPM dependencies in the *package.json* file. In most cases you will want the same packages installed in the electron application and the web server.

To install an npm package into the either the server of the electron app:

1. Go the the folder which holds the *package.json* file for the application.

2. Run the following command to install the package and its dependencies:

    npm install [package name]

What about the "References" tree entry in the application?

To get access to the node modules, you do not need to include them in the references section of the application (found in the tree panel on the left), unlike in the web application. However, there is a way of adding NPM modules there. That is for adding "self installing" npm modules that will do things like add a new type of cell to the application.

Creating Workspaces for the Web Service

An apogee workspace operates as a web service as follows:

1. Request

A request is sent to the Apogee server with the following format

```
http://[host]/[workspace identifier]/[endpoint identifier]
```

In addition it may also have query parameters and a body.

2. Routing to an Endpoint

The descriptor defines the workspaces and their associated endpoints. The the proper endpoint for a request is selected with the workspace identifier and the endpoint identifier in the URL. (A single workspace may respond to multiple endpoint requests.)

3. Loading the Request into the Workspace

The selected workspace for the request is loaded into the server in its initial state. Then, based on the descriptor, data from the request is populated into the workspace.

- body - If the endpoint has the body configured, the body of the request is passed into the specified cell in the workspace.
- query parameters - If the endpoint has the query parameters configured, the query parameters are passed into a cell in the workspace.

3. Workspace Update

Once the request values are loaded into the workspace, the workspace will update. (It is as if you manually entered the request fields into the applications.)

4. Response

Upon completion of the workspace update, the response is returned. If there is output from the endpoint, this is defined in the descriptor as an output cell. In this case the value from the cell is copied into the body of the response.

If there is an error in the workspace, which you can manually trigger by throwing an error from the code, then a status 500 error response will be given.

Descriptor

As described above, the descriptor defines how requests are processed, mapping them to workspaces and endpoints.

Before we give the formal format we will show a simple example, which comes included with the web service so you can examine it directly.

Example Descriptor File

Here is an example descriptor file. First we show it with some markup on it to display what the entries mean.

Below is the actual descriptor file text.

```
{
    "workspaces": {
        "simple": {
            "source": "deploy/SimpleWorkspace.json",
            "endpoints": {
                "processnumber": {
                    "inputs": {
                        "body": "main.inputBody",
```

```
                    "queryParams": "main.inputParams"
                },
                "output": "main.outputNumber"
            }
        }
    },
    "altsimple": {
        "source": "deploy/AltSimpleWorkspace.json",
        "endpoints": {
            "processnumber": {
                "inputs": {
                    "body": "main.inputBody",
                    "queryParams": "main.inputParams"
                },
                "output": "main.outputNumber"
            },
            "processbodyonly": {
                "inputs": {
                    "body": "main.inputBody"
                },
                "output": "main.outputNumber"
            },
            "maxinput": {
                "output": "main.MAX_INPUT_VALUE"
            },
            "noout": {
                "inputs": {
                    "queryParams": "main.inputParams"
                }
            }
        }
    },
    "reflect": {
        "source": "deploy/ReflectWorkspace.json",
        "endpoints": {
            "basic": {
                "inputs": {
```

```
                    "body": "main.input"
                },
                "output": "main.output"

            }
        }
    }
}
```

This example descriptor has three workspaces in it, listed below together with the endpoints for each.

- Workspace Files:
    - **deploy/SimpleWorkspace.json** - This has endpoints that start with the path "simple"
        - Endpoints:
            - **[host]/simple/processnumber** - This endpoint passes the request body into the cell *main.inputBody* and the query params into the cell *main.inputParams*. The response is taken from the cell *main.outputNumber*.
    - **deploy/AltSimpleWorkspace.json** - This has endpoints that start with the path "delay"
        - Endpoints:
            - **[host]/altsimple/processnumber** - This endpoint passes the request body into the cell *main.inputBody* and the query params into the cell *main.inputParams*. The response is taken from the cell *main.outputNumber*. It is the same as the *SimpleWorkspace* above except it has a delay built in.
            - **[host]/altsimple/processbodyonly** - This endpoint passes the query params into the cell *main.inputParams*. The response is taken from the cell *main.outputNumber*.
            - **[host]/altsimple/maxinput** - This endpoint takes no input. It does however return the output from the cell *main.MAX_INPUT_VALUE*.
            - **[host]/altsimple/noout**- This endpoint passes the request body into the cell *main.inputBody* and the query params into the cell *main.inputParams*. There is no body in the response.
    - **deploy/ReflectWorkspace.json** - This has endpoints that start with the path "reflect"

