Symfony Cheat Sheet

Symfony CLI: Installation

```
curl -sS https://get.symfony.com/cli/installer | bash
```

Symfony CLI: Basic Commands

Command	Description
`symfony check:requirements`	Check system requirements for running Symfony
`symfony version`	Display the Symfony CLI version
`symfony help`	Display all available commands
`symfony help <command>`	Display help for a specific command

Creating a New Symfony Project

Command	Description
`symfony new <directory>`	Create a new Symfony project in `<directory>`
`symfony new --webapp <directory>`	Create a new web application
`symfony new --api <directory>`	Create a new API application
`composer create-project symfony/skeleton <directory>`	Create a new Symfony project using Composer

Flex Commands

Command	Description
`composer symfony:dump-env <env>`	Compile .env files to .env.local.php
`composer symfony:recipes`	List available recipes
`symfony:recipes:install`	(Re)install recipes for packages
`symfony:recipes:update`	Update installed recipes

Routing Attributes

```
use Symfony\Component\Routing\Attribute\Route;
```

Attribute	Description
`#[Route('/path')]`	Define a route for a controller method
`#[Route('/path', name: 'name')]`	Define a named route for a method
`#[Route('/path', methods: ['GET'])]`	Define allowed HTTP methods
`#[Route('/blog/{slug}')]`	Define a route with a parameter
`#[Route('/page/{page}', requirements: ['page' => '\d+'])]`	Define a route with a parameter constraint
`#[Route(path: ['en' => '/about-us', 'nl' => '/over-ons'], name: 'about')]`	Define localized routes

AbstractController Methods

Method	Description
`$this->render('template.html')`	Returns a `Response` with the rendered template
`$this->redirectToRoute('homepage')`	Returns a `RedirectResponse` to a named route
`$this->redirectToRoute('name', ['param' => 'value'])`	Returns a redirect to a route with parameters
`$this->redirectToRoute('name', [], 301)`	Returns a `RedirectResponse` with status code
`$this->redirect('http://example.com')`	Returns a `RedirectResponse` to an external URL
`$this->createNotFoundException($msg)`	Returns a `NotFoundHttpException`
`$this->createForm(FormType::class, $data)`	Creates a new Form instance based on the type

Form Methods

Method	Description
`$form->handleRequest($request)`	Handle a form submission
`$form->isSubmitted()`	Check if the form was submitted
`$form->isValid()`	Check if the form is valid

Form Fields: Text Fields

Type	Description
`TextType`	A basic `text` input field
`TextareaType`	A multi-line `textarea` field
`EmailType`	An email field (`<input type="email">`)
`PasswordType`	A password field
`SearchType`	A search field
`UrlType`	A URL field
`TelType`	A `tel` (phone number) input field
`UuidType`	A UUID field

Form Fields: Numeric Fields

Type	Description
`IntegerType`	A `number` field for integers
`MoneyType`	A money field, with a specifiable currency
`NumberType`	A number field
`PercentType`	A number field for percentages
`RangeType`	A `range` input field (slider)

Form Fields: Choice Fields

Type	Description
`ChoiceType`	Field for selecting one or more options
`EnumType`	Choose from PHP enum cases
`EntityType`	Choose from a Doctrine entity
`CountryType`	Choose a country
`LanguageType`	Choose a language
`LocaleType`	Choose a locale
`TimezoneType`	Choose a timezone
`CurrencyType`	Choose a currency

The `expanded` and `multiple` options decide whether a dropdown, radio buttons or checkboxes are rendered.

Form Fields: Symfony UX Fields

Type	Description
`CropperType`	A Cropper.js image cropper field
`DropzoneType`	A Dropzone file upload field

Form Fields: Date and Time Fields

Type	Description
`DateType`	A date field
`DateIntervalType`	A date interval field
`DateTimeType`	A date and time field
`TimeType`	A time field
`BirthdayType`	A date field for birthdays
`WeekType`	Select a year and week

Form Fields: Other Fields

Type	Description
`CheckboxType`	A single checkbox field
`FileType`	A file upload field
`RadioType`	A radio button field
`HiddenType`	A hidden field

Form Fields: Field Groups

Type	Description
`CollectionType`	A group of fields that can be added or removed
`RepeatedType`	A field that is repeated (e.g. password confirmation)

See https://symfony.com/doc/current/reference/forms/types.html for a full list of form types.

