

# 1С-Битрикс: Управление сайтом

Разработчик Bitrix Framework

## Содержание

### Введение 8

<b>Глава 1. Что такое Bitrix Framework?</b> .....	<b>10</b>
---	-----------

Как изучать BITRIX FRAMEWORK? .....	11
Как правильно построить процесс изучения? .....	11
Сколько времени уйдет на освоение Bitrix Framework? .....	12
ГДЕ БРАТЬ ИНФОРМАЦИЮ? .....	13
Сообщество разработчиков .....	18
ДЛЯ ТЕХ, КТО ПЕРЕХОДИТ НА BITRIX FRAMEWORK С ДРУГИХ ПЛАТФОРМ .....	19
Bitrix Framework и Drupal .....	20
Bitrix Framework и Joomla .....	20
Глоссарий .....	21

<b>Глава 2. Архитектура продукта</b> .....	<b>33</b>
--	-----------

АРХИТЕКТУРА ПРОДУКТА .....	34
Архитектура MVC для Bitrix Framework .....	34
Немного теории .....	34
Структура .....	35
Элементы структуры Bitrix Framework .....	35
СТРУКТУРА ФАЙЛОВ .....	36
Файлы и База данных .....	37
Структура файлов .....	39
ПРАВА ДОСТУПА .....	41
Доступ на файлы и каталоги .....	41
ИНФОРМАЦИЯ НА САЙТЕ И РАБОТА С НЕЙ .....	44
САЙТ В ПОНЯТИИ BITRIX FRAMEWORK .....	44
Страница .....	45
Шаблон сайта .....	54
Язык и языковые файлы .....	56

<b>Глава 3. Технологии</b> .....	<b>59</b>
----------------------------------	-----------

Административная панель .....	59
Агенты .....	60
Кеширование .....	61
Производительность .....	61
Кеширование .....	62
Примеры .....	63
События .....	64
Различия в использовании функций .....	66
Отложенные функции .....	67

<b>Глава 4. Интеграция дизайна</b> .....	<b>70</b>
--	-----------



ИСПОЛЬЗОВАНИЕ ПРАВ ДОСТУПА .....	71
ШАБЛОН ДИЗАЙНА САЙТА.....	73
Шаблон дизайна .....	74
Структура файлов шаблона сайта .....	78
Разработка шаблона дизайна .....	79
Управление шаблоном дизайна.....	83
Разработка шаблонов страниц .....	89
Использование файлов языковых сообщений .....	91
Примеры работы и решения проблем.....	91
ФАЙЛЫ ЯЗЫКОВЫХ СООБЩЕНИЙ .....	94
Механизм реализации .....	95
Загрузка и выгрузка локализации .....	96
Изменение фраз в компонентах и модулях .....	99
ВКЛЮЧАЕМЫЕ ОБЛАСТИ.....	100
Использование включаемых областей .....	100
Управление включаемыми областями .....	102
Средства навигации.....	107
Карта сайта .....	107
Меню в Bitrix Framework.....	109
Цепочка навигации.....	152
РЕКЛАМНЫЕ ОБЛАСТИ .....	162
Типы рекламы.....	163
Управление показом рекламы с помощью ключевых слов .....	165
УПРАВЛЕНИЕ СЛУЖЕБНЫМИ ДАННЫМИ ШАБЛОНА .....	168
Редактирование служебных областей .....	168
Управление кодировкой страниц .....	170
Управление метаданными.....	175
Управление заголовком документа .....	178
Управление стилями .....	186
НАСТРОЙКА ВНЕШНЕГО ВИДА ДОПОЛНИТЕЛЬНЫХ ЭЛЕМЕНТОВ ОФОРМЛЕНИЯ САЙТА.....	194
<b>Глава 5. Как создать простой сайт .....</b>	<b>196</b>
ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА САЙТ .....	196
ВЕРСТКА БАЗОВОГО ШАБЛОНА .....	198
ИНТЕГРАЦИЯ КОМПОНЕНТОВ.....	200
Примеры размещения компонентов.....	202
КАСТОМИЗАЦИЯ ШАБЛОНОВ КОМПОНЕНТОВ .....	203
Создание структуры сайта.....	207
НАСТРОЙКА ИНФОБЛОКОВ.....	209
УПРАВЛЕНИЕ СЛУЖЕБНЫМИ ДАННЫМИ .....	210
<b>Глава 6. Инфоблоки .....</b>	<b>212</b>
РАБОТА С ИНФОБЛОКАМИ ШТАТНЫМИ СРЕДСТВАМИ .....	213

ИНФОБЛОКИ 2.0 .....	218
ПЛАН ДЕЙСТВИЙ ПРИ ПРОБЛЕМАХ .....	222
ФИЛЬТРАЦИЯ .....	223
РАБОТА С ИНФОБЛОКАМИ ЧЕРЕЗ API .....	226
Работа с пользовательскими свойствами инфоблоков .....	229
Выборка из нескольких инфоблоков .....	241
Копирование инфоблока .....	244
Инфоблоки в Документообороте .....	248
Вывод свойств элемента инфоблока .....	249
Некоторые ошибки при работе с инфоблоками .....	250
<b>Глава 7. Компоненты .....</b>	<b>251</b>
ПРОСТЫЕ И КОМПЛЕКСНЫЕ .....	253
РАЗМЕЩЕНИЕ КОМПОНЕНТА В СИСТЕМЕ И ЕГО ПОДКЛЮЧЕНИЕ .....	256
СТРУКТУРА КОМПОНЕНТА .....	259
ОПИСАНИЕ КОМПОНЕНТА .....	263
ПАРАМЕТРЫ КОМПОНЕНТА .....	265
ФАЙЛ RESULT_MODIFIER.PHP .....	271
ФАЙЛ COMPONENT_EPILOG.PHP .....	273
РАБОТА КОМПЛЕКСНОГО КОМПОНЕНТА В SEF РЕЖИМЕ .....	276
ТИПИЧНЫЕ ОШИБКИ .....	278
ДОБАВЛЕНИЕ ПРОИЗВОЛЬНОГО PHP КОДА .....	280
КЕШИРОВАНИЕ КОМПОНЕНТОВ .....	280
Автокеширование .....	282
Cache Dependencies (тегированный кеш) .....	284
РАБОТА С КОМПОНЕНТАМИ .....	287
Кастомизация шаблона .....	288
Файлы result_modifier и component_epilog .....	304
Использование событий .....	315
Кастомизация компонентов .....	317
Создание компонентов .....	320
Ошибки при работе с компонентами .....	353
<b>Глава 8. Модули .....</b>	<b>355</b>
СТРУКТУРА ФАЙЛОВ .....	357
ОПИСАНИЕ И ПАРАМЕТРЫ .....	358
АДМИНИСТРАТИВНЫЕ СКРИПТЫ .....	363
ВЗАИМОДЕЙСТВИЕ МОДУЛЕЙ .....	364
Пример взаимодействия .....	366
УСТАНОВКА И УДАЛЕНИЕ .....	367
КАСТОМИЗАЦИЯ И СОЗДАНИЕ МОДУЛЕЙ .....	368
Пример изменения работы модуля .....	369
Создание собственных модулей .....	370



Конструктор модулей для MARKETPLACE .....	401
Работа с конструктором .....	402
Сборка обновлений модуля .....	409
<b>Глава 9. Программирование в Bitrix Framework .....</b>	<b>414</b>
ЗОЛОТЫЕ ПРАВИЛА .....	414
ФАЙЛ INIT.PHP .....	416
UTF-8 ИЛИ WINDOWS-1251 ? .....	419
РАБОТА С БАЗАМИ ДАННЫХ .....	423
ЯЗЫКОВЫЕ ФАЙЛЫ .....	426
Языковые файлы в собственных компонентах .....	428
Замена языковых фраз продукта .....	429
ИНТЕРФЕЙС "ЭРМИТАЖ" С ТОЧКИ ЗРЕНИЯ РАЗРАБОТЧИКА .....	430
Добавление кнопок на панель управления .....	431
Добавление контекстного меню .....	432
Toolbar компонента .....	433
Контекстное меню элементов списка .....	434
Административные страницы в публичке .....	435
Новые методы буферизации .....	435
НЕМНОГО ТЕОРИИ .....	437
Замечания по \$arParams и \$arResult .....	437
Псевдонимы входящих переменных .....	438
Правила написания исходного кода на PHP .....	441
БЕЗОПАСНОСТЬ .....	450
Санитайзер .....	450
Защита от фреймов .....	454
ПРИМЕРЫ, ХИТРОСТИ И СОВЕТЫ .....	455
Командная PHP-строка .....	455
Мелкая оптимизация при разработке .....	456
Сортировка в компонентах news.list и catalog.section .....	460
Использование агентов .....	463
Использование событий .....	471
ПОЛЬЗОВАТЕЛЬСКИЕ ПОЛЯ .....	478
Создание полей .....	480
Примеры работы .....	481
Добавление, редактирование, удаление пользовательских свойств и их значений .....	484
Поля к нештатным объектам и новые объекты .....	489
ГАДЖЕТЫ .....	490
Структура гаджета .....	490
ТЕСТИРОВАНИЕ ПРОЕКТОВ .....	491
Монитор качества .....	493
Несколько советов .....	508

<b>Глава 10. Многосайтowość .....</b>	<b>509</b>
Использование многосайтовой версии .....	510
Многоязычность .....	511
Чем определяется количество сайтов в системе.....	513
Языковые версии сайта .....	515
Технология переноса посетителей между сайтами.....	516
Конфигурирование многосайтowości .....	518
Создание и настройка сайта .....	519
Настройки сайта и настройки языков .....	523
Как система различает сайты .....	526
Многосайтowość на одном домене .....	527
Многосайтowość на разных доменах.....	529
Настройка многосайтowości на разных доменах .....	530
Многосайтowość на разных доменах на IIS .....	536
Псевдомногосайтowość на разных доменах.....	539
Примеры настроек сервера Apache .....	543
Выделение разделов сайта в поддомены .....	545
Работа с данными в многосайтовой конфигурации .....	546
Работа со структурой сайта .....	546
Какие объекты можно позиционировать по сайтам .....	549
Какие настройки модулей разделяются по сайтам .....	553
Какую статистику можно анализировать в разрезе по сайтам .....	554
Типовые вопросы возникающие при работе с многосайтовой конфигурацией .....	556
Удаление сайта и связанных объектов.....	556
Как закрыть только один из сайтов для посещения пользователей .....	557
Компонент для переключения сайтов .....	558
Сохранение авторизации пользователя при переходе по различным сайтам .....	559
<b>Глава 11. Бизнес-процессы для разработчика .....</b>	<b>560</b>
Шаблон бизнес-процесса.....	561
Бизнес-процесс .....	564
Типы бизнес-процессов .....	565
Последовательный бизнес-процесс.....	565
Бизнес-процесс со статусами .....	566
Выбор типа бизнес-процесса .....	567
Действия .....	567
Основные стандартные действия.....	568
Свойства действий .....	569
Составные действия .....	571
Создание собственных действий.....	573
Произвольный PHP код в бизнес-процессе .....	600
Как запустить из одного бизнес процесса другой? .....	600

Вывод в лог .....	601
Вывод в лог. Переменные .....	603
Арифметические действия в бизнес-процессе .....	604
Вычисление ID начальника .....	604
<b>Глава 12. Дополнительные сведения и примеры .....</b>	<b>606</b>
Если нет описания API .....	606
ВЕБ-СЕРВИСЫ .....	609
Пример создания windows-приложение для добавления новостей .....	610
ПРИМЕР МИГРАЦИИ БАЗЫ САЙТА .....	621
Предварительные операции .....	622
Сравнение структур баз .....	635
Перенос данных .....	648

## Введение

### Цитатник веб-разработчиков.

**⚠ Евгений Смолин:** Сам кувыркался с непонятками (пока курсы учебные не прочитал и руками не попробовал то, что там написано). Битрикс из тех систем, где изучение "методом тыка" не очень эффективно без предварительного изучения учебных курсов. Уважаемые новички, потратьте немного своего драгоценного времени, пройдите пару-тройку учебных курсов и масса вопросов просто испарится - там есть ответы на множество вопросов.

Курс для разработчиков завершает линейку учебных курсов по **Bitrix Framework**. Получение сертификата по курсу возможно только после успешной сдачи тестов по всей линейке курсов.

Чтобы научиться программировать в **Bitrix Framework**, нет необходимости изучать всю линейку курсов. Но есть моменты, которые необходимо знать разработчикам о системе, они раскрыты в начальных курсах:

- **Интерфейс программы** - в главе [Элементы управления](#) курса **Контент-менеджер**.
- **Компоненты 2.0 (начальные сведения)** в главе [Компоненты 2.0 \(начальные сведения\)](#) курса **Контент-менеджер**.
- **Информационные блоки (начальные сведения)** - в главе [Информационные блоки \(начальные сведения\)](#) курса **Контент-менеджер**.
- **Управление доступом** к файлам, элементам контента, модулям и другие права доступа в главе [Управление доступом](#) курса **Администратор. Базовый**.
- **Работа с инструментами** системы - в главе [Работа с инструментами](#) курса **Администратор. Базовый**.
- Модуль **Поиск** - в главе [Поиск](#) курса **Администратор. Базовый**.
- Модуль **Информационные блоки** - в главе [Модуль Информационные блоки](#) курса **Администратор. Базовый**.
- Вся информация по администрированию модулей размещена в курсах:
  - [Администрирование. Модули](#) - модули "1С-Битрикс: Управление сайтом"
  - [Администратор. Бизнес](#) - модули "1С-Битрикс: Управление сайтом", связанные с коммерческой деятельностью в Интернете.
  - [Администратор "1С-Битрикс: Корпоративный портал"](#) - модули "1С-Битрикс: Корпоративный портал"



## Начальные требования к подготовке

Для успешного изучения курса и овладения мастерством разработки сайтов на **Bitrix Framework** необходимо владеть (хотя бы на начальном уровне):

- основами PHP;
- основами HTML, CSS.

**⚠ Примечание:** В тексте курса вы встретите цитаты, высказанные в разное время разработчиками системы и разработчиками проектов на базе **Bitrix Framework**. Надеемся, что такие неформальные замечания внесут некоторое разнообразие в процесс изучения. Заодно опытные специалисты поделятся и своим опытом.

Имена авторов цитат даются в том написании, в каком авторы зарегистрировали себя на сайте "1С-Битрикс".

## Глава 1. Что такое Bitrix Framework?

Цитатник веб-разработчиков.

**⚠ Степан Овчинников:** Удивительно, но платность Битрикса вызывает почему-то гнев тех, кто за него никогда не платил.



**Bitrix Framework** - это созданная на основе PHP платформа для разработки веб-приложений. На этой платформе компанией «1С-Битрикс» созданы два популярных продукта: «**1С-Битрикс: Управление сайтом**» и «**1С-Битрикс: Корпоративный портал**».



Продукт «**1С-Битрикс: Управление сайтом**» представляет собой программное ядро для всестороннего управления веб-проектами любой сложности.

Продукт «**1С-Битрикс: Корпоративный портал**» - готовый продукт, позволяющий создать корпоративный портал компании, с возможностью доработки штатного функционала под потребности компании.

В отличие от [Zend Framework](#) при разворачивании **Bitrix Framework** мы получаем не только набор классов, но и развитый интерфейс администрирования.

В базовой поставке идёт большой набор компонентов, и именно он обеспечивает быстрое развёртывание и внедрение проектов.

Продукты на базе **Bitrix Framework** выходят в нескольких редакциях. Изучая систему и повторяя уроки необходимо быть уверенным, что ваша локальная установка имеет модуль, с которыми вы экспериментируете. Помодульное сравнение редакций: для [1С-Битрикс: Управление сайтом](#), для [1С-Битрикс: Корпоративный портал](#).

**⚠ Примечание:** В учебном курсе будут приводиться примеры из задач стоящих перед разработчиками разных продуктов. Механизмы решения этих задач можно применять в любом другом продукте, созданном на **Bitrix Framework**. В этом плане слова **сайт** и **корпоративный портал** в рамках этого курса можно рассматривать как синонимы.

## Как изучать Bitrix Framework?

### **Опыт подготовки разработчиков от одного из партнеров**

У нас стажёры (типовoy опyт: 5 лет институтa, возможно, фриланс и знание PHP) с обучением по курсам, внутренней документации и консультациями разработчика, примерно через месяц понимают как и что устроено. Общий курс обучения - порядка 2-3 месяцев.

То есть через 1,5 месяца стажёр сдаёт тестовое задание (собранный на типовых компонентах сайт), а через 3 уже вливается в команду.

Более короткие сроки обучения приводят к тому, что разработчики пишут крайне неоптимальные вещи с точки зрения **Bitrix Framework** или испытывают трудности при решении типовых задач. Студия выросла с 5 до 20 человек. Всех обучали сами.

## Как правильно построить процесс изучения?

Не простой вопрос, так как зависит во многом от стоящих перед человеком задач, наличия временных возможностей и начального уровня знаний.

**Идеальный вариант.** Пошаговое изучение всех курсов в линейке [сертификационных курсов](#) от Контент-менеджера через курсы Администратор до курса Разработчик Bitrix Framework. С одновременным изучением API и пользовательской документации.

**Реальный вариант.** Как правило, - это неделя на изучение, и потом работа над новым проектом в рамках веб-студии или на фрилансе. В этом случае необходимо знать интерфейс программы (глава [Элементы управления](#) курса Контент-менеджер) и данный курс. При работе впоследствии с проектом - постоянное обращение к документации для разработчиков, пользовательской документации и другим учебным курсам.

### **Цитатник веб-разработчиков.**

#### **Виктор Векслер:**

Мы выбрали подход кейсов в обучении, стажеры получают 5-10 кейсов в течении первого месяца, которыми и занимаются, при этом выделяется куратор, который общается, поддерживает и направляет стажера. При этом стажер учится на реальных задачах и не сильно мешает куратору.

Самое удобное, расписать заранее мини ТЗ и выдавать верстку для работы (в случае, если стажер - будущий программист), в случае верстальщика - шаблоны на верстку. При этом, после того как он



сделает свою верстку, он должен получить профессиональную уже сделанную работу.

Любой соискатель при собеседовании получает тестовое задание, оно сложное (для начинающего). Он получает 2 psd шаблона, дистрибутив Битрикса с окружением и мануал по интеграции. Его задача разобраться и интегрировать данные шаблоны в Битрикс, если он не может этого делать, мы его не принимаем.

С первого дня, стажер получает первый кейс - это сайт турфирмы, основная задача здесь разобрать инфоблоки и немного познакомится с Битриксом вообще. На данный проект в среднем уходит 1 неделя. Далее 2-ой кейс - это снять реальный сайт и сделать его полную копию на Битриксе (в среднем 3-4 дня). Далее пишет элементарный компонент - один день (может меньше). После, если у нас есть легкие проекты, он работает как подмастерье у программиста, если нет делает кейс интернет-магазина, если он осилил его, то он уже может переходить на сайты визитки.

Считаю такой подход очень продуктивным, на них уже работали порядка 20 сотрудников.

[Скачать кейсы](#)

## Сколько времени уйдет на освоение Bitrix Framework?

Это зависит от уровня стартовой подготовки и от способностей обучаемого. И в немалой степени зависит не только от знаний, сколько от умения отдаляться от прошлого опыта и смотреть на все новыми глазами.

В целом опыт подготовки специалистов говорит, что программист, знающий PHP, в состоянии за неделю-две изучить систему на основе штатных онлайн курсов и начать разрабатывать простенький сайт на основе стандартного функционала. Обычно через полгода человек начинает уже работать абсолютно самостоятельно на проектах любого уровня сложности.

**⚠ Примечание:** Bitrix Framework активно развивается. Его нельзя изучить один раз и жить на этом знании все оставшееся время. Надо постоянно быть в курсе событий.

**⚠ Внимание!** Приведенные в курсе примеры работы с API могут устаревать в силу постоянного развития системы. Отдел документации отслеживает это развитие, но задержки с внесением изменений в курсы могут быть.



## Где брать информацию?

### Цитатник веб-разработчиков.

**Степан Овчинников:** Битрикс огромен. Перечень вопросов, которые человек задает сразу, на первом проекте, относительно мал. Но вот объем знаний, нужных на втором этапе погружения, когда делается нетривиальное, действительно очень велико. Его нельзя наработать быстро.

Все продукты компании «1С-Битрикс» сопровождаются полной документацией, которая расположена на сайте в разделе [Документация](#).

Документация для продукта «1С-Битрикс: Управление сайтом» включает в себя:

- [Онлайн документация для пользователей](#)

**Основные сведения**

"1С-Битрикс: Управление сайтом" - технологическое ядро для создания и управления сайты. Использование этой системы управления содержимым сайта - экономичный способ разработки, поддержки и развития интернет-проекта.

Программный продукт "1С-Битрикс: Управление сайтом" может быть использован корпоративными заказчиками и индивидуальными разработчиками в качестве фундамента для создания новых проектов и управления уже существующими сайтами.

"1С-Битрикс: Управление сайтом" позволяет создавать неограниченное количество сайтов с применением одной копии (лицензии) продукта, размещая ядро и базу данных системы в единственном экземпляре на сервере.

**Основные возможности "1С-Битрикс: Управление сайтом":**

- управление структурой и содержанием сайта;
- публикация новостей, пресс-релизов и другой часто обновляемой информации;
- управление показом рекламы на сайте;
- создание и управление форумами;
- рассылка сообщений группам подписчиков;
- учет статистики посещений;
- контроль за ходом рекламных кампаний;
- осуществление других операций по управлению интернет-проектом.

Продукт позволяет минимизировать расходы на сопровождение веб-сайта за счет простоты управления статической и динамической информацией. Для управления проектом вам не потребуются дополнительные услуги специалистов в области веб-дизайна. С помощью "1С-Битрикс: Управление сайтом" управлять веб-сайтом сможет штатный сотрудник компании, обычный пользователь персонального компьютера без специальных навыков программирования и HTML-верстки.

Справочная документация представляет собой описание интерфейса системы управления сайтом, ее основных модулей и наиболее типичных операций. Также в ней содержится полное описание прикладного программного интерфейса.

Рекомендуется таким группам пользователей, как **Контент-менеджер** и



## Администратор

- [Онлайн документация для разработчиков](#)

Класс	Описание
<a href="#">CMain</a>	Основной класс.
<a href="#">CDatabase</a>	Класс для работы с базой данных.
<a href="#">CDBResult</a>	Класс результата выполнения запроса.
<a href="#">CAgent</a>	Класс для работы с агентами.
<a href="#">CEvent</a>	Класс для отправки почтовых событий.
<a href="#">CEventMessage</a>	Класс для работы с шаблонами почтовых событий.
<a href="#">CEventType</a>	Класс для работы с типами почтовых событий.
<a href="#">CFile</a>	Класс для работы с файлами и изображениями.
<a href="#">CUUser</a>	Класс для работы с пользователями.
<a href="#">CGroup</a>	Класс для работы с группами пользователей.
<a href="#">CSite</a>	Класс для работы с сайтами.
<a href="#">CLanguage</a>	Класс для работы с языками.
<a href="#">C MainPage</a>	Класс для использования на индексной странице портала.
<a href="#">CMenu</a>	Обработка меню.
<a href="#">CModule</a>	Управление модулями.
<a href="#">COption</a>	Класс для работы с параметрами модулей.
<a href="#">CPageOption</a>	Класс для работы с параметрами действующими только на странице.
<a href="#">CPageCache</a>	Класс для кэширования HTML.
<a href="#">CPHPCache</a>	Класс для кэширования PHP переменных и HTML.
<a href="#">CRatings</a>	Класс для работы с рейтингами.
<a href="#">CUUserFieldEnum</a>	Класс для работы с пользовательскими полями типа "список".

Документация, предназначенная для технических специалистов со знанием PHP и HTML. В ней содержатся сведения о технологиях и основных принципах, заложенных в систему, описание классов и функций.

Рекомендуется таким группам пользователей, как **Администратор** и **Разработчик**.

- [Документация в файлах формата .chm](#)

Содержит копию онлайн документации в формате справки Windows (.chm). Данная документация обновляется реже, чем источник на сайте. Удобно использовать при отсутствии интернета.



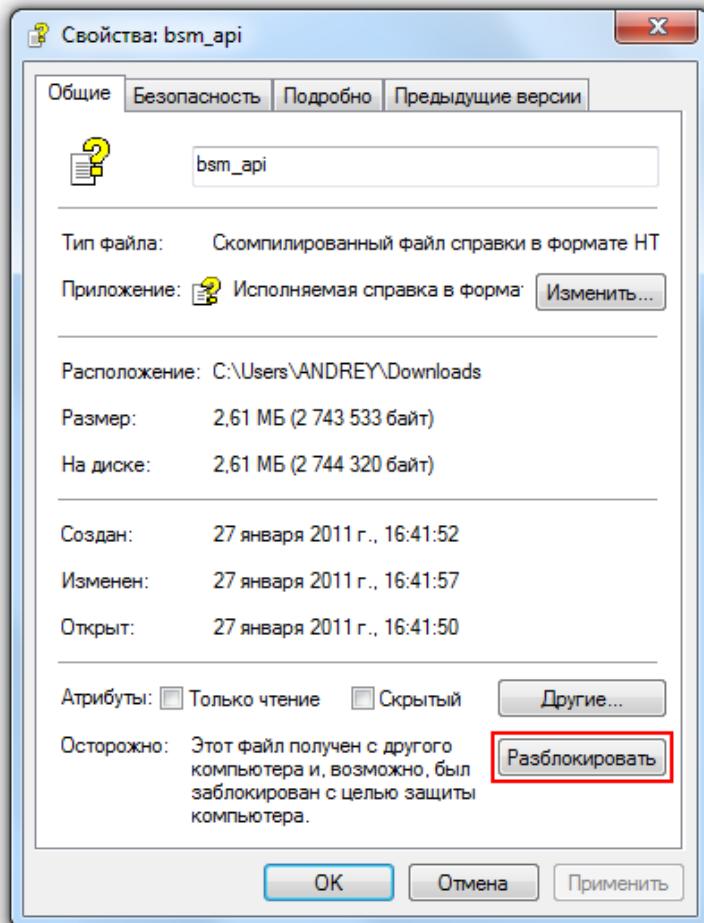
**Внимание!**



Если вы не видите содержимое файла формата .chm, то причина - настройки безопасности операционной системы: скачанные из интернета файлы находятся под наблюдением системы, которая ограничивает доступ к ним.

Чтобы отключить подобное отношение к файлу необходимо:

- перейти в свойства файла на вкладку **Общие**. Внизу у Вас отобразится сообщение о том, что этот файл был заблокирован системой с целью защиты компьютера.
- нажать кнопку **Разблокировать**.



Отсутствие кнопки **Разблокировать** возможно в двух случаях:

- Файл лежит не локально, а на сетевом ресурсе.
- Если файл лежит на локальном диске, но путь к нему содержит спецсимволы (# и прочие).

Рекомендуется таким группам пользователей, как **Контент-менеджер**, **Администратор** и **Разработчик**.

- [Онлайн Курсы](#)



The screenshot shows the 1C-Bitrix Content Manager interface. On the left, there's a sidebar titled "Учебный курс" (Training Course) with a graduation cap icon. Below it is a toolbar with icons for checkmark, list, folder, and print. The main content area is titled "Контент-менеджер" (Content Manager). A yellow header bar says "Примеры решения типовых задач" (Examples of solving typical tasks) and "2 / 97". The main content area contains text about how most solutions for typical tasks are described in the course texts and links to specific pages for each example.

Большая часть решений типовых задач Контент-менеджера описана в текстах курса, в соответствующих местах. Ниже приведены ссылки на такие страницы.

- [Как добавить новость на сайт](#)
- [Как наполнить фотогалерею](#)
- [Как наполнить видеотеку](#)
- [Как вернуть потерянные данные в случае неприменимой ситуации](#)
- [Как настроить рабочий стол и разместить на нем средства для вывода различных данных](#)
- [Как создать раздел или страницу](#)
- [Как ограничить доступ к странице или разделу](#)
- [Как создать и редактировать пункты меню сайта](#)
- [Как создать таблицу](#)
- [Как работать с гиперссылками](#)
- [Как добавлять на страницу заранее подготовленный текст или код](#)
- [Как быстро отредактировать несколько элементов](#)
- [Как вставить видео, размещенное на другом сервере, например, на YouTube](#)

Методические пособия по работе с продуктом, которые включают в себя описание и примеры работы с системой.

Рекомендуется таким группам пользователей, как **Контент-менеджер, Администратор и Разработчик**

- [Частые вопросы](#)



## Частые вопросы

- » Вопросы по установке и настройке продукта (36)
  - » Производительность (2)
  - » Работа с меню (4)
  - » Работа с модулем инфоблоков (11)
  - » Ошибки при установке и работе продукта (45)
    - » Интеграция с 1С (8)
    - » Ошибки PHP (6)
    - » Ошибки базы данных (12)

## Работа с меню

- Выпадающее меню не отображается над флеш (flash) анимацией.
- Как правильно задать поле "Условие" для пункта меню?
- Как создать выпадающее верхнее меню?
- Как добавить пункт меню в административный раздел?

### Выпадающее меню не отображается над флеш (flash) анимацией.

Проблема попадания выпадающего меню под объект Flash решается с установкой wmode="transparent" для тега OBJECT или аналогичного ему параметра для тега EMBED.

Пример:

```
<object codebase="http://download.macromedia.com/pub/shock-
wave/cabs/flash/swflash.cab#version=7,0,19,0" height="199" width="543" align="top"
classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000">
<param NAME=wmode VALUE=transparent>
<param value="/bitrix/templates/13107_139365/images/name.swf" name="movie">
<param value="high" name="quality">
</object>
```

[Наверх](#)

Подборка решений наиболее часто встречающихся проблем при работе с продуктом в удобном представлении.

Рекомендуется таким группам пользователей, как **Администратор** и **Разработчик**

- [Учебные видеоролики](#)

Видеоролики демонстрируют основные функциональные возможности продукта.

**⚠ Внимание!** Материалы роликов могут отставать от текущих версий продуктов.

Рекомендуется таким группам пользователей, как **Контент-менеджер** и **Администратор**.

При работе с **Bitrix Framework** очень большое значение имеет описание **API**. К сожалению, составление описаний API нового функционала никогда не выходит одновременно с функционалом. В этом случае рекомендуем воспользоваться уроком [Если нет описания API](#).

## Сообщество разработчиков

### Цитатник веб-разработчиков.

**Евгений Смолин:** Ну а вам совет - не бойтесь Битрикса, Битрикс это всерьез и надолго. Начинайте работать с ним, появятся вопросы или не пожелаете нанимать разработчиков, то обращайтесь к форуму, в ТП. Почти всегда помочь оказываю, причем оказывают помочь такие монстры, что даже опытным разработчикам интересно смотреть на решения.

За время развития продукта сложилось свое сообщество разработчиков на **Bitrix Framework**. Это сообщество представлено на сайте компании «1С-Битрикс» и обменивается опытом работы и результатами работы на форуме компании, в группах социальной сети и на собственных сайтах.

**⚠ Совет:** Пока вы не овладеете в полной мере [терминологией Bitrix Framework](#), рекомендуем при описании проблемы избегать терминологии, известной вам по другим CMS.

На [форуме](#) вы можете получить консультации у опытных создателей сайтов на платформе **Bitrix Framework**. Как и на форуме любого другого сообщества не приветствуются оскорбительные и некорректные высказывания в чей-либо адрес. При уважительном отношении к сообществу и корректном описании вашей проблемы, вы обязательно получите поддержку у сообщества.

**⚠ Внимание!** Форум сайта не является службой техподдержки. Техподдержка работает согласно [регламенту](#) и в другом [разделе сайта](#).

Рекомендуется познакомиться с опытом разработок сайтов на системе **Bitrix Framework**, который описывается в [блогах](#) разработчиков **Bitrix Framework** и в [группах](#) Социальной сети сайта компании 1С-Битрикс.

Наиболее актуальные группы сети:

- [Пожелания к доработке «1С-Битрикс»](#)
- [Академия «1С-Битрикс» выпускников](#)
- [Ошибки системы «1С-Битрикс»](#)
- [Примеры частных решений](#)



Многие опытные разработчики сайтов на основе нашей платформы ведут свои блоги и разделы на сайтах, где делятся опытом. Вот некоторые из них:

- [www.bitrixonrails.ru](http://www.bitrixonrails.ru)
- [www.web4us.ru/poweredbix](http://www.web4us.ru/poweredbix)
- [www.d-it.ru](http://www.d-it.ru)
- [www.info-expert.ru/articles/bitrix](http://www.info-expert.ru/articles/bitrix)
- [blog.itconstruct.ru/category/1s-bitriks/nyuansy](http://blog.itconstruct.ru/category/1s-bitriks/nyuansy)
- [www.bxdev.ru](http://www.bxdev.ru)
- [yunaliev.ru/category/development/1c-bitrix](http://yunaliev.ru/category/development/1c-bitrix)
- [bitrixgems.ru](http://bitrixgems.ru)
- [alexvaleev.ru](http://alexvaleev.ru)
- [blog.sokov.org/category/bitrix](http://blog.sokov.org/category/bitrix)
- [life.screenshots.ru/category/the-code-inside/vnutri-koda](http://life.screenshots.ru/category/the-code-inside/vnutri-koda)
- [bitrix-help.ru/db/](http://bitrix-help.ru/db/)
- [blog-o-bitrix.ru/](http://blog-o-bitrix.ru/)

Не малая часть примеров и способов работы с **Bitrix Framework** в этом курсе взята с этих сайтов.

### **Для тех, кто переходит на Bitrix Framework с других платформ**

Для программистов, переходящих на **Bitrix Framework** с других платформ и CMS возникают дополнительные сложности, вызванные «давлением» предыдущего опыта.

Чтобы научиться эффективно работать в **Bitrix Framework**, нужно не сравнивать то, что вы знаете по другим системам, а стараться понять как то или иное реализуется в этой системе. В плане обучения «сравнительный» подход не работает. Просто отвлекитесь от старых знаний и изучите новую систему используя только знания **PHP** и сайтостроения, а не сравнивая идеологии и технологии. Легче будет освоить. А сравнивать будете потом, когда освоите **Bitrix Framework**.

Зато у вас может возникнуть преимущество знания двух систем. В последнее время заказчики нередко ставят задачу миграции сайтов с других систем на **Bitrix Framework**. Настолько нередко, что партнеры компании «1С-Битрикс» даже разрабатывают специальные решения под эти задачи. (Например: [CMS Конвертер](#).)

Прямое сравнение **Bitrix Framework** и других систем далеко не всегда корректно. Тем не менее, такие вопросы возникают и потому приведем некоторые мнения, высказанные партнерами компании «1С-Битрикс» и программистами, работающими на **Bitrix Framework**.



## Bitrix Framework и Drupal

Основной структурной единицей CMS *Drupal* служит узел (**node**). По сути дела, любая страница сайта на *Drupal* (за исключением служебных) - это либо список анонсов узлов, либо полное отображение одного узла. Вывод любой страницы может сопровождаться выводом дополнительных блоков, но, так или иначе, они являются вторичными по отношению к **node**.

В **Bitrix Framework** реализована идеология инфоблоков, которые структурно можно уподобить таблице в базе данных. [Инфоблок](#) представляет собой совокупность объектов, обладающих одинаковым набором свойств.

Все инфоблоки равноправны в том смысле, что любой инфоблок (или даже несколько инфоблоков) может использоваться для вывода как в основной области страницы, так и в дополнительных областях. Таким образом, **node** в CMS *Drupal* является лишь частным случаем инфоблока - и, фактически, в этой системе имеется только один инфоблок, тогда как в **Bitrix Framework** их может быть неограниченное количество.

## Bitrix Framework и Joomla

- Шаблоны сайта в **Bitrix Framework** примерно соответствуют по концепции шаблонам сайта в **Joomla**.
- Создание шаблона сайта для **Bitrix Framework** по готовой верстке заключается в выделении блоков и размещении вместо этих блоков компонентов. Далее эти компоненты настраиваются на источник данных и для них редактируются шаблоны вывода в соответствии с версткой сайта.
- В шаблоне сайта для **Bitrix Framework** нет позиций под модули с номером, как в **Joomla**. И программист не может из административной части указать какой модуль в какую позициюставить, просто изменив число. Размещение компонентов в **Bitrix Framework** реализовано по-другому.
- Модули в **Bitrix Framework** - это сущность для объединения необходимых программистам функций "в одном флаконе" и разделения по редакциям. Аналог в **Joomla** - расширения.
- Модуль в **Joomla** - это компонент в **Bitrix Framework**. Компонент получает данные откуда-то и выводит их в нужном виде. За правильный вывод отвечает шаблон компонента.

У одного компонента может быть несколько шаблонов, по-разному выводящих информацию. Например, если вам нужно вывести на сайте новости и статьи, вы создаете две папки на диске, кидаете в каждую из них по комплексному компоненту **Новости** и настраиваете их на источник данных. Дальше вы можете перерабатывать шаблоны.



- В **Bitrix Framework** разделяются динамические и статические данные. Есть чисто динамические блоки, например, каталог товаров. Есть статические. Есть смешанные. Однако, для того, чтобы вывести динамическую информацию, например, каталог товаров, вы должны сделать для него «домик» - папку на диске, в которой будет находиться комплексный компонент «каталог» и обрабатывать обращения к динамической информации.

**⚠ Примечание:** В социальной сети компании "1С-Битрикс" есть группа [OpenSource.Конвертер](#), в которой можно познакомиться с опытом других разработчиков по миграции сайтов с других систем на **Bitrix Framework**.

