Руководство

по модулю

ЭКСПЕРТИЗА ПРОИЗВОДИТЕЛЬНОСТИ

для разработчиков

Тел.: +7 (495) 008-84-52 +7 (927) 220-35-95 Электронная почта: info@acrit.ru

Введение

Модуль «Экспертиза производительности» предназначен для быстрого анализа проблем производительности сайта, а также для организации регулярных автоматических проверок с целью оперативного реагирования на возникающие проблем.

Модуль построен на принципе плагинов (один плагин это одна сфера проверки сайта, т.е. один тест), благодаря чему на каждом проекте можно не только пользоваться встроенными плагинами, но и разработать собственные, если встроенные решают не все задачи.

Модуль использует ядро D7, работает на различных версиях PHP.

Общее устройство и работа модуля

Принцип устройства модуля прост: он состоит из ядра модуля и подключаемых плагинов-тестов (хранятся в папке tests модуля).

Ядро модуля – это библиотека собственных классов (lib), классы сторонних разработчиков (classes), также административные файлы Битрикс (admin), папка со скриптами запуска тестов из командной строки (cron), папка с материалами для генерации PDF-отчета (pdf), языковые файлы модуля (lang), установочные файлы (install), а также файл include.php.

Подключаемые плагины-тесты хранятся в папке tests. Каждый плагин находится в собственной папке (имя папки ни на что не влияет), в которой обязательно должен быть файл class.php – это основной файл плагина.

Запуск тестов происходит четырьмя способами:

1. ручной запуск каждого теста по отдельности,
2. ручной запуск всех тестов сразу,
3. ручной запуск всех тестов из командной строки сервера (используя SSH),
4. автоматический запуск всех тестов из планировщика Cron.

При запуске тестов из административного раздела они выполняются пошагово, при запуске из командной строки или планировщиком – за один шаг.

Профили

Профиль это одно задание для автоматической проверки, содержащее как минимум одну проверку (может содержать сразу все имеющиеся проверки).

Для сохранения в профиль необходимо на административной странице модуля оставить отмеченными галочки только для нужных тестов и нажать кнопку «Автозапуск проверки» - «Сохранить текущий выбор в профиль». В открывшемся окне необходимо указать название и описание профиля (описание не обязательно, но советуем его заполнять, чтобы другие администраторы понимали для чего профиль нужен, и кто его создал), а также email, на который будет отправляться сообщение с PDF-отчетом. На вкладке «Тесты» можно скорректировать набор тестов профиля. После сохранения он будет доступен в общем списке профилей.

Общий список профилей открывается через нажатие кнопки «Автозапуск проверки» - «Список профилей». Отсюда можно каждый профиль открыть для просмотра и редактирования, а также настроить на автоматический запуск (если на сервере имеется такая возможность). Также профиль можно деактивировать, что будет означать, что автоматическая проверка по нему не будет работать, даже если настроена.

Структура файлов теста

Тест – это подключаемый файл class.php, располагающийся в собственной директории внутри папки test. Вся логика теста находится именно в class.php.

Дополнительно в папке теста могут лежать и другие необязательные файлы: .detail_text.php (подробное описание теста), .default_option.php (дефолтные опции конфигурации), .form.php (форма настроек теста). Если тест использует дополнительные классы, они должны находиться в папке classes, остальные php-файлы должны размещаться в папке include. Если тест использует дополнительные таблицы базы данных, они должны располагаться в файлах db/mysql/install.sql и db/mysql/uninstall.sql.

Папка lang содержит языковые файлы теста. Для файлов .detail_text.php и .form.php, class.php языковые файлы подключаются автоматически.

Структура файла class.php

Файл class.php должен содержать класс, реализовывающий абстрактный класс \Acrit\Examination\Test. Класс должен иметь константу GROUP, содержащую код одной из группы тестов:

- Test::GROUP_BASE
- Test::GROUP_PERFOMANCE
- Test::GROUP_SECURITY
- Test::GROUP_MISC

Необязательная константа SITE_DEPENDENT (boolean) позволяет отметить тест независимым от конкретного сайта.

Ниже приведена информация по методам класса (символом звездочки отмечены обязательные методы).

public function __construct() –метод для создания объекта.

public static function getName()* – метод должен возвращать имя теста.

public static function getPreviewText()* – метод должен возвращать краткое описание теста (подробное описание теста содержится в файле .detail_text.php).

protected static function installSql() – метод для установки кастомных таблиц в базе данных. Автоматически не выполняется. Удобно его вызывать в методе before.

protected static function uninstallSql() – удаление кастомных таблиц. Автоматически не выполняется.

public function before() – метод, выполняющийся перед каждым стартом теста (выполняется один раз перед первым шагом).

public function after() – метод, выполняющийся после окончания теста.

public function execute()* - метод, выполняющий процесс проверки, основной метод класса.

protected function processResults()* - метод для обработки полученных данных, генерации и сохранения отчета, вывода результатов процесса.

Принцип работы теста

В методах execute() и processResults() доступны переменные класса $this->arState и $this->arTestResult. Первое используется для сохранения собранных в ходе проверки данных (эти данные в дальнейшем будут обрабатываться методом processResults), второе – для управления результатами.

