iTrack
Эффективные веб-решения

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

Руководство по интеграции.
Проекты на 1С-Битрикс Управление сайтом

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

1. Введение.

Что такое "интеграция"?
С точки зрения разработчика это создание удовлетворяющего требованиям Технического задания (Проектной документации) сайта на платформе 1С-Битрикс с использованием предоставленной верстки и дизайна.

Допустим, интеграция проведена, проект полностью выполняет функции, описанные в Техническом задании, внешний вид страниц сайта проекта соответствует дизайну. На первый взгляд все сделано. Однако если проект проинтегрирован некачественно, могут возникнуть следующие серьезные проблемы (перечислены наиболее распространенные):

1. 1С-Битрикс, как и подобает качественному коммерческому программному продукту, постоянно развивается и выпускает обновления и исправления обнаруженных ошибок. При первом после окончания разработки (или очередном) обновлении проекта ряд функций сайта начинают работать неправильно или вообще не работают.

2. СУБД, с использованием которой разрабатывался проект, была заменена. Это делается для увеличения надежности, масштабируемости и производительности при нагрузках, например MySQL меняют на OracleXE. После замены СУБД сайт начал работать некорректно или вообще не работает.

3. Хостер проекта изменил некоторые настройки сервера, в результате чего проект начал работать некорректно или вообще перестал работать.

4. Сайт плохо управляем. Например, чтобы поменять номер телефона на странице контактов необходимо привлекать программиста или требуются права администратора сайта. Чтобы изменить направление сортировки или количество выводимых на станице анонсов опять-таки требуется привлекать квалифицированного программиста.

5. Путем подбора параметров строки запроса браузера, посетители сайта получают несанкционированный доступ к закрытой информации на сайте.

6. При увеличении посещаемости сайта сильно замедляется работа системы. Проект "тормозит" или "зависает".
Качественная интеграция производится с учетом вышеперечисленных проблемных аспектов. В результате тесного взаимодействия с 1С-Битрикс, анализа и накопленного опыта компания iTrack выработала правила качественной интеграции, обеспечивающие выполнение следующих условий:

1. Сайт управляем. Основные функции компонентов сайта являются на-страиваемыми. Сайтом могут управлять рядовые сотрудники, не имеющие технического образования.

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

2. Сайт может работать на любой СУБД, поддерживаемой API Битрикс, а также поддерживаемых 1С-Битрикс версиях PHP, apache (если нет указаний конкретных версий данных продуктов в ТЗ)

3. Обновления 1С-Битрикс не влияют на работоспособность сайта.

4. Изменения настроек хостинга не влияют на функционирование проекта. Проект использует стандартные настройки большинства хостингов.

5. Закрытая информация сайта надежно защищена.

6. Сайт адаптирован к возможным увеличениям нагрузки. Основные компоненты сайта используют механизмы кэширования запросов к базе данных.

2. Программирование.

2.1 Работа с модулями, стандартными компонентами, классами, функциями API Битрикса.

В служебной папке /bitrix во время интеграции можно изменять только файлы, расположенные в /bitrix/php_interface, /bitrix/templates. Остальные файлы менять запрещается.
Все внесенные в модули изменения (/bitrix/modules) будут затерты при очередном обновлении.

Подключать служебные файлы Битрикс в обход API нельзя!

```
@include($_SERVER["DOCUMENT_ROOT"].$dir.".section.php");
//запрещено!
```

Не допускаются прямые обращения к СУБД и вызовы недокументированных функций!

В модулях Битрикс существует большое количество готовых компонентов, которые можно и нужно использовать для выполнения проекта. Нужно тщательно поискать подходящий компонент и если такого компонента нет, создавать свой. Стандартные компоненты Битрикс хранятся в папке: /bitrix/components/bitrix/.

При использовании стандартных компонентов Битрикс нужно понимать, что их писали квалифицированные разработчики и в них нет "лишних деталей". Прежде чем кастомизировать стандартный компонент нужно понять логику его работы, в том числе связанную с проверкой соответствующих прав. Все "непонятные" функции API, использованные в компоненте, нужно выписать и изучить. Если остаются вопросы, их необходимо задать техническим специалистам iTrack. Только после этого разрешается приступать к кастомизации компонента.

