

The Ibexa logo is rendered in a lowercase, rounded, sans-serif font. The letters 'i', 'e', and 'a' are colored in a vibrant red, while the letters 'b', 'x', and 'o' are in a bright cyan. The logo is positioned in the upper left quadrant of the slide.

ibexa

Summit 25

What's new for Ibexa DXP developers in 2025?

Adam Wójs, Director of Engineering @ Ibexa

A large, stylized graphic on the right side of the slide. It features the year '2025' in a large, glowing cyan font. Below the year is a horizontal cyan bar that is partially filled, resembling a progress indicator. Underneath the bar, the word 'loading...' is written in a smaller, glowing cyan font. The background of this graphic is a dark, futuristic digital interface with various data points and lines, all in shades of cyan and blue. The graphic is set against a dark teal background that has a torn paper effect at the top and bottom edges, revealing a colorful mosaic pattern of hexagons in shades of red, orange, yellow, and cyan.

2025
loading...

\$ whoami

Adam Wójs



- Director of Engineering @ Ibexa
- > 14 years of experience
- Support > Engineering > Leadership

 /adamwojs

 /adamwojs

Agenda

- Key features
- Upgrade to Symfony 7
- New Extension Points & APIs
- Documentation

Key Features



AI



Collaboration



Discounts



Upgrade to Symfony 7

\$ Upgrade to Symfony 7

IN PROGRESS



Bump minimal PHP version to 8.3



Resolve Ibexa DXP deprecations



Resolve Symfony 5.4 deprecations (including security layer)



Bump Symfony version to 6.4

Resolve Symfony 6.x deprecations

Bump Symfony version to 7.x

\$ Upgrade to Symfony 7

IN PROGRESS

Admin > System information

System information

Product

Composer

Repository

Hardware

PHP

Symfony Kernel

Services

Symfony Kernel

Environment
dev

Debug mode
Enabled

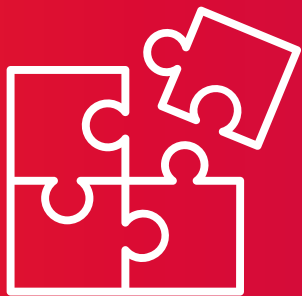
Version
6.4.17

Project directory
/var/www/html

Cache directory
/var/www/html/var/cache/dev

Log directory
/var/www/html/var/log

Character set
UTF-8



New Extension Points & APIs

\$ Sending notifications

ibexa/notification from now on allows you to send internal notifications.

<https://github.com/ibexa/notifications/pull/11>

\$ Sending notifications

```
29 $user = $userService->loadUserByLogin('john.smith@example.com')
30
31 $notification = new SystemNotification('Hello John!', ['ibexa']);
32 $notification->setIcon('bell');
33 $notification->setContent("It's works!");
34 $notification->setRoute(
35     new RouteReference(
36         'ibexa.content.view',
37         [
38             'contentId' => 52,
39         ]
40     )
41 );
42
43 $notifier->send($notification, new UserRecipient($user));
```

\$ Sending notifications (demo)

ibexo

Site: All context | AU

Notifications (19)



- | | | | |
|--|------------------|---|------------------------|
| | Hello World! | Greeting from the future! | January 28, 2025 11:45 |
| | Catalog is ready | Catalog "Product Catalog 2025" is ready to download | January 19, 2025 21:42 |
| | Catalog is ready | Catalog "Product Catalog 2025" is ready to download | January 19, 2025 21:36 |
| | Catalog is ready | Catalog "Product Catalog 2025" is ready to download | January 19, 2025 21:34 |
| | Catalog is ready | Catalog "Product Catalog 2025" is ready to download | January 19, 2025 21:34 |

Viewing 5 out of 19 items

< 1 2 3 4 >

Preview

Edit

Move



Relations



Hide

Description

You are now ready to start your project.

Landing page

To get preview, choose site context and go to "View" tab or click the "Preview" button.

