



Pole position on the grid

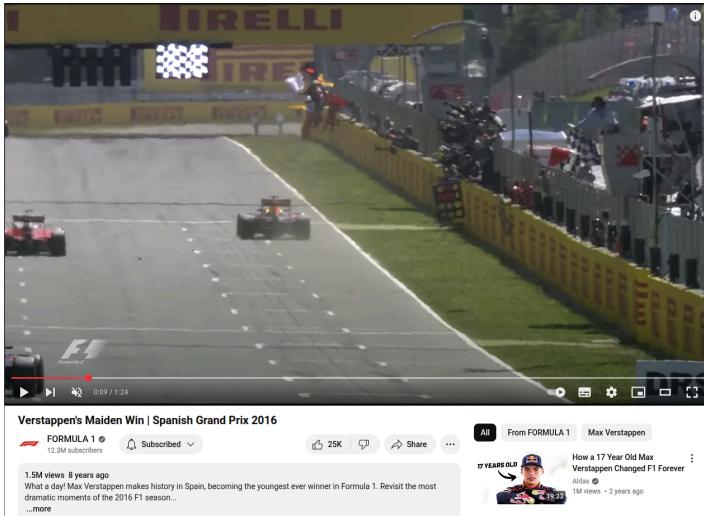


Awesome new Sylius Grid features to put you in the driver's seat



Estelle Gaits

- Web developer @Akawaka
- Sylius Key Contributor
- In June 2016, I was there :



Estelle Gaits

stlgaits · she/her

PHP / Symfony web developer

Edit profile

26 followers · 105 following

Akawaka

France

in/estellegaits

Achievements



Organizations



stlgaits / README.md

Adishatz 🌟

I'm Estelle. I am a PHP / Symfony backend developer from France 🇫🇷

Once upon a time, I used to be an MFL Teacher 🎓 (French & Spanish) in London 🇬🇧



Estelle Gaits

@totoche65 • May 13, 2022

PHP / Symfony developer

60 Reputation 3 Longest streak 104 Posts read

#webdev #productivity #devtools

#git #php

daily.dev

- ⚡ I'm a web developer at Akawaka
- ⚡ I'm currently learning Sylius and Sulu
- ⚡ I would like to learn Sulu, Svelte and GO as well as continue to explore API Platform.

My battleground

PHP SYMFONY PHPSTORM PHPTSTAN PHPUNIT MYSQL MARIADB DOCKER

CSS3 BOOTSTRAP TAILWINDCSS JAVASCRIPT REACT STIMULUS JQUERY

VISUALSTUDIOCODE

Loïc Frémont

- Technical Expert @Akawaka
- Sylius Key Contributor
- Sylius Stack Main Maintainer

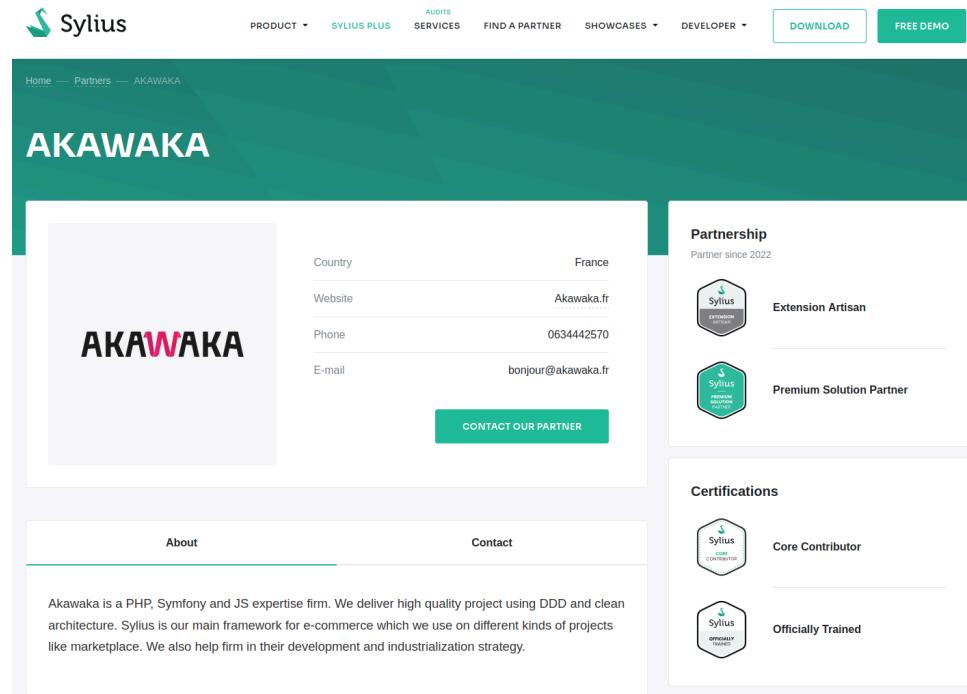
GitHub profile of Loïc Frémont (@loic425):

- Pinned:**
 - Sylius/Sylius (Public) - Open Source eCommerce Framework on Symfony
 - Sylius/Stack (Public) - The Sylius stack is a set of tools for your Symfony projects.
 - Sylius/SyliusResourceBundle (Public) - Simpler CRUD for Symfony applications
 - monofony/Monofony (Public) - Main repository for all Monofony bundles
 - akawakaweb/sylius-fixtures-plugin (Public) - This plugin provides an easy way to extend existing Sylius fixtures.
- Achievements:** Includes badges for contributing to Monofony, Behat, and other repositories.
- Organizations:** Shows his involvement with Akawaka, Monofony, and Behat.
- Activity overview:** Shows 1,816 contributions in the last year, with a heatmap of commits by month and day of the week. Contributions are broken down by repository: Sylius/Stack, Sylius/SyliusResourceBundle, Sylius/Sylius, and 37 other repositories.
- Contributions distribution:** A donut chart showing the breakdown of contributions: 64% Commits, 22% Pull requests, 2% Issues, and 12% Code review.

1. Akawaka & Sylius
2. What's under the hood of the
Sylius Stack?
3. What if... you were Toto Wolff?
4. Mystery surprise 🤫
5. Quick practice before it's your
time to shine!
6. Conclusion

Akawaka & Sylius

a true ❤️ story



The screenshot shows the Sylius Partner page for Akawaka. At the top, there's a navigation bar with links for PRODUCT, SYLIUS PLUS, AUDITS, SERVICES, FIND A PARTNER, SHOWCASES, DEVELOPER, DOWNLOAD, and FREE DEMO. Below the navigation, the page title is "AKAWAKA". On the left, there's a large image of the Akawaka logo, which consists of the word "AKAWAKA" in white with a pink "W". To the right of the image, there are contact details: Country (France), Website (Akawaka.fr), Phone (0634442570), and E-mail (bonjour@akawaka.fr). A "CONTACT OUR PARTNER" button is located below these details. On the right side of the page, there are two sections: "Partnership" (Partner since 2022) and "Certifications". The "Partnership" section includes icons for "Sylius Extension Artisan" and "Premium Solution Partner". The "Certifications" section includes icons for "Core Contributor" and "Officially Trained".

Home — Partners — AKAWAKA

AKAWAKA

AKAWAKA

Country France

Website Akawaka.fr

Phone 0634442570

E-mail bonjour@akawaka.fr

CONTACT OUR PARTNER

Partnership

Partner since 2022

Sylius Extension Artisan

Premium Solution Partner

Certifications

Sylius Core Contributor

Sylius Officially Trained

Akawaka & Sylius

a true ❤️ story

- Sylius Partner since 2022

The screenshot shows the Sylius Partner page for Akawaka. At the top, there's a navigation bar with links for PRODUCT, SYLIUS PLUS, AUDITS, SERVICES, FIND A PARTNER, SHOWCASES, DEVELOPER, DOWNLOAD, and FREE DEMO. Below the navigation, the page title is "AKAWAKA". On the left, there's a large image of the Akawaka logo, which consists of the word "AKAWAKA" in white and pink. To the right of the image, there's contact information: Country (France), Website (Akawaka.fr), Phone (0634442570), and E-mail (bonjour@akawaka.fr). A "CONTACT OUR PARTNER" button is located below this information. On the right side of the page, there are two sections: "Partnership" and "Certifications". The "Partnership" section includes icons for "Extension Artisan" and "Premium Solution Partner", both from Sylius. The "Certifications" section includes icons for "Core Contributor" and "Officially Trained", also from Sylius. Below the main content, there's a "About" section with a brief description of Akawaka's expertise and project delivery strategy.

Home — Partners — AKAWAKA

AKAWAKA

AKAWAKA

Country France

Website Akawaka.fr

Phone 0634442570

E-mail bonjour@akawaka.fr

CONTACT OUR PARTNER

About

Akawaka is a PHP, Symfony and JS expertise firm. We deliver high quality project using DDD and clean architecture. Sylius is our main framework for e-commerce which we use on different kinds of projects like marketplace. We also help firm in their development and industrialization strategy.

