{{ AngularJS + Django }}

Writing angularjs templates inside django templates.

{{ The curly bracket situation }}

How to insert AngularJS template syntax in Django templates?

The **templatetag** template tag

Outputs one of the syntax characters used to compose template tags.

```
{% templatetag openvariable %} a_var {% templatetag closevariable %}
```

renders to

```
{{ a_var }}
```

The **bypass** template tag (ng.py)

Render the raw content of the

Warning: This template tag has to rebuild tokens already parsed. So the output won't be exactly the input.

(e.g. `{{a_var}}` becomes `{{ a_var }}`)

```
{% load ng %}
…
{% bypass %}{{ a_var }}{% endbypass %}
…
```

renders to

```
{{ a_var }}
```

The **insert** template tag (ng.py)

Load a file from a relative path below settings.EXTERNAL_TEMPLATE_PATH without parsing its contents.

```
EXTERNAL_TEMPLATE_PATH = os.path.join(BASE_DIR, ‘ng_templates’)
```

```
{% load ng %}
…
<div>{% insert ‘path/to/file’ %}</div>
…
```

The **ng-bind** directive

Replace the text node of a html element.

```
<span ng-bind="a_var" ></span>
```

Is the same as

```
<span>{{ a_var }}</span>
```

Multiple controller example on prediki.com

The **ng-init** directives

Set initial values of the angularjs scope by rendering django variabls into ng-init directives.

```
<div ng-controller="PredictionCtrl">
  <h4><i>{% trans "current prediction" %}</i></h4>
  <div class=“prediction" ng-init="prediction='{{ prediction|escapejs }}'"
       ng-bind="prediction">{{ prediction }}</div>
</div>
```

renders to

```
<div ng-controller="PredictionCtrl">
  <h4><i>current prediction</i></h4>
  <div class=“prediction" ng-init=“prediction=‘A Value’"
       ng-bind="prediction">A Value</div>
</div>
```

kanu[at]zweckfrei.org

ng.py

```
import os
from django.template import Node, Library
from django.template.base import TOKEN_BLOCK, TOKEN_VAR, TOKEN_COMMENT
from django.conf import settings
register = Library()


class InsertFileeNode(Node):

    def __init__(self, filepath):
        self.filepath = filepath

    def render(self, context):
        base = settings.EXTERNAL_TEMPLATE_PATH
        fullpath = os.path.normpath(
            os.path.join(base, self.filepath.resolve(context)),
        )
        if not fullpath.startswith(base):
            if settings.DEBUG:
                return "[not a valid template]"
            else:
                return ""
        with open(fullpath, 'r') as fp:
                output = fp.read()
        return output


@register.tag
def insert(parser, token):
    """
    Load a file from the filesystem and insert it without
    parsing its contents.
    The filepath must be a relative path from settings.EXTERNAL_TEMPLATE_PATH.

    {% insert 'path/to/file' %}
    """
    bits = token.split_contents()
    if len(bits) != 2:
        raise TemplateSyntaxError("only 1 argument")

    filepath = parser.compile_filter(bits[1])
    return InsertFileeNode(filepath)


class BypassNode(Node):

    def __init__(self, raw):
        self.raw = raw

    def render(self, context):
        return self.raw


@register.tag
def bypass(parser, token):
    """
    Bypass the parsing of a code block.

    {% bypass %}
    <div ng-controller="ACtrl">{{ a_variable }}</div>
    {% endbypass %}

    Note that there is no real way bypass the parsing. The only possible way
    is to rebuild the already parsed tokens.
    A {{a_var}} will be rebuild as {{ a_var }}
    """
    raw = u""
    while parser.tokens:
        token = parser.next_token()
        if token.token_type == TOKEN_BLOCK and token.contents == 'endbypass':
            break
        elif token.token_type == TOKEN_BLOCK:
            raw += u"{%% %s %%}" % token.contents
        elif token.token_type == TOKEN_VAR:
            raw += u"{{ %s }}" % token.contents
        else:
            raw += unicode(token.contents)

    return BypassNode(raw)
```