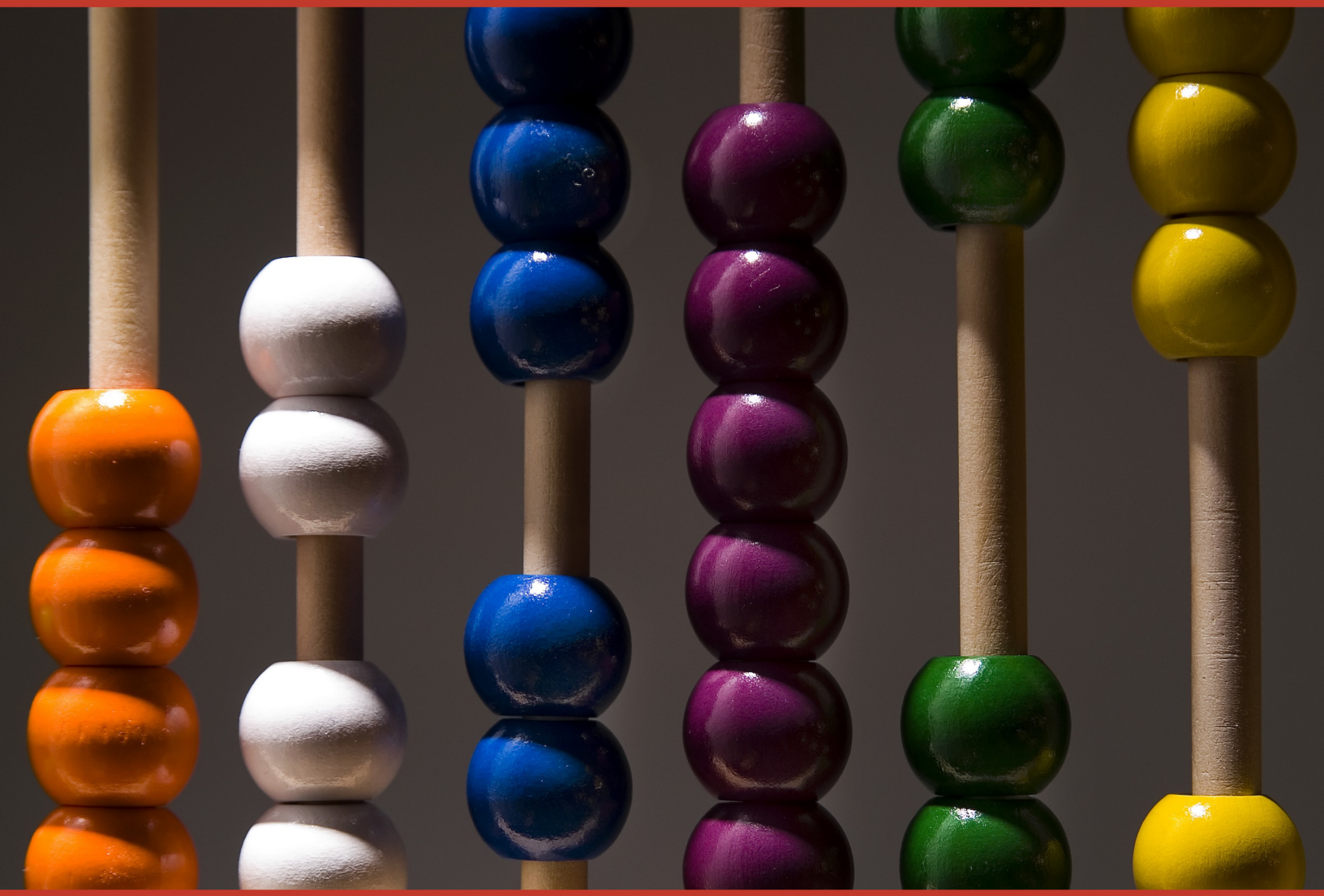

Symfony Certification
Unofficial self-study guide

275 training questions with answers and detailed explanations
Hands-on exercises and takeaways.
Symfony 2.3 LTS

by **Raúl Fraile**

Symfony Certification

Unofficial self-study guide

Raúl Fraile

This book is for sale at http://leanpub.com/symfony-selfstudy

This version was published on 2015-08-18

Tweet This Book!