Contact

Partnership

Partner since 2022

Sylius Extension Artisan

Sylius Premium Solution Partner

Certifications

Core Contributor

Officially Trained

Akawaka & Sylius

a true ❤️ story

- Sylius Partner since 2022
- 4 amazing Sylius Key Contributors :
 - Loïc Frémont
 - Valentin Silvestre
 - Florian Merle
 - Estelle Gaits

The screenshot shows the Sylius Partner page for Akawaka. At the top, there's a navigation bar with links for Home, Partners, AKAWAKA, PRODUCT, SYLIUS PLUS, AUDITS, SERVICES, FIND A PARTNER, SHOWCASES, DEVELOPER, DOWNLOAD, and FREE DEMO. The main content area features the Akawaka logo and some contact information: Country (France), Website (Akawaka.fr), Phone (0634442570), and E-mail (bonjour@akawaka.fr). There's also a "CONTACT OUR PARTNER" button. To the right, there are sections for Partnership (Partner since 2022) showing badges for Sylius Extension Artisan and Premium Solution Partner, and a section for Certifications showing badges for Core Contributor and Officially Trained. Below the main content, there's a brief description of Akawaka's expertise: "Akawaka is a PHP, Symfony and JS expertise firm. We deliver high quality project using DDD and clean architecture. Sylius is our main framework for e-commerce which we use on different kinds of projects like marketplace. We also help firm in their development and industrialization strategy."

Search menu...

@ Dashboard

Library

@ Configuration

Conferences

Talks

Speakers

Dashboard



TALKS

5



SPEAKERS

3



CONFERENCES

5

2 Semaines

Mois

Année



New talks

TITLE

Voluptate architecto voluptatem debitis.
Stéphane Decock

Maiores voluptas perspiciatis cupiditate voluptas ut ut quo.
Mikolaj Król

Consequuntur modi explicabo voluptatibus debitisi nam.
Jacques Bodin-Hullin

Dicta aut et id voluptatem quia quod.
Manuele Menozzi

Velit perspiciatis assumenda voluptatem tempore.
Loïc Caillieux

[Show all](#)

New speakers

NAME

Julien Jacottet
Mezcalito

Hélène Gravelier
Synolia

Gregor Šink
Creatim d.o.o.

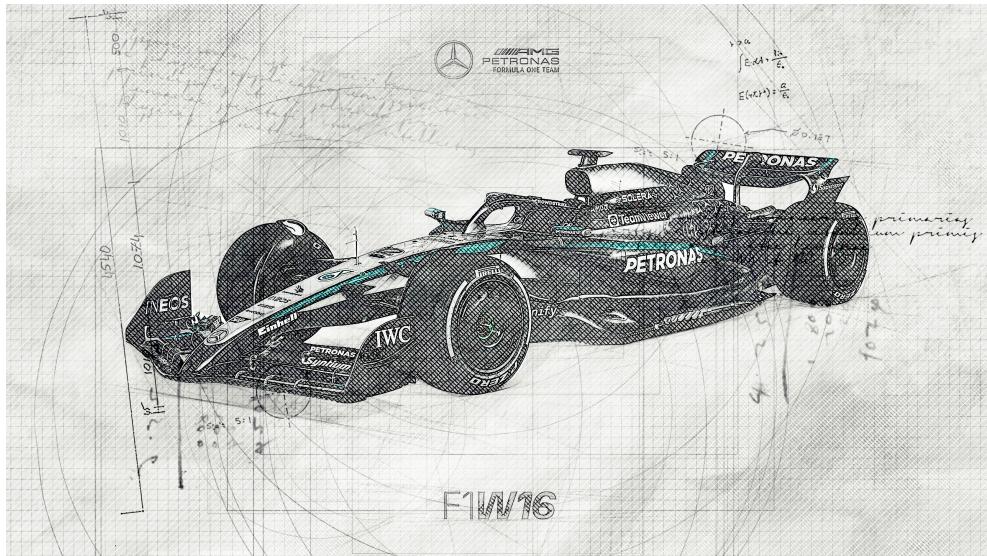
Joachim Løvgaard
Sylius Core Team Member, Setono

Stephan Hochdörfer
bitExpert AG

[Show all](#)

What's under the hood of the Sylius Stack?

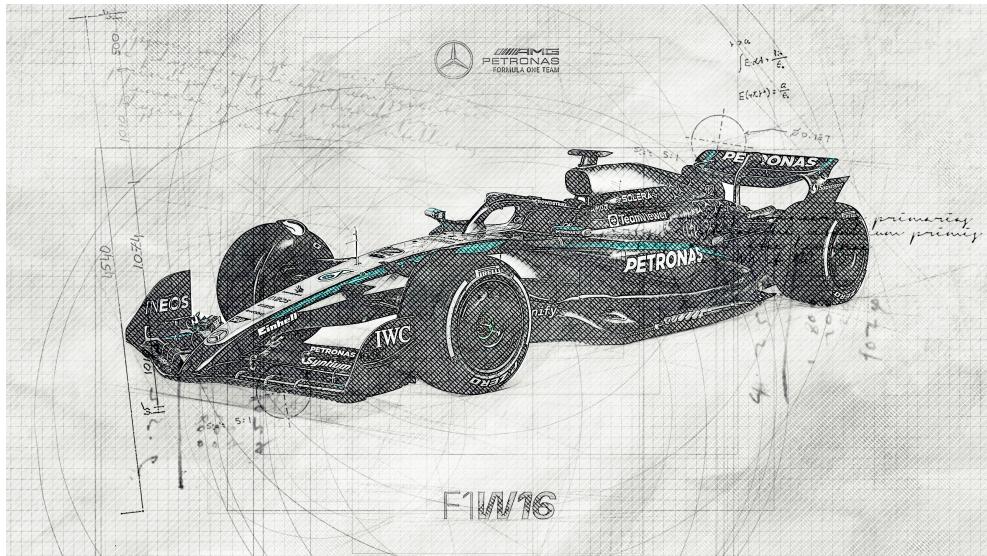
Create a lean & mean back-office in no time
with :



What's under the hood of the Sylius Stack?

Create a lean & mean back-office in no time
with :

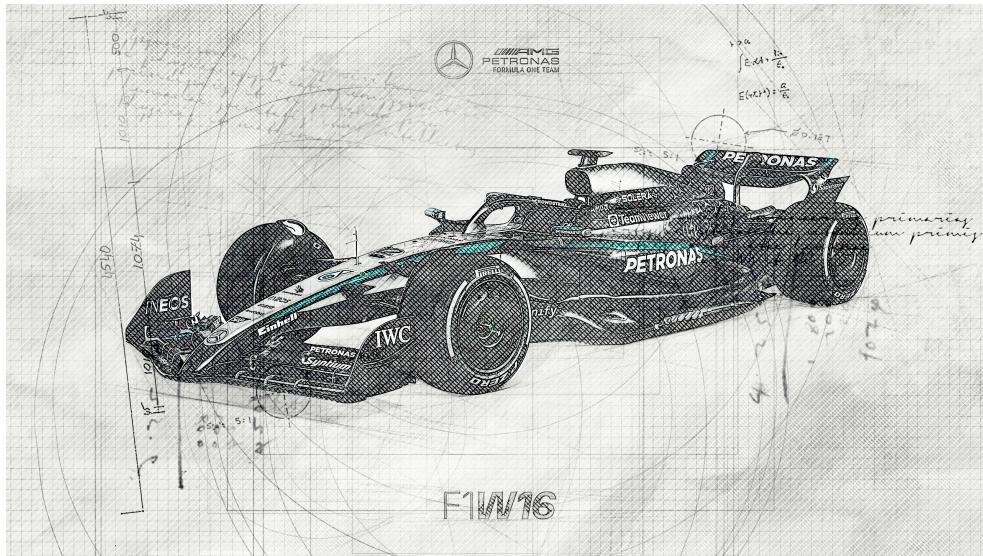
- **Sylius Grid bundle**



What's under the hood of the Sylius Stack?

Create a lean & mean back-office in no time
with :

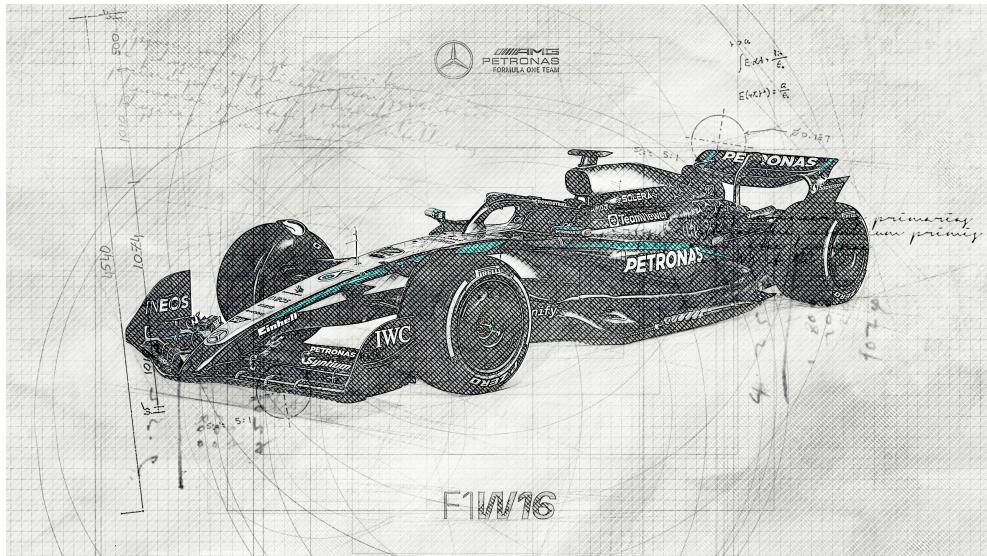
- **Sylius Grid bundle**
- Sylius Resource bundle



What's under the hood of the Sylius Stack?

