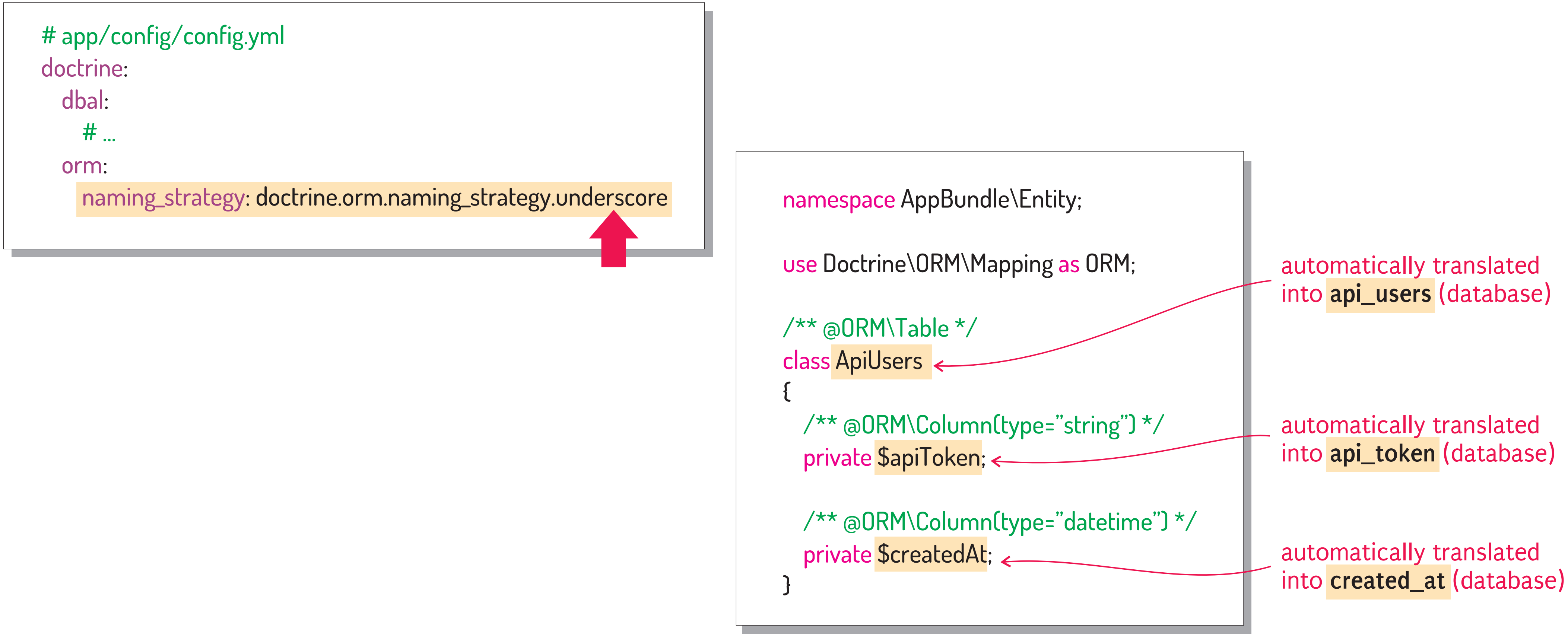
Symfony

Tips & Tricks

2016
calendar

Symfony

Setting custom table/properties names using built-in Doctrine naming strategy



```yaml
# app/config/config.yml
doctrine:
   dbal:
      # ...
   orm:
      naming_strategy: doctrine.orm.naming_strategy.underscore
```

```php
namespace AppBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/** @ORM\Table */
class ApiUsers
{
   /** @ORM\Column(type="string") */
   private $apiToken;

   /** @ORM\Column(type="datetime") */
   private $createdAt;
}
```

automatically translated into **api_users** (database)

automatically translated into **api_token** (database)

automatically translated into **created_at** (database)

http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/reference/namingstrategy.html

January

sun	mon	tue	wed	thu	fri	sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24/31	25	26	27	28	29	30

Form

Symfony

Best Practices (controller related)

Use “action” and “method” options in the controller

```
$form = $this->createForm(PaymentType::class, $order, [
      ‘action’  => $this->generateUrl(‘payment_gw’),
      ‘method’ => ‘PUT’,
]);
```

Use custom options to pass dynamic data to constructor

```
$defaultShipping = $this->getDefaultShipping($customer);

~~$form = $this->createForm(~~
~~   new PlaceOrderType($defaultShipping),~~
~~);~~
```

