UNDERSTANDING DJANGO'S MODEL FORMSETS IN DETAIL AND THEIR ADVANCED USAGE.

Similar to the regular formsets, Django also provides model formset that makes it easy to work with Django models. Django model formsets provide a way to edit or create multiple model instances within a single form. Model Formsets are created by a factory method. The default factory method is modelformset_factory(). It wraps formset factory to model forms. We can also create inlineformset_factory() to edit related objects. inlineformset_factory wraps modelformset_factory to restrict the queryset and set the initial data to the instance's related objects.

Step1: Create model in models.py

```
class User(models.Model):
    first_name = models.CharField(max_length=150)
    last_name = models.CharField(max_length=150)
    user_group = models.ForeignKey(Group)
    birth_date = models.DateField(blank=True, null=True)
```

Step2: in forms.py

```
from django.forms.models import modelformset_factory
from myapp.models import User

UserFormSet = modelformset_factory(User, exclude=())
```

This will create formset which is capable of working with data associated with User model. We can also pass the queryset data to model formset so that it can do the changes to the given queryset only.

```
formset = UserFormSet(queryset=User.objects.filter(first_name__startswith='M'))
```

We can produce an extra form in the template by passing 'extra' argument to the modelformset_factory method, we can use this as follows.

```
UserFormSet = modelformset_factory(User, exclude=(), extra=1)
```

We can customize the form that will be displayed in the template by passing the new customized form to modelformset_factory. For eg: in our current example if want birth_date as date picker widget then we can achieve this with the following change in our forms.py.

```
class UserForm(forms.ModelForm):

    birth_date = forms.DateField(widget=DateTimePicker(options={"format": "YYYY-MM-DD", "pickSeconds": False}))
    class Meta:
        model = User
        exclude = ()

UserFormSet = modelformset_factory(User, form=UserForm)
```

In general Django's model formsets do validation when at least one from data is filled, in most of the cases there we'll be needing a scenario where we require at least one object data to be added or another scenario where we'd be required to pass some initial data to form, we can achieve this kind of cases by overriding basemodelformset as following,

in forms.py

```
class UserForm(forms.ModelForm):

    birth_date = forms.DateField(widget=DateTimePicker(options={"format": "YYYY-MM-DD", "pickSeconds": False}))
    class Meta:
        model = User
        exclude = ()
        exclude = ()

    def __init__(self, *args, **kwargs):
        self.businessprofile_id = kwargs.pop('businessprofile_id')
        super(UserForm, self).__init__(*args, **kwargs)

        self.fields['user_group'].queryset = Group.objects.filter(business_profile_id = self.businessprofile_id)

BaseUserFormSet = modelformset_factory(User, form=UserForm, extra=1, can_delete=True)

class UserFormSet(BaseUserFormSet):

    def __init__(self, *args, **kwargs):
        #  create a user attribute and take it out from kwargs
        # so it doesn't messes up with the other formset kwargs
        self.businessprofile_id = kwargs.pop('businessprofile_id')
        super(UserFormSet, self).__init__(*args, **kwargs)
        for form in self.forms:
            form.empty_permitted = False

    def _construct_form(self, *args, **kwargs):
        # inject user in each form on the formset
        kwargs['businessprofile_id'] = self.businessprofile_id
        return super(UserFormSet, self)._construct_form(*args, **kwargs)
```

Step3: in views.py

```
from myapp.forms import UserFormSet
from django.shortcuts import render_to_response

def manage_users(request):

    if request.method == 'POST':
        formset = UserFormSet(businessprofile_id=businessprofileid, data=request.POST)
        if formset.is_valid():
            formset.save()
            # do something
    else:
        formset = UserFormSet(businessprofile_id=businessprofileid)
    return render_to_response("manage_users.html", {"formset": formset})
```

Step4: in template

The simplest way to render your formset is as follows.

```
      <form method="post" action="">
            {{ formset }}
     </form>
```

Previous post

Next post

By Ramya Ambati Posted On 05 May 2012

SENIOR DEVELOPER at MICROPYRAMID