Create a lean & mean back-office in no time
with :

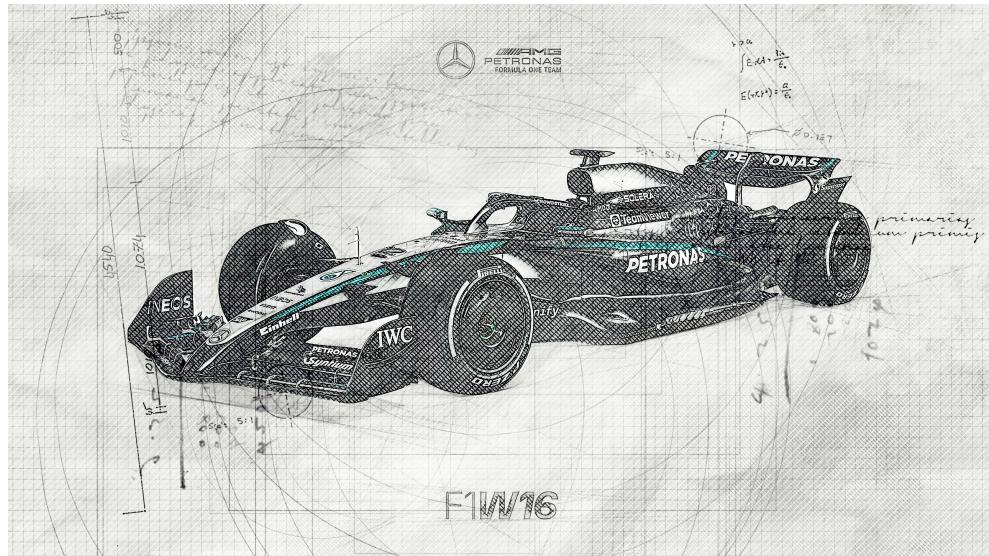
- **Sylius Grid bundle**
- Sylius Resource bundle
- Doctrine ORM and Doctrine DBAL drivers for
SyliusGridBundle



What's under the hood of the Sylius Stack?

Create a lean & mean back-office in no time
with :

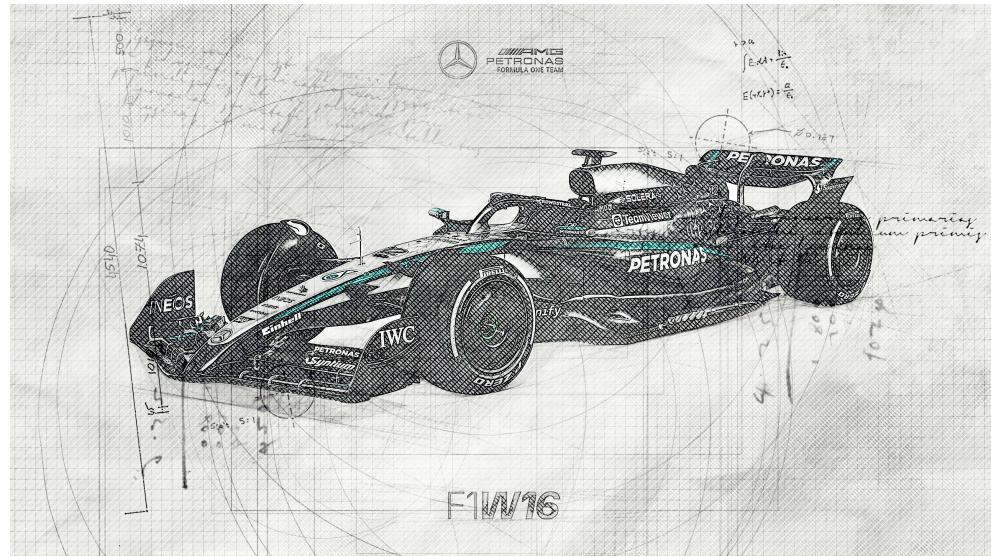
- **Sylius Grid bundle**
- Sylius Resource bundle
- Doctrine ORM and Doctrine DBAL drivers for
SyliusGridBundle
- Bootstrap Admin UI



What's under the hood of the Sylius Stack?

Create a lean & mean back-office in no time
with :

- **Sylius Grid bundle**
- Sylius Resource bundle
- Doctrine ORM and Doctrine DBAL drivers for SyliusGridBundle
- Bootstrap Admin UI
- Symfony UX, AssetMapper and more !



What if... you were Toto Wolff?

Let's create a Formula 1 admin panel using :

- the OpenF1 API
- the Sylius Stack
 - boosted `make:grid` command
 - new Sylius Grid Attributes
 - `#[AsGrid]`
 - `#[AsFilter]`



Starting slowly



Practice 1: let's generate a simple grid view

Dashboard / Drivers



Drivers

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAM
	1	Max	Verstappen	NED	Red Bull Racing
	2	Logan	Sargeant	USA	Williams
	4	Lando	Norris	GBR	McLaren
	10	Pierre	Gasly	FRA	Alpine
	11	Sergio	Perez	MEX	Red Bull Racing
	14	Fernando	Alonso	ESP	Aston Martin
	16	Charles	Leclerc	MON	Ferrari
	18	Lance	Stroll	CAN	Aston Martin
	20	Kevin	Magnussen	DEN	Haas F1 Team
	22	Yuki	Tsunoda	JPN	AlphaTauri

Showing 1 to 10 of 20 entries

< Previous 1 2 Next >

What do we need to generate an index grid?

- **routing & operations** (Sylius Resource attributes)
- a data **model** (can be a Sylius Resource)
- a **data provider** (Doctrine, API repo, etc)
- a **Sylius Grid** definition (structure: fields, filters, sorting, pagination and actions)

1- Create a Resource

```
#[AsResource(  
    templatesDir: '@SyliusAdminUi/crud',  
    operations: [  
        new Index(grid: DriverGrid::class),  
    ],  
)]  
final readonly class DriverResource implements ResourceInterface  
{  
    public function __construct(  
        public int $number,  
        public string $firstName,  
        public string $lastName,  
        public string $countryCode,  
        public string $teamName,  
        public string|null $image = null,  
    ) {  
    }  
  
    public function getId(): int  
    {  
        return $this->number;  
    }  
}
```

1- Create a Resource

```
#[AsResource(
    templatesDir: '@SyliusAdminUi/crud',
    operations: [
        new Index(grid: DriverGrid::class),
    ],
)]
final readonly class DriverResource implements ResourceInterface
{
    public function __construct(
        public int $number,
        public string $firstName,
        public string $lastName,
        public string $countryCode,
        public string $teamName,
        public string|null $image = null,
    ) {
    }

    public function getId(): int
    {
        return $this->number;
    }
}
```

1- Create a Resource

```
#[AsResource(  
    templatesDir: '@SyliusAdminUi/crud',  
    operations: [  
        new Index(grid: DriverGrid::class),  
    ],  
)]  
final readonly class DriverResource implements ResourceInterface  
{  
    public function __construct(  
        public int $number,  
        public string $firstName,  
        public string $lastName,  
        public string $countryCode,  
        public string $teamName,  
        public string|null $image = null,  
    ) {  
    }  
  
    public function getId(): int  
    {  
        return $this->number;  
    }  
}
```

■ Improved `make:grid` console command

3- Create a Grid

3- Create a Grid

 Now you can also create a grid based on a non-Doctrine entity, including... Sylius Resources !

3- Create a Grid

 Now you can also create a grid based on a non-Doctrine entity, including... Sylius Resources !

```
symfony console make:grid 'App\Resource\DriverResource'
```

3- Create a Grid

 Now you can also create a grid based on a non-Doctrine entity, including... Sylius Resources !

```
symfony console make:grid 'App\Resource\DriverResource'
```

 stlgaits commented last week • edited by loic425

The current implementation of the `make:grid` command assumes a Doctrine entity as the only valid input. This PR introduces enhanced flexibility to the command by allowing developers to generate grids based on resource classes beyond standard Doctrine entities.

With this change, you can use any PHP class—such as a Sylius Resource implementation—as the basis for your grid configuration. This makes the command more versatile and better suited to projects using custom resource patterns or non-standard data sources.

Example usage:

```
symfony console make:grid 'App\BoardGameBlog\Infrastructure\\Sylius\Resource\BoardGameResource'
```

This command will now correctly scaffold a grid using the specified class, even if it is not a traditional Doctrine entity.

3- Create a Grid

```
final class DriverResourceGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function __construct() { // TODO inject services if required }

    public static function getName(): string
    {
        return 'app_driver_resource';
    }

    public function getResourceClass(): string
    {
        return DriverResource::class;
    }

    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addField(StringField::create('firstName')->setLabel('FirstName')->setSortable(true))
            // ... all fields are added except "id"
            ->addActionGroup(MainActionGroup::create(CreateAction::create()))
            ->addActionGroup(BulkActionGroup::create(DeleteAction::create()))
            ->addActionGroup(ItemActionGroup::create(
                // ShowAction::create(),
                UpdateAction::create(),
                DeleteAction::create())))
    }
}
```

3- Create a Grid

```
final class DriverResourceGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function __construct() { // TODO inject services if required }

    public static function getName(): string
    {
        return 'app_driver_resource';
    }

    public function getResourceClass(): string
    {
        return DriverResource::class;
    }

    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addField(StringField::create('firstName')->setLabel('FirstName')->setSortable(true))
            // ... all fields are added except "id"
            ->addActionGroup(MainActionGroup::create(CreateAction::create()))
            ->addActionGroup(BulkActionGroup::create(DeleteAction::create()))
            ->addActionGroup(ItemActionGroup::create(
                // ShowAction::create(),
                UpdateAction::create(),
                DeleteAction::create())))
    }
}
```

3- Create a Grid

```
final class DriverResourceGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function __construct() { // TODO inject services if required }

    public static function getName(): string
    {
        return 'app_driver_resource';
    }

    public function getResourceClass(): string
    {
        return DriverResource::class;
    }

    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addField(StringField::create('firstName')->setLabel('FirstName')->setSortable(true))
            // ... all fields are added except "id"
            ->addActionGroup(MainActionGroup::create(CreateAction::create()))
            ->addActionGroup(BulkActionGroup::create(DeleteAction::create()))
            ->addActionGroup(ItemActionGroup::create(
                // ShowAction::create(),
                UpdateAction::create(),
                DeleteAction::create())))
    }
}
```

3- Create a Grid

```
final class DriverResourceGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function __construct() { // TODO inject services if required }

    public static function getName(): string
    {
        return 'app_driver_resource';
    }

    public function getResourceClass(): string
    {
        return DriverResource::class;
    }

    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addField(StringField::create('firstName')->setLabel('FirstName')->setSortable(true))
            // ... all fields are added except "id"
            ->addActionGroup(MainActionGroup::create(CreateAction::create()))
            ->addActionGroup(BulkActionGroup::create(DeleteAction::create()))
            ->addActionGroup(ItemActionGroup::create(
                // ShowAction::create(),
                UpdateAction::create(),
                DeleteAction::create())))
    }
}
```

Doctrine ORM manager for class "App\Resource\DriverResource" not found.

