Cheatography

Symfony 4 Cheat Sheet

by Marcel Berteler (pluk77) via cheatography.com/55947/cs/14851/

Entity Manager in a controller

$this->getDoctrine()->getManager();	Get default entity manager
$this->getDoctrine()->getManagerForClass(MyEntity::class)	Get entity specific manager

Magic repository functions

find($id)	Returns one entity
findOneBy($criteria, $orderBy)	Returns one entity
findBy($criteria, $orderBy, $limit, $offset)	Returns collection
$criteria = ['field' => 'value']	Single field / value
$criteria = ['field' => ['value1', 'value2']]	Single where-in
$criteria = ['field1' => 'value', 'field2' => 'value2']	multiple fields / values

```
$repository = $this->getDoctrine()->getRepository(Product::class);
$product = $repository->findOneBy(['name' => 'Keyboard']);
```

Doctrine criteria - filtering collections

where($Expression)	Replaces previous statement
[and/or]**Where**($Expression)	Adds AND /OR where statement
orderBy($array)	Sets OrderBy
setFirstResult($firstResult)	Sets first result
setMaxResult($max)	Sets Maximum results

```
$userCollection = $group->getUsers();

$criteria = Criteria::create()
->where(Criteria::expr()->eq("birthday", "1982-02-17"))
->orderBy(array("username" => Criteria::ASC))
->setFirstResult(0)
->setMaxResults(20)
;
$birthdayUsers = $userCollection->matching($criteria);
```

Repositories in a controller

```
$this->getDoctrine()->getRepository(MyEntity::class);
```

Doctrine ExpressionBuilder and Expr

andX($arg1, $arg2, ...)	Multiple arguments AND
orX($arg1, $arg2, ...)	Multiple arguments OR
[n]**eq**($field, $value)	[Not] Equal
gt[e]($field, $value)	Greater than [or equal]
lt[e]($field, $value)	Less than [or equal]
isNull($field)	Is Null
[not]**in**($field, array $values)	[not] In
memberOf ($field, $value)	**ExpressionBuilder only** Member of
isMemberOf($field, $value)	**Expr only** Member of
isNotNull($field)	**Expr only** Is not Null
between($field, $x, $y)	**Expr only** Between $x and $y
trim($field)	**Expr only** Trim
concat($x, $y)	**Expr only** Concat
literal ($string)	**Expr only** Literal
lower($field)	**Expr only** Lower case
upper($field)	**Expr only** Upper case
count($field)	**Expr only** count

```
# Doctrine Expr #
Criteria::expr()->orX(
Criteria::expr()->eq('id', $facilityId),
Criteria::expr()->eq('active', TRUE)
);

# Doctrine ExpressionBuilder #
$qb = $this->createQueryBuilder('f');
$qb->where(
$qb->expr()->eq('id', '?arg1')
)->setParameter('arg1', $facilityId);
```

By **Marcel Berteler** (pluk77)
cheatography.com/pluk77/

Not published yet.
Last updated 10th July, 2018.

Cheatography

Symfony 4 Cheat Sheet

by Marcel Berteler (pluk77) via cheatography.com/55947/cs/14851/

Doctrine Query Builder

setParameter($parameter, $value)

addCriteria(Criteria $criteria)

[get/set]**MaxResults**($maxResults)

[get/set]**FirstResult** ($firstResult)

getQuery()

add($dqlPartName, $dqlPart, $append = false)

$dqlPartName: select, from, join, set, where, groupBy, having, orderBy

Wrappers for add():

[add]**select**($select= null)

delete($delete = null, $alias = null)

update($update = null, $alias = null)

set($key, $value)

from($from, $alias, $indexBy = null)

[inner/left]**join**($join, $alias, $conditionType = null, $condition = null, $indexBy = null)

[and/or]**where**($where)

[add]**groupBy**($groupBy)

[and/or]**having**($having)

[add]**orderBy**($field, $order = null)

Doctrine Query

getResult($hydrationMode = self::HYDRATE_OBJECT)	Retrieve a collection
getSingleResult($hydrationMode = null)	Retrieve a single result or exception
getOneOrNullResult($hydrationMode = null)	Retrieve a result or NULL
getArrayResult ()	Retrieve an array
getScalarResult()	Retrieves a flat/rectangular result set of scalar values