Don’t pass dynamic data to constructor

```
$form = $this->createForm(PlaceOrderType::class, ..., [
    ‘default_shipping’ => $defaultShipping,
]);
```

Use custom options

Pass global settings to constructor

```
parameters:
    app.us_shipping.enabled: true

services:
    app.form.shipping:
        class: AppBundle\Form\ShippingType
        arguments:
            - %app.us_shipping.enabled%
        tags:
            - { type: form.type }
```

```
class ShippingType extends AbstractType
{
    private $usShipping;

    public function __construct($usShipping)
    {
        $this->usShipping = (bool) $usShipping;
    }

    // ...
}
```

February

sun	mon	tue	wed	thu	fri	sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29					

Improved WHERE... IN

Before Doctrine 2.5

transform the ArrayCollection into an array of Ids

```
$categories = ...

$categoryIds = [];
foreach ($categories as $category) {
   $categoryIds[] = $category->getId();
}

$queryBuilder = $this
   ->where('model.category IN (:category_ids)')
   ->setParameter('category_ids', $categoryIds)
;
```

Doctrine 2.5+

WHERE... IN supports the use of ArrayCollection

```
$categories = ...

$queryBuilder = $this
   ->where('model.category IN (:categories)')
   ->setParameter('categories', $categories)
;
```

http://doctrine-orm.readthedocs.org/projects/doctrine-orm/en/latest/changelog/migration_2_5.html#query-api-where-in-query-using-a-collection-as-parameter

March

sun	mon	tue	wed	thu	fri	sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

Variadic Filters
(for unlimited arguments)

Defining a variadic filter

```
$filter = new Twig_SimpleFilter('image', function (
    $path, $options = []
) {

    // ...

}, ['is_variadic' => true]);
```

a single variadic parameter holds any
number of passed parameters (unlimited)

Getting the values passed

```
$filter = new Twig_SimpleFilter('image', function (
    $path, $options = []
) {
    $path     = ...
    $width   = $options['width'];
    $height  = $options['height '];
    $opacity = $options['opacity '];
}, ['is_variadic' => true]);
```

https://github.com/twigphp/twig/pull/1699

April

sun	mon	tue	wed	thu	fri	sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

Symfony

Dealing with Sensitive Data: Alternative to Environment Variables

1. Create a “credentials” file in the prod server

```
# /etc/credentials/example.com.yml
parameters:
    database_user:          ...
    database_password:      ...

    mailer_user:            ...
    mailer_password:        ...

    secret:                 ...

    github_client_id:       ...
    github_client_secret:   ...
```

2. Load the credentials file in the application

```
# app/config/config.yml
imports:
    - { resource: parameters.yml }
    - { resource: services.yml }
    - { resource: security.yml }
    - { resource: admin.yml }
    - { resource: '/etc/credentials/example.com.yml',
                  ignore_errors: true }
# ...
```

3. In “dev” machine

- Credentials file doesn’t exist but no error is triggered (because of ignore_errors)
- App uses the regular parameters.yml file

4. In “prod” machine

- Credentials file overrides parameters.yml
- The right parameters are available anywhere (e.g. console commands)
- Developers can’t see the production configuration options
- Requires discipline to add/remove options

https://github.com/symfony/symfony/pull/16403

May

sun	mon	tue	wed	thu	fri	sat
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Symfony

Success and Failure Handlers

Success/failure handlers are available since Symfony 2.0 (2011) but lots of developers don’t use them because they are not documented

The security.interative_login event

```
services:
  login_listener:
    class: AppBundle\Listener\LoginListener
    arguments: ['@secutiry.token_storage', '@doctrine']
    tags:
      - { name: 'kernel.event_listener', event: 'security.interactive_login' }
```

the most common event to “do things” after the user successfully logs in

Defining a success handler

```
namespace AppBundle\Security;

use Symfony\Component\Security\Http\Authentication\AuthenticationSuccessHandlerInterface;

class LoginHandler implements AuthenticationSuccessHandlerInterface
{
  public function onAuthenticationSuccess(Request $request, TokenInterface $token)
  {
    // do something ...
    return $this->redirect('...');
    return new Response('...');
  }
}
```

you just need to implement this interface...