[Exception](#)[Logs](#)[1](#)[Stack Trace](#)

Sylius\Component\Grid\Exception\



RuntimeException

- + in vendor/sylius/grid-bundle/src/Bundle/Doctrine/ORM/Driver.php (line 43)
- + in vendor/sylius/grid-bundle/src/Component/Data/DataSourceProvider.php -> **getDataSource** (line 40)
- + in vendor/sylius/grid-bundle/src/Component/Data/DataProvider.php -> **getDataSource** (line 41)
- + in vendor/sylius/grid-bundle/src/Component/Data/Provider.php -> **getData** (line 34)
- + in vendor/sylius/resource-bundle/src/Bundle/Grid\View\ResourceGridViewFactory.php -> **getData** (line 46)
- + in vendor/sylius/resource-bundle/src/Bundle/Grid\View\LegacyGridViewFactory.php -> **create** (line 45)
- + in vendor/sylius/resource-bundle/src/Component/src/Grid/State/RequestGridProvider.php -> **create** (line 62)
- + in vendor/sylius/resource-bundle/src/Component/src/State/Provider.php -> **provide** (line 50)
- + in vendor/sylius/resource-bundle/src/Component/src/State/Provider/ReadProvider.php -> **provide** (line 44)
- + in vendor/sylius/resource-bundle/src/Component/src/State/Provider/FactoryProvider.php -> **provide** (line 36)
- + in vendor/sylius/resource-bundle/src/Component/src/Symfony\EventDispatcher/State/DispatchPostReadEventProvider.php -> **provide** (line 36)
- + in vendor/sylius/resource-bundle/src/Component/src/Symfony/Serializer/State/DeserializeProvider.php -> **provide** (line 38)
- + in vendor/sylius/resource-bundle/src/Component/src/Symfony/Form/State/FormProvider.php -> **provide** (line 39)

4- Add provider to grid

```
final class DriverGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->setProvider(DriverGridProvider::class)
            ->addField(
                StringField::create('firstName')
                    ->setLabel('FirstName')
                    ->setSortable(true)
            )
            // ...
    }
}
```

4- Add provider to grid

```
final class DriverGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->setProvider(DriverGridProvider::class)
            ->addField(
                StringField::create('firstName')
                    ->setLabel('FirstName')
                    ->setSortable(true)
            )
            // ...
    }
}
```

4- Add provider to grid

```
final class DriverGrid extends AbstractGrid implements ResourceAwareGridInterface
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->setProvider(DriverGridProvider::class)
            ->addField(
                StringField::create('firstName')
                    ->setLabel('FirstName')
                    ->setSortable(true)
            )
            // ...
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```

2- Create a custom Grid Data Provider

```
namespace App\Grid;

use Pagerfanta\Adapter\FixedAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with an empty paginator
        return new Pagerfanta(new FixedAdapter(0, []));
    }
}
```



Search menu...

Dashboard

Formula One

Drivers

Team radios

Sessions

Teams

Dashboard / Drivers

Drivers



No results found

Adjust your search and try again.

Use hardcoded data

```
// ...

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with a fixed data paginator
        return new Pagerfanta(new FixedAdapter(4, $this->getDrivers()));
    }

    private function getDrivers(): iterable
    {
        yield new DriverResource(number: 1, firstName: 'Max', lastName: 'Verstappen', countryCode: 'NED', teamName: 'Red Bull')
        yield new DriverResource(number: 2, firstName: 'Logan', lastName: 'Sargeant', countryCode: 'USA', teamName: 'Williams')
        yield new DriverResource(number: 4, firstName: 'Lando', lastName: 'Norris', countryCode: 'GBR', teamName: 'McLaren')
        yield new DriverResource(number: 44, firstName: 'Lewis', lastName: 'Hamilton', countryCode: 'GBR', teamName: 'Mercedes')
    }
}
```

Use hardcoded data

```
// ...

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with a fixed data paginator
        return new Pagerfanta(new FixedAdapter(4, $this->getDrivers()));
    }

    private function getDrivers(): iterable
    {
        yield new DriverResource(number: 1, firstName: 'Max', lastName: 'Verstappen', countryCode: 'NED', teamName: 'Red Bull Racing')
        yield new DriverResource(number: 2, firstName: 'Logan', lastName: 'Sargeant', countryCode: 'USA', teamName: 'Williams')
        yield new DriverResource(number: 4, firstName: 'Lando', lastName: 'Norris', countryCode: 'GBR', teamName: 'McLaren')
        yield new DriverResource(number: 44, firstName: 'Lewis', lastName: 'Hamilton', countryCode: 'GBR', teamName: 'Mercedes')
    }
}
```

Use hardcoded data

```
// ...

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        // start with a fixed data paginator
        return new Pagerfanta(new FixedAdapter(4, $this->getDrivers()));
    }

    private function getDrivers(): iterable
    {
        yield new DriverResource(number: 1, firstName: 'Max', lastName: 'Verstappen', countryCode: 'NED', teamName: 'Red Bull')
        yield new DriverResource(number: 2, firstName: 'Logan', lastName: 'Sargeant', countryCode: 'USA', teamName: 'Williams')
        yield new DriverResource(number: 4, firstName: 'Lando', lastName: 'Norris', countryCode: 'GBR', teamName: 'McLaren')
        yield new DriverResource(number: 44, firstName: 'Lewis', lastName: 'Hamilton', countryCode: 'GBR', teamName: 'Mercedes')
    }
}
```

Search menu...

[Dashboard](#)[Formula One](#)[Drivers](#)[Team radios](#)[Sessions](#)[Teams](#)

Dashboard / Drivers



Drivers

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAM
	1	Max	Verstappen	NED	Red Bull Racing
	2	Logan	Sargeant	USA	Williams
	4	Lando	Norris	GBR	McLaren
	44	Lewis	Hamilton	GBR	Mercedes

Showing 1 to 3 of 3 entries

[◀ Previous](#) [1](#) [Next ▶](#)

Now, let's use real data from the API!



```
namespace App\Grid\Provider;

use Pagerfanta\Adapter\ArrayAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;
use Symfony\Contracts\HttpClient\HttpClientInterface;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function __construct(
        private HttpClientInterface $openF1Client,
    ) {}

    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        return new Pagerfanta(new ArrayAdapter(iterator_to_array($this->getDrivers())));
    }

    private function getDrivers(): iterable
    {
        // ...
    }
}
```

```
namespace App\Grid\Provider;

use Pagerfanta\Adapter\ArrayAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;
use Symfony\Contracts\HttpClient\HttpClientInterface;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function __construct(
        private HttpClientInterface $openF1Client,
    ) {}

    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        return new Pagerfanta(new ArrayAdapter(iterator_to_array($this->getDrivers())));
    }

    private function getDrivers(): iterable
    {
        // ...
    }
}
```

```
namespace App\Grid\Provider;

use Pagerfanta\Adapter\ArrayAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;
use Symfony\Contracts\HttpClient\HttpClientInterface;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function __construct(
        private HttpClientInterface $openF1Client,
    ) {}

    public function getData(Grid $grid, Parameters $parameters): PagerFantaInterface
    {
        return new Pagerfanta(new ArrayAdapter(iterator_to_array($this->getDrivers())));
    }

    private function getDrivers(): iterable
    {
        // ...
    }
}
```

```
namespace App\Grid\Provider;

use Pagerfanta\Adapter\ArrayAdapter;
use Pagerfanta\Pagerfanta;
use Pagerfanta\PagerfantaInterface;
use Sylius\Component\Grid\Data\DataProviderInterface;
use Sylius\Component\Grid\Definition\Grid;
use Sylius\Component\Grid\Parameters;
use Symfony\Contracts\HttpClient\HttpClientInterface;

final readonly class DriverGridProvider implements DataProviderInterface
{
    public function __construct(
        private HttpClientInterface $openF1Client
    ) {}

    // ...

    private function getDrivers(): iterable
    {
        // ...
    }
}
```

```
use Sylius\Component\Grid\Data\DataProviderInterface;
use Symfony\Contracts\HttpClient\HttpClientInterface;

final readonly class DriverGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(): iterable
    {
        $responseData = $this->openF1Client->request('GET', '/v1/drivers?session_key=9158')->toArray();

