Tempates

template() function or **view()** decorator can be used.
Bottle will look for templates in the *./views/* folder or any folder specified in the **bottle.TEMPLATE_PATH** list.
The syntax of template (http://www.bottlepy.org/docs/dev/stpl.html)

In [3]:
```
from bottle import template,route
```

In [4]:
```
#In the way of function template()
@route('/hello')
@route('/hello/<name>')
def hello(name='World'):
    return template('hello_template', name=name)
    #It will load "hello_template.tpl" with the name variable set.
```

In [5]:
```
from bottle import view
```

In [6]:
```
#In the way of decorator view()
@route('/hello')
@route('/hello/<name>')
@view('hello_template')
def hello(name='World'):
    return dict(name=name)
    #Here, we transmit the name variable set by returing a dict.
```

If we modify the template files, **bottle.TEMPLATES.clear()** should be called to clear the caching.

In [ ]:
```

```

Basic API Usage

bottle.SimpleTemplate() can return a object.
And **render()** function of the object can be used to transmit values.

In [7]:
```
from bottle import SimpleTemplate
```

In [8]:
```
tpl = SimpleTemplate('Hello {{name}}!')
```

In [9]:
```
tpl.render(name='world')
```

Out[9]:
```
u'Hello world!'
```

In [ ]:
```

```

Pass a dictionary into the template using keyword arguments

In [11]:
```
from bottle import template
```

In [12]:
```
my_dict={'number': '123', 'street': 'Fake St.', 'city': 'Fakeville'}
```

In [13]:
```
template('I live at {{number}} {{street}}, {{city}}', **my_dict)
```

Out[13]:
```
u'I live at 123 Fake St., Fakeville'
```

In [ ]:

Python expression can be used

In [14]:
```
template('Hello {{name.title() if name else "stranger"}}!', name=None)
```

Out[14]:
```
u'Hello stranger!'
```

In [15]:
```
template('Hello {{name.title() if name else "stranger"}}!', name='mArC')
```

Out[15]:
```
u'Hello Marc!'
```

In [16]:
```
#"title()" is a function of strings
```

In [ ]:

HTML special characters are escaped

to prevent XSS attacks
But we can disable escaping by using exclamation mark(!).

In [17]:
```
#Escape
template('Hello {{name}}!', name='<b>World</b>')
```

Out[17]:
```
u'Hello &lt;b&gt;World&lt;/b&gt;!'
```

In [18]:
```
#Disable Escaping
template('Hello {{!name}}!', name='<b>World</b>')
```

Out[18]:
```
u'Hello <b>World</b>!'
```

In [ ]:

Syntax for template files

- Variables
  They are used in **{{...}}**, such as *{{name}}*.
- Python code is allowed, but with two additional synatax rules:

  1. Indentation is ignored.
  2. Code blocks should be ended by "**% end**"
     For example:

....

```
<ul>
% for item in basket:
  <li>{{item}}</li>
% end
</ul>
```

....

- Code lines
  They are used after **% ...**, such as *% name = 'Bob'*
- Code Blocks
  They are used in **<% ... %>**, such as
  ....
  <%

  ```
  # A block of python code
  name = name.title().strip()
  ```

  %>
  ....
- Use **%** or **<%**, **%>** but not python codes
  For example:
  \% This text-line starts with the '%' token.
  \<% Another line that starts with a token but is rendered as text.
  {{'\%'}} this line starts with an escaped token.
- Whitespace Control
  1、

  ```
  <div>
   % if True:
    <span>content</span>
   % end
  </div>
  ```

  will generate into -->

  ```
  <div>
    <span>content</span>
  </div>
  ```

  ....
  2、

  ```
  <div>\\
   %if True:
     <span>content</span>\\
   %end
  </div>
  ```

  will generate into -->

  ```
  <div><span>content</span></div>
  ```

- Some functions are provided
    1. `include(sub_template, **variables)`
       Just like the **include()** in C code.
       For example:
       % include('header.tpl', title='Page Title') Page Content % include('foother.tpl')
       ....
    2. `rebase(name, **variables)`
       Mark the current template to be later included into a different template.
       For example:
       In **a.tpl**:

       ```
       % rebase('base.tpl', title='Page Title')
       <p>Page Content ...</p>
       ```

       In **base.tpl**:

       ```
       <html>
       <head>
         <title>{{title or 'No title'}}</title>
       </head>
       <body>
         {{!base}}
       </body>
       </html>
       ```

       When we use **a.tpl**,
       `<p>Page Content ...</p>` will be included into **base.tpl** and replace `{{!base}}`
    3. defined(name)
       Check whether **name** variable is defined in the current template.
    4. get(name, default=None)
       Get the **name** variable just like the **get** function of dictionary.
    5. setdefault(name, default)
       Get the **name** variable just like the **setdefault** function of dictionary.

```
In  [ ]:
```