DRUPAL DEVELOPER DAYS

Burgas, Bulgaria - 2024

An Introduction to Caching in Drupal

A global picture

Luis Ruiz Peidró

lpeidro

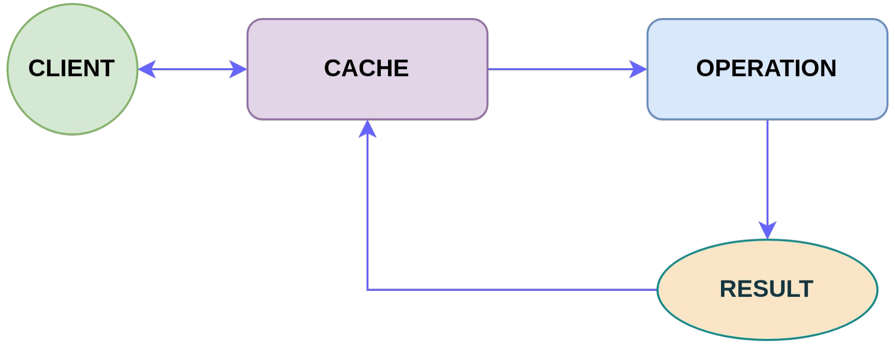
Drupal Backend Developer

metadrop

Index

Introduction

Definition



It is a data storage layer that stores the results of complex operations so that when they are required again, there is no need to redo them.

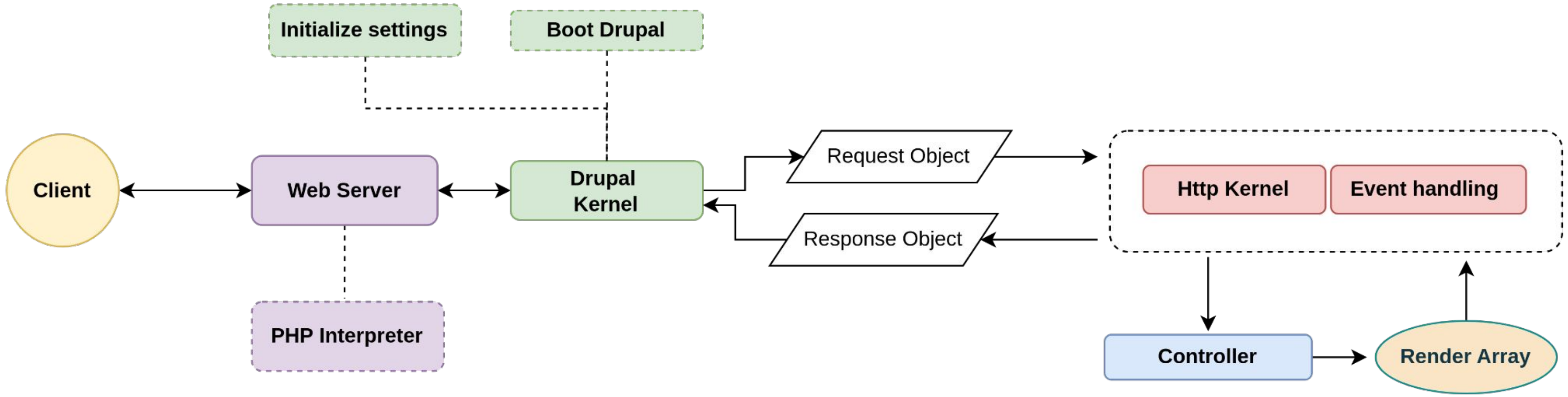
Why perform an operation more than once if the result is the same?

Benefits

- **Increase data availability**: For example, ensuring that a webpage loads immediately for the client.

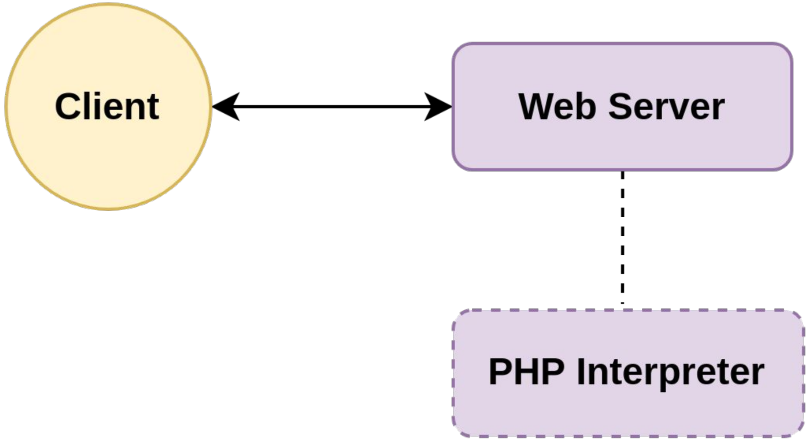
- **Resource savings**: Complex operations do not need to be executed every time they are requested.