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                firstName: $row['first_name'],
                lastName: $row['last_name'],
                countryCode: $row['country_code'],
                teamName: $row['team_name'],
                image: $row['headshot_url'],
            );
        }
    }
}
```

 Drivers

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAM
	1	Max	Verstappen	NED	Red Bull Racing
	2	Logan	Sargeant	USA	Williams
	4	Lando	Norris	GBR	McLaren
	10	Pierre	Gasly	FRA	Alpine
	11	Sergio	Perez	MEX	Red Bull Racing
	14	Fernando	Alonso	ESP	Aston Martin
	16	Charles	Leclerc	MON	Ferrari
	18	Lance	Stroll	CAN	Aston Martin
	20	Kevin	Magnussen	DEN	Haas F1 Team
	22	Yuki	Tsunoda	JPN	AlphaTauri

Showing 1 to 10 of 20 entries

< Previous 1 2 Next >

Practice 2:  The brand new #[AsGrid] attribute

Resource class & provider arguments

```
use App\Entity\Meeting;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    provider: MeetingGridProvider::class,
    resourceClass: Meeting::class, // any PHP object, including a Sylius resource
)]
final class MeetingGrid extends AbstractGrid
{
    public function __invoke(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

Resource class & provider arguments

```
use App\Entity\Meeting;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    provider: MeetingGridProvider::class,
    resourceClass: Meeting::class, // any PHP object, including a Sylius resource
)]
final class MeetingGrid extends AbstractGrid
{
    public function __invoke(
        GridBuilderInterface $gridBuilder,
    ): void
        // ...
    }
}
```

Resource class & provider arguments

```
use App\Entity\Meeting;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    provider: MeetingGridProvider::class,
    resourceClass: Meeting::class, // any PHP object, including a Sylius resource
)]
final class MeetingGrid extends AbstractGrid
{
    public function __invoke(
        GridBuilderInterface $gridBuilder,
    ): void
        // ...
    }
}
```

Resource class & provider arguments

```
use App\Entity\Meeting;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    provider: MeetingGridProvider::class,
    resourceClass: Meeting::class, // any PHP object, including a Sylius resource
)]
final class MeetingGrid extends AbstractGrid
{
    public function __invoke(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

Resource class & provider arguments

```
use App\Entity\Meeting;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    provider: MeetingGridProvider::class,
    resourceClass: Meeting::class, // any PHP object, including a Sylius resource
)]
final class MeetingGrid extends AbstractGrid
{
    public function __invoke(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

build method and name arguments

```
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    buildMethod: 'customBuildMethod',
    name: 'meeting', // optional - FQCN by default
    // ...
)]
final class MeetingGrid extends AbstractGrid
{
    public function customBuildMethod(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

build method and name arguments

```
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    buildMethod: 'customBuildMethod',
    name: 'meeting', // optional - FQCN by default
    // ...
)]
final class MeetingGrid extends AbstractGrid
{
    public function customBuildMethod(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

build method and name arguments

```
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    buildMethod: 'customBuildMethod',
    name: 'meeting', // optional - FQCN by default
    // ...
)]
final class MeetingGrid extends AbstractGrid
{
    public function customBuildMethod(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

build method and name arguments

```
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;

#[AsGrid(
    buildMethod: 'customBuildMethod',
    name: 'meeting', // optional - FQCN by default
    // ...
)]
final class MeetingGrid extends AbstractGrid
{
    public function customBuildMethod(
        GridBuilderInterface $gridBuilder,
    ): void
    {
        // ...
    }
}
```

[#AsGrid]

[#AsGrid]

- `_invoke()` instead of `buildGrid()` - but `buildGrid` still supported by default if no `_invoke()`

[#AsGrid]

- **_invoke()** instead of `buildGrid()` - but `buildGrid` still supported by default if no `_invoke()`
- **buildMethod** argument to declare custom build method

[#AsGrid]

- **_invoke()** instead of `buildGrid()` - but `buildGrid` still supported by default if no `_invoke()`
- **buildMethod** argument to declare custom build method
- **provider** argument instead of `->setProvider()`

[#AsGrid]

- **_invoke()** instead of `buildGrid()` - but `buildGrid` still supported by default if no `_invoke()`
- **buildMethod** argument to declare custom build method
- **provider** argument instead of `->setProvider()`
- **resourceClass** argument instead of `getResourceClass()`

[#AsGrid]

- **_invoke()** instead of `buildGrid()` - but `buildGrid` still supported by default if no `_invoke()`
- **buildMethod** argument to declare custom build method
- **provider** argument instead of `->setProvider()`
- **resourceClass** argument instead of `getResourceClass()`
- **name** argument (FQCN by default) instead of `getName()`



Emirates FLY BETTER

150

Practice 3 : Filter for specific drivers

using the brand new #[AsFilter] attribute

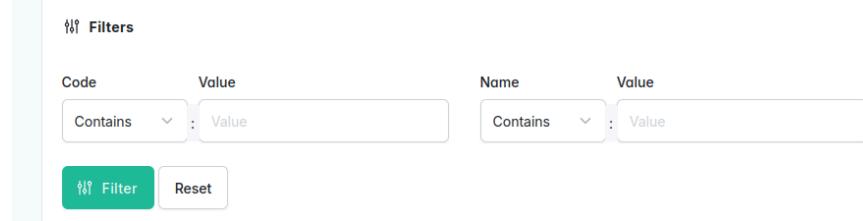
Filter types:

String filter example

Filters

Code	Value	Name	Value
Contains	:	Contains	:
Value		Value	

Filter **Reset**



Filter types:

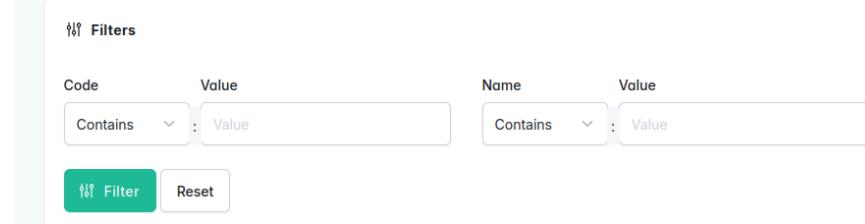
- String

String filter example

Filters

Code	Value	Name	Value
Contains	:	Contains	:
Value		Value	

Filter **Reset**



Filter types:

- String
- Boolean

String filter example

The screenshot shows a user interface for applying filters. At the top, there is a title "String filter example". Below it, the word "Filters" is displayed next to a magnifying glass icon. There are two main filter sections. The first section is labeled "Code" and "Value", containing a dropdown menu set to "Contains" and a text input field with the placeholder "Value". The second section is labeled "Name" and "Value", also containing a dropdown menu set to "Contains" and a text input field with the placeholder "Value". At the bottom of the filter area are two buttons: a teal-colored "Filter" button with a magnifying glass icon and a "Reset" button.

Filter types:

- String
- Boolean
- Date

String filter example

The screenshot shows a user interface for applying filters. At the top, there is a header labeled "Filters". Below the header, there are two sets of filter fields. Each set consists of a "Code" field and a "Value" field. In the first set, the "Code" field contains "Contains" and the "Value" field contains "Value". In the second set, the "Code" field also contains "Contains" and the "Value" field contains "Value". At the bottom of the filter area, there are two buttons: a green "Filter" button and a white "Reset" button.

Filter types:

- String
- Boolean
- Date
- Entity

String filter example

The screenshot shows a user interface for applying filters. At the top, there is a header labeled "Filters". Below the header, there are two sets of filter fields. The first set is for "Code" and the second for "Name". Each set includes a dropdown menu labeled "Contains" with a downward arrow, followed by a colon ":" and a text input field containing the placeholder "Value". Below these fields are two buttons: a green "Filter" button with a magnifying glass icon and a white "Reset" button.

Code	Value	Name	Value
Contains	:	Contains	:
Value		Value	

Code **Name**

Contains : Value Contains : Value

Filter **Reset**

Filter types:

- String
- Boolean
- Date
- Entity
- Money

String filter example

The screenshot shows a user interface for applying filters. It features two main sections: 'Code' and 'Name'. Each section has a dropdown menu set to 'Contains' and a text input field containing the value 'Value'. Below these fields are two buttons: a green 'Filter' button and a white 'Reset' button.

Code	Value	Name	Value
Contains	: Value	Contains	: Value

Code **Name**

Contains : Value Contains : Value

Filter **Reset**

Filter types:

- String
- Boolean
- Date
- Entity
- Money
- Exists

String filter example

The screenshot shows a user interface for applying filters. At the top, there is a header labeled "Filters". Below the header, there are two filter sections. The first section is for "Code" and the second for "Name". Each section has a dropdown menu set to "Contains" and a text input field containing the value "Value". Below these sections are two buttons: a green "Filter" button and a white "Reset" button.

Code	Value	Name	Value
Contains	: Value	Contains	: Value

Code **Name**
Contains : Value Contains : Value
Filter **Reset**

Filter types:

- String
- Boolean
- Date
- Entity
- Money
- Exists
- Select

String filter example

The screenshot shows a 'Filters' section with two rows of filter fields. The first row has a 'Code' column with a dropdown menu set to 'Contains' and a 'Value' input field containing 'Value'. The second row has a 'Name' column with a dropdown menu set to 'Contains' and a 'Value' input field containing 'Value'. Below each row is a green 'Filter' button and a grey 'Reset' button.

Code	Value	Name	Value
Contains	: Value	Contains	: Value

Filter **Reset**

Filter types:

- String
- Boolean
- Date
- Entity
- Money
- Exists
- Select
- Custom

String filter example

The screenshot shows a user interface for applying filters. At the top, there is a header labeled "Filters". Below it, there are two rows of filter fields.

Code	Value	Name	Value
Contains	:	Contains	:
Value		Value	

Below the filter rows are two buttons: a green "Filter" button and a white "Reset" button.

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

#[AsFilter]

