GlobalSignalProcessor

1. Description

GlobalSignalProcessor is designed as an app to build a centrial signal listener based on TCMS models. Then for signals, a centrial handler will route signal to relative handler. Every handler will create a task and implement the downstream functionality like Email Notification and ChangeLog.

2. Features

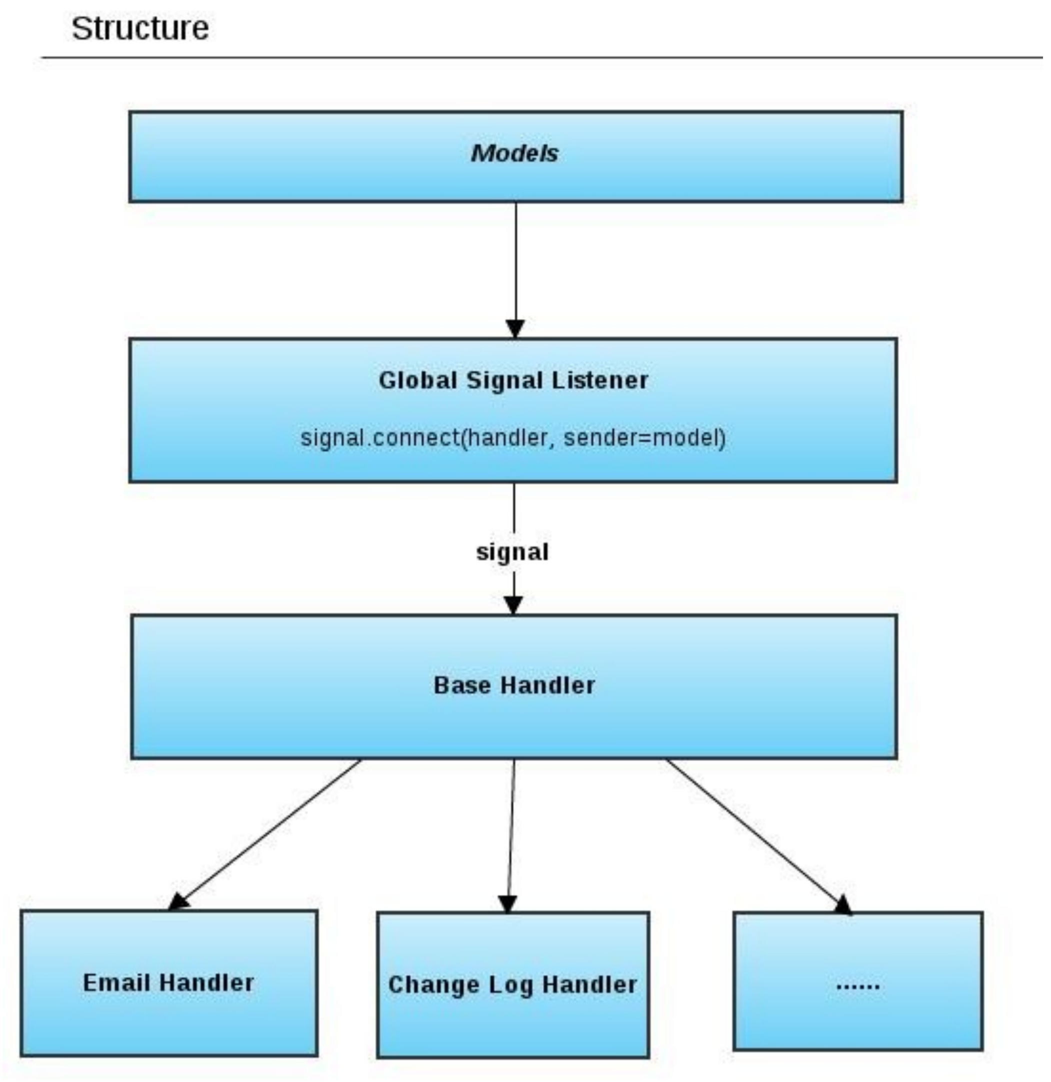
GlobalSignalProcessor can provide centrial signal listener and centrial signal handler for a project. It can be introduced as a generic signal solution for any django projects. Its features list as below:

1. Provide a configurable way for developers to decide which models and what operations need listened.
2. Global listener can listen all the data operation which is through ORM.(create, delete, update, bulk_update)
3. Provide detailed operation info for handler.(model, instances, operation type)
4. Build a base handler to restrict the definition of all subclasses, and provide the common actions.
5. Introduce async mode when handling massive signals to make signals not blocking.

3. Implementation

3.1 Design Structure

The design of this app is simple. GlobalSignalProcessor connect sender(model) to receiver(handler) one time to build a globalsignal listener. Considering there may be many functions work based on signals. BaseHandler has abstracted all the possible common actions and pre-process origin signals to provide a neat interface for subclasses.



3.2 Code Workflow

This app defined six classes.

1. SignalConfig

   SignalConfig defines the signals and operation types which will be used for custom configuration.

2. SignalHandlerType

   SignalHandlerType are defined to connect models with handlers based on custom configuration of downstream functions(EmailHandler, ChangeLogHandler and so on).

3. SignalHandlerTask

   SignalHandlerTask inherit from threading.Thread. It's used for async handling.