...and return a Response instance

Enabling the success handler

```
# app/config/services.yml
services:
  app.login_handler:
    class: AppBundle\Security\LoginHandler
    arguments: ...

# app/config/security.yml
firewalls:
  main:
    pattern: ^/
    form_login:
      success_handler: app.login_handler
```

the LoginHandler class is executed when the user logs in successfully and after the event.interactive_login

https://github.com/symfony/symfony-docs/issues/4258

June

sun	mon	tue	wed	thu	fri	sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30		

Symfony

Monolog

Creating Simple Custom Formatters

Creating custom formatters

```
# app/config/config.yml
services:
  app.log.formatter:
    class: ‘Monolog\Formatter\LineFormatter’
    public: false
    arguments:
      - “%%level_name%% [%%datetime%%] [%%channel%%] %%message%%\n
        %%context%%\n\n”
      - ‘H:i:s’
      - false
```

the format of each log line

the format of %datetime% placeholder

ignore \n inside the log messages?

Enable the custom log formatter

```
# app/config/config_dev.yml
monolog:
  handlers:
    main:
      type: stream
      path: “%kernel.logs_dir%/%kernel.environment%.log”
      level: debug
      formatter: app.log.formatter
```

add the formatter option

http://symfony.com/doc/current/cookbook/logging/monolog.html#changing-the-formatter

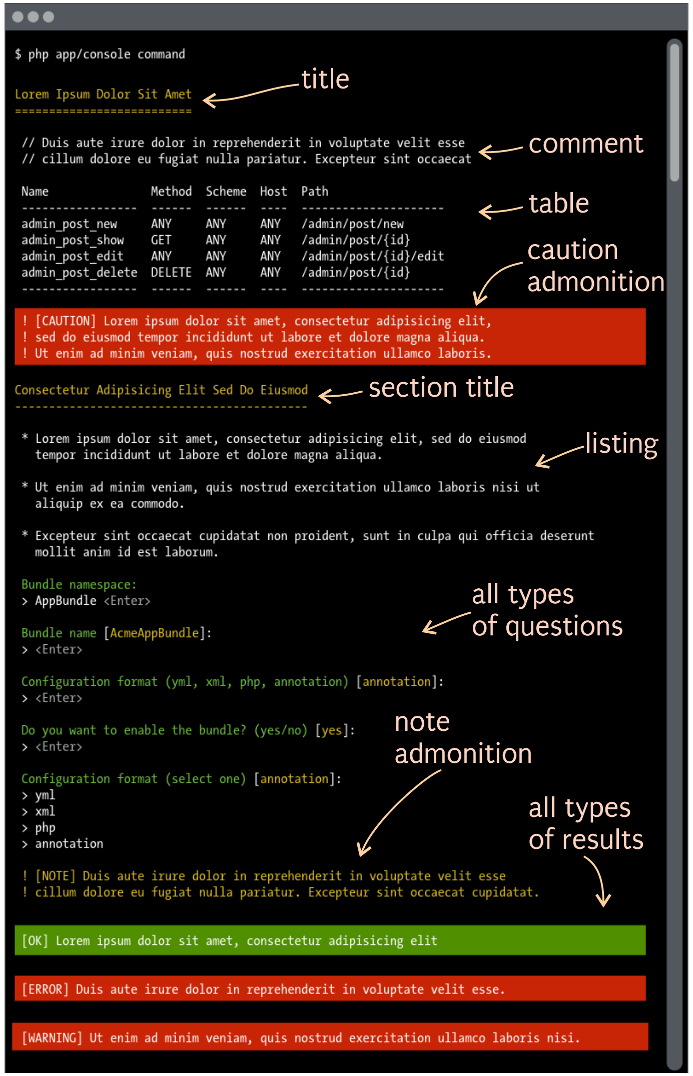
July

sun	mon	tue	wed	thu	fri	sat
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24/31	25	26	27	28	29	30