## Глоссарий

Каждая платформа имеет свои термины, используемые в работе. Чтобы без проблем ориентироваться в документации, учебных курсах, при общении с разработчиками желательно объяснять ваши трудности не «на пальцах», а на языке понятном специалистам. В главе в помощь вам приведен глоссарий терминов системы и несколько общих терминов, активно используемых в системе.

Термин	Описание
Портал	Один набор файлов, хранящихся в каталоге <b>/bitrix/modules/</b> , и одна копия базы. Портал включает в себя один или более сайтов. Синонимом данного термина может служить: «экземпляр продукта», «одна инсталляция системы», «одна копия системы».
<a href="#">Сайт</a>	Совокупность таких понятий как: <ul style="list-style-type: none"><li>• <b>учетная запись в базе данных</b> - создается в административном меню <b>Сайты</b>, включает в себя следующие основные параметры:<ul style="list-style-type: none"><li>◦ <b>идентификатор</b> - набор символов, идентифицирующий сайт;</li><li>◦ <b>доменное имя</b> - одно или более доменное имя сайта;</li><li>◦ <b>папка сайта</b> - путь к каталогу в котором будет храниться публичная часть сайта;</li><li>◦ <b>язык сайта;</b></li><li>◦ <a href="#"><b>формат даты;</b></a></li><li>◦ <a href="#"><b>формат времени;</b></a></li><li>◦ <b>URL</b> - протокол и доменное имя по умолчанию (например, <a href="http://www.site.ru">http://www.site.ru</a>);</li></ul></li></ul>

- **DocumentRoot** - если [многосайтость](#) организована на разных доменах, то в данном параметре должен храниться путь в файловой системе сервера к корню сайта;
- **условия подключения шаблонов** - каждый сайт может иметь один или более шаблонов для отображения скриптов своей публичной части, каждый такой шаблон может быть подключен по тому или иному условию;
- **публичная часть** - совокупность скриптов (страниц), лежащих в «папке сайта» и принадлежащих этому сайту;
- **настройки** - каждый модуль может иметь ряд настроек, связанных с сайтом, например у модуля «Информационные блоки» эти настройки представляют из себя привязку информационного блока к тому или иному сайту, у модуля «Техподдержка» - привязку статуса, категории и т.п. к сайту.

Синонимом можно считать - **дизайн сайта, скин сайта**. Для показа одного сайта можно использовать несколько различных шаблонов. В шаблон сайта входят:

- набор файлов в каталоге **/bitrix/templates/*ID\_шаблона*/**, где **ID\_шаблона** - поле **ID** в форме редактирования шаблона сайта. Структура данного каталога:
  - **/components/** - каталог с компонентами, принадлежащими тому или иному модулю;
  - **/lang/** - языковые файлы, принадлежащие как данному шаблону в целом, так и отдельным компонентам;
  - **/images/** - каталог с изображениями данного шаблона;
  - **/page\_templates/** - каталог с шаблонами страниц и их описанием, хранящимся в файле **.content.php**;
  - **/include\_areas/** - каталог с файлами - содержимым включаемых областей;
  - **header.php** - пролог данного шаблона;
  - **footer.php** - эпилог данного шаблона;
  - **styles.css** - CSS стили, используемые на страницах сайта, когда используется данный шаблон;
  - **template\_styles.css** - CSS стили, используемые в самом шаблоне;
  - **.тип меню.menu\_template.php** - шаблон [вывода меню](#) соответствующего типа;
  - **chain\_template.php** - шаблон по умолчанию для

## Шаблон сайта

вывода навигационной цепочки;

- а также ряд других вспомогательных произвольных файлов, входящих в данный шаблон.

Раздел сайта	Каталог в файловой системе сервера. В <b>Bitrix Framework</b> структура сайта - это файловая структура сервера, поэтому страницы сайта - это файлы, а разделы сайта - соответственно каталоги.
--------------	--

Является частью того или иного модуля, представляет из себя логически завершенный код, хранящийся в одном файле, принимающий ряд параметров, выполняющий ряд действий и выводящий какой-либо результат (например, в виде HTML кода). Использование компонента является наиболее предпочтительным способом организации вывода информации как в публичной, так и в административной частях. В понятие **компонент** входят следующие элементы:

### Компонент

- **основной файл** - подключается с помощью функции [CMain::IncludeFile](#), представляет из себя скрипт на языке PHP, реализующий логику компонента;
- **языковые файлы** - используются в компоненте для представления его результата на различных языках;
- **описание** - это PHP код, используемый визуальным редактором для управления компонентом и его параметрами;

Компонент 2.0	Компоненты являются блоками, с помощью которых строится публичная часть сайта.  Основные особенности компонентов версии 2.0: <ul style="list-style-type: none"><li>• В компонентах 2.0 разделены логика и представление. Для одной логики может быть несколько представлений, в том числе зависящих от шаблона текущего сайта. Логика - это сам компонент, представление - это шаблон вывода компонента.</li><li>• Компоненты централизованно хранятся в одной папке. Это обеспечивает большую целостность и понятность структуры сайта. Папка доступна для обращений, а значит, компонент и его шаблоны могут легко подключать свои дополнительные ресурсы.</li><li>• Компонент хранит все, что ему нужно для работы, в своей папке. Поэтому компоненты можно легко переносить между</li></ul>
---------------	---

	проектами. Компонент неделим.
	<ul style="list-style-type: none"><li>• Компоненты бывают простые и комплексные. Простые служат для создания различных областей на одной странице (список новостей, топ элементов каталога, случайное фото и т.д.). А комплексные используются для создания полнофункциональных разделов сайта (полный каталог товаров, новостной раздел, форум и т.д.).</li></ul>

Используется в функции **CMain::IncludeFile** в качестве первого параметра и представляет из себя путь к основному файлу компонента относительно одного из следующих каталогов (в порядке убывания приоритета):

#### Путь к компоненту

- /bitrix/templates/*ID* шаблона сайта/;
- /bitrix/templates/.default/;
- /bitrix/modules/*ID* модуля/install/templates/

Где:

**ID шаблона сайта** - идентификатор текущего шаблона сайта,  
**ID модуля** - идентификатор модуля которому принадлежит компонент.

Включаемые области	Это специально выделенная область на странице сайта, которую можно редактировать отдельно от основного содержания страницы. Реализуется с помощью специального программного компонента.
--------------------	---

#### Язык

Это учетная запись в базе данных доступная для редактирования в административном меню на странице **Языки интерфейса** со следующими полями:

- Идентификатор;
- Название;
- Формат даты;
- Формат времени;
- Кодировка.

Как в публичной, так и административной частях, язык в первую очередь используется в работе с языковыми файлами. В административной части язык определяет формат времени, даты, кодировку страниц (в публичной - данные параметры определяются настройками сайта).

## Языковой файл

Представляет из себя PHP скрипт, хранящий переводы языковых фраз на тот или иной язык. Данный скрипт состоит из массива \$MESS, ключи которого - идентификаторы языковых фраз, а значения - переводы на соответствующий язык. Для каждого языка существует свой набор языковых файлов, хранящихся как правило в каталогах /lang/. Языковые файлы, как правило, используются в [административных скриптах](#) модулей и в компонентах. Для работы с языковыми файлами предназначен модуль **Перевод**.

Это файл, хранящийся в одном из каталогов:

- /bitrix/templates/**ID** шаблона сайта/page\_templates/
- /bitrix/templates/.default/page\_templates/

## Шаблон страницы

Данный файл представляет из себя заготовку публичной страницы. Как правило, используется при создании новой страницы в модуле **Управление структурой**. Для задания порядка сортировки шаблонов страниц используется файл **.content.php** находящийся в одном из вышеуказанных каталогов.

## Пролог

В общем случае, под данным термином понимается верхняя левая часть страницы.

Для публичной части, пролог соответствующего шаблона сайта хранится в файле **/bitrix/templates/**ID** шаблона сайта/header.php**.

Для административной части - пролог храниться в файле **/bitrix/modules/main/interface/prolog\_admin.php**.

В свою очередь пролог может быть разделен на служебную и визуальную части. В служебной части подключаются все необходимые классы, создаются соединения с базой, создаются ряд служебных экземпляров объектов, таких как \$USER, \$APPLICATION и т.д. В визуальной части выводится верхняя левая часть страницы.

Если в публичной части необходимо подключить неразделенный пролог, то используем следующий код:

```
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
```

Если по тем или иным причинам необходимо разделить пролог на служебную (**prolog\_before.php**) и визуальную (**prolog\_after.php**) части, то используем следующий код:

```
require($_SERVER["DOCUMENT_ROOT"].  
"/bitrix/modules/main/include/prolog_before.php");  
...  
require($_SERVER["DOCUMENT_ROOT"].  
"/bitrix/modules/main/include/prolog_after.php");
```

В общем случае под данным термином понимается нижняя правая часть страницы.

Для публичной части, эпилог соответствующего шаблона сайта хранится в файле **/bitrix/templates/ID шаблона сайта/footer.php**.

Для административной части - эпилог хранится в файле **/bitrix/modules/main/interface/epilog\_admin.php**.

В свою очередь эпилог может быть разделен на служебную и визуальную части. В служебной части производится ряд таких действий, как: отсылка почтовых сообщений, отработка обработчиков события [OnAfterEpilog](#) и др. В визуальной части выводится нижняя правая часть страницы.

### Эпилог

Если в публичной части необходимо подключить неразделенный эпилог, то используем следующий код:

```
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");
```

Если по тем или иным причинам необходимо разделить эпилог на визуальную (**epilog\_before.php**) и служебную (**epilog\_after.php**) части, то используем следующие коды:

```
require($_SERVER["DOCUMENT_ROOT"].  
"/bitrix/modules/main/include/epilog_before.php");  
...  
require($_SERVER["DOCUMENT_ROOT"].  
"/bitrix/modules/main/include/epilog_after.php");
```

### Тело страницы

Тело страницы это часть PHP/HTML кода, расположенного в скрипте между подключениями пролога и эпилога. Тело страницы не является частью шаблона сайта и представляет из себя индивидуальное содержимое публичной либо административной страницы.

### [Навигационная](#)

Это элемент дизайна, предназначенный в первую очередь для улучшения навигации по сайту. Навигационная цепочка [выводится](#)

цепочка

в визуальной части пролога и состоит из заголовков разделов сайта с соответствующими ссылками на них. Помимо заголовков разделов, добавляемых автоматически, вы также можете [добавлять](#) произвольные пункты в навигационную цепочку.

Административная часть	Это множество скриптов, в которых подключен административный пролог и эпилог. Их подключение подразумевает:
	<ul style="list-style-type: none"><li>• данная страница не принадлежит ни одному сайту;</li><li>• в тоже время она принадлежит какому-либо модулю;</li><li>• имеет строго определённый административный интерфейс;</li><li>• все параметры локализации данной страницы зависят от выбранного текущего языка;</li><li>• на этой странице используется дополнительная проверка прав, задаваемых в настройках соответствующего модуля.</li></ul>

В общем случае, под данным термином понимается множество скриптов, у которых подключен пролог и эпилог одного из шаблонов сайта. Их подключение подразумевает:

Публичная часть	<ul style="list-style-type: none"><li>• данная страница принадлежит какому либо сайту;</li><li>• имеет интерфейс текущего шаблона сайта;</li><li>• все параметры локализации данной страницы зависят от текущего сайта.</li></ul>
-----------------	---

Режим Правка	Режим, при котором текущая страница публичного раздела отображается в особом виде: выделены используемые компоненты, включаемые области, редактируемые области и т.д. Каждая такая область имеет набор кнопок для быстрого перехода к редактированию данного элемента страницы.
--------------	---

Индексная страница (файл)	Это имя файла, который будет использован веб-сервером в случае, если запрашиваемый URL заканчивается на спэш и не содержит в себе имени файла. Порядок, в котором будутискаться индексные страницы для различных веб-серверов, задается по разному, например:
---------------------------	---

- **Apache** - в файле **httpd.conf**, параметр `DirectoryIndex`;
- **IIS** - в свойствах сайта, закладка **Documents > Enable default content page**;

К сожалению, в PHP значение данного параметра недоступно, поэтому для определения индексной страницы в коде необходимо

пользоваться функцией [GetDirIndex](#).

Бюджет (аккаунт) пользователя	<p>Синонимами данного термина можно считать «регистрация пользователя», «логин пользователя». Это запись в базе данных, содержащая параметры зарегистрированного пользователя с такими обязательными полями как:</p> <ul style="list-style-type: none"><li>• логин;</li><li>• пароль;</li><li>• E-Mail.</li></ul> <p>А также ряд дополнительных полей, содержащих личную информацию о пользователе, информацию по его работе, административные заметки.</p>
<b><u>Мета-тэг</u></b>	Мета-тэг это один из элементов HTML, позволяющий задать информацию о странице. Указать, например, кодировку страницы, ее ключевые слова, автора, задать краткое описание страницы. Как правило, содержимое мета-тэгов используется в служебных целях, например, роботами поисковых систем, индексирующих сайт. Мета-тэг задается внутри тэга <b>&lt;head&gt;</b> .
Локализация	Подразумевает представление информации в переводе на соответствующий язык, в соответствующей кодировке данного языка и с использованием соответствующих форматов представления данных, характерных для этого языка (например, дата, время, денежных единицы, числа и т.д.).
Хост	В применении к функциям главного модуля, под данным термином понимается доменное имя или IP адрес, посредством которого можно обратиться к тому или иному сайту.
Доменное (домен) имя	Строго говоря, это одно из полей DNS таблицы ( <b>domain name service</b> ), содержащее в себе строго структурированное имя интернет сайта, заданное по определённым правилам. Основная задача DNS таблицы - это ассоциация доменных имен с IP адресами сайтов. Пример доменного имени: 1c-bitrix.ru
IP адрес	Это "имя" компьютера в сети, заданное по правилам протоколов TCP/IP. IP адрес состоит из четырех октетов, часть из которых идентифицирует подсеть, в которой находится компьютер, а часть - непосредственно этот компьютер в рамках соответствующей подсети. Пример IP адреса: 198.63.210.79



Accept-Language	Набор языков, установленных в браузере посетителя сайта. К примеру, для MS Internet Explorer их можно выставить в меню Tools > Internet Options > General > Languages. Для Mozilla Firefox: Инструменты > Настройки > Основное > Языки.
Время в Unix-формате	Количество секунд, прошедшее с 1 января 1970 года, с точностью до микросекунды. На сегодняшний день, время в Unix-формате может фиксироваться только до 2038 года.
Права в Unix системах	<p>В Unix-подобных операционных системах (являющихся доминирующими на сегодняшний день для интернет серверов) поддерживаются три вида прав - чтение, запись и выполнение, которые присваиваются каждому файлу или директории. Права эти повторяются три раза: для владельца файла, для группы пользователей, к которой владелец принадлежит, и для всех остальных пользователей. Как правило, права указываются в числовом формате:</p> <ul style="list-style-type: none"><li>• 4 - чтение;</li><li>• 2 - запись;</li><li>• 1 - выполнение.</li></ul> <p>Сумма этих чисел дает окончательный набор прав, например, 6 - это чтение и запись, но без выполнения, 7 - все права, 5 - чтение и выполнение.</p> <p>Таким образом, например, право 764 будет означать: 7 - все права для владельца файла, 6 - чтение и запись для группы пользователей, к которой принадлежит владелец файла и 4 - чтение для всех остальных пользователей.</p> <p>В PHP все права задаются в виде восьмиричных чисел, поэтому их надо задавать с обязательным указанием префикса - 0. Пример: 0755.</p>
Путь относительно корня	Путь к файлу, начинающийся от каталога, указанного в параметре <b>DocumentRoot</b> в настройках веб-сервера, заданный по правилам формирования URL-адресов. Пример: <a href="#">/ru/about/index.php</a>
Полный путь	Включает в себя протокол, домен, порт и путь относительно корня к странице (каталогу). Пример: <a href="http://www.bitrixsoft.ru/ru/about/index.php">http://www.bitrixsoft.ru/ru/about/index.php</a>
Абсолютный путь	Абсолютный путь к файлу включает в себя <b>DocumentRoot</b> и путь

относительно корня.

	Путь к корню сайта в файловой системе сервера. Задается в настройках веб-сервера, например:
DocumentRoot	<ul style="list-style-type: none"> <li>• для Apache - в файле httpd.conf, параметр <b>DocumentRoot</b>;</li> <li>• для IIS - в свойствах сайта, закладка <a href="#">Home Directory &gt; Local Path</a>.</li> </ul>

**Сабмит** Отправка данных HTML формы на сервер.

	Под термином понимается сессия PHP. Сессия может открываться в момент захода на сайт и закрывается при закрытии окна браузера. Также новая сессия открывается при авторизации пользователя, если закончить сеанс авторизации (разлогиниться) - сессия закрывается. Синонимом термина сессия можно считать один "заход на сайт".
--	---

Как правило данный термин применяют к тексту, в котором произведены следующие замены:

	Исходное	Результат
	<	&lt;
	>	&gt;
	"	&quot;
	&	&amp;

Подобные замены позволяют выводить текст внутри HTML кода, не опасаясь, что он будет интерпретирован браузером как часть этого HTML кода.

	В применении к SQL-запросам для Oracle версии, под данным термином подразумевается связывание имен переменных (либо полей таблицы) с их значениями. Как правило, подобная технология используется для полей типа <b>BLOB</b> , <b>CLOB</b> , <b>LONG</b> и т.п. предназначенных для хранения больших объемов данных.
--	--

**Дамп** В применении к базе данных, под термином подразумевается выгрузка содержимого и, возможно, структуры таблиц в файл в определённом формате, для их возможной дальнейшей загрузки

обратно в данную, либо любую другую базу. Для каждой базы данных существуют свои утилиты, позволяющие сделать дамп, например для MySQL утилита **mysqldump** позволяет выгрузить в формате обычных SQL запросов, для Oracle утилита **exp** позволяет выгрузить в своем внутреннем формате. В применении к переменным, дамп подразумевает отображение структуры и содержимого переменной в текстовом виде.

cron (крон)	В Unix-подобных операционных системах утилита <b>cron</b> позволяет организовать запуск скриптов по четко указанному расписанию.
Буферизация	Под этим термином в PHP понимается такой режим, при котором весь исходящий поток данных из PHP скрипта (например, HTML код) запоминается предварительно в памяти и не отдается браузеру пользователя. Буферизацию в PHP можно включить с помощью функции <b>ob_start</b> . В дальнейшем буферизацию можно отключить, например с использованием функции <b>ob_end_flush</b> , при этом все накопленные данные будут отосланы браузеру. Буферизация позволяет произвольно манипулировать исходящим потоком данных, на этом принципе основана <a href="#">технология отложенных функций</a> .
Постоянное соединение (persistent)	При создании соединения с базой, в памяти создаётся дескриптор данного соединения. Если соединение обычное, то после отработки скрипта этот дескриптор удаляется, если соединение <b>постоянное</b> , он остается и может быть использован другими процессами при необходимости. Достоинством постоянного соединения является то, что, как правило, времени на него работу требуется меньше. Но есть и недостаток - количество открытых постоянных соединений ограничивается в настройках базы данных и при превышении этого лимита посетитель не сможет зайти на сайт пока не освободятся новые соединения.
Почтовое событие	Это почтовое сообщение, имеющее свой тип и отправляемое по соответствующему почтовому шаблону. Почтовое событие инициализирует поля типа почтового события конкретными значениями. Порядок расположения этих полей в письме, а также текст письма, определяется почтовым шаблоном. Для создания почтового события предназначен класс <a href="#">CEvent</a> . Используется в <a href="#">Почтовой системе</a> .
Почтовый шаблон	Определяет текст почтового сообщения, а также порядок расположения полей (placeholder'ов), заданных в типе почтового события. Почтовые шаблоны доступны в административном разделе на

	<p>странице <b>Почтовые шаблоны</b> (Настройки системы &gt; Почтовые шаблоны). Для манипуляции почтовыми шаблонами предназначен класс <b>CEventMessage</b>. Используется в <b>Почтовой системе</b>.</p>
Тип почтового события	<p>Определяет набор специальных полей (placeholder'ов), которые могут быть использованы в почтовом шаблоне. В момент создания почтового события эти поля будут инициализированы конкретными значениями.</p> <p>Для манипуляции типами почтовых событий предназначен класс <b>CEventType</b>. Используется в <b>Почтовой системе</b>.</p>
Кастомизация	<p>Изменение логики работы компонента или шаблона компонента под частные задачи.</p>



## Глава 2. Архитектура продукта

### Цитатник веб-разработчиков.

*Степан Овчинников: совместимость новых версий Битрикса со всем, что было написано ранее, поддерживается разработчиками постоянно. Для партнеров, поддерживающих разные проекты, уверенность в такой совместимости является на мой взгляд одной из самых важных положительных сторон.*

Любое программное обеспечение, развиваясь, должно соответствовать заявленной изначально цели. Эту задачу решает архитектурное проектирование. Архитектура продукта - подход к проектированию, гарантирующий, что программное обеспечение будет отвечать своему предназначению.

*Архитектура программного обеспечения* - (англ. *software architecture*) — это структура программы или вычислительной системы, которая включает программные компоненты, видимые снаружи свойства этих компонентов, а также отношения между ними.

Архитектура Bitrix Framework решает следующие задачи:

- **Преемственность.** Каждый новый релиз продуктов поддерживает все предыдущие решения и технологии. Это позволяет осуществлять переход на новые версии продуктов сайтов, созданных на практически любой предыдущей версии.
- **Единство принципов работы** с любой версией и любым решением на базе системы.
- **Безопасность.** Архитектура позволяет создать достаточный уровень безопасности для сайтов любой направленности.
- **Масштабируемость.** Не наложено никаких ограничений на развитие проектов по мере роста контента, сервисов, числа пользователей.
- **Производительность.** Скорость работы системы зависит от качества настройки ее элементов, то есть в большей степени на производительность влияет уровень подготовки разработчика проекта, возможности хостинга.
- **Возможность развития системы** усилиями сторонних разработчиков. Архитектура не накладывает никаких ограничений на создание собственных модулей, компонентов, решений.

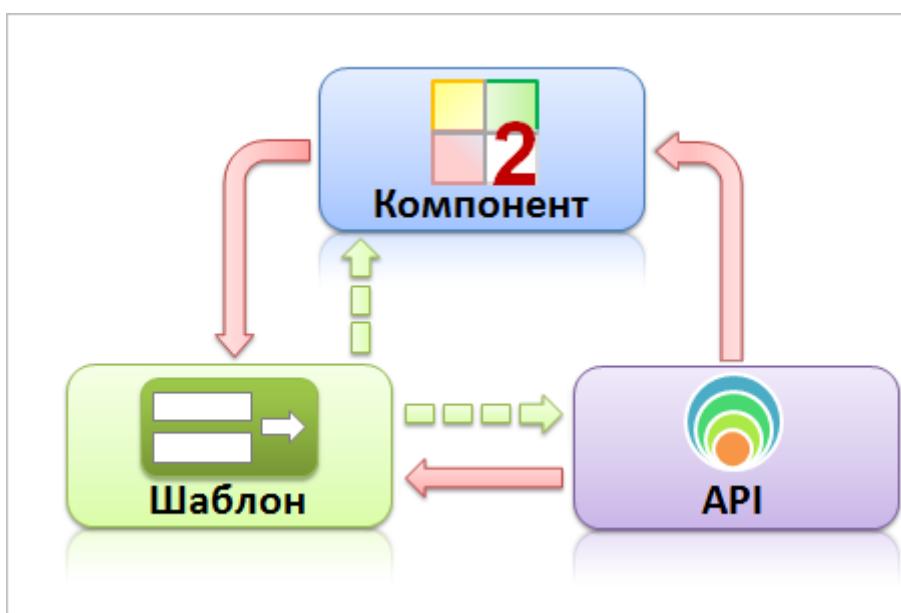
## Архитектура продукта

Цитатник веб-разработчиков.

**Степан Овчинников:** Я убежден, что логика, представления и данные в Битриксе разделены самым разумным для задач CMS образом.

## Архитектура MVC для Bitrix Framework

Шаблон MVC для Bitrix Framework:



- **Модель** - это API;
- **Представление** - это шаблоны;
- **Контроллер** - это компонент.

Сплошные линии - прямые связи, пунктир - косвенные связи.

## Немного теории

**MVC** (Model-view-controller, «Модель-представление-поведение», «Модель-представление-контроллер») — архитектура программного обеспечения, в которой модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента, так, что модификация одного из компонентов оказывает минимальное воздействие на другие компоненты.

Шаблон MVC позволяет разделить данные, представление и обработку действий пользователя на три отдельных компонента.

- **Модель (Model).** Модель предоставляет данные (обычно для **Представления**), а также реагирует на запросы (обычно от **Контроллера**), изменяя своё состояние.
- **Представление (View).** Отвечает за отображение информации (пользовательский интерфейс).

- **Поведение (Controller).** Интерпретирует данные, введённые пользователем, и информирует модель и представление о необходимости соответствующей реакции.

Важно отметить, что как **Представление**, так и **Поведение**, зависят от **Модели**. Однако **Модель** не зависит ни от **Представления**, ни от **Поведения**. Это одно из ключевых достоинств подобного разделения. Оно позволяет строить **Модель** независимо от визуального **Представления**, а также создавать несколько различных **Представлений** для одной **Модели**.

## Структура

**Bitrix Framework** обладает продуманной и удобной структурой, что заслуженно оценили многочисленные программисты и партнеры компании. По уровням архитектуры структуру можно описать так:

Bitrix Framework:	сайт:	компонент:	страница:
• модули	• шаблон	• вызов	• header
• компоненты	• компоненты	• параметры	• workarea
• файлы страниц	• страница	• шаблон	• footer

## Элементы структуры Bitrix Framework

### Модули

**Модуль** - это **модель данных и API** для доступа к этим данным. Статические методы классов модуля могут вызываться в компонентах, шаблонах, других модулях. Также внутри контекста **Bitrix Framework** могут создаваться экземпляры классов.

Несколько десятков модулей системы содержат набор функций, необходимых для реализации какой-то глобальной, большой задачи: веб-формы, работа интернет-магазина, организация социальной сети и другие. Модули также содержат инструментарий для администратора сайта для управления этими функциями.

 **Внимание!** На уровне ядра и модулей вмешательство в работу системы крайне не рекомендуется.

**Ядро продукта** - файлы, находящиеся в директории **/bitrix/modules/** а так же файлы системных компонентов: **/bitrix/components/bitrix/**.



## Компоненты

**Компонент** - это **контроллер и представление** для использования в публичном разделе. Компонент с помощью API одного или нескольких модулей манипулирует данными. Шаблон компонента (представление) выводит данные на страницу.

Компоненты входят в состав модулей, но решают более узкую, частную задачу — например, выводят список новостей или товаров. Вносить свои изменения в код продукта рекомендуется на уровне компонентов. Программист может модифицировать их как угодно, использовать свои наработки и использовать неограниченное число шаблонов на каждый из компонентов. На одной странице сайта может располагаться несколько компонентов, кроме того, их можно включать в шаблон сайта. Таким образом, программист имеет возможность собрать сайт как конструктор, после чего доработать необходимые компоненты для получения желаемого результата как в функциональном, так и в визуальном плане.

Чтобы работать с API нужно просто понять структуру компонентов **Bitrix Framework**.

 **Примечание:** Модуль - это набор каких-либо сущностей. Компонент - это то, что этими сущностями управляет.

Посмотрим на примере модуля **Инфоблоки**. Этот модуль представляет собой совокупность таблиц в базе данных и php-классов, которые могут проводить какие-либо операции с данными из таблиц (например, `CIBlockElement::GetList()` или `CIBlockElement::GetByID ()`). Компонентом является уже, например, **Новость детально**, который имеет собственные настройки (показывать дату, картинку и т.д. и т.п.) и работает с методами php-классов модуля.

## Страница

Страница представляет из себя PHP файл, состоящий из пролога, тела страницы (основной рабочей области) и эпилога. Формирование страницы сайта производится динамически на основе используемого шаблона страницы, данных выводимых компонентами и статической информации, размещенной на странице.

## Структура файлов

**Вопрос:** Почему допускается хранение контента в файловой системе, пусть даже статичного? Не место ли контенту в базе данных?

**Вадим Думбравану:** так удобнее, причем как разрабатывать, так и управлять.

## Файлы и База данных

**Bitrix Framework** реализован на файлах, что дает больше свободы разработчику сайта. Поскольку файл в системе - это просто исполняемый файл, то и исполнять он может что угодно: хоть собственный PHP-код программиста, хоть стандартные компоненты - в любом порядке. Как ни странно, эта полная свобода может напугать начинающего разработчика, но с опытом это проходит.

 **Примечание:** исполнение PHP это большое преимущество статической страницы **Bitrix Framework**.

Файлы можно править как по FTP, так и в SSH, не прибегая к дополнительным инструментам СУБД. Их легко копировать, перемещать, делать резервные копии и т.п. Строго говоря, вы можете весь контент хранить в БД. Но для простых статичных сайтов это будет явное усложнение и замедление.

Реализация на файлах кажется проблематичной в том плане, что от такой системы ожидается десятки тысяч файлов на диске. Обычно это не так. Динамическая информация (новости, каталог товаров, статьи) сохраняются в БД модулем **Информационные блоки**. Тогда для вывода, например, десятка тысяч товаров в интернет-магазине используется одна единственная физическая страница (файл). В этом файле вызывается компонент инфоблоков, который в свою очередь выбирает и выводит товары из базы данных.

Например, для каталога товаров действительно нужно создать папку на диске, но только одну, например, **/catalog**, поместить туда комплексный компонент и далее страницы товаров могут иметь вид, например: [http://\\*\\*\\*.ru/catalog/1029.html](http://***.ru/catalog/1029.html) Естественно, что эти адреса будут "мнимыми" и обрабатываться системой. Файлов под них в папке **/catalog** не создается.

Однако, для каждого товара будет создан файл в кэше, чтобы, при следующем обращении покупателя сервер не напрягался с запросами к БД. Это и позволяет запускать магазины уровня [Эльдорадо](#).

При должном умении публичная часть может состоять из десятка физических файлов. Весь контент может быть в инфоблоках, включая меню. Но обычно статические страницы (например, **О компании**) удобнее редактировать как файл, а не как запись БД. Но если таких статических страниц становится неограниченно много, то это повод, чтобы структурировать их и разместить не на диске, а в инфоблоках.

Размер системы довольно большой, так как в её состав включено множество компонентов необходимых для быстрого старта и работы административной части. Компоненты не консолидированы, потому что система модульная. Модули, компоненты и шаблоны имеют определенную структуру. Это важно и для обновлений системы и для разработки своих компонентов.



Большое количество файлов - свойство аналогичных систем. (У **ZendFramework** есть такая же особенность). При правильной конфигурации хостинга эту проблему возьмут на себя прекомпиляторы php. Критичным может оказаться размер выделяемого хостером места и большое число файлов системы. (Проблемой становится не штатная работа **Bitrix Framework**, а, например, работа систем бэкапов у хостеров. На большом количестве файлов они начинают себя чувствовать не очень хорошо.) Поэтому для выбора хостера рекомендуем пользоваться списком [рекомендуемых хостингов](#).

**Резюме.** В качестве инструмента хранения структуры сайта выбрана именно файловая система, а не база данных в силу того что:

- Файл дает больше свободы разработчику сайта. Поскольку файл в системе - это просто исполняемый файл.
- Так понятнее для управления. В корне такого представления - структура статических страниц HTML, разложенных по папкам. Путем некоторого совершенствования (внедряя небольшое количество PHP-кода), мы из такого сайта сразу получаем работающий на **Bitrix Framework** проект.
- В какой-то мере это - традиция, которая имела большое значение на заре становления CMS.
- Такое представление соответствует опыту контент-менеджеров, которые работают с локальными файловыми системами (папки и файлы).

Структура сайта также может быть и в БД (инфоблоки), но управлять иерархией в реляционной БД не очень-то удобно.

Рассмотрим использование файлов в **Bitrix Framework** на примерах:

- **Файловая система и меню.** Меню в файлах позволяет не подключать БД там, где это реально не нужно. То же самое относится к свойствам страниц и разделов, а также правам доступа к файлам. Теоретически можно собрать информационный сайт, где вообще не будет ни одного обращения к БД. Будет работать быстрее, особенно на разделяемом хостинге. Есть и бонусы: при копировании раздела сразу естественным образом копируются меню, права доступа, свойства раздела.
- **Файловая система и пользователи.** Пользователям из административного раздела открыт доступ к файлам ядра и другим программным файлам. Но пользователи бывают разные. Например, техподдержка 1С-Битрикса. Если веб-разработчик не уверен в своих пользователях, то он всегда может запретить им как редактирование кода PHP, так и целых разделов (ядра). По современной концепции **Bitrix Framework** в публичной части не должно быть кода PHP - все должно быть инкапсулировано в компоненты. Тогда пользователь редактирует или "голую" статику, или настраивает компонент.
- **Файловая система и языковые версии.** Было бы трудно сопровождать языковую информацию в БД. Информация в языковых файлах меняется крайне редко - проще раз в год отредактировать строчку в языковом файле, чем хранить эти



статические фразы в базе. И повторимся: база данных - это медленно и избыточно.

## Структура файлов

Файловая структура **Bitrix Framework** организована таким образом, что программные компоненты ядра продукта были отделены от пользовательских файлов, а также файлов, определяющих внешнее представление сайта. Данная особенность позволяет:

- избежать нежелательной модификации ядра продукта при работе с файлами системы;
- исключить возможность изменения публичной части сайта при загрузке обновлений продукта.
- настроить внешний вид сайта практически под любую вашу задачу

Вся система целиком лежит в каталоге **/bitrix/**, в него входят следующие подкаталоги и файлы:

- **/admin/** - административные скрипты;
- **/cache/** - файлы кэша;
- **/activities/** - папки действий для бизнес-процессов;
- **/components/** - папка для системных и пользовательских компонентов;
- **/gadgets/** - папки гаджетов;
- **/js/** - файлы javascript модулей;
- **/stack\_cache/** - файлы кеша "с вытеснением";
- **/themes/** - темы административного раздела;
- **/wizards/** - папки мастеров;
- **/images/** - изображения используемые как системой в целом, так и отдельными модулями;
- **/managed\_cache/** - управляемый кеш;
- **/modules/** - каталог с модулями системы, каждый подкаталог которого имеет свою строго определённую структуру;
- **/php\_interface/** - вспомогательный служебный каталог, в него входят следующие каталоги и файлы:
  - **dbconn.php** - параметры соединения с базой;
  - **init.php** - дополнительные параметры портала;
  - **after\_connect.php** - подключается сразу же после создания соединения с базой;

- **dbconn\_error.php** - подключается при ошибке в момент создания соединения с базой;
- **dbquery\_error.php** - подключается при ошибке в момент выполнения SQL запроса;
- **/ID сайта/init.php** - дополнительные параметры сайта; файл подключается сразу же после определения специальной константы с идентификатором сайта - **SITE\_ID**;
- **/templates/** - каталог с шаблонами сайтов и компонентов , в него входят следующие подкаталоги:
  - **/.default/** - подкаталог с общими файлами, используемыми тем или иным шаблоном по умолчанию, структура данного каталога аналогична нижеописанной структуре каталога содержащего конкретный шаблон;
  - **/ID шаблона сайта/** - подкаталог с шаблоном сайта, в него входят следующие подкаталоги и файлы:
    - **/components/** - каталог с кастомизированными шаблонами компонентов;
    - **/lang/** - языковые файлы принадлежащие как данному шаблону в целом, так и отдельным компонентам;
    - **/images/** - каталог с изображениями данного шаблона;
    - **/page\_templates/** - каталог с шаблонами страниц и их описанием хранящимся в файле **.content.php**. Когда пользователь создает новую страницу, он может выбрать, по какому шаблону из представленных в этом каталоге это будет сделано;
    - **header.php** - пролог данного шаблона;
    - **footer.php** - эпилог данного шаблона;
    - **styles.css** - CSS стили шаблона;
- **/tools/** - при инсталляции в этот каталог копируются дополнительные страницы, которые могут быть непосредственно использованы на любых страницах сайта: помощь, календарь, показ изображения и т.п.;
- **/updates/** - каталог, автоматически создаваемый системой обновлений;
- **header.php** - стандартный файл, подключающий в свою очередь конкретный пролог текущего шаблона сайта; данный файл должен использоваться на всех страницах публичной части;
- **footer.php** - стандартный файл, подключающий в свою очередь конкретный эпилог текущего шаблона сайта; данный файл должен использоваться на всех страницах публичной части;
- **license\_key.php** - файл с лицензионным ключом;
- **spread.php** - файл используемый главным модулем для переноса куков посетителя на дополнительные домены различных сайтов;

- **redirect.php** - файл используемый модулем **Статистика** для фиксации событий перехода по ссылке;
- **rk.php** - файл по умолчанию используемый модулем **Реклама** для фиксации событий клика по баннеру;
- **stop\_redirect.php** - файл используемый модулем **Статистика** для выдачи какого либо сообщения посетителю, попавшему в стоп-лист;
- **activity\_limit.php** - файл используемый модулем **Статистика** для выдачи сообщения роботу при превышении им лимита активности;

В зависимости от используемой редакции некоторые каталоги и файлы могут отсутствовать.

## Права доступа

В системе **Bitrix Framework** поддерживается два уровня разграничения прав доступа:

- Доступ на файлы и каталоги.
- Права в рамках логики модуля.

