Hammy Goonan

PYTHON

Flask-WTF tricks

Hammy Goonan
Feb 23, 2016 • 2 min read

I use Flask-WTF's What the Form module for most of my Flask projects as it takes care of things like csrf tokens and so on which is a nice piece of mind.

I have recently come across a couple of really powerful features that have been hiding from me in plain sight.

I was trying to create new, edit and duplicate forms and was painfully aware how similar it all was. It had a pretty big old 'code smell', there was a lot of repetition and a lot of places data could fall through the cracks. I wanted to be able to use one form and one route to be able to handle it all.

Populating from data

Where previously I'd been using convoluted templates to populate fields depending on if they'd been set or not, you can just pass a data object to the form and it will prepopulate:

```
user = User.query.get(1)
form = UserForm(obj=user)
```

Easy! Now your `UserForm` will be populated with the required fields.

Even better, if you can initialise an empty model it makes life even easier (in this example I'm using SQLAlchemy).

```
@app.route('/new/', methods=['GET', 'POST'])
@app.route('/edit/<id>', methods=['GET', 'POST'])
def advert(id=None):
    """Create new adverts."""
    if code:
        item = Item.query.get(id)
    else:
        item = Item()
    form = ItemForm(obj=item)
    ...
```

Saving/Updating data from form

It works the other way as well. You can update an object with the data returned using the `populate_obj()` method of the `Form` object:

```
item = Item.query.get(id)
form = ItemForm(obj=item)
if form.validate_on_submit():
    form.populate_obj(item)
    db.session.add(item)
    db.session.commit()
```

If you need to manipulate some of the data before populating the object you can edit the `form.<field>.data` attributes which, unlike the `requests.form` attributes are not an `ImmutableDictionary`:

```
item = Item.query.get(id)
form = ItemForm(obj=item)
if form.validate_on_submit():
    tag = request.form.get('tag')
    form.tag.data = Tag.query.filter_by(name=tag).first()
    form.populate_obj(item)
    db.session.add(item)
    db.session.commit()
```

Automatic form generation

One I haven't had a chance to use it yet, but it is also possible to auto-generate your forms with appropriate validation fields (from here):

```
from flaskext.wtf import Form
from wtforms.ext.appengine.db import model_form
from models import MyModel

MyForm = model_form(MyModel, Form, field_args={
    'name' : {
        'validators' : [validators.Length(max=10)]
    }
})
```

The above code generates a form using the data in `MyModel`. If there are any database restraints on any of the fields they will be used to generate validators. The form that is created will extends from `Form` (it would be possible to extend from a non-built in form) and then additional validation rules are added using the `field_args` keyword argument.

Iterate over form fields

This one is pretty straight forward but can save you a lot of time. You can iterate over a form object in a template:

```
{% for field in form %}
    <tr>
        <th>{{ field.label }}</th>
        <td>{{ field }}</td>
    </tr>
{% endfor %}
```

I strongly recommend reading over the WTForms documentation. There's a lot of really powerful tools in there.

Sign up for more like this.

Enter your email

Subscribe

Connecting to host database from a Docker container

Generally speaking, in a production environment, you don't want things like Nginx and your database running in Docker containers. You want those services running on the host and your application(s) in a container. The tricky part here is that localhost in a container is the...

Hammy Goonan
Apr 29, 2021 • 1 min read

Hammy Goonan © 2022
Powered by Ghost