\$ Custom notification

```
29 final class WelcomeNotification extends Notification implements
    SystemNotificationInterface
30 {
31     public function asSystemNotification(UserRecipientInterface $recipient, ?string
    $transport = null): ?SystemMessage
32     {
33         $message = new SystemMessage($recipient->getUser());
34         $message->setType('welcome');
35         $message->setContext([
36             'subject' => $this->getSubject(),
37             'content' => $this->getContent(),
38         ]);
39
40         return $message;
41     }
42 }
```

\$ Price stamps

“The idea is to have **composition by stamping PriceInterface** with dedicated implementations of StampInterface and performing price changes”

<https://github.com/ibexa/product-catalog/pull/1212>

\$ Price stamp (contract)

```
1 namespace Ibexa\Contracts\ProductCatalog\Values\Price;
2
3 use Ibexa\Contracts\ProductCatalog\Values\StampInterface;
4 use Money\Money;
5
6 interface PriceStampInterface extends StampInterface
7 {
8     public function getNewPrice(): Money;
9 }
```

\$ Price envelope (contract)



```
1 interface PriceEnvelopeInterface
2 {
3     /**
4      * Adds one or more stamps.
5      *
6      * @return static
7      */
8     public function with(PriceStampInterface ...$stamps): self;
9
10    # ...
11
12    public function last(string $stampFqcn): ?PriceStampInterface;
13
14    public function all(?string $stampFqcn = null): array;
15 }
16
```

\$ Currency exchange stamp

```
1 final class CurrencyExchangeStamp implements PriceStampInterface
2 {
3     public function __construct(
4         private Money $originalPrice,
5         private Money $convertedPrice,
6         private float $rate,
7         private DateTimeInterface $timestamp = new DateTimeImmutable()
8     ) {}
9
10    public function getNewPrice(): Money
11    {
12        return $this->convertedPrice;
13    }
14
15    # ...
16 }
```


\$ Currency exchange stamp

```
1 final class CurrencyExchangePriceResolver implements PriceResolverInterface
2 {
3     # ...
4     private function convertPrice(PriceInterface&PriceEnvelopeInterface $price, CurrencyInterface $currency):
5         PriceInterface
6     {
7         $sourcePrice = $price->getMoney();
8         $targetPrice = $this->converter->convert($sourcePrice, new Currency($currency->getCode()));
9         $rate = $this->getExchangeRate($price->getCurrency(), $currency);
10
11         return $price->with(new CurrencyExchangeStamp($sourcePrice, $targetPrice, $rate));
12 }
```

\$ Currency exchange stamp (demo)



```
demo@ibexa-summit-2025 % ddev php bin/console demo:price-stamp:currency-exchange PIM-25396 --currency=EUR  
Target price: € 24.00
```

```
demo@ibexa-summit-2025 % ddev php bin/console demo:price-stamp:currency-exchange PIM-25396 --currency=PLN  
Target price: PLN 101.04  
Source price: € 24.00  
Exchange rate: 4.210000  
Exchange date: 2025-01-28 12:44:00
```

```
demo@ibexa-summit-2025 % ddev php bin/console demo:price-stamp:currency-exchange PIM-25396 --currency=USD  
Target price: $ 30.00  
Source price: € 24.00  
Exchange rate: 1.250000  
Exchange date: 2025-01-28 12:45:00
```

\$ Symbol attribute

IN PROGRESS

- Symbol attribute type allows to efficiently represent string-based values and enforcing its format in product specifications.
- Example use cases: ISBN, EAN13, Manufacturer Part Number
- Ibexa DXP v4.6 LTS update

<https://github.com/ibexa/product-catalog-symbol-attribute>

\$ Custom format

IN PROGRESS

You can define custom symbol attribute formats using YAML configuration:

```
1 ibexa_product_catalog_symbol_attribute:  
2   formats:  
3     manufacturer_part_number:  
4       name: 'Manufacturer Part Number'  
5       pattern: '/^[A-Z]{3}-\d{5}$/'  
6       examples:  
7         - 'RPI-14645'  
8         - 'MSS-24827'  
9         - 'SEE-15444'
```

