Event Listener

№ урока:	9	**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 идёт рассмотрение системы событий в Symfony и таких компонентов данной системы как Event Listener и Event Subscriber. Использование данных компонентов позволяет отслеживать системные события приложения, которые происходят в самом фреймворке.

Изучив материал данного занятия, учащийся сможет:

- Понимать устройство событий в Symfony приложении.
- Уметь использовать Event Listener и Event Subscriber для отслеживания событий, которые происходят внутри самого фреймворка во время работы приложения.
- Обрабатывать события приложения.

Содержание урока

1. Понятие Event
2. Разница Event Listener и Event Subscriber
3. Пример использования
4. Дебаг ивентов
5. Приоритет

Резюме

- В любом Symfony приложении уже по умолчанию есть созданная система отслеживания событий, благодаря ней можно узнавать, как те или иные компоненты обрабатывают данные и анализировать их при потребности. К примеру есть kernel.exception - это события, которые отвечают за обработку ошибок, воспользовавшись Event Listener можно сделать отслеживание событий ошибок, которые происходят внутри.
- Есть два типа классов, которые можно использовать для отслеживания событий в приложении, это Event Listener и Event Subscriber. Event Listener может отслеживать целое одно событие и получать от него данные. Но в случае с Event Subscriber - данный класс уже понимает какие именно события он обрабатывает и какого именно типа, благодаря чему можно сделать полноценное отслеживание событий в определенные моменты.
- Все классы типа Event Listener создаются в папке src/EventListener, а классы типа Event Subscriber - создаются в папке src/EventSubscriber.
- После того как классы были добавлены в папки и уже имеют код, который позволяет обрабатывать ошибки, нужно зарегистрировать данные классы в файле конфигурации config/services.yaml в качестве сервисов, при этом нужно указать через теги какие именно события они должны обрабатывать.
- Чтобы проверить какие классы типов Event Listener или Event Subscriber были зарегистрированы в системе - нужно воспользоваться командой: php bin/console debug:event-dispatcher. Если же нам нужно узнать какие классы именно таких двух типов зарегистрированы к какому-то отдельному событию в приложении - нужно воспользоваться предыдущей командой, только уже в конце указать тег события, который нас интересует: php bin/console debug:event-dispatcher kernel.exception.

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

- При вызове команд на просмотр того, какие классы для отслеживания событий зарегистрированы в системе - можно увидеть то, как данные команды выводят приоритеты всех классов и функций, которые обрабатывают события в системе. Число приоритета указывает какую функцию или класс вызывать в первую очередь. Т.е. чем выше приоритет был указан - тем быстрее данная функция будет вызвана.

Закрепление материала

- Какая разница между Event Listener и Event Subscriber?
- Как работает Event Listener и какая структура класса?
- Какая структура класса Event Subscriber и как он позволяет делать обработку нескольких событий через свои функции?
- Как нужно правильно регистрировать в config/services.yaml классы Event Listener и Event Subscriber?
- Для чего нужны приоритеты в функциях Event Subscriber?
- С помощью какой команды можно посмотреть какие обработчики событий были зарегистрированы в приложении?

Дополнительное задание

Задание
Сначала создаём Event Listener, который будет обрабатывать все ошибки приложения через kernel.exception. Event Listener будет выводить только код ошибки и его сообщение. После этого регистрируем данный класс в config/services.yaml, указывая при этом, что этот класс обрабатывает именно ошибки в системе через тег kernel.exception. Переходим на несуществующий маршрут и смотрим то, как созданный нами Event Listener выводит сообщение об ошибке. После этого переходим к созданию нового Event Listener, для этого создаём файл класса в src/EventSubscriber и наполняем его соответствующим кодом на обработку трёх типов событий от kernel.exception. Когда класс готов уже к использованию, можем его также зарегистрировать в config/services.yaml, но при этом удалить регистрацию Event Listener, который был создан. В дальнейшем можно вызвать любую ошибку и посмотреть как Event Subscriber будет дополнительно обрабатывать ошибки кроме обработчика ошибок, который работает по умолчанию.

Самостоятельная деятельность учащегося

Задание 1
Создать Event Listener на kernel.view, который будет выводить информацию ответа о шаблоне вывода. https://symfony.com/doc/current/reference/events.html#kernel-view - здесь показано то, какие функции данного события есть. После этого зарегистрировать данный класс в config/services.yaml, правильно при этом указав теги событий, которые он обрабатывает.

Задание 2
Создать Event Subscriber для событий от kernel.response. В данном случае обработчик событий будет выводить информацию об ответах запроса. Узнать информацию о функциях для работы с событиями данного типа можно перейдя по следующей ссылке https://symfony.com/doc/current/reference/events.html#kernel-response. Желательно добавить больше одной функции для обработки события в Event Subscriber. После того как класс будет готов к использованию - зарегистрировать его в config/services.yaml, указав правильно теги событий, которые он должен обрабатывать.

Задание 3

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

Выбрать из документации любое событие от kernel и написать простой Event Subscriber, который будет выводить базовую информацию о текущем событии. После этого зарегистрировать в config/services.yaml и поставить для функции обработки как можно более низкий приоритет, после чего, вызвав event-dispatcher в консоли, посмотреть будет ли вызываться функция в последний момент.

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 ®
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