Please help Raúl Fraile by spreading the word about this book on Twitter!

The suggested hashtag for this book is #symfony_selfstudy.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search?q=#symfony_selfstudy

"The only true wisdom is in knowing you know nothing." — Socrates

Contents

8. Twig

Exam goals

8.1. Auto escape
8.2. Template inheritance
8.3. Global functions
8.4. Filters
8.5. Template includes
8.6. Control statements (loops and conditions)
8.7. URLs generation
8.8. Call a controller from a view
8.9. Translations

Questions

1. Which of the following sentences are true about Twig?

1. The templating engine does not process PHP tags
2. Templates are compiled down to native PHP code at runtime
3. Templates are required to have the extension `.twig`
4. `path()` and `url()` are custom Twig functions added by Symfony

2. In Twig, are comments rendered when debug mode is enabled?

1. Yes
2. No
3. Only in the `dev` environment
4. Only when `twig.debug` is set to `true`

3. What is the syntax for comments in Twig?

1. `{{ ... }}`
2. `{# ... #}`
3. `{* ... *}`

4. `{- ... -}`

4. Are Twig templates recompiled when changes are made when debug mode is enabled?

1. Yes
2. No
3. Only in the `dev` environment
4. Only when `apc.stat` is set to `1` in the PHP settings

5. Output escaping in Twig...

1. ... is disabled by default
2. ... is enabled by default
3. ... can be disabled with the `e` filter
4. ... is necessary to prevent XSS attacks

6. If the current time is `16:22:42 Europe/Madrid`, does the following template print `16:22`?

```
{# time.html.twig #}
{{ now | date(timezone="Europe/Madrid", format="H:i") }}
```

1. Yes
2. No

7. Where is defined the `url()` function?

1. It belongs to the Twig core
2. In the official Twig extensions repository
3. Twig bridge
4. TwigBundle

8. What is the command to check the syntax of one or more Twig files?

1. `twig:validate`
2. `twig:syntax`
3. `twig:lint`
4. There is not such a command

9. What is the command to convert individual Twig templates to PHP ones?

1. `twig:export`
2. `twig:convert`
3. `twig:compile`
4. There is not such a command

10. What is the output of the following code inside a controller that extends from the base controller?

```
return $this->render('{{ var }}', ['var2' => 'hello']);
```

1. No errors but empty response as `var` is not defined
2. Error: The variable `var` is not defined
3. Error: The `render()` method expects a file path, not Twig code
4. Error: The `render()` method does not exist, it must be `renderView()`

11. How are variables declared in Twig templates?

1. `{{ define var = 'hello' }}`
2. `{% define var = 'hello' %}`
3. `{{ set var = 'hello' }}`
4. `{% set var = 'hello' %}`

12. Which of the following variable assignations are valid in Twig?

1. `{% set name = 'John' %}`
2. `{% set name %}John{% endset %}`
3. `{% set name %}{{ 'John' }}{% endset %}`
4. `{% set name, name = 'John', 'John' %}`

13. How can we get the current route name from Twig?

1. `{{ app.request.attributes._route }}`
2. `{{ app.request.attributes._routeName }}`
3. `{{ app.routing.route }}`
4. `{{ app.routing.routeName }}`

14. What application specific variables are available in the `app` global variable?

1. `app.security`

2. `app.user`
3. `app.request`
4. `app.session`

15. How can you get the name of the current environment from a Twig template?

1. `{{ env.name }}`
2. `{{ app.env.name }}`
3. `{{ app.environment }}`
4. `{{ app.environment.name }}`

16. How can you get the HTTP method of the current request from a Twig template?

1. `{{ app.request.method }}`
2. `{{ app.request.getMethod() }}`
3. `{{ app.request.http }}`
4. It is not possible to get the HTTP method from Twig without custom extensions

17. How can you get the current charset from a Twig template?

1. `{{ app.request.charset }}`
2. `{{ app.request.attributes._charset }}`
3. `{{ _charset }}`
4. It is not possible to get the current charset from Twig

