RESTful Service API Interaction

This Java package allows the service API interaction to be accessed via a restful API on `localhost`.

Endpoints

The following endpoints are available:

- *"air-quality"*
- *"book"*
- *"current-weather"*
- *"charity-search"*
- *"charity-by-city"*
- *"dictionary"*
- *"ingredient"*
- *"joke"*
- *"nearest-transport"*
- *"news"*
- *"random-recipe"*
- *"recipe"*
- *"recipe-instructions"*
- *"stocks"*
- *"transport-search"*
- *"weather-forecast"*

As per the service APIs included except any spaces in the service names are replaced with hyphens.

Parameters are to be supplied in the URL in lowercase form.

- e.g., the dictionary service endpoint requires a `WORD` parameter (as per the outer README) and is supplied (in lowercase) as:

- Services that require multiple parameters are supplied by an ampersand (&) separated list.

Note that there are no `POST` requests available (only `GET`), setting the request method to `POST` in the request header will result in an error.

Responses

All responses are of the form:

Where

- **service** - the name of the service; this is either the endpoint, for a recognised service or `error` if a client HTTP error occurs.

- **message** - a natural language string representing the service response - useful for voice assistants.

- **metadata** - extra data that may be useful, but unsuitable for natural language e.g., image/web links.

- **code** - the HTTP status code using this standard.

Deployment

The backend is containerised to make for simple deployment.

Docker

To start the application in a Docker container, you must first build the image with:

```
docker build --tag services .
```

And then to start the application on port `8080`:

```
docker run --rm -it -p 8080:8080 services services
```

Without Docker

If you would prefer to not use Docker, you will need JDK 11 and Maven installed.

To start the application, ensuring no service is currently using port `8080`, run the following command in a terminal in
the "services" directory (where the `pom.xml` is located):

```
mvn clean; mvn org.springframework.boot:spring-boot-maven-plugin:run
```

An executable `jar` file can be used instead, built and executed manually using:

```
mvn clean package
```

You can also start the application using a Python script with the following command:

```
python3 tools/run.py
```

For simplicity, a shell script is included to run the service interaction as a RESTful API:

```
./services
```

Depending on your system configuration, you may need to give the script execution permissions which can only be done with administrator privileges (`sudo`):

```
 chmod +x services
```

And then perform your HTTP requests. For example, after the application is running, open a terminal and enter:

```
curl "localhost:8080/current-weather?city=london"
```

This will return something similar to:

Note that the URL must be quoted to escape the question marks and ampersands if performing HTTP requests from the
command line.

Command Line Arguments

There are different execution options for the API, summarised below.

Note that these can only be enabled by passing them as command line arguments to the Python `run` script or running the
application as a `jar`. Command line arguments are unavailable with Java if not executing using a `jar` file (using
the `java -jar ...` command).

The following arguments are all optional.

- A desired port can be specified using `-port=PORT` (or one of `--port`, `-p` and `--p`). Note that this must be an
  integer and is set to `8080` by default.

- Logging information to the console can be enabled with the `-l` (or `--l`) flag - logging is disabled by default.

- The application can be built into a single jar and executing using the `-jar` (or `--jar`) flag.
  - This option is only available using the Python `run` script.

When using the `jar` file (with `java -jar ...`):

- The `-l` flag is supplied **without** the leading `-` character(s) - i.e. as `l`.
- The `-port` flag is **always** supplied with a single `-` and the same goes for `-p`.

Running `python3 tools/run.py -h` gives you a similar breakdown of possible command line arguments and their aliases.

- The `-h` flag is not available outside of the Python script.

Testing

To execute unit tests(served at: http://localhost:8000/surefire-report.html):

```
docker run -p 80:8000 -ti --rm services /services/tools/test
```

To execute PMD analysis(served at: http://localhost:8000/pmd.html):

```
docker run -p 80:8000 -ti --rm services /services/tools/analysis
```

To execute test coverage (served at: http://localhost:8000/jacoco):

```
docker run -p 80:8000 -ti --rm services /services/tools/coverage
```