        - Endpoints:
            - **[host]/reflect/basic** - This endpoint passes the request body into the cell *main.input*. The response, which is identical, is taken from the cell *main.output*.

Descriptor Format

The descriptor is a JSON file with the following format:

- "workspaces" - a list of workspaces to be loaded and the configuration of each. This is a JSON object with the keys being the name used in the request path (see path description below)
    - [workspace request path name] - This is the URL path entry that appears directly after the host name. It is the key for a workspace configuration, given below.
        - "source" - This is the file path on the server to load the workspace JSON file. This can be either the full workspace JSON as generated from the Apogee application (the electron application is recommended since it uses the same Node.js environment for running the application) or the model portion of the JSON. In the server, only the model portion of the workspace is loaded and used.
        - "endpoints" - This is a map of endpoints for this given workspace. The key in this map is the result path entry for this endpoint, following the request path entry for the workspace in the URL. The value for this key is the configuration for the endpoint, given below.
            - "inputs" - This is a map of the inputs. The key is the types of inputs. The value is the full path name in the Apogee workspace for the member that receives the input value. The following are the types of input. Note that all these values are optional. If no inputs are specified then the output will be read from the workspace without any input command. This may be used, for example, to read static values from the workspace.
                - "body" - This key is associated with the member that receives the body of the request. OPTIONAL
                - "queryParams" - The key is associated with the member that receives the query parameters of the request. The data is inserted as a JSON object with the key-value pairs being the request parameters. OPTIONAL
                - "trigger" - This should be used if there is no body or query parameter input. A fixed value will be written into the associated member. The value used is the valus specified under "inputTriggerValue" in the endpoint config if that value is specified. Otherwise the value "true" is used. OPTIONAL

            - "output" - This key is associated with the member full name that will be used as the response body. If this field is absent then there is no response body. OPTIONAL
            - "inputTriggerValue" - This value is used if the input option "trigger" is used. The value associated with this key is the value copied in to the trigger table when an request is done.
            - "settings" - This entry contains endpoint settings. Any settings specified here override values passed in from the workspace. Currently there are no settings implemented.
        - "settings" - This entry contains workspace specific settings. Any settings specified here override values passed in from the application. Currently there are no settings implemented.
- "settings" - This value for this key are the global settings for the server. Any values given here override default settings. Currently there are no settings implemented.