18. What is the output of the following Twig template?

```php
// IndexController.php
$data = [
    'first' => 0,
    'first-page' => 1
];

return $this->render('index.html.twig', [
    'page' => 5,
    'data' => $data
]);
```

```twig
{# index.html.twig #}
{{ data.first-page }}
```

1. No output (empty response)
2. `0`
3. `1`
4. `-5`

19. Would the following Twig template throw an error?

```php
return $this->render('index.html.twig', []);
```

```twig
{# index.html.twig #}
{{ data }}
```

1. No. Twig never throws errors for undefined variables
2. Only when the `twig.strict_variables` is `true`
3. Only when the `twig.strict_variables` is `false`
4. Yes. Twig always throws errors for undefined variables

20. Which of the following filters are not available in Twig?

1. `strtolower`
2. `lowercase`
3. `lower`
4. `lowerize`

21. Is it possible to define additional global variables to be available in all Twig templates?

1. Yes, adding them to `twig.globals` in `config.yml`
2. Yes, registering a listener for the `kernel.view` event and injecting the variables before the template is parsed
3. Yes, creating a Twig extension and making use of the method `getGlobals()`
4. No

22. Which of the following solutions would be better to have a variable called `env` in all Twig templates containing the current environment name? (Only 1 answer)

1. Adding `{% set env = app.environment %}` in the base template which every other template inherits from
2. Adding `env: "%kernel.environment%"` to `twig.globals`
3. Overriding the render() method of the base controller
4. Looking for the definition of the *twig* service in TwigBundle and adding `<call method="addGlobal"> <argument>env</argument> <argument>%kernel.environment%</argument> </call>`

23. Given the following Twig template, what would be its output?

```
{# spaceless.html.twig #}
{% spaceless %}
    <div   id="hello">
        <span >Hello world</span>
    </div>
{% endspaceless %}
```

1. `<div id="hello"><span>Hello world</span></div>`
2. `<div id="hello"><span>Helloworld</span></div>`
3. `<divid="hello"><span>Hello world</span></div>`
4. `<div   id="hello"><span >Hello world</span></div>`

24. In Twig, `for` loops create a special variable to get some information such as the current iteration or whether the current iteration is the first/last. What is its name?

1. `loop`
2. `index`
3. `for`
4. `app`

25. What error do you get if you render the following Twig template from a controller, if the route name is `book_list`?

```
{# render_controller.html.twig #}
{% render(controller(app.request.attributes.get("_route"))) %}
```

1. `Maximum function nesting level reached`
2. `Unable to parse the controller name`
3. `Class "book" does not exist`
4. None of the above are correct

26. Which of the following are good use cases for the `cycle` function in Twig?

1. Zebra stripes on HTML tables
2. HTML nested menus
3. Previous/Next links in pagination
4. Breadcrumbs

27. What is the output of the following Twig template?

```
{# object_first.html.twig #}
{{ {one: 1, two: 2} | first }}
```

1. `one`
2. `1`
3. Error: `Array to string conversion`
4. Error: `Object to string conversion`

28. The goal of the following template is to choose a different layout for AJAX requests. Would it work?

```
{# conditional_layout.html.twig #}
{% set ajax = app.request.xmlHttpRequest %}
{% set prefix = 'AppBundle::' %}
{% extends prefix ~ (ajax ? "layout_ajax.html.twig" : "layout.html.twig") %}
{# ... #}
```

1. No, the method `xmlHttpRequest` doesn't exist
2. No, the `extends` instruction cannot depend on runtime variables
3. No, when present, `extends` must be the first instruction
4. Yes

29. In nested loops, how can you access to the `loop.index` variable of the parent loop?

1. `{{ parent.index }}`
2. `{{ parent.loop.index }}`
3. `{{ loop.parent.loop.index }}`
4. `{{ loop.loop.parent.loop.index }}`