Категорически запрещается модифицировать код стандартных компонентов.

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

При очередном обновлении все изменения компонентов будут затерты.

2.2 Написание собственных компонентов, кастомизация стандартных.

Кастомизировать стандартные компоненты можно двумя путями:

1. Несущественные изменения логики компонента (дополнительные поля для вывода, вывод полей связанных объектов). В этом случае можно кастомизировать только шаблон компонента. Копируем стандартный шаблон компонента templates/.default/ в папку шаблона сайта /bitrix/templates/TEMPLATE_NAME/components/bitrix/COMPONENT_NAME/ . При необходимости изменения массивов передаваемых в шаблон ($arResult, $arParams) создаем в указанной выше папке кастомного шаблона компонента файл result_modifier.php. Этот файл будет подключаться после component.php стандартного компонента, но до template.php кастомного шаблона.

По поводу конечного места, куда копировать шаблоны проекта, небольшое замечание. Тут стоит оценить проект в целом, и его дальнейшее развитие, возможно посоветоваться с менеджером проекта. В некоторых случаях шаблоны выгоднее хранить в /bitrix/templates/.default. Это удобно, например, если у проекта несколько шаблонов (обычный и для печати).

2. Существенные изменения логики компонента. При такой кастомизации стандартных компонентов их нужно сначала копировать, а затем модифицировать в папке: /bitrix/components/PROJECT_NAME/.

Если создается новый компонент, или модифицируется стандартный компонент, то такой компонент должен появиться в визуальном редакторе в правой колонке (выпадающий список) в разделе с названием выполняемого проекта. Стандартные компоненты модулей Битрикс отображаются в визредакторе в соответствующих разделах выпадающего списка: "Информационные блоки", "Форум", "Интернет-магазин". Для выполняемого проекта "MyProject" соответственно в выпадающем списке в визредакторе должен появиться пункт "MyProject", при выборе которого правая колонка отображает подразделы, где расположены компоненты проекта. Администратор сайта в визредакторе должен иметь возможность выбрать в правой колонке название проекта, и перетащить на выбранную страницу любой компонент, используемый в проекте. ID новых компонент, а также их папок, должны быть уникальными в рамках одного дерева компонент. Лучше именовать их с префиксом проекта (myproject_mycomponentfolder).

Если компонент выводит html и используется на страницах сайта, он в обязательном порядке должен иметь соответствующие конфигурационные файлы для корректного своего отображения и отображения своих настроек в визредакторе (.description.php). Часто встречается ситуация, когда разработчик написал полезный компонент и разместил его на определенной странице, но разместить данный компонент на другой странице администратору без привлечения программиста невозможно, т.к. разработчик не создал файл описания компонента (.description.php) и компонента нет в визредакторе - "перетянуть" его на страницу невозможно. Настроить данный компонент через визредактор также невозможно. Единственный способ настроить

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

параметры компонента: изменить php-код его подключения. Единственный способ разместить компонент на другой странице - скопировать код подключения компонента <?$APPLICATION->IncludeComponent(…)?>

Для предотвращения подобной ситуации необходимо обязательно создавать файл описания компонента!

2.3 Работа с API Битрикс.

При работе с Битрикс нужно использовать API Битрикс. Причем только документированные в Руководстве для разработчика функции с документированными параметрами.

Делать запросы к базе данных напрямую, или используя соответствующие методы класса CDatabase категорически запрещается!

Разработчики Битрикс могут в любой момент поменять формат таблиц СУБД и что тогда случится с проектом?
Следует особо обратить внимание и использовать в работе принцип разбиения API Битрикс "на две части":

1. Глобальные функции, типа GetIBlockElementList. Данные функции являются "wrappers" к более низкоуровневым классам (в данном случае к методу CIBlockElement::GetList) и проверяют права пользователя, активность элементов и другие важные параметры. Глобальные функции необходимо применять в публичной части сайта в первую очередь, т.к. они обеспечивают максимальную "логическую" безопасность запросов, и только если они не удовлетворяют требованиям задачи, необходимо использовать API из "второй части".

