Wutu Documentation

Release 0.1.0

Šarnas Navickas <zaibacu@gmail.com>

January 08, 2016

Contents

Contents:

CHAPTER 1

Installation

1.1 Requirements

- Python 3.4+ version
- nodeJS and npm (If planning on running UI test suite)

1.2 Installing

```
pip install wutu
```

CHAPTER 2

Usage

2.1 Quick Start

First, lets create our project with embedded module

```
from wutu import Wutu
app = Wutu(index="index.html")

@app.create_module
def greetings_module():
    hellos = ["Hello", "Hola", "Labas"]
    return {
        "get": lambda req: [{"text": hello} for hello in hellos],
        "get_entity_name": lambda req: "greeting"
    }

app.run(host="localhost", port=5555)
```

This app creates AngularJS controller (GreetingsModuleController) and service (GreetingsModuleService).

By defining key *get* in result dictionary, we say, that *HTTP GET* should go to this method (similar behavior goes with *POST*, *PUT* and *DELETE* methods). *get_entity_name* is optional, by defining it we say what is module entity (in this case - *greeting*), so module can automatically create methods like:

- get_greeting
- create_greeting
- update_greeting
- delete_greeting

and variable *greeting_list* which is our greeting model. It can be named in the way you like, just be aware of silly method names

Now, create our html page. It may wary, but vital parts should be covered.

```
<!DOCTYPE html>
<html lang="en" ng-app="wutu">
<head>
    <meta charset="UTF-8">
    <title>test</title>
    <script type="text/javascript" src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.4/angular
    <script type="text/javascript" src="/wutu.js"></script>
</head>
```

```
<body>
    <div ng-controller="GreetingsModuleController">
        <span ng-repeat="greeting in greeting_list">
            {{ greeting.text }}
        </span>
    </div>
</body>
</html>
```

Vital parts are:

- including *angularJS* script
- including */wutu.js* script

And required by *AngularJS*:

- *ng-app* notation on html tag
- bind required controller

2.2 Adding AngularJS Plugins

First of all, you must include plugin's JavaScript into your main html file. Same goes with *AngularJS*, same goes here.

Eg.

```
<script type="text/javascript" src="//code.angularjs.org/X.Y.Z/angular.min.js"></script>
<script type="text/javascript" src="//code.angularjs.org/X.Y.Z/angular-cookies.js"></script>
```

This would include your plugin.

Second step is to tell app constructor, that you will use certain plugin. In *ngCookies* case it would go like this

```
from wutu import Wutu
app = Wutu(index="index.html", ngmodules=["ngCookies"])
```

Very similar to *AngularJS* module constructor

2.3 Static Files

There is no static file hosting by design. Same position goes with other popular frameworks: *Flask*, *Django* etc. The reason is very simple - there are tools in the market which are already perfect for that, and there is no reason to drop whole project performance because of those files. It is still possible to do hackish solution and make them work , but such solution is only viable in development environment.

Our recommendation would be Nginx static file hosting: https://www.nginx.com/resources/admin-guide/serving-static-content/

CHAPTER 3

Compiler

Compiler handles *AngularJS* services and controllers creation. It is used by default, when modules are registered.

CHAPTER 4

JavaScript Utils

AngularJS is an awesome framework, but in the end it still lacks some utilities. Most of them are that way by design - to force users write clean code, and make creators' life easier. However, *Wutu* aims at fast and easy start, so these utilities can make you start faster. In the end, user is fully allowed to write proper *AngularJS* code...

4.1 Promise unwrap

AngularJS once had this feature (it was removed after 1.2 version), it is very handy, but produces a lot of pain for *AngularJS* developers and some confusion for the users. This unwrapping is done in other way, thus removing part of problems, but is not as comfortable to use.

```
<unwrap promise="some_function()">
        <span>{{ data.content }}</span>
</unwrap>
```

If we assume, that *some_function* does a http call and returns dictionary with parameter called *content*, this would certainly work. Keep in mind, that this will make page execute those promise statements on page load. To avoid that, it is possible to use *ng-if* mechanism (if element is not visible - at the start - it is not executed).