30. Given the following Twig extension, what would be the output of `{{ hello("bye") }}`?

```php
// AppBundle/Twig/Extension/MagicExtension.php
class MagicExtension extends \Twig_Extension
{

    public function getFunctions()
    {
        return [
            new \Twig_SimpleFunction(
                '*',
                [$this, 'getMagic'],
                ['is_safe' => ['html']]
            )
        ];
    }

    public function getMagic()
    {
        return implode(':', func_get_args());
    }

    public function getName()
    {
        return 'magic';
    }
}
```

1. `bye`
2. `hello:bye`
3. `bye:hello`
4. Error: `The function "hello" does not exist`

31. What is the output of the following template if `not_found.html.twig` doesn't exist and `error.html.twig` contains `An error occurred`?

```twig
{# test.html.twig #}
{{ include('not_found.html.twig') | default('error.html.twig') }}
```

1. `An error occurred`
2. Error: `The filter "default" does not exist`
3. Error: `Unable to find template "not_found.html.twig"`
4. Error: `Unexpected token "An" of value "error"`

32. In which of the following scenarios would you use the Twig `verbatim` tag?

1. To output Twig code without parsing it
2. To optimize the template by removing unnecessary code
3. To define additional blocks in the parent template
4. None of the above are correct

33. What is the logical name of a Twig template located at `app/Resources/views/layout.html.twig`?

1. `Framework::layout.html.twig`
2. `:views:layout.html`
3. `app::layout.html.twig`
4. `::layout.html.twig`

Answers

1. Which of the following sentences are true about Twig?

Answers 1, 2 and 4 are correct. Twig does not process PHP tags, it just prints out the PHP code as if it were plain text. Before using Twig templates, they are compiled down to native PHP and cached for future use, so the performance impact is really low compared to plain PHP code.

2. In Twig, are comments rendered when debug mode is enabled?

Answer 2 is correct. Comments are never rendered in Twig, they are just ignored.

Executing Twig code online

TwigFiddle[1] is an exellent tool to execute Twig code online against different versions of Twig. If you want to make a quick test, it is probably the fastest option.

3. What is the syntax for comments in Twig?

Answer 2 is correct. Comments in Twig templates are wrapped between `{#` and `#}`. They can be

[1] http://twigfiddle.com/

multiline and their content is never rendered.

Like with PHP, unclosed comments throw an exception (a `Twig_Error_Syntax` exception):

```
{# unclosed_comment.html.twig #}
{#
{{ 'hello' }}
```

4. Are Twig templates recompiled when changes are made when debug mode is enabled?

Answer 1 is correct. When the `debug` mode is enabled, Twig templates are recompiled if the templating engine detects that the file has changed. This is the default mode in the `dev` environment, but it can be used in any other environment, even with `prod`. The `apc.stat` setting has no effect when compiling Twig templates, as APC (or OpCache) only caches PHP code (but when `apc.stat` is `0`, clearing the template cache won't update the APC cache).

In Symfony, Twig templates are compiled down in the `twig` directory of the environment cache (i.e. `app/cache/dev`). For example, the following simple template:

```
{# test.html.twig #}
{# print something #}

{{ 'hello world' }}
```

Is compiled down to:

```
// app/cache/dev/twig/8a/89/24ac44...php
/* AppBundle::test.html.twig */
class __TwigTemplate_8a8924ac extends Twig_Template
{
    public function __construct(Twig_Environment $env)
    {
        parent::__construct($env);

        $this->parent = false;

        $this->blocks = [];
    }

    protected function doDisplay(array $context, array $blocks = array())
    {
```

```
        // line 2
        echo "";

        // line 3
        echo "hello world";
    }

    public function getTemplateName()
    {
        return "AppBundle::test.html.twig";
    }

    public function getDebugInfo()
    {
        return array (  22 => 3,  19 => 2,);
    }
}
```

As you can see, it's plain PHP code. The class extends from `Twig_Template` and the `doDisplay()` method is responsible or rendering the template. The `{{ 'hello' }}` instruction has been converted to a simple `echo`, while the comment has just been ignored. The `getDebugInfo()` method is used for debugging purposes, and maps lines of the PHP compiled template and the Twig template. Here, the 3rd line of the template corresponds with the 22nd line of the PHP file.