Request flow



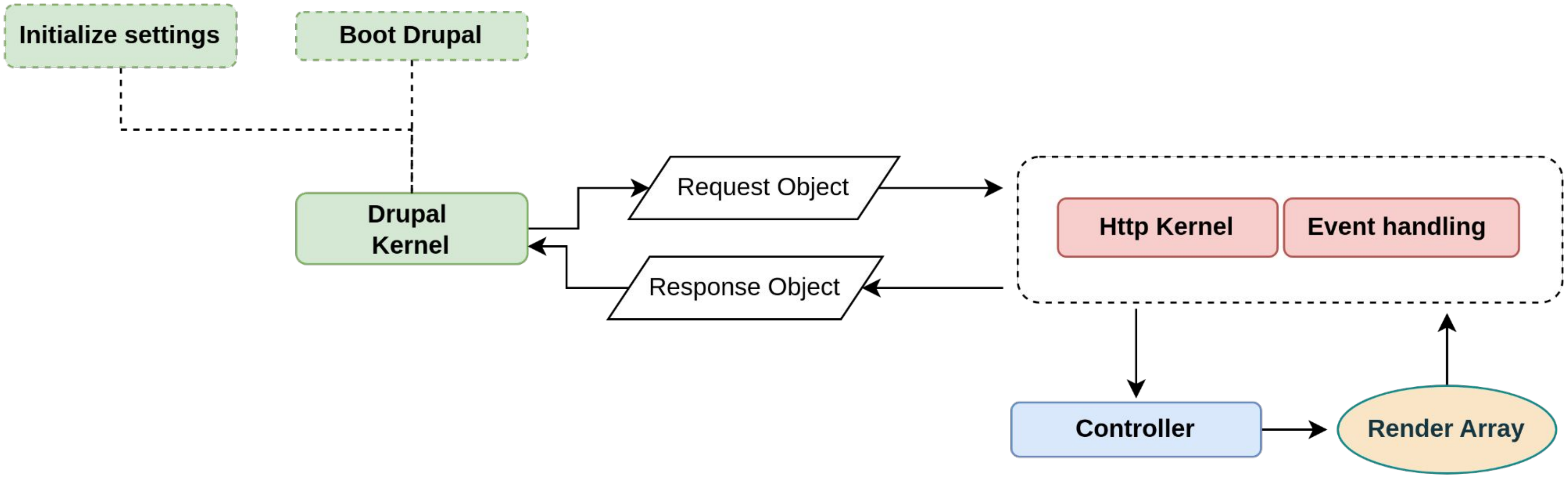
PHP
and Drupal

OPcache



- OPcache is a PHP extension.
- Store precompiled PHP script.
- Activating this extension is essential for any Drupal site to improve performance.

Drupal Cache System



Cache bins => cache.bootstrap, cache.discovery, cache.config, cache.entity, cache.render...

Cache backend - definition

- The component responsible for **managing cached data** (storing, retrieving, invalidating) . It is the intermediary layer between the data source and the one requesting the data.

- The **main feature** that differentiates one Cache Backend from another is **how and where it stores** the data, for example, cache objects can be stored in: Database, Files, Memory...

Cache backend - types (examples)

- **Database Backend**: This is the default Cache Backend. In this case, it uses the Drupal database to store cached data.

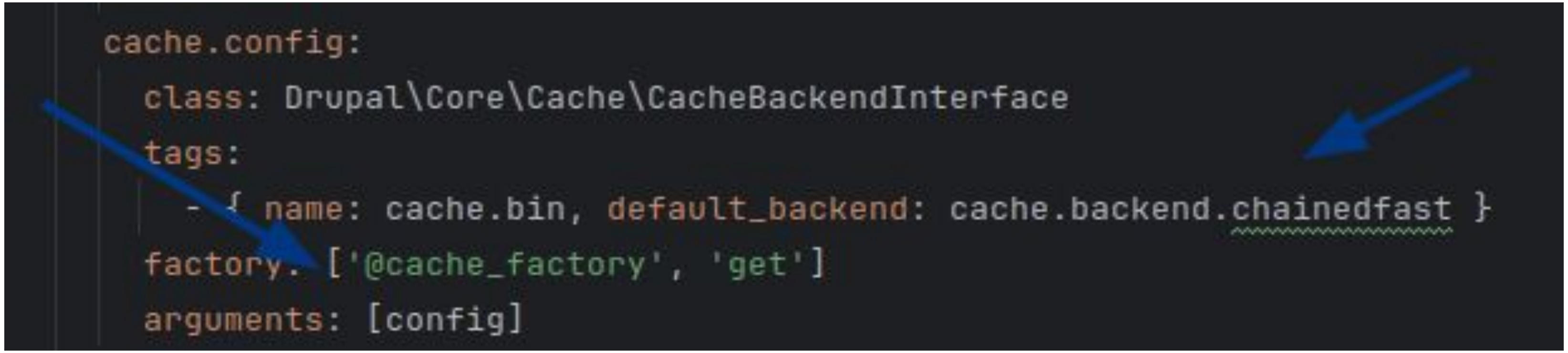
- **Chained Fast Backend**: It is a Cache Backend with two cache layers: one fast but less consistent (in memory) and another more consistent (in database).

