Usage of Category Attribute Value Form

Since many steps will be repeated when adding/modifying an instance with category, including the table should display different category attributes when modifying category, storage of category attributes values, etc.

To make all of these easier to use, category provides relatetd functions currently, so that these duplicate steps can be completed directly.

Related component

1. category.views.CategoryViews: CategoryAttributeValuesFormMixin, CategoryAttributeValuesCreateView, CategoryAttributeValuesUpdateView
2. category.views.forms.CategoryForms: BaseCategoryAttributeValueForm
3. category.templatetags.categoryTags: renderCategoryAttributeValuesForm

Usage

1. Let the views that you use to add/modify instances inherit CategoryAttributeValuesCreateView and CategoryAttributeValuesUpdateView respectively.
2. Continuing, override get_form_kwargs() in these two views, please refer to the explanation for the reason, and the content is as follows:

```
    def get_form_kwargs(self):
        kwargs = super().get_form_kwargs()
        kwargs['attributeValueModel'] = <Your Attribute Value Model>
        <... All kwargs you want to add in ...>
        return kwargs
```

1. Continuing from the above two points, set attribute in the view: `categoryType = <Your category type>`. This category type can be obtained from constant in CategoryService, and which categoryType to use depends on what objects the view is used to add/modify. For example, product.views.ProductView.AddNew used to add Product will set `categoryType = CategoryService.PRODUCT_CATEGORY`.
2. Add an instance form inherits BaseCategoryAttributeValueForm, and set form_class of adding/modifying views to this form.
3. In template, those who call the API do not need to write category and fields of categoryAttributes by themselves, just add `{% load categoryTags %}` to the file header directly and add `{% renderCategoryAttributeValuesForm form categoryType %}` to where needs to have category and fields of categoryAttributes. **(Notice: The form parameter to access is the form whose view uses this template tag, and categoryType can be directly typed in categoryType. The CategoryAttributeValuesMixin has already processed this context_data.)**
4. Lastly, when you want to save the instance, you only need to call `form.save()` directly, which will save the instance itself and attributeValues.

Explanation

1. Both of CategoryAttributeValuesCreateView and CategoryAttributeValuesUpdateView inherit CategoryAttributeValuesFormMixin, and `get_form_kwargs()` in CategoryAttributeValuesFormMixin defines some kwargs that need to be passed into form and will affect operation of form.
2. Continuing, `CategoryService.saveAttributeValues()` is used in the steps of `form.save()`, and instanceAttributeValueModel is necessary to operate properly. But it annot be defined in `CategoryAttributeValuesFormMixin.get_form_kwargs()` in advance, users have to override `get_form_kwargs()` in CategoryAttributeValuesCreate/UpdateView and add `{'attributeValueModel': <attributeValueModel>}` to operate properly on their owns. If they don't do that, an exception will be raised in `BaseCategoryAttributeValueForm.__init__()` to ask for overriding.

3. If you still have some questions about using, you can review the code in 'Related component' or ask Jacky.
4. If you feel that there are some problems when using, you can refer to product.views.ProductView.AddNew.