5. Output escaping in Twig...

Answers 2 and 4 are correct. Output escaping is enabled by default in Twig templates and it can be disabled *per instruction* with the `raw` filter. It also prevents XSS[2] attacks: user-provided content will be escaped so HTML tags won't be parsed by the browser. Internally, `escape` uses the `htmlspecialchars()` PHP function.

For example, imagine that you have a forum where users can post messages. If HTML code is inserted in the message and are rendered unescaped, they would be interpreted by the browser. With output escaping, if the variable `code` contains `<a href="http://example.com"></a>`, `{{ code }}` is escaped and the browser doesn't interpret it as HTML code:

```
&lt;a href=&quot;http://example.com&quot;&gt;&lt;/a&gt;gt;
```

[2] http://en.wikipedia.org/wiki/Cross-site_scripting

6. If the current time is `16:22:42 Europe/Madrid`, does the following template print `16:22`?

Answer 2 is correct, as there is an error in the template. The variable `now` doesn't exist, it should be the string `"now", which is interpreted by the` date' filter as the current date and time:

```
{# date.html.twig #}
{{ "now" | date(timezone="Europe/Madrid", format="H:i") }}
```

In case you are wondering, named arguments are supported since Twig 1.12. The previous code is equivalent to these two:

```
{# date.html.twig #}
{{ "now" | date("H:i", "Europe/Madrid") }}

{{ "now" | date("H:i", timezone="Europe/Madrid") }}
```

`date` input values

Internally, the `date` filter makes use of the `date()` PHP function, so you can use any value that is accepted by the function, such as `+1 hour`. `{{ "+1 hour"|date("H:i", "Europe/Madrid") }}` would print `17:22`.

7. Where is defined the `url()` function?

Answer 3 is correct. The Twig bridge provides integration for Twig with different Symfony components, such as the `Routing` component. Twig functions such as `url()` or `path()` are defined there.

Twig bridge

In addition to `url()` and `path()`, the Twig bridge integrates Twig with other components like `Form`, `HttpKernel`, `Security` and `Yaml`. All functions and filters that are useful in a Symfony project but not for a templating engine in isolation, are probaby defined here: form-related functions, the `trans` and `transchoice` filters or other important functions like `render()`, `controller()` or `is_granted()`.

8. What is the command to check the syntax of one or more Twig files?

Answer 3 is correct. The `TwigBundle` bundle defines the command `twig:lint` to check the syntax of Twig files. For the validation process, it first tokenizes the contents of the Twig file and then tries to parse it. If an error is thrown, the file contains a syntax error. Remember that linters don't check other errors, like undefined variables or wrong array positions, so the following template will be valid for the linter but will throw an error when is executed:

```
{# no_array_key.html.twig #}
{% set data = [1, 2, 3] %}
{{ data[4] }}
```

Unlike the PHP linter, the `twig:lint` command **takes into account all tags, functions and filters defined by bundles**.

9. What is the command to convert individual Twig templates to PHP ones?

Answer 4 is correct. Out of the box, there is no command to convert Twig templates to PHP ones. By the way, the `TwigBundle` bundle defines a cache warmer to compile all twig templates and store them in cache when executing `cache:warm`.

10. What is the output of the following code inside a controller that extends from the base controller?

Answer 3 is correct. The `render()` method defined in the base controller throws an exception as it expects a file path, not actual Twig code.

11. How are variables declared in Twig templates?

Answer 4 is correct. The `set` tag is used to declare variables in Twig templates. The assigned value can be any valid Twig expression and several variables can be assigned in the same instruction.

`set` in loops

In Twig, loops are scoped, so any variable declared inside a loop won't be available outside.

12. Which of the following variable assignations are valid in Twig?

All answers are valid, and in all of them, the `name` variable contains `John`. The `set` tag can also be

used to capture the output generated by its body 'capture' chunks of text, that's why answers 2 and 3 work. Answer 4 works as well because in Twig you can set the same variable more than once.

13. How can we get the current route name from Twig?

Answer 1 is correct. The `app` global variable provides the `Request` object in `app.request`. This object contains a parameter bag called `attributes`, with information about the controller (`_controller`), the route (`_route`) and the route parameters (`_route_params`).

