SensioLabs

Symfony

Hacking & Extending Symfony2
SF2C3

The Console Component

The Console Component

The Console component eases the creation of beautiful and testable command line interfaces.

Redondant and tedious tasks

CRON jobs and batch processing

Code generation

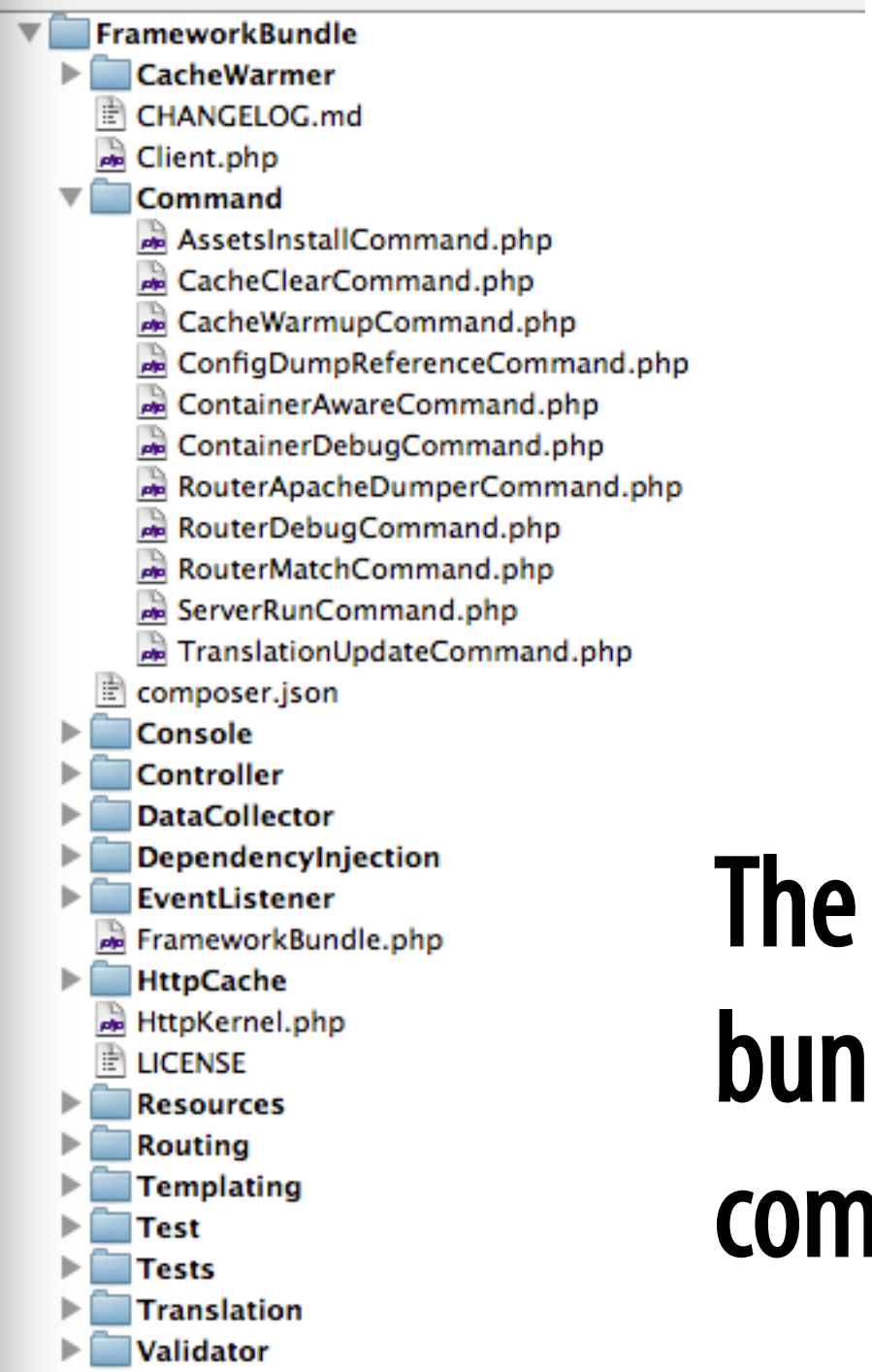
Interactive setup tools

Cache clearing / generation

…

Improve your productivity and effiency.

Bootstrapping a new Command



The Command folder of a bundle stores your command classes

What's a command?

A command is equivalent to a controller except that it converts an Input into an Output.