```
namespace App\Grid\Filter;

use Sylius\Component\Grid\Attribute\AsFilter;
use Sylius\Component\Grid\Data\DataSourceInterface;
use Sylius\Component\Grid\Filtering\FilterInterface;
use Symfony\Component\Form\Extension\Core\Type\CountryType;

#[AsFilter(
    formType: CountryType::class, // Symfony Form Type to use
    template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig', // The Twig template
)]
final class CountryFilter implements FilterInterface
{
    public function apply(DataSourceInterface $dataSource, string $name, $data, array $options): void
    {
        // We handle the filtering part in the DriverGridProvider
    }
}
```

Insert the filter into our Grid

```
#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addFilter(
                Filter::create('country', CountryFilter::class)
                    ->setFormOptions([
                        'alpha3' => true,
                        'autocomplete' => true,
                    ])
                    ->setLabel('app.ui.country')
            )
            // ...
    }
}
```

Insert the filter into our Grid

```
#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addFilter(
                Filter::create('country', CountryFilter::class)
                    ->setFormOptions([
                        'alpha3' => true,
                        'autocomplete' => true,
                    ])
                    ->setLabel('app.ui.country')
            )
            // ...
    }
}
```

Insert the filter into our Grid

```
#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addFilter(
                Filter::create('country', CountryFilter::class)
                    ->setFormOptions([
                        'alpha3' => true,
                        'autocomplete' => true,
                    ])
                    ->setLabel('app.ui.country')
            )
            // ...
    }
}
```

Insert the filter into our Grid

```
#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addFilter(
                Filter::create('country', CountryFilter::class)
                    ->setFormOptions([
                        'alpha3' => true,
                        'autocomplete' => true,
                    ])
                    ->setLabel('app.ui.country')
            )
            // ...
    }
}
```

Actual filtering logic inside the provider

```
final readonly class DriverApiGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(array $criteria): iterable
    {
        $query = ['session_key' => 9158];

        if (!empty($criteria['country'] ?? null)) {
            $query['country_code'] = $criteria['country'];
        }

        $responseData = $this->openF1Client
            ->request(method: 'GET', url: '/v1/drivers', options: ['query' => $query])
            ->toArray()
            ;

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                // ...
            );
        }
    }
}
```

Actual filtering logic inside the provider

```
final readonly class DriverApiGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(array $criteria): iterable
    {
        $query = ['session_key' => 9158];

        if (!empty($criteria['country'] ?? null)) {
            $query['country_code'] = $criteria['country'];
        }

        $responseData = $this->openF1Client
            ->request(method: 'GET', url: '/v1/drivers', options: ['query' => $query])
            ->toArray()
            ;

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                // ...
            );
        }
    }
}
```

Actual filtering logic inside the provider

```
final readonly class DriverApiGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(array $criteria): iterable
    {
        $query = ['session_key' => 9158];

        if (!empty($criteria['country'] ?? null)) {
            $query['country_code'] = $criteria['country'];
        }

        $responseData = $this->openF1Client
            ->request(method: 'GET', url: '/v1/drivers', options: ['query' => $query])
            ->toArray()
            ;

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                // ...
            );
        }
    }
}
```

Actual filtering logic inside the provider

```
final readonly class DriverApiGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(array $criteria): iterable
    {
        $query = ['session_key' => 9158];

        if (!empty($criteria['country'] ?? null)) {
            $query['country_code'] = $criteria['country'];
        }

        $responseData = $this->openF1Client
            ->request(method: 'GET', url: '/v1/drivers', options: ['query' => $query])
            ->toArray()
            ;

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                // ...
            );
        }
    }
}
```

Actual filtering logic inside the provider

```
final readonly class DriverApiGridProvider implements DataProviderInterface
{
    // ...

    private function getDrivers(array $criteria): iterable
    {
        $query = ['session_key' => 9158];

        if (!empty($criteria['country'] ?? null)) {
            $query['country_code'] = $criteria['country'];
        }

        $responseData = $this->openF1Client
            ->request(method: 'GET', url: '/v1/drivers', options: ['query' => $query])
            ->toArray()
            ;

        foreach ($responseData as $row) {
            yield new DriverResource(
                number: $row['driver_number'],
                // ...
            );
        }
    }
}
```

Search menu...[Dashboard](#)[Formula One](#)[Drivers](#)

Dashboard / Drivers

Drivers

Filters

Country

fr

France

French Guiana

French Polynesia

French Southern Territories

South Africa

Central African Republic

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAMNAME
	1	Max	Verstappen	NED	Red Bull Racing
	2	Logan	Sargeant	USA	Williams
	4	Lando	Norris	GBR	McLaren
	10	Pierre	Gasly	FRA	Alpine
	11	Sergio	Perez	MEX	Red Bull Racing
	14	Fernando	Alonso	ESP	Aston Martin
	16	Charles	Leclerc	MON	Ferrari
	18	Lance	Stroll	CAN	Aston Martin
	20	Kevin	Magnussen	DEN	Halo F1 Team

Search menu...[Dashboard](#)[Formula One](#)[Drivers](#)[Dashboard](#) / Drivers

Drivers

Filters

Country

France

FilterReset

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAMNAME
	10	Pierre	Gasly	FRA	Alpine
	31	Esteban	Ocon	FRA	Alpine

Showing 1 to 2 of 2 entries

< Previous 1 Next >

[#AsFilter]

[#AsFilter]

- **formType** argument instead of getFormType()

[#AsFilter]

- **formType** argument instead of getFormType()
- **template** argument instead of defining in config/packages/sylius_grid.php

[#AsFilter]

- **formType** argument instead of getFormType()
- **template** argument instead of defining in config/packages/sylius_grid.php
- **type** argument for the name of your custom filter type (FQCN by default)

Connect Grids together

Add a link to another grid with filtered data

Search menu...[Dashboard](#)[!\[\]\(709f390d6469a8d4b40fea0ff3fbfcad_img.jpg\) Formula One](#)[Drivers](#)[Team radios](#)[Sessions](#)[Dashboard](#) / [Team radios](#)

Team radios

[Filters](#)[Show 10 ▾](#)

DRIVERNUMBER	DATE	ACTIONS
55	2023-09-15 09:32:51	
63	2023-09-15 09:34:32	
77	2023-09-15 09:35:06	
24	2023-09-15 09:36:47	
24	2023-09-15 09:38:27	
1	2023-09-15 09:39:02	
23	2023-09-15 09:39:37	
11	2023-09-15 09:40:43	
81	2023-09-15 09:44:39	
1	2023-09-15 09:45:45	

Showing 1 to 10 of 29 entries

< Previous [1](#) [2](#) [3](#) [Next >](#)

```
namespace App\Grid;

use Sylius\Bundle\GridBundle\Builder\Filter\StringFilter;
use Sylius\Bundle\GridBundle\Builder\GridBuilderInterface;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;
use Sylius\Component\Grid\Attribute\AsGrid;

#[AsGrid]
final class TeamRadioGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            // ...
            ->addFilter(
                StringFilter::create(name: 'driver_number', type: 'equal')
                    ->setLabel('app.ui.driver_number')
            )
            // ...
    }
}
```

```
namespace App\Grid;

use Sylius\Bundle\GridBundle\Builder\Filter\StringFilter;
use Sylius\Bundle\GridBundle\Builder\GridBuilderInterface;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;
use Sylius\Component\Grid\Attribute\AsGrid;

#[AsGrid]
final class TeamRadioGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            // ...
            ->addFilter(
                StringFilter::create(name: 'driver_number', type: 'equal')
                    ->setLabel('app.ui.driver_number')
            )
            // ...
    }
}
```

```
namespace App\Grid;

use Sylius\Bundle\GridBundle\Builder\Filter\StringFilter;
use Sylius\Bundle\GridBundle\Builder\GridBuilderInterface;
use Sylius\Bundle\GridBundle\Grid\AbstractGrid;
use Sylius\Component\Grid\Attribute\AsGrid;

#[AsGrid]
final class TeamRadioGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            // ...
            ->addFilter(
                StringFilter::create(name: 'driver_number', type: 'equal')
                    ->setLabel('app.ui.driver_number')
            )
            // ...
    }
}
```

```
use Sylius\Bundle\GridBundle\Builder\Action\Action;
use Sylius\Bundle\GridBundle\Builder\ActionGroup\ItemActionGroup;

#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addActionGroup(
                ItemActionGroup::create(
                    Action::create('team_radios', 'show')
                        ->setOptions([
                            'link' => [
                                'route' => 'app_admin_team_radio_index',
                                'parameters' => [
                                    'criteria' => [
                                        'driver_number' => [
                                            'value' => 'resource.number', // driverResource->number
                                        ],
                                    ],
                                ],
                            ],
                        ],
                    ]
                )
            ->setLabel('app.ui.show_team_radios')
            ->setIcon('tabler:radio'),
    }
}
```

```
use Sylius\Bundle\GridBundle\Builder\Action\Action;
use Sylius\Bundle\GridBundle\Builder\ActionGroup\ItemActionGroup;

#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addActionGroup(
                ItemActionGroup::create(
                    Action::create('team_radios', 'show')
                        ->setOptions([
                            'link' => [
                                'route' => 'app_admin_team_radio_index',
                                'parameters' => [
                                    'criteria' => [
                                        'driver_number' => [
                                            'value' => 'resource.number', // driverResource->number
                                        ],
                                    ],
                                ],
                            ],
                        ],
                    )
                )
            ->setLabel('app.ui.show_team_radios')
            ->setIcon('tabler:radio'),
    }
}
```