4. BaseHandler

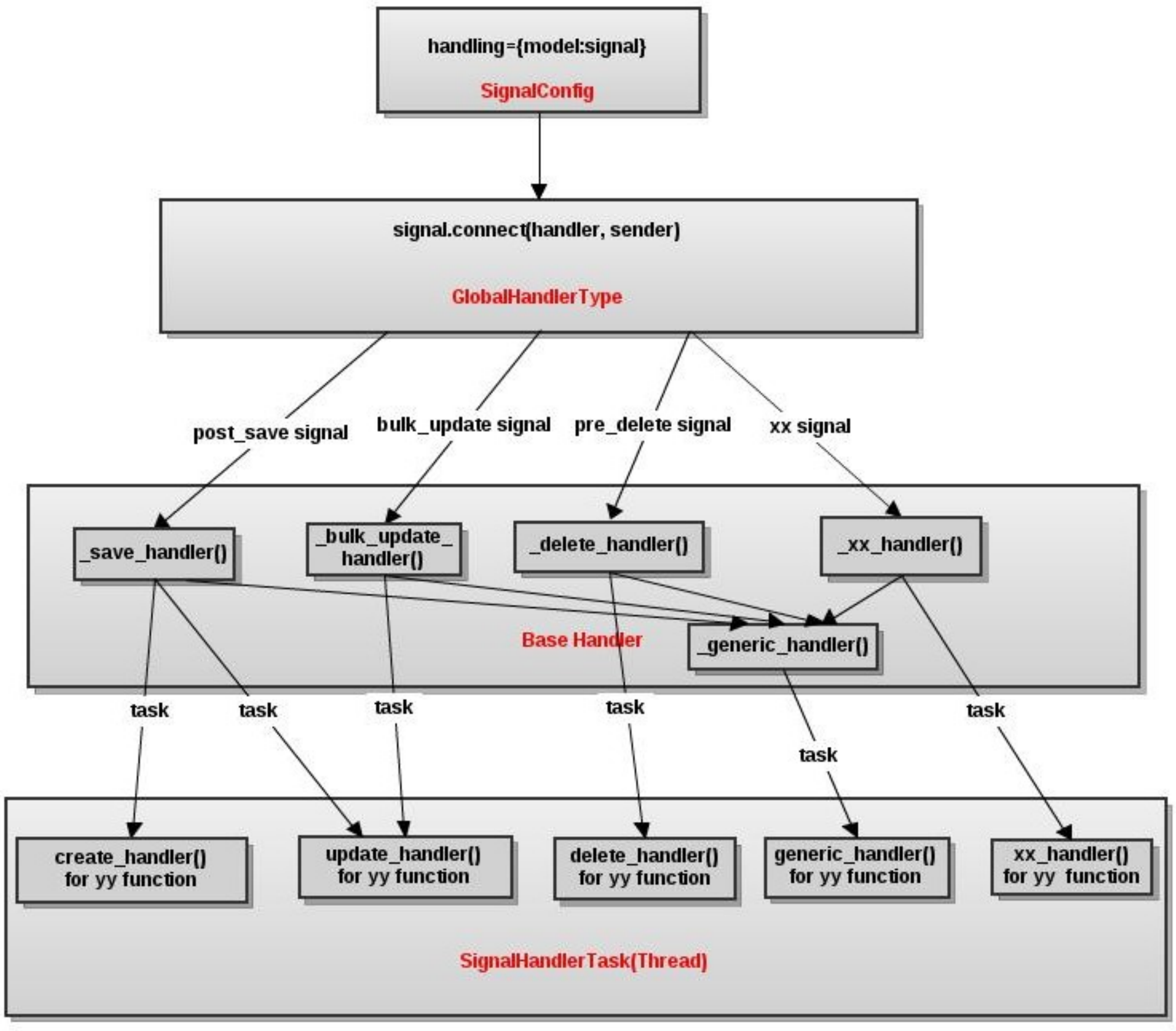
   BaseSignalHandler defined a base handler of all function handlers. It will classify signals and call the specific implementation of subclasses.

5. EmailHandler

   EmailHandler and ChangeLogHandler are subclasses of BaseHandler. They defines handlers and implement the specific features.

6. ChangeLogHandler

   Same as EmailHandler The code workflow show as following image:



4. Notes

4.1 Custom managers which expect global signal listener should

inherit from GlobalSignalManager.

GlobalSignalManager return the custom QuerySet(GlobalSignalQuerySet) to listen the update operation. To ensure that update operation can emit signals, GlobalSignalQuerySet overrides update method based on QuerySet. Custom managers which are defined as default manager of models should inherit from GlobalSignalManager. GlobalSignalManager are implemented as blew:

```
 class GlobalSignalQuerySet(models.query.QuerySet):
    """
    Use for listening the bulk update operation.
    """
    def update(self, **kwargs):
        instances = super(self.__class__, self).update(**kwargs)
        SIG_BULK_UPDATE.send(sender=self.model, instance=self, **kwargs)
        return instances
class GlobalSignalManager(models.Manager):
    """
    Custom manager which expect global signal listener should inherit from this class.
    """
    def get_query_set(self):
        return GlobalSignalQuerySet(self.model, using=self._db)
```

4.2 Models which expect global signal listener should inherit from class TCMSActionModel.

TCMSActionModel set GlobalSignalManager as its default manager. It will return GlobalSignalQuerySet to listen the bulk update operation. Source code show as below:

```
 class TCMSActionModel(models.Model, UrlMixin):
"""
TCMS action models.
Use for global log system.
Models which expect global signal listener should inherit from this class.
"""
objects = GlobalSignalManager()

class Meta:
    abstract = True
```

4.3 Please do not use method of 'bulk_create' if expecting global

signal listener.

Since bulk_create does not return created object ids. globalSignalHandler does not support this case for now. For detailed info, please refer to django source code.