## Доступ на файлы и каталоги

Этот уровень прав проверяется в прологе, задается с помощью специального файла **.access.php**, содержащего PHP массив следующего формата:

<code>\$_PERM[файл/каталог][ID группы пользователей] = "ID права доступа";</code>
---

Где:

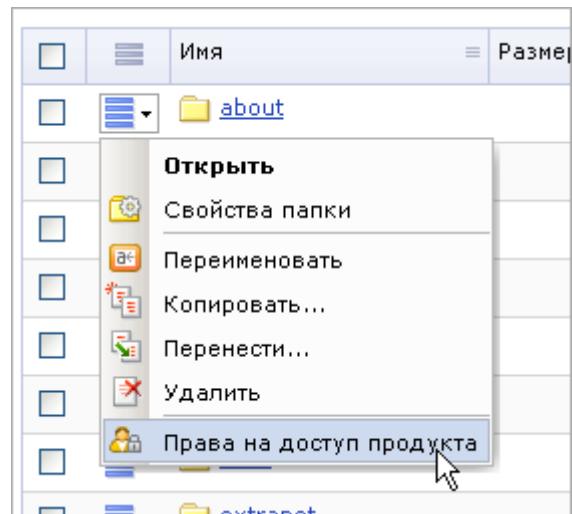
- **файл/каталог** - имя файла или каталога для которых назначаются права доступа;
- **ID группы пользователей** - ID группы пользователей на которую распространяется данное право (допустимо также использование символа - \*, что означает - для всех групп);
- **ID права доступа** - на сегодняшний день поддерживаются следующие значения (в порядке возрастания):
  - **D** - запрещён (при обращении к файлу доступ будет всегда запрещён);
  - **R** - чтение (при обращении к файлу доступ будет разрешен);
  - **U** - документооборот (файл может быть отредактирован в режиме документооборота);
  - **W** - запись (файл может быть отредактирован непосредственно);
  - **X** - полный доступ (подразумевает право на "запись" и модификацию прав доступа).



В административной части сайта права доступа на файлы и каталоги можно назначать в специально предназначеннной для этого форме:

Если пользователь принадлежит нескольким группам, то берется максимальное право из всех прав доступа заданных для этих групп.

Если для текущего файла или каталога явно не задан уровень прав, то берется уровень прав заданный для вышележащих каталогов.



## Пример 1

Файл /dir/.access.php

```
<?
$PERM["index.php"]["2"] = "R";
$PERM["index.php"]["3"] = "D";
?>
```

При попытке открытия страницы /dir/index.php пользователь, принадлежащий группе **ID=3**, будет иметь право доступа **D** (запрещено), пользователь из группы **ID=2** будет иметь право **R** (чтение). Пользователь, принадлежащий обоим группам, будет иметь максимальный уровень доступа - **R** (чтение).

## Пример 2

Файл /.access.php

```
<?
$PERM["admin"]["*"] = "D";
$PERM["admin"]["1"] = "R";
$PERM["/"]["*"] = "R";
$PERM["/"]["1"] = "W";
?>
```

Файл /admin/.access.php

```
<?
$PERM["index.php"]["3"] = "R";
?>
```

При доступе к странице `/admin/index.php` пользователь, принадлежащий группе **ID=3**, будет иметь доступ, а пользователю, принадлежащему группе **ID=2**, будет в доступе отказано. При доступе к странице `/index.php` все посетители будут иметь доступ.

### Права в рамках логики модуля

Если речь идет об обычных статичных публичных страницах, то к ним применяется только уровень 1 доступа на файлы и каталоги.

Если пользователь имеет на файл как минимум право **R** (чтение) и если данный файл является функциональной частью того или иного модуля, то проверяется 2-ой уровень прав, задаваемый в настройках соответствующего модуля. Например: При заходе на страницу **Список обращений** в техподдержке администратор видит все обращения, сотрудник техподдержки - только те за которые он ответственен, а обычный пользователь - только свои обращения. Так работает право доступа в рамках логики модуля **Техподдержка**.

Используются две методологии разграничения прав доступа 2-го уровня (уровень прав в рамках логики модуля):

- права;
- роли.

Отличие их заключается в том, что если пользователь обладает несколькими правами, то выбирается максимальное. Если же пользователь обладает несколькими ролями, то он соответственно будет обладать суммарными возможностями этих ролей.

Модули, в которых поддерживаются роли, можно увидеть в фильтре **Модуль** на странице **Настройки > Пользователи > Уровни доступа** в Административном разделе. Во всех остальных модулях и во всех остальных настройках системы используются права.

Пример:

- **Права.** Если вы принадлежите группам для которых в модуле **Статистика** заданы права **Полный административный доступ** и например, **Просмотр статистики без финансовых показателей**, то вы будете обладать максимальным правом - **Полный административный доступ**.
- **Роли.** Если вы принадлежите к группам для которых в модуле **Техподдержка** заданы роли **Клиент техподдержки** и **Демо-доступ**, то вы одновременно будете обладать возможностями этих двух ролей. Т.е. вы сможете видеть все обращения в режиме демо-доступа и одновременно с этим можете создавать свои обращения как клиент техподдержки.



## Информация на сайте и работа с ней

Тема **Информация на сайте и работа с ней** детально раскрыта в соответствующих главах курсов [Контент-менеджер](#) и [Администратор. Базовый](#). Без изучения этих тем работа разработчика в системе Bitrix Framework невозможна.

## Сайт в понятии Bitrix Framework

Сайт - это совокупность:

- **Учетной записи в базе данных**, которая создаётся в административном меню на странице **Список сайтов** ([Настройки > Настройки продукта > Сайты > Список сайтов](#)) и включает в себя следующие основные параметры:
  - **идентификатор** - набор символов, идентифицирующий сайт;
  - **доменное имя** - одно или более доменное имя сайта;
  - **папка сайта** - путь к каталогу, в котором будет храниться публичная часть сайта;
  - **язык сайта**;
  - **формат даты**;
  - **формат времени**;
  - **URL** - протокол и доменное имя по умолчанию (например, <http://www.site.ru>)
  - **DocumentRoot** - параметр заполняется при использовании многосайтности системы.
  - **условия подключения шаблонов**: каждый сайт может иметь один или более шаблонов для отображения страниц публичной части, каждый такой шаблон может быть подключен по тому или иному условию.
- **Публичной части** - совокупности скриптов (страниц) лежащих в **папке сайта** и принадлежащих этому сайту.
- **Настроек**. Каждый модуль может иметь ряд настроек связанных с сайтом, например, у модуля **Информационные блоки** эти настройки представляют из себя привязку информационного блока к тому или иному сайту, у модуля **Техподдержка** - привязку статуса, категории и т.п. к сайту.

В Bitrix Framework имеется возможность на базе одного экземпляра продукта создавать и поддерживать неограниченное количество сайтов. Особенностями системы многосайтности являются:

- единые права на управление модулями сайта;
- единый набор бюджетов пользователей на все сайты;
- единая система ведения статистики на все сайты.



Детально эта функция системы описана в главе [Многосайтовость](#)

## Структура

Структура сайта в рамках **Bitrix Framework**:

- **Шаблон** - определяет представление сайта пользователям. Существуют шаблоны компонентов и шаблоны сайта.
- **Компоненты** - задают вывод данных.
- **Страница** - элемент структуры сайта.

В этой главе будут описаны **Страница** и **Шаблон сайта**, как элементы структуры. **Компоненты** описаны в отдельной главе.

## Страница

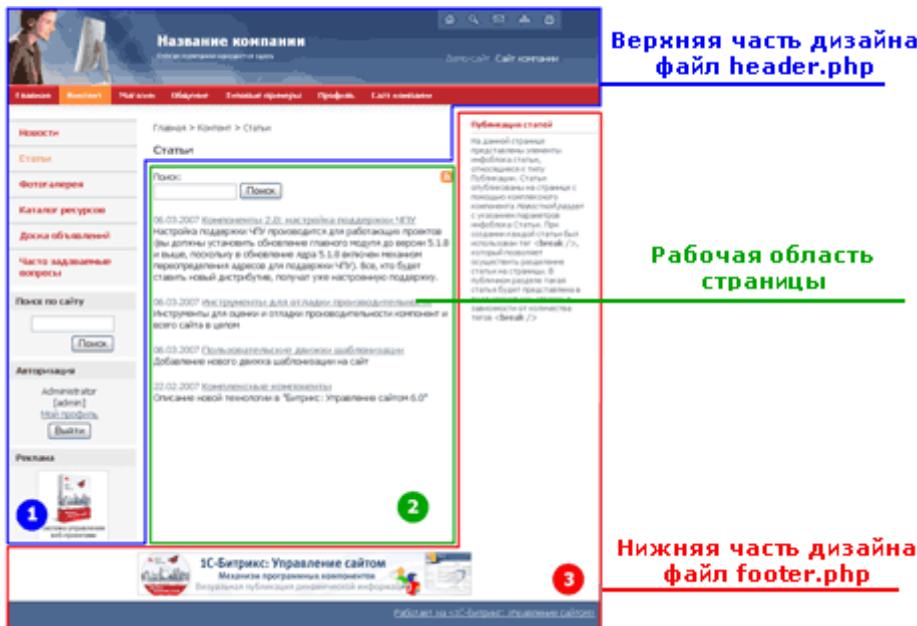
### Структура

**Страница** представляет из себя PHP файл, состоящий из пролога, тела страницы (основной рабочей области) и эпилога:

- header
- workarea
- footer

Формирование страницы сайта производится динамически, на основе используемого шаблона страницы, данных, выводимых компонентами, и статической информации, размещенной на странице. Создание шаблонов сайта и размещение на них компонентов осуществляется разработчиками сайтов.

В общем случае все страницы сайта имеют следующую структуру:



- Верхняя - **header**. Включает в себя, как правило, верхнюю и левую часть дизайна со статической информацией (логотипом, лозунгом и так далее), верхним горизонтальным меню и левым меню (если они есть в дизайне). Может включать в себя информационные динамические материалы.
  - Основная рабочая область - **work area**. Рабочая область страницы, в которой размещаются собственно информационные материалы сайта. В качестве Основной рабочей области может подключаться как физический файл, так и создаваемый системой на основе комплексных компонентов, динамический код.
- Если в качестве Основной рабочей области подключается физический файл, то такая страница называется **статической**. Если подключается динамический код, то такая страница называется **динамической**.
- Нижняя - **footer**. Включает в себя, как правило, статическую информацию (контактная информация, сведения об авторе и владельце сайта и так далее), нижнее горизонтальное меню и правое меню (если они есть в дизайне). Может включать в себя информационные материалы.

**⚠ Примечание:** Подробнее со структурой страницы можно познакомиться в [документации](#) и в уроке [Шаблон дизайна](#).

**Верхняя** и **нижняя** части дизайна формируются на основе шаблона дизайна сайта. Т.е. информация, отображаемая в данных областях, определяется параметрами шаблона сайта.

В общем случае структура страницы выглядит как:

```
<?
// подключение пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
```



```
?>
тело страницы
<?
// подключение эпилога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");
?>
```

Благодаря технологии [отложенных функций](#) часть визуальных элементов выводимых в прологе может быть задана в теле страницы, это такие элементы как:

- заголовок страницы (выводится функцией **CMain::ShowTitle**);
- навигационная цепочка (выводится функцией **CMain::ShowNavChain**);
- CSS стили (выводятся функцией **CMain::ShowCSS**);
- мета-теги (выводятся функцией **CMain::ShowMeta**);
- и др.

Принципиальной особенностью данной технологии является то, что она позволяет отложить исполнение некоторых функций, выполняя их в эпилоге, а результаты их выполнения подставляя в вышеприведенный код.

Ряд задач не могут быть решены с помощью технологии отложенных функций. Например, когда необходимо производить какие-либо действия в **Прологе** над **значениями**, которые в предыдущем примере задавались бы в теле страницы (например, свойства страницы). В этом случае возникает необходимость разбить пролог на служебную и визуальную части и эти **значения** задавать между ними.

Достигается это следующим образом:

```
<?
// подключение служебной части пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");

// здесь можно задать например, свойство страницы
// с помощью функции $APPLICATION->SetPageProperty
// и обработать затем его в визуальной части эпилога

// подключение визуальной части пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_after.php");
?>
Содержимое страницы
?>
```

```
// подключение эпилога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");
?>
```

В служебной части пролога происходит:

- подключение к базе;
- отработка агентов;
- инициализация служебных констант;
- проверка прав доступа к файлам и каталогам;
- подключение необходимых модулей
- исполнение обработчиков событий **OnPageStart**, **OnBeforeProlog**;
- ряд других необходимых действий.

Особенностью служебной части пролога является то, что она не выводит никаких данных (не посылает **header** браузеру).

В **визуальной части пролога** происходит подключение файла **/bitrix/templates//ID шаблона сайта/header.php**, где *ID шаблона сайта* - идентификатор текущего шаблона сайта. Данный файл хранит левую верхнюю часть текущего шаблона сайта.

Эпилог тоже может быть разбит на визуальную и служебную части:

```
<?
// подключение служебной части пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");

// подключение визуальной части пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_after.php");

?>
Содержимое страницы
<?

// подключение визуальной части эпилога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/epilog_before.php");

// подключение служебной части эпилога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/epilog_after.php");
```



```
?>
```

В **визуальной части эпилога** происходит подключение файла */bitrix/templates//ID шаблона сайта/footer.php*, где *ID шаблона сайта* - идентификатор текущего шаблона сайта. Данный файл хранит правую нижнюю часть текущего шаблона сайта. Помимо этого - выводится ряд невидимых IFRAME'ов, используемых технологией [переноса посетителей](#).

В служебной части эпилога происходит:

- отправка почтовых сообщений;
- исполнение обработчиков события;
- отключение от базы;
- ряд других служебных действий.

Довольно часто возникают задачи, когда нет необходимости в подключении визуальной частей пролога и эпилога. В этом случае достаточно подключить служебные части пролога и эпилога:

```
<?
// подключение служебной части пролога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");
?>
тело страницы
<?
// подключение служебной части эпилога
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/epilog_after.php");
?>
```

Для корректной работы системы, обязательным являются подключения служебных частей пролога и эпилога.

## Шаблоны

**Шаблон страницы** - это PHP файл, содержимое которого соответствует правилам формирования структуры страницы. Шаблоны могут использоваться при создании новой страницы.

Шаблоны страниц хранятся в каталогах:

- */bitrix/templates/.default/page\_templates/*;
- */bitrix/templates//ID шаблона сайта/page\_templates/*.



В каждом таком каталоге могут находиться непосредственно сами файлы шаблонов страниц, а также служебный файл **.content.php**, основная задача которого - хранить описания и порядок сортировки шаблонов страниц. Эта информация хранится в массиве **\$TEMPLATES** структура которого представлена ниже:

```
Array
(
    [имя файла] => Array
        (
            [name] => заголовок шаблона страницы
            [sort] => индекс сортировки
        )
)
```

При формировании списка шаблонов страниц используется следующий алгоритм:

- получаем ID шаблона сайта для текущего сайта который подключается без PHP условия;
  - последовательно подключаем файлы:
    - /bitrix/templates/.default/page\_templates/.section.php
    - /bitrix/templates/ID шаблона сайта/page\_templates/.section.php
- В каждом из этих файлов будет описан свой массив **\$TEMPLATES**. После подключения этих файлов, мы будем иметь объединенный массив **\$TEMPLATES**. Далее для каждого элемента этого массива, представляющего из себя описание одного шаблона страницы, выполняем следующее:
- проверяем физическое существование шаблона страницы
  - если существует, то добавляем его в список шаблонов
- сортируем полученный список шаблонов по индексу сортировки (см. выше структуру массива **\$TEMPLATES**)

## Свойства

Свойства раздела хранятся в файле **.section.php** соответствующего каталога (раздела сайта). Свойства страницы задаются, как правило, либо в теле страницы, либо между служебной частью и визуальной частью пролога.

Свойства раздела автоматически наследуются всеми подразделами и страницами данного раздела. При необходимости вы можете отредактировать свойства любой отдельно взятой страницы раздела, подправив ее параметры под конкретную ситуацию.

В работе со свойствами используются следующие функции:



## Методы установки свойств

[CMain::SetPageProperty](#) - устанавливает свойство страницы.

```
<?
$APPLICATION->SetPageProperty("keywords", "веб, разработка, программирование");
?>
```

[CMain::SetDirProperty](#) - устанавливает свойство раздела.

```
<?
$APPLICATION->SetDirProperty("keywords", "дизайн, веб, сайт");
?>
```

## Показ свойства

[CMain::ShowProperty](#) - выводит свойство страницы или свойство раздела с использованием технологии отложенных функций.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<title><?$APPLICATION->ShowProperty("page_title")?></title>
</head>
<body link="#525252" alink="#F1555A" vlink="#939393" text="#000000">
...</body>
```

## Получение значения свойства

[CMain::GetProperty](#) - возвращает свойство страницы или свойство раздела.

```
<?
$keywords = $APPLICATION->GetProperty("keywords");
if (strlen($keywords)>0) echo $keywords;
?>
```

[CMain::GetPageProperty](#) - возвращает свойство страницы.

```
<?
$keywords = $APPLICATION->GetPageProperty("keywords");
if (strlen($keywords)>0) echo $keywords;
?>
```



[CMain::GetPagePropertyList](#) - возвращает массив всех свойств страницы.

```
<?
$arProp = $APPLICATION->GetPagePropertyList();
foreach($arProp as $key=>$value)
    echo ";
?>
```

[CMain::GetDirProperty](#) - возвращает свойство раздела.

```
<?
$keywords = $APPLICATION->GetDirProperty("keywords");
if (strlen($keywords)>0) echo $keywords;
?>
```

[CMain::GetDirPropertyList](#) - возвращает массив свойств раздела, собранных рекурсивно до корня сайта.

```
<?
$arProp = $APPLICATION->GetDirPropertyList();
foreach($arProp as $key=>$value)
    echo ";
?>
```

## Работа с мета-тегами

Для работы с мета-тегами используются свойства страницы и раздела. Для работы с ними используются следующие функции:

[CMain::ShowMeta](#) - выводит свойство страницы или свойство раздела обрамляя их HTML-тегом с использованием технологии отложенных функций.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<?$APPLICATION->ShowMeta("keywords_prop", "keywords")?>
<?$APPLICATION->ShowMeta("description_prop", "description")?>
</head>
<body link="#525252" alink="#F1555A" vlink="#939393" text="#000000">
...>
```

[CMain::GetMeta](#) - возвращает свойство страницы или свойство раздела обрамляя их HTML-тегом .

```
<?
```

```
$meta_keywords = $APPLICATION->GetMeta("keywords_prop", "keywords");
if (strlen($meta_keywords)>0) echo $meta_keywords;
?>
```

## Параметры

[Параметры страницы](#) предназначены для передачи параметров в функции модулей для изменения их стандартного поведения. Например, при необходимости отключить запоминание последней страницы в сессии (при использовании постраничной навигации) или изменить стандартный формат вывода даты в функциях модуля **Информационные блоки**.

Параметры страницы доступны только в пределах страницы. Они не сохраняются ни в базе, ни в сессии.

Для работы с параметрами страницы предназначен класс [CPageOption](#).

ID	Название	Параметр	Описание	Значение по умолчанию
main	Главный модуль	nav_page_in_session	Если значение - "Y", то в сессии будет запоминаться последняя открытая страница в постраничной навигации, если значение "N" - не будет.	Y
iblock	Инфоблоки	FORMAT_ACTIVE_DATES	Если значение - FULL, то даты относящиеся к элементу информационного блока ( поля ACTIVE_FROM и ACTIVE_TO ) будут возвращаться в полном формате (вместе со временем), если значение SHORT, то - в кратком формате (без времени).	SHORT

Примеры использования:

```
<?
CPageOption::SetOptionString("main", "nav_page_in_session", "N");
?>
```

```
?>
<?
CPageOption::SetOptionString("iblock", "FORMAT_ACTIVE_DATES", "FULL");
?>
```

## Порядок выполнения

Общий [порядок выполнения страницы](#) следующий:

- Служебная часть пролога;
- Визуальная часть пролога;
- Тело страницы;
- Визуальная часть эпилога;
- Служебная часть эпилога.

Параметры компонента и шаблона доступны из программных модулей компоненты и шаблона как массив **\$arParams**. Результатом работы программного модуля компоненты является массив **\$arResult**, подаваемый на вход шаблона компонента. Результирующий HTML-код выводится обычным оператором **echo** на поток (при этом он встраивается в нужное место страницы).

Во время работы компоненты и шаблона имеется возможность использовать функционал модулей **Bitrix Framework**, которые, в свою очередь, могут обращаться к базе данных продукта.

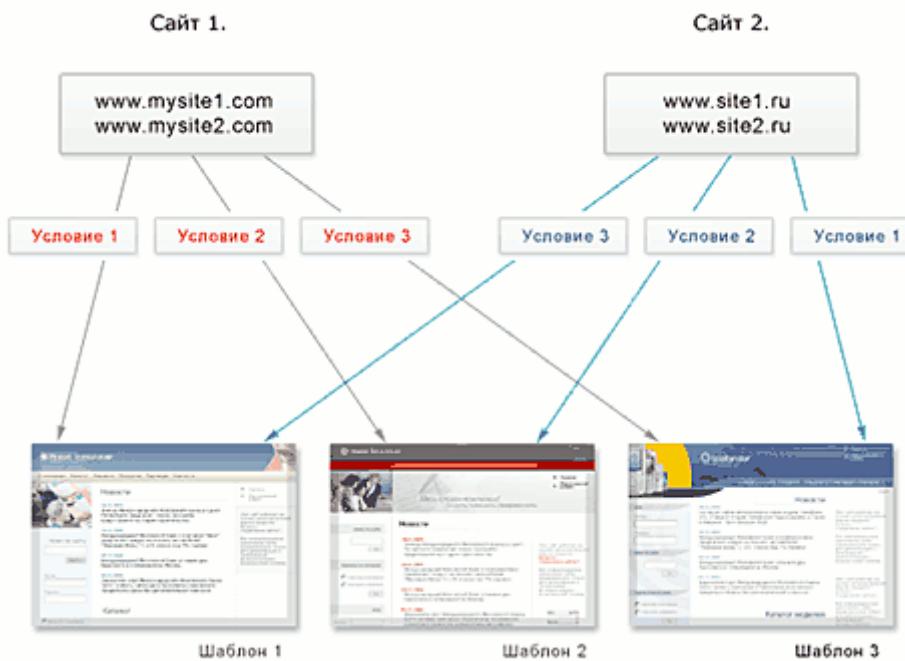
## Шаблон сайта

Отображение страниц в публичном разделе сайта выполняется на основе шаблонов дизайна сайта.

**Шаблон дизайна** - это внешний вид сайта, в котором определяется расположение различных элементов на сайте, художественный стиль и способ отображения страниц. Включает в себя программный html-код, графические элементы, таблицы стилей, дополнительные файлы для отображения контента. Может так же включать в себя шаблоны компонентов, шаблоны готовых страниц и снппеты.

В общем случае шаблон сайта задает «обрамление» страницы, а за вывод динамической информации отвечают [компоненты](#).

Количество используемых на сайте шаблонов дизайна не ограничено. Для каждого шаблона определяется условие, при котором данный шаблон будет применяться к страницам сайта:



Настройка условий применения того или иного шаблона определяется отдельно для каждого сайта (в форме создания и редактирования сайта: [Настройки > Настройки продукта > Сайты > Список сайтов](#)):

*Шаблон	Сорт.	Тип условия	Условие
Корпоративный сайт	1	[без условия]	<без условия>
Прохождение курса обучения	100	Для папки или файла	/communication/learning/cou...
Версия для печати	101	Параметр в URL	print = Y
Каталог товаров "1С"	102	Для групп пользователей	Зарегистрированные пользователи <input checked="" type="checkbox"/> Партнеры <input type="checkbox"/> Подписчики <input type="checkbox"/> Редакторы сайта <input type="checkbox"/> Администраторы техподдержки

В шаблон сайта входят:

- Набор файлов в каталоге `/bitrix/templates/ID` шаблона сайта, где **ID шаблона сайта** - поле **ID** в форме редактирования шаблона сайта. Ниже представлена структура данного каталога:
  - файл **header.php** - пролог данного шаблона;
  - файл **footer.php** - эпилог данного шаблона;
  - файл **styles.css** - CSS стили шаблона;
  - /components/** - каталог с шаблонами компонентов, принадлежащими тому или иному модулю;
  - /lang/** - языковые файлы, принадлежащие как данному шаблону в целом, так и отдельным компонентам;

- **/images/** - каталог с изображениями данного шаблона сайта;
- **/page\_templates/** - каталог с шаблонами страниц и их описанием, хранящимся в файле **.content.php**. Когда пользователь создает новую страницу, он может выбрать, по какому шаблону он это сделает.
- **/include\_areas/** - каталог с файлами - содержимым включаемых областей. Произвольная (т.е. название не регламентировано) папка для файлов включаемых областей компонента (**main.include**). Обычно вызов компонента делают из шаблона сайта. Файлы включаемых областей группируют в эту папку для удобства, т.к. они могут иметь оформление, специфичное для конкретного шаблона сайта. Там могут быть название, лого, контактная информация - чтобы пользователь редактировал только эти данные, не затрагивая шаблон сайта.
- а также ряд других вспомогательных произвольных файлов входящих в данный шаблон.

Стили шаблона сайта подключаются последними - это крайняя возможность переопределить, например, стили стандартных компонентов. Иначе потребуется масштабная кастомизация шаблонов компонентов. Если есть необходимость подключить свои стили последними, то это можно сделать с помощью функции [CMain::AddHeadString](#).

## Язык и языковые файлы

 **Язык** - это учетная запись в базе данных доступная для редактирования в административном меню на странице *Настройки > Настройки продукта > Языки интерфейса* со следующими основными полями:

- *Идентификатор*
- *Название*
- *Формат даты*
- *Формат времени*
- *Кодировка*

Как в публичной, так и в административной частях, язык используется в первую очередь для выбора того или иного языкового файла.

В административной части язык определяет [формат времени и даты](#), [кодировку страниц](#). В публичной части - данные параметры определяются настройками сайта.

## **Цитатник веб-разработчиков.**

**Антон Долганин:** В компонентах фразы выношу в ланг-файлы, просто потому что это системная часть и там хотелось бы видеть порядок.

## Языковые файлы

**Языковой файл** - PHP скрипт, хранящий переводы языковых фраз на тот или иной язык.

Данный скрипт состоит из массива \$MESS, ключи которого - идентификаторы языковых фраз, а значения - переводы на соответствующий язык.

Пример языкового файла для русского языка:

```
<?
$MESS ['SUP_SAVE'] = "Сохранить";
$MESS ['SUP_APPLY'] = "Применить";
$MESS ['SUP_RESET'] = "Сбросить";
$MESS ['SUP_EDIT'] = "Изменить";
$MESS ['SUP_DELETE'] = "Удалить";
?>
```

Пример языкового файла для английского языка:

```
<?
$MESS ['SUP_SAVE'] = "Save";
$MESS ['SUP_APPLY'] = "Apply";
$MESS ['SUP_RESET'] = "Reset";
$MESS ['SUP_EDIT'] = "Change";
$MESS ['SUP_DELETE'] = "Delete";
?>
```

Для каждого языка существует свой набор языковых файлов, хранящихся в подкаталогах /lang/ структуры файлов системы или модуля.

Языковые файлы как правило используются в административных скриптах модулей или в компонентах и в зависимости от этого подключаются одной из следующих функций:

- [IncludeModuleLangFile](#) - в административных скриптах
- [IncludeTemplateLangFile](#) - в компонентах

Для удобства поиска и дальнейшей модификации языковых фраз можно пользоваться параметром страницы show\_lang\_files=Y, позволяющим быстро [найти и исправить](#) ту или иную языковую фразу в модуле [Перевод](#).

## Примеры

Весь массив словаря можно посмотреть простой командой:



```
<? echo'  
';print_r($MESS);echo'  
'; ?>
```

Вместо порядкового номера месяца получить его название в 2 падежах:

```
<?  
echo $MESS['MONTH_'.date('n')]; // Июнь  
echo $MESS['MONTH_'.date('n').'_S']; // Июня  
  
?>
```

Так же можно поступить со днями недели, вывести название стран и так далее.

## Глава 3. Технологии

### Цитатник веб-разработчиков.

**Алексей Коваленко:** *А в данной ситуации считаю, что инструмент должен выбираться исходя из проекта, а не прихотей разработчика.*

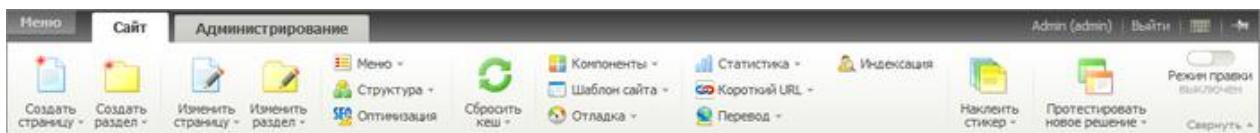
Общая информация по технологиям и принципам заложенным в систему. В курсе описаны самые часто используемые из них. С другими вы можете познакомится в документации.

#### Технологии Bitrix Framework:

- **Панель управления** - технология позволяющая облегчить администрирование сайта.
- **Агенты** - технология позволяющая запускать PHP функции с заданной периодичностью.
- **Кеширование** - позволяет кешировать ресурсоемкие куски кода, при этом заметно увеличивая производительность сайта.
- **Отложенные функции** - технология позволяющая задавать некоторые элементы страницы (заголовок, дополнительные пункты навигационной цепочки, мета-тэги, кнопки в панель управления, CSS стили) непосредственно в теле страницы.
- **Обработка событий** - технология изменения выполнения какой-нибудь API функции с помощью событий.
- **Внешняя авторизация** - технология использования собственных алгоритмов проверки и(или) внешние БД для хранения пользователей.
- **Почтовая система** - позволяет отправлять E-Mail письма с использованием заранее заданных почтовых шаблонов.
- **Перенос посетителей** - технология позволяющая синхронизировать по мере возможности набор cookie для разных сайтов, имеющих разные доменные имена и принадлежащие одному порталу.

### Административная панель

**Административная панель управления** - HTML-код, который может быть выведен авторизованному пользователю при наличии у него достаточных прав на операции, представленные на панели управления. HTML-код представляет из себя область с кнопками, в самом верху страницы, каждая из которых предназначена для той или иной операции.



Подключение панели осуществляется с помощью функции [CMain::ShowPanel](#). Данная функция использует технологию отложенных функций, что позволяет добавлять кнопки в панель непосредственно в теле страницы.

Добавить кнопку в панель можно с помощью функции [CMain::AddPanelButton](#). Эту же функцию можно использовать в скрипте `/bitrix/php_interface/include/add_top_panel.php`, который автоматически будет подключен при выводе панели.

Панель управления имеет два основных режима:

- **Сайт** - режим для работы над содержимым сайта.
- **Администрирование** - административный раздел для полнофункционального управления всем интернет-проектом.

Подробную информацию о работе с элементами управления в интерфейсе **Эрмитаж**смотрите в курсе [Контент-менеджер](#). Исторически в **Bitrix Framework** существовало еще два вида административной панели. Оба они похожи между собой (но сильно отличаются от **Эрмитажа**) и их описание так же есть в этом курсе.

## Агенты

**Агенты** - технология, позволяющая запускать произвольные PHP функции (агенты) с заданной периодичностью.

В самом начале загрузки каждой страницы (непосредственно перед событием `OnPageStart`) система автоматически проверяет, есть ли агент, который нуждается в запуске и в случае необходимости - исполняет его.

**⚠ Примечание:** Временная точность запуска агентов напрямую зависит от равномерности и плотности посещаемости сайта. Если вам необходимо организовать запуск каких либо PHP функций в абсолютно точно заданное время, то необходимо воспользоваться стандартной утилитой **cron**, предоставляемой большинством хостингов.

Кроме этого, не рекомендуется вешать на агенты ресурсоёмкие операции, для них существует фоновый запуск по **cron'y**.

Для того чтобы агент выполнился в заданное время, его необходимо зарегистрировать в системе при помощи метода [CAgent::AddAgent](#). Удалить регистрацию агента можно с помощью функции [CAgent::RemoveAgent](#).

Если функция-агент принадлежит модулю, то перед ее выполнением, этот модуль будет автоматически подключаться, а именно будет подключаться файл **/bitrix/modules/*ID модуля*/include.php**. В этом случае необходимо убедиться, что функция-агент будет доступна после подключения этого файла.

Если функция-агент не принадлежит ни одному из модулей, то ее необходимо разместить в файле **/bitrix/php\_interface/init.php**. Этот файл автоматически подключается в прологе.

Особенности создания функций-агентов:

- В функции-агенте не доступен глобальный объект `$USER` класса `CUser`. Поэтому если предполагается использование этого объекта, то в начале функции необходимо разместить следующий код:

```
global $USER;  
if (!is_object($USER)) $USER = new CUser;
```

- В качестве возвращаемого значения функция-агент должна вернуть PHP код, который будет использован при следующем запуске данной функции.

## Кеширование

Цитатник веб-разработчиков.

**Антон Долганин:** На данный момент кеширование Битрикса фактически совершенно, и не стоит изобретать своих велосипедов.

## Производительность

При большом объеме базы данных может возникнуть проблема производительности. Связано это со следующими причинами:

- обращения к этому массиву информации на чтение или на запись порождают конкурентные запросы;
- запросы сами по себе быстрые, но их такое число, что БД начинает выстраивать из них очередь;
- запросы медленные и тяжёлые, и к тому же очень частые.

Именно для разгрузки наиболее загруженных как по ресурсам, так и по времени, мест, и применяют многоуровневое кеширование. Каждую из технологий кеширования можно применять для каждого компонента в отдельности, выбирая оптимальный вариант для конкретного случая.

**⚠ Примечание:** пока разработчик не определится со стратегией кеширования и тем, что он хочет получить от её, сплошное включение кеширования может не дать ощутимых результатов.

Если в качестве примера брать интернет-магазин, то для каждого товара будет создан файл в кеше, чтобы при следующем обращении покупателя сервер не напрягался с запросами к БД. Это и позволяет запускать магазины уровня [Эльдорадо](#).

## Кеширование

**Кеширование** - технология, позволяющая кешировать результаты работы редко обновляемых и ресурсоемких кусков кода (например, активно работающих с базой данных).

Для реализации этого созданы два класса:

- [\*\*CPageCache\*\*](#) - класс для кеширования HTML результата выполнения скрипта;
- [\*\*CPHPCache\*\*](#) - класс для кеширования PHP переменных и HTML результата выполнения скрипта.

Система **Bitrix Framework** включают в себя разные технологии кеширования:

- **Кеширование компонентов** (или **Автокеширование**) - все динамические компоненты, которые используются для создания веб-страниц, имеют встроенную поддержку управления кешированием.

Для использования этой технологии достаточно [включить автокеширование](#) одной кнопкой на административной панели. При этом все компоненты, у которых был включен режим автокеширования, создадут кеши и полностью перейдут в режим работы без запросов к базе данных.

- **Неуправляемое кеширование** - возможность задать правила кеширования ресурсоемких частей страниц. Результаты кеширования сохраняются в виде файлов в каталоге **/bitrix/cache/**. Если время кеширования не истекло, то вместо ресурсоемкого кода будет подключен предварительно созданный файл кеша.

Кеширование называется неуправляемым, поскольку кеш не перестраивается автоматически после модификации исходных данных, а действует указанное время после создания, которое задается в диалоге **Параметры компонента**.

Правильное использование кеширования позволяет увеличить общую производительность сайта на порядок. Однако необходимо учитывать, что неразумное использование кеширования может привести к серьезному увеличению размера каталога **/bitrix/cache/**.

- **Управляемый кеш** - автоматически обновляет кеш компонентов при изменении данных.
- **HTML кеш** лучше всего включить на какой-нибудь редко изменяющийся раздел с регулярным посещением анонимных посетителей. Технология проста в

эксплуатации, не требует от пользователя отслеживать изменения, защищает дисковой квотой от накрутки данных и само-восстанавливает работоспособность при превышении квоты или изменении данных.

- **Кеширование меню.** Для кеширования меню применяется специальный алгоритм, который учитывает тот факт, что большая часть посетителей - это незарегистрированные пользователи.

Кеш меню управляемый и обновляется при редактировании меню или изменении прав доступа к файлам и папкам через административный интерфейс и API.

Основные настройки кеширования расположены на странице **Настройки кеширования** ([Настройки > Настройки продукта > Автокеширование](#)).

The screenshot shows the 'Cache Settings' page. At the top, there's a header with a gear icon and the title 'Настройки кеширования'. Below the header, a breadcrumb navigation shows: Рабочий стол > Настройки > Настройки продукта > Автокеширование. A tab bar at the top right includes 'Кеширование компонентов' (selected), 'Управляемый кеш' (highlighted in green), 'HTML кеш', 'Очистка файлов кеша', and a dropdown arrow. The main content area has a heading 'Настройка управляемого кеширования'. A green message states 'Управляемый кеш компонентов включен.' Below it is a button 'Выключить управляемый кеш (не рекомендуется)'. A yellow callout box contains text about 'Cache Dependencies': 'Технология управляемого кеширования Cache Dependencies автоматически обновляет кеш компонентов при изменении данных. Если управляемое кеширование включено, вам не потребуется вручную обновлять кеш компонентов при изменении новостей или товаров, изменения сразу станут видны посетителям сайта.' It also links to 'Cache Dependencies' and notes that not all components support managed caching. A bottom note says 'Замечание: не все компоненты могут поддерживать управляемое кеширование.'