Doctrine Query (cont)

getSingleScalarResult()	Retrieves a single scalar value or exception
AbstractQuery::HYDRATE_OBJECT	Object graph **default**
AbstractQuery::HYDRATE_ARRAY	Array graph
AbstractQuery::HYDRATE_SCALAR	Flat rectangular result with scalars
AbstractQuery::HYDRATE_SINGLE_SCALAR	Single scalar value
AbstractQuery::HYDRATE_SIMPLEOBJECT	Very simple object **fast**
setCacheMode($cacheMode)	
Cache::MODE_GET	may read, not write
Cache::MODE_PUT	no read, write all
Cache::MODE_NORMAL	read and write
Cache::MODE_REFRESH	no read, only refresh
setCacheable($trueFalse)	Enable/disable result caching

Persisting Entities

```
// get the entity manager that manages this entity
$em = $this->getDoctrine()->
getManagerForClass(MyEntity::class);
// create a new entity
$entity = new MyEntity();
// populate / alter properties
$entity->setName('Default Entity');
// tell Doctrine you want to (eventually) save
$em->persist($entity);
// actually executes the queries
$em->flush();
```

Authorization

IS_AUTHENTICATED_FULLY	User has successfully authenticated, not via 'remember me cookie'
IS_AUTHENTICATED_REMEMBERED	All logged in users
IS_AUTHENTICATED_ANONYMOUSLY	All users, even anonymous ones

All roles you assign to a user must begin with the ROLE_ prefix.

Authorization via security.yaml

```
# config/packages/security.yaml
security:
    # ...
    firewalls:
        # ...
        main:
            # ...
    access_control:
        # require ROLE_SUPER_ADMIN for /admin/users*
        - { path: ^/admin/users, roles:
ROLE_SUPER_ADMIN }
        # require ROLE_ADMIN for /admin*
        - { path: ^/admin, roles: ROLE_ADMIN }
```

No limit on amount of URL patterns. Each is a regular expression. First match will be used.

Prepend the path with ^ to ensure only URLs beginning with the pattern are matched.

Authorization in a controller

```
# Example of using wrapper #
public function hello($name)
{
    // The second parameter is used to specify on what
object the role is tested.
    $this->denyAccessUnlessGranted('ROLE_ADMIN', null,
'Unable to access this page!');
    // ...
}
# Example of using AuthorizationChecker
use
Symfony\Component\Security\Core\Authorization\Authori
zationCheckerInterface
use
Symfony\Component\Security\Core\Exception\AccessDenie
dException;
public function hello($name,
AuthorizationCheckerInterface $authChecker)
{
    if (false === $authChecker-
>isGranted('ROLE_ADMIN')) {
        throw new AccessDeniedException('Unable to
access this page!');
    }
    // ...
}
# Example of using annotation #
use
Sensio\Bundle\FrameworkExtraBundle\Configuration\Secur
ity;
/**
 * @Security("has_role('ROLE_ADMIN')")
 */
public function hello($name)
{
    // ...
}
```

Authorization in template

```
{% if is_granted('ROLE_ADMIN') %}
    <a href="...">Delete</a>
{% endif %}
```

-

Create a new form in a controller

```
# Create a new form with a default name #
$form = $this->createFormBuilder($data)
  ->add('dueDate', null, array(
     'widget' => 'single_text'))
  ->add('save', SubmitType::class)
  ->getForm();
# Create a form with a non-default name #
$form = $this->container->get('form.factory')
  ->createNamedBuilder(
     'form1', FormType::class, $data)
  ->add('dueDate', null, array(
     'widget' => 'single_text'))
  ->add('save', SubmitType::class)
  ->getForm();
```

Create a form from a class in a controller

```
# Create a form from a form class with a default name
#
$form = $this->createForm(TaskForm::class, $data,
array(
  'action' => $this->generateUrl('target_route'),
  'method' => 'GET',
  ));
# Create a form from a form class with a non-default
name #
$form = $this->container->get('form.factory')
  ->createNamed('form1', TaskForm::class, $data,
array(
     'action' => $this->generateUrl('target_route'),
     'method' => 'GET',
     ));
```