Cache bin

Cache Bins act like boxes in the Drupal cache system. Instead of having all cached data in one place, for example in a database table, we can **store** them **in different compartments**.

Cache Bins are defined by what **type of data they store**.

Cache bin example (core.services.yml)



```yaml
cache.config:
  class: Drupal\Core\Cache\CacheBackendInterface
  tags:
    - { name: cache.bin, default_backend: cache.backend.chainedfast }
  factory: ['@cache_factory', 'get']
  arguments: [config]
```

Some Cache Bins defined by core

- **cache.bootstrap**: necessary information for Drupal's execution.
- **cache.config**: configuration and configuration entities.
- **cache.entity**: stores the values of existing content entities.
- **cache.render**: contains cached HTML strings.

Cached Object Dependencies - Invalidation

Cache tags: establishes a dependency on the sources.

Max-Age: establishes a temporary dependency, which is an expiration date for the cache object.

Cached Object Dependencies - Variations

Cache context: establishes a dependency on the context, creating variations based on the context.

For example, a block on a page may vary based on the role of the client requesting it, generating cache versions based on the role.

Cache metadata examples

Cache tag:

- node:5
- user:3
- node_list
- node_list:article
- config:system.performance
- library_info

Cache context:

- user.roles
- theme
- user.roles:anonymous
- languages
- url

Cache Invalidation

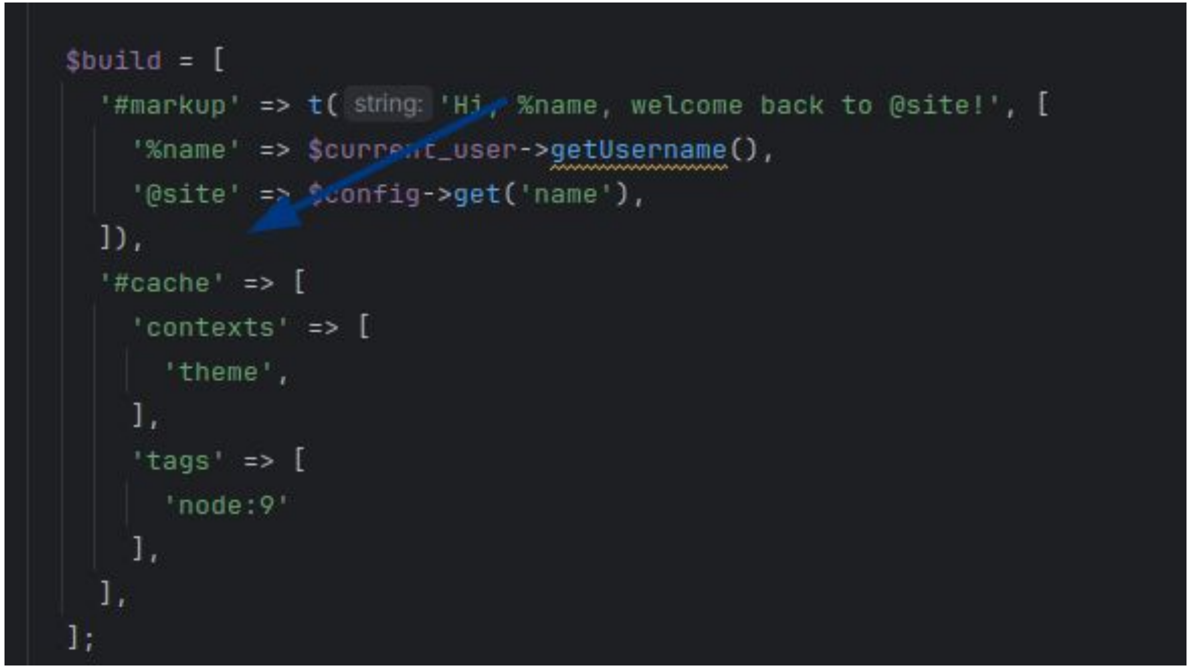
Cache invalidation. The process where cached data is invalidated because it has become outdated.