## Примеры

### Пример кеширования классом **CPHPCache**

```
$cntIBLOCK_List = 10;  
$cache = new CPHPCache();  
$cache_time = 3600;  
$cache_id = 'arIBlockListID' . $cntIBLOCK_List;  
$cache_path = '/arIBlockListID/';  
if ($cache_time > 0 && $cache->InitCache($cache_time, $cache_id, $cache_path))  
{  
    $res = $cache->GetVars();
```



```
if (is_array($res["arIBlockListID"])) && (count($res["arIBlockListID"]) > 0))
    $arIBlockListID = $res["arIBlockListID"];
}

if (!is_array($arIBlockListID))
{
    $res = CIBlock::GetList(
        Array(),
        Array(
            'TYPE' => 'catalog',
            'SITE_ID' => SITE_ID,
            'ACTIVE' => 'Y',
            "CNT_ACTIVE" => "Y",
            "!CODE" => 'test%'
        ), true
    );
    while($ar_res = $res->Fetch())
    {
        if($ar_res['ELEMENT_CNT'] > 0)
            $arIBlockListID[] = $ar_res['ID'];
    }
    //////////// end cache ///////////
    if ($cache_time > 0)
    {
        $cache->StartDataCache($cache_time, $cache_id, $cache_path);
        $cache->EndDataCache(array("arIBlockListID"=>$arIBlockListID));
    }
}
```

## События

Иногда бывает необходимо повлиять на ход выполнения какой-нибудь API функции. Но если ее изменить, то эти изменения будут утеряны при очередном обновлении. Для таких случаев и разработана **система событий**. В ходе выполнения некоторых API функций, в определённых точках установлены вызовы определённых функций, так называемых **обработчиков событий**.

**⚠ Примечание:** С обработчиками следует обращаться очень внимательно. Поскольку событийная модель *BirixFramework* довольно богатая, то при небрежности в коде обработчика могут появиться трудноуловимые ошибки. Они могут основательно попортить нервы разработчику.

Какие функции-обработчики должны быть вызваны в каком месте (при каком событии) - нужно устанавливать вызовом функции, регистрирующей обработчики. В данный момент их две: [AddEventHandler](#) и [RegisterModuleDependences](#). Сам набор событий для каждого модуля описан в документации по каждому модулю. Вот, например, ссылка на [события главного модуля](#).

**RegisterModuleDependences** - функция для регистрации обработчиков, расположенных в модулях и используются для взаимодействия между модулями системы. Эту функцию необходимо вызвать один раз при инсталляции модуля, после этого функция-обработчик события будет автоматически вызываться в определённый момент, предварительно подключив сам модуль.

Удаляется с помощью [UnRegisterModuleDependences](#) при удалении модуля.

### Пример

```
// функции обработчики модуля компрессии подключаются дважды - в начале и в конце
// каждой страницы
RegisterModuleDependences("main",      "OnPageStart",      "compression",      "CCompress",
"OnPageStart", 1);
RegisterModuleDependences("main",      "OnAfterEpilog",     "compression",      "CCompress",
"OnAfterEpilog");

// инсталлятор модуля рекламы регистрирует пустой обработчик
// при возникновении события OnBeforeProlog модуль рекламы будет просто
// подключаться на каждой странице
// что сделает возможным выполнять его API функции без предварительного
// подключения в теле страницы
RegisterModuleDependences("main", "OnBeforeProlog", "advertising");
```

Каждый модуль может предоставить другим модулям интерфейс для неявного взаимодействия - набор событий. Такое взаимодействие позволяет сделать модули максимально независимыми друг от друга. Модуль ничего не знает об особенностях функционирования другого модуля, но может взаимодействовать с ним через интерфейс событий.

**AddEventHandler** - функция предназначена для регистрации произвольных обработчиков, которые не расположены в модулях. Ее необходимо вызывать до возникновения события на тех страницах, где требуется его обработать. Например, если



событие нужно обработать на всех страницах, где оно возникает, то функцию можно вызвать в [/bitrix/php\\_interface/init.php](/bitrix/php_interface/init.php).

## Пример

```
// регистрация обработчика в /bitrix/php_interface/init.php
AddEventHandler("main", "OnBeforeUserLogin", "MyOnBeforeUserLoginHandler");
function MyOnBeforeUserLoginHandler($arFields)
{
    if(strtolower($arFields["LOGIN"])=="guest")
    {
        global $APPLICATION;
        $APPLICATION->throwException("Пользователь с именем входа Guest не может быть
авторизован.");
        return false;
    }
}
```

## Различия в использовании функций

Действия, которые вы будете осуществлять с помощью событий должны быть где-то физически прописаны, и они должны быть зафиксированы, что они срабатывают на нужное событие.

**RegisterModuleDependences** производит регистрацию в БД, в **AddEventHandler** в файле **init.php**. То есть использование первой функции приводит к дополнительной нагрузке на БД. Её лучше использовать в ситуациях когда выполняемые действия должны быть зафиксированы раз и навсегда и именно в БД.

Как правило, события делятся по месту возникновения и назначению на следующие группы:

- Предназначенные для отмены дальнейшего выполнения метода. Например, событие [OnBeforeDeleteUser](#) позволяет отменить удаление пользователя при заданных условиях (наличие критических связанных объектов), событие [OnBeforeUserLogin](#) - запретить авторизацию пользователю;
- Позволяющие выполниться в определённых методах, при завершении их исполнения. Например, [OnAfterUserLogin](#) - после проверки имени входа и пароля, событие [OnUserDelete](#) - перед непосредственным удалением пользователя из БД, позволяет удалить связанные объекты;
- Возникающие во время исполнения страницы, для того чтобы включить свой код в определённые места на странице. Например, [OnBeforeProlog](#), (подробнее см. [порядок выполнения страниц](#)).

### Совет от Антона Долганина:

*Если необходимо расширить журнал событий новыми видами событий (например, при действиях с инфоблоками), то нужно использовать обработчик событий **OnEventLogGetAuditTypes**. Массив, который он вернет, приplusplusуется к стандартному массиву в админке. Именно с помощью него дописывает типы модуль Форум, например.*

Список и описание событий, доступных тому или иному модулю размещаются в [Документации для разработчика](#).

### Отложенные функции

**Отложенные функции:** - технология, позволяющая задавать заголовок страницы, пункты навигационной цепочки, CSS стили, дополнительные кнопки в панель управления, мета-теги и т.п. с помощью функций используемых непосредственно в теле страницы. Соответствующие результаты работы этих функций выводятся в прологе, то есть выше по коду, чем они были заданы.

Технология была создана в первую очередь для использования в компонентах, которые, как правило, выводятся в теле страницы, но при этом внутри них могут быть заданы заголовок страницы, добавлен пункт в навигационную цепочку, добавлена кнопка в панель управления и так далее. Отложенные функции нельзя использовать в шаблоне.

#### Алгоритм работы данной технологии:

- Любой исходящий поток из PHP скрипта буферизируется.
- Как только в коде встречается одна из следующих функций:
  - CMain::ShowTitle,
  - CMain::ShowCSS,
  - CMain::ShowNavChain,
  - CMain::ShowProperty,
  - CMain::ShowMeta,
  - CMain::ShowPanel

либо другая функция, обеспечивающая откладывание выполнения какой либо функции, то:

- весь буферизированный до этого контент запоминается в очередном элементе **стека А**;
- в **стек А** добавляется пустой элемент, который в дальнейшем будет заполнен результатом выполнения **отложенной** функции;
- имя **отложенной** функции запоминается в **стеке В**;

- буфер очищается и буферизация снова включается.

Таким образом, существует **стек А**, в котором находится весь контент страницы, разбитый на части. В этом же стэке есть пустые элементы, предназначенные для их дальнейшего заполнения результатами **отложенных** функций.

Также существует **стек В**, в котором запоминаются имена и параметры **отложенных** функции в порядке их следования в коде.

- В конце страницы в **служебной части эпилога** выполняются следующие действия:
  - все **отложенные** функции из **стека В** начинают выполняться одна за другой;
  - результаты их выполнения вставляются в специально предназначенные для этого места в **стек А**;
  - весь контент из **стека А** "склеивается" (конкатенируется) и выводится на экран.

Таким образом, технология позволяет фрагментировать весь контент страницы, разбивая его на части с помощью специальных функций, обеспечивающих временное откладывание выполнения других функций (**отложенных** функций). В конце страницы все **отложенные** функции выполняются одна за другой и результаты их выполнения вставляются в отведенные для этого места внутри фрагментированного контента страницы. Затем весь контент склеивается и отправляется браузеру посетителя сайта.

 **Внимание!** При использовании этой технологии необходимо учитывать, что над результатами работы функций обеспечивающих откладывание других функций **нельзя выполнять какие либо действия**.

Пример кода, в котором отложенная функция не будет отрабатывать код в шаблоне как ожидается:

```
if (!$APPLICATION->GetTitle())
echo "Стандартная страница";
else
echo $APPLICATION->GetTitle();
```

А такой код будет работать:

```
$APPLICATION->AddBufferContent('ShowCondTitle');

function ShowCondTitle()
{
    global $APPLICATION;
    if (!$APPLICATION->GetTitle())
        return "Стандартная страница";
```

```

else
    return $APPLICATION->GetTitle();
}

```

Ещё один пример

```

$page_title = $APPLICATION->ShowTitle(); // эта функция ничего не возвращает
if (strlen($page_title)<=0) $page_title = "Заголовок страницы по умолчанию";
echo $page_title;

```

этот код не будет работать по той причине, что все отложенные функции выполняются в самом конце страницы, в служебной части эпилога.

#### Группы функций задействованные в данной технологии:

Имя функции обеспечивающей откладывание	Выполнение какой функции откладывается	Дополнительные связанные функции
<a href="#">CMain::ShowTitle</a>	<a href="#">CMain::GetTitle</a>	<a href="#">CMain::SetTitle</a>
<a href="#">CMain::ShowCSS</a>	<a href="#">CMain::GetCSS</a>	<a href="#">CMain::SetTemplateCSS</a> <a href="#">CMain::SetAdditionalCSS</a>
<a href="#">CMain::ShowNavChain</a>	<a href="#">CMain::GetNavChain</a>	<a href="#">CMain::AddChainItem</a>
<a href="#">CMain::ShowProperty</a>	<a href="#">CMain::GetProperty</a>	<a href="#">CMain::SetPageProperty</a> <a href="#">CMain::SetDirProperty</a>
<a href="#">CMain::ShowMeta</a>	<a href="#">CMain::GetMeta</a>	<a href="#">CMain::SetPageProperty</a> <a href="#">CMain::SetDirProperty</a>
<a href="#">CMain::ShowPanel</a>	<a href="#">CMain::GetPanel</a>	<a href="#">CMain::AddPanelButton</a>

Технология позволяет создавать отложенные функции с помощью метода [CMain::AddBufferContent](#).



## Глава 4. Интеграция дизайна

В разделе подробно рассмотрены следующие вопросы:

- управление шаблоном дизайна сайта;
- работа с включаемыми и рекламными областями;
- управление средствами навигации по сайту: меню и цепочкой навигации;
- основные принципы локализации продукта;
- работа с визуальными компонентами;
- оптимизация сайта.

Минимальный уровень требований для изучения: владение базовыми технологиями для разработки веб-сайта, такими как HTML, CSS, PHP.

*Веб-дизайн – это прежде всего разработка интерфейса, среды взаимодействия пользователя с информацией, а не просто «красивая картинка». Надо учитывать особенности веб-среды, такое понятие как удобство использования (usability), направленность на цели создания сайта. Важно учесть основные сценарии поведения пользователя, особенности целевой аудитории.*

Вам при работе с дизайном собственного сайта необходимо помнить, что любая работа с дизайном сайта имеет свои предпочтения. Если вы хотите внести какие-то изменения в дизайн, то:

- сначала попробуйте сделать это редактированием шаблона самого сайта и файлов CSS;
- если предыдущее невозможно, то попробуйте сделать это средствами редактирования страницы сайта;
- и только в крайнем случае переходите к редактированию шаблонов компонента и файлов CSS компонента. При этом необходимо копировать шаблоны, а не редактировать их в системной папке

Такая последовательность действий необходима потому, что при обновлении продукта происходит и обновление шаблонов компонентов. Если вы изменили (кастомизировали) шаблон компонента, то он обновляться не будет. При этом возможны потеря функционала обновленной версии или снижение уровня безопасности.

**⚠ Внимание!** Если у вас включено кэширование компонентов, то возможна ситуация, когда после внесения изменений в шаблоны компонента или его файл стилей вы не заметите этих изменений. Дело в самом кэшировании, которое показывает вам не реальный вид, а закэшированный от предыдущих сессий. В этом случае нужно просто обновить кэш компонента, воспользовавшись кнопкой Обновить на административной панели.

**Вопрос с форума:** Хотелось бы научиться верстать (интегрировать) шаблоны под битрикс, но у меня опыта нет. Буду благодарен всем за любые подсказки, которые помогут в короткие сроки достичь цели.

*Иван, веб-разработчик:*

1. Действительно хотеть это.
2. Очень много работать в выбранном направлении.
3. Начинать с [CSS руководство](#), [HTML manual](#), и т.д. для javascript и php думаю найдете сами. Возможно стоит вначале посмотреть вот этот ресурс: [htmlbook.ru](#)
4. Никогда не верстать от "сделаю так, посмотрю на результат". Действуйте от знаний, полученных в п. 3.
5. Не паритесь если не получается и делать заново пока не получится и не разберетесь в основах основ почему именно так работает, а так не работает.
6. Вам еще предстоит узнать, что такое IE, вы стрессо-устойчивы? Это вам поможет.
7. Никогда не брать чужие куски кода/верстки и использовать "как есть", всегда добираться для сути реализации.

## Использование прав доступа

При создании шаблона сайта достаточно часто возникает задача ограничения доступа к тем или иным элементам. Предусмотренный в системе механизм проверки прав доступа может быть использован в процессе создания шаблона сайта в следующих целях:

- **Для управления показом пунктов меню**

При редактировании меню в расширенном режиме для каждого пункта может быть задано условие показа. Например:



Название: Мои курсы  
Ссылка: mycourses.php  
Сортировка: 20  
Удалить

Доп. ссылки для подсветки:

Тип условия: Выражение PHP  
Условие: \$GLOBALS['USER']->IsAuthorized()

Параметры:	Название	Значение
	=	

Вставить пункт >>

Смотри также [Настройка пунктов меню](#) в курсе **Администратор. Базовый**.

- **Для управления шаблоном меню**

Уровень прав доступа пользователей может влиять на структуру шаблона меню, используемые элементы, изображения и т.д. Пример проверки уровня прав доступа пользователя для шаблона меню приводится ниже:

```
<?if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>

<?if (!empty($arResult))?:?>
<div class="blue-tabs-menu">
    <ul>
<?foreach($arResult as $arItem):?>

    <?if ($arItem["PERMISSION"] > "D")?:?>
        </i><a
        href=<?= $arItem["LINK"]?>><nobr><?=$arItem["TEXT"]?></nobr></a></i>
    <?endif?>

    <?endforeach?>

    </ul>
</div>
<div class="menu-clear-left"></div>
<?endif?>
```

**⚠ Важно!** Условия, включающие проверку значения переменной `$PERMISSION`, используются только для меню сайта.

- **Для управления шаблоном сайта**



Для каждого шаблона дизайна может быть настроено условие его применения к сайту. Данная настройка выполняется на странице управление параметрами сайта ([Настройки системы > Сайты > Изменить](#)). Например:

*Шаблон	Сорт.	Тип условия	Условие
Корпоративный сайт	150	[без условия]	<без условия>
Версия для печати	150	Параметр в URL	print = Y
Прохождение курса обучения	150	Для папки или файла	/communication/learning/course
(нет)	151	[без условия]	<без условия>
(нет)	152	[без условия]	<без условия>
(нет)	153	[без условия]	<без условия>

В приведенном примере условие определяет, что шаблон **Версия для печати** будет применяться, если в URL параметр print=Y.

Наиболее гибким инструментом настройки условий показа является **Условие PHP**. Примеры php-условий для показа шаблона сайта:

\$USER->IsAuthorized()	Проверяется, является ли текущий пользователь авторизованным в системе.
\$USER->IsAdmin()	Проверяется, является ли текущий пользователь администратором.
in_array('5',\$USER-> GetUserGroupArray())	Проверяется, относится ли текущий пользователь к указанной группе (в данном случае к группе с ID равным 5).

Смотрите также [Настройка шаблона сайта](#) в курсе **Администратор. Базовый**.

- **Для управления элементами шаблона дизайна**

Управление показом элементов шаблона сайта, их формой, цветом и другими параметрами, может осуществляться также исходя из уровня прав доступа пользователей сайта. Детали смотри в уроке [Разработка шаблона дизайна](#).

- **Управление отдельными элементами сайта**

Использование механизма проверки прав доступа позволяет организовать управление отдельными элементами сайта (страницами, разделами, рекламой, форумами и т.д.) различными пользователями. Смотри соответствующие разделы курсов [Контент-менеджер](#) и [Администратор. Базовый](#).

## Шаблон дизайна сайта

## Цитатник веб-разработчиков.

**Ильясов Алмаз:** Обычно разработка стандартного шаблона из psd-макета занимает не более суток, это конечно с учетом времени на верстку валидного кросс-браузерного xhtml макета. На более сложные макеты может уйти до 2-х суток. Всё что идёт дольше - это уже либо интеграция, либо разработка шаблонов компонентов (обычно и то и другое вместе).

Информация данного раздела дает представление о структуре, технологии создания и использования шаблонов дизайна сайта. Также в этом разделе описывается возможность использования шаблонов для рабочей и редактируемых областей страницы сайта.

**Шаблон дизайна** - это внешний вид сайта, в котором определяется расположение различных элементов на сайте, художественный стиль и способ отображения страниц. Включает в себя программный html- и php-код, графические элементы, таблицы стилей, дополнительные файлы для отображения контента. Может так же включать в себя шаблоны компонентов, шаблоны готовых страниц и сниппеты.

Использование шаблонов позволяет проводить гибкую настройку дизайна сайтов, разделов и страниц сайта. Например, возможно использование специального праздничного дизайна в течение указанного периода времени, автоматическое управление внешним видом сайта в зависимости от группы посетителей и т.д.

У сайта может быть много шаблонов дизайна, и, наоборот, один шаблон может быть использован на нескольких сайтах.

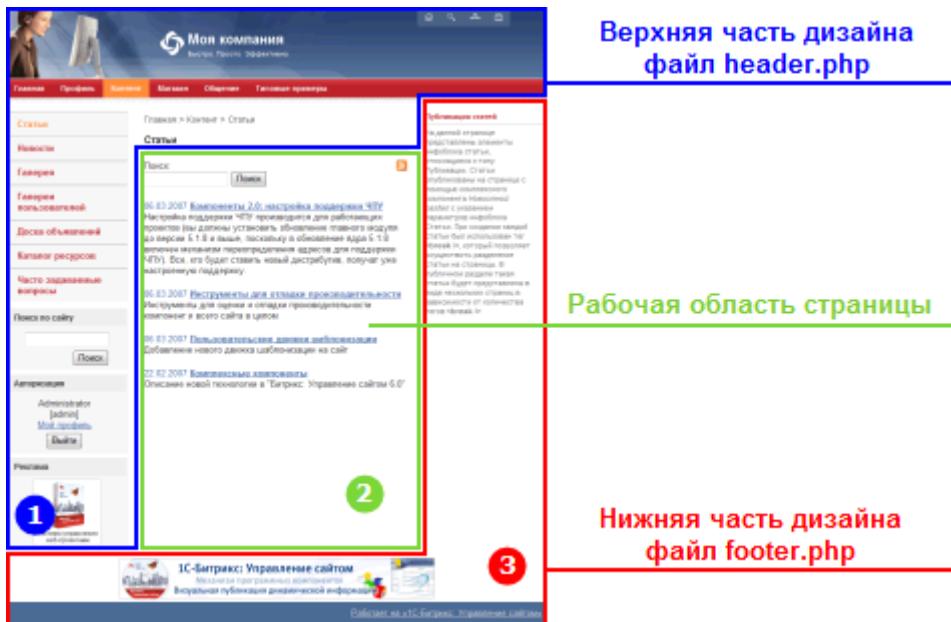
### Шаблон дизайна

Шаблон сайта определяет:

- оформление сайта (дизайн, верстку страниц, набор основных каскадных стилей);
- типы меню и их расположение;
- наличие рекламных областей (областей для размещения баннеров);
- наличие включаемых областей в шаблоне и страницах сайта;
- наличие в дизайне сайта формы авторизации, оформления подписки и т.д.

Все используемые в системе шаблоны хранятся в отдельных папках каталога /bitrix/templates/ (например, /bitrix/templates/demo/ или /bitrix/templates/template1/). Также существует специальная папка .default, которая не является полноценным шаблоном сайта, а содержит шаблоны компонентов и файлы, общие для остальных шаблонов сайта.

Шаблон дизайна сайта обычно состоит из трех основных частей:



- Верхней части дизайна (**header**);
- Рабочей области страницы (**Work area**);
- Нижней части дизайна (**footer**).

**Header** - верхняя часть дизайна, заголовок. Включает в себя, как правило, верхнюю и левую часть дизайна со статической информацией (логотипом, лозунгом и так далее), верхним горизонтальным меню и левым меню (если они есть в дизайне). Может включать в себя информационные динамические материалы. Хранится в отдельном файле ...<идентификатор\_шаблона>/header.php.

**Work area** - рабочая область страницы, в которой размещаются собственно информационные материалы сайта. Рабочая область - это все создаваемые пользователями документы, хранящиеся в файлах <имя\_документа>.php в соответствующих папках сайта.

**Footer** - нижняя часть дизайна со статической информацией (как правило: контактная информация, сведения об авторе и владельце сайта и так далее), нижним горизонтальным меню и правым меню (если они есть в дизайне). Может включать в себя информационные материалы. Хранится в отдельном файле ...<идентификатор\_шаблона>/footer.php.

## Композиция шаблона

Композиция шаблона сайта может любой. Ниже приведены несколько типовых примеров, где:

1. Header
2. Work Area



1С·БИТРИКС

Компания «1С-Битрикс» Системы управления веб-проектами

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

## 3. Footer

**Моя компания**  
Быстро. Просто. Эффективно.

Главная Профиль Контент Магазин Общение Социальная сеть Типовые примеры

Поиск по сайту  Поиск

Авторизация admin [admin] Мой профиль Выйти

Подписка на рассылку  Новости компании  Новинки каталога Подписаться

Реклама 1С-Битрикс: управление сайтом

**Каталог книг**

- Бизнес-литература (3)
- Детская литература (3)
- Компьютеры и Интернет (8)
- Наука и образование (9)
  - История (4)
  - Политология (3)
- Фантастика (3)

**Этот негодай Балмер, или Человек, который управляет "Майкрософтом"** (пер. с англ. Клингман И.)

...История «компьютерного хулигана» Стива Балмера, которому «бог „Майкрософта“» Билл Гейтс передал в январе 2000 года бразды правления компанией. Уникальная биография амбициозного гения современных высоких технологий, в которой шаг за шагом прослеживается его путь наверх!

Розничная цена: 113.10 руб Купить В корзину

**Управление сложными интернет-проектами** (пер. Головко А.)

Современные сложные Интернет-проекты требуют новых подходов к управлению процессом проектирования. Легендарный руководитель отрасли информационных технологий предоставляет практические решения ключевых проблем, связанных с разработкой Интернет-приложений...

Розничная цена: 188.50 руб

**1С-Битрикс: Управление сайтом**  
Универсальная система для управления веб-проектами  
Продукт решает задачи 95% сайтов

Работает на «1С-Битрикс: Управление сайтом»

**Моя компания**  
Быстро. Просто. Эффективно.

Главная Профиль Контент Магазин Общение Социальная сеть Типовые примеры

Поиск по сайту  Поиск

Авторизация admin [admin] Мой профиль Выйти

Подписка на рассылку  Новости компании  Новинки каталога Подписаться

Реклама 1С-Битрикс: управление сайтом

**Каталог книг**

- Бизнес-литература (3)
- Детская литература (3)
- Компьютеры и Интернет (8)
- Наука и образование (9)
  - История (4)
  - Политология (3)
- Фантастика (3)

**Этот негодай Балмер, или Человек, который управляет "Майкрософтом"** (пер. с англ. Клингман И.)

...История «компьютерного хулигана» Стива Балмера, которому «бог „Майкрософта“» Билл Гейтс передал в январе 2000 года бразды правления компанией. Уникальная биография амбициозного гения современных высоких технологий, в которой шаг за шагом прослеживается его путь наверх!

Розничная цена: 113.10 руб Купить В корзину

**Управление сложными интернет-проектами** (пер. Головко А.)

Современные сложные Интернет-проекты требуют новых подходов к управлению процессом проектирования. Легендарный руководитель отрасли информационных технологий предоставляет практические решения ключевых проблем, связанных с разработкой Интернет-приложений...

Розничная цена: 188.50 руб

**1С-Битрикс: Управление сайтом**  
Универсальная система для управления веб-проектами  
Продукт решает задачи 95% сайтов

Работает на «1С-Битрикс: Управление сайтом»



**1**

**2**

**3**

Выбор того или иного варианта композиции сайта — дело опыта и вкуса. Каждый из них имеет свои плюсы и минусы. И предложенные три схемы — неполный перечень возможной структуры. Чтобы правильно выбрать вариант композиции для своего сайта, нужно понимать принципы работы со статической информацией, компонентами, динамически выводящими информацию, и то, как они взаимодействуют между собой. Кроме того, необходимо ясно представлять квалификацию тех, кто будет заниматься поддержкой сайта, и, собственно, сам тип выводимой информации.

Статическая информация, которая не нуждается (либо редко нуждается) в замене, как правило, размещается в статических зонах **Footer** и **Header**. Заменить ее можно в кодах самих файлов, но делать это придется квалифицированному разработчику, либо разработчик должен организовать такую замену с помощью компонентов системы силами редакторов сайта.

### Подключение частей дизайна

Сборка типовых страниц сайта выполняется путем подключением верхней и нижней частей дизайна для каждой страницы сайта. В общем случае структура страницы сайта выглядит так:

```
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
```



```
$APPLICATION->SetTitle("1С-Битрикс:  
?>  
Тело документа.  
<  
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");  
?>
```

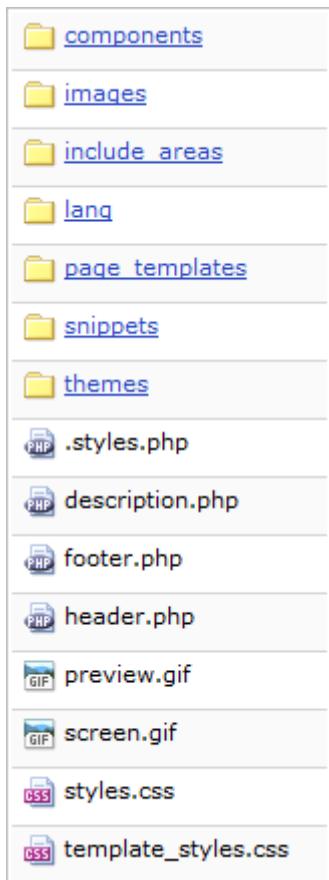
**⚠ Примечание:** Помимо статической информации, в шаблоне и в рабочей области могут располагаться:

- Визуальные компоненты
- Включаемые области
- Произвольный PHP-код

Эти элементы сайта предназначены для вывода динамической информации.

## Структура файлов шаблона сайта

Пример общей структуры файлов и папок шаблона сайта:



- каталог **components** – предназначен для шаблонов компонентов;
- каталог **images** – предназначен для картинок шаблона (которые не зависят от просматриваемой страницы), копируется из верстки сайта;
- каталог **include\_areas** – содержит включаемые области шаблона;
- каталог **lang** – содержит файлы языковых сообщений;
- каталог **page\_templates** – для шаблонов страниц и редактируемых областей;
- каталог **snippets** – содержит снippetsы – маленькие фрагменты html-кода для ускорения работы контент-менеджера по созданию часто встречающихся блоков кода;
- каталог **themes** – тема оформления шаблона;
- файл **header.php** – часть шаблона ДО контента;
- файл **footer.php** – часть шаблона ПОСЛЕ контента;
- файл **description.php** – название и описание шаблона;
- файл **.styles.php** – описания стилей для визуального

редактора страниц;

- файл **template\_styles.css** – стили шаблона (стили применяемые в самом шаблоне дизайна сайта);
- файл **styles.css** – стили для контента и включаемых областей. Эти стили можно применять в визуальном редакторе.

## Разработка шаблона дизайна

### **Цитатник веб-разработчиков.**

*Роман Петров: Периодически возникает вопрос: продайте мне Битрикс с шаблоном... Ведь для Joomla шаблоны есть, а для Битрикс? Открываю небольшую тайну: для Битрикс все немного по другому.*

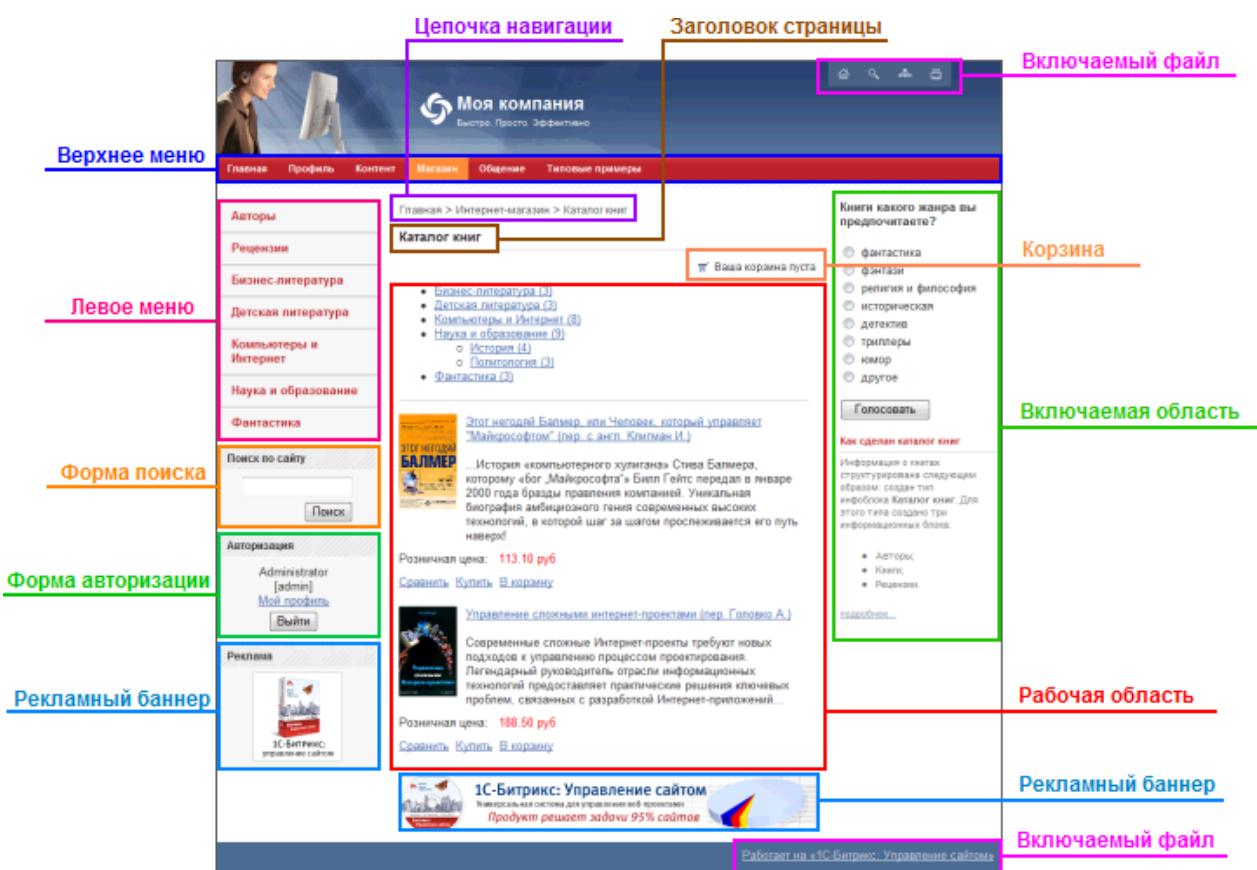
*Справедливости ради следует отметить, что разработка простого шаблона для 1С-Битрикс на основе имеющейся верстки - достаточно несложное занятие. Сложность заключается в том, что мало кому требуется сделать простой шаблон. Везде нужно реализовать бизнес-логику заказчика. Ведь Битрикс - решение для бизнеса, а не для "поиграться".*

Процесс создания шаблона сайта включает два основных этапа:

- создание прототипа шаблона дизайна;
- создание полнофункционального шаблона.

## Разработка прототипа шаблона дизайна

Прототип представляет собой сверстанный в html шаблон дизайна сайта. При верстке в шаблоне выделяются функциональные области, например:



- заголовок страницы;
- меню;
- цепочка навигации;
- форма авторизации;
- форма поиска;
- включаемые области и файлы;
- рекламные области;
- и т.д.

### Создание полнофункционального шаблона дизайна

На данном этапе выполняется замена HTML элементов дизайна на соответствующие функциональные элементы: программный код и вызовы компонентов. В результате чего уже получается PHP-шаблон дизайна сайта.



**⚠ Примечание:** При создании шаблона сайта возможно использование различных программных условий, влияющих на отображение тех или иных элементов шаблона для различных разделов сайта. Для этого для раздела сайта нужно определить некоторое свойство, значение которого будет проверяться в шаблоне сайта:

```
<if ($APPLICATION->GetProperty("SECT_PROP")=="Y"):>
```

Подобные условия можно вводить для любых элементов шаблона. Например, можно отключать показ включаемых областей или управлять показом навигационной цепочки, и т.п.

### Рекомендации по созданию шаблона

Основные моменты, которые нужно учитывать при создании шаблона:

- При подготовке графического дизайна следует заранее разметить линию раздела дизайна на пролог (**header.php**) и эпилог (**footer.php**).
- Следует выделить основные элементы дизайна, для последующей модификации таблицы стилей: шрифты, цвета заливки и т.п.
- Разрабатывая дизайн меню различных уровней, желательно выделять повторяющиеся элементы - для упрощения создания шаблона меню и дальнейшего управления этими меню.



- Для облегчения сопровождения различных языковых версий сайта по возможности следует использовать вместо графических элементов текстовые.
- При нарезке графического дизайна и подготовке HTML шаблона, необходимо заранее предусмотреть место расположения основных компонентов системы управления сайтом. Выделить области меню, рекламные области, области размещения дополнительных форм.
- Рекомендуется производить подготовку шаблона с учетом последующей табличной сборки. Одновременно допускается использование слоев.
- При нарезке графического дизайна выделяются однотонные области. При сборке шаблона эти области могут быть представлены ячейками таблиц со сплошной заливкой цвета.
- Размещать графические изображения, относящиеся к шаблону, следует в папке `/bitrix/templates/<имя_шаблона>/images`.
- Редактирование шаблона дизайна сайта в визуальном режиме будет происходить корректно, если в атрибутах HTML-тегов не содержится php-код, а также, если, например строки и ячейки таблицы не прерываются php-кодом при формировании таблицы. Если в коде шаблона дизайна сайта есть такие особенности, то редактировать его следует только в режиме кода.
- Каскадные стили, используемые в шаблоне, рекомендуется разделять на две таблицы стилей, хранящиеся в двух разных файлах. Оба файла находятся в директории `/bitrix/templates/<идентификатор_шаблона>/`. Файл **styles.css** содержит стили для представления информационного содержания страницы на сайте. Файл **template\_styles.css** содержит стили для отображения текстов в самом шаблоне дизайна.

Создание шаблона сайта рекомендуется выполнять на локальной демо-версии продукта. Готовый шаблон необходимо экспортировать средствами системы в виде комплекта файлов формата **tar.gz** на удаленный сервер и развернуть его.

**⚠ Примечание:** Крайне не рекомендуется использование комплексных компонентов в шаблоне дизайна. Та как в этом случае правила переписывания адресов начинают работать для всего сайта. Это может отражаться на работе ЧПУ других компонентов и страницы `/404.php`. Комплексные компоненты должны находиться в **Work#Area**.

**⚠ Внимание!** При использовании визуального редактора для редактирования шаблона сайта, необходимо быть очень внимательным: возможны непредвиденные искажения кода.

## Опыт веб-разработчиков.

**Роман Петров:** Мы, например, интегрируем готовую верстку в Битрикс и поэтому в большинстве случаев у нас стили задаются для всего сайта сразу, стили компонентов мы чаще всего удаляем, а сами шаблоны компонентов складываем в **.default** потому что это сильно экономит время при наличии нескольких шаблонов сайта.

Мы складываем все шаблоны в **.default** потому, что если на сайте несколько шаблонов дизайна, очень тяжело вносить правки в каждый из них. Гораздо проще поменять в одном месте.

**Sergey Leshchenko:** Значит, я храню в:

**.default** - все шаблоны компонентов, все стили всех шаблонов сайтов, картинки/спрайты, js и прочее с преследованием цели минимизации обращений к веб-серверу, но без фанатизма.

*site\_template\_name/ - styles.css со стилями для визуального редактора, шаблоны общих компонентов специфичные только под данным шаблоном сайта.*

**Евгений Малков:** Все стили храню в папке шаблона сайта в одном файле **styles.css**. Картинки, используемые в css, в **/images/** в корне сайта. Когда точно знаю, что будет 1 сайт то храню шаблоны компонент в **.default**, когда несколько сайтов - в шаблоне сайта. Иногда, при другом шаблоне для ряда страниц (например для главной), в **styles.css** этого шаблона подключаю стиль основного шаблона. От верстальщика приходит 1 файл стилей на все шаблоны и делить его, особенно для компонент, сложно.

## Управление шаблоном дизайна

Все шаблоны могут быть загружены в формате **<имя\_файла>.tar.gz**. Также шаблон дизайна может быть добавлен путем копирования папки шаблона в систему.

