Универсальный модуль авторизации пользователей на базе PHP-фреймворка Symfony

Хоружий Андрей Андреевич

УО "Брестский государственный университет имени А.С. Пушкина"

При проектировании и создании веб-приложений практически всегда возникает вопрос, каким образом реализовать ограничение (контроль доступа) пользователей веб-приложений в зависимости от их роли в системе, прав или привилегий. Даже в наипростейшем случае (форум или блог) следует запретить пользователям редактирование/удаление чужих сообщений, модераторам – разграничить модерируемые разделы и т.д.

В популярных фреймворках, предназначенных для разработки веб-приложений на языке PHP, уже имеются довольно развитые механизмы для обеспечения задач авторизации в веб-приложении. Так, например, во фреймворке Laravel [1] логика авторизации пользователей размещается в классах-политиках, вызов указанных правил авторизации осуществляется из контроллера либо представления (шаблона веб-страницы). Фреймворк Symfony [2] поддерживает два механизма: списки контроля доступа (Access Control Lists) [3] и систему «избирателей» (voters) [4]. Механизм ACL достаточно сложен и гибок, поскольку позволяет хранить сведения о правах доступа к каждому объекту (к каждой записи в каждой таблице базы данных) по отдельности. Механизм «избирателей» более прост: решение о разрешении или запрете действия пользователя над объектом веб-приложения принимается после прохождения цепочки классов-«избирателей», каждый из который может либо проигнорировать запрос, либо проголосовать за разрешение доступа, либо за запрет.

Во всех случаях реализация механизма авторизации для нужд конкретного веб-приложения требует от программистов написания и тестирования значительного объема кода на PHP, шаблонов веб-страниц и др. Возникает интерес в создании универсальных модулей, одинаково применимых для нужд различных веб-приложений. Целью настоящей работы является создание универсального модуля авторизации пользователей для веб-приложений на базе фреймворка Symfony. Перечислим основные требования к создаваемому модулю:

- настройка правил авторизации в конкретном веб-приложении должна осуществляться через интерфейс веб-приложения пользователем с правами администратора, а не разработчиками путем редактирования PHP-кода;
- универсальность как самих правил авторизации, так и интерфейса модуля, позволяющая подключать его ко многим веб-приложениям без значительных доработок.

В соответствии с общими рекомендациями, принятыми разработчиками фреймворка Symfony, модуль авторизации реализован в виде разделяемого бандла (reusable bundle). Все классы размещены в пространстве имен triguk/AuthorizationBundle.

Основная идея, заложенная в модуле авторизации, может быть сформулирована следующим образом. Любому пользователю может быть сопоставлена одна или несколько ролей. Любой **роли** может быть выдано одно или несколько разрешений на выполнение **действий** (чтение, изменение, удаление, просмотр всего списка и т.д.;) с различными **классами** объектов веб-приложения (соответствуют классам-сущностям и таблицам базы данных) в том или ином **масштабе** (все объекты, часть объектов или только те, владельцем которых является пользователь). Примеры:

- роли **admin** может быть выдано разрешение на действие **delete** с классом **AppBundle:User** в масштабе **all** (что означает, что все администраторы могут удалять записи о любых пользователях);
- роли **moderator** может быть выдано разрешение на действие **edit** с классом **AppBundle:ForumPost** в масштабе **group** (что означает, что все модераторы могут редактировать сообщения в тех разделах форума, которые им назначены);
- роли **user** может быть выдано разрешение на действие **edit** с классом **AppBundle:ForumPost** в масштабе **owner** (что означает, что все пользователи могут редактировать свои собственные сообщения).

Программный код обеспечивает хранение перечней ролей (класс triguk\AuthorizationBundle\Entity\AuthRole), действий (класс triguk\AuthorizationBundle\Entity\AuthPermission), классов объектов (triguk\AuthorizationBundle\Entity\AuthObject), масштабов (triguk\AuthorizationBundle\Entity\AuthObject). Каждый из перечисленных классов благодаря ORM-системе Doctrine предназначен для хранения данных в своей таблице базы данных. Класс triguk\AuthorizationBundle\Entity\AuthGrant предназначен для хранения сведений о выдаче тех или иных разрешений (с указанием роли, действия, класса объектов и масштаба). Соответствующая ему таблица базы данных состоит из четырех полей (все являются внешними ключами), содержащими ссылки на соответствующие таблицы из перечисленных ранее.

Хранение и обновление записей осуществляется с помощью классов-репозиториев, причем репозитории для ролей, действий, классов объектов и масштабов наследуют класс \Doctrine\ORM\EntityRepository без дополнений, в то время как класс-репозиторий triguk\AuthorizationBundle\Repository\AuthGrantRepository содержит ряд сервисных функций, связанных с хранением данных (поиск, очистка и синхронизация разрешений).