```
use Sylius\Bundle\GridBundle\Builder\Action\Action;
use Sylius\Bundle\GridBundle\Builder\ActionGroup\ItemActionGroup;

#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addActionGroup(
                ItemActionGroup::create(
                    Action::create('team_radios', 'show')
                        ->setOptions([
                            'link' => [
                                'route' => 'app_admin_team_radio_index',
                                'parameters' => [
                                    'criteria' => [
                                        'driver_number' => [
                                            'value' => 'resource.number', // driverResource->number
                                        ],
                                    ],
                                ],
                            ],
                        ],
                    )
                )
            ->setLabel('app.ui.show_team_radios')
            ->setIcon('tabler:radio'),
    }
}
```

```
use Sylius\Bundle\GridBundle\Builder\Action\Action;
use Sylius\Bundle\GridBundle\Builder\ActionGroup\ItemActionGroup;

#[AsGrid]
final class DriverGrid extends AbstractGrid
{
    public function buildGrid(GridBuilderInterface $gridBuilder): void
    {
        $gridBuilder
            ->addActionGroup(
                ItemActionGroup::create(
                    Action::create('team_radios', 'show')
                        ->setOptions([
                            'link' => [
                                'route' => 'app_admin_team_radio_index',
                                'parameters' => [
                                    'criteria' => [
                                        'driver_number' => [
                                            'value' => 'resource.number', // driverResource->number
                                        ],
                                    ],
                                ],
                            ],
                        ],
                    ]
                )
            ->setLabel('app.ui.show_team_radios')
            ->setIcon('tabler:radio'),
    }
}
```

Search menu...

[Dashboard](#)[Formula One](#)[Drivers](#)[Team radios](#)[Sessions](#)

Dashboard / Drivers

 Drivers

Filters

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAMNAME	ACTIONS
	1	Max	Verstappen	NED	Red Bull Racing	
	2	Logan	Sargeant	USA	Williams	
	4	Lando	Norris	GBR	McLaren	
	10	Pierre	Gasly	FRA	Alpine	
	11	Sergio	Perez	MEX	Red Bull Racing	
	14	Fernando	Alonso	ESP	Aston Martin	
	16	Charles	Leclerc	MON	Ferrari	
	18	Lance	Stroll	CAN	Aston Martin	
	20	Kevin	Magnussen	DEN	Haas F1 Team	

Search menu...

[Dashboard](#)[Formula One](#)[Drivers](#)[Team radios](#)[Sessions](#)

Dashboard / Drivers

 Drivers

Filters

Show 10 ▾

IMAGE	NUMBER	FIRSTNAME	LASTNAME	COUNTRYCODE	TEAMNAME	ACTIONS
	1	Max	Verstappen	NED	Red Bull Racing	
	2	Logan	Sargeant	USA	Williams	
	4	Lando	Norris	GBR	McLaren	
	10	Pierre	Gasly	FRA	Alpine	
	11	Sergio	Perez	MEX	Red Bull Racing	
	14	Fernando	Alonso	ESP	Aston Martin	
	16	Charles	Leclerc	MON	Ferrari	
	18	Lance	Stroll	CAN	Aston Martin	
	20	Kevin	Magnussen	DEN	Haas F1 Team	

Search menu...[Dashboard](#) [Formula One](#)[Drivers](#)[Team radios](#)[Sessions](#)[Dashboard](#) / [Team radios](#)

Team radios

 [Filters](#)

Driver number

[Filter](#)[Reset](#)[Show 10 ▾](#)

DRIVERNUMBER	DATE	ACTIONS
1	2023-09-15 09:39:02	
1	2023-09-15 09:45:45	
1	2023-09-15 10:04:14	

Showing 1 to 3 of 3 entries

< Previous 1 Next >

Push, Push, Push!





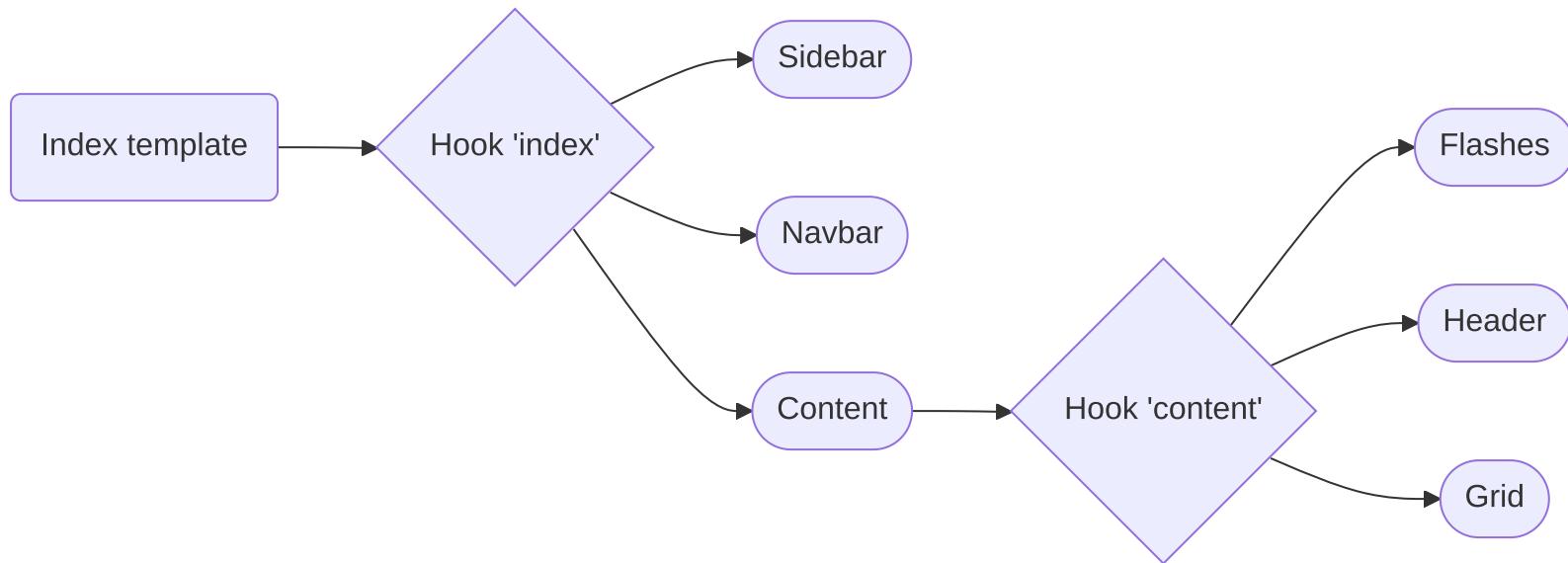
New Grid Live Component

it's alive...



Twig hooks overview

index operation



Twig hooks overview

index operation

```
└ (Template) [↑ 0, ◎ 0 ms] content (@SyliusBootstrapAdminUi/shared/crud/common/content.html.twig)
  └ (Hook) sylius_admin.driver.index.content, sylius_admin.common.index.content
    └ (Template) [↑ 0, ◎ 1 ms] flashes (@SyliusBootstrapAdminUi/shared/crud/common/content/flashes.html.twig)
    └ (Template) [↑ 0, ◎ 0 ms] header (@SyliusBootstrapAdminUi/shared/crud/common/content/header.html.twig)
      └ (Hook) sylius_admin.driver.index.content.header, sylius_admin.common.index.content.header
        └ (Template) [↑ 0, ◎ 1 ms] breadcrumbs (@SyliusBootstrapAdminUi/shared/crud/index/content/header/breadcrumbs.html.twig)
        └ (Template) [↑ 0, ◎ 0 ms] title_block (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block.html.twig)
          └ (Hook) sylius_admin.driver.index.content.header.title_block, sylius_admin.common.index.content.header.title_block
            └ (Template) [↑ 0, ◎ 11 ms] title (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block/title.html.twig)
            └ (Template) [↑ 0, ◎ 1 ms] actions (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block/actions.html.twig)
    └ (Template) [↑ 0, ◎ 0 ms] grid (shared/crud/index/content/grid.html.twig)
      └ (Hook) sylius_admin.driver.index.content.grid, sylius_admin.common.index.content.grid
        └ (Template) [↑ 0, ◎ 1 ms] filters (@SyliusBootstrapAdminUi/shared/crud/index/content/grid/filters.html.twig)
        └ (Template) [↑ 0, ◎ 39 ms] data_table (@SyliusBootstrapAdminUi/shared/crud/index/content/grid/data_table.html.twig)
        └ (Template) [↑ 0, ◎ 0 ms] no_data_block (@SyliusBootstrapAdminUi/shared/crud/index/content/grid/no_results.html.twig)
```

Twig hooks updated

index operation