- **Cache Max Age**: sets an expiration date for the cached object.
- **Cache tags**: set tags to associate cached objects with the source.

Interaction with the Cache API

Interacting directly with the **Cache API** of Drupal, taking care of caching objects, retrieval, and invalidation ourselves.

Or indirectly through the **Render API**. This second system is the most common.



```php
$build = [
  '#markup' => t( string: 'Hi, %name, welcome back to @site!', [
    '%name' => $current_user->getUsername(),
    '@site' => $config->get('name'),
  ]),
  '#cache' => [
    'contexts' => [
      'theme',
    ],
    'tags' => [
      'node:9'
    ],
  ],
];
```

Cache Dependency Propagation

The dependencies defined in the different components are propagated to their ancestors for the invalidation to be effective.

Response caching

Internal Page Cache:

- Anonymous user.
- Static pages.
- Cache context and max age does not work.
- Only enable in small and medium sites.

Internal Dynamic Cache

- Anonymous user and logged users.
- Static and dynamic pages.
- Enable in any case.

Internal Dynamic Cache

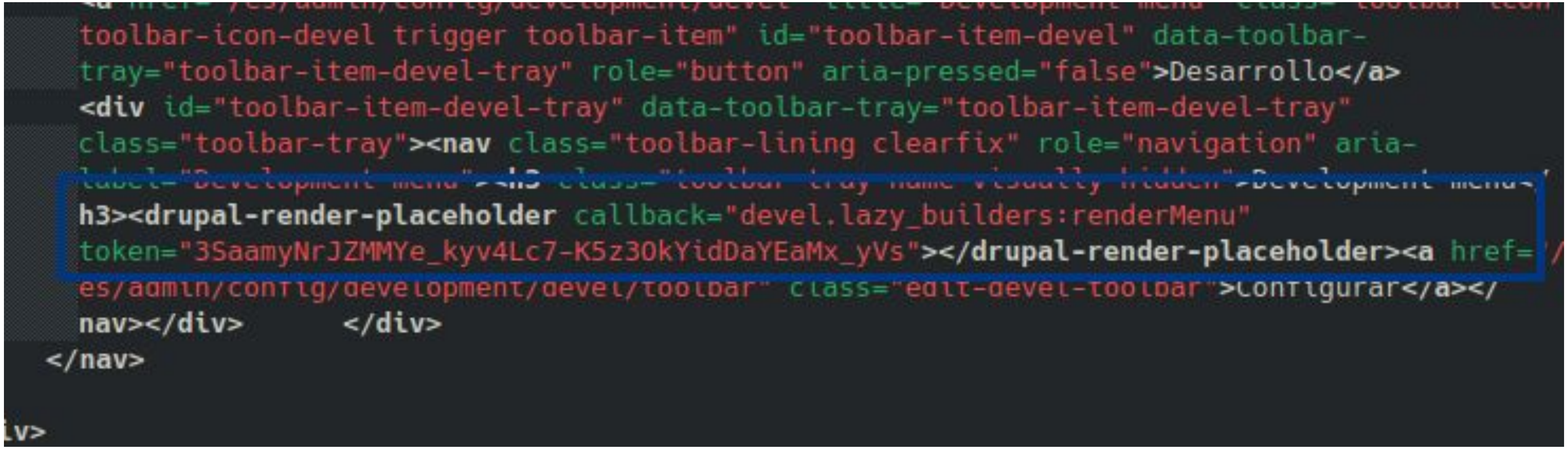
It caches response objects with dynamic components by replacing those components with **placeholders**.

It waits until the last moment to render those components and replace the placeholders.

Internal Dynamic Cache

It caches response objects with dynamic components by replacing those components with **placeholders**.

It waits until the last moment to render those components and replace the placeholders.