**⚠ Примечание:** В рамках курса [Администратор. Базовый](#) также описано [редактирование шаблона сайта](#).

Управление шаблонами дизайна осуществляется в административном разделе на странице **Шаблоны сайта** ([Настройки > Настройки продукта > Сайты > Шаблоны сайтов](#)):

Шаблоны сайта

Рабочий стол > Настройки > Настройки продукта > Сайты > Шаблоны сайтов

Добавить шаблон | Загрузить шаблон | Настроить | Excel

ID	Название	Описание
books	Корпоративный сайт 	На главную страницу сайта выведен каталог книг.
learning	Прохождение курса обучения	Шаблон модуля обучения
print	Версия для печати	Шаблон для печати

В указанном разделе вы можете выполнить следующие операции:

- Отредактировать существующий шаблон;
- Создать новый;
- Импортировать шаблон;
- Экспортировать шаблон.

Для наглядного представления шаблона в списке может использоваться его скриншот. Скриншот размещается в папке соответствующего шаблона в файле с именем **screen.gif** (например, /bitrix/templates/books/screen.gif).



The screenshot shows the Bitrix Admin Panel interface. On the left, there's a sidebar with a gear icon and the title 'Шаблоны сайта'. Below it is a navigation tree with categories: 'Рабочий стол > Настройки', 'Добавить шаблон', 'books', 'learning', and 'print'. A status bar at the bottom left says 'Выбрано: 3 Отмечено: 0'. On the right, a browser window shows a website with a header 'Моя компания' and a sub-header 'Быстро. Просто. Эффективно'. The website has a navigation menu with items 'Главная', 'Контент', 'Магазин', 'Общение', 'Типовые примеры', and 'Профиль'. Below the menu is a search bar labeled 'Поиск по сайту' with a 'Поиск' button. To the right of the search is a sidebar titled 'Каталог книг' containing a list of book categories:

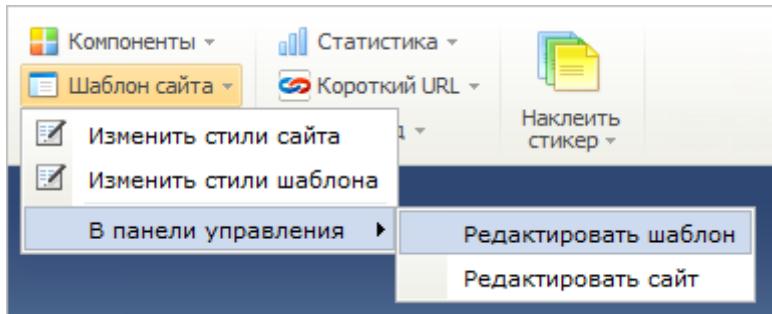
- ◆ Беллетристика. Популярная литература (4)
- ◆ Бизнес-литература (3)
- ◆ Детская литература (3)
- ◆ Компьютеры и Интернет (8)
- ◆ Наука и образование (9)
  - ◊ История (4)
  - ◊ Политология (3)
- ◆ Фантастика (3)

At the bottom right of the sidebar, there's a note: 'Этот негодяй Балмер, или Человек, который уничтожил "Майкрософт" (пер. с анг.)'.

## Редактирование шаблона

Чтобы просмотреть или изменить структуру и программный код шаблона, перейдите в режим редактирования, выбрав в меню действий пункт **Изменить**.

**! Примечание:** Перейти к редактированию шаблона можно и из публичной части сайта. Для этого используйте пункт **Редактировать шаблон** меню кнопки **Шаблон сайта** на административной панели:



Есть возможность редактировать шаблон дизайна, как с использованием визуального редактора, так и работая с исходным кодом. Возможность визуального редактирования шаблона определяется настройками **Главного модуля**.



Настройки   Авторизация   Журнал событий   Система обновлений   Доступ

### Настройка параметров модуля

**Системные настройки**

Язык по умолчанию для административной части: [ru] Russian

Название сайта: Интернет-магазин

URL сайта (без http://): Например: www.mysite.com localhost

Имя префикса для названия cookies (без точек и пробелов): BITRIX\_SM

Распространять куки на все домены:

Посыпать в заголовке статус 200 на 404 ошибку:

Режим вывода ошибок (error\_reporting): Только ошибки

Использовать визуальный редактор для редактирования шаблонов сайта:

При отмеченной опции в форме редактирования шаблона будет присутствовать опция **Использовать визуальный редактор**.

Шаблон   Стили сайта   Стили шаблона   Дополнительные файлы

### Настройки шаблона сайта

ID: store\_minimal\_blue (/bitrix/templates/store\_minimal\_blue/)

Название: Фиксированный шаблон

Описание: Легкий и светлый шаблон с фиксацией по ширине

\*Внешний вид шаблона сайта (рабочую область заменить #WORK AREA#):

Использовать визуальный редактор

Инструментальная панель (Формат, Шрифт, Размер, В, И, У, Краска, Мобильная версия)

Рабочая область (редактор WYSIWYG с компонентами Content, Services, Communication, Shop, Shop: Shop)

Компоненты: Контент, Сервисы, Общение, Магазин, Решение: Магазин



**⚠ Внимание!** Редактирование шаблона дизайна сайта в визуальном режиме будет происходить корректно, если в атрибутах HTML-тегов не содержится php-код, а также, если, например строки и ячейки таблицы не прерываются php-кодом при формировании таблицы. Если в коде шаблона дизайна сайта есть такие особенности, то редактировать его следует только в режиме кода. Также не рекомендуется редактировать шаблон при наличии сложной верстки.

При редактировании шаблона в визуальном редакторе отображаются **объединенные верхняя и нижняя части дизайна** сайта. В HTML-код вставляются компоненты и функции на языке программирования PHP, которые обеспечивают показ различного рода информации: метаданных, заголовка страницы, таблицы каскадных стилей, административной панели и последующее управление этой информацией с использованием визуальных инструментов.

**⚠ Внимание!** В шаблоне дизайна сайта крайне не рекомендуется использовать комплексные компоненты.

**⚠ Обратите внимание** на наличие в коде шаблона разделителя `#WORK_AREA#`, который используется для указания границы между верхней и нижней частью дизайна. В этом месте будет выполняться подключение рабочей области страницы сайта. В визуальном режиме редактирования шаблона рабочая область обозначается **WORK AREA**. Сохранение шаблона без этого разделителя **невозможно**.

## Экспорт шаблона

С помощью интерфейса системы используемый на сайте шаблон может быть выгружен в файл формата `<имя_шаблона>.tar.gz`. Для выгрузки шаблона служит пункт контекстного меню **Скачать**.

ID	Название	Описание
books	Корпоративный сайт	На главную страницу сайта выведен каталог книг.
learning	Прохождение курса обучения	Шаблон модуля обучения

Контекстное меню, открытое на строке с ID "books":

- Изменить
- Копировать
- Скачать** (выделено курсором)
- Удалить



## Импорт шаблонов

Готовый шаблон можно экспорттировать в виде комплекта файлов с помощью менеджера файлов либо с помощью специального интерфейса системы. На странице со списком шаблонов имеется специальная кнопка контекстной панели **Загрузить шаблон**.

При нажатии на кнопку открывается форма:

- С помощью кнопки **Обзор...** укажите файл с шаблоном для загрузки.

**⚠ Примечание:** Файлы шаблона должны быть в кодировке *UTF-8*.

При загрузке по умолчанию шаблон будет распакован и помещен в папку с именем, соответствующим имени загружаемого файла (`/bitrix/templates/<идентификатор_шаблона>/`). Например, если имя загружаемого файла `template1.tar.gz`, то шаблон будет автоматически помещен в папку `.../template1/`, а самому шаблону будет присвоен идентификатор (ID) **template1**.

- Для того чтобы шаблону был присвоен другой идентификатор, а сам шаблон был размещён в папке с соответствующим именем, в поле **Код шаблона** указывается нужный код шаблона.
- Также можно привязать загруженный шаблон как шаблон по умолчанию для выбранного сайта с помощью соответствующей опции.



## Создание шаблона

Шаблон дизайна сайта может быть создан непосредственно в системе с помощью формы **Новый шаблон**, для перехода к которой служит кнопка **Добавить шаблон**, расположенная на контекстной панели.

При создании нового шаблона через интерфейс задается:

- его идентификатор;
- название;
- описание для показа в списке;
- код шаблона внешнего вида сайта;
- таблицы стилей:
  - Закладка **Стили сайта** служит для описания таблиц каскадных стилей (CSS), используемых на страницах сайта. Описание стилей хранится в файле **styles.css** в папке шаблона сайта.
  - Закладка **Стили шаблона** служит для описания таблиц каскадных стилей (CSS), используемых в шаблоне. Описание стилей хранится в файле **template\_styles.css** в папке шаблона сайта.
- набор используемых включаемых компонентов и картинок.

При сохранении шаблона автоматически создается поддиректория `/bitrix/templates/<идентификатор_шаблона>`.

Все графические элементы, используемые в шаблоне, рекомендуется размещать в директории `/bitrix/templates/<идентификатор_шаблона>/images/`.

**⚠ Примечание:** На время создания шаблона рекомендуется отключить кеширование.

## Разработка шаблонов страниц

Система **Bitrix Framework** позволяет создавать и использовать шаблоны-заготовки для рабочей и включаемых областей страницы сайта. Использование шаблонов-заготовок особенно полезно и эффективно в тех случаях, когда страницы сайта имеют довольно сложную структуру (верстку).

The screenshot shows a web-based content management system interface. At the top, there's a header with a logo of a person wearing a headset, the text "Моя компания" (My Company), and a tagline "Быстро. Просто. Эффективно". Below the header is a navigation menu with links: Главная, Профиль, Контент, Магазин, Общение, Типовые примеры. The "Контент" link is highlighted in orange. On the left, there's a sidebar with links: Статьи, Новости, Галерея, Галереи пользователей, Доска объявлений, Каталог ресурсов, Часто задаваемые вопросы, and Поиск по сайту. The main content area shows a breadcrumb trail: Главная > Контент > Статьи. Below it, there's a search bar with a placeholder "Поиск:" and a "Поиск" button. A red box highlights a section titled "Публикация статей" which contains text about how articles are published using a complex component and how they are divided into multiple pages using the <b:break> tag. Below this, there are two news items: one from 06.03.2007 about components for CHPU support, and another from 06.03.2007 about tools for waste recycling.

Шаблоны страниц и редактируемых областей хранятся в папке `/page_templates/`, находящейся в каталоге соответствующего шаблона сайта или в папке `.default`, если предполагается использование одинаковых шаблонов страниц для всех шаблонов сайта.

При создании страницы в режиме визуального HTML-редактора достаточно выбрать из списка шаблонов, по которому создается страница, и добавить нужную информацию.

This screenshot shows the visual editor interface for creating a new page. At the top, there's a toolbar with buttons for "Редактирование" (Editing) and "Меню" (Menu). Below the toolbar, the title "Редактирование страницы" (Edit page) is displayed. A dropdown menu labeled "Шаблон:" (Template:) is open, showing three options: "Стандартная страница" (Standard page), "Включаемая область для страницы" (Page include area), and "Включаемая область для раздела" (Section include area). The second option is currently selected. The main workspace shows a header section with "Заголовок" (Header) and "Описание" (Description), and a sidebar on the right titled "Компоненты" (Components) listing "Контент" (Content), "Сервисы" (Services), "Общение" (Communication), "Магазин" (Shop), "Служебные" (Utility), and "Мои компоненты" (My components). The bottom of the screen features a standard Windows-style ribbon with tabs like "Normal", "Шрифт" (Font), "Размер" (Size), and "Корпоративный сайт" (Corporate site).

Список доступных шаблонов страниц и редактируемых областей создается с помощью файла **.content.php**, также размещаемого в папке `/page_templates/` соответствующего шаблона сайта. Данный файл содержит массив соответствий файлов шаблонов страниц и включаемых областей и их названий для представления в списке.

```
<?
if(!defined("B_PROLOG_INCLUDED"))           // B_PROLOG_INCLUDED!==true)die();
$TEMPLATE["standard.php"] = Array("name"=>"Стандартная страница", "sort"=>1);
$TEMPLATE["page_inc.php"] = Array("name"=>"Включаемая область страницы",
"sort"=>2);
$TEMPLATE["sect_inc.php"] = Array("name"=>"Включаемая область раздела", "sort"=>3);
?>
```

Дополнительно задаются значения сортировки названий шаблонов страниц в выпадающем списке.

## Использование файлов языковых сообщений

Система позволяет использовать один и тот же шаблон дизайна для нескольких сайтов на разных языках. Данная возможность реализуется с помощью механизма языковых сообщений:

- в HTML верстке шаблона определяются области, в которых должен отображаться тот или иной текст;
- затем в выделенные области вместо текстовых сообщений подставляется код, выполняющий подключение и показ соответствующих языковых сообщений на нужном языке.

Описание механизма использования языковых сообщений приводится в разделе [Файлы языковых сообщений](#).

## Примеры работы и решения проблем

Цитатник веб-разработчиков.

**Алексей Майдокин:** Моё мнение - вёрстке и другим сложным штукам место в шаблонах, компонентах и тому подобном, короче там, куда простым смертным просто так не добраться. Сложная вёрстка в области контента - это как двигатель в салоне легковушки.

Подробную информацию об условиях применения шаблонов можно посмотреть в следующих уроках:

- [Применение шаблона дизайна](#) (курс Контент-менеджер)
- [Шаблоны дизайна](#) (курс Администратор. Базовый)

### Простые примеры применения шаблонов в зависимости от различных условий

Если свойство раздела phone равно Y

```
$APPLICATION->GetDirProperty("phone")=="Y"
```

Если текущий раздел равен /ru/catalog/phone/

```
$APPLICATION->GetCurDir() == "/ru/catalog/phone/"
```

Если текущий пользователь - администратор

```
$USER->IsAdmin()
```

### Подключение favicon.ico для разных шаблонов

Для того чтобы в разных шаблонах сайтов были разные значки, надо в шаблоне в **header.php** добавить вызов значка из шаблона:

```
<link rel="icon" type="image/x-icon"  
href="<?=SITE_TEMPLATE_PATH;?>/images/favicon.ico" />
```

### Отдельный шаблон для главной страницы сайта

**Цитата:** По умолчанию ставлю шаблон, который рассчитан для внутренних страниц сайта. На главной странице будет свой шаблон. Подскажите, как составить условие для отображения нужного шаблона на главной странице сайта?

В добавок к основному шаблону для всех страниц сайта необходимо подключить требуемый шаблон через условие **Для папки или файла**, в котором указать /index.php с учетом [индекса сортировки](#) применения шаблона.



## **Изменение шаблона сайта, в зависимости от временного параметра**

**Цитата:** Как реализовать изменение шаблона сайта, в зависимости от временного параметра (Всего два шаблона, но они должны меняться каждый час)?

Для этого необходимо сделать изменение текущего шаблона условию **Выражение PHP**.

### **Первый вариант**

По нечетным часам:

```
((date("H") +6) %2) == 1
```

По четным часам:

```
((date("H") +6) %2) == 0
```

где +6 - указывает временную зону.

### **Второй вариант**

```
date("H") %2
```

или

```
!date("H") %2
```

## **Применение шаблона сразу по двум условиям**

**Цитата:** Подскажите, как задать шаблон одновременно по 2 условиям (для группы пользователей и для папки и файла) в настройках БУС.

Для этого необходимо использовать изменение текущего шаблона по условию **Выражение PHP**:

```
in_array($groupID, $USER-> GetUserGroupArray()) || strpos($APPLICATION->GetCurDir(), "/dir/") != false || $APPLICATION->GetCurPage() == "/dir/file.php"
```

## **Применение шаблона только к файлам с расширением \*.php**

**Цитата:** Подскажите, пожалуйста, выражение PHP, чтобы шаблон действовал на все страницы, заканчивающиеся на .php, а .html не трогал.

Решение: изменение шаблона условию **Выражение PHP**:

```
substr($APPLICATION->GetCurPage(true), -4) == ".php"
```

## Изменение дизайна «шапки» сайта для разных разделов

**Задача:** Сайт разделен на несколько разделов. По замыслу у каждого раздела должна быть своя «шапка» в дизайне. Более в дизайне ничего не меняется. Как лучше реализовать смену «шапок» разделов?

**Решение:** В шаблон подключается компонент Включаемая область (для раздела):

```
<div id="header">
<?$_APPLICATION->IncludeComponent("bitrix:main.include", ".default", array(
    "AREA_FILE_SHOW" => "sect",
    "AREA_FILE_SUFFIX" => "headerinc",
    "AREA_FILE_RECURSIVE" => "Y",
    "EDIT_TEMPLATE" => "sect_headerinc.php"
),
false
);?>
</div>
```

Код шапки каждого из разделов будет храниться в файле **sect\_headerinc.php**. Параметр

```
"AREA_FILE_RECURSIVE" => "Y"
```

означает, что такая же "шапка" появится у всех подразделов данного раздела, если родительский **sect\_headerinc.php** не будет специально перекрыт у кого-то из детей.

## Файлы языковых сообщений

Система **Bitrix Framework** позволяет использовать один и тот же шаблон дизайна для нескольких сайтов на разных языках. Данная возможность реализуется с помощью механизма языковых сообщений:

- в HTML верстке шаблона определяются области, в которых должен отображаться тот или иной текст;
- затем в выделенные области вместо текстовых сообщений подставляется код, выполняющий подключение и показ соответствующих языковых сообщений на нужном языке.

Например, с использованием языковых сообщений могут быть созданы заголовки таблиц, надписи на кнопках, сообщения компонентов и т.д.



**⚠ Примечание:** Механизм языковых сообщений используется, в том числе, для поддержки многоязычного интерфейса административного раздела системы.

		ID	Дата изм.	Акт.	Сорт.	Заголовок	Документов
		1	29.03.2007 17:21:57	Да	300	Published	0
		2	29.03.2007 17:21:57	Да	100	Draft	0
		3	29.03.2007 17:21:57	Да	200	Ready	1
Выбрано: 3		Отмечено: 0					

		ID	Modified	Active	Sort.	Title	Documents
		1	03/29/2007 17:21:57	Yes	300	Published	0
		2	03/29/2007 17:21:57	Yes	100	Draft	0
		3	03/29/2007 17:21:57	Yes	200	Ready	1
Selected: 3		Checked: 0					

## Механизм реализации

**Языковое сообщение** - группа фраз на разных языках, имеющая один смысл.

- В папке шаблона сайта создается папка с именем **/lang/**:  
/bitrix/templates/<идентификатор шаблона>/lang/
- В папке **/lang/** создаются папки с идентификаторами используемых языков: **/en/, /de/, /ru/** и т.д. Например:  
/bitrix/templates/<идентификатор шаблона>/lang/ru/
- В созданных папках размещаются соответствующие файлы языковых сообщений. Файлы языковых сообщений характеризуются следующими параметрами:
  - Имя файла языковых сообщений соответствует имени файла, в котором выполняется его вызов. Например, если предполагается, что вызов файла с языковыми сообщениями будет выполняться в прологе шаблона сайта (файл **header.php**), то файл языковых сообщений должен быть сохранен с именем **header.php**;
  - Список сообщений в файле хранится в следующем виде:

```
<?
$MESS ['COMPANY_NAME'] = "Company Name";
$MESS ['MAIN_PAGE'] = "Home page";
$MESS ['PRINT'] = "Print version";
$MESS ['AUTH_LOGIN'] = "Authorization";
```

```
$MESS ['RATES_HEADER'] = "Currency rates";
$MESS ['SEARCH'] = "Site search";
$MESS ['SUBSCR'] = "Subscription";
?>
```

- Затем в начало файла, для которого предусмотрено использование языковых сообщений (например, **header.php**), добавляется функция:

```
<?
IncludeTemplateLangFile(__FILE__);
?>
```

Функция `IncludeTemplateLangFile(__FILE__)` выполняет подключение файла языковых сообщений для текущего языка.

- Далее все текстовые сообщения заменяются на функции вызова соответствующих языковых сообщений:

```
<echo GetMessage("SEARCH");?>
```

В качестве параметра функции `GetMessage()` используется код подключаемого сообщения. Функция проверяет наличие в подключенном языковом файле сообщения с соответствующим кодом и отображает его пользователю.

## Загрузка и выгрузка локализации

### Загрузка и выгрузка в CSV-файл

Для ручной правки файлов локализации используется функционал выгрузки и загрузки CSV-файла.

Для этого необходимо:

- Перейти в необходимую папку (например **activity**), в закладке **Выгрузка файла** ([Настройки > Локализация > Просмотр файлов](#)) выбрать **Выгрузить все переводы** для экспорта всех строк локализации данной папки (модуля), или **Выгрузить только непереведенные** для экспорта только непереведенных строк в CSV-файл.
- Нажать кнопку **Экспортировать**, появится диалог загрузки созданного CSV-файла локализации:

Загрузка файла Выгрузка файла

Экспорт в CSV

Обработка файлов:  Выгрузить все переводы  Выгрузить только непереведенные

Экспортировать

Название CSV-файла по умолчанию будет состоять из названий папок (модуля), входящих в локализуемую часть. В нашем случае - это **bitrix\_activities.csv**

- Отредактировать необходимые строки и сохранить все изменения в том же CSV-файле.
- Далее на закладке **Загрузка файла** выбрать способ добавления локализации в систему: **только новые переводы** или **заменять языковые файлы**, выбрать CSV-файл и нажать кнопку **Импортировать**:

Загрузка файла Выгрузка файла

Импорт CSV файла

Файл CSV: sons\bitrix\_activities.csv" Обзор...

Обработка файлов:  добавлять только новые переводы  полностью заменять языковые файлы

Импортировать

- Локализация будет добавлена в систему.

## Сбор переводов

После создания локализации системы можно создать полный пакет локализации для каждого языка системы.

Для этого необходимо:

- Перейти на страницу **Сбор переводов** ([Настройки > Локализация > Выгрузка и загрузка](#)).

Сбор переводов

\* Выберите язык для сборки: ru

\* Введите дату сборки (YYYYMMDD): 20122908

Сконвертировать в национальную кодировку:

Выберите кодировку: windows-1251

Упаковать файлы (tar.gz):

**Собрать локализацию**

- Выбрать необходимые опции в закладке **Сбор переводов**:
  - язык для сборки;
  - дату сборки в формате YYYYMMDD;
  - выбрать национальную кодировку, в которую будут сконвертированы все файлы локализации (по желанию, иначе файлы будут собраны в текущей кодировке сайта);
  - по желанию упаковать файлы в архив tar.gz;
- Для запуска процесса сбора пакета локализации нажать кнопку **Собрать локализацию**.

После того, как пакет будет собран, будет предоставлена ссылка, где хранится архив (если была выбрана опция упаковки в архив tar.gz) или файлы всего пакета.

Чтобы импортировать пакет локализации в систему, необходимо:

- Перейти на страницу **Сбор переводов** ([Настройки > Локализация > Выгрузка и загрузка](#)):

Загрузка переводов

\* Веберите файл (tar.gz): \_install\_13092010.tar.gz [Обзор...](#)

\* Выберите язык для сборки (если язык отсутствует в списке, то сначала [добавьте](#) его): ru

Локализация в национальной кодировке:

Выберите кодировку: windows-1251

**Загрузить локализацию**

- Выбрать необходимые опции в закладке **Загрузка переводов:**
  - файл с архивом tar.gz;
  - язык для сборки (необходимый язык интерфейса должен быть уже установлен в системе, либо можно его [создать](#) по ссылке **Добавить**);
  - выбрать национальную кодировку, в которую были ранее сконвертированы все файлы локализации;
- Для запуска процесса импорта пакета локализации в систему нажать кнопку **Загрузить локализацию**.

### Изменение фраз в компонентах и модулях

Иногда при разработке дизайна проекта, необходимо изменить несколько слов или фраз в стандартных компонентах или (что хуже) модулях. Если шаблон компонента можно скопировать и кастомизировать, то с самими компонентами и тем более модулями все плохо. Сложно что-то изменить, не отказавшись от обновлений.

Тем не менее, есть технология замены любых языковых фраз продукта. Решение не идеальное, но оно дает довольно большой простор для полета фантазии разработчика. Суть технологии в том, что языковые фразы продукта после подключения языкового файла заменяются на фразы, определенные разработчиком. Список замен должен находиться в файле `/bitrix/php_interface/user_lang/<код языка>/lang.php`

В этом файле должны определяться элементы массива \$MESS в виде \$MESS [ 'языковой файл' ] [ 'код фразы' ] = 'новая фраза', например:

```
<?
$MESS["/bitrix/components/bitrix/system.auth.form/templates/.default/lang/ru/template.php"]
]["AUTH_PROFILE"] = "Мой любимый профиль";
$MESS["/bitrix/modules/main/lang/ru/public/top_panel.php"]["top_panel_tab_view"] =
"Смотрим";
$MESS["/bitrix/modules/main/lang/ru/interface/index.php"]["admin_index_sec"] =
"Проактивка";
?>
```

Первая строка меняет текст ссылки в компоненте формы авторизации; вторая строка меняет название вкладки публичной панели; третья меняет строку для индексной страницы панели управления.

Теоретически могут быть проблемы сопровождения этого файла при изменении кода фразы или расположения языкового файла, но на практике такие изменения бывают крайне и крайне редки.

## Включаемые области

**Включаемая область** - это специально выделенная область на странице сайта, которую можно редактировать отдельно от основного содержания страницы.

Включаемые области служат для размещения справочной информации, различных форм (подписки, голосования, опросов), новостей и любой другой статической и динамической информации. Также в виде включаемой области могут быть выполнены области с указанием авторских прав, графические ссылки, контактная информация, логотип компании и т.п.

В разделе представлена информация по созданию и использованию включаемых областей для шаблонов и страниц сайта.

## Использование включаемых областей

В системе существует возможность создания различных типов включаемых областей, например:

- **включаемая область страницы** – выводится только при просмотре определенной страницы;

The screenshot shows a website interface with a red header bar containing 'Магазин', 'Общение', and 'Типовые примеры'. Below the header, the URL 'Главная > Контент > Статьи' is visible. A sidebar on the left is titled 'Статьи' and contains a search bar with 'Поиск:' and a 'Поиск' button, along with a small RSS icon. The main content area displays two articles: '06.03.2007 Компоненты 2.0: настройка поддержки ЧПУ' and '06.03.2007 Инструменты для отладки производительности'. To the right of the content is a blue-bordered box titled 'Публикация статей' containing descriptive text about news block parameters and article splitting.

На данной странице представлены элементы инфоблока статьи, относящиеся к типу Публикации. Статьи опубликованы на странице с помощью комплексного компонента Новостной раздел с указанием параметров инфоблока Статьи. При создании каждой статьи был использован тег `<break />`, который позволяет осуществить разделение статьи на страницы. В публичном разделе такая статья будет представлена в виде нескольких страниц в зависимости от количества тегов `<break />`

- **включаемая область раздела** – выводится на всех страницах определенного раздела сайта;



Главная Профиль Контент Магазин Общение Типовые примеры

**Отчет**

**Инструкции**

**Регистрация**

**Планы**

**Отчет по программе**

**Поиск по сайту**

Поиск

**Авторизация**

Administrator  
[admin]  
[Мой профиль](#)

[Выйти](#)

**Подписка на рассылку**

Новости компании

Главная > Интернет-магазин > Аффилиаты

### Счет аффилиата

Фильтр по периоду

Период:	01.05.2008	... 09.08.2008
---------	------------	----------------

[Установить](#) [Сбросить](#)

Дата	Приход	Расход	Комментарий
За указанный период движений денег по счету нет			
На счете на 08.08.2008	0.00	руб	

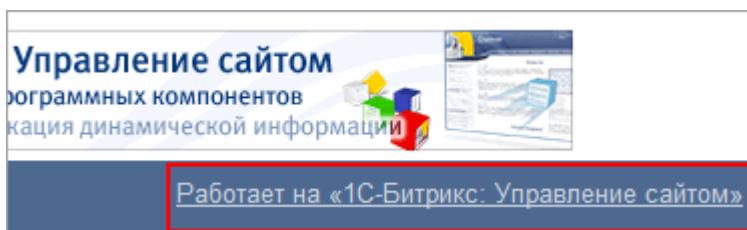
**Аффилиаты**

Аффилиат – это вид партнера компании, который непосредственно не занимается продажей товара, а располагает у себя на сайте ссылку на Интернет-магазин компании, и пользователи его сайта, перешедшие по этой ссылке и купившие товар в магазине, считаются им привлеченными. Аффилиату от таких продаж перечисляется определенный процент или фиксированная сумма, предусмотренная его планом.

На данной странице будет представлена информация по движению денежных средств на счете в случае, если пользователь является аффилиатом.

Авторизованному пользователю, не являющемуся аффилиатом, будет предложена форма регистрации аффилиата. Для неавторизованного пользователя будет выведена форма авторизации и форма для регистрации на сайте.

- **включаемый файл** – в области выводится информация какого-либо файла (например, название компании, информация об авторских правах и т.д.).



Количество включаемых областей может быть расширено. При этом потребуется некоторая модификация шаблона сайта (например, добавление в шаблон дизайна компонентов или php-кода для подключения дополнительных редактируемых областей).

Кроме того, включаемые области могут отображаться на страницах сайта в соответствии с любыми другими условиями. Например, только на главной странице сайта или только для авторизованных посетителей, и т.д.



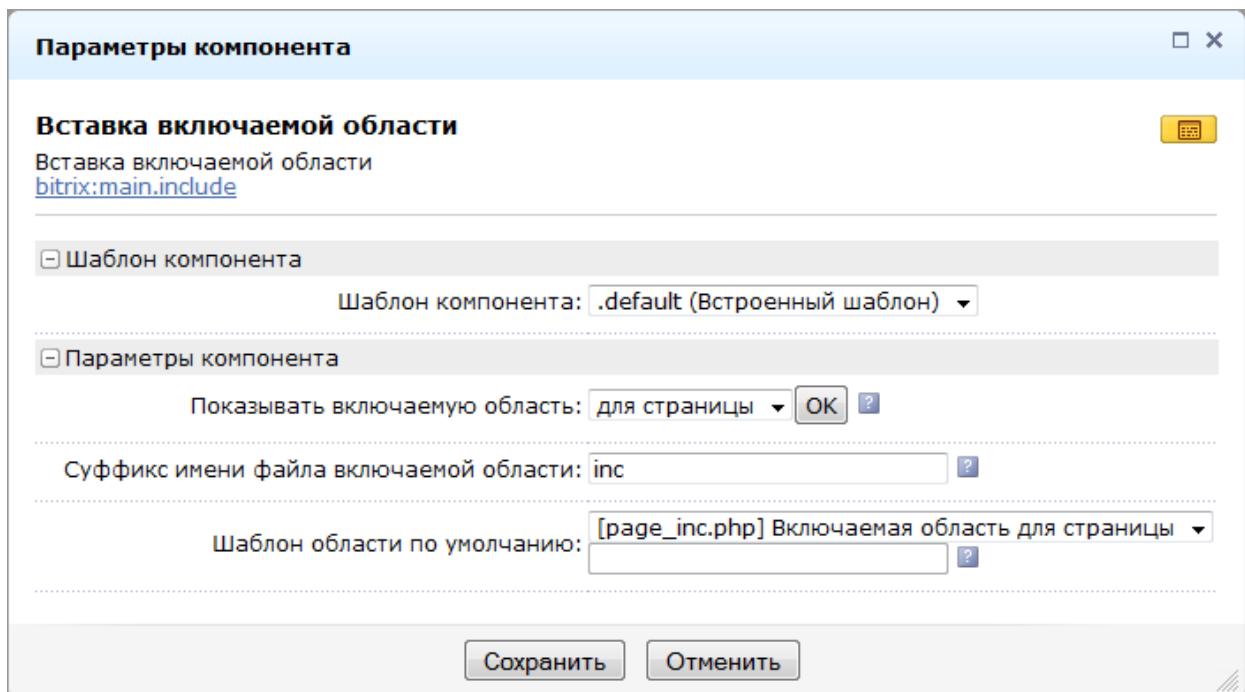
## Управление включаемыми областями

Содержимое включаемых областей хранится в отдельных PHP или HTML файлах. Области для страниц или разделов сохраняются с некоторым суффиксом. Например, в поставляемых файлах продукта в качестве обозначения включаемой области для страницы используется суффикс **\_inc** (например, `index_inc.php`), а включаемая область для раздела сайта сохраняется в файле с именем **sect** и добавлением к нему суффикса (например, `sect_inc.php`).

**⚠ Важно!** *Файл с включаемой областью должен быть сохранен в той же директории, что и страница, для которой он был создан. Включаемая область для раздела - в папке этого раздела.*

Подключение областей в шаблоне дизайна сайта выполняется с помощью компонента **Вставка включаемой области** либо с помощью функции `IncludeFile()`.

Суффикс, используемый для обозначения включаемых областей, определяется одноименной опцией в настройках компонента **Вставка включаемой области**. Компонент можно размещать не только в шаблоне дизайна, но и страницах сайта с условием, что суффикс файла должен быть задан отличным от того, который используется в шаблоне.



**⚠ Примечание:** Тип включаемой области определяется опцией **Показывать включаемую область**.

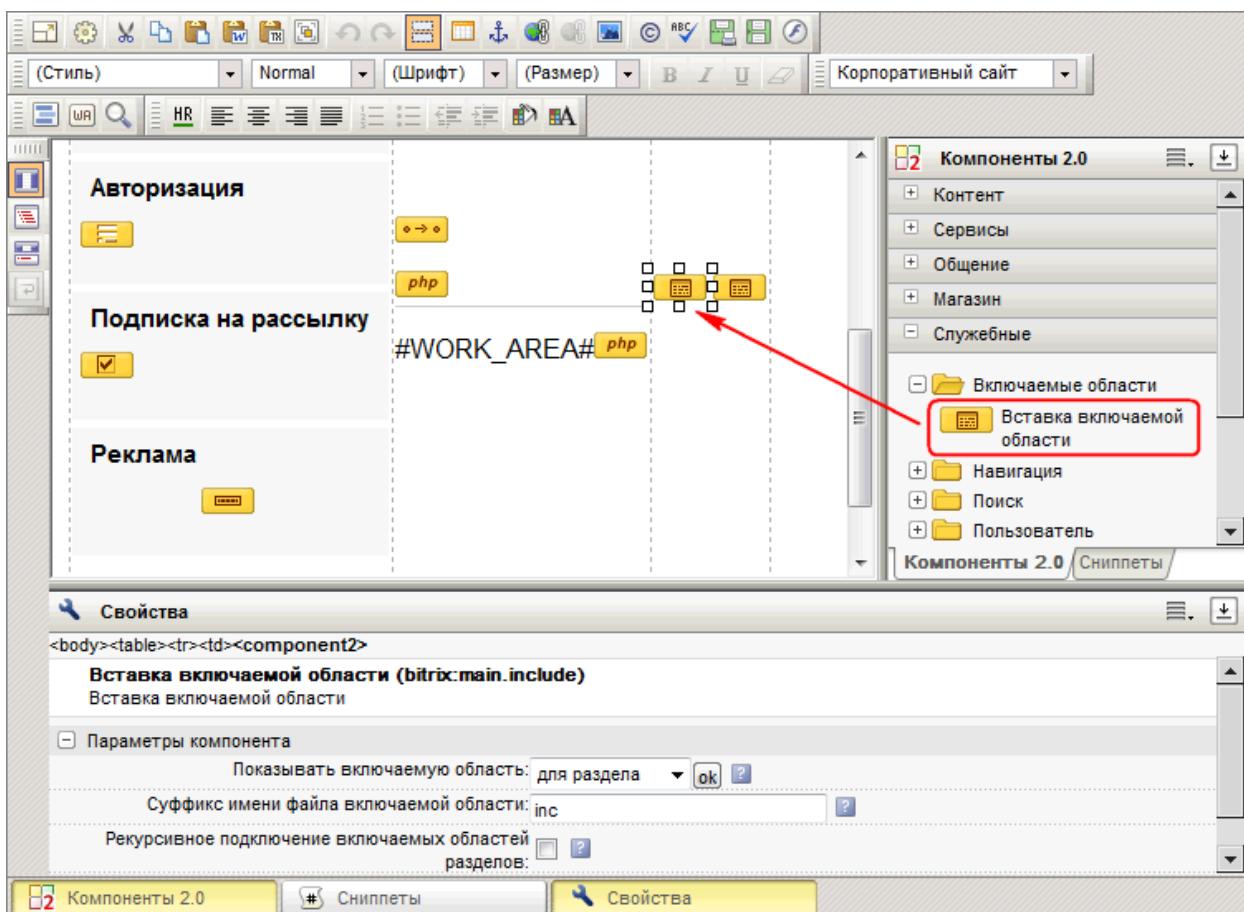
Если компонент расположить в шаблоне дизайна сайта, то информация из файла будет выводиться на всем сайте. Установка параметра доступна только пользователю с правами **edit\_php**.

**⚠ Примечание:** Подробное описание параметров компонента смотрите на странице [пользовательской документации](#).

## Размещение включаемой области

Для размещения включаемой области выполните следующее:

- Откройте для редактирования шаблон сайта или страницу в визуальном редакторе.
- Добавьте компонент **Вставка включаемой области** в шаблон сайта (или в тело страницы) и настройте его параметры.