Creating a new Command

```php
namespace SensioLabs\Bundle\HangmanBundle\Command;

use Symfony\Component\Console\Command\Command;

class HangmanPlayCommand extends Command
{
    protected function configure()
    {
        // configure the command...
    }
}
```

```
class HangmanPlayCommand extends Command
{
    protected function configure()
    {
        $this
            ->setName('hangman:play')
            ->setDescription('Play ...')
            ->setHelp('Manual ...')
            ->setAliases(array('play-hangman'))
            ->addArgument(...)
            ->addOption(...)
        ;
    }
}
```

```
protected function configure()
{
    $this->setHelp(<<<EOF
The <info>game:hangman</info> command starts a new game of the
famous hangman game:

<info>game:hangman 8</info>

Try to guess the hidden <comment>word</comment> whose length is
<comment>8</comment> before you reach the maximum number of
<comment>attempts</comment>.

You can also configure the maximum number of attempts
with the <info>--max-attempts</info> option:

<info>game:hangman 8 --max-attempts=5</info>
EOF);
}
```

Adding required arguments

```php
// Mandatory arguments
$this->addArgument('length', InputArgument::REQUIRED);
$this->addArgument('length', InputArgument::REQUIRED, 'The length');

// Optional arguments
$this->addArgument('length', InputArgument::OPTIONAL, '', 8);


// Arguments with multiple values
$this->addArgument('ids', InputArgument::IS_ARRAY);
```

Adding options

```
// Option with a mandatory value
$this->addOption('max', 'm', InputOption::VALUE_REQUIRED);

// Option with an optional value (and a default one)
$this->addOption('max', 'm', InputOption::VALUE_OPTIONAL);
$this->addOption('max', 'm', InputOption::VALUE_OPTIONAL, '', 10);

// Option with an empty value
$this->addOption('max', 'm', InputOption::VALUE_NONE);

// Option with several values
$this->addOption('max', 'm', InputOption::VALUE_IS_ARRAY);
```

Adding the command business logic

```
protected function execute(
    InputInterface $input,
    OutputInterface $output
)
{
    // the business logic here...
}
```

The InputInterface interface definition

```php
namespace Symfony\Component\Console\Input;

interface InputInterface
{
    function getFirstArgument();
    function hasParameterOption($values);
    function getParameterOption($values, $default = false);
    function bind(InputDefinition $definition);
    function validate();
    function isInteractive();

    function getArguments();
    function getArgument($name);
    function getOptions();
    function getOption($name);
}
```

The OutputInterface interface definition

```
namespace Symfony\Component\Console\Input;

interface OutputInterface
{
    function write($messages, $newline, $type);
    function writeln($messages, $type = 0);

    function setVerbosity($level);
    function getVerbosity();
    function getDecorated($decorated);
    function isDecorated();
    function setFormatter($formatter);
    function getFormatted();
}
```

Reading the input data

```
// Read the input
$length = $input->getArgument('length');
$max    = $input->getOption('max');


// Write the output
$output->writeln('The length is '.$length);
$output->writeln('The max number of attempts is '.$max);
```

Validating Input Arguments

Validating the input arguments

```
if ($input->getArgument('length') < 5) {
    throw new \InvalidArgumentException('The length
option must be greater than 5.');
}
```

```
if ($input->getOption('max') < 1) {
    throw new \InvalidArgumentException('The max option
must be greater than 1.');
}
```

Formating the Output

The formatter helper class

The FormatterHelper class provides methods to colorize an output message.

```
class FormatterHelper extends Helper
{
    function formatSection($section, $message, $style);
    function formatBlock($messages, $style, $large);
}
```

The formatter helper class

```
$formatter->formatBlock('A green information', 'info');
$formatter->formatBlock('A yellow comment', 'comment');
$formatter->formatBlock('A red error', 'error');
$formatter->formatBlock('A custom style', 'bg=blue;fg=white');
```

```
Hugo-3:SF2C1 hugo.hamon$ php app/console game:hangman 8
 A green information
 A yellow comment
 A red error
 A custom style
Hugo-3:SF2C1 hugo.hamon$
```

The formatter helper class

```
// Get the formatter helper
$f = $this->getHelperSet()->get('formatter');

$output->writeln(
    $f->formatBlock('Welcome...', 'bg=blue;fg=white', true)
);

$output->writeln(
    $f->formatSection('Info', 'You...', 'info', true)
);
```

The formatter helper class

```
Hugo-3:SF2C1 hugo.hamon$ php app/console game:hangman 8

  Welcome in the Hangman Game

[Info] You have 10 attempts to guess the hidden word.

Hugo-3:SF2C1 hugo.hamon$
```

Interact with the end user

The Dialog helper class

```
class DialogHelper extends Helper
{
    public function ask(...);

    public function askConfirmation(...);

    public function askHiddenResponse(...);

    public function askAndValidate(...);
}
```

Getting the dialog helper

```
$dialog = $this
    ->getHelperSet()
    ->get('dialog')
;
```

Asking a question from the CLI

```
$answer = $dialog->ask(
    $output,
    'Do you like Symfony?'
);

$answer = $dialog->askHiddenResponse(
    $output,
    'Type your password... '
);
```

Asking a question and validating the answer

```
$answer = $dialog->askAndValidate(
    $output,
    'Type a letter... ',
    function ($letter) {
        if (!preg_match('/^[a-z]$/i', $letter)) {
            throw new \Exception('...');
        }

        return $letter;
    }
);
```

Asking a question and validating the answer

```
Hugo-3:SF2C1 hugo.hamon$ php app/console game:hangman 8

  Welcome in the Hangman Game

[Info] You have 10 attempts to guess the hidden word.

Type a letter...A
Type a letter...B
Type a letter...C
Type a letter...%
 The expected letter must be a single character between A and Z.
Type a letter...D
Type a letter...
```

Access the Service Container

The ContainerAwareCommand class

The ContainerAwareCommand class allows to have access to the services container from a command.