\$ Custom format

IN PROGRESS

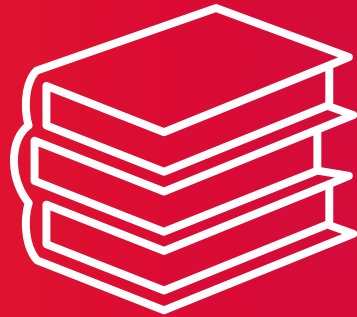
You can define checksum for format by implementing ChecksumInterface:

```
1 namespace Ibexa\Contracts\ProductCatalogSymbolAttribute\Value;
2
3 use Ibexa\Contracts\ProductCatalog\Values\AttributeDefinitionInterface;
4
5 interface ChecksumInterface
6 {
7     public function validate(AttributeDefinitionInterface $attributeDefinition, string $value): bool;
8 }
```

\$ Custom format (demo)

IN PROGRESS

```
1 final class ISBN13Checksum implements ChecksumInterface
2 {
3     public function validate(AttributeDefinitionInterface $attributeDefinition, string $value): bool
4     {
5         $digits = $this->getDigits($value);
6         if (count($digits) !== 13) {
7             return false;
8         }
9
10        $checksum = 0;
11        foreach ($digits as $i => $digit) {
12            $checksum += $i % 2 === 0 ? $digit : 3 * $digit;
13        }
14
15        return $checksum % 10 === 0;
16    }
17
18    # ...
19 }
```



Documentation

PHP API Reference

The screenshot shows a web browser displaying the Ibeta PHP API Reference. The page is titled "PaymentMethodServiceInterface" and is categorized under "Commerce". It provides details about the interface, including its purpose, methods, and parameters.

Navigation: The left sidebar shows a tree view of the API reference, with "Payment" selected. The top navigation bar includes "Developer Documentation", "Change version", a search bar, and a "View on GitHub" link.

Page Content:

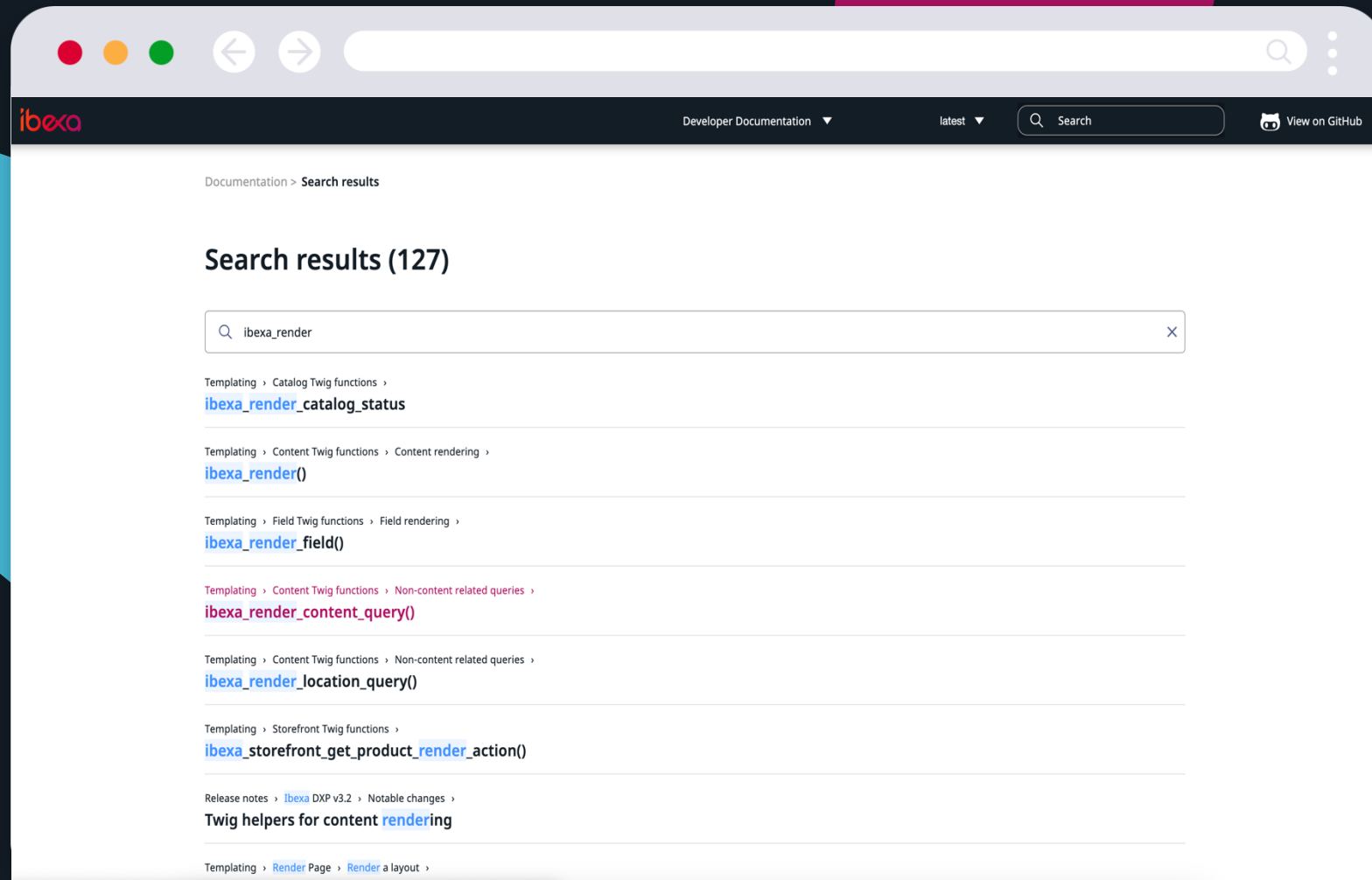
- Breadcrumb:** Ibeta > Contracts > Payment > PaymentMethodServiceInterface
- Category:** Commerce
- Section:** PaymentMethodServiceInterface (with a "Copy FQCN" button)
- File:** PaymentMethodServiceInterface.php : 20
- Interface:** Service for managing payment methods.
- Methods:**
 - createPaymentMethod()** (with a "Copy FQCN" button): Creates a new payment method. The signature is: `public createPaymentMethod(PaymentMethodCreateStruct $createStruct) : PaymentMethodInterface`
- Parameters:**

Name	Type	Default value	Description
\$createStruct	PaymentMethodCreateStruct	-	Struct with data needed to create a new payment method.
- Return values:** [PaymentMethodInterface](#)
- Tags:** Throws

Right Sidebar: A summary of the **PaymentMethodServiceInterface** methods:

- createPaymentMethod()
- deletePaymentMethod()
- findPaymentMethods()
- getPaymentMethod()
- getPaymentMethodByIdentifier()
- isPaymentMethodUsed()
- updatePaymentMethod()

Search results page



The screenshot shows a web browser window displaying the search results page for 'ibexa_render'. The browser's address bar is empty, and the page title is 'Search results (127)'. The search bar contains the text 'ibexa_render'. The results are listed as follows:

- Documentation > Search results
- Search results (127)
- Search bar:
- Templating > Catalog Twig functions > [ibexa_render_catalog_status](#)
- Templating > Content Twig functions > Content rendering > [ibexa_render\(\)](#)
- Templating > Field Twig functions > Field rendering > [ibexa_render_field\(\)](#)
- Templating > Content Twig functions > Non-content related queries > [ibexa_render_content_query\(\)](#)
- Templating > Content Twig functions > Non-content related queries > [ibexa_render_location_query\(\)](#)
- Templating > Storefront Twig functions > [ibexa_storefront_get_product_render_action\(\)](#)
- Release notes > [ibexa DXP v3.2](#) > Notable changes > [Twig helpers for content rendering](#)
- Templating > [Render Page](#) > [Render a layout](#)



Thank you!