Processing Overridable Templates (for Studio 38, Project Jetson)

# History

|  |  |
| --- | --- |
| Created by | Rupert |
| Created | 27-01-2009 |
| Last modified by |  |
| Last modified |  |

# Introduction

This document describes a simple mechanism, how templates should be used in reusable apps. Imagine a simple forum application, which can be used either as a standalone application on your website, let’s call it the “general forum” or “hooked” under a user’s personal area, let’s call this case “the forum of Aidas”. The general forum should be available via

[www.example.com/forum/](http://www.example.com/forum/)

and the forum of Aidas should have the URL

[www.example.com/person/aidas/forum/](http://www.example.com/person/aidas/forum/).

Our forum app lives under an application folder forum, the templates used are placed like below. Mind, that the templates given under forum are forum\_list.html and thread\_list.html (amongst others) in contrast to the templates under people/forum, where only thread\_list.html exists.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MyDjangoProject | | | | |
| |--- | ... | | | |
| |--- | apps | | | |
|  | |--- | ... | | |
|  | |--- | people | | |
|  | |--- | forum | | |
|  | |--- | ... | | |
|  | templates | | | |
|  | |--- | ... | | |
|  | |--- | forum | | |
|  |  | |--- | forum\_list.html | |
|  |  | |--- | thread\_list.html | |
|  |  | |--- | ... | |
|  | |--- | people | | |
|  |  | |--- | ... | |
|  |  | |--- | forum | |
|  |  |  | |--- | thread\_list.html |

Our purpose is to minimize template duplication. As most of those templates will display the same contents just extending different base templates, the design of the templates should use the following advices to give most flexibility when overriding specific templates is needed and to avoid any template duplication.

# Usage in views

Suppose, you have an entry like this in your urls.py:

|  |  |  |
| --- | --- | --- |
| url(r*'example/(?P<object\_url\_part>.\*)forum/',* ‘show\_forum’, | | |
|  | { | |
|  |  | *'include'* : [None, URL\_ID\_PERSON], |
|  | } | |
| ), | | |

For further details about defining URL entries this way, refer to [1].

The called view function show\_forum first should get the “related object” from the provided object\_url\_part using the get\_object\_from\_url function. Additionally, the get\_object\_from\_url function returns a base\_template variable, which holds a predefined path to a base template, which the rendered template shall extend. The base base\_template variable is then passed to the rendered template. Further details about base\_template are discussed in the next chapter.

Next, you should call the select\_template\_for\_object function. This function gets three paramameters: the name of the template to use, in our example case below forum\_list.html, the “related object” (in our example case, this will be either None for the general forum or <Person Aidas>, who lives under the people application and a relative directory, where the template is placed. This is usually the name of the application, where our view is defined, in our example case forum. The select\_template\_for\_object function looks in the template directory forum under the “object’s template directory” for the given template file. If there is such a template, the found template will be returned (In our example case, this would be people/forum/forum\_list.html). Otherwise the “default template” with the same name will be searched under the provided relative directory, in our case forum/forum\_list.html.

|  |  |  |
| --- | --- | --- |
| from general.base.utils.loader import select\_template\_for\_object | | |
| from general.base.views import get\_object\_from\_url | | |
| ... | | |
| def show\_forum(request, object\_url\_part, \*\*kwargs): | | |
|  | ... | |
|  | (obj, base\_template) = get\_object\_from\_url(object\_url\_part, \*\*kwargs) | |
|  | obj = extra\_context[*'object'*] # we get the object from extra\_context | |
|  | t = select\_template\_for\_object(*"forum\_list.html"*, obj, *"forum"*) | |
|  | context = { | |
|  |  | *'base\_template'* : base\_template |
|  |  | ... |
|  |  | } |
|  | return t.render(context) | |

# Template Design

A typical template would look like this:

|  |  |
| --- | --- |
| {% extends base\_template %} | |
| {% load i18n base %} | |
|  | |
| {% comment %} | |
|  | context vars used: |
|  | object The object related to the forum (or None) |
|  | base\_template A "base template" used for template inheritance. |
| {% endcomment %} | |
| ... | |
| <H1>Welcome to My Brandnew Forum</H1> | |
| ... | |
| <p>{% include\_select "thread\_list.html" for object under "forum" %}</p> | |

Mind the first line of the listed template code. The extended base template is given by the passed in context variable base\_template. This makes the template more flexible, if just the extended base template should be different for usage with different “related objects”.

The last line in the given template uses a template tag include\_select. This template tag does the same as the select\_template\_for\_object function at the template level.

# Template Tags

To be able to use “dynamic” template inclusion in templates, there is a new template tag include\_select available under {% load base %}.

Sample usage:

|  |
| --- |
| {% include\_select "template\_1.html" "template\_2.html" "template\_3.html" %} |

or

|  |
| --- |
| {% include\_select "template\_1.html" for obj under "newforum"%} |

Usage (as railroad diagram): With <<template\_var>> as a context variable name containing some string and <<string>> as any string expression:

<<string>>

<<template\_var>>

expr

the include\_select has either the form

expr

or

expr

<<obj>>pr

expr

# References

|  |  |
| --- | --- |
|  | „ A generic URL-Object-Mapping mechanism “, available via <https://code.jetsonproject.com/svn/jetson/trunk/jetson/resources/modeling/base/Objects_from_url.docx> |