Utilize the CLI

Describe the usage of bin/magento commands in the development cycle.

Demonstrate an ability to create a deployment process.

Modes: *default*, *developer*, *production*. MAGE-MODE env variable

Commands:

- `bin/magento deploy:mode:show`
- `magento deploy:mode:set {mode} [-s|--skip-compilation]` – skip compilation when changing to production

cannot switch to default mode, only developer or production

Default:

- errors logged in var/report, not displayed
- static created dynamically - copied! changes not visible. cached

Developer:

- exceptions displayed, not logged
- exception thrown if bad event subscriber.
- var/report detailed
- static created dynamically - symlinked???, changes visible immediately
- error handler - throws exception instead of logging (notice etc.)

Production - max speed, no errors, no file generation:

- admin can't enable/disable cache types
- errors logged, not displayed
- static not created dynamically, must be deployed
- not need for www-data to write, pub/static can be read-only

How to add CLI commands (off-topic)

Magento 2 CLI is based on the Symfony Console component.

To create new CLI command you need:

- Create command class (the recommended location is {module}/Console/Command) This class must extends from \Symfony\Component\Console\Command\Command and have 2 methods:

`configure` - to set the name, description, command line arguments etc

`execute` - is the place where you write your code

```
<?php
namespace Vendor\Module\Console\Command;

use Symfony\Component\Console\Input\InputArgument;
use Symfony\Component\Console\Input\InputOption;

class ExampleCommand extends \Symfony\Component\Console\Command\Command
{
    protected function configure()
    {
        $this->setName('example:hello')
            ->setDescription('Hello world command');

        // Positional argument
        $this->addArgument(
            'myargument',
            InputArgument::REQUIRED,
            'Positional required argument example'
        );

        // Not required option
        $this->addOption(
            'myoption',
            null,
            InputOption::VALUE_OPTIONAL,
            'Option example',
            ScopeConfigInterface::SCOPE_TYPE_DEFAULT
        );

        parent::configure();
    }

    protected function execute(\Symfony\Component\Console\Input\InputInterface
$input, \Symfony\Component\Console\Output\OutputInterface $output)
    {
        $output->writeln('hello world');
    }
}
```

- Declare your command in \Magento\Framework\Console\CommandListInterface using dependency injection ({module}/etc/di.xml). See also CommandListInterface implementation: \Magento\Framework\Console\CommandList

{module}/etc/di.xml:

```
<type name="Magento\Framework\Console\CommandListInterface">
    <arguments>
        <argument name="commands" xsi:type="array">
            <item name="exampleHello"
xsi:type="object">Vendor\Module\Console\Command\ExampleCommand</item>
        </argument>
    </arguments>
</type>
```

- Clean the cache and compiled code directories

rm -rf cache/* page_cache/* di/* generation/*

Links

- Magento DevDocs - How to add CLI commands
- Magento DevDocs - Command naming guidelines
- Symfony Documentation - The Console Component
- Magento - sample-module-command