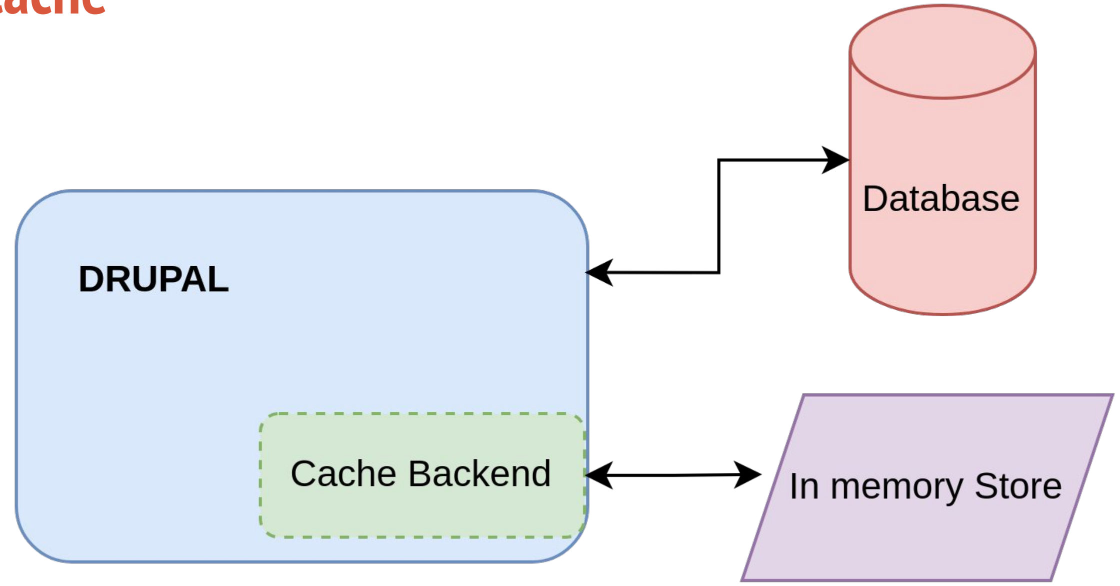


```
    <a href="/es/admin/config/development/devel" title="Development menu" class="toolbar-icon
    toolbar-icon-devel trigger toolbar-item" id="toolbar-item-devel" data-toolbar-
    tray="toolbar-item-devel-tray" role="button" aria-pressed="false">Desarrollo</a>
    <div id="toolbar-item-devel-tray" data-toolbar-tray="toolbar-item-devel-tray"
    class="toolbar-tray"><nav class="toolbar-lining clearfix" role="navigation" aria-
    label="Development menu"><h3 class="toolbar-tray-name visually-hidden">Development menu</
    h3><drupal-render-placeholder callback="devel.lazy_builders:renderMenu"
    token="3SaamyNrJZMMYe_kyv4Lc7-K5z3OkYidDaYEaMx_yVs"></drupal-render-placeholder><a href="/
    es/admin/config/development/devel/toolbar" class="edit-devel-toolbar">Configurar</a></
    nav></div>      </div>
  </nav>

iv>
```