В процессе работы проверки метод execute выполняется один или более раз, при этом сам метод определяет, когда следует закончить проверку, что управляется ключом Status массива $this->arTestResult.

Тест должен выполниться еще раз (задается в методе execute):

```
$this->arTestResult['Status'] = static::STATUS_BREAK;
```

Тест успешно завершен, ошибок нет:

```
$this->arTestResult['Status'] = static::STATUS_SUCCESS;
```

Тест завершен, но есть ошибки:

```
$this->arTestResult['Status'] = static::STATUS_ERROR;
```

Если необходимо вывести progressbar:

```
$this->arTestResult['Progress'] = 0;
```

Если процесс пошаговый, то желательно как-то уведомлять пользователя о том, что процесс выполняется:

```
$this->arTestResult['TextProgress'] = static::getMessage('PROCESS_STARTED');
```

Также доступны для управления следующие ключи $this->arTestResult:

- Errors (массив ошибок),
- Warnings (массив предпреждений),
- DetailsSuccess (массив доп. сообщений при успешном завершении),
- DetailsWarnings (массив доп. сообщений при предупреждении),
- DetailsErrors (массив доп. сообщений при ошибке),

Когда тест завершен, из метода execute должен вызываться метод processResults, который обрабатывает полученные результаты и формирует отчет и сообщение.

Дополнительные возможности

Для получения языковых фраз удобно использовать не Loc::getMessage(), а static::getMessage(), это полный аналог, но помимо стандартного функционала

этот метод умеет получать «локальные» языковые фразы данного теста, благодаря чему запись становится короче, например, эти записи идентичны:

```
Loc::getMessage('ACRIT_EXAMINATION_TEST_PERFMON_ERROR_NOT_INSTALLED');

static::getMessage('ACRIT_EXAMINATION_TEST_PERFMON_ERROR_NOT_INSTALLED');

static::getMessage('ERROR_NOT_INSTALLED');
```

Для получения опций следует использовать метод класса:

```
static:: getOption($strOption, $strValue=false);
```

Также модуль имеет класс ReportTable для создания html-таблиц в отчетах. Пример работы этого класса можно найти в любом тесте, генерирующем таблицу.

Особенности работы при запуске из командной строки

Модуль умеет автоматически (по нажатию кнопки) для каждого профиля настраивать задания планировщика, однако не на всех серверах это доступно. Если на Вашем сервере это недоступно, следует вручную настроить задание на сервере (при наличии панели управления хостингом это делается в панели, иначе – вручную через SSH). В любом случае, текст команды для запуска можно «подсмотреть» во всплывающем окне при настройке автозапуске.

Запуск из командной строки (в т.ч. запуск из планировщика) имеет ряд особенностей. Первая особенность – кодировка. На многих серверах параметры php для сайта и для командной строки в разной степени различаются (особенно это касается настроек mbstring). А необходимы настройки идентичные. Поэтому в команду запуска из планировщика модуль автоматически добавляет конфигурационные параметры mbstring.

Вторая особенность – все тесты проводятся за один шаг, хотя execute() в тестах выполняется в точности как и при обычном запуске – необходимое число раз.

Следующая особенность – т.к. задание выполняется за один шаг, то на сервер может создаваться высокая нагрузка, и на некоторых серверах это может вызвать проблемы, если заданы какие-либо ограничения на нагрузку (например, скрипт после нагрузки в течение 30 секунд может автоматически завершаться, при этом задание не будет до конца выполнено).

Итоговая команда может выглядеть по-разному на каждом сервере, т.к. на каждом сервере свои особенности. Пример команды:

```
/usr/bin/php -d "mbstring.func_overload"=2 -d "mbstring.internal_encoding"=UTF-8 -f
/home/bitrix/www/bitrix/modules/acrit.examination/cron/exec_profile.php profile=1 start=Y
```

Здесь profile=1 указывает, что будет выполняться профиль с ID=1, а параметр start=Y – это технический параметр, при этом - обязательный.

Для автоматической настройки требуется указание пути в php. Указать путь в php на своем сайте Вы можете в настройках модуля. Также там имеется возможность включить или отключить добавление в команду параметров mbstring.

Внимание! Первый запуск настоятельно рекомендуем провести через SSH, т.к. в этом случае сразу видны возможные проблемы при настройке, в т.ч. проблемы с отправкой почты. Также, настраивая планировщик, желательно указать свой email в его настройках, чтобы при наличии ошибок сообщения об этом приходили на email.

Внимание! Задания планировщика должны выполняться от имени того же пользователя, от имени которого работает сайт. Иначе возможны проблемы в дальнейшей работе. Так, при использовании виртуальной машины Битрикс, если запускать команды через SSH войдя под пользователем root, задание будет выполняться от имени root, поэтому, если все файлы сайта принадлежат пользователю bitrix, то файлы и папки модуля, создаваемые во время выполнения задачи (например, это папка reports для отчетов и сами отчеты в данной папке) будут принадлежать пользователю root, и сайт не сможет работать с этими файлами. При этом тест модуля на проверку доступа будет «ругаться» на эти папки и файлы.

Внимание! Если на сервере не работает отправка email-сообщений, PDF-отчет не может быть отправлен на почту.