Utilize modes and application initialization

Identify the steps for application initialization.

app/bootstrap.php:

- composer autoloader, functions, umask, timezone UTC, php precision

\Magento\Framework\App\Bootstrap::*create*

- configure autoloader - PSR-4 prepend generation\Magento

\Magento\Framework\App\Bootstrap::*createApplication*

- just call object manager->create

\Magento\Framework\App\Bootstrap::*run*

- set error handler
- assert maintenance
- assert installed
- response = application->*launch*()
- response->*sendResponse*()
- on error: application->catchException
- on missed error:
  - dev mode: print exception trace
  - normal mode: log, message “An error has happened during application run. See exception log for details.”

Application class

bootstrap->createApplication()

- \Magento\Framework\App\Http - index.php, pub/index.php load config area by front name front controller->dispatch event `controller_front_send_response_before`
- \Magento\Framework\App\Cron - pub/cron.php config area `crontab` load translations dispatch event `default`
- \Magento\MediaStorage\App\Media - pub/get.php access /media/* when using DB image storage and physical file doesn’t exist
- \Magento\Framework\App\StaticResource - pub/static.php 404 in production /$area/$resource/$file/… params, load config by params sends file in response assetRepo->createAsset - \Magento\Framework\View\Asset\File assetPublisher->publish - materialize (copy/symlink) file if doesn’t exist
- \Magento\Indexer\App\Indexer - module-indexer, unused?
- \Magento\Backend\App\UserConfig - module-backend, unused?

Notes:

- Responsibility - launch() and return response

- Roughly different application class matches different physical file entry points (index.php, media.php, get.php, static.php, cron.php)
- Front controller exists only in Http application
- There's no CLI application, Symfony command is used

HTTP application

\Magento\Framework\App\Http::launch

1. detect config area by front name

```
<?php
$areaCode = $this->_areaList->getCodeByFrontName($this->_request->getFrontName());
$this->_state->setAreaCode($areaCode);
$this->_objectManager->configure($this->_configLoader->load($areaCode));
```

`\Magento\Framework\App\AreaList` - areas from argument di.xml (AreaList)

- frontend = [frontname null, router "standard"] —*default when nothing matched*
- adminhtml - [frontNameResolver=…, router "admin"] \Magento\Backend\App\Area\FrontNameResolver::getFrontName(checkhost) system config `admin/url/use_custom`, `admin/url/custom`
- crontab = null
- webapi_rest = [frontName `/rest`]
- webapi_soap = [frontname `/soap`]

1. ObjectManagerInterface->configure() - selected area code
2. result = FrontControllerInterface->dispatch()
3. ResultInterface->renderResult() into response object
4. event `controller_front_send_response_before` (request, response)

How would you design a customization that should act on every request and capture output data regardless of the controller?

- event `controller_front_send_response_before`

Describe front controller responsibilities

Front controller exists only in Http application (pub/index.php)

- Same entry point, how to execute different logic? Via different DI preference depending on detected config area (areaList->getCodeByFrontName)
- *Default* global preference app/etc/di.xml - Magento\Framework\App\FrontController
- "frontend", "adminhtml", "crontab" area code - no preference, use default*App\FrontController*
- "webapi_rest (frontName `/rest`) - preference module-webapi/etc/webapi_rest/di.xml - *\Magento\Webapi\Controller\Rest*
- "webapi_soap" (frontname `/soap`) - preference module-webapi/etc/webapi_soap/di.xml - *\Magento\Webapi\Controller\Soap*

\Magento\Framework\App\FrontController:

- routerList
- action = router[].match
- result = action.dispatch() or action.execute()
- noroute action fallback

Router match - action can be:

- generic \Magento\Framework\App\ActionInterface::execute - not used?
- \Magento\Framework\App\Action\AbstractAction::dispatch - context, request, response, result factory, result redirect factory

Dispatch/execute action - result can be:

- \Magento\Framework\Controller\ResultInterface - renderResult, setHttpResponseCode, setHeader

  Implementations:

  - Result\Raw -> Result\AbstractResult
  - Result\Json -> Result\AbstractResult
  - Result\Forward -> Result\AbstractResult
  - \Magento\Framework\View\Result\Layout -> Result\AbstractResult
  - \Magento\Framework\View\Result\Page -> \Magento\Framework\View\Result\Layout
  - Result\Redirect -> Result\AbstractResult
- \Magento\Framework\App\ResponseInterface - sendResponse

  Implementations:

  - Console\Response
  - \Magento\MediaStorage\Model\File\Storage\FileInterface -> \Magento\Framework\App\Response\Http
  - \Magento\Framework\HTTP\PhpEnvironment\Response -> \Zend\Http\PhpEnvironment\Response
  - \Magento\Framework\Webapi\Response -> \Magento\Framework\HTTP\PhpEnvironment\Response
  - \Magento\Framework\Webapi\Rest\Response -> \Magento\Framework\Webapi\Response

\Magento\Webapi\Controller\Rest -> \Magento\Framework\App\FrontControllerInterface:

- preference for FrontController set in etc/webapi_rest/di.xml
- process path [/$store]/… - specific store, [/all]/… - admin store (0), /… - default store
- a. process schema request /schema
- b. or process api request (resolve route, invoke route -> service class with params)

\Magento\Webapi\Controller\Soap -> \Magento\Framework\App\FrontControllerInterface:

- process path (same as REST)
- a. generate WSDL ?wsdl
- b. or generate WSDL service list ?wsdl_list
- c. or handle SOAP (native PHP)

In which situations will the front controller be involved in execution, and how can it be used in the scope of customizations?

- Front controller is only used in HTTP application - pub/index.php
- involved in `frontend` and `adminhtml` (HTTP\App\FrontController), `webapi_rest` (Controller\Rest), `webapi_soap` (Controller\Soap) areas
- HTTP customization - register router and place custom code in match() method