Third Party Systems

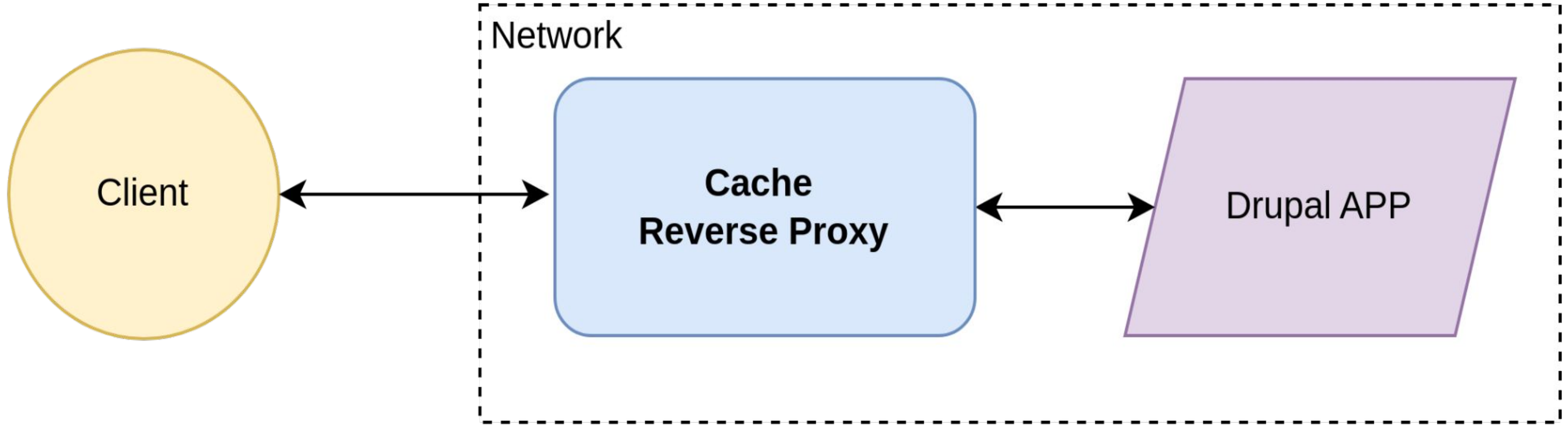
In Memory Cache



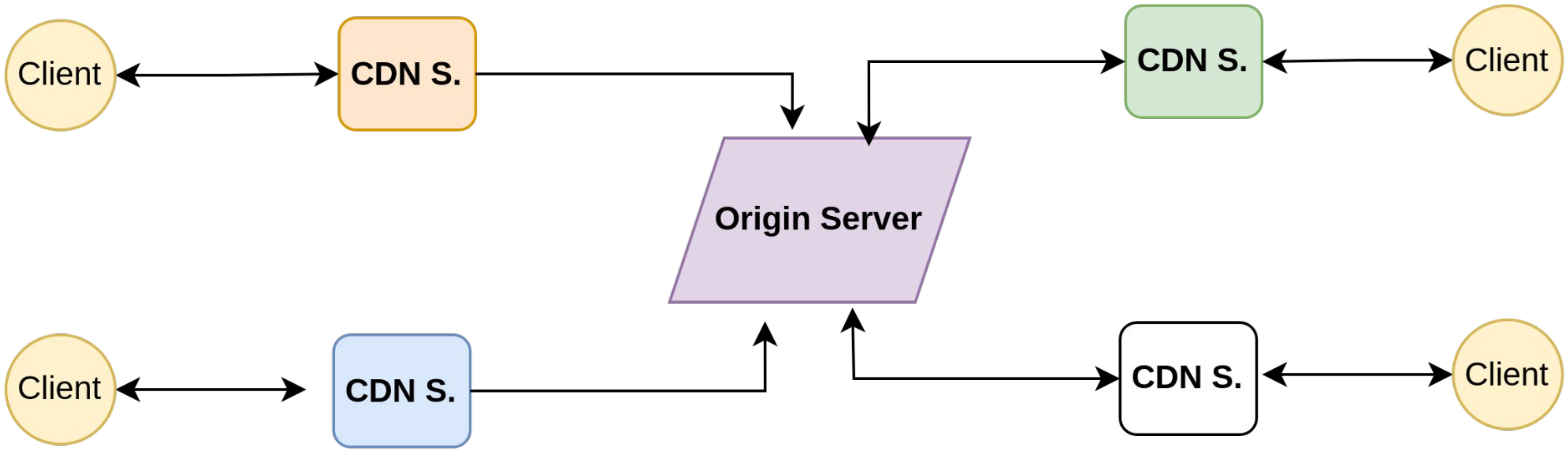
In Memory Cache Integration

- Memcache API and Integration module.
- Redis module.

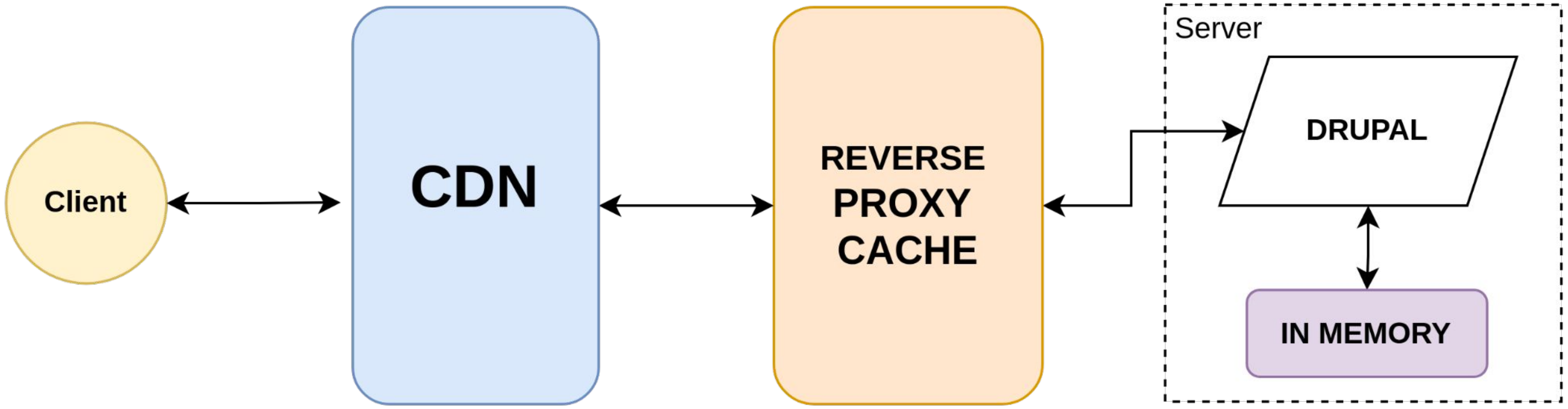
Caching Reverse Proxy



CDN (Content Delivery Network)