Следует отметить, что эти функции вместе с тем генерируют зачастую ненужные запросы. И если есть уверенность, что такие проверки не обязательны, то рекомендуется использовать все же классы, из п.2.

2. Классы. Классы API решают низкоуровневые задачи и, как правило, более гибки, чем глобальные функции. Использование методов этих классов в публичной части "небезопасно", и требует предварительной проверки прав пользователя и возможно других параметров. За гибкость приходится платить. Разумеется, sql-инъекцию через такие классы (как и через остальные классы/фукнции API Битрикс) сделать нельзя, но получить данные из закрытого для пользователя инфоблока - запросто, если предварительно не провести необходимые проверки.

2.4 Использование собственных констант, функций, классов.

Часто встречаются ситуации, когда в коде компонентов (шаблонов меню) необходимо использовать общесистемные настройки или конкретные инфоблоки. Например, необходимо чтобы на сайте во всех компонентах использовалось глобальное время кэширования, или если оно не задано, то время кэширования компонента. В этом случае

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

рекомендуется все настройки определить константами в начале инициализационного файла проекта /bitrix/php_interface/init.php:

Константы обязательно должны иметь префикс, образованный из названия проекта, чтобы не перепутаться с константами Битрикс, как правило не использующими префиксы!

Если компоненты используют служебный инфоблок, выносить ID служебного инфоблока в настройки компонента не нужно; нужно определить константу и использовать ее.

Иногда требуется в качестве настроек использовать не скалярные значения (константы php), а массивы или объекты. В таком случае в /bitrix/php_interface/init.php нужно определить глобальную переменную с префиксом проекта.

Нередко появляется необходимость определить собственные функции (например, callbacks) или классы. Рекомендуется определять их внутри файла /bitrix/php_interface/init.php. Можно при необходимости создать файл в директории /bitrix/php_interface/ в котором определить свои функции и классы и подключить файл в начале /bitrix/php_interface/init.php.
Рекомендуется к применению примерно следующая структура, в /bitrix/php_interface/init.php храним только пути до файлов с обработчиками:

```
<?
@include('include/handlers/main.php');
@include('include/handlers/fileman.php');
@include('include/handlers/iblock.php');
@include('include/handlers/forum.php');
@include('include/handlers/blog.php');

@include('include/utils/main.php');
@include('include/utils/someprojectclass.php');
@include('include/utils/othersystemclass.php');
?>
```

Как видно по названиям директорий и файлов, функции и классы для разных модулей разнесены по разным файлам/классам. А уже конкретный файл собирает в себе все необходимое для работы с тем или иным модулем:

```
<?
AddEventHandler("forum", "onBeforeMessageAdd", Array("CForumHandlers",
"onBeforeMessageAddHandler"));
AddEventHandler("forum", "onAfterMessageAdd", Array("CForumHandlers",
"onAfterMessageAddHandler"));
AddEventHandler("forum", "onBeforeMessageDelete", Array("CForumHandlers",
"onBeforeMessageDeleteHandler"));
AddEventHandler("forum", "onBeforeTopicDelete", Array("CForumHandlers",
"onBeforeTopicDeleteHandler"));

class CForumHandlers
{
        public function onAfterMessageAddHandler($ID, &$arFields)
```

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

.....
?>
Это будет выгодно на развитом проекте, когда для доступа к той или иной функции-обертке, или к обработчику, будет максимально быстрый и безопасный (инкапсулированный) доступ. Для небольших проектов возможно «складирование» своих функций просто в init.php

Нельзя подключать свои функции с помощью CMain::IncludeFile, т.к. эта конструкция используется для подключения включаемых областей.

Также все определяемые разработчиком при интеграции функции (PHP, JS) должны быть собраны в одном месте, а не разбросаны по компонентам и разным файлам. Место "сбора" указано выше.

Как было сказано, функции и классы должны использовать префикс проекта, чтобы не перемешаться с функциями и классами Битрикс. В именовании классов и функций необходимо использовать один из распространенных стандартов, описанных ниже. Предпочтительным является объектно-ориентированный подход в связи с его общепризнанным преимуществом - инкапсуляция, повторное использование кода и др. Всегда удобнее использовать класс с четко определенным интерфейсом, чем "кучу" глобальных функций и глобальных переменных.

