**Low Level Caching in Django 5**

Videono:61

Codefile:ch52

# Low-Level Caching in Django

## 1. Configure Cache in settings.py

Django supports multiple backends. Examples:

# Database Caching

CACHES = {

    'default': {

        'BACKEND': 'django.core.cache.backends.db.DatabaseCache',

        'LOCATION': 'student\_cache',  # DB table name

    }

}

# File Based Caching

# CACHES = {

#     'default': {

#         'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',

#         'LOCATION': 'student\_cache',

#     }

# }

# Local Memory Caching

# CACHES = {

#     'default': {

#         'BACKEND': 'django.core.cache.backends.locmem.LocMemCache',

#         'LOCATION': 'unique-snowflake',

#     }

# }

👉 For database caching, first run:

python manage.py createcachetable student\_cache

## 2. Low-Level Cache API

Import:

from django.core.cache import cache

Available methods:

| **Method** | **Purpose** |
| --- | --- |
| cache.set(key, value, timeout, version) | Store data in cache |
| cache.get(key, default, version) | Retrieve data |
| cache.add(key, value, timeout, version) | Add only if key doesn’t exist |
| cache.get\_or\_set(key, default, timeout, version) | Get value if exists, else set it |
| cache.set\_many(dict, timeout) | Set multiple keys at once |
| cache.get\_many(keys, version) | Get multiple keys |
| cache.delete(key, version) | Delete a single key |
| cache.delete\_many(keys, version) | Delete multiple keys |
| cache.clear() | Clear entire cache |
| cache.touch(key, timeout) | Reset expiry for a key |
| cache.incr(key, delta) | Increment integer value |
| cache.decr(key, delta) | Decrement integer value |
| cache.close() | Close cache connection (mainly for DB/Redis backends) |

from django.core.cache import cache

cache.set(key, value, timeout=DEFAULT\_TIMEOUT, version=None)

cache.get(key, default=None, version=None)

cache.add(key, value, timeout=DEFAULT\_TIMEOUT, version=None)

cache.get\_or\_set(key, default, timeout=DEFAULT\_TIMEOUT, version=None)

cache.set\_many(dict, timeout)

cache.get\_many(keys, version=None)

cache.delete(key, version=None)

cache.delete\_many(keys, version=None)

cache.clear()

cache.touch(key, timeout=DEFAULT\_TIMEOUT, version=None)

cache.incr(key, delta=1, version=None)

cache.decr(key, delta=1, version=None)

cache.close()

## 3. Examples with Explanation

### ✅ Basic Set & Get

def course(request):

    # Try to get value of 'movie', if not found return "has\_expired"

    mv = cache.get('movie', 'has\_expired')

    if mv == 'has\_expired':  # Key not found

        cache.set('movie', 'Harry Potter', 60)  # Key expires after 60s

        mv = cache.get('movie')

    return render(request, 'student/course.html', {'mv': mv})

### ✅ Using get\_or\_set

def course(request):

    # If 'movie' exists → return it

    # If not → set 'Harry Potter' for 15 seconds

    mv = cache.get\_or\_set('movie', 'Harry Potter', 15)

    return render(request, 'student/course.html', {'mv': mv})

### ✅ Versions in Cache

def course(request):

    # Default version = 1

    mv = cache.get\_or\_set('movie', 'Harry Potter', 15)

    # Same key but different version → stored separately

    mv1 = cache.get\_or\_set('movie', 'The One', 15, version=2)

    print(mv1)  # Output: The One

    return render(request, 'student/course.html', {'mv': mv})

### ✅ Multiple Keys (set\_many / get\_many)

def course(request):

    # Set multiple values

    cache.set\_many({'student\_name': 'Talha', 'student\_roll': 100}, 30)

    # Get multiple values at once

    stu = cache.get\_many(['student\_name', 'student\_roll'])

    return render(request, 'student/course.html', {'stu': stu})

### ✅ Delete Key

def course(request):

    cache.delete('movie', version=2)  # Delete only version=2

    return render(request, 'student/course.html')

### ✅ Clear Entire Cache

def course(request):

    cache.clear()  # Removes all keys from cache

    return render(request, 'student/course.html'

### ✅ Touch Key (Extend Expiry)

def course(request):

    cache.set('counter', 1, 30)  # Expires in 30s

    cache.touch('counter', 60)   # Extend expiry to 60s

    return render(request, 'student/course.html')

### ✅ Increment / Decrement

def course(request):

    cache.set('views', 1, 60)

    cache.incr('views', 2)   # views = 3

    cache.decr('views', 1)   # views = 2

    return render(request, 'student/course.html', {'views': cache.get('views')})

## 4. Notes & Best Practices

* 🔑 **Expiration (timeout)**:
  + timeout=None → never expires.
  + timeout=0 → won’t be cached.
* 🔑 **Overwriting**: cache.set() will overwrite existing values, even before timeout.
* 🔑 **Safe add**: Use cache.add() if you don’t want to overwrite.
* 🔑 **Versions**: Same key can have different versions (default = 1).
* 🔑 **Production**: Use **Memcached** or **Redis** instead of DB or file cache for better performance.