14. What application specific variables are available in the `app` global variable?

All answers are correct. The `app` global variable provides 6 application specific variables: `security`, `user`, `request`, `session`, `environment` and `debug`.

15. How can you get the name of the current environment from a Twig template?

Answer 3 is correct. The `app.environment` variable contains a string with the current environment: `dev`, `prod`, etc. While there are better ways (data collectors or HTTP headers), it can be used to output additional information for debugging purposes:

```
{# layout.html.twig #}
{# ... #}

{% if app.environment == 'dev' %}
    IP: {{ app.request.clientIp }}
{% endif %}

{# ... #}
```

16. How can you get the HTTP method of the current request from a Twig template?

Answers 1 and 2 are both correct. As `app.request` contains the `Request` object, you can get the current HTTP method using the `getMethod()` method. In addition, when using `app.request.method`, Twig does the following checks:

1. `method` is a valid property
2. `method` is a valid method

3. `getMethod()` is a valid method
4. `isMethod()` is a valid method

As `Request::getMethod()` exists, `{{ app.request.method }}` returns the same method as `{{ app.request.getMethod() }}`.

17. How can you get the current charset from a Twig template?

Answer 3 is correct. In addition to the `app` global variable (defined by Symfony), `self`, `context` and `charset` (defined by Twig itself) are always available in Twig templates.

18. What is the output of the following Twig template?

Answer 4 is correct. When attributes contain special characters such as `-`, that can be interpreted as the minus operator, the `attributes` function must be used. In this example, Twig is interpreting the code as **`{{ data.first - page }}`**.

19. Would the following Twig template throw an error?

Answer 2 is correct. Errors for undefined variables are only thrown when `twig.strict_variables` is `true`, otherwise it just returns `null`. By default, Symfony uses `%kernel.debug%` to set the value of `twig.strict_variables`, so errors will be thrown in the `dev` environment, but not in `prod`.

20. Which of the following filters are not available in Twig?

Answers 1, 2 and 4 are correct. To make a string lowercase, Twig provides the `lower` filter, which makes use of the `strtolower()` PHP function under the hood.

Similarly, `upper`, `capitalize` and `title` are also available to make strings uppercase, capitalized (all characters lowercase except the first one) or titlecased (all characters lowercase except the first one of each word):

```
{# prints "hello world" #}
{{ 'HELLO WORLD' | lower }}

{# prints "HELLO WORLD" #}
{{ 'hello world' | upper }}

{# prints "Hello world" #}
{{ 'HELLO WORLD' | capitalize }}
{{ 'hello world' | capitalize }}

{# prints "Hello World" #}
{{ 'HELLO WORLD' | title }}
{{ 'hello world' | title }}
```

21. Is it possible to define additional global variables to be available in all Twig templates?

Answers 1 and 3 are correct. The easiest way to add global variables for all Twig templates is using the `twig.globals` setting, and accepts primitive values, service container parameters and services. For more advanced use cases, Twig extensions can define global variables too. It is not possible to use a listener to add global variables as there is no event generated just before the template is parsed (`kernel.view` is only generated when the return value of a controller is not a `Response` object). Overriding the `render()` method of the base controller would also be an option.

22. Which of the following solutions would be better to have a variable called `env` in all Twig templates containing the current environment name? (Only 1 answer)

All answers could work, but the easiest and cleanest way is by adding the variable to the `twig.globals` setting. Answer 1 would not work for templates that don't inherit from the base template. Similarly, answer 3 would force all controllers to inherit from the base controller. Answer 4 is not recommended at all as you would be editing Symfony itself.

23. Given the following Twig template, what would be its output?

Answer 4 is correct. The `spaceless` tag removes whitespaces **between** HTML tags, not within tags or text.

24. In Twig, `for` loops create a special variable to get some information such as the current

iteration or whether the current iteration is the first/last. What is its name?

