Demonstrate the ability to manage the cache

Describe cache types and the tools used to manage caches.

- config
- layout
- block_html
- collections
- db_ddl
- eav
- full_page
- reflection
- translate
- config_integration
- config_integration_api
- config_webservice

Description for most cache type can be found on the Magento DevDocs - Manage the cache

Commands:

- `magento setup:db-schema:upgrade`
- `magento cache:status`, `magento cache:enable`, `magento cache:disable`
- `magento cache:clean`, `magento cache:flush`

Init:

frontend:

```
\Magento\Framework\App\ObjectManager\ConfigLoader::load
cacheType = config, frontend = default
\Magento\Framework\App\Cache\Frontend\Pool::_initialize
\Magento\Framework\App\Cache\Frontend\Factory::create
\Zend_Cache::_makeFrontend
\Zend_Cache_Core::__construct
```

backend:

- \Zend_Cache_Backend
- \Zend_Cache_Backend_File
- \Magento\Framework\Cache\Backend\Database

How do you add dynamic content to pages served from the full page cache?

1. Mark any block `cacheable="false"` in layout xml - whole page is uncacheable. Example - checkout
2. Disable caching in controller using *headers*: `$page->setHeader('Cache-Control', 'no-store, no-cache, must-revalidate, max-age=0', true);`
3. Marking block property isScopePrivate - will be loaded via AJAX - deprecated
4. ESI when Varnish enabled, set TTL - Example - menu block
5. Configure page variations - extend *http context*, more cached versions of same page - store view, customer group, language, currency, is logged in `\Magento\Framework\App\Http\Context::getVaryString`

Only *GET* and *HEAD* are cached

Clear cache \Magento\Framework\DataObject\IdentityInterface

when giving product page, must somehow send varnish tags

Any block can implement IdentityInterface. After rendering layout and before sending output, all blocks are examined for implementing this interface. Cache tags are collected as merge of all blocks getIdentities() tags.

```
\Magento\PageCache\Model\Layout\LayoutPlugin::afterGetOutput
  X-Magento-Tags = merge(all blocks.getIdentities)
```

block ListProduct:

- every product[].getIdentities
  - cat_p_{productId}
  - *if changed categories* - cat_p_c_{categoryId}
  - *if changed status* - every category[] cat_p_c_{categoryId}
  - *if frontend* - 'cat_p'
- cat_c_p_{categoryId}

block product/view:

- \Magento\Catalog\Model\Product::getIdentities:
  - cat_p_{productId}
  - *if changed categories* - cat_p_c_{categoryId}
  - *if changed status* - every category[] cat_p_c_{categoryId}
  - *if frontend* - 'cat_p'
- *if current_category* - cat_c_{categoryId}

after reindex, must somehow clean cache

- any indexer.execute – by MView
- any indexer.executeFull
- \Magento\Framework\Indexer\CacheContext::registerTags

plugin \Magento\Indexer\Model\Processor:

\Magento\Indexer\Model\Processor\CleanCache::afterUpdateMview

- event `clean_cache_after_reindex`
- clean cache cacheContext->getIdentities()

\Magento\Indexer\Model\Processor\CleanCache::afterReindexAllInvalid

- event `clean_cache_by_tags`
- clean cache cacheContext->getIdentities()

module-cache-invalidate observer `clean_cache_after_reindex`
\Magento\CacheInvalidate\Observer\InvalidateVarnishObserver::execute
\Magento\CacheInvalidate\Model\PurgeCache::sendPurgeRequest

Describe how to operate with cache clearing.

How would you clean the cache? In which case would you refresh cache/flash cache storage?

> To purge out-of-date items from the cache, you can clean or flush cache types:

- Cleaning a cache type deletes all items from enabled Magento cache types only. In other words, this option does not affect other processes or applications because it cleans only the cache that Magento uses.

  Disabled cache types are not cleaned.

- Flushing a cache type purges the cache storage, which might affect other processes applications that are using the same storage.

Flush cache types if you've already tried cleaning the cache and you're still having issues that you cannot isolate.

– Magento DevDocs - Manage the cache

Sessions and caching data should never be stored in one database in Redis. In that case you'll not have problems with flushing the cache.

Describe how to clear the cache programmatically.

To clear the cache programmatically you neeed to call next the methods:

- \Magento\Framework\App\CacheInterface::remove($identifier) - remove cached data by identifier
- \Magento\Framework\App\CacheInterface::clean($tags = []) - clean cached data by specific tag

What mechanisms are available for clearing all or part of the cache?

Dispatch a `clean_cache_by_tags` event with parameter of the object you want to clear from the cache.

Example: \Magento\Framework\Model\AbstractModel (afterSave, afterDelete methods)

```
<?php
public function afterSave()
{
    $this->cleanModelCache();
    $this->_eventManager->dispatch('model_save_after', ['object' => $this]);
    $this->_eventManager->dispatch('clean_cache_by_tags', ['object' => $this]);
    $this->_eventManager->dispatch($this->_eventPrefix . '_save_after', $this-
>_getEventData());
    $this->updateStoredData();
    return $this;
}

public function cleanModelCache()
{
    $tags = $this->getCacheTags();
    if ($tags !== false) {
        $this->_cacheManager->clean($tags); //
\Magento\Framework\App\CacheInterface
    }
    return $this;
}
```

Default `clean_cache_by_tags` event observers are:

- Magento\PageCache\Observer\FlushCacheByTags - if Built-In caching is enabled
- Magento\CacheInvalidate\Observer\InvalidateVarnishObserver- if Varnish caching is enabled

Links

- Magento DevDocs - Magento cache overview
- Magento DevDocs - Configure caching
- Magento DevDocs - Partial caching
- Magento DevDocs - Full Page caching
- Magento DevDocs - Private content