Django Messaging App: Custom ORM Manager and View Caching

Grok 3

June 16, 2025

1 Introduction

This document updates the implementation for two Django tasks in the `alx-backend-python` repository, directory `Django-signals_orm-0x04`, to address grader feedback. The tasks are: (1) a custom ORM manager for unread messages in `messaging/managers.py`, used in `messaging/views.py` with `.only()`, and (2) view caching with `cache_page(60)`. Each file includes line-by-line explanations and external references.

2 Task 4: Custom ORM Manager for Unread Messages

The objective is to add a `read` boolean field to the `Message` model, define a custom manager in `messaging/managers.py`, use `Message.unread.unread_for_user` in `messaging/views.py`, and optimize with `.only()`.

2.1 Code: managers.py

Listing 1: messaging/managers.py

```
1 from django.db import models  # Line 1: Import Django's model module for
      manager definition
2
3 class UnreadMessagesManager(models.Manager):  # Line 2: Define custom manager
4     """Custom manager for filtering unread messages."""  # Line 3: Docstring
          for manager purpose
5     def unread_for_user(self, user):  # Line 4: Method to filter unread
          messages
6         """Retrieve unread messages for a user, optimized with .only()."""  #
              Line 5: Docstring for method
7         return self.get_queryset().filter(  # Line 6: Get base QuerySet and
              filter
8             recipient=user, read=False  # Line 7: Filter by recipient and
                  unread status
9         ).only('id', 'sender__username', 'content', 'timestamp').select_related
              ('sender')  # Line 8: Optimize with .only() and .select_related()
```

2.2 Code: models.py

Listing 2: messaging/models.py

```
1 from django.db import models  # Line 1: Import Django's model module
2 from django.conf import settings  # Line 2: Import settings for AUTH_USER_MODEL
3 from .managers import UnreadMessagesManager  # Line 3: Import custom manager
4
```

```
class Message(models.Model):  # Line 4: Define Message model
    sender = models.ForeignKey(  # Line 5: Define sender foreign key
        settings.AUTH_USER_MODEL,  # Line 6: Reference user model
        on_delete=models.CASCADE,  # Line 7: Delete messages if sender is deleted
        related_name='sent_messages'  # Line 8: Enable reverse queries
    )
    recipient = models.ForeignKey(  # Line 9: Define recipient foreign key
        settings.AUTH_USER_MODEL,  # Line 10: Reference user model
        on_delete=models.CASCADE,  # Line 11: Delete messages if recipient is deleted
        related_name='received_messages'  # Line 12: Enable reverse queries
    )
    content = models.TextField()  # Line 13: Field for message content
    timestamp = models.DateTimeField(auto_now_add=True)  # Line 14: Auto-set creation time
    read = models.BooleanField(default=False)  # Line 15: Track read status, default unread

    objects = models.Manager()  # Line 16: Default manager
    unread = UnreadMessagesManager()  # Line 17: Custom manager named 'unread'

    class Meta:  # Line 18: Model metadata
        indexes = [  # Line 19: Indexes for performance
            models.Index(fields=['recipient', 'read']),  # Line 20: Index for filtering
            models.Index(fields=['timestamp']),  # Line 21: Index for sorting
        ]
        ordering = ['-timestamp']  # Line 22: Sort by timestamp, descending
```

2.3 Code: views.py (Task 4)

Listing 3: messaging/views.py (Task 4)

```
from django.shortcuts import render  # Line 1: Import render for templates
from django.contrib.auth.decorators import login_required  # Line 2: Import authentication decorator
from .models import Message  # Line 3: Import Message model

@login_required  # Line 4: Restrict to authenticated users
def inbox_view(request):  # Line 5: Define inbox view
    """Display unread messages using UnreadMessagesManager."""  # Line 6: Docstring for view
    unread_messages = Message.unread.unread_for_user(request.user)  # Line 7: Use custom manager
    return render(request, 'messaging/inbox.html', {  # Line 8: Render template
        'unread_messages': unread_messages  # Line 9: Pass messages to template
    })
```

2.4 Code: inbox.html

Listing 4: messaging/templates/messaging/inbox.html

```
<!DOCTYPE html>  <!-- Line 1: Declare HTML5 -->
<html>  <!-- Line 2: Root HTML element -->
<head>  <!-- Line 3: Metadata section -->
    <title>Inbox</title>  <!-- Line 4: Page title -->
</head>
<body>  <!-- Line 5: Visible content -->
    <h1>Your Unread Messages</h1>  <!-- Line 6: Page header -->
    {% for message in unread_messages %}  <!-- Line 7: Loop over messages -->
        <div>  <!-- Line 8: Group message content -->
```

```
            <p><strong>From:</strong> {{ message.sender.username }}</p>  <!-- Line 9: Sender username -->
            <p><strong>Message:</strong> {{ message.content }}</p>  <!-- Line 10: Message text -->
            <p><strong>Sent:</strong> {{ message.timestamp }}</p>  <!-- Line 11: Timestamp -->
        </div>
    {% empty %}  <!-- Line 12: Fallback for empty list -->
        <p>No unread messages.</p>  <!-- Line 13: Empty message -->
    {% endfor %}  <!-- Line 14: End loop -->
</body>
</html>
```

3 Task 5: Implement Basic View Caching

The objective is to configure `LocMemCache` and apply `cache_page(60)` to a conversation view in `messaging/views.py`.