## Создание и редактирование включаемой области

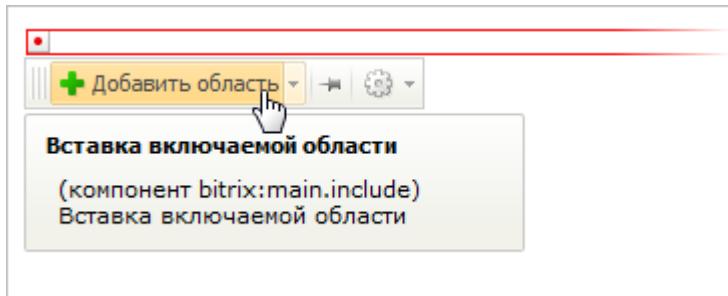
Создание включаемых областей может быть выполнено:

- из административного раздела в **Менеджере файлов** ([Контент > Структура сайта > Файлы и папки](#)), создав файл с соответствующим именем;



- из публичного раздела сайта в режиме правки. В тех местах, где предполагается вывод включаемых областей, будут показаны иконки для быстрого перехода к созданию этих областей.

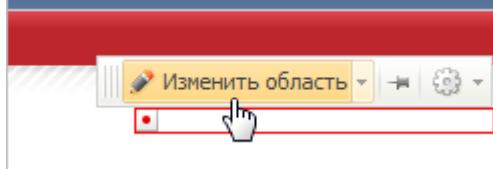
**⚠ Примечание:** Файл включаемой области будет создан и назван в соответствии указанным в настройках компонента суффиксом - для опции **для раздела**, или именем файла - для опции **из файла**.



После выбора команды **Добавить область** будет запущен визуальный редактор для создания содержимого включаемой области.

Аналогично перейти к редактированию включаемых областей можно:

- непосредственно из публичного раздела сайта в режиме правки;



- либо из административного раздела, открыв для редактирования соответствующий файл в **Менеджере файлов**.

	Имя	Размер файла	Изменен	Тип	Права на доступ продукта
<input type="checkbox"/>	..				
<input type="checkbox"/>	blog		02.02.2012 11:58:55	Папка	Полный доступ
<input type="checkbox"/>	forum		02.02.2012 11:58:55	Папка	Полный доступ
<input type="checkbox"/>	gallery		02.02.2012 11:58:55	Папка	Полный доступ
<input type="checkbox"/>	Меню типа «left»	805 Б	02.02.2012 11:58:55	Скрипт PHP	Полный доступ
<input type="checkbox"/>	index.php	7 КБ	02.02.2012 11:58:55	Скрипт PHP	Полный доступ
<input type="checkbox"/>	sect_inc.php	669 Б	02.02.2012 11:58:55	Скрипт PHP	Полный доступ

Выбрано: 6 Отмечено: 0

**⚠ Внимание!** Если в качестве включаемой области будет использоваться вариант **из файла**, то необходимо проверить, что файл подключен из системы, а не вызван напрямую. Делается это с помощью следующей строки:



```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
```

Пример содержимого включаемого файла:

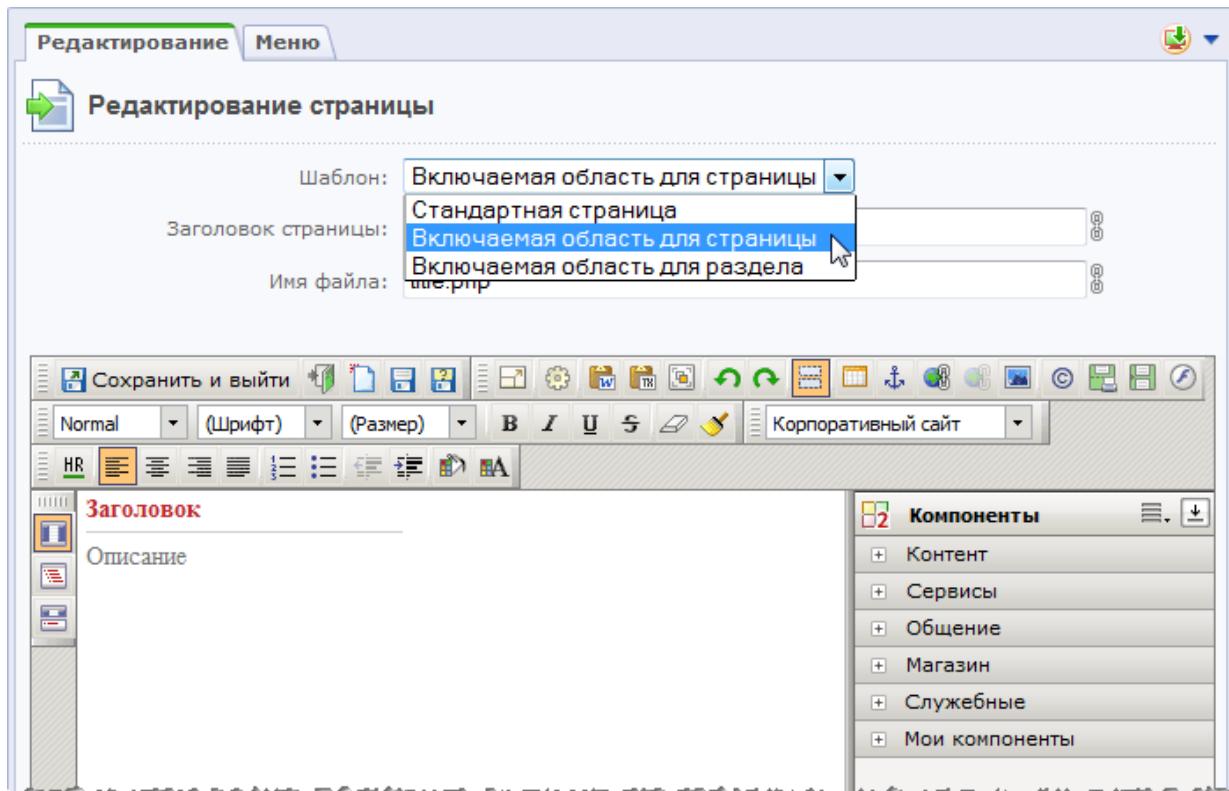
```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
<div class="bx-main-title">World Book</div>
<span class="bx-main-subtitle">Все книги мира</span>
```

## Шаблоны включаемых областей

Включаемые области создаются на основе шаблонов, хранящихся в папках с именем /page\_templates/:

- /bitrix/templates/.default/page\_templates/ - если данный шаблон включаемой области используется для всех шаблонов дизайна сайта;
- /bitrix/templates/<идентификатор шаблона>/page\_templates/ - если для шаблона сайта используются отдельные шаблоны включаемых областей.

Чтобы в визуальном редакторе можно было выбирать шаблон, на основе которого создается редактируемая область, список шаблонов для редактируемых областей должен быть добавлен в файл .content.php.





Файл .content.php хранится в папке /page\_templates/ в каталоге соответствующего шаблона сайта.

Пример содержимого файла:

```
<?
if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();

IncludeTemplateLangFile(__FILE__);

$TEMPLATE["standard.php"] = Array("name"=>GetMessage("standart"), "sort"=>1);
$TEMPLATE["page_inc.php"] = Array("name"=>GetMessage("page_inc"), "sort"=>2);
$TEMPLATE["sect_inc.php"] = Array("name"=>GetMessage("sect_inc"), "sort"=>3);
?>
```

Следует обратить внимание на то, что имя шаблона может быть передано параметром при подключении редактируемой области в шаблоне сайта (см. выделение синим цветом в примере ниже).

Если подключение редактируемых областей выполняется с помощью PHP функции `IncludeFile()`, помещаемой в соответствующие места шаблона дизайна, то код может иметь такой вид:

```
<?
$APPLICATION->IncludeFile(substr($APPLICATION->GetCurPage(),
0, strlen($APPLICATION->GetCurPage())-4)."._inc.php",
Array(),
Array("MODE"=>"html", "NAME"=>GetMessage("PAGE_INC"),
"TEMPLATE"=>"page_inc.php"));
?>
<?
$APPLICATION->
IncludeFile($APPLICATION->GetCurDir()."sect_inc.php",
Array(), Array("MODE"=>"html",
"NAME"=>GetMessage("SECT_INC"),
"TEMPLATE"=>"sect_inc.php"));
?>
```

### Удаление включаемых областей из демонстрационного шаблона

В коде шаблона необходимо закомментировать или удалить строку, отвечающую за подключение включаемой области. Обычно, это файлы, заканчивающиеся на `_inc.php`.

Строка имеет примерно следующий вид:

```
<?
```

```
$APPLICATION->IncludeFile(  
substr($APPLICATION->GetCurPage(), 0, strlen($APPLICATION->GetCurPage())-4)."_inc.php",  
Array(),  
Array("MODE"=>"html", "NAME"=>GetMessage("PAGE_INC"), "TEMPLATE"=>"page_inc.php"));  
>
```

## Средства навигации

Средства навигации включаются в шаблон сайта. Для их реализации рекомендуется использовать специальные компоненты.

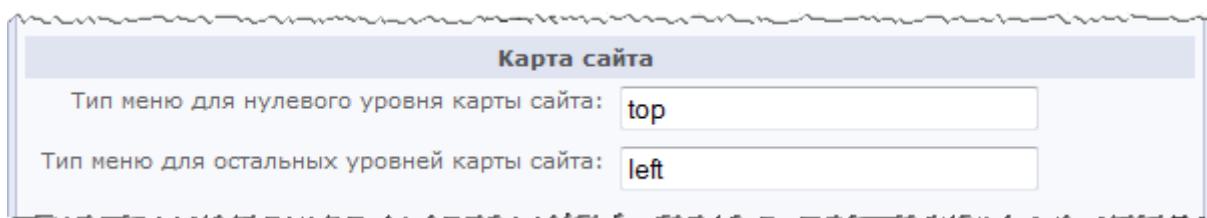
### Карта сайта

**Карта сайта** - один из элементов навигации на сайте, считается стандартным элементом современного сайта. Как правило, заголовки страниц в списке служат ссылками на эти страницы.

### Создание карты сайта

Для создания карты сайта используется компонент **Карта сайта** (*bitrix:main.map*). Компонент расположен в папке *Контент > Карта сайта* панели **Компоненты 2.0** визуального редактора.

Карта строится на основе меню, используемых в системе. Типы меню указываются в настройках главного модуля, раздел **Карта сайта** (*Настройки > Настройки продукта > Настройки модулей > Главный модуль*).



**⚠ Примечание:** В качестве основы для построения опорного уровня и ветвей карты сайта можно использовать сразу несколько типов, перечисляя их через запятую.

- Создайте отдельную страницу и разместите на ней компонент **Карта сайта**.
- Настройте параметры компонента



**Параметры компонента**

**Карта сайта (bitrix:main.map)**  
Карта сайта

**Шаблон компонента**  
Шаблон компонента:

**Настройки кеширования**  
Тип кеширования:    
Время кеширования (сек.):

Текущие настройки ядра: **автокеширование** включено, **управляемый кеш** включен. Если компонент поддерживает управляемое кеширование, вы можете увеличить время кеширования. [Изменить настройки кеширования](#).

**Дополнительные настройки**  
Устанавливать заголовок страницы:

**Настройки компонента**  
Максимальный уровень вложенности (0 - без вложенности):    
Количество колонок:    
Показывать описания:

**Максимальный уровень вложенности** – это глубина вложенности разделов, которая должна отображаться в меню. Если число в этом поле более чем 4, то рекомендуется еще раз продумать структуру сайта.

**Количество колонок** – укажите во сколько колонок будет строиться изображение карты сайта. Это число во многом зависит от дизайна сайта, используемого уровня вложенности, структуры сайта. Перенос пунктов карты по колонкам происходит на базе пунктов меню первого уровня. То есть, если один из пунктов меню имеет структуру намного более глубокую, чем другие пункты, то при выборе глубокого уровня вложенности на карте сайта вы можете получить длинную колонку этого пункта и короткие колонки с другими пунктами.

**Показывать описания** – будут показаны описания разделов - физических папок. Описания разделов информационных блоков не отображаются.

**⚠ Примечание:** Если в настройках сайта (**Настройки > Настройки продукта > Сайты > Список сайтов > \_ имя сайта \_**) не совсем верно указана папка сайта (например, нет закрывающего слеша), то компонент не сможет создать карту сайта.

## Примеры работы

**Скажите как в карту сайта включить разделы сайта не входящие в меню?**



Обычно для этого требуется кастомизация шаблона компонента, но есть и другой выход. Самое простое - сделать для этих разделов свой тип меню и добавить его через запяты в настройках главного модуля в опции типов меню карты сайта.

## Меню в Bitrix Framework

### Типы меню

**Тип меню** – принцип организации меню. По умолчанию в дистрибутиве используется два типа меню: Верхнее и Левое.

Типов меню может быть несколько, в зависимости от задач сайта: верхнее, левое, нижнее и т.п. В каждом компоненте меню могут быть применены два типа меню: одно как основное, второе как дополнительное при условии использования многоуровневых шаблонов.

В самом общем случае на сайте существует одно "основное" меню, соответствующее самому верхнему уровню иерархии и отображаемое во всех разделах сайта. Также в системе часто используется "второстепенное" меню (или меню второго уровня), включающее ссылки на подразделы и документы текущего раздела.

Моя компания  
Быстро. Просто. Эффективно

Главная Профиль Контент Магазин Общение Типовые примеры

**Авторы**  
**Рецензии**  
**Бизнес-литература**  
**Детская литература**  
**Компьютеры и Интернет**  
**Наука и образование**  
**Фантастика**

Главная > Интернет-магазин > Каталог книг

**Основное меню**

В вашей корзине 1 товар

- Бизнес-литература (3)
- Детская литература (3)
- Компьютеры и Интернет (8)
- Наука и образование (9)
  - История (4)
  - Политология (3)
- Фантастика (3)

Книги какого жанра вы предпочитаете?

фантастика  
 фэнтези  
 религия и философия  
 историческая  
 детектив  
 триллеры  
 юмор  
 другое

Голосовать

**Второстепенное меню**

Поиск по сайту

Поле для поиска

Поиск

Этот негодяй Балмер, или Человек, который управляет "Майкрософтом" (пер. с англ. Клигман И.)

Информация о книгах структурирована следующим образом: создан тип инфоблока Каталог книг. Для этого типа создано три информационных блока:



Меню в системе наследуемое. Это значит, что если для одного компонента Меню в шаблоне выбран определенный тип меню, то это меню будет транслироваться ниже на все разделы и страницы сайта с этим шаблоном, если в этих разделах и страницах не было созданного собственного меню. Этот механизм удобен для главного меню сайта, обычно ему присваивают тип **Верхнее**.

**⚠ Примечание:** Если необходимо, чтобы в нижележащем разделе просто не отображалось вышестоящее меню, создайте меню в нужном разделе без создания в нем пунктов меню.

Меню разделов, как правило, создается для каждого раздела свое и транслируется на все страницы раздела. При необходимости в подразделах можно создать свое собственное меню и применить к нему собственный тип.

Типы используемых на сайте меню задаются из административного раздела на странице настроек модуля **Управление структурой**.

Например, пусть в системе используются два типа меню:

Типы меню:	Тип	Название
	left	Левое меню
	top	Верхнее меню

- левое меню – тип "left";
- верхнее (основное) меню – тип "top".

Тип меню, заданный в настройках модуля **Управления структурой**, будет использован как префикс файла с шаблоном меню (например, `top.menu_template.php`), а также для идентификации файлов с пунктами меню (например, `.top.menu.php`). Кроме того, имя типа меню используется для подключения меню в шаблоне дизайна.



**! Примечание:** Типы меню могут быть заданы отдельно для каждого сайта.

Тип	Название
left	Меню раздела
top	Главное меню

Типы меню задаются произвольно (только символами латинского алфавита). Однако для упрощения управления меню рекомендуется давать типам меню значимые имена. Например, `top`, `left`, `bottom`.

### Построение и показ меню

В общем случае задача формирования меню включает:

- выделение HTML элементов для построения меню;
- создание шаблона меню (создание шаблона компонента **Меню**);
- включение функции показа меню (вызов компонента **Меню**) в общем шаблоне ("прологе" и "эпилоге");
- заполнение меню в соответствии со структурой сайта.

### Структура меню

Любое меню на сайте строится на основе двух составляющих:

- массива данных, определяющего состав меню, задает названия и ссылки для всех пунктов меню. Управление массивом данных осуществляется через административный интерфейс;
- шаблона внешнего представления меню. Шаблон меню – это PHP код, определяющий внешний вид меню (шаблон компонента **Меню**). Шаблон меню обрабатывает массив данных, выдавая на выходе HTML-код.

### Массив данных меню

Данные для каждого типа меню хранятся в отдельном файле, имя которого имеет следующий формат: `.<тип меню>.menu.php`. Например, для хранения данных меню типа `left` будет использоваться файл `.left.menu.php`, а для хранения данных меню типа `top` – файл `.top.menu.php`.



Файлы меню размещаются в папках тех разделов сайта, где требуется показ соответствующих типов меню. Если для данного раздела не создан соответствующий файл меню, система производит поиск файла в каталоге уровнем выше.

Например, т.к. основное меню (в демо-версии продукта, это меню типа **top**) должно выводиться во всех разделах, то файл данного меню помещается только в корневой каталог сайта:

**Управление структурой: Главная**

Рабочий стол > Контент > Структура сайта > Моя компания

Дополнительно

Имя:

Найти Отменить

Новая папка Новый файл Загрузить файл Добавить меню

Свойства папки Показать права на доступ для

Путь: / OK Поиск Настроить

На странице: 20 Показано 1

Имя	Размер файла	Изменен	Тип	Права на доступ
Авторизация		02.02.2012 11:57:11	Папка	Полный доступ
Социальная сеть		02.02.2012 11:58:55	Папка	Полный доступ
Общение		02.02.2012 11:58:54	Папка	Полный доступ
Контент		02.02.2012 11:59:06	Папка	Полный доступ
Интернет-магазин		02.02.2012 11:59:07	Папка	Полный доступ
Типовые примеры		02.02.2012 11:58:56	Папка	Полный доступ
Персональный раздел		02.02.2012 11:58:29	Папка	Полный доступ
Поиск		02.02.2012 11:57:12	Папка	Полный доступ
Главная		03.02.2012 15:09:39	Папка	Полный доступ
Меню типа «top»	608 Б	02.02.2012 11:58:56	Скрипт PHP	Полный доступ
PHP 1	169 Б	02.02.2012 13:32:25	Скрипт PHP	Полный доступ
PHP 404 Not Found	515 Б	02.02.2012 11:57:11	Скрипт PHP	Полный доступ
PHP Авторизация	1 КБ	02.02.2012 11:57:11	Скрипт PHP	Полный доступ
PHP Каталог книг	4 КБ	02.02.2012 11:59:09	Скрипт PHP	Полный доступ
PHP test	338 Б	08.02.2012 10:54:49	Скрипт PHP	Полный доступ

Выбрано: 15 Отмечено: 0



Соответственно меню второго уровня (в демо-версии продукта, это меню **left**) выводится отдельно для каждого раздела сайта. Поэтому в папке каждого раздела создается свой файл для данного типа меню:

**Управление структурой: Общение**

Рабочий стол > Контент > Структура сайта > Моя компания > Общение

Дополнительно

Имя:

Найти Отменить

Новая папка Новый файл Загрузить файл Добавить меню

Свойства папки Показать права на доступ для

Путь: /communication OK Поиск Настроить

На странице: 20 Показано

Имя	Размер файла	Изменен	Тип	Права на доступ продукта
...				
Блоги		02.02.2012 11:58:51	Папка	Полный доступ
Форум		02.02.2012 11:58:28	Папка	Полный доступ
e-Learning		02.02.2012 11:58:54	Папка	Полный доступ
Техническая поддержка		02.02.2012 11:58:52	Папка	Полный доступ
Опросы		02.02.2012 11:58:28	Папка	Полный доступ
Веб-формы		02.02.2012 11:58:50	Папка	Полный доступ
Меню типа «left»	627 Б	02.02.2012 11:58:54	Скрипт PHP	Полный доступ
Общение	3 КБ	02.02.2012 11:58:28	Скрипт PHP	Полный доступ

Выбрано: 8 Отмечено: 0



Управление структурой: Каталог книг

Рабочий стол > Контент > Структура сайта > Моя компания > Интернет-магазин > Каталог книг

Дополнительно

Имя:   
Найти Отменить

Новая папка Новый файл Загрузить файл Добавить меню  
Свойства папки Показать права на доступ для  
Путь: /e-store/books OK Поиск Настроить

На странице: 20 Показано

Имя	Размер файла	Изменен	Тип	Права на доступ
...				
Авторы		02.02.2012 11:57:38	Папка	Полный доступ
Рецензии		02.02.2012 11:57:38	Папка	Полный доступ
Меню типа «left»	206 Б	02.02.2012 11:57:38	Скрипт PHP	Полный доступ
Каталог книг	4 КБ	02.02.2012 11:57:38	Скрипт PHP	Полный доступ

Выбрано: 4 Отмечено: 0

Система **Bitrix Framework** позволяет также создавать меню динамического типа. Т.е. массив данных таких меню генерируется автоматически на основании некоторых данных, получаемых с помощью программного кода. Данный код должен храниться в папке соответствующего раздела сайта в файле с именем .<тип меню>.menu\_ext.php.

Примером такого меню может служить левое меню раздела **Каталог книг**, представленное в демо-версии продукта. Здесь первые два пункта меню **Авторы** и **Рецензии** созданы обычным способом, а остальные (**Бизнес-литература**, **Детская литература** и т.д.) формируются динамически.



The screenshot shows a web interface for managing web projects using the 1C-Bitrix system. At the top, there's a banner with a woman wearing a headset and a computer monitor, followed by the text "Моя компания" and "Быстро. Просто. Эффективно". Below the banner is a navigation menu with tabs: Главная, Профиль, Контент, Магазин (which is highlighted in orange), Общение, and Типовые примеры.

The main content area has two columns. The left column contains a sidebar with categories: Авторы, Рецензии, Бизнес-литература (highlighted in red), Детская литература, Компьютеры и Интернет, Наука и образование, and Фантастика. The right column shows the "Catalog of books" (Каталог книг). It includes a breadcrumb trail: Главная > Интернет-магазин > Каталог книг. Below the breadcrumb is a message about items in the cart: "В вашей корзине 1 товар". A list of book categories is provided:

- [Бизнес-литература \(3\)](#)
- [Детская литература \(3\)](#)
- [Компьютеры и Интернет \(8\)](#)
- [Наука и образование \(9\)](#)
  - [История \(4\)](#)
  - [Политология \(3\)](#)
- [Фантастика \(3\)](#)

To the right of the list is a poll asking "Книги какого жанра вы предпочитаете?" (What genre of books do you prefer?). It lists several genres with radio buttons:  
• фантастика  
• фэнтези  
• религия и философия  
• историческая  
• детектив  
• триллеры  
• юмор  
• другое

A "Голосовать" (Vote) button is located at the bottom right of the poll area.

В данном случае в качестве пунктов меню используются названия групп каталога **Книги**, созданного на основе информационных блоков. Программный код, на основе которого генерируется меню, хранится в файле `.left.menu_ext.php` в папке `/e-store/books/`.

**Управление структурой: /e-store/books**

Рабочий стол > Контент > Структура сайта > Файлы и папки > **e-store > books**

**Дополнительно**

Имя:

Новая папка | Новый файл | Загрузить файл | Добавить меню

Свойства папки | Показать права на доступ для

Путь: **/e-store/books** |  | Поиск | Настроить

На странице: 20 | Показано

Имя	Размер файла	Изменен	Тип	Права на доступ продукта
..		11:57:38	Папка	Полный доступ
<a href="#">authors</a>		11:57:38	Папка	Полный доступ
<a href="#">reviews</a>		11:57:38	Папка	Полный доступ
Меню типа «left»	206 Б	11:57:38	Скрипт PHP	Полный доступ
<a href="#">.left.menu_ext.php</a>	512 Б	11:57:38	Скрипт PHP	Полный доступ
<a href="#">index.php</a>	4 КБ	11:57:38	Скрипт PHP	Полный доступ
<a href="#">index_inc.php</a>	882 Б	11:58:28	Скрипт PHP	Полный доступ

Выбрано: 6 | Отмечено: 0

В файлах .<тип меню>.menu.php могут использоваться следующие стандартные переменные:

- **\$sMenuTemplate** - абсолютный путь к шаблону меню (данная переменная используется крайне редко);
- **\$aMenuLinks** - массив, каждый элемент которого описывает очередной пункт меню.

Структура данного массива:

```
Array
(
    [0] => пункт меню 1
)
```

```
Array
(
    [0] => заголовок пункта меню
    [1] => ссылка на пункте меню
    [2] => массив дополнительных ссылок для подсветки пункта меню:
        Array
        (
            [0] => ссылка 1
            [1] => ссылка 2
            ...
        )
    [3] => массив дополнительных переменных передаваемых в шаблон меню:
        Array
        (
            [имя переменной 1] => значение переменной 1
            [имя переменной 2] => значение переменной 2
            ...
        )
    [4] => условие, при котором пункт меню появляется
        это PHP выражение, которое должно вернуть "true"
)
[1] => пункт меню 2
[2] => пункт меню 3
...
)
```

## Примеры файлов меню

```
<?
// пример файла .left.menu.php

$aMenuLinks = Array(
    Array(
        "Каталог курсов",
        "index.php",
        Array(),
    ),
)
```



```
Array(),
"""
),
Array(
"Мои курсы",
"mycourses.php",
Array(),
Array(),
"\$GLOBALS['USER']->IsAuthorized()"
),
Array(
"Журнал обучения",
"gradebook.php",
Array(),
Array(),
"\$GLOBALS['USER']->IsAuthorized()"
),
Array(
"Анкета специалиста",
"profile.php",
Array(),
Array(),
"\$GLOBALS['USER']->IsAuthorized()"
),
);
?>
```

```
<?
// пример файла .left.menu_ext.php

if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();

global $APPLICATION;
```



```
$aMenuLinksExt = $APPLICATION->IncludeComponent(
    "bitrix:menu.sections",
    """",
    Array(
        "ID" => $_REQUEST["ELEMENT_ID"],
        "IBLOCK_TYPE" => "books",
        "IBLOCK_ID" => "30",
        "SECTION_URL" => "/e-store/books/index.php?SECTION_ID=#ID#",
        "CACHE_TIME" => "3600"
    )
);

$aMenuLinks = array_merge($aMenuLinks, $aMenuLinksExt);
?>
```

### Организация показа меню

Показ меню на страницах сайта выполняется с помощью компонента **Меню (bitrix:menu)**. Например, вызов верхнего меню на демо-сайте имеет следующий вид:

```
<?$APPLICATION->IncludeComponent(
    "bitrix:menu",
    "horizontal_multilevel",
    Array(
        "ROOT_MENU_TYPE" => "top",
        "MAX_LEVEL" => "3",
        "CHILD_MENU_TYPE" => "left",
        "USE_EXT" => "Y"
    )
);?>
```

Данный код помещается в предусмотренные для вывода меню области шаблона сайта.

### Построение меню сайта

Построение меню для показа происходит следующим образом:

- в общий шаблон показа включается вызов вывода меню на экран;



- при загрузке компонент проверяет наличие в текущем разделе сайта файла, содержащего массив значений для меню;
- затем компонент вызывает шаблон построения для данного типа меню и выводит HTML меню на экран.

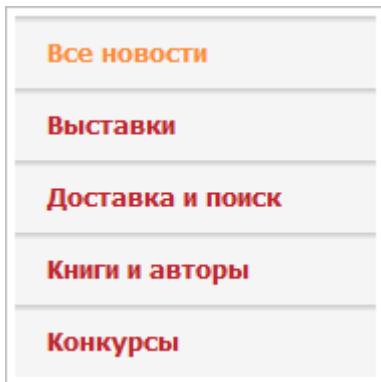
## Шаблоны меню

Вывод данных в компоненте **Меню** реализован с помощью шаблонов. Шаблоны могут создаваться пользователями самостоятельно.

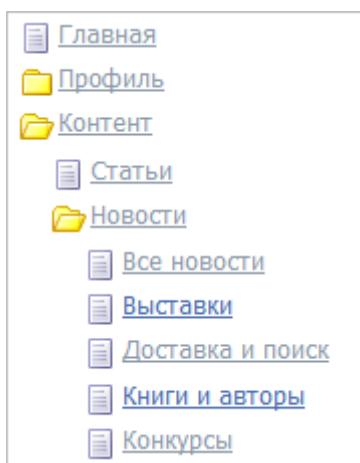
### Системные шаблоны

В дистрибутиве по умолчанию включены шесть шаблонов:

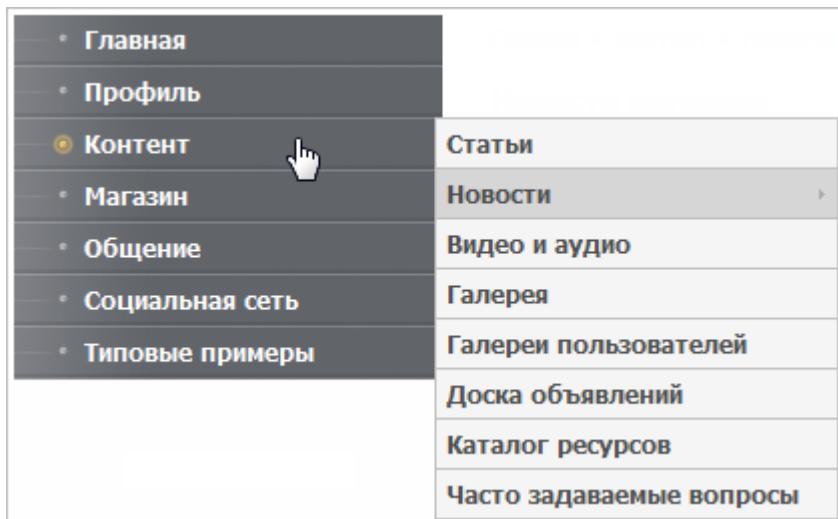
- **Default** (Вертикальное меню по умолчанию) – шаблон для вертикального меню. Самый простой шаблон. При выборе в параметрах компонента глубины вложения более 1 вы увидите список страниц сайта в общей иерархии. То есть индексная страница сайта (раздела) будет размещена на одном уровне с вложенной страницей (разделом). Это затрудняет осмысление структуры сайта посетителем. Поэтому шаблон рекомендуется для простых видов меню и главных меню верхнего уровня. Шаблон достаточно прост для кастомизации под конкретный дизайн.



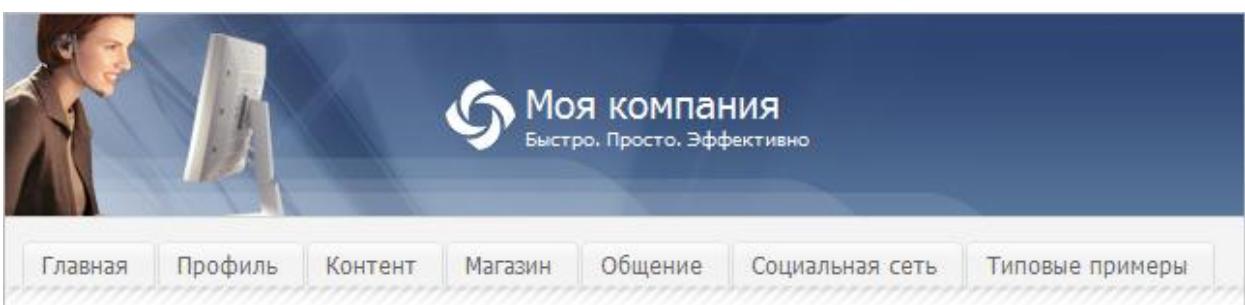
- **Tree** (Древовидное меню) – шаблон для вертикального меню. Реализует меню в виде древовидной структуры аналогично Проводнику Windows. Вложенные страницы показаны в виде страниц в папке (разделе), что существенно облегчает понимание пользователем структуры сайта. Это меню не всегда удобно при разветвленной структуре интернет-проекта, так как существенно растягивает по вертикали колонку, где оно расположено.



- **Vertical\_multilevel** (Вертикальное многоуровневое выпадающее меню) – шаблон для вертикального меню. Реализует меню с выпадающими пунктами меню нижнего уровня, что сохраняет легкость восприятия структуры сайта посетителем, характерное для древовидного меню, но при этом не растягивает дизайн при разветвленной структуре.

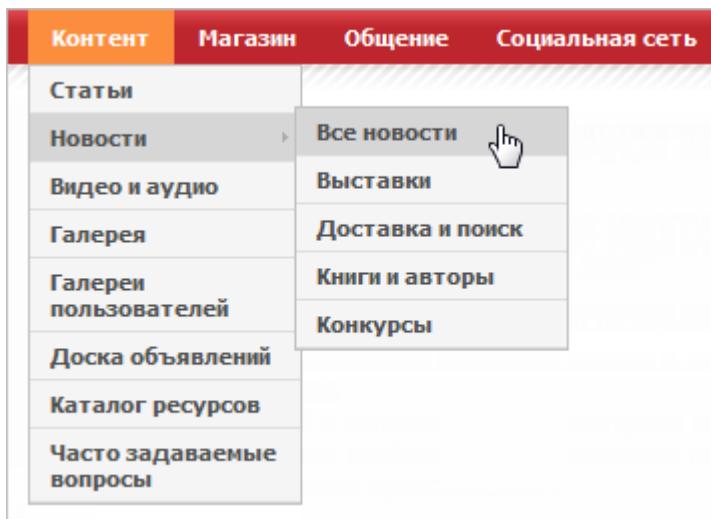


- **Grey\_tabs** (Серое меню в виде закладок) и **Blue\_tabs** (Голубое меню в виде закладок) – шаблоны для горизонтального меню. Отличаются только внешним видом. Это самые простые шаблоны, аналогичные шаблону по умолчанию (**default**) для вертикального меню.





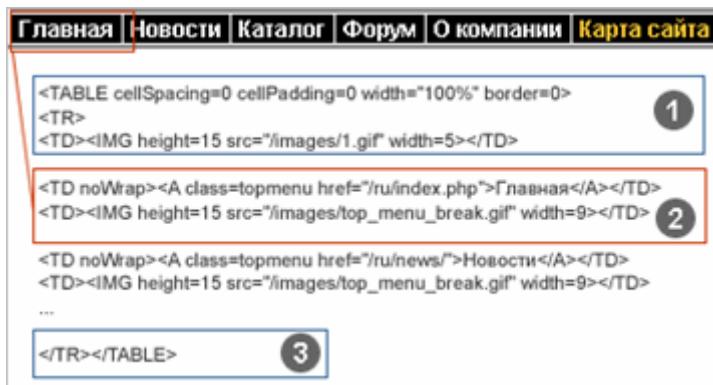
- **Horizontal\_multilevel** (Горизонтальное многоуровневое выпадающее меню) – шаблон для горизонтального меню. Аналогично **Vertical\_multilevel** (Вертикальному многоуровневому выпадающему меню) и реализует меню с выпадающими пунктами меню нижнего уровня.



## Создание Шаблонов меню

### Выделение HTML элементов для построения меню

Создание шаблонов меню начинается с выделения необходимых HTML областей в шаблоне сайта:



- неизменной верхней и нижней части шаблона;
- повторяющихся элементов. Например, для горизонтального меню – это ячейки таблицы, а для вертикального – строки.

## Создание шаблона меню

Все шаблоны меню имеют одинаковую структуру:

- область пролога шаблона меню;

- область с описанием замен для различных условий обработки шаблона;
- область тела шаблона меню;
- область эпилога шаблона меню.

В php шаблоне для вывода меню используется массив **\$arItem** - копия массива пунктов меню, в котором каждый пункт в свою очередь представляет собой массив, использующий следующие параметры:

- **TEXT** - заголовок пункта меню;
- **LINK** - ссылка на пункте меню;
- **SELECTED** - активен ли пункт меню в данный момент, возможны следующие значения:
  - **true** - пункт меню выбран;
  - **false** - пункт меню не выбран;
- **PERMISSION** - право доступа на страницу, указанную в **LINK** для текущего пользователя. Возможны следующие значения:
  - **D** - доступ запрещён;
  - **R** - чтение (право просмотра содержимого файла);
  - **U** - документооборот (право на редактирование файла в режиме документооборота);
  - **W** - запись (право на прямое редактирование);
  - **X** - полный доступ (право на прямое редактирование файла и право на изменение прав доступа на данный файл);
- **ADDITIONAL\_LINKS** - массив дополнительных ссылок для подсветки меню;
- **ITEM\_TYPE** - флаг, указывающий на тип ссылки, указанной в **LINK**, возможны следующие значения:
  - **D** - каталог (**LINK** заканчивается на "/");
  - **P** - страница;
  - **U** - страница с параметрами;
- **ITEM\_INDEX** - порядковый номер пункта меню;
- **PARAMS** - ассоциативный массив параметров пунктов меню. Параметры задаются в расширенном режиме редактирования меню.

