Гибкая модульная архитектура SaaS платформ на Yii2

Виктор Пикаев
aka @HaruAtari

Исходный код

Http://haru-atari.com/redirect/devconf16

Вступление, вызывающее интерес

Требования к архитектуре

- Самодостаточные модули.
- Версионирование модулей.
- Список подключаемых модулей (и их версий) для каждого клиента свой.
- Возможность динамически менять список активных модулей (и их версий).

Содержание

- Организация модулей и их подключение
- Взаимодействие с модулями
- Работа с базой данных
- Миграции
- Выбор оптимального уровня изоляции модулей

Организация модулей и их подключение

Каждый модуль состоит из двух частей:

1. **Главный модуль** (унаследован от `MainModule`). Содержит интерфейс взаимодействия с активной версией.
2. **Версии** (унаследованы от `VersionModule`) - подмодули, которые реализуют логику работы.

Организация модулей и их подключение

```
app
+-modules
  +-mod_a
    +-ModA (extends MainModule)
    +-modules
      +-v1
      | +-V1 (extends VersionModule)
      +-v2
        +-V2 (extends VersionModule)
```

Организация модулей и их подключение

Список модулей и их иерархия хранятся в базе данных.

Для каждого модуля указывается:

- Список версий (с указанием активной).
- Его id в системе.
- Названия его класса и классов версий.

Организация модулей и их подключение

- Поведение `ApplicationInitializerBehavior` при инициализации приложения собирает дерево модулей и подключает их.
- Для каждой активной версии запускается статическая инициализация (устанавливает правила роутинга и обработчики событий).
- Активная версия каждого модуля подключается под id главного модуля.

Организация модулей и их подключение

```
app
+-modules
  +-mod_a
    +-ModA (extends MainModule)
    +-modules
      +-v1
      | +-V1 (extends VersionModule)
      +-v2
        +-V2 (extends VersionModule)

\Yii::$app->getModeul('mod_a');
// app\modules\mod_a\modules\v1\V1
```

Взаимодействие с модулями

- Взаимодействие с модулями осуществляется через стандартный механизм событий Yii2.
- События обрабатываются на уровне классов главных модулей.
- Для удобства добавлены обертки для работы с событиями.
- Константы с именами событий располагаются в главных модулях.
- Классы событий могут располагаться где угодно.

```
ModA::addEventListener(ModA::SAMPLE_EVENT, $handler);

ModA::triggerEvent(ModA::SAMPLE_EVENT, new SampleEvent());
```

Взаимодействие с модулями

Для доступа к классам в версиях используются фасады из главных модулей `MainModule::getClass()` и `MainModule::getObject()`. Фасады ищут нужный класс в активной версии модуля.

```
ModA::getClass('models\SampleModel');
// \app\modules\mod_a\modules\v1\models\SampleModel

ModA::getObject('models\SampleModel', ['constructorArgs']);
// \app\modules\mod_a\modules\v1\models\SampleModel
```

Работа с базой данных

- Все модульные модели должны использовать трейт `ModuleActiveRecordTrait`.
- Трейт переопределяет метод `tableName()`.
- Имена таблиц содержат префикс в виде полного id модуля и id текущей (не активной) версии.
- Таким образом каждая версия работает со своими таблицами и при смене версии данные не теряются.

Работа с базой данных

```
$model = ModA::getClass('models\SampleModel');
$model::tableName();
// mod_a_v1_sample_model
// ===== -- **********

$model = ModB::getClass('models\SampleModel');
$model::tableName();
// mod_a_mod_b_v1_sample_model
// ===== ===== -- **********
```

Миграции

- Миграции делятся на миграции ядра и модульные.
- Миграциям добавлены пространства имен.
- Модульные миграции хранятся в версиях.
- Таблица `migration` для каждой версии своя.
- Для работы с модульными миграциями используются параметры `--moduleId` и `--versionId`.

Миграции

```
./yii migrate/create sample
app/migrations/m***_sample

./yii migrate/create sample --moduleId=mod_a --versionId=v2
app/modules/mod_a/modules/v2/migrations/m***_sample

./yii migrate/create sample --moduleId=mod_a
app/modules/mod_a/modules/v1/migrations/m***_sample

./yii migrate/create sample --moduleId=mod_a/mod_b
app/modules/mod_a/modules/mod_b/modules/v1/migrations/m***_sample
```

Выбор оптимального уровня изоляции модулей

- Работа с событиями через класс `Event`.
- Имена событий указываются строками.
- Фасады `getClass()` и `getObject()` находятся в ядре и получают id модуля через аргументы.

Вопросы?

Http://haru-atari.com
HaruAtari@gmail.com
@HaruAtari