Trying It Out

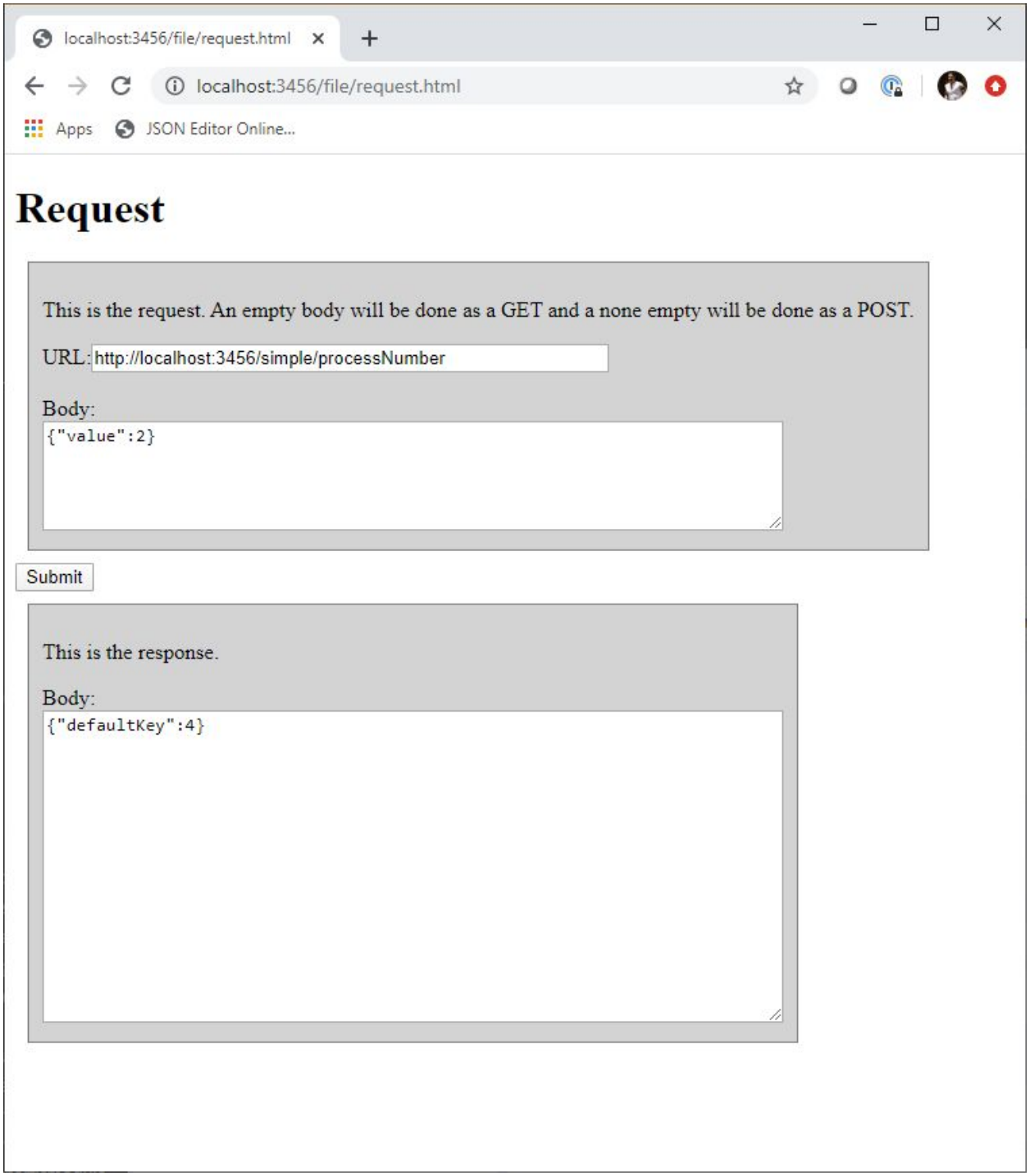
The Apogee server also includes a file server. Preloaded there is a page that lets you make web requests.

This web page is at the following URL:

```
[host]/file/request.html
```

It is displayed below, along with the input to make a request from the included workspaces.

This displays the format of request that the above endpoints take. You can also open the included workspaces in the Apogee application to see what they do.


localhost:3456/file/request.html
localhost:3456/file/request.html
Apps
JSON Editor Online...
Request
This is the request. An empty body will be done as a GET and a none empty will be done as a POST.
URL: http://localhost:3456/simple/processNumber
Body:
{"value":2}
Submit
This is the response.
Body:
{"defaultKey":4}

Deploying a Workspace - the Descriptor

To deploy a new workspace and its associated endpoints, you need to do the following:

- Copy the workspace into the *deploy* folder inside the apogee server installation directory.
- Update the descriptor to define to include the workspace and the associated endpoints.
- Restart the web server.

This is admittedly a bit cumbersome considering this is supposed to be an "easy" coding environment. This process will be improved in the future.

File Server

As mentioned above, the Apogee server includes a simple file server, to include static files.

This serves files that are in the file directory inside the installation directory.

To request a file, the url is:

    [host]/file/[path to the file]