Symfony automatically injects the service container into any command that extends this class.

The ContainerAwareCommand class

```php
namespace Symfony\Bundle\FrameworkBundle\Command;

use Symfony\Component\Console\Command\Command;
use Symfony\Component\DependencyInjection\ContainerInterface;
use Symfony\Component\DependencyInjection\ContainerAwareInterface;

abstract class ContainerAwareCommand extends Command implements ContainerAwareInterface
{
    private $container;

    protected function getContainer()
    {
        if (null === $this->container) {
            $this->container = $this->getApplication()->getKernel()->getContainer();
        }

        return $this->container;
    }

    public function setContainer(ContainerInterface $container = null)
    {
        $this->container = $container;
    }
}
```

Accessing the service container

```
$container = $this->getContainer();
```

Reading configuration parameters

```
$salt = $container->getParameter('secret');
```

Accessing the doctrine registry service

```
$container = $this->getContainer();

$doctrine = $container->get('doctrine');
$em = $doctrine->getManager();

$score = new Score();
$score->setScore(10);
$score->setPlayer('john.doe');

$em->persist($score);
$em->flush();
```

Accessing the templating service

```
$container = $this->getContainer();

$templating = $container->get('templating'):

$content = $templating->render(
    'SensioHangmanBundle:Game:finish.txt.twig',
    array('game' => $this->game)
);
```

Accessing the router service

```
$container = $this->getContainer();

$router = $container->get('router');

$url = $router->generate(
    'game_finish',
    array('user' => 'smith'),
    true
);
```

Translating messages

```
$container = $this->getContainer();

$translator = $container->get('translator'):

$content = $translator->trans(
    'Hello %user%!',
    array('user' => 'hhamon'),
    null,
    'fr'
);
```

Recording log messages

```
$container = $this->getContainer();

$logger = $container->get('logger');

$logger->info('Game finished!');
```

Dealing with the filesystem

```php
$container = $this->getContainer();

$fs = $container->get('filesystem');

$fs->touch('/path/to/toto.txt');
```

Testing Commands

Testing commands

Symfony comes with a CommandTester class, which allows you to execute a command and get the Input and Output objects back.

Then, with PHPUnit, it's just a matter of introspecting these objects and all the changes made by the command execution.

Basic Example

```php
namespace Sensio\Bundle\DemoBundle\Command;

class HelloWorldCommand extends Command
{
    // ...

    protected function execute($input, $output)
    {
        $name = $input->getOption('name');

        $output->writeln('Your name is <info>'. $name .'</info>');
    }
}
```

The StreamOutput class

The StreamOutput class write the command output to a stream. For example, a stream can also be a file stream.

Some examples

```
$output = new StreamOutput(fopen('php://stdout', 'w'));
$output = $output->getStream();

$output = new StreamOutput(fopen('output.log', 'a'));
$output = $output->getStream();
```

```php
class SayHelloCommandTest extends \PHPUnit_Framework_TestCase
{
    public function testSayHello()
    {
        $input = new ArrayInput(array('name' => 'Hugo'));
        $input->setInteractive(false);

        $output = new StreamOutput(fopen('output.log', 'a'));

        $command = new SayHelloCommand();
        $command->run($input, $output);

        $this->assertEquals(
            'Your name is <info>Hugo</info>',
            $output->getStream()
        );
    }
}
```

```
Hugo:Demo Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.5.15 by Sebastian Bergmann.

...

Time: 0 seconds, Memory: 19.75Mb

OK (3 tests, 3 assertions)
Hugo:Demo Hugo$
```

```php
namespace Symfony\Component\Console\Tester;

class CommandTester
{
    function __construct(Command $command);

    function execute($input, $options);

    function getDisplay();

    function getInput();

    function getOutput();
}
```

```php
class SayHelloCommandTest extends \PHPUnit_Framework_TestCase
{
    public function testSayHello()
    {
        $cmd = new SayHelloCommand();
        $tester = new CommandTester($cmd);

        $tester->execute(array('name' => 'Hugo'), array(
            'interactive' => false
        ));

        $this->assertEquals(
            'Your name is <info>Hugo</info>',
            $tester->getDisplay()
        );
    }
}
```

```
Hugo:Demo Hugo$ phpunit -c app/phpunit.xml.dist
PHPUnit 3.5.15 by Sebastian Bergmann.

...

Time: 0 seconds, Memory: 19.75Mb

OK (3 tests, 3 assertions)
Hugo:Demo Hugo$
```

Training Department

SensioLabs Training

92-98 Boulevard Victor Hugo

92 115 Clichy Cedex

FRANCE

Phone: +33 140 998 211

Email: training@sensiolabs.com

symfony.com - trainings.sensiolabs.com