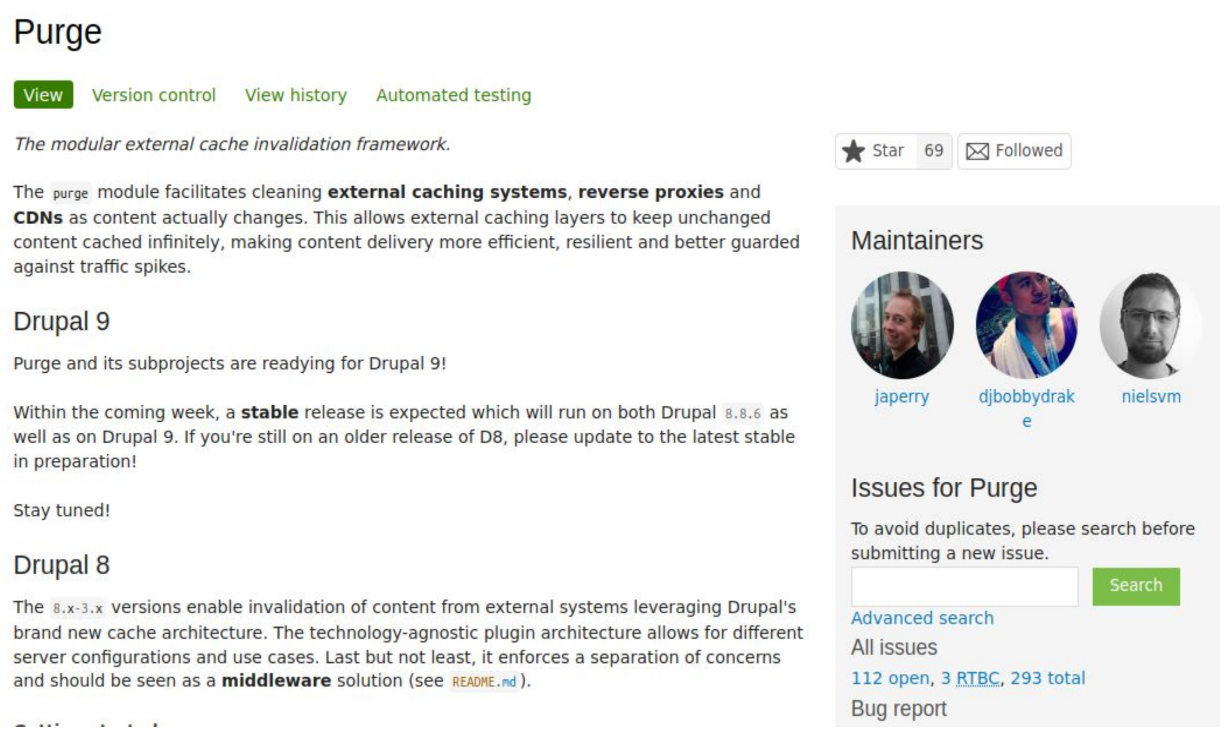
Complex architecture



Invalidation cache in complex architecture

- **Max Age (TTL)**. The easiest way:
  - Communicated to the different caches in the headers of the response.
  - The cache system recreated, or ask to the source if it has to be recreated.
  - There are some delais between the update of the source, and the recreation of the cache.
- **Cache tags**:
  - Send the cache tags in the response headers.
  - Drupal has to inform about invalidations.
- **URLs**:
  - Drupal requests the cache to remove the cached objects with a specific URL.

Purge Caches



Purge

View Version control View history Automated testing

The modular external cache invalidation framework.

The `purge` module facilitates cleaning **external caching systems**, **reverse proxies** and **CDNs** as content actually changes. This allows external caching layers to keep unchanged content cached infinitely, making content delivery more efficient, resilient and better guarded against traffic spikes.

Drupal 9

Purge and its subprojects are readying for Drupal 9!

Within the coming week, a **stable** release is expected which will run on both Drupal `8.8.6` as well as on Drupal 9. If you're still on an older release of D8, please update to the latest stable in preparation!

Stay tuned!

Drupal 8

The `8.x-3.x` versions enable invalidation of content from external systems leveraging Drupal's brand new cache architecture. The technology-agnostic plugin architecture allows for different server configurations and use cases. Last but not least, it enforces a separation of concerns and should be seen as a **middleware** solution (see `README.md`).