Basic form

```
class TaskForm extends AbstractType
{
  public function buildForm(
         FormBuilderInterface $builder,
         array $options)
  {
    $builder
      ->add('dueDate', DateType::class, array(
        'widget' => 'single_text',
        'label' => 'Due Date',
        'required' => false,
        'attr' => array('maxlength' => 10),
        'constraints' => array(new Length(
            array('max' => 10)))
        ))
      ->add('save', SubmitType::class);
  }
  public function configureOptions(
     OptionsResolver $resolver)
  {
    $resolver->setDefaults(array(
      'method' => 'GET',
    ));
  }
}
```

Cheatography

Symfony 4 Cheat Sheet

by Marcel Berteler (pluk77) via cheatography.com/55947/cs/14851/

Form Options

'**action**' => ''	Where to send the form's data on submission (usually a URI)
'**allow_extra_fields**' => false	Allow additional fields to be submitted
'**error_mapping**' => array('matchingCityAndZipCode' => 'city')	Modify the target of a validation error
'**extra_fields_message**' => 'This form should not contain extra fields.'	Validation error message if additional fields are submitted
'**inherit_data**' => false	Inhered data from parent form or not
'**method**' => 'POST'	HTTP method used to submit the form
'**post_max_size_message**' => 'The uploaded file was too large.'	Validation message for size of post form data
'**validation_groups**' => false	Disable the Validation of Submitted Data

```
public function configureOptions(OptionsResolver $resolver) {
$resolver->setDefaults(array(
# Options go here #
));
}
```

TwigBridge - Forms

{{ **form**(view, variables) }}	Render whole form
{{ **form_start**(view, variables) }}	Render start tag
{{ **form_errors**(view) }}	Render global errors
{{ **form_row**(view, variables) }}	Render all the fields

TwigBridge - Forms (cont)

{{ **form_rest**(view, variables) }}	Render all other fields
{{ **form_end**(view, variables) }}	Render end tag + all other fields
{{ **form_row**(view.field) }}	Render field label, error and widget
{{ **form_label**(view.field, label, variables) }}	Render field label
{{ **form_errors**(view.field) }}	Render field error
{{ **form_widget**(view.field, variables) }}	Render field widget

```
{# render a label, but display 'Your Name' and add a "foo" class to it #}
{{ form_label(form.name, 'Your Name', {'label_attr': {'class': 'foo'}}) }}

{# render a widget, but add a "foo" class to it #}
{{ form_widget(form.name, {'attr': {'class': 'foo'}}) }}

{# render a field row, but display a label with text "foo" #}
{{ form_row(form.name, {'label': 'foo'}) }}
```

Log Levels

emergency()	System is unusable.
alert()	Action must be taken immediately.
critical()	Critical conditions.
error()	Runtime errors that do not require immediate action but should typically be logged and monitored.
warning()	Exceptional occurrences that are not errors.
notice()	Normal but significant events.
info()	Interesting events.

Symfony 4 Cheat Sheet

by Marcel Berteler (pluk77) via cheatography.com/55947/cs/14851/

Log Levels (cont)

debug()	Detailed debug information.

```
use Psr\Log\LoggerInterface;

public function index(LoggerInterface $logger) {
$logger->info('I just got the logger');
}
```

Console

bin\console	List available commands and show the Symfony version
server:run	Run the built-in web server
assets:install --symlink	Install bundle assets as a symlink or hardcopy
debug:autowire	Lists classes/interfaces you can use for autowiring
debug:config	Dumps the current configuration for an extension
debug:container	Displays current services for an application
debug:form	Lists classes/interfaces you can use for autowiring
debug:route	Displays current routes for an application

```
Usage:
php bin\console command [options] [arguments]

Options:
-h, --help Display this help message
-q, --quiet Do not output any message
-n, --no-interaction Do not ask any interactive question
-e, --env=ENV The environment name [default: "dev"]
--no-debug Switches off debug mode
-v|vv|vvv, --verbose Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug
```

Response from a controller

render($view, $parameters, $response = null)	Render the template and return a Response
json($data, $status = 200, $headers = array(), $context = array())	Encode data and return a Json response
file($file, $fileName = null, $disposition = ResponseHeaderBag::DISPOSITION_ATTACHMENT)	Return a file response
redirectToRoute($route, $parameters = array(), $status = 302)	Redirect to route
redirect($url, $status = 302)	Redirect to external URL
forward($controller, $path = array(), $query = array())	Forwards the request to another controller

```
return $this->render('admin/post/show.html.twig', [
    'post' => $post,
    'delete_form' => $deleteForm->createView(),
]);

return $this->json($data);
```