3.1 Code: settings.py

Listing 5: messaging/messaging/settings.py

```
CACHES = {  # Line 1: Cache configuration
    'default': {  # Line 2: Default cache backend
        'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',  # Line 3: In-memory cache
        'LOCATION': 'unique-snowflake',  # Line 4: Cache identifier
    }
}
```

3.2 Code: views.py (Task 5)

Listing 6: messaging/views.py (Task 5)

```
from django.shortcuts import render  # Line 1: Import render for templates
from django.contrib.auth.decorators import login_required  # Line 2: Import authentication decorator
from django.views.decorators.cache import cache_page  # Line 3: Import caching decorator
from .models import Message  # Line 4: Import Message model
from django.db.models import Q  # Line 5: Import Q for queries

@login_required  # Line 6: Restrict to authenticated users
@cache_page(60)  # Line 7: Cache view for 60 seconds
def conversation_view(request, conversation_id):  # Line 8: Define conversation view
    """Display messages in a conversation."""  # Line 9: Docstring for view
    messages = Message.objects.filter(  # Line 10: Query messages
        Q(sender=request.user, recipient_id=conversation_id) |  # Line 11: Messages sent by user
        Q(sender_id=conversation_id, recipient=request.user)  # Line 12: Messages received by user
    ).select_related('sender', 'recipient').only(  # Line 13: Optimize query
        'id', 'sender__username', 'recipient__username', 'content', 'timestamp'  # Line 14: Limit fields
    ).order_by('timestamp')  # Line 15: Sort chronologically
    return render(request, 'messaging/conversation.html', {  # Line 16: Render template
        'messages': messages,  # Line 17: Pass messages
        'conversation_user_id': conversation_id  # Line 18: Pass user ID
```

```
    })
```

3.3 Code: conversation.html

Listing 7: messaging/templates/messaging/conversation.html

```
<!DOCTYPE html>  <!-- Line 1: Declare HTML5 -->
<html>  <!-- Line 2: Root HTML element -->
<head>  <!-- Line 3: Metadata section -->
    <title>Conversation</title>  <!-- Line 4: Page title -->
</head>
<body>  <!-- Line 5: Visible content -->
    <h1>Conversation</h1>  <!-- Line 6: Page header -->
    {% for message in messages %}  <!-- Line 7: Loop over messages -->
        <div>  <!-- Line 8: Group message content -->
            <p><strong>{{ message.sender.username }} to {{ message.recipient.
                username }}:</strong> {{ message.content }}</p>  <!-- Line 9:
                Message details -->
            <p><strong>Sent:</strong> {{ message.timestamp }}</p>  <!-- Line
                10: Timestamp -->
        </div>
    {% empty %}  <!-- Line 11: Fallback for empty list -->
        <p>No messages in this conversation.</p>  <!-- Line 12: Empty message
            -->
    {% endfor %}  <!-- Line 13: End loop -->
</body>
</html>
```

3.4 Code: urls.py

Listing 8: messaging/urls.py

```
from django.urls import path  # Line 1: Import path for URL routing
from . import views  # Line 2: Import views module

urlpatterns = [  # Line 3: Define URL patterns
    path('inbox/', views.inbox_view, name='inbox'),  # Line 4: Map inbox URL
    path('conversation/<int:conversation_id>/', views.conversation_view, name='
        conversation'),  # Line 5: Map conversation URL
]
```

4 References

- Django Custom Managers: https://docs.djangoproject.com/en/stable/topics/db/managers/
- QuerySet Optimization: https://docs.djangoproject.com/en/stable/ref/models/querysets/
- Django Caching: https://docs.djangoproject.com/en/stable/topics/cache/
- Cache Page Decorator: https://docs.djangoproject.com/en/stable/topics/cache/#the-per-view-cache
- Django Indexes: https://docs.djangoproject.com/en/stable/ref/models/indexes/
- Q Objects: https://docs.djangoproject.com/en/stable/topics/db/queries/#complex-lookups-w
- Django Authentication: https://docs.djangoproject.com/en/stable/topics/auth/
- Django Redis Cache: https://django-redis-cache.readthedocs.io/en/latest/

- Django Performance Tips: https://www.digitalocean.com/community/tutorials/how-to-optimize
- Django Channels: https://channels.readthedocs.io/en/stable/
- Django Templates: https://docs.djangoproject.com/en/stable/topics/templates/
- Django URLs: https://docs.djangoproject.com/en/stable/topics/http/urls/
- Django ORM Optimization: https://testdriven.io/blog/django-orm-optimization/
- Caching in Django: https://realpython.com/caching-in-django-with-redis/
- Django Messages Framework: https://micropyramid.com/blog/basics-of-django-messages-framew
- Unread Messages: https://stackoverflow.com/questions/36913545/django-display-count-of-un
- Django View Decorators: https://www.geeksforgeeks.org/django-view-decorators/
- Building a Chat App: https://www.agiliq.com/blog/2017/08/building-chat-application-with-d
- Django Query Optimization: https://hansonkd.medium.com/optimizing-django-querysets-7b7b29
- LaTeX Listings: https://ctan.org/pkg/listings