Answer `1` is correct. The `loop` array is created for the `for` loop scope and contains several keys:

- `index`: Current iteration starting from `1`.
- `index0`: Current iteration starting from `0`.
- `revindex`: Number of iterations from the end of the loop (`1` indexed).
- `revindex0`: Number of iterations from the end of the loop (`0` indexed).
- `first`: It's `true` when `index` is `1` (first iteration).
- `last`: It's `true` when `revindex` is `1` (last iteration).
- `length`: Number of elements in the sequence.
- `parent`: Parent context, which contains variables defined before the `for` loop or passed to the template from the controller.

For example, the following template would generate the resulting trace:

```
{# loop_variable.html.twig #}
{% for letter in ['a', 'b', 'c'] %}
    {# ... #}
{% endfor %}
```

Iteration	index	index0	revindex	revindex0	first	last	length
1	1	0	3	2	true	false	3
2	2	1	2	1	false	false	3
3	3	2	1	0	false	true	3

25. What error do you get if you render the following Twig template from a controller, if the route name is `book_list`?

Answer `2` is correct. The `controller()` function expects a controller name, not the current route name. As the route name (`book_list`) is not in the format `Bundle:Controller:Action`, it cannot be parsed.

Answer `1` would be true if you change `_route` by `_controller`, as it contains the value of the current controller in the right format. The following code generates an infinite loop, so when the maximum level is reached it throws an error:

```
{# infinite_loop.html.twig #}
{% render(controller(app.request.attributes.get("_controller"))) %}
```

26. Which of the following are good use cases for the `cycle` function in Twig?

Answer 1 is correct. Zebra stripes on HTML tables is a common use case for the `cycle` function, as it can be used to easily apply different styles to even and odd rows:

```
{# cycle.html.twig #}
{% set rows = [[1, 'one'], [2, 'two']] %}

<table>
    {% for row in rows %}
        <tr class="{{ cycle(['odd', 'even'], loop.index0) }}">
            <td>{{ row[0] }}</td>
            <td>{{ row[1] }}</td>
        </tr>
    {% endfor %}
</table>
```

The previous template generates the following HTML table:

```
<table>
    <tr class="odd">
        <td>1</td>
        <td>one</td>
    </tr>
    <tr class="even">
        <td>2</td>
        <td>two</td>
    </tr>
</table>
```

27. What is the output of the following Twig template?

Answer 2 is correct. The `first` filter works with arrays, hashes and strings, and returns the first element of the input data:

```twig
{# first_filter.html.twig #}
{{ ['hello', 'bye'] | first }} {# result: 'hello' #}
{{ {one: 'hello', two: 'bye'} | first }} {# result: 'hello' #}
{{ 'hello'| last }} {# result: 'h' #}
```

The `last` filter works the same way, but returns the last element of the input data:

```twig
{# last_filter.html.twig #}
{{ ['hello', 'bye'] | last }} {# result: 'bye' #}
{{ {one: 'hello', two: 'bye'} | last }} {# result: 'bye' #}
{{ 'hello'| last }} {# result: 'o' #}
```

28. The goal of the following template is to choose a different layout for AJAX requests. Would it work?

Answer 4 is correct, the approach is correct. It is possible to make a layout conditional, as well as making it depend on a variable. In fact, the template name can be any valid expression.

29. In nested loops, how can you access to the `loop.index` variable of the parent loop?

Answer 3 is correct. In Twig, each loop has its own scope, but the `loop` variable contains a reference to the parent scope. For example, the following template prints out `1 1 1 2 2 2 3 3 3`:

```twig
{# parent_loop.html.twig #}
{% for i in 'a'..'c' %}
    {% for j in 'a'..'c' %}
        {{ loop.parent.loop.index }}
    {% endfor %}
{% endfor %}
```

30. Given the following Twig extension, what would be the output of `{{ hello("bye") }}`?

Answer 2 is correct. It's basically a *catch-all* function. Twig supports dynamic functions since version 1.5, so if you define a function that includes `*`, it can be replaced by any string.

Symfony itself makes use of dynamic functions for the `render_*` function, which is executed when using `render_esi` or `render_hinclude`. This function is defined in the Twig bridge.