Console

Symfony

Console Style Guide: Create Visually Consistent Commands Effortlessly



```
$ php app/console command

Lorem Ipsum Dolor Sit Amet
==========================

 // Duis aute irure dolor in reprehenderit in voluptate velit esse
 // cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat

 Name              Method  Scheme  Host  Path
 ----------------- ------- ------- ----- --------------------
 admin_post_new    ANY     ANY     ANY   /admin/post/new
 admin_post_show   GET     ANY     ANY   /admin/post/{id}
 admin_post_edit   ANY     ANY     ANY   /admin/post/{id}/edit
 admin_post_delete DELETE  ANY     ANY   /admin/post/{id}
 ----------------- ------- ------- ----- --------------------

! [CAUTION] Lorem ipsum dolor sit amet, consectetur adipisicing elit,
! sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
! Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris.

Consectetur Adipisicing Elit Sed Do Eiusmod
-------------------------------------------

 * Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod
   tempor incididunt ut labore et dolore magna aliqua.

 * Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut
   aliquip ex ea commodo.

 * Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt
   mollit anim id est laborum.

 Bundle namespace:
 > AppBundle <Enter>

 Bundle name [AcmeAppBundle]:
 > <Enter>

 Configuration format (yml, xml, php, annotation) [annotation]:
 > <Enter>

 Do you want to enable the bundle? (yes/no) [yes]:
 > <Enter>

 Configuration format (select one) [annotation]:
 > yml
 > xml
 > php
 > annotation

 ! [NOTE] Duis aute irure dolor in reprehenderit in voluptate velit esse
 ! cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat.

 [OK] Lorem ipsum dolor sit amet, consectetur adipisicing elit

 [ERROR] Duis aute irure dolor in reprehenderit in voluptate velit esse.

 [WARNING] Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi.
```

All common features are covered

```php
$io = new SymfonyStyle($input, $output);

// common output elements
$io->title(string $message);
$io->section(string $message);
$io->text(string|array $message);
$io->comment(string|array $message);

// more advanced output elements
$io->note(string|array $message);
$io->caution(string|array $message);
$io->listing(array $elements);
$io->table(array $headers, array $rows);

// ask for user's input
$io->ask(string $question, string|null $default = null, callable|null $validator = null);
$io->askHidden(string $question, callable|null $validator = null);
$io->confirm(string $question, bool $default = true);
$io->choice(string $question, array $choices, string|int|null $default = null);

// display the result of the command or some important task
$io->success(string|array $message);
$io->error(string|array $message);
$io->warning(string|array $message);
```

You can use it in your own commands too

```php
use Symfony\Component\Console\Style\SymfonyStyle;

class MyCustomCommand extends ContainerAwareCommand
{
   // ...

   protected function execute(InputInterface $input, OutputInterface $output)
   {
      $io = new SymfonyStyle($input, $output);
      $io->...
      // ...
   }
}
```

August

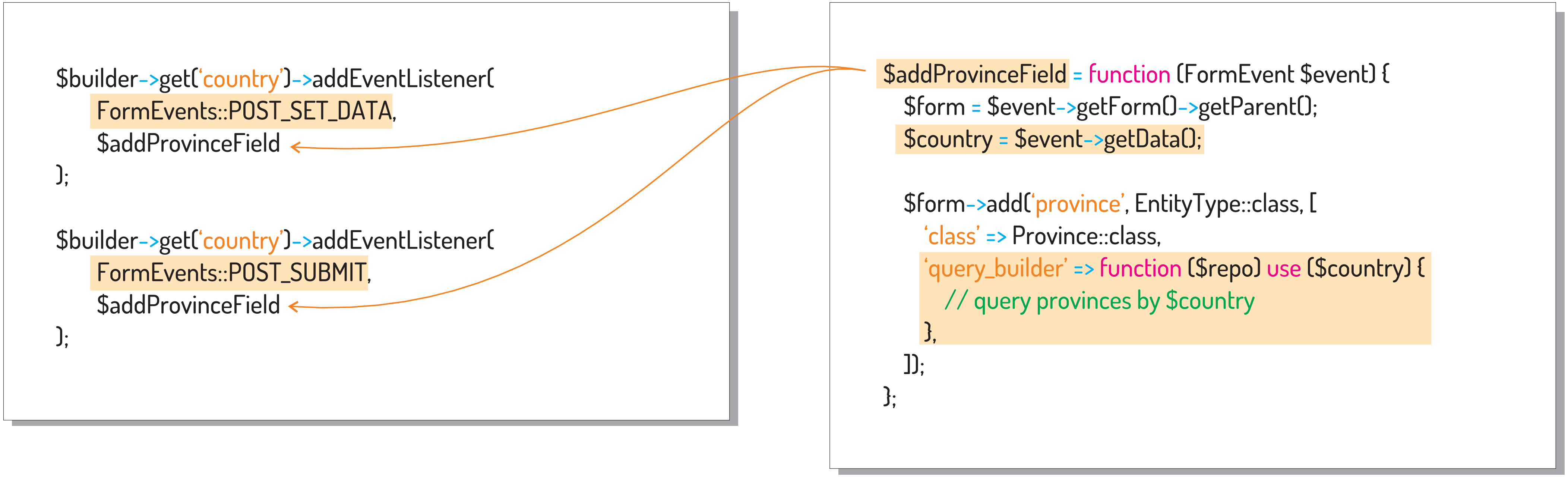
sun	mon	tue	wed	thu	fri	sat
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Form