Star 69 Followed

Maintainers

japerry djbobbydrake nielsvm

Issues for Purge

To avoid duplicates, please search before submitting a new issue.

Search

Advanced search

All issues

112 open, 3 RTBC, 293 total

Bug report

Response Headers

Why are response headers **useful**:

- To send configuration about caching to other systems, for example, Browser, CDN, etc. (Max Age, Cache tags)
- For us, to debug.

Standard HTTP headers

Cache control: controlling caching in both the browser and other cache layers like CDNs and proxies.

Server: Software used by handling the request. With this parameter we can found how is sending the request, for example, a CDN.

Drupal headers

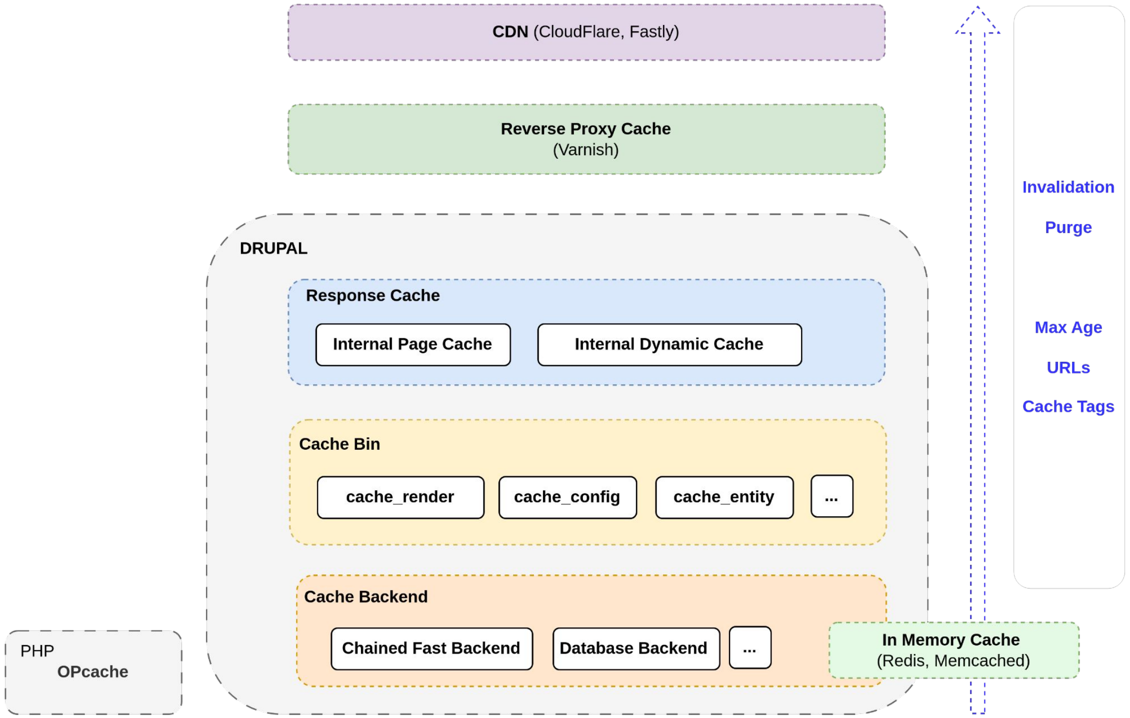
Standar Drupal Headers

- X-Drupal-Dynamic-Cache
- X-Drupal-Cache

Debugging headers:

- X-Drupal-Cache-Context.
- X-Drupal-Cache-Tags

Summary


CDN (CloudFlare, Fastly)
Reverse Proxy Cache
(Varnish)
DRUPAL
Response Cache
Internal Page Cache
Internal Dynamic Cache
Cache Bin
cache_render
cache_config
cache_entity
...
Cache Backend
Chained Fast Backend
Database Backend
...
In Memory Cache
(Redis, Memcached)
PHP
OPcache
Invalidation
Purge
Max Age
URLs
Cache Tags

Conclusion

A good implementation of caching is essential in any Drupal project and involves both the development and infrastructure teams.

Questions

Thank you for joining us!

Stay connected with **Drupal Bulgaria Association** for more exciting events, news, and updates!

Also visit our website: **https://ddd24.drupalcamp.bg**

And don‘t forget to visit our sponsors' stands.

Their support helps make events like this possible!

1x INTERNET

JAKALA

dropsolid DIGITAL EXPERIENCE COMPANY

drupalaustria

sepia web innovation