Validation: Basic Constraints

```
use Symfony\Component\Validator\Constraints as Assert;
```

Attribute	Description
`#[Assert\NotBlank]`	Value is **not** an empty string/array, `false` or `null`
`#[Assert\Blank]`	Value is an empty string or `null`
`#[Assert\NotNull]`	Value is **not** strictly equal to `null`
`#[Assert\IsNull]`	Value is exactly equal to `null`
`#[Assert\Type('string')]`	Value is of a specific type

Validation: Comparison Constraints

Attribute	Description
`#[Assert\<Not>EqualTo('Foo')]`	Value is (not) equal to another value
`#[Assert\<Not>IdenticalTo('Foo')]`	Value is (not) identical to another value
`#[Assert\LessThan<OrEqual>(5)]`	Value is less than (or equal to) another value
`#[Assert\GreaterThan<OrEqual>(5)]`	Value is greater than (or equal to) another value
`#[Assert\Range(min: 2, max: 10)]`	Number or `DateTime` object is within a range

Validation: String Constraints

Attribute	Description
`#[Assert\Length(2, 10)]`	String is between <min> and <max> characters long
`#[Assert\Email]`	String is a valid email address
`#[Assert\Url]`	String is a valid URL
`#[Assert\Hostname]`	String is a valid hostname
`#[Assert\Ip]`	String is a valid IP address
`#[Assert\NoSuspiciousCharacters]`	String does not contain spoofing characters
`#[Assert\Uuid]`	String is a valid UUID
`#[Assert\CssColor]`	String is a valid CSS color

Validation: Password Constraints

Attribute	Description
`#[Assert\UserPassword]`	String is the authenticated user's password
`#[Assert\NotCompromisedPassword]`	Password is not found in any data breaches
`#[Assert\PasswordStrength]`	Password has a enough entropy

Validation: Number Constraints

Attribute	Description
`#[Assert\Positive<OrZero>]`	Value is positive (or zero)
`#[Assert\Negative<OrZero>]`	Value is negative (or zero)

Validation: Date and Time Constraints

Attribute	Description
`#[Assert\Date]`	Value is a valid date string in `Y-m-d` format
`#[Assert\DateTime]`	Value is a valid datetime in the specified format
`#[Assert\Time]`	Value is a valid time string in `H:i:s` format
`#[Assert\Timezone]`	Value is a valid timezone identifier

Validation: Choice Constraints

Attribute	Description
`#[Assert\Choice(['A', 'B', 'C'])]`	Value is one of the specified choices
`#[Assert\Country]`	Value is a valid ISO 3166-1 country code
`#[Assert\Language]`	Value is a valid two-letter language code
`#[Assert\Locale]`	Value is a valid locale identifier

Validation: File Constraints

Attribute	Description
`#[Assert\File]`	Value is a valid file path or `File` object
`#[Assert\Image]`	Same as `File`, but only accepts image MIME types

Validation: Financial and other Number Constraints

Attribute	Description
`#[Assert\Bic]`	Value is a valid Business Identifier Code (BIC)
`#[Assert\CardScheme]`	Value is a valid credit card number
`#[Assert\Currency]`	Value is a valid 3-letter ISO 4217 currency
`#[Assert\Luhn]`	Value passes the Luhn algorithm
`#[Assert\Iban]`	Value is a valid IBAN
`#[Assert\Isbn]`	Value is a valid ISBN

See https://symfony.com/doc/current/reference/constraints.html for a full list of validation constraints.

Symfony Twig Extensions

On top of the default Twig functions and filters, Symfony provides additional functions and filters. See https://symfony.com/doc/current/reference/twig_reference.html for a full reference.

Twig: Routing Functions

Function	Description
`path('home')`	Returns the relative URL for a route
`path('blog_post', {id: 3})`	Returns the relative URL for a route with parameters
`url('home')`	Returns the absolute URL for a route
`url('blog_post', {id: 3})`	Returns the absolute URL for a route with params
`url('home', [], true)`	Returns a scheme-relative URL for a route
`absolute_url('/blog/3')`	Returns the absolute URL for a path
`relative_path('/blog/3')`	Returns the relative URL for a path

Twig: Form Functions

Function	Description
`form(form)`	Renders the complete form
`form(form, {'method': 'GET'})`	Renders the form, overriding some options
`form_start(form)`	Renders the opening <`form`> tag
`form_end(form)`	Renders the closing </`form`> tag
`form_label(field)`	Renders the label for the given field
`form_help(field)`	Renders the help text for the given field
`form_errors(field)`	Renders any errors for the given field
`form_widget(field)`	Renders the HTML widget for the given field
`form_row(field)`	Renders the complete row for the given field, including the label, errors, help and widget
`form_rest(form)`	Renders all fields that have not been rendered yet

Twig: Form Tags

Tag	Description
`{% form_theme form 'bootstrap_5_layout.html.twig' %}`	Apply a form theme to a single form

