# Restful API

REST is the way HTTP should be used.
Today we only use a tiny bit of the HTTP protocol's methods – namely `GET` and `POST`. The REST way to do it is to use all of the protocol's methods.

| HTTP Method | Action | Examples |
|---|---|---|
| GET | Obtain information about a resource | http://example.com/api/orders (retrieve order list) |
| GET | Obtain information about a resource | http://example.com/api/orders/123 (retrieve order #123) |
| POST | Create a new resource | http://example.com/api/orders (create a new order, from data provided with the request) |
| PUT | Update a resource | http://example.com/api/orders/123 (update order #123, from data provided with the request) |
| DELETE | Delete a resource | http://example.com/api/orders/123 (delete order #123) |

Example of Rest API design (Todo app)

| HTTP Method | URI | Action |
|---|---|---|
| GET | http://[hostname]/todo/api/v1.0/tasks | Retrieve list of tasks |
| GET | http://[hostname]/todo/api/v1.0/tasks/[task_id] | Retrieve a task |
| POST | http://[hostname]/todo/api/v1.0/tasks | Create a new task |
| PUT | http://[hostname]/todo/api/v1.0/tasks/[task_id] | Update an existing task |
| DELETE | http://[hostname]/todo/api/v1.0/tasks/[task_id] | Delete a task |

## A. Setting up

1. Install flask
   1. sudo pip install Flask

2. create app.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == '__main__':
    app.run(debug=True)
```

3. Run "python app.py" —> Open browser with url "localhost:5000"

you will see "Hello World!"

**********

## B. Simple example

```
from flask import Flask, jsonify, request

app = Flask(__name__)

tasks = [
    {
        'id': 1,
        'title': u'Milk , Cheese, Pizza, Fruit, Tylenol',
        'done': False
    },
    {
        'id': 2,
        'title': u'Learn Python',
        'description': u'Need to find a good Python tutorial on the web',
        'done': False

    }
]

@app.route("/")
def hello():
        return "Hello World”

# http://localhost:5000/todo/api/v1.0/tasks
@app.route('/todo/api/v1.0/tasks', methods=['GET'])
def get_tasks():
    return jsonify({'tasks': tasks})

# http://localhost:5000/todo/api/v1.0/tasks/2
@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['GET'])
def get_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    return jsonify({'tasks': task[0]})

#curl -i -H "Content-Type: application/json" -X POST -d '{"title":"Read a
book"}' http://localhost:5000/todo/api/v1.0/tasks
@app.route('/todo/api/v1.0/tasks', methods=['POST'])
def create_task():
    if not request.json or not 'title' in request.json:
        abort(400)
    task = {
        'id': tasks[-1]['id'] + 1,
        'title': request.json['title'],
```

```
        'description': request.json.get('description',''),
        'done': False
    }
    tasks.append(task)
    return jsonify({'task': task}), 201


#curl -i -H "Content-Type: application/json" -X PUT -d '{"done":true}'
http://localhost:5000/todo/api/v1.0/tasks/2
@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['PUT'])
def update_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    if not request.json:
        abort(400)
    if 'title' in request.json and type(request.json['title']) != unicode:
        abort(400)
    if 'description' in request.json and type(request.json['description']) is
        not unicode:
        abort(400)
    if 'done' in request.json and type(request.json['done']) is not bool:
        abort(400)
    task[0]['title'] = request.json.get('title', task[0]['title'])
    task[0]['description'] = request.json.get('description', task[0]
    ['description'])
    task[0]['done'] = request.json.get('done', task[0]['done'])
    return jsonify({'task': task[0]})


@app.route('/todo/api/v1.0/tasks/<int:task_id>', methods=['DELETE'])
def delete_task(task_id):
    task = [task for task in tasks if task['id'] == task_id]
    if len(task) == 0:
        abort(404)
    tasks.remove(task[0])
    return jsonify({'result': True})


if __name__ == '__main__':
    app.run(debug=True)
```

# Authorization setup

```
pip install flask-httpauth
```

```
//////////////////////////////////
from flask import make_response
from flask_httpauth import HTTPBasicAuth
auth = HTTPBasicAuth()

//////////////////////////////////

@auth.get_password
def get_password(username):
    if username == 'ekarat':
        return 'python'
    return None

@auth.error_handler
def unauthorized():
   #abort(401)
   return make_response(jsonify({'error': 'Unauthorized access'}),
   401)
//////////////////////////////////

@app.route('/todo/api/v1.0/tasks', methods=['GET'])
@auth.login_required
def get_tasks():
    return jsonify({'tasks': tasks})
```

//Fail

```
$ curl -i http://localhost:5000/todo/api/v1.0/tasks
HTTP/1.0 401 UNAUTHORIZED
```

```
//Success
curl -u ekarat:python -i http://localhost:5000/todo/api/v1.0/tasks
HTTP/1.0 200 OK
```

Reference


1. @ ใน python คืออะไร
    1. https://www.kanouivirach.com/2015/08/%E0%B8%97%E0%B8%B3%E0%B8%84%E0%B8%A7%E0%B8%B2%E0%B8%A1%E0%B9%80%E0%B8%82%E0%B9%89%E0%B8%B2%E0%B9%83%E0%B8%88%E0%B8%81%E0%B8%B1%E0%B8%9A-python-decorators/
    2. https://medium.com/@skblackcat/decorator-%E0%B9%83%E0%B8%99-python-

%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3-%E0%B9%81%E0%B8%A5%E0%B8%B0%E0%B8%A1%E0%B8%B1%E0%B8%99%E0%B9%83%E0%B8%8A%E0%B9%89%E0%B8%97%E0%B8%B3%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3%E0%B9%84%E0%B8%94%E0%B9%89%E0%B8%9A%E0%B9%89%E0%B8%B2%E0%B8%87-ee74e35e84fb

Note
Push app.py to heroku at
https://pure-reef-85577.herokuapp.com/todo/api/v1.0/tasks

How I did that, here
https://github.com/datademofun/heroku-basic-flask

```
Commands after changed
git add --all
git commit -m 'changes'
git push heroku master
```