Рассмотрим построение шаблона меню на примере **Левого меню**, представленного в демо-версии продукта (шаблон .default компонента **Меню bitrix:menu**):

```
<?if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
<?if (!empty($arResult)):?>
<ul class="left-menu">
```

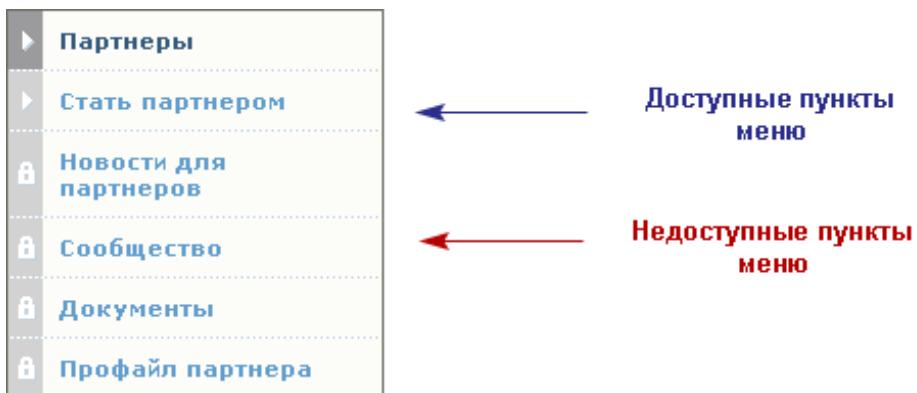


```
<?foreach($arResult as $arItem):?>
<if($arItem["SELECTED"]):?>
<li><a href=<?= $arItem["LINK"]?>" class="selected">
<?= $arItem["TEXT"]?></a></li>
<else:&?>
<li><a href=<?= $arItem["LINK"]?>">
<?= $arItem["TEXT"]?></a></li>
<endif?>
<endforeach?>
</ul>
<endif?>
```

Повторяющаяся часть меню, выделенная на предыдущем шаге, выносится в тело шаблона меню.

При создании шаблона меню потребуется также создать дополнительные стили в таблице стилей (CSS). Например, для текстового меню: цвет пункта меню и цвет текущего (активного) пункта меню.

Отдельного представления в шаблоне могут потребовать заголовки разделов (например, название текущего раздела при просмотре подразделов). Также можно предусмотреть использование графических или текстовых обозначений, например, того, что данный пункт ссылается на подразделы или документ текущего раздела и т.д.



**! Примечание:** Все шаблоны меню хранятся в папке компонента:  
`/bitrix/components/bitrix/menu/templates/`.

Быстрый доступ к редактированию шаблона каждого типа меню можно осуществить в режиме **Правки** с помощью пункта **Редактировать шаблон компонента** меню командной кнопки по управлению компонентом.



С помощью информационных блоков могут быть реализованы:

**! Примечание:** Шаблон меню, если он является системным, перед изменением необходимо скопировать в текущий шаблон сайта.

При редактировании такого шаблона из публичной части сайта системой автоматически будет предложена возможность копирования:

**Копирование шаблона компонента**

**Меню**  
Выводит меню указанного типа  
bitrix:menu

Системный шаблон компонента перед изменением необходимо скопировать в шаблон сайта.

Текущий шаблон компонента: **.default** (Встроенный шаблон)

Название нового шаблона компонента:

Копировать в шаблон сайта:  по умолчанию / **.default** (Общий шаблон)  
 текущий / books (Корпоративный сайт)  
 другой:

Применить новый шаблон компонента:

Перейти к редактированию шаблона:

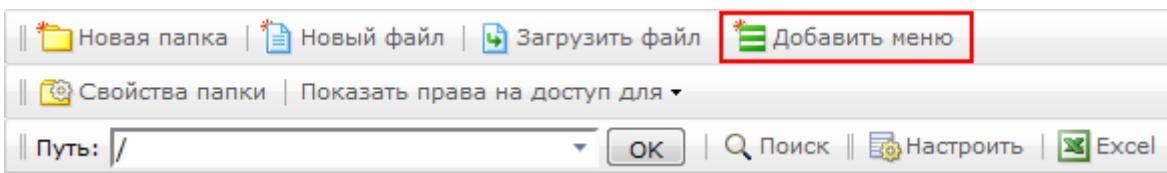
**Сохранить** **Отменить**

## Управление меню

Управление меню выполняется как с помощью средств административного, так и публичного раздела. Про управление меню из публичного раздела читайте в курсе [Контент-менеджер](#).

## **Создание меню**

Перейти к созданию меню раздела из административного раздела с помощью кнопки **Добавить меню**, расположенной на контекстной панели **Менеджера файлов**:



Меню будет создано для раздела, папка которого открыта в данный момент в **Менеджере файлов**.

**⚠ Примечание:** В результате данных операций создается файл данных меню с именем .<тип\_ меню>.тепи.php. Однако в менеджере файлов имя файла данных автоматически представляется в виде ссылки **Меню типа "<тип\_меню>"**.

### Редактирование меню

**⚠ Примечание:** При редактировании меню выполняется изменение файла .<тип\_ меню>.тепи.php (например, .top.тепи.php). Однако работа с данным файлом ведется через специальный интерфейс системы. Это позволяет исключить необходимость работы непосредственно с программным кодом и дает возможность редактировать пункты меню в визуальном режиме.

Перейти к редактированию меню из **административного раздела** можно, открыв на редактирование файл соответствующего меню в **Менеджере файлов**.



**Управление структурой: Общение**

Рабочий стол > Контент > Структура сайта > Моя компания > Общение

Дополнительно

Имя:   
Найти Отменить

Новая папка Новый файл Загрузить файл Добавить меню  
Свойства папки Показать права на доступ для  
Путь: /communication OK Поиск Настроить

На странице: 20 Показано

Имя	Размер файла	Изменен	Тип	Права на доступ продукта
...				
Блоги		02.02.2012 11:58:51	Папка	Полный доступ
Форум		02.02.2012 11:58:28	Папка	Полный доступ
e-Learning		02.02.2012 11:58:54	Папка	Полный доступ
Техническая поддержка		02.02.2012 11:58:52	Папка	Полный доступ
Опросы		02.02.2012 11:58:28	Папка	Полный доступ
Веб-формы		02.02.2012 11:58:50	Папка	Полный доступ
Меню типа «left»	627 Б	02.02.2012 11:58:54	Скрипт PHP	Полный доступ
PHP Общение	3 КБ	02.02.2012 11:58:28	Скрипт PHP	Полный доступ

Выбрано: 8 Отмечено: 0

В системе предусмотрено два режима редактирования меню (переключение между ними выполняется с помощью соответствующей кнопки, расположенной на контекстной панели страницы редактирования меню):

- **упрощенный режим** редактирования;



**Меню**

### Параметры меню

Тип меню: [left] Меню раздела

Название	Ссылка	Сортировка	Удалить
Форум	/communication/forum/	10	<input type="checkbox"/>
Веб-формы	/communication/web-forms/	20	<input type="checkbox"/>
Блоги	/communication/blog/	30	<input type="checkbox"/>
Техподдержка	/communication/support/	40	<input type="checkbox"/>
Опросы	/communication/voting/	50	<input type="checkbox"/>
Обучение	/communication/learning/	60	<input type="checkbox"/>
		70	
		80	

**Сохранить** **Применить** **Отменить**

Режим позволяет определить тип меню и указать название пункта меню, ссылку для перехода и значение индекса сортировки.

- **расширенный режим редактирования.**

**Меню**

### Параметры меню

Тип меню: [left] Меню раздела

Шаблон для меню: (по умолчанию)

Пункты меню:

**Вставить пункт >>**

Название: Форум Ссылка: /communication/forum/ Сортировка: 10 <input type="checkbox"/> Удалить	Доп. ссылки для подсветки:  Тип условия: [без условия] Условие: <без условия> Параметры: Название Значение =
--	---

**Вставить пункт >>**



В этом режиме для управления доступны следующие данные:

- тип редактируемого меню;
- шаблон, на основе которого будет генерироваться меню (поле используется в случае, если создаваемое меню должно генерироваться на основе шаблона, отличного от используемого по умолчанию);
- название пункта меню;
- ссылка для перехода;
- индекс сортировки;
- набор дополнительных ссылок, которые соответствуют этому же пункту меню. В данном поле задается набор ссылок на страницы, при переходе на которые будет также подсвечиваться данный пункт меню. Например, чтобы при просмотре любой страницы раздела **Каталог книг** подсвечивался пункт меню **Каталог книг**, в данном поле нужно указать ссылку на папку, содержащую все страницы раздела (или перечислить необходимые страницы): `/e-store/books/`;
- условия показа. Например, позволяет внести ограничения на показ данного пункта меню пользователям с определенными правами доступа;
- **дополнительные параметры** – набор произвольных параметров, которые могут быть обработаны в шаблоне показа меню и представлены соответствующим образом. Например, если пункт меню является заголовком секции, это может быть указано в параметрах пункта так: название параметра - **SEPARATOR**, значение - **Y**. При разработке шаблона можно проверять значение этого параметра и при показе выделять данный пункт меню разделителем.

Параметры хранятся в ассоциированном массиве **\$PARAMS** в виде пар `имя => значение`. При построении меню по шаблону, в самом шаблоне может быть добавлена проверка параметра, например:

```
if ($PARAMS["MY_PARAM"]=="Y")
```

**⚠ Примечание:** При желании количество дополнительных параметров в форме можно увеличить с помощью соответствующей опции в настройках модуля **Управление структурой**, секция **Настройки для сайтов**.

**⚠ Примечание:** Подробное описание всех полей формы можно посмотреть на странице [пользовательской документации](#).

## Примеры создания меню

### Предварительные операции

Для примера мы создадим в структуре сайта новый раздел **test\_menu**, который также добавим в главное меню.



**Мастер создания нового раздела**

Создание нового раздела в папке / [Создать новый раздел в Панели управления](#)

Заголовок раздела:

Имя папки:

[перейти к редактированию индексной страницы](#)

**добавить пункт меню**

Ограничить доступ к разделу (не публиковать)

[< Назад](#) [Далее >](#) [Готово](#) [Отмена](#)

**Мастер создания нового раздела**

Создание нового раздела в папке / [Создать новый раздел в Панели управления](#)

Имя нового пункта:

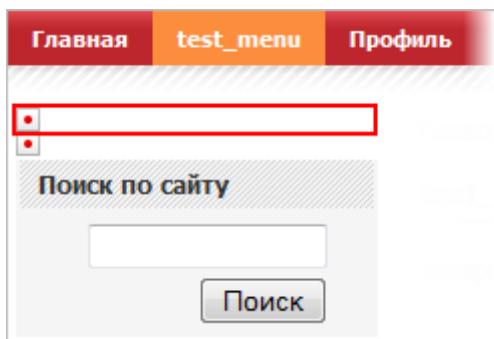
Тип меню:

Вставить перед пунктом:

[< Назад](#) [Далее >](#) [Готово](#) [Отмена](#)

Для удобства будем использовать компонент меню, который уже расположен в левой части шаблона сайта.

В параметрах компонента, для примера, в опции **Тип меню для первого уровня** укажем **Главное меню**.



**! Примечание:** По умолчанию этот компонент уже настроен для вывода **Меню раздела**, поэтому операция по созданию меню и его настройке будет пропущена.

## Древовидное меню

Древовидное меню – самое распространенное меню. Оно достаточно простое и вместе с этим информативное. Создается на базе статических и динамических элементов: разделов, страниц и инфоблоков.

**! Примечание:** При использовании древовидного меню, если необходимо указать именно раскрывающуюся папку, нужно указывать путь до папки, а не до индексного файла этой папки.

### Одноуровневое древовидное меню

**Задача:** создать древовидное меню.

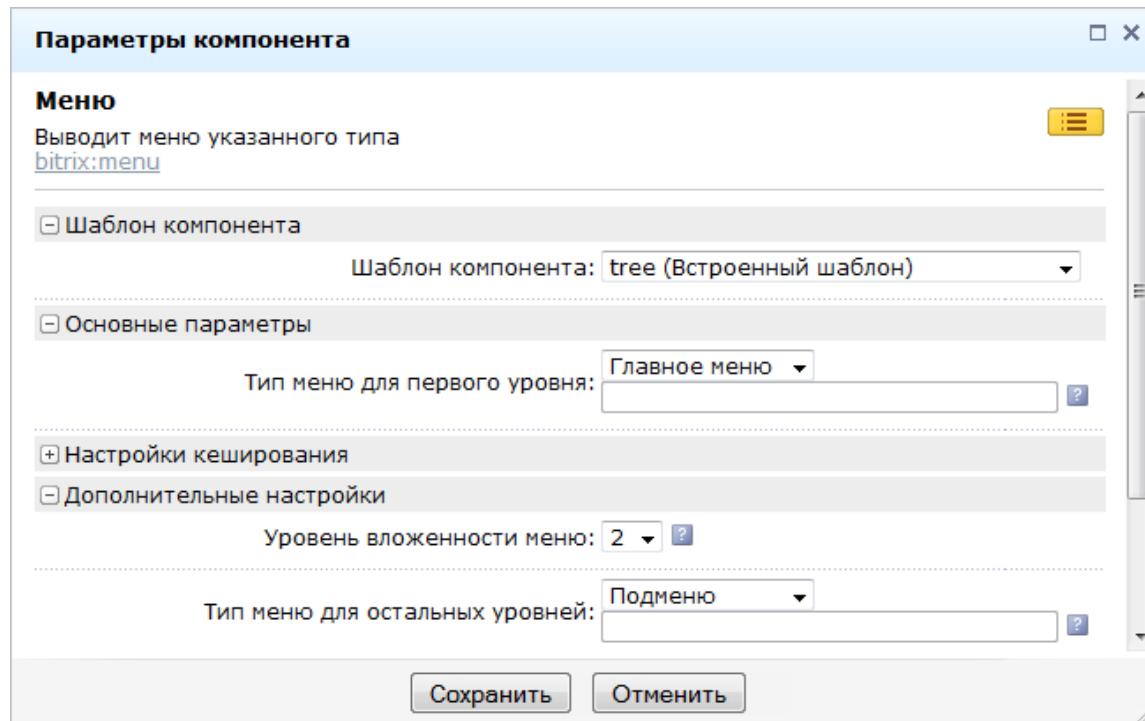
- Создайте дополнительный тип меню: **Подменю** (podmenu).

Настройки для сайтов		
Использовать индивидуальные настройки для каждого сайта:		
<input checked="" type="checkbox"/> Настройки для сайта: Моя компания		
Типы меню:	Тип	Название
	left	Меню раздела
	top	Главное меню
	podmenu	Подменю

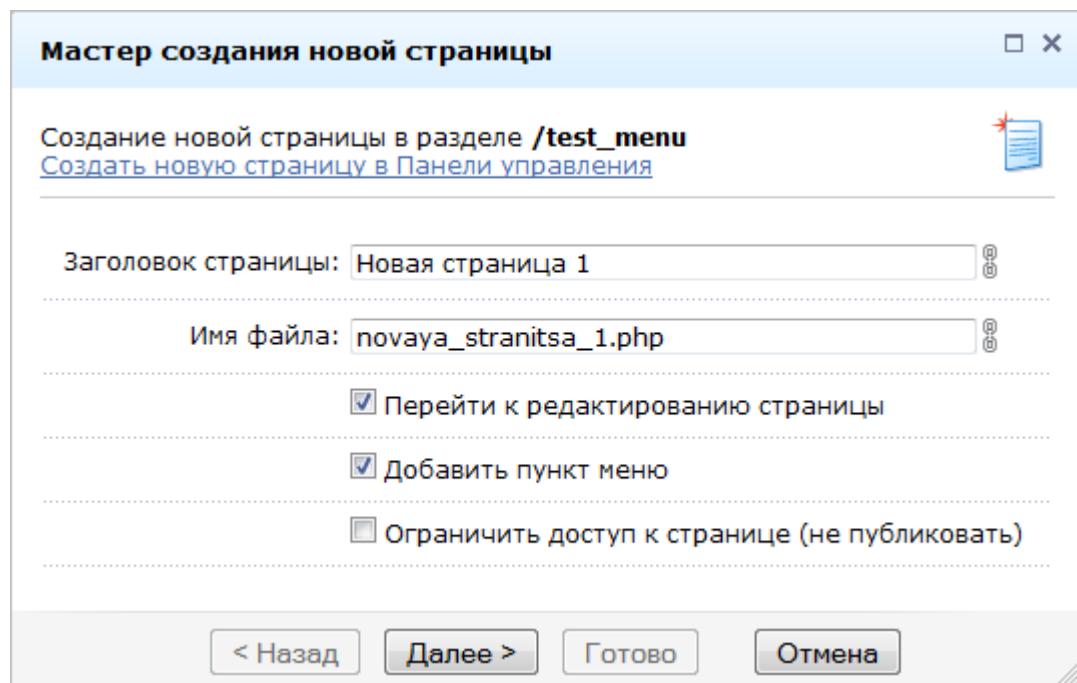
- В параметрах компонента установите следующие параметры:
  - Шаблон компонента - tree (Встроенный шаблон);
  - Тип меню для первого уровня - Главное меню;
  - Тип меню для остальных уровней - Подменю;

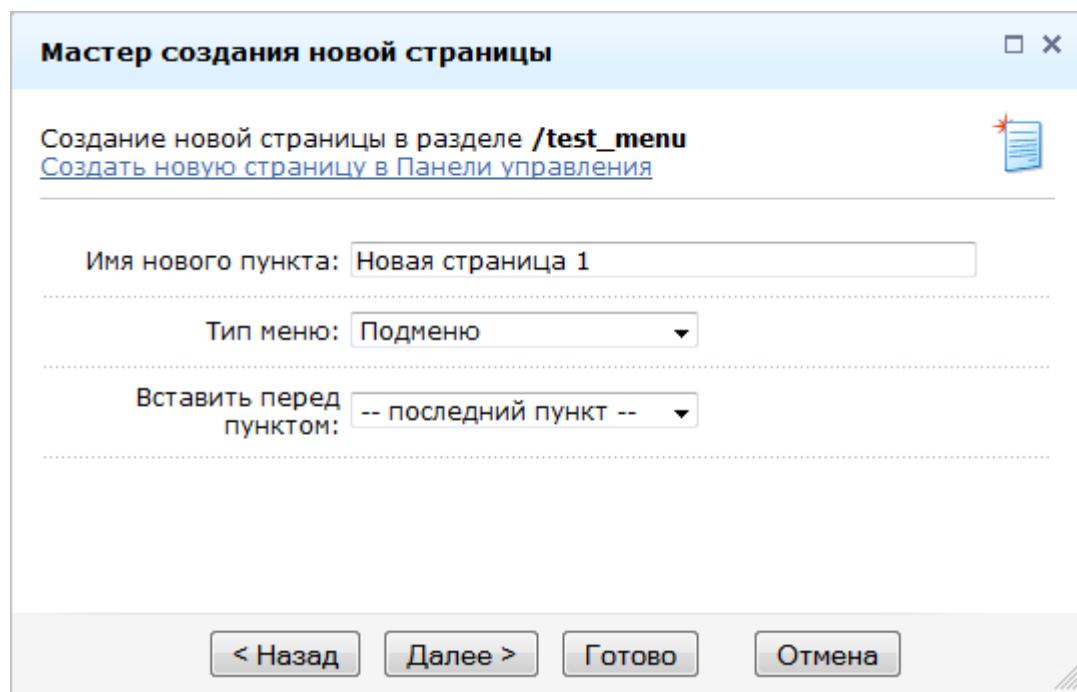


## ◦ Уровень вложенности меню - 2.



- Перейдите в раздел **test\_menu** и создайте в нем страницы и разделы. При их создании необходимо отметить опцию **Добавить пункт меню** и выбрать меню типа **Подменю**.





**Результат.** Результатом этой работы будет меню такого вида:

Каталог `/test_menu` будет содержать следующие файлы:



Упорядочить ▾ Добавить в библиотеку ▾ Общий доступ ▾

Имя
.podmenu.menu.php
.section.php
index.php
novaya_stranitsa_1.php
novaya_stranitsa_2.php

www\_BUS

- auth
- bitrix
- club
- communication
- content
- e-store
- examples
- images
- personal
- search
- site\_yj
- test\_menu
- upload

Файл .podmenu.menu.php будет иметь следующую структуру:

```
<?
$MenuLinks = Array(
    Array(
        "Новая страница 1",
        "/test_menu/novaya_stranitsa_1.php",
        Array(),
        Array(),
        """
    ),
    Array(
        "Новая страница 2",
        "/test_menu/novaya_stranitsa_2.php",
        Array(),
        Array(),
        """
    )
);
?>
```

## Многоуровневое древовидное меню

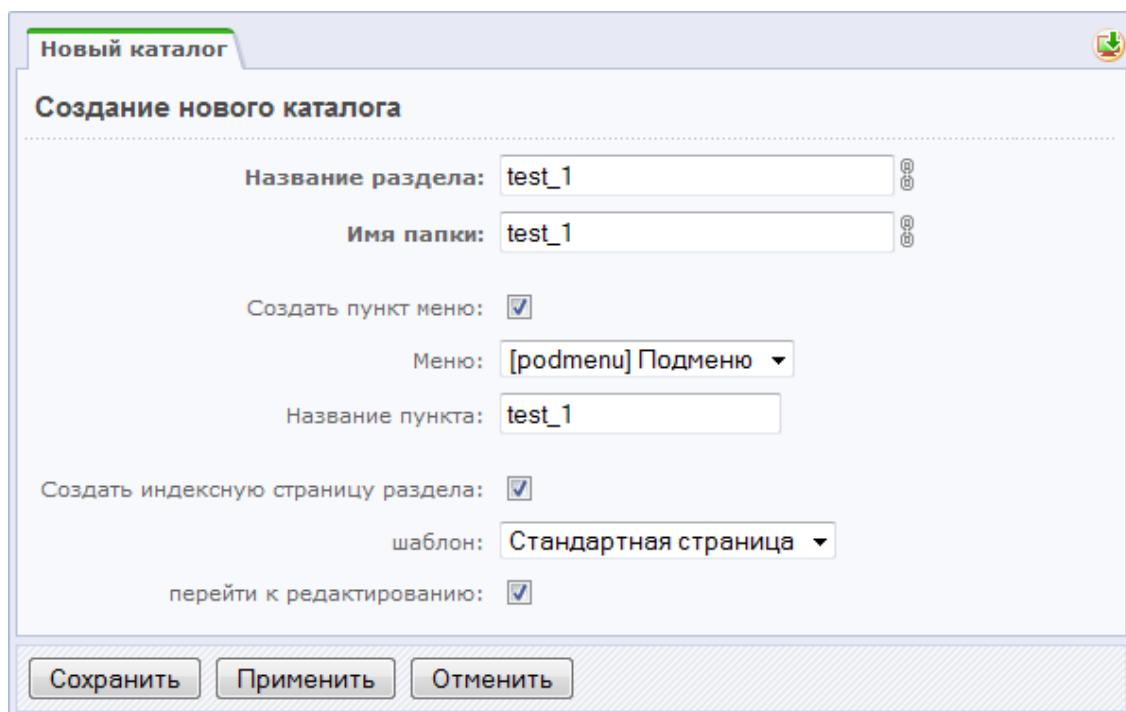


Шаблоны компонента **Меню** поддерживают создание многоуровневого меню с глубиной вложения до 4-х уровней. Покажем это на примере. Выполнять работу на данном этапе удобнее в административной части.

**Задача:** создать четырехуровневое меню.

**Решение.** Решение осуществим на примере уже созданного выше меню с шаблоном **tree**.

- Перейдите в административный раздел на страницу **Управление структурой** (*Структура сайта > Файлы и папки*)
- В разделе */test\_menu* с помощью кнопки **Новая папка** контекстной панели создайте новый каталог со следующими параметрами:
  - Имя папки** - test\_1;
  - Название раздела** - test\_1;
  - Тип меню** - Подменю;
  - Название пункта** - test\_1.



**! Примечание:** Для полноты картины можно создать в папках по дополнительной странице (кроме индексной). При создании их не забывайте добавить их в тип меню – **Подменю**.

- Перейдите во вновь созданный раздел и создайте по описанному выше алгоритму папку test\_2.

**! Примечание:** Если в последней по вложенности папке не создать файл меню, то в публичной части сайта она будет отображаться как страница, а не папка.



- При необходимости добавления новых пунктов или редактирования существующих перейдите в нужный раздел (например, /test\_menu) и на контекстной панели нажмите кнопку **Добавить меню**. В открывшейся форме перейдите к упрощенному режиму редактирования и выберите тип меню равный **Подменю**, после чего отредактируйте нужные пункты меню.

Название	Ссылка	Сортировка	Удалить
Новая страница 1	/test_menu/novaya_stranitsa_1.php	10	<input type="checkbox"/>
Новая страница 2	/test_menu/novaya_stranitsa_2.php	20	<input type="checkbox"/>
test_1	/test_menu/test_1/	30	<input type="checkbox"/>
		40	<input type="checkbox"/>
		50	<input type="checkbox"/>
		60	<input type="checkbox"/>

Сохранить    Применить    Отменить

**⚠ Примечание:** При удалении страниц и разделов из административной части потребуется последующее редактирование пунктов меню вручную.

**Результат:** результатом работы будет созданное древовидное меню в четыре уровня:

**⚠ Примечание:** Система по умолчанию позволяет создать только четыре уровня вложения. Практика показывает, что данной глубины вложения хватает на большинство типовых проектов.



В крайне редких случаях требуется реализовать меню с уровнем вложенности более 4. Это достигается корректировкой файла стилей соответствующего шаблона компонента.

- Откройте для редактирования файл **css** используемого шаблона компонента.

В конце описания стилей есть несколько групп, в названия которых включено `*Items text`. В этих стилях нужно добавить строки. Рассмотрим, как это сделать на примере группы стилей `/*Items text color & size */` файла **css** шаблона **Горизонтальное многоуровневое выпадающее меню**.

В исходном варианте эта группа имеет следующее содержание:

```
#horizontal-multilevel-menu li a,  
#horizontal-multilevel-menu li:hover li a,  
#horizontal-multilevel-menu li.jshover li a,  
#horizontal-multilevel-menu li:hover li:hover li a,  
#horizontal-multilevel-menu li.jshover li.jshover li a,  
#horizontal-multilevel-menu li:hover li:hover li:hover li a,  
#horizontal-multilevel-menu li.jshover li.jshover li.jshover li a,  
#horizontal-multilevel-menu li:hover li:hover li:hover li:hover li a,  
#horizontal-multilevel-menu li.jshover li.jshover li.jshover li.jshover li a
```

Для добавления еще одного уровня необходимо добавить еще две строки:

```
#horizontal-multilevel-menu li:hover li:hover li:hover li:hover li a,  
#horizontal-multilevel-menu li.jshover li.jshover li.jshover li.jshover li.jshover li a
```

Дополнив их соответственно `:hover li` - в одной строке и `.jshover li` - в другой строке.

- Дополните аналогичным способом остальные группы стилей, имеющих в своем названии `*Items text`.
- Сохраните внесенные изменения.

Теперь вы можете использовать глубину вложений равную 5.

## Выпадающее меню

Чтобы сделать многоуровневое выпадающее меню нужно использовать шаблон многоуровнего выпадающего меню, вертикального или горизонтально, в зависимости от



дизайна проекта. Настройка компонента и основные правила создания этого меню абсолютно аналогичны древовидному меню. Приведем их в обобщенном виде:

- Необходимо иметь два типа меню. Одно (первичное) будет применено как основное меню в разделе, другое (вторичное) – как источник формирования собственно выпадающего меню.
- В каждом разделе должно быть обязательно создано вторичное меню с указанием пунктов вторичного меню.
- При указании путей до разделов, которые должны быть развернуты в выпадающем меню, необходимо указывать путь до папки подраздела, а не до индексного файла подраздела.
- Выпадающее меню может быть построено не только на базе статических разделов и страниц, но и на основе инфоблоков.
- Система допускает только четыре уровня вложения.

The screenshot shows the Bitrix Content Manager interface. At the top, there's a navigation bar with tabs: Контент (Content), Магазин (Shop), Общение (Communication), and Социальная сеть (Social Network). Below this, a sidebar lists various content types: Статьи (Articles), Новости (News), Видео и аудио (Video and audio), Галерея (Gallery), Галереи пользователей (User galleries), Доска объявлений (For sale board), Каталог ресурсов (Resource catalog), and Часто задаваемые вопросы (FAQ). To the right of the sidebar, a dropdown menu is open under the 'Новости' (News) item. The menu items are: Все новости (All news) (which is highlighted with a cursor icon), Выставки (Exhibitions), Доставка и поиск (Delivery and search), Книги и авторы (Books and authors), and Конкурсы (Competitions).

## Построение меню из информационных блоков

Построение меню из динамических элементов - информационных блоков - позволяет снять с контент-менеджера часть нагрузки по поддержке сайта. Не нужно будет выполнять работы по актуализации меню в связи с появившимися новыми разделами и страницами. Для решения этой задачи необходимо использовать компонент **Пункты меню (bitrix:menu.sections)**.

Перед созданием такого меню у вас должен быть создан необходимый тип инфоблока и сам инфоблок. Желательно создать пару разделов и элементов для наглядности.

- Выполните команду **Редактировать параметры компонента** из меню компонента **Меню**.



- В разделе **Дополнительные настройки** установите флажок в поле **Подключать файлы с именами вида .тип\_меню.menu\_ext.php**.
- Сохраните внесенные изменения.
- Перейдите в административную часть сайта.
- Перейдите в раздел, для которого планируется создавать меню с помощью инфоблоков.
- Создайте пустой файл под именем `.podmenu.menu_ext.php` в директории, в меню которой должны подключаться пункты инфоблока (например, `/test_menu`).

**⚠ Примечание:** Первая часть имени файла должна совпадать с названием меню, для которого применяется данный компонент. То есть для верхнего меню первая часть должна называться `.top`, для любого другого меню первая часть должна называться `.luboءe_drugoe`.

- Откройте файл для редактирования в визуальном редакторе.
- Добавьте в тело файла компонент **Пункты меню (bitrix:menu.sections)**.

Настройте параметры компонента:

- Выберите тип информационного блока и сам информационный блок (например, **Каталог товаров**).
- Установите глубину вложений любую, более 1.
- Сохраните внесенные изменения. Форма создания файла закроется, файл появится в общем списке файлов папки.
- Откройте его вновь для редактирования, но уже в режиме **Редактировать как PHP**.
- Допишите в файле код проверки включения кода из ядра:

```
<?
if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
global $APPLICATION;
$aMenuLinksExt = $APPLICATION->IncludeComponent(...
```

- После вызова компонента допишите код подключения к меню:

```
$aMenuLinks = array_merge($aMenuLinks, $aMenuLinksExt);
```

Конечный код указанного файла должен быть примерно таким:

```
<?
if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
global $APPLICATION;
$aMenuLinksExt = $APPLICATION->IncludeComponent(
```



```
"bitrix:menu.sections",
",
Array(
    "ID" => $_REQUEST["ID"],
    "IBLOCK_TYPE" => "books",
    "IBLOCK_ID" => "5",
    "SECTION_URL" => "/catalog/phone/section.php?",
    "DEPTH_LEVEL" => "1",
    "CACHE_TYPE" => "A",
    "CACHE_TIME" => "3600"
)
);
$aMenuLinks = array_merge($aMenuLinks, $aMenuLinksExt);
?>
```

- Сохраните внесенные изменения.

**Результат:** Перейдите в публичную часть в раздел, для которого создавался файл **.тип\_меню.menu\_ext.php**. Вы увидите созданное новое меню:

The screenshot shows a website navigation bar with tabs: Главная, test\_menu (highlighted in orange), Профиль, Контент, Магазин, Общение, Социальная сеть. Below the bar, the 'test\_menu' section is displayed. It contains a breadcrumb trail: Главная > test\_menu. The main content area is titled 'test\_menu' and displays the text 'содержимое индексной страницы раздела test\_menu'. To the left of the content, there is a sidebar with a tree-like menu structure. The first level includes links: Главная, test menu (highlighted with a red box), Новая страница 1, Новая страница 2, and test 1 (highlighted with a blue box). The 'test 1' item has a dropdown menu with the following items: Обувь, Бытовая техника, Продукты, Сборка компьютеров, Тара, Сигареты, Инвентарь, Услуги, Мебель, and Бонусные комплекты. The 'test menu' link in the sidebar also has a red box around it.



Красным цветом выделены пункты, созданные в рамках главного меню (массив **\$aMenuLinks**, возвращающий пункты главного меню), синим цветом – полученные из инфоблоков (массив **\$aMenuLinksExt**).

## Два меню рядом

Рассмотрим довольно простой пример размещения двух меню в ряд на примере классического трехколоночного шаблона сайта, который мы использовали до сих пор.

- Перейдите к [редактированию шаблона](#) сайта.
- В режиме редактирования кода в левой колонке сайта создайте таблицу с одной строкой и двумя ячейками.
- Переместите в левую ячейку компонент меню, использовавшийся в качестве верхнего меню, а в правую – компонент меню, использовавшийся в качестве левого меню.
- Сохраните внесенные изменения.
- Перейдите в публичную часть сайта.
- Выполните команду **Редактировать параметры компонента** из меню компонента меню, стоящего в левой ячейке.
- Задайте в качестве шаблона шаблон **default**.
- В поле **Уровень вложенности меню** выставьте значение 1.
- Сохраните внесенные изменения.
- Создайте пункты в меню типа **top** (которое в левой ячейке).
- Выполните команду **Редактировать параметры компонента** из меню компонента меню, стоящего справа.
- Настройте его так, как мы настраивали в примере построения меню из инфоблоков.
- Выполните команду **Редактировать пункты меню** для компонента меню, стоящего в правой ячейке.
- Сохраните внесенные изменения.

После перезагрузки вы увидите простое меню по умолчанию. При переходе во внутренние разделы справа от основного меню будут раскрываться меню активного раздела в зависимости от его настроек (статичное или динамическое, из инфоблоков).



## Графическое меню

Иногда встает задача создания меню с графическими элементами в качестве пунктов меню. Эту задачу можно решить разными способами.

Самый простой – за счет редактирования файла CSS. Этот способ имеет преимущество в том плане, что текст, используемый в компонентах меню для разных пунктов, будет редактироваться обычным способом. Изображение, выводимое css, будет фоновым. Так как данный способ не относится непосредственно к Bitrix Framework, то он рассмотрен не будет.

Рассмотрим вариант представления пункта меню в качестве картинки, реализованный за счет возможностей расширенного режима редактирования меню. В этом случае текст на изображении средствами системы выводиться не будет, надпись названия раздела должна быть сделана непосредственно на картинке пункта меню.

**Задача:** создать меню, где пункты оформляются выводом картинок. Отображение картинок должно быть различным: для активного пункта меню и пассивного пункта меню. Если по каким-то причинам картинки меню не загружены на сайт (либо для пункта меню не заданы параметры), должна отображаться картинка по умолчанию.

- Создайте по паре картинок на каждый из пунктов меню и картинку для пунктов без параметров или без картинки.
- Создайте папку `/images/menu` в рамках структуры сайта.
- Загрузите в эту папку созданные картинки.

**⚠ Примечание:** Для одного из пунктов меню картинки не загрузите, чтобы проверить работу кода.

- Перейдите в административной части сайта в раздел с меню, для которого хотите назначить картинки.
- Откройте меню для редактирования в Расширенном режиме.

### Главная

#### test\_menu

[Статьи](#)[Новости](#)[Видео и аудио](#)[Галерея](#)[Галереи пользователей](#)[Доска объявлений](#)[Каталог ресурсов](#)[Часто задаваемые вопросы](#)

#### Профиль

#### Контент

#### Магазин

#### Общение

#### Социальная сеть

#### Типовые примеры



- В строке **Параметры** каждого пункта меню в колонке **Название** введите **ACT**.
- В строке **Параметры** каждого пункта меню в колонке **Значение** укажите путь до картинки, соответствующей активному состоянию пункта меню, от корня сайта.
- Примените внесенные изменения. После перезагрузки добавится еще одна строка **Параметры**.

**⚠ Примечание:** Увеличить количество доступных сразу дополнительных параметров меню можно в настройках модуля **Управление структурой** с помощью соответствующей опции.

- Заполните для каждого пункта меню поля **Название** (NOACT) и **Значение** (укажите путь до картинки, соответствующей неактивному состоянию пункта меню, от корня сайта).
- Сохраните внесенные изменения.

Мы задали параметры для системы, по которым она будет определять какую картинку ей выводить. Теперь переходим к редактированию шаблона. Для простоты будем использовать шаблон **default**.

- Перейдите в публичную часть сайта.
- Выполните команду **Редактировать параметры компонента** из меню компонента **Меню**.
- Назначьте для компонента шаблон **default**.

- Перейдите к редактированию шаблона компонента, скопируйте его и откройте для редактирования.

Вам нужно заменить код вызова пункта меню со штатного на обращение к параметрам, заданным для данного меню. Делается это в этом участке кода шаблона:

```
<?if($arItem["SELECTED"]):?>
    <li><a href=<?= $arItem["LINK"]?>
        class="selected"><?= $arItem["TEXT"]?></a></li>
    <?else:>
        <li><a href=<?= $arItem["LINK"]?>"><?= $arItem["TEXT"]?></a></li>
    <?endif?>
```

- Замените первую ссылку в коде на строку:

```
<a href=<?= $arItem["LINK"]?>><img border="0" src=<?=(array_key_exists("ACT",  
$arItem["PARAMS"]))  
file_exists($_SERVER["DOCUMENT_ROOT"].$arItem["PARAMS"]["ACT"])  
$arItem["PARAMS"]["ACT"] : "/images/menu/default.png")?>" /></a>
```

- Замените вторую ссылку в коде на строку:

```
<a href=<?=$arItem["LINK"]?>><img border="0" src=<?=(array_key_exists("NOACT",  
$arItem["PARAMS"])  
&&  
file_exists($_SERVER["DOCUMENT_ROOT"].$arItem["PARAMS"]["NOACT"])  
?>  
$arItem["PARAMS"]["NOACT"] : "/images/menu/default.png")?>>/></a>
```

Строки различаются между собой только выбором параметра из расширенных настроек: `ACT` или `NOACT`.

**⚠ Примечание:** В коде использован путь `/images/menu/default.png` для указания картинки, выводимой, если отсутствует картинка для пункта меню или если для него не указаны параметры. Вам нужно использовать собственное имя этой картинки. И другой путь, если вы сохраняете картинки в другой папке.