Field Dependencies

```
$builder
    ->add('country', 'entity', [ ... ])
    ->add('province', 'entity', [ ... ])
;
```

Use POST_* hooks on fields



```
$builder->get('country')->addEventListener(
    FormEvents::POST_SET_DATA,
    $addProvinceField
);

$builder->get('country')->addEventListener(
    FormEvents::POST_SUBMIT,
    $addProvinceField
);
```

```
$addProvinceField = function (FormEvent $event) {
  $form = $event->getForm()->getParent();
  $country = $event->getData();

  $form->add('province', EntityType::class, [
    'class' => Province::class,
    'query_builder' => function ($repo) use ($country) {
      // query provinces by $country
    },
  ]);
};
```

September

sun	mon	tue	wed	thu	fri	sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

Improve Class Loading Performance

New “exclude-from-classmap” option

These classes are excluded from the optimized autoloader

```
// composer.json
{
  "autoload": {
    "exclude-from-classmap": ["/Tests/", "/test/", "/tests/"]
  }
}
```

```
$ composer dump-autoload --optimize
```

```
// composer.json
{
  "autoload": {
    "exclude-from-classmap": ["**/Tests/", "/test/*”]
  }
}
```

** matches anything

* matches anything except /

https://github.com/symfony/symfony/pull/16397

October

sun	mon	tue	wed	thu	fri	sat
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18 Symfony first commit (11 years ago)	19	20	21	22
23/30	24/31	25	26	27	28	29

Symfony

Tests

Meaningful Data Providers

array keys are the labels of each data set

```
public function getUserData()
{
    return [
      'register user only'              => [true, false, false],
      'register and enable'             => [true, true, false],
      'register, enable and notify' => [true, true, true],
    ];
}

/** @dataProvider getUserData */
public function testRegistration($register, $enable, $notify)
{
    // ...
}
```

When an error happens...

```
$ phpunit -c app
F..
Time: 137 ms, Memory: 12.50Mb

There was 1 failure:
1) AppBundle\Tests\Controller\DefaultControllerTest::testRegistration
   with data set "register user only" (true, false, false)
```

meaninful error messages

http://www.thisprogrammingthing.com/2015/making-dataproviders-more-maintainable/

November

sun	mon	tue	wed	thu	fri	sat
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

Controller

Symfony

Acessing Request Parameters:
Useful Methods Defined in the ParameterBag

Acessing request parameters

```
use Sensio\Bundle\FrameworkExtraBundle\Configuration\Route;
use Symfony\Bundle\FrameworkBundle\Controller\Controller;
use Symfony\Component\HttpFoundation\Request;

/** @Route(”/blog”) */
class BlogController extends Controller
{
  /** @Route(”/”, name=”blog_index”) */
  public function indexAction()
  {
    $value1 = $request->query->get(’parameter1’);
    $value2 = $request->request->get(’parameter2’);

    // ...
  }
}
```

everybody uses
the get() method,
but there
are other
useful methods

ParameterBag defines some useful methods

```
$pageNumber = $request->query->getInt(’page’);
$isPublished = $request->query->getBoolean(’published’);

// ‘ABC+12.34DEF’ -> ‘ABCDEF’
$request->query->getAlpha(’parameter’);

// ‘ABC+12.34DEF’ -> ‘ABC1234DEF’
$request->query->getAlnum(’parameter’);

// ‘ABC+12.34DEF’ -> ‘1234’
$request->query->getDigits(’parameter’);
```

http://symfony.com/doc/current/components/http_foundation/introduction.html

December

sun	mon	tue	wed	thu	fri	sat
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

Tips & Tricks from SymfonyCon 2015 (10 years of Symfony):

http://www.slideshare.net/javier.eguiluz/new-symfony-tips-tricks-symfonycon-paris-2015 (Thanks @javiereguiluz)

https://speakerdeck.com/webmozart/symfony2-forms-dos-and-donts (Thanks @webmozart)

http://symfony.com/blog/new-in-symfony-2-8-console-style-guide

made with 💚 by @andreiabohner