```
L (Template) [↑ 0, ◎ 0 ms] content (@SyliusBootstrapAdminUi/shared/crud/common/content.html.twig)
  L (Hook) sylius_admin.driver.index.content, sylius_admin.common.index.content
    L (Template) [↑ 0, ◎ 0 ms] flashes (@SyliusBootstrapAdminUi/shared/crud/common/content/flashes.html.twig)
    L (Template) [↑ 0, ◎ 0 ms] header (@SyliusBootstrapAdminUi/shared/crud/common/content/header.html.twig)
      | L (Hook) sylius_admin.driver.index.content.header, sylius_admin.common.index.content.header
      |   L (Template) [↑ 0, ◎ 0 ms] breadcrumbs (@SyliusBootstrapAdminUi/shared/crud/index/content/header/breadcrumbs.html.twig)
      |   L (Template) [↑ 0, ◎ 0 ms] title_block (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block.html.twig)
      |     L (Hook) sylius_admin.driver.index.content.header.title_block, sylius_admin.common.index.content.header.title_block
      |       L (Template) [↑ 0, ◎ 4 ms] title (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block/title.html.twig)
      |       L (Template) [↑ 0, ◎ 0 ms] actions (@SyliusBootstrapAdminUi/shared/crud/common/content/header/title_block/actions.html.twig)
    L (Template) [↑ 0, ◎ 0 ms] grid (shared/crud/index/content/grid.html.twig)
      L (Hook) sylius_admin.driver.index.content.grid, sylius_admin.common.index.content.grid
        L (Template) [↑ 0, ◎ 0 ms] filters (@SyliusBootstrapAdminUi/shared/crud/index/content/grid/filters.html.twig)
        L (Component) [↑ 0, ◎ 99 ms] data_table (sylius_grid_data_table)
        L (Template) [↑ 0, ◎ 0 ms] no_data_block (@SyliusBootstrapAdminUi/shared/crud/index/content/grid/no_results.html.twig)
```

Overview of the new DataTableComponent

```
#[AsLiveComponent(name: 'sylius_grid_data_table')]
final class DataTableComponent
{
    #[LiveProp(writable: true)]
    public string|null $grid = null;

    #[LiveProp(writable: true)]
    public int $page = 1;

    #[LiveProp(writable: true)]
    public array|null $criteria = null;

    #[LiveProp(writable: true)]
    public array|null $sorting = null;

    #[LiveProp(writable: true)]
    public int|null $limit = null;
}
```

Transform your grid into a Live Component

```
sylius twig_hooks:
hooks:
'sylius_admin.book.index.content.grid':
    data_table:
        component: 'sylius_grid_data_table'
        props:
            grid: '@=_context.grid'
            page: '@=_context.page'
            criteria: '@=_context.criteria'
            sorting: '@=_context.sorting'
            limit: '@=_context.limit'
```

Overview of the new DataTableComponent

```
#[AsLiveComponent(name: 'sylius_grid_data_table')]
final class DataTableComponent
{
    #[LiveProp(writable: true)]
    public string|null $grid = null;

    #[LiveProp(writable: true)]
    public int $page = 1;

    #[LiveProp(writable: true)]
    public array|null $criteria = null;

    #[LiveProp(writable: true)]
    public array|null $sorting = null;

    #[LiveProp(writable: true)]
    public int|null $limit = null;
}
```

Transform your grid into a Live Component

```
sylius twig_hooks:
hooks:
'sylius_admin.book.index.content.grid':
    data_table:
        component: 'sylius_grid_data_table'
        props:
            grid: '@=_context.grid'
            page: '@=_context.page'
            criteria: '@=_context.criteria'
            sorting: '@=_context.sorting'
            limit: '@=_context.limit'
```

Overview of the new DataTableComponent

```
#[AsLiveComponent(name: 'sylius_grid_data_table')]
final class DataTableComponent
{
    #[LiveProp(writable: true)]
    public string|null $grid = null;

    #[LiveProp(writable: true)]
    public int $page = 1;

    #[LiveProp(writable: true)]
    public array|null $criteria = null;

    #[LiveProp(writable: true)]
    public array|null $sorting = null;

    #[LiveProp(writable: true)]
    public int|null $limit = null;
}
```

Transform your grid into a Live Component

```
sylius twig_hooks:
hooks:
'sylius_admin.book.index.content.grid':
data_table:
component: 'sylius_grid_data_table'
props:
grid: '@=_context.grid'
page: '@=_context.page'
criteria: '@=_context.criteria'
sorting: '@=_context.sorting'
limit: '@=_context.limit'
```

Overview of the new DataTableComponent

```
#[AsLiveComponent(name: 'sylius_grid_data_table')]
final class DataTableComponent
{
    #[LiveProp(writable: true)]
    public string|null $grid = null;

    #[LiveProp(writable: true)]
    public int $page = 1;

    #[LiveProp(writable: true)]
    public array|null $criteria = null;

    #[LiveProp(writable: true)]
    public array|null $sorting = null;

    #[LiveProp(writable: true)]
    public int|null $limit = null;
}
```

Transform your grid into a Live Component

```
sylius twig_hooks:
hooks:
'sylius_admin.book.index.content.grid':
    data_table:
        component: 'sylius_grid_data_table'
        props:
            grid: '@=_context.grid'
            page: '@=_context.page'
            criteria: '@=_context.criteria'
            sorting: '@=_context.sorting'
            limit: '@=_context.limit'
```

▶ 0:00 / 0:09



Use it in any template

Including your grid in a details page.

```
<!-- templates/session/show/body.html.twig -->
{{ component('sylius_grid_data_table', {
    grid: 'driver',
    criteria: {
        session: session.id,
    },
}) }}
```

Use it in any template

Including your grid in a details page.

```
<!-- templates/session/show/body.html.twig -->
{{ component('sylius_grid_data_table', {
    grid: 'driver',
    criteria: {
        session: session.id,
    },
}) }}
```

▶ 0:00 / 0:06





0:00 / 0:32



Quick practice before it's your time to shine!

Q1: Find the odd one out.

Quick practice before it's your time to shine!

Q1: Find the odd one out.

- `#[AsFilter]`

Quick practice before it's your time to shine!

Q1: Find the odd one out.

- `#[AsFilter]`
- `#[AsGrid]`

Quick practice before it's your time to shine!

Q1: Find the odd one out.

- `#[AsFilter]`
- `#[AsGrid]`
- `#[AsProvider]`

Quick practice before it's your time to shine!

Q1: Find the odd one out.

- `#[AsFilter]`
- `#[AsGrid]`
- `#[AsProvider]`
- `#[AsResource]`

Quick practice before it's your time to shine!

Q1: Find the odd one out.

- `#[AsFilter]`
- `#[AsGrid]`
- `#[AsProvider]`
- `#[AsResource]`

Q2: The OpenF1 API does not include a **/teams** endpoint, but you would like to create a grid listing all teams. You've created this simple model... :

```
final readonly class Team
{
    public function __construct(
        public string $id,
        public string $name,
        public string|null $color = null,
    ) {
    }
}
```

Q2: What command could you run to generate the missing grid?

Q2: What command could you run to generate the missing grid?

- `symfony console create:grid 'App\Model\Team'`

Q2: What command could you run to generate the missing grid?

- `symfony console create:grid 'App\Model\Team'`
- `symfony console make:resource 'App\Model\Team'`

Q2: What command could you run to generate the missing grid?

- `symfony console create:grid 'App\Model\Team'`
- `symfony console make:resource 'App\Model\Team'`
- `symfony console make:grid 'App\Model\Team'`

Q2: What command could you run to generate the missing grid?

- `symfony console create:grid 'App\Model\Team'`
- `symfony console make:resource 'App\Model\Team'`
- `symfony console make:grid 'App\Model\Team'`
- `symfony console make:grid 'App\Resource\TeamResource'`

Q2: What command could you run to generate the missing grid?

- `symfony console create:grid 'App\Model\Team'`
- `symfony console make:resource 'App\Model\Team'`
- `symfony console make:grid 'App\Model\Team'`
- `symfony console make:grid 'App\Resource\TeamResource'`

symfony console make:grid 'App\Model\Team'

Dashboard / Teams

Teams

Filters

▼

Show 10 ▾

NAME	COLOR	ACTIONS
Alfa Romeo	Red	 
AlphaTauri	Blue	 
Alpine	Blue	  Show drivers
Aston Martin	Green	 
Ferrari	Red	 
Haas F1 Team	Grey	 
McLaren	Orange	 
Mercedes	Green	 
Red Bull Racing	Blue	 
Williams	Cyan	 

Showing 1 to 10 of 12 entries

< Previous 1 2 Next >

Q2: How can you add a Team filter for the F1 drivers' grid?

```
->addFilter(Filter::create('teamName', TeamFilter::class))
```

Q2: How can you add a Team filter for the F1 drivers' grid?

```
->addFilter(Filter::create('teamName', TeamFilter::class))
```

```
#[AsFilter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)]  
final class TeamFilter implements FilterInterface
```

Q2: How can you add a Team filter for the F1 drivers' grid?

```
->addFilter(Filter::create('teamName', TeamFilter::class))
```

- ```
#[AsFilter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)]
final class TeamFilter implements FilterInterface
```
- ```
#[AsGridFilter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)  
final class TeamFilter implements FilterInterface
```

Q2: How can you add a Team filter for the F1 drivers' grid?

```
->addFilter(Filter::create('teamName', TeamFilter::class))
```

- ```
#[AsFilter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)]
final class TeamFilter implements FilterInterface
```
- ```
#[AsGridFilter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)  
final class TeamFilter implements FilterInterface
```
- ```
#[Filter(formType: TeamFilterType::class, template: '@SyliusBootstrapAdminUi/shared/grid/filter/select.html.twig',)]
final class TeamFilter implements FilterInterface
```

# Conclusion

Congratulations  
Pole position, baby!





Démo Open F1

## TODO - QRCode

<https://stack.sylius.com/> - Stack (docs)

<https://github.com/Sylius/Stack> - Stack

<https://github.com/Sylius/SyliusGridBundle> - Grid

<https://openf1.org/> - Open F1 API by br-g (Bruno Godefroy)

<https://github.com/Sylius/SyliusGridBundle/pull/371>

Demo Project =>

<https://github.com/loic425/openf1>