Если необходимо создать несколько связанных классов, необходимо "нарисовать" диаграмму классов и согласовать ее со специалистами iTrack. Для решения подобного рода задач рекомендуется использование паттернов проектирования, как проверенных на практике успешных и масштабируемых решений.

Для скрытых переменных и методов классов желательно использовать префикс "_". Каждую глобальную переменную, константу, функцию, файл, класс, метод, член класса необходимо тщательно документировать (в том числе входные и выходные параметры метода/функции, с подробным пояснением; если входным параметром является массив, нужно описать его структуру и если это ассоциативный массив - прокомментировать назначение его ключей).

2.5 Требования к шаблонам сайта, компонентов.

Шаблон главной страницы.
Желательно, чтобы главная страница целиком содержалась в header.php, footer.php должен быть пуст. Это «защищает» сложную верстку главной страницы от непреднамеренного искажения в визуальном редакторе (были случаи, что страницу открывали в визредакторе, ничего не меняли и сохраняли, после чего сайт "разъезжался"). Управление должно осуществляться через включаемые области или настройки компонентов.
Файлы изображений и js, относящиеся к шаблону, нужно хранить в папке шаблона сайта (изображения - в подпапке images папки шаблона сайта). Общие для всех шаблонов ресурсы должны храниться в корне сайта (общие изображения - в папке /SITE_DIR/images). Не забываем использовать при подключении ресурсов константы:

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

SITE_TEMPLATE_ID, SITE_DIR (а также SITE_SERVER_NAME - в случае необходимости).

Шаблоны компонентов.
Необходимо иметь в виду, что в режиме Битрикс "редактировать сайт" код подключения компонентов (меню, включаемой области) обрамляется дополнительной верстkой для поддержки административного режима Битрикс.

Поэтому необходимо чтобы компоненты (меню, включаемые области) возвращали "завершенный" html-код, корректно отображающийся внутри "административной" разметки.

Нельзя начинать/завершать таблицу в шаблоне или контентной области, а возвращать ее "середину" из компонента (меню, включаемой области). Нельзя открывать тэг(и) перед вызовом компонента (меню, включаемой области), а закрывать внутри компонента (меню, включаемой области).

Добавление/удаление компонента (меню, включаемой области) со страницы не должно влиять на корректность html-представления страницы. Не должно быть компонентов (меню, включаемых областей), которые требуют вставки только в определенные тэги, например между "<table>...</table>" или "<tr>...</tr>" или "<select>...</select>". Удаление/добавление таких компонентов (меню, включаемых областей) со страницы чревато "разъезжанием" верстки.
Рекомендуется возвращать из компонентов (меню, включаемой области) корректную верстку с одним из корневых элементов: table, div, span, select.

2.6 Кэширование.

Нужно обязательно применять кэширование при использовании комплексных выборок из инфоблоков и других "потенциально медленных" вызовов API Битрикс. Узнать, что на сайте вызывает "торможение" несложно – для этого необходимо тщательно изучить работу переменных запроса: show_page_exec_time, show_include_exec_time, show_sql_stat. Битрикс предоставляет более чем достаточное количество полезной информации, необходимой для профилирования проекта.

Кэшировать запросы к API нужно в компонентах. Если того требует логика, то допускается кеширование в собственных классах/функциях. Время кэширования компонентов обязательно выносится в настройки компонента (но может дублироваться еще и константой), время кэширования функций, методов классов, меню лучше определять константой (с префиксом проекта!) в init.php (чтобы не лазить во все шаблоны меню и внутрь функций и методов для изменения времени кэширования).
При использовании кэширования обычно допускают 2 вида ошибок: неправильный идентификатор кэша (ID кэша) и неправильный паттерн использования классов/методов кэширования. В результате кэш или не работает, или "глючит" на постраничной навигации, или возвращает данные другого компонента.

Примеры правильной генерации ID кэша повсеместно приводятся в документации по Битрикс. Нужно помнить, что если компонент использует постраничную навигацию, в

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

ID кэша нужно включить результат DBResult::NavStringForCache($PAGE_ELEMENT_COUNT).