Twig: Security Functions

Function	Description
`is_granted('ROLE_USER')`	Check if the current user has a role
`is_granted('edit', post)`	Check if the current user can edit a post
`csrf_token('delete-item')`	Generate a CSRF token for a specific intention
`logout_path('main')`	Generate a relative logout URL for the given firewall
`logout_url('main')`	Generate an absolute logout URL for the given firewall

Twig: Asset Functions

Function	Description
`asset('style.css')`	Returns the public path for an asset
`asset_version()`	Returns the asset version
`importmap()`	Outputs the importmap

Twig: Translation Filters

Filter	Description
`'foo'|trans`	Translate a message
`'foo'|trans({name: 'value'})`	Translate a message with parameters
`'foo'|trans({name: 'value'}, 'app')`	Translate a message from a specific domain
`'foo'|trans({}, null, 'nl')`	Translate a message to a specific locale

Twig: Translation Tags

Tag	Description
`{% trans %}message{% endtrans %}`	Translate a translation message in the default locale
`{% trans with {'name': 'value'} %}...`	Translate a message with parameters
`{% trans from 'app' %}...`	Translate a message from a specific domain
`{% trans into 'nl' %}...`	Translate a message to a specific locale
`{% trans_default_domain 'app' %}`	Set the default translation domain for this template

Twig: String Filters

Filter	Description
`'dateOfBirth'|humanize`	Transform a string into a human-readable format
`':+1:'|emojify`	Convert emoji shortcodes to Unicode emojis

Console: Basic Usage

Command	Description
`bin/console`	List all available commands
`bin/console <namespace>`	List all available commands in a namespace / bundle
`bin/console <command>`	Run a command
`bin/console help <command>`	Display help for a command
`bin/console --env=<env>`	Run the console in `dev`/`test`/`prod`
`bin/console -v|vv|vvv`	Increase output verbosity
`bin/console cache:clear`	Clear the cache
`bin/console cache:warmup`	Warm up an empty cache

Console: Debugging Commands

Command	Description
`bin/console debug:asset-map`	Display all mapped assets
`bin/console debug:autowiring`	List all autowireable services
`bin/console debug:config <bundle>`	Dump bundle configuration
`bin/console debug:container`	List all public services
`bin/console debug:container <service>`	Get information about a service
`bin/console debug:container --env-vars`	Display container env vars
`bin/console debug:container --tags`	List tagged public services
`bin/console debug:container --tag=<t>`	List services with a specific tag
`bin/console debug:dotenv`	Display parsed dotenv values
`bin/console debug:dotenv <search>`	Display matching dotenv values
`bin/console debug:event-dispatcher`	List all registered event listeners
`bin/console debug:firewall <name>`	Display detailed information about a firewall
`bin/console debug:form`	Display form type information
`bin/console debug:messenger`	List dispatchable messages
`bin/console debug:router`	List all registered routes
`bin/console debug:router <name>`	Get information about a route
`bin/console debug:translation <loc>`	Display messages for locale
`bin/console router:match <path>`	Show matching route(s) for a path

Console: Linting Commands

Command	Description
`bin/console lint:container`	Check service arguments for type mismatches
`bin/console lint:twig <directory>`	Lint Twig templates for syntax errors
`bin/console lint:yaml <directory>`	Lint YAML files for syntax errors
`bin/console lint:xliff <directory>`	Lint XLIFF translation files for syntax errors

Console: AssetMap Commands

Command	Description
`bin/console asset-map:compile`	Compile assets for production
`bin/console importmap:require <package>`	Add an NPM package to the import map
`bin/console importmap:install`	Download all importmap packages
`bin/console importmap:update`	Update all importmap packages
`bin/console importmap:remove <package>`	Remove a package from the importmap
`bin/console importmap:outdated`	List outdated importmap packages
`bin/console importmap:audit`	Check dependencies for vulnerabilities

Console: Doctrine Commands

Command	Description
`bin/console doctrine:database:create`	Create the configured database
`bin/console doctrine:database:drop`	Drop the configured database
`bin/console doctrine:mapping:info`	List mapped entities
`bin/console doctrine:schema:update --dump-sql`	Generate SQL statements to sync the DB schema
`bin/console doctrine:schema:update --force`	Sync the DB schema with the mapping information
`bin/console doctrine:schema:validate`	Validate the mapping files
`bin/console doctrine:query:sql <sql>`	Run a SQL query
`bin/console doctrine:query:dql <dql>`	Run a DQL query
`bin/console doctrine:fixtures:load`	Load data fixtures to the DB

Console: Autocompletion

```
bin/console completion bash | sudo tee /etc/bash_completion.d/console
```