Валидатор и формы

№ урока:	8	**Курс:**	Веб разработка на PHP Symfony
Средства обучения:	Eclipse PHP Development Tools, командная строка, любой современный браузер		

Обзор, цель и назначение урока

В данном уроке рассматривается то, как можно использовать валидацию данных в приложении и то, как можно использовать компонент, который позволяет работать с формами в самом приложении, т.е создать их и использовать для того, чтобы отправлять данные.

Изучив материал данного занятия, учащийся сможет:

- Использовать компонент Validation в Symfony для того, чтобы уметь валидировать любые данные, которые нужно будет проверить на нужный тип.
- Валидировать Entity в приложении с помощью компонента Validation и подбирать отдельные типы валидации к каждому полю.
- Использовать компонент Forms в приложении для того, чтобы создавать полноценные формы, обрабатывать их данные и в итоге настраивать их так как нужно.
- Создавать валидацию данных для самых полей формы при их отправке.

Содержание урока

1. Описание компонента Validation
2. Валидация скалярных данных
3. Валидация сложных типов данных
4. Компонент Forms
5. Создание форм
6. Процессинг форм
7. Валидация форм

Резюме

- Validation представляет из себя полноценный компонент в Symfony который используется для того, чтобы делать валидацию данных в Entity, контроллерах и формах приложения. Это обеспечивает дополнительную безопасность и если сделать качественную оптимизацию - то в итоге это не позволит злоумышленникам как-то навредить приложению.
- Чтобы установить Validation компонент, нужно воспользоваться командой:
  composer require symfony/validator doctrine/annotations
  Вся настройка данного компонента делается в файле config/framework.yaml в параметре validation.
- Чтобы начать использовать компонент валидации в Entity приложения, нужно подключить класс Assert, который и предоставляет для использования другие подклассы, с помощью которых можно уже для отдельных полей приложения указывать какая валидация должна быть на то или иное поле. Класс для подключения: use Symfony\Component\Validator\Constraints as Assert.
- После того как класс самого валидатора подключен и в доступе теперь есть все другие классы для использования валидации полей, можно приступать к настройке типов валидации полей. Например, указав в аннотациях @Assert/NotBlank перед каким-то

CyberBionic systematics

CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine

Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

полем, мы укажем что для данного поля мы хотим использовать валидацию из класса NotBlank, который проверяет является ли поле пустым или нет. Все типы, которые доступны для валидации, можно найти в официальной документации Symfony.

- Компонент Validation можно также использовать и в самих контроллерах для того, чтобы производить валидацию входных данных. Для этого нужно подключить сначала интерфейс класса валидатора и уже потом начать делать валидацию данных которые нас интересуют.
- Компонент Forms представляет из себя очень удобный инструмент для того, чтобы быстро разрабатывать формы в приложении, и, кроме этого, делать обработку самих форм не так уже и сложно. Перед тем как начать использовать данный компонент в коде приложения, нужно убедиться, что он уже установлен. Поэтому для установки нужно воспользоваться командой:
  composer require symfony/form.
- Компонент Forms позволяет динамически создавать формы через PHP код прямо в контроллерах, так и в отдельных файлах и привязывать к полям определенных Entity. В случае если нужно создать форму в отдельном файле, то следует воспользоваться командой
  php bin/console make:form
  В дальнейшем с помощью всех функций, которые показаны в видео уроке, можно будет сделать полноценное построение форм.

Закрепление материала

- Перед тем как начать использовать компонент Validation в Entity, какой класс нужно подключить, который в итоге предоставляет другие классы с типами валидации для всех полей?
- Как можно использовать валидацию определенного типа указывая в аннотации поля?
- Каким образом происходит валидация данных прямо внутри контроллера?
- Как можно сделать валидацию данных в контроллере целого массива данных и как вывести ошибки?
- Как создается форма в контроллере, её поля и как её вывести в шаблон?
- Какие функции следует использовать для того, чтобы обработать саму форму прямо в контроллере?
- В случае если мы не сделали валидацию полей самого Entity, то как можно сделать валидацию полей формы?

Дополнительное задание

Задание
Организуйте валидацию полей к Entity Product, выберите несложные типы и протестировать как оно работает в самом Entity. В дальнейшем после этого сделайте валидацию произвольного массива данных из 1-2. Под массивами данных выбирайте при этом соответствующие типы из документации. После этого создайте свою первую форму и выведите в шаблон и сделайте её обработку, при этом привязав к Entity Product и посмотрите, как оно в итоге сохраняет данные в таблицу product. После этого реализуйте валидацию к данным полям в зависимости от типа и задать ограничения на сами поля.

Самостоятельная деятельность учащегося

Задание 1
Выберите любой тип поля валидации скалярных данных из официальной документации Symfony по компоненту Validation и примените к Entity. То есть подоберите к полям Entity правильные

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

типы валидации и сделайте их динамичными. К примеру, если имеется поля phone - то сделать валидацию как для номера телефона.

Задание 2
Создайте массив данных, который будет иметь в себе такую структуру:

```
'name' => 'Testing',
'description' => [
        'desc1' => 356,
        'desc2' => 'Super description',
],
'phone' => '383222931',
'age' => 55,
'tags' => [
        'music',
        'art',
        'tech'
]
```

После создания данного массива, сделайте его валидацию, подбирая правильные типы всех полей правильно и также сделайте вывод ошибок в шаблоне, в случае если неправильно где-то сработает валидация.

Задание 3
Создайте форму с такими полями как title, description, author, price и привяжите к отдельному Entity. Задавайте типы полей в зависимости от того какой тип данных поля. После этого выведите в шаблон и реализуйте отправку формы и сохранение данных в отдельный Entity, который имеет такие же поля.

Задание 4
Возьмите предыдущую форму и добавьте валидацию ко всем полям в зависимости от полей формы. Валидация должна быть сделана в самом коде формы. После этого протестируйте как работает с неправильными и правильными данными.

Рекомендуемые ресурсы

Официальная документация Symfony
https://symfony.com/doc/current/index.html

Документация Symfony на рус. языке (имеет в себе некоторые устаревшие материалы)
https://symfony.com.ua/doc/current/index.html

CyberBionic systematics
CyberBionic Systematics ®
19 Eugene Sverstyuk Str., 5 floor
Kyiv, Ukraine
Tel. 0 800 750 312
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com