Если компонент подключает внутри себя другой компонент и используется кэширование в режиме "редактировать сайт", нужно включать в идентификатор кэша вызов: $APPLICATION->GetShowIncludeAreas().
Еще проще включать этот вызов в любой идентификатор кэша. Примеры использования паттернов кэширования также приводятся в документации достаточно подробно и с комментариями. Нужно понять логику их работы и использовать.
Учитывать группы пользователя в ID кэша также очень важно. Иначе неавторизованный посетитель сможет увидеть закэшированные данные (возможно закрытые) авторизованного посетителя.

Для привязки кэша к функции в ID кэша рекомендуется использовать константу __FUNCTION__, в методе класса __CLASS__, а в компоненте __FILE__.

Вывод: используйте в ID кэша все, что влияет на генерацию кэша, не забывая про служебные режимы Битрикс, авторизацию и постраничную навигацию.

2.7 Обновление продукта.

Обновлять систему можно только по согласованию с менеджером проекта!
Дело в том, что перед обновлением желательно сделать резервную копию публичной части, ядра и базы, а встроенные скрипты Битрикс не всегда делают это корректно. Поэтому перед обновлением на сервере необходимо провести подготовительные административные работы, не относящиеся к компетенции разработчика. Несогласованное обновление может нарушить работоспособность проекта.

2.8 Стиль программирования.

Расстановка фигурных скобок и отступы.
Существует несколько стилей расстановки фигурных скобок, все они диктуются существующими стилями в других C-подобных языках программирования. Можно пользоваться любым удобным стилем (одним для всего кода), но рекомендуется следовать «рациональному» стилю, принятому в iTrack:

Это один из наиболее распространенных стилей, так как им пользовались Керниген (Kernighan) и Ричи (Ritchie), авторы языка C.

```
<?php
  if($flag){
    echo "Hello world!";
  }
?>
```

Пробелы вокруг символов.
Бинарные операторы следует обрамлять пробелами:

```
<?php
```

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

```
 // Неправильно
 $a=$b+$c*$d;
 // Правильно
 $a = $b + $c * $d;
?>
```

Символ пробела ассоциируется с новым словом, поэтому формула читается не как непонятный набор символов, а как нечто осмысленное.

Комментарии.

Расставлять комментарии следует по принципу “чем больше, тем лучше” — пройдёт некоторое время и забудется, что делал тот или иной программный блок. Принято комментировать код на английском языке или не комментировать вообще, так как в случае кириллицы существует большое количество кодировок, из-за этого могут возникнуть проблемы с чтением файлов на хостинге.

PHP собрал в себе практически все комментарии современных языков программирования, наряду с однострочными комментариями в стиле shell-скриптов (#).

```
<?php
 # Программный модуль index.php
 echo "Hello world!";
?>
```

и C++ (///)

```
<?php
 // Программный модуль index.php
 echo "Hello world!";
?>
```

можно использовать многострочный комментарий в стиле C:

```
<?php
 /* Это многострочный комментарий в стиле C
    он охватывает несколько строк – не допускается
    вложенных комментариев
 */
 echo "Hello world!";
?>
```

К хорошему тону относится использование однострочных комментариев для короткого комментария, а многострочного — для комментария, охватывающего несколько строк. Не следует использовать однострочные комментарии для большого текста, особенно в начале файла или важного блока кода

```
<?php
 ////////////////////////////////////////////////////////////
 // Гостевая книга
 ////////////////////////////////////////////////////////////
?>
```

Имена переменных и функций

Существует несколько стилей названия переменных

$var_bell — стиль C: нижний регистр, знак подчёркивания.

$VarBell — стиль Pascal: каждая подстрока в названии начинается с большой буквы.

$varBell — стиль Java: первая строка начинается с маленькой буквы, все после-

г. Москва,
Семеновский пер., дом 15
http://www.itrack.ru
info@itrack.ru
+7 (495) 638-52-14

дующие с большой.
Разрешается использовать любой из вышеперечисленных стилей — главное придерживаться в коде одного стиля. Стиль принятый в iTrack – Java-стиль.

Замечание
Константы необходимо записывать в верхнем регистре YANDEX_BOT.
Названия переменным и функциям необходимо давать осмысленные.