Similarly, you can define dynamic filters as well. The following example defines the dynamic filter `add_*`, which adds two numbers:

```
// AppBundle/Twig/Extension/MagicExtension.php
class MagicExtension extends \Twig_Extension
{

    public function getFilters()
    {
        return [
            new \Twig_SimpleFilter(
                'add_*',
                [$this, 'getAdd'],
                ['is_safe' => ['html']]
            )
        ];
    }

    public function getAdd($number, $input)
    {
        return $number + $input;
    }

    public function getName()
    {
        return 'magic';
    }
}
```

So, if you execute `{{ 1 | add_2 }}` the output will be `3`.

31. What is the output of the following template if `not_found.html.twig` doesn't exist and `error.html.twig` contains `An error occurred`?

Answer 3 is correct. The `default` filter returns the passed value if the input is undefined or empty, but it doesn't catch errors.

32. In which of the following scenarios would you use the Twig `verbatim` tag?

Answer 1 is correct. When Twig finds a `verbatim` section, its content is not parsed, but treated as

raw text. For example, the following template outputs `{{ [1, 2, 3] | join(', ') }}`:

```
{# verbatim.html.twig #}
{% verbatim %}
    {{ [1, 2, 3] | join(', ') }}
{% endverbatim %}
```

33. What is the logical name of a Twig template located at `app/Resources/views/layout.html.twig`?

Answer 4 is correct. When using logical names, they are composed of three fragments:

- Bundle name.
- Directory inside `Resources/views`.
- Template file name.

In the current example, as the template is not inside any bundle but belongs to the application, the first fragment is empty. Also, as it is not located in any subdirectory of `Resources/views`, the second fragment is empty as well. The following table contains some examples of templates and their logical names:

Location	Logical name
`app/Resources/views/layout.html.twig`	`::layout.html.twig`
`app/Resources/views/Static/tos.html.twig`	`:Static:tos.html.twig`
`src/AppBundle/Resources/views/mail.txt.twig`	`AppBundle::mail.txt.twig`

Takeaways

- **General**
  - The Twig bridge defines functions and filters that make use of Symfony components, such as `url()`, `path()`, `is_granted()` or `render()`. They are not included in the core of Twig.
  - Loops have their own scope and define a special variable called `loop`, which can be used to get the index and access to the parent scope.

  - Named parameters are supported in Twig since 1.12. `{{ "now" | date(timezone="Europe/Madrid", format="H:i") }}`, `{{ "now" | date("H:i", "Europe/Madrid") }}` and `{{ "now" | date("H:i", timezone="Europe/Madrid") }}` are equivalent.
  - The syntax of Twig templates can be checked with the `twig:lint` command. It takes into account all tags, functions and filters defined by bundles.
- **Output scaping**
  - Output escaping is enabled by default in Twig templates and it can be disabled per instruction with the raw filter.
  - It basically prevents XSS attacks.
- **Filters and functions**
  - Dynamic functions and filters allow to define functions/filters with placeholders using the character `*`.
  - The `spaceless` tag removes whitespaces between HTML tags, not within tags or text.
  - The `verbatim` tag can be used to render Twig code without parsing it.
- **Global variables**
  - `app`, `self`, `context` and `charset` are always available in Twig templates. `app` is added by Symfony, while the other three are defined by Twig.
  - The app global variable provides 6 application specific variables: `security`, `user`, `request`, `session`, `environment` and `debug`.
  - Global variables can be set from a custom extension (overriding the `getGlobals()` method) or with the `twig.globals` option.
- **Caching**
  - Twig templates are compiled down to PHP code and stored in the `%cache_dir%/twig` directory.
  - Twig templates are recompiled if the templating engine detects that the file has changed only if the debug mode is enabled.
  - Compiled templates contain information for debugging purposes. For example, the `getDebugInfo()` method maps lines of the PHP compiled template and the Twig template.

Answer sheet

8. Twig

Question	Correct answers
1	1, 2, 4
2	2
3	2
4	1
5	2, 4
6	2
7	3
8	3
9	4
10	3
11	4
12	1, 2, 3, 4
13	1
14	1, 2, 3, 4
15	3
16	1, 2
17	3
18	4
19	2
20	1, 2, 4
21	1, 3
22	2
23	4
24	1
25	2
26	1
27	2
28	4
29	3
30	2
31	3
32	1
33	4