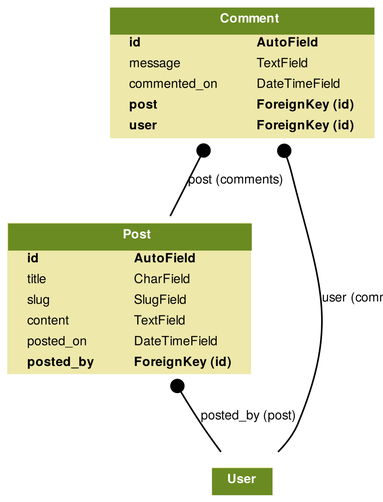
Simple "Model" driven RESTful API

Using Django and Django Rest Framework to create a Autodocumented API with Swagger

Django-Model: Example Mini-Blog



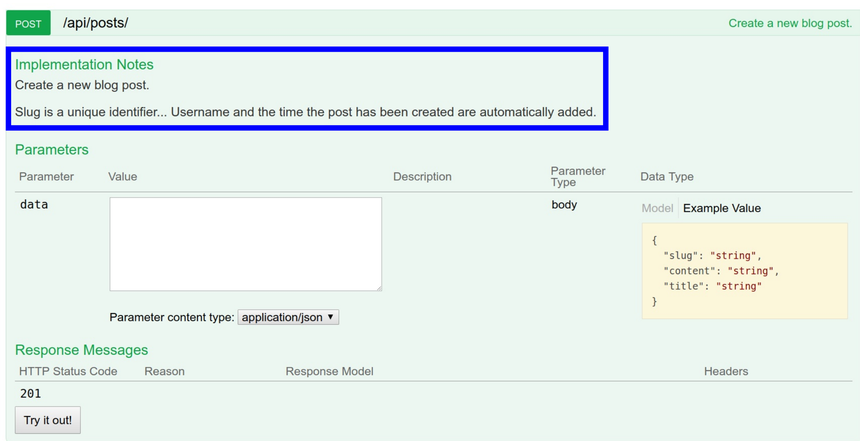

```python
class Comment(models.Model):
    message = models.TextField(verbose_name=_("message"))
    commented_on = models.DateTimeField(verbose_name=_("post date"))
    post = models.ForeignKey(Post, related_name='comments', verbose_name=_("post"))
    user = models.ForeignKey(settings.AUTH_USER_MODEL, verbose_name=_("user"), related_name='comment_user')


class Post(models.Model):
    title = models.CharField(max_length=200, verbose_name=_("title"))
    slug = models.SlugField()
    content = models.TextField(verbose_name=_("content"))
    posted_on = models.DateTimeField(verbose_name=_("post date"))
    posted_by = models.ForeignKey(settings.AUTH_USER_MODEL, verbose_name=_("posted by"))
```

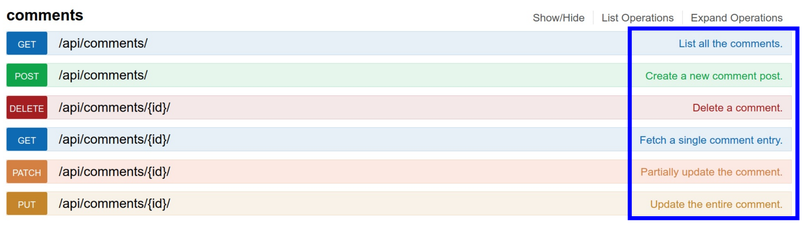
Serializer: Layer between Model and View

```python
class UserSerializer(serializers.HyperlinkedModelSerializer):
    posts = serializers.HyperlinkedRelatedField(queryset=Post.objects.all(), view_name='post-detail', many=True)

    class Meta:
        model = User
        fields = ('username', 'posts')


class PostSerializer(serializers.HyperlinkedModelSerializer):
    comments = serializers.HyperlinkedRelatedField(view_name='comment-detail', many=True, read_only=True)
    posted_by = serializers.ReadOnlyField()
    posted_on = serializers.ReadOnlyField()

    class Meta:
        model = Post
        fields = ('title', 'slug', 'content', 'posted_on', 'posted_by', 'comments')


class CommentSerializer(serializers.ModelSerializer):
    user = serializers.ReadOnlyField()
    commented_on = serializers.ReadOnlyField()

    class Meta:
        model = Comment
        fields = ('message', 'commented_on', 'post', 'user')
```

View: Python docstrings are transformed to "Swagger"

```python
class UserViewSet(viewsets.ReadOnlyModelViewSet):
    """
    This endpoint presents the users in the system.

    list:
    List all the users. As you can see, the users posts are hyperlinked.

    retrieve:
    Retrieve a single user. As you can see, the the user's posts are hyperlinked.
    """
    queryset = User.objects.all()
    serializer_class = UserSerializer


class PostViewSet(viewsets.ModelViewSet):
    """
    This endpoint presents the posts in the system.

    list: List all the posts.

    retrieve: Fetch a single post entry.

    create: Create a new blog post.

    Slug is a unique identifier...
    Username and the time the post has been created are automatically
    added.

    update: Update the entire post.

    Details for updating...

    partial_update: Partially update the post.

    destroy: Delete a blog post.
    """
    queryset = Post.objects.all()
    serializer_class = PostSerializer

    def perform_create(self, serializer):
        now = timezone.now()
        serializer.save(posted_by=self.request.user, posted_on=now)


class CommentViewSet(viewsets.ModelViewSet):
    """
    This endpoint presents the comments in the system.

    list: List all the comments.

    retrieve: Fetch a single comment entry.

    create: Create a new comment post.

    update: Update the entire comment.

    partial_update: Partially update the comment.

    destroy: Delete a comment.
    """
    queryset = Comment.objects.all()
    serializer_class = CommentSerializer

    def perform_create(self, serializer):
        now = timezone.now()
        serializer.save(user=self.request.user, commented_on=now)
```





Code available on Github: https://github.com/geometalab/model-driven-drf-swagger

Test-Drive this example: https://mdd.hixi.ch/docs - login: jenny/userpassword | tom/userpassword