# Understanding caching techniques in CodeIgniter

**Topics :** Codeigniter
**Written on** March 01, 2024



In CodeIgniter, caching is a technique used to store the output of a particular operation so that the same operation can be performed faster in the future by retrieving the cached data instead of recalculating it. CodeIgniter provides several caching mechanisms to improve the performance of your web applications. Let's explore some of the caching techniques available in CodeIgniter:

## 1. Page Caching:

Page caching involves storing the entire output of a web page in a file, which can be served directly by the web server without executing the PHP code again. This is particularly useful for pages that are relatively static and don't change frequently.

To enable page caching in CodeIgniter, you can use the `$this->output->cache()` method in your controller method:

```
$this->output->cache($minutes); // Cache the output for a specified number of minutes
```

## 2. Database Query Caching:

Database query caching involves storing the result sets of frequently executed database queries in memory or disk, so that subsequent executions of the same queries can be served from the cache instead of hitting the database again.

To enable database query caching in CodeIgniter, you can set the caching preferences in the database configuration file (`application/config/database.php`):

```
$db['default']['cache_on'] = TRUE; // Enable database query caching
$db['default']['cachedir'] = APPPATH.'cache'; // Set the cache directory
```

## 3. Fragment Caching:

Fragment caching involves caching specific parts of a web page, such as sections or widgets, rather than the entire page. This allows you to cache only the parts of the page that are expensive to generate or don't change frequently.

To implement fragment caching in CodeIgniter, you can use the Cache Library to manually cache

specific sections of your views:

```
$this->load->driver('cache');

if (!$output = $this->cache->get('cache_key')) {
    // Generate the output
    $output = $this->load->view('partial_view', $data, TRUE);

    // Cache the output
    $this->cache->save('cache_key', $output, $expiration_time);
}

echo $output;
```

## 4. Caching with Redis or Memcached:

CodeIgniter also supports caching with external caching systems like Redis or Memcached, which provide fast, distributed, and scalable caching solutions.

To use Redis or Memcached caching in CodeIgniter, you need to install the appropriate caching driver and configure it in the `config.php` file:

```
$config['redis']['socket_type'] = 'tcp';
$config['redis']['socket'] = '/var/run/redis/redis.sock';
$config['redis']['password'] = NULL;
$config['redis']['timeout'] = 0;
```

```
$config['memcached']['host'] = '127.0.0.1';
$config['memcached']['port'] = 11211;
$config['memcached']['weight'] = 1;
```

Then, you can use the Cache Library to store and retrieve data from Redis or Memcached.

## 5. Browser Caching:

Browser caching involves instructing the client's browser to cache static assets such as images, CSS, and JavaScript files. This can significantly reduce page load times for returning visitors.

To enable browser caching in CodeIgniter, you can set appropriate caching headers in your controller or via `.htaccess` files:

```
$this->output->set_header('Cache-Control: max-age=3600'); // Cache for 1 hour
```

```
$this->output->set_header('Expires: ' . gmdate('D, d M Y H:i:s', time() +
3600) . ' GMT');
```

Apache

```
<FilesMatch "\.(jpg|jpeg|png|gif|js|css)$">
   Header set Cache-Control "max-age=3600, public"
</FilesMatch>
```