Реализация авторизационной логики базируется на использовании всего одного класса-«избирателя» со следующим алгоритмом:

- получить у веб-приложения сведения о пользователе, запрашиваемом объекте, действии;
- получить сведения о всех ролях, сопоставленных пользователю;
- получить из репозитория список выданных разрешений для заданных ролей, объекта и действия;
- если разрешений нет, вернуть результат «отказ»;
- если есть разрешение с масштабом **all**, вернуть результат «разрешено»;
- если есть разрешение с масштабом **owner**, определить владельца объекта, вернуть результат «разрешено» если пользователь является владельцем объекта, иначе вернуть результат «отказ»;
- если есть разрешение с масштабом **group**, определить владельца объекта и выяснить, входит ли владелец объекта и пользователь, получающий разрешение, в одну группу; вернуть результат «разрешено» в случае успеха, иначе вернуть результат «отказ».

Редактирование правил авторизации осуществляется непосредственно из веб-приложения пользователем-администратором. С этой целью в модуле реализованы классы контроллеров и форм, соответствующие twig-шаблоны. Маршруты реализованы в рамках префикса /admin/, т.е. полный путь может иметь вид http://website.com/admin/authgrant/.

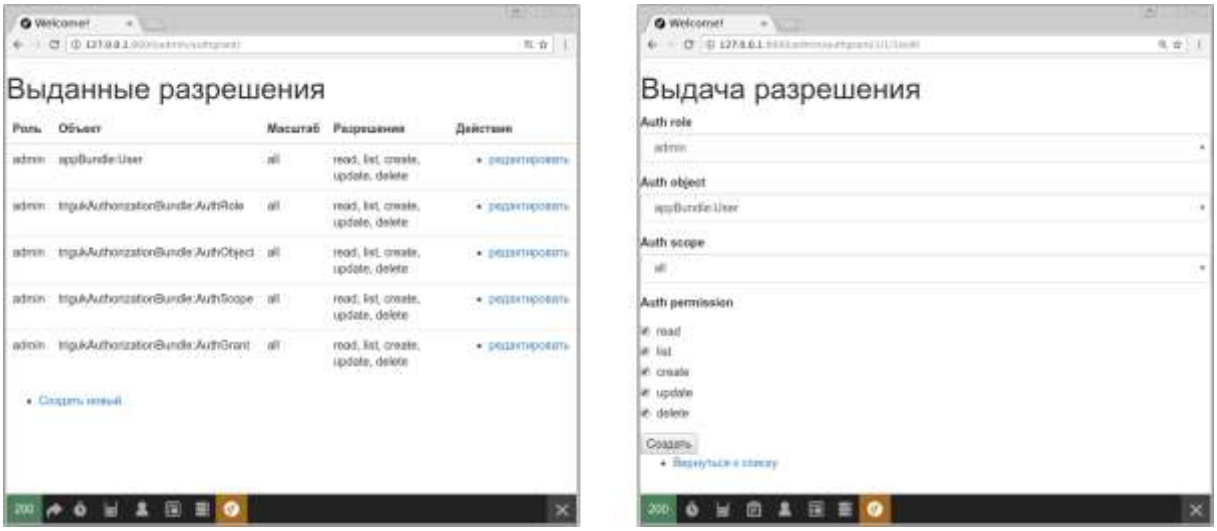

Рисунок 1 – Интерфейс редактора правил авторизации (слева – список всех выданных разешений, справа – редактирование разрешений для роли).

Адаптация модуля авторизации под нужды конкретного веб-приложения требует от разработчиков реализации в классах-сущностях методов getOwner() и getGroup(). Общий интерфейс здесь не предусмотрен, его реализация не требуется, а значит, реализация каждого метода является необязательной и требуется лишь там, где требуется применение правил авторизации.

Созданный нами модуль (bundle) является разделяемым, с возможностью повторного использования, пригоден к использованию в несложных веб-приложениях, не требующих сложных правил авторизации (ограничений доступа пользователей к ресурсам). Настройка правил авторизации выполняется пользователем непосредственно через интерфейс веб-приложения. Адаптация веб-приложения требует лишь незначительной правки конфигурационных файлов, а также дополнения некоторых классов-сущностей методами, обеспечивающими определение владельца того или иного объекта, группу того или иного пользователя.

СПИСОК ЛИТЕРАТУРЫ

1. Laravel - The PHP Framework For Web Artisans [Electronic resource]. – Mode of access: https://laravel.com/. – Date of access: 20.12.2016.

2. Symfony, High Performance PHP Framework for Web Development [Electronic resource]. – Mode of access: https://symfony.com/. – Date of access: 20.12.2016.

3. How to Use Access Control Lists (ACLs) [Electronic resource] // Symfony, High Performance PHP Framework for Web Development. – Mode of access: http://symfony.com/doc/current/security/acl.html. – Date of access: 24.12.2016.

4. How to Use Voters to Check User Permissions [Electronic resource] // Symfony, High Performance PHP Framework for Web Development. – Mode of access: http://symfony.com/doc/current/security/voters.html. – Date of access: 24.12.2016.

5. The Bundle System [Electronic resource] // Symfony, High Performance PHP Framework for Web Development. – Mode of access: http://symfony.com/doc/current/bundles.html. – Date of access: 24.12.2016.