Building a Restful Blog API with Flask

Blog API

POST	/posts/
PUT	/posts/<id>
PATCH	/posts/<id>
DELETE	/posts/<id>
GET	/posts/<id>
GET	/posts/

POST	/comments/
PUT	/comments/<id>
PATCH	/comments/<id>
DELETE	/comments/<id>
GET	/comments/<id>
GET	/comments/

GET	/comments/?post_id=<id>
GET	/posts/<id>/comments/

Request mime type:	application/json
Response mime type:	application/json

The Flask application

blog/app.py

```
from flask import Flask
from flask.ext.sqlalchemy import Sqlalchemy

app = Flask(__name__)

app.config[“SQLALCHEMY_DATABASE_URI”]=“postgresql://host/db”
db = Sqlalchemy(app)

post = Blueprint(‘post’,__name__)
app.register_blueprint(post)

comment = Blueprint(‘comment’,__name__)
app.register_blueprint(comment)

if __name__ == "__main__":
    app.run()
```

Data Models (resources)

blog/models.py

```
from flask import url_for
from app import db

class Post(db.Model):
    __tablename__=“posts”
    id= db.Column(db.Integer, primary_key=True, autoincrement=True)
    title = db.Column(db.String(256))
    body = db.Column(db.Text())


class Comment(db.Model):
    __tablename__=“comments”
    id= db.Column(db.Integer, primary_key=True, autoincrement=True)
    post_id= db.Column(db.Integer, nullable=False)
    body = db.Column(db.Text())
```

POST /posts/

blog/post_api.py

```
from flask import jsonify,request
from app import app
from helper import created
from model import Post, Comment


@post.route(‘/posts/’,methods=['POST'])
def post():
    return created(Post.post(request.json))
```

blog/helper.py

```
def created(resource)
    response = jsonify(resource.to_dict())
    response.status_code = 201
    response.headers['location']=resource.get_url()
    return response
```

POST /posts/ (continued)

blog/model.py

```
from flask import abort,url_for

class Post(db.Model):
    @staticmethod
    def post(json):
            post = Post()
        post.from_json(json)
        db.session.add(self)
        db.session.commit()
        return post

    def from_json(self, json):
        try:
            self.Title = json['title']
            self.Body = json['body']
        except:
            abort(400)

    def get_url(self);
            return url_for('post.get',id=self.id,_external=True)

    def to_dict(self):
            return {title=self.Title,body=self.Body}
```

PUT /posts/<id>

blog/post_api.py

```
@post.route('/posts/<int:id>',methods=['PUT'])
def put():
    return jsonify(Post.put(id,request.json))
```

blog/model.py

```
class Post(db.Model):
    @staticmethod
    def put(id, json):
        post = Post.query.get_or_404(id)
        post.from_json(json)
        db.session.commit()
        return post

    def from_json(self, json):
        try:
            self.Title = json['title']
            self.Body = json['body']
        except:
            abort(400)
```

PATCH /posts/<id>

blog/post_api.py

```
@post.route(‘/posts/<int:id>',methods=[''PATCH'])
def patch():
    return jsonify(Post.patch(id,request.json))
```

blog/model.py

```
class Post(db.Model):
    @staticmethod
    def patch(id, json):
        post = Post.query.get_or_404(id)
        post.from_json_patch(json)
        db.session.commit()
        return post

    def from_json_patch(self,json):
        if json.get(’title’) is not None:
            self.Title = json[’title’]
        if json.get(’body’) is not None:
            self.Body = json[’body’]
```

DELETE /posts/<id>

blog/post_api.py

```
@post.route('/posts/<int:id>',methods=['DELETE'])
def delete():
    return jsonify(Post.delete(id))
```

blog/model.py

```
class Post(db.Model):
    @staticmethod
    def delete(id):
        post = Post.query.get_or_404(id)
        db.session.delete(post)
        db.session.commit()
        return {}
```

GET /posts/<id>

blog/post_api.py

```
@post.route('/posts/<int:id>',methods=['GET'])
def get(id):
    return jsonify(Post.get(id))
```

blog/model.py

```
class Post(db.Model):
    @staticmethod
    def get(id):
        return Post.query.get_or_404(id).to_dict()

    def to_dict(self):
        return {title=self.Title,body=self.Body}
```

GET /posts/<id>
(continued)

blog/post_api.py

```
#alternative implementation
@post.route('/posts/<int:id>',methods=['GET'])
@json
def get(id):
    return Post.query.get_or_404(id)
```

blog/helper.py

```
def json(f):
    @functools.wraps(f)
    def wrapped(*args,**kwargs):
        rv = f(*args,**kwargs)
        if not isinstance(rv,dict):
            rv = rv.to_dict()
        return jsonify(rv)
    return wrapped
```

GET /posts/

blog/post_api.py

```
@post.route(‘/posts/‘,methods=['GET'])
def index():
    return jsonify(Post.index())
```

blog/model.py

```
class Post(db.Model):
   @staticmethod
    def index():
        return {'urls’:[post.get_url() for post in Post.query.all()]}

    def get_url(self);
        return url_for(‘post.get’,id=self.id,_external=True)
```

GET /comments/

GET /comments/?post_id=<id>

blog/comment_api.py

```
@comments.route(‘/comments/‘,methods=['GET'])
def index():
    return jsonify( Comment.index( request.args.get(‘post_id’)) )
```

blog/model.py

```
class Comment(db.Model):
  @staticmethod
  def index(id=None):
      if id is None:
          return {'urls’:[comment.get_url() for comment in Comment.query.all()]}
      try:
          post_id = int(id)
      except:
          abort(400)
      Post.query.get_or_404(post_id)
      return {'urls’:[comment.get_url() for comment in Comment.query.filter_by(post_id=post_id).all()]}
```

GET /posts/<id>/comments/

blog/comment_api.py

```
@comments.route('/posts/<int:id>/comments/',methods=['GET'])
def index_for_post(id):
    return jsonify( Comment.index(id) )
```

blog/model.py

```
class Comment(db.Model):
    @staticmethod
    def index(id):
        Post.query.get_or_404(id)
        return {'urls':[comment.get_url() for comment in Comment.query.filter_by(post_id=id).all()]}
```

HTTP Error Responses

blog/app.py

```
@app.errorhandler(404):
def not_found(e):
    response = jsonify({'error':'not found','message':e.args[0]})
    response.status_code=404
    return response

@app.errorhandler(500):
def internal_server_error(e):
    response = jsonify({‘error':'internal server error’,'message':e.args[0]})
    response.status_code=500
    return response

@app.errorhandler(400):
def bad_request(e):
    response = jsonify({'error':'bad request’,'message':e.args[0]})
    response.status_code=400
    return response

@app.errorhandler(403):
def forbidden(e):
    response = jsonify({'error':'forbidden’,'message':e.args[0]})
    response.status_code=403
    return response
```

Next Presentation

Restful Flask API

Security
Rate limiting
Paging
Validation
Caching
Testing