При переходе в публичную часть мы увидим такое меню:

The screenshot shows a top navigation bar with six items: Главная, test\_menu (highlighted in orange), Профиль, Контент, Магазин, and Общение. Below the navigation bar is a main content area. On the left, there's a vertical list of four items: 'default' (top two are gray ovals, bottom two are gray rounded rectangles). The third item from the top, 'активный пункт' (Active Item), is highlighted with a blue oval border. To its right is a breadcrumb trail 'Главная > test\_menu' and a large bold heading 'Новая страница 1'. Below the heading is a single line of text 'Новая страница 1'. At the bottom of the page is a footer with the text '© 2013 Test Company'.



## Примеры решения частных задач в меню

**Опыт разработчиков:** Дмитрий Яковенко

С версии 9.0.5 в компоненте *bitrix:menu* введен параметр "Откладывать выполнение шаблона меню", позволяющий добавлять пункты меню в компонентах.

```
$GLOBALS['BX_MENU_CUSTOM']->AddItem('left', array('TEXT' => 'Моб. версия', 'LINK' => $APPLICATION->GetCurPage(false) . '?mobile'))
```

Первый параметр - тип меню. Второй - массив, описывающий пункт меню.

## Подсветка ссылок

Поле **Доп.ссылки для подсветки** в расширенной форме редактирования меню позволяет включать подсветку определенных пунктов меню при переходе на страницу. Это бывает нужно в случаях, когда пользователь не должен забывать, откуда он пришел, либо для обращения внимания на определенную страницу.

При посещении страницы, указанной в поле **Доп.ссылки для подсветки**, будет происходить подсветка настраиваемого пункта меню. Путь к страницам задается относительно корня сайта.

## Как сделать, чтобы отдельный пункт меню открывался в новом окне?

Для этого нужно в [расширенном режиме редактирования](#) меню прописать у нужных пунктов следующие **дополнительные параметры**:

Название: target

Значение: target="\_blank"

В [шаблоне меню](#), после вывода элемента меню, необходимо заменить код:

```
<a href=<?=$arItem["LINK"]?>"><?=$arItem["TEXT"]?></a>
```

на следующий:

```
<a href=<?=$arItem["LINK"]?>">  
<?=$arItem["PARAMS"]["target"]?>><?=$arItem["TEXT"]?></a>
```

## Отображение пункта меню для определенных групп пользователей



В расширенном режиме редактирования меню для желаемого пункта необходимо выбрать **тип условия** равный **Для групп пользователей**, а в самом **условии** отметить группы, которые этот пункт будут видеть.

**Тип условия:** Для групп пользователей

**Условие:** требуемые\_группы\_пользователей

### Отображение пункта меню неавторизованным пользователям

В расширенном режиме редактирования меню необходимо установить следующее условие:

**Тип условия:** выражение PHP

**Условие:** !\$USER->IsAuthorized()

### Отображение пункта меню при нахождении в определенном разделе сайта

Система позволяет отображать пункт меню только при нахождении в указанном разделе сайта или на его страницах. Для этого в расширенном режиме редактирования меню в поле **Тип условия** необходимо выбрать опцию **Для папки и файла**, а в поле **Условие** указать путь.

**⚠ Примечание:** Условие **Для папки и файла** лучше всего применять для статических страниц. На динамических страницах он работать не будет работать, так как производит проверку по части адреса, а в динамических страницах всегда будут присутствовать выбранные значения. Для динамических URL удобнее применять условие **Параметр в URL**.

### Осуществление вывода пункта меню, ссылающегося на определенный товар, связанный с товаром, открытым на данной странице

Для этого можно использовать тип условия **Параметр в URL**. Пункт будет отображен на страницах с определенным параметром в URL. Параметр работает с URL, в которых есть символ ?. То есть, с динамическими страницами.

Страницы, созданные на базе инфоблоков, имеют URL вида: [http://сайт/раздел/index.php?SECTION\\_ID=\\*\\*\\*](http://сайт/раздел/index.php?SECTION_ID=***). Предположим, что на странице с **SECTION\_ID=123** должен быть отражен пункт меню, ведущий на страницу **SECTION\_ID=456**.

Создадим пункт меню, ведущий на страницу [http://сайт/раздел/index.php?SECTION\\_ID=456](http://сайт/раздел/index.php?SECTION_ID=456). В поле **Тип условия** выберем **Параметр в URL**, в первом поле **Условие** укажем **SECTION\_ID**, а во втором **123**.



## Всплывающие подсказки для пунктов меню

Для этого необходимо в расширенном режиме редактирования меню добавить **дополнительный параметр A\_TITLE**, и записать в него содержание всплывающей подсказки.

**Название:** A\_TITLE

**Значение:** текст\_всплывающей\_подсказки

В шаблоне меню:

```
<?if($arItem["SELECTED"]):?>
    </i><a href=<?=$arItem["LINK"]?>" class="selected"><?=$arItem["TEXT"]?></a></li>
<?else:&?>
    <li><a href=<?=$arItem["LINK"]?>"><?=$arItem["TEXT"]?></a></li>
<?endif?>
```

нужно заменить первую ссылку в коде на строку:

```
<a href=<?=$arItem["LINK"]?>" class="selected"
title=<?=$arItem["PARAMS"]["A_TITLE"]?>"><?=$arItem["TEXT"]?></a>
```

а вторую на:

```
<a href=<?=$arItem["LINK"]?>" title=<?=$arItem["PARAMS"]["A_TITLE"]?>"><?=$arItem["TEXT"]?></a>
```

## Как поставить картинки рядом с пунктами меню?

Для этого необходимо добавить в меню **дополнительный параметр** (редактирование меню в расширенном режиме), например **IMG**, и записать в него адрес изображения, которое хотите вывести рядом с данным пунктом.

**Название:** IMG

**Значение:** путь\_к\_картинке

В шаблон меню, после вывода элемента меню, необходимо после строки (в зависимости от шаблона):

```
<a href=<?=$arItem["LINK"]?>">
```

добавить следующее:



```
" border="0" />
```

### Разные картинки-пункты меню для разных языков

Меню состоит из пунктов-картинок, заданных CSS классами. Сайт двухязычный. Сама структура меню уже разделена и подключена, необходимо разделить оформление.

#### Решение:

Укажите в шаблоне меню следующее:

```
<body class="lang-<?=LANG?>">
```

в CSS:

```
.menu li.item1 {  
background-image:url(title-en.gif);  
}  
.lang-ru .menu li.item1 {  
background-image:url(title-ru.gif);  
}
```

### Отображение собственного, не такого как в шаблоне, меню для определенных разделов сайта

Организовать такое меню можно с помощью смены шаблона сайта ([Настройки > Настройки продукта > Сайты > Список сайтов](#)), редактирование сайта, секция Шаблон, условие **для папки или файла**). Причем, если шаблон не сложный, то можно прямо в коде шаблона поставить проверку и в одном случае выводить одно меню в другом - другое:

```
if($APPLICATION->GetCurPage() == "/index.php") {  
    вывод меню для главной страницы  
}  
else {  
    вывод второго меню  
}
```



## Как сделать меню tree в постоянно открытом состоянии?

### Решение:

Нужно взять стандартный шаблон меню **tree**, скопировать его в свой шаблон. После чего в файле **template.php** необходимо 14 строчку:

```
<li class="close">
```

заменить на:

```
</li>
```

При этом развернутые пункты будут прятаться по щелчку на изображении.

## Как сделать так, чтобы при открытии страницы через древовидное меню, само меню не сворачивалось, а показывало, на какой странице ты находишься?

### Решение:

Нужно взять стандартный шаблон меню **tree**, скопировать его в свой шаблон. После чего в файле **template.php** необходимо 14 строчку:

```
<li class="close">
```

заменить на:

```
<li <?if (!$arItem["SELECTED"]):?>class="close"<?endif?>>
```

**⚠ Примечание:** Предложенное решение актуально лишь до 2-го уровня вложенности меню.

Следующий код, который необходимо вставить в файл шаблона меню, позволяет меню не сворачиваться, при уровне вложенности больше 2:

```
...
<if (!empty($arResult)):?>

<?
//анализ открытых узлов дерева
$lastLevel = 0;
$selected = false;

foreach(array_reverse($arResult) as $arItem){
```



```
if ($arItem["SELECTED"]) {  
    $lastLevel = $arItem["DEPTH_LEVEL"];  
    $selected = true;  
}  
  
if ($selected and $arItem["DEPTH_LEVEL"] < $lastLevel){  
    $arResult[ $arItem["ITEM_INDEX"] ]["SELECTED"] = true;  
    $lastLevel--;  
}  
}  
?  
  
<div class="menu-sitemap-tree">  
<ul>  
<?=$previousLevel = 0;foreach($arResult as $arItem):?>  
...  

```

### Выпадающее меню с неактивными пунктами первого уровня

Необходимо, что бы нажимая на пункт главного меню, имеющего выпадающее меню второго уровня, не осуществлялся переход в него (чтобы этот пункт не являлся ссылкой), а перейти можно было бы только в любой из пунктов второго уровня выпадающего из него.

В случае если пункт главного меню не имеет меню второго уровня, переход осуществлялся бы непосредственно в него.

Для такой реализации необходимо в коде шаблона вывода меню типа:

```
<?if ($arItem["DEPTH_LEVEL"] == 1):?>  
    <a href=<?= $arItem["LINK"]?>" class="<if ($arItem["SELECTED"]):?>root-item-  
selected<else:>root-item<endif?>"><?=$arItem["TEXT"]?></a>
```

убрать ссылки.

В результате должно получиться следующее:

```
<?if ($arItem["DEPTH_LEVEL"] == 1):?>  
    <li><?=$arItem["TEXT"]?></li>
```

### Скрытие бокового меню по свойству страницы



Задача такая. В шаблоне сайта прописан вывод бокового меню. Нужно, чтобы оно не показывалось только на тех страницах, у которых прописано какое-нибудь свойство, отменяющее показ бокового меню.

### Решение:

Если меню расположено в header'e, то функцию GetProperty использовать нельзя, потому что свойства страницы задаются после подключения header'a. Поэтому меню можно вывести "отложенно" следующим способом:

- в header'e, где надо показать меню добавить следующий код:

```
$APPLICATION->ShowProperty('menu');
```

- в свойствах страницы, если надо запретить меню:

```
$APPLICATION->SetPageProperty('hide_menu', 'Y');
```

- в footer'e:

```
if( 'Y' != $APPLICATION->GetPageProperty('hide_menu') ){
    ob_start();
    echo '<b>проверка отложенного меню!</b>';
    // ....здесь происходит вывод меню компонентом или другим способом.... //
    $APPLICATION->SetPageProperty('menu', ob_get_clean());
}
```

Само меню "выводится" в футере, если свойство hide\_menu не установлено в значение Y. Но оно не выводится на самом деле в футере, а отправляется в свойство menu, которое "выше" по коду можно показать "отложенно" через ShowProperty. Если меню запрещено, то в значении свойства menu будет пусто и ничего не покажется в шаблоне. Если не запрещено, то для этого примера - там, где прописано \$APPLICATION->ShowProperty('menu'), будет выведена фраза проверка отложенного меню!.

### Меню и кеш

**Цитата:** Забился весь диск под завязку. Стал разбираться, и оказалось, что папка bitrix/managed\_cache/MYSQL/menu весит 16Гб! На сайте присутствуют 2 меню горизонтальное - навигация по страницам сайта и вертикальное - по разделам каталога.

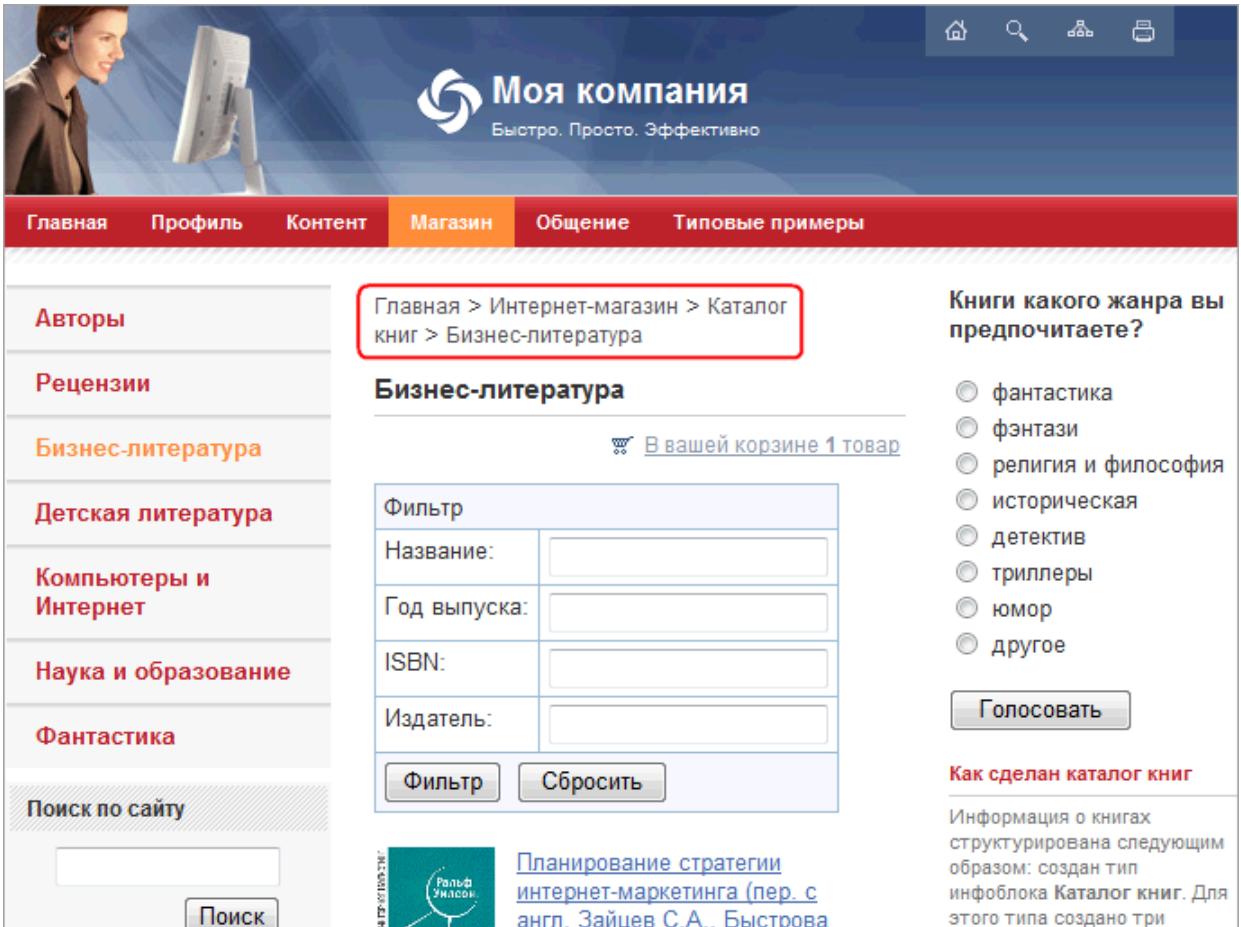
Кеш меню зависит от адреса страницы. Если страниц много, то кеш может получаться большим. В этом случае может оказаться более эффективным отключить кеш в компоненте меню.

## Цепочка навигации

**Цепочка навигации** - последовательный список ссылок на разделы и страницы сайта и показывает уровень "погружения" текущей страницы в структуру сайта.

Значения, подставляемые в навигационную цепочку, могут быть заданы как для каждого раздела, так и для отдельного документа.

Цепочка навигации помогает посетителю легко ориентироваться на сайте: быстро вернуться на главную страницу или подняться на один (или более) уровень вверх, т.е. перейти из подраздела в раздел.



The screenshot shows a website interface for a company named 'Моя компания' (My Company). The top navigation bar includes links for Главная (Home), Профиль (Profile), Контент (Content), Магазин (Store), Общение (Communication), and Типовые примеры (Typical Examples). A sidebar on the left lists categories: Авторы (Authors), Рецензии (Reviews), Бизнес-литература (Business Literature), Детская литература (Children's Literature), Компьютеры и Интернет (Computers and Internet), Наука и образование (Science and Education), and Фантастика (Fantasy). A search bar at the bottom left has a 'Поиск' (Search) button. The main content area displays a breadcrumb trail: Главная > Интернет-магазин > Каталог книг > Бизнес-литература. Below this, there is a section titled 'Бизнес-литература' with a link to 'В вашей корзине 1 товар' (1 item in your cart). To the right, there is a sidebar for book preferences with radio buttons for various genres: фантастика (fantasy), фэнтези (fantasy), религия и философия (religion and philosophy), историческая (historical), детектив (detective), триллеры (thrillers), юмор (humor), and другое (other). A 'Голосовать' (Vote) button is also present. At the bottom right, there is a section titled 'Как сделан каталог книг' (How the catalog of books was made) with a note about the structure of the information.

В этом разделе приводится описание структуры шаблона и данных, на основе которых выполняется построение и отображение цепочки навигации, а также приемы управления показом цепочки навигации на страницах сайта.



## Управление пунктами навигационной цепочки

### **Управление через визуальный интерфейс системы**

По умолчанию в системе используется механизм управления названиями пунктов навигационной цепочки через свойства разделов.

Заголовок раздела сайта задается в служебном файле **.section.php**, который находится в соответствующем разделе. В данном файле могут быть использованы следующие переменные:

- **\$sSectionName** - заголовок раздела;
- **\$sChainTemplate** - абсолютный путь к шаблону навигационной цепочки (данная переменная используется крайне редко)

Пример файла **.section.php**:

```
<?
$ sSectionName           =           "О компании";
$ sChainTemplate        =      $_ SERVER["DOCUMENT_ROOT"]. "/ru/about/chain_template.php";
?>
```

Визуально же название ссылки на раздел сайта в навигационной цепочке указывается с помощью поля **Заголовок** в форме настройки свойств раздела.

Переход к форме настройки свойств раздела можно осуществить:

- из публичного раздела (читай курс [Контент-менеджер](#));
- из административного раздела (читай курс [Администратор. Базовый](#)).

Чтобы изменить название ссылки на раздел в цепочке навигации, нужно отредактировать **Заголовок** папки. Название ссылки будет изменено сразу же после сохранения новых свойств раздела.

Главная > Интернет-магазин > Каталог книг > Бизнес-литература

**⚠ Примечание:** Для того чтобы ссылка на какой-либо раздел не выводилась в навигационной цепочке сайта, нужно удалить название раздела из поля **Заголовок** и сохранить внесенные изменения.



## Управление через код страницы

С помощью php функции `AddChainItem()` в цепочку навигации могут быть добавлены дополнительные пункты. В качестве значений пунктов навигационной цепочки могут использоваться как статические, так и динамические величины:

```
<?
//---Первым параметром функции AddChainItem() задается название,
//---которое будет показано в навигационной цепочке, вторым параметром задается
//---ссылка для перехода.
//---Значения параметров могут быть как статическими, так и динамическими.
//---в приведенном примере название раздела задано статической величиной, а
//---ссылка формируется динамически.

$APPLICATION->AddChainItem("Детальная информация о товаре",
"catalog.php?BID=".$arIBlock["ID"]."&ID=".$arSection["ID"]);

//---В приведенном ниже примере обе величины формируются динамически.
//---В качестве названия подставляется текущее значение раздела каталога.

$APPLICATION->AddChainItem($arSection["NAME"],
"catalog.php?BID=".$arIBlock["ID"]."&ID=".$arSection["ID"]);

?>
```

Для того чтобы выводить в навигационную цепочку название текущей страницы, необходимо вставить вызов функции `AddChainItem()` в файле **footer.php**, т.е. после вывода содержимого рабочей области.

```
<?$APPLICATION->AddChainItem($APPLICATION->GetTitle());?>
```

Некоторые пункты навигационной цепочки могут не содержать ссылки на какой-либо раздел или документ сайта, т.е. могут быть представлены в виде обычного текста (например, название текущей страницы).

Главная > Контент > Новости интернет-магазина > Выставки

Такие пункты создаются путем добавления в шаблон показа навигационной цепочки следующего кода:



```

if($arResult[$index]["LINK"] <> "")
$strReturn .= '<li><a href="'. $arResult[$index]["LINK"].'" title="'.$title.'">' . $title . '</a></li>';
else
$strReturn .= '<li>' . $title . '</li>';

```

Отдельные компоненты могут также добавлять в навигационную цепочку заголовок текущей страницы сайта или, например, заголовок текущей новости или товара каталога. Так, например, комплексный компонент **Новости** (**bitrix:news**) последовательно добавляет в навигационную цепочку названия каталогов и групп новостей по мере погружения вглубь по уровням, если это установлено в его настройках.

The screenshot shows the Bitrix CMS interface with a red navigation bar at the top containing links: Главная, Профиль, Контент (Content), Магазин, Общение, Типовые примеры. Below the bar is a sidebar with categories: Все новости (All news), Выставки (Exhibitions), Доставка и поиск (Delivery and search), Книги и авторы (Books and authors), and Конкурсы (Competitions). A search input field and a 'Поиск' (Search) button are also in the sidebar. The main content area displays a breadcrumb navigation path: Главная > Контент > Новости интернет-магазина > Книги и авторы. The 'Книги и авторы' link is highlighted with a red box. Below the path is the heading 'Новости магазина'. There are three news items listed: 1. 07.03.2007 [Российские писатели вошли в список молодых литературных талантов США](#) (RSS icon). Description: Авторитетный в англоязычном литературном мире британский журнал Granta составил новый перечень наиболее многообещающих молодых писателей США. В него вошли и писатели русского происхождения. 2. 05.03.2007 [Самые популярные авторы](#). Description: Известен список популярных авторов за последний год. Пятерка лучших: Б. Акунин, Л.П. Франкл, А. Кэрр, С. Минаев, А. Гавальда. 3. 28.12.2006 [Названы лучшие книги и издательства 2006 года](#). Description: В Российской государственной библиотеке (РГБ) состоялась

Аналогичным образом в навигационную цепочку добавляются названия форумов и тем форумов.

В этом случае значение пункта навигационной цепочки для данной страницы определяется непосредственно в документе. Для этого используется функция `AddChainItem()`.

### Управление показом цепочки

Показ навигационной цепочки выполняется с помощью специального кода в шаблоне сайта, использующего [технологию отложенных функций](#):

```

<?
$APPLICATION->ShowNavChain();

```



?>

Функция вызова навигационной цепочки может использоваться не только в шаблоне сайта, но и в коде страницы или отдельных компонентов, представленных на странице. Например, вывод навигационной цепочки используется в компоненте поиска:

## Поиск

MVC компонент компоненты 2.0 конкурс рецензия Подсказки Советы

Каталог

Результаты поиска 1 - 8 из 8  
Начало | Пред. | 1 | След. | Конец

["Куписам" - интернет-супермаркет](#)

Иллюстрированный каталог всевозможных товаров и услуг - предложения от российских интернет-магазинов. Поиск по каталогу. Иллюстрированный каталог всевозможных товаров и услуг - предложения от российских интернет-магазинов. Поиск по каталогу.

Изменен: 10.07.2008  
Путь: [Главная / Контент / Каталог ресурсов](#)

---

[Интернет-магазин](#)

Модуль Интернет-магазин позволяет организовать продажу любых товаров на сайте. На нашем сайте приведен пример организации продажи книг. Раздел Магазин (/e-store/) состоит из двух подразделов: Каталог книг (/e-store/books/) и раздел Аффилиатов (/e-store/affiliates/). Информация о книгах структурирована следующим образом: создан тип инфоблока Каталог книг . Для этого типа создано три информационных блока: Авторы; Книги; Рецензии. Из...

Изменен: 30.07.2008  
Путь: [Главная / Интернет-магазин](#)

**⚠ Примечание:** В демо-версии продукта шаблон цепочки навигации, используемой в компоненте поиска, хранится в каталоге шаблона компонента поиска в файле `chain_template.php`.

Показ навигационной цепочки может быть отключен на определенных страницах или в определенном разделе сайта. Управление отображением навигационной цепочки также осуществляется с помощью свойств страницы (раздела). Для этого необходимо:

- На странице настроек модуля **Управление структурой**, секция **Настройки для сайтов**, создать свойство для страниц **Не показывать навигационную цепочку** с кодом `not_show_nav_chain`. Данное свойство является предустановленным в системе;



Типы свойств:	Тип	Название
	description	Описание страницы
	keywords	Ключевые слова
	title	Заголовок окна браузера
	keywords_inner	Продвигаемые слова
	not_show_nav_chain	Не показывать навигационную

- если навигационная цепочка не должна отображаться на определенной странице или страницах какого-либо раздела, то для этой страницы или раздела нужно установить значение данного свойства равным `Y`.

**Свойства раздела**

**Свойства раздела /e-store/books** [Редактировать свойства папки в Панели управления](#) 

**Заголовок раздела** ? [?](#)

Заголовок: Каталог книг

**Свойства раздела** ? [?](#)

Описание страницы: 1С-Битрикс: Управление сайтом 

Ключевые слова: 1С-Битрикс, CMS, PHP, bitrix, система управления 

Заголовок окна браузера:

Продвигаемые слова:

Не показывать навигационную цепочку:  `Y` 

ROBOTS: index, follow 

**Сохранить** **Отменить** 

Кроме того, значение свойства **Не показывать навигационную цепочку** может быть задано непосредственно в коде страницы с помощью функции `SetPageProperty()`:

```
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetPageProperty("description", "«1С-Битрикс: Управление сайтом»
- система разработки и управление web-проектами");
```



```
$APPLICATION->SetPageProperty("keywords", "Сайт, управление сайтом, система  
управления");  
$APPLICATION->SetPageProperty("NOT_SHOW_NAV_CHAIN", "Y");  
$APPLICATION->SetTitle("Пробная версия продукта «1С-Битрикс: Управление  
сайтом»");  
?>
```

## Управление шаблоном навигационной цепочки

Цепочка навигации подключается в шаблоне дизайна сайта с помощью компонента **Навигационная цепочка (bitrix:breadcrumb)**. Для него может быть создано любое количество шаблонов, т.е. внешних видов. Все они хранятся в папке компонента `/bitrix/components/bitrix/breadcrumb/templates/<название шаблона>/`. Все созданные шаблоны будут отображаться в настройках компонента. Таким образом для каждого шаблона сайта может быть установлен свой шаблон оформления компонента цепочки навигации. Структура шаблона показа навигационной цепочки аналогична структуре шаблона показа меню.

Управление навигационной цепочкой и ее шаблоном оформления осуществляется также, как при работе с другими компонентами 2.0. В режиме правки сайта с помощью кнопки управления вы можете быстро перейти к форме изменения параметров компонента либо скопировать шаблон компонента, а затем отредактировать его.

Алгоритм построения цепочки навигации и формирования ее внешнего вида:

1. Сбор пунктов навигационной цепочки ведется начиная от корня сайта и заканчивая текущим разделом. Для каждого очередного раздела подключается файл `.section.php`. Если в данном файле будет инициализирована переменная `$sChainTemplate`, то ее значение будет использовано в качестве пути к шаблону навигационной цепочки. В процессе перебора разделов, каждое последующее значение этой переменной перетирает предыдущее, таким образом, чем "глубже" раздел в иерархии разделов сайта, тем "важнее" его переменная `$sChainTemplate`.
2. Если после сбора пунктов навигационной цепочки путь к шаблону не определён, то проверяется существование файла:  
`/bitrix/templates/ID текущего шаблона сайта/chain_template.php`
4. Если такой файл существует, то путь к нему и принимается за путь к шаблону навигационной цепочки, в противном случае используется значение по умолчанию:  
`/bitrix/templates/.default/chain_template.php`

Шаблон навигационной цепочки при ее выводе будет подключаться каждый раз на очередном пункте цепочки. Поэтому основная его задача - обеспечить внешний вид только одного пункта цепочки.

Основными переменными используемыми в шаблоне являются:

- **\$sChainProlog** - HTML код выводимый перед навигационной цепочкой
- **\$sChainBody** - HTML код определяющий внешний вид одного пункта навигационной цепочки
- **\$sChainEpilog** - HTML код выводимый после навигационной цепочки
- **\$strChain** - HTML код всей навигационной цепочки собранный к моменту подключения шаблона

Выше представленные переменные будут хранить в себе HTML-код, который определит внешний вид навигационной цепочки.

Также в шаблоне будут доступны следующие дополнительные переменные:

- **\$TITLE** - заголовок очередного пункта навигационной цепочки
- **\$LINK** - ссылка на очередном пункте навигационной цепочки
- **\$arCHAIN** - копия массива элементов навигационной цепочки
- **\$arCHAIN\_LINK** - ссылка на массив элементов навигационной цепочки
- **\$ITEM\_COUNT** - количество элементов массива навигационной цепочки
- **\$ITEM\_INDEX** - порядковый номер очередного пункта навигационной цепочки

Пример шаблона компонента навигационной цепочки:

```
<?
if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();

if(empty($arResult))
    return "";

$strReturn = '<ul class="breadcrumb-navigation">';
for($index = 0, $itemSize = count($arResult); $index < $itemSize; $index++)
{
    if($index > 0)
        $strReturn .= '<li><span>&ampnbsp&gt;&ampnbsp</span></li>';
    $title = htmlspecialchars($arResult[$index]["TITLE"]);
```



```
if($arResult[$index]["LINK"] <> "")  
    $strReturn .= '<li><a href="'. $arResult[$index]["LINK"].'" title="'.$title.'">' . $title . '</a></li>;  
else  
    $strReturn .= '<li>' . $title . '</li>;  
}  
$strReturn .= '</ul>;  
return $strReturn;  
?>
```

**⚠ Примечание:** При подключении навигационной цепочки с помощью функции **ShowNavChain()** ее шаблон может быть задан дополнительно для отдельного раздела сайта. Для этого непосредственно в файле `.section.php` определяется переменная `$sChainTemplate`, где задается полный путь к шаблону показа навигационной цепочки. Например:

```
$sChainTemplate = "/bitrix/templates/demo/chain_template.php"
```

Шаблон навигационной цепочки может быть также задан при вызове функции **ShowNavChain()** как один из параметров функции.

```
$APPLICATION->ShowNavChain("/bitrix/templates/.default/chain_template_bottom.php")
```

## Примеры работы

### Как добавить свой пункт в цепочку навигации?

С помощью функции [AddChainItem\(\)](#) :

```
<?  
$APPLICATION->AddChainItem("Форум "Отзывы"", "/ru/forum/list.php?FID=3");  
?>
```

### На главной не отображается цепочка навигации

**Цитата:** В шаблоне сайта в цепочке навигации выводится название только инфоблоков. Заголовки страниц (допустим контакты) не выводят, заголовок "главная" тоже не отображается. Подскажите что не так?!

Для главной страницы необходимо в **свойствах страницы** в поле NOT\_SHOW\_NAV\_CHAIN установить значение **N**.

Для остальных страниц необходимо проверить, установлены ли заголовки в свойствах разделов. Именно заголовки разделов берутся для создания пунктов навигационной цепочки.

Главная страница может также не отображаться по причине неправильно установленной опции **Номер пункта, начиная с которого будет построена навигационная цепочка**: 0 (значение по умолчанию) означает, что построение навигационной цепочки начнется от корня сайта. В случае если заполнено поле **Путь, для которого будет построена навигационная цепочка**, то номер пункта считается в указанном пути.

### Повторение пунктов в цепочке навигации

**Цитата:** Не подскажете, как так получается, что в каталоге в цепочке навигации повторяется дважды название каталога:

*Главная / Магазин / Видеонаблюдение / Видеонаблюдение*

*Первая ссылка на каталог выглядит так: /catalog/video/*

*а вторая: /catalog/video/section.php?SECTION\_ID=0*

Первое название берется из свойств директории **video** (файл `.section.php`), а второе - компонентом, расположенным на странице (в данном случае адрес страницы `section.php`).

**⚠ Примечание:** Например, у компонента *Новости* (`bitrix:news`) в его параметрах присутствуют соответствующие опции: *Включать инфоблок в цепочку навигации* и *Включать раздел в цепочку навигации*.

Повторение элементов в цепочке навигации также может быть вызвано наличием нескольких компонентов на странице, которые настроены на добавление своих пунктов в цепочку.

### Как сделать, что бы в навигационной цепочке присутствовал только физический раздел?

Используется комплексный компонент "Новости" для вывода раздела из инфоблока. В навигационной цепочке помимо физического раздела, появляется некликабельное название раздела самого инфоблока. В настройках ни раздел, ни инфоблок невключен в НЦ.



**Решение:** Добавьте в шаблон строку:

```
"ADD_SECTIONS_CHAIN" => $arParams["ADD_SECTIONS_CHAIN"],
```

## Рекламные области

На сайте может быть представлено несколько различных видов рекламы. Это может быть как стандартная баннерная реклама, так и текстовые рекламные области. Реклама может быть постоянной или показываться с определенной вероятностью, задаваемой администратором. Показ рекламы может быть регламентирован по определенным разделам сайта и т.д. Создание и управление рекламой осуществляется средствами модуля **Рекламы**.

Показ рекламы выполняется в специально выделенных областях шаблона дизайна сайта – рекламных областях.

The screenshot displays a website layout with several advertising sections. On the left, there's a sidebar for 'Подписка на рассылку' (Newsletter subscription) with checkboxes for 'Новости компании' and 'Новинки каталога'. Below it is a section titled 'Реклама' (Advertising) containing a banner for '1C-Битрикс: управление сайтом' (1C-Bitrix: Website Management). To the right, there's a main content area with a heading '06.10.2009 Самые популярные авторы' (Most popular authors) and a list of authors. At the bottom, there's a footer banner for '1C-Битрикс: Управление сайтом' (1C-Bitrix: Website Management) with the text 'Работает на «1С-Битрикс: Управление сайтом»' (Works on '1C-Bitrix: Website Management').

Подключение баннера в рекламной области может выполняться с помощью компонента **Баннер (bitrix:advertising.banner)**. Компонент выводит баннер заданного типа.

**⚠ Примечание:** *Данный компонент не учитывает таргетинг по ключевым словам. Если необходимо его учесть, то следует использовать функцию \$APPLICATION->ShowBanner().*

Код подключения баннера в рекламной области с помощью PHP функции ShowBanner() имеет вид:



```
<?
//---Пример размещения рекламной области в левой части сайта.
//---Аналогично производится выбор любого другого типа:
//---TOP, BOTTOM, COUNTER, ... (первый параметр функции)
//---Могут быть использованы как предустановленные, так и задаваемые
//---пользователем
//---В качестве двух дополнительных параметров может указываться
HTML-код обрамляющий рекламную область сверху и снизу.
$APPLICATION->ShowBanner("LEFT", '<div align="center">', '<br></div><br>');
?>
```

Для каждой рекламной области определяется, реклама какого типа будет доступна для показа в данной области. В приведенном выше примере это рекламные баннеры типа LEFT и BOTTOM.

### Типы рекламы

**Тип рекламы** - это условное название группы рекламных баннеров, обладающих некоторым общим признаком. Например:

- место показа – все баннеры должны показываться в определенном месте страницы сайта;
- тематика (например, рекламируется один товар);
- рекламодатель (реклама определенной компании);
- и т.д.

Название типа рекламного блока задается администратором произвольно. Например, для верхней рекламы может быть установлен тип TOP, LEFT, BOTTOM и т.п.



Подписка на рассылку

**06.10.2009 Самые популярные авторы**  
Известен список популярных авторов за последний год. Пятерка лучших: Б. Акунин, Л.П. Франкл, А. Карр, С. Минаев, А. Гавальда.

Новости компании  
 Новинки каталога

Новости 1 - 5 из 10  
Начало | Пред. | 1 2 | След. | Конец

Подписаться

Реклама

**LEFT**

**BOTTOM**

1С-Битрикс: Управление сайтом  
Механизм программных компонентов  
Визуальная публикация динамической информации

Работает на «1С-Битрикс: Управление сайтом»

**⚠ Важно!** Рекомендуется, чтобы баннеры из одной группы имели одинаковый размер. Это позволит избежать деформации страницы при показе рекламы.

Управление типами рекламы выполняется через административный интерфейс модуля **Рекламы** ([Сервисы > Реклама > Типы баннеров](#)):

**Типы баннеров**

Рабочий стол > Сервисы > Реклама > Типы баннеров

Найти: ID Найти Отменить

Добавить тип Настроить Excel

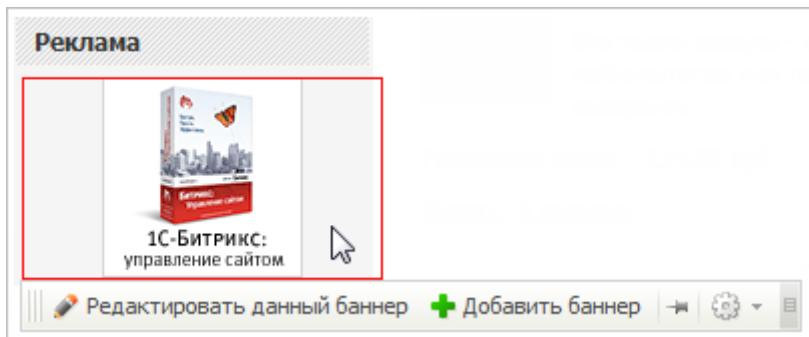
На странице: 20 Типы баннеров 1 – 2 из 2

ID	Изменено	Акт.	Сорт.	Имя	Описание	Баннеров
<a href="#">BOTTOM</a>	10.07.2008 09:45:05	Да	▲	1 Нижний баннер		<a href="#">3</a>
<a href="#">LEFT</a>	10.07.2008 09:45:05	Да	▲	1 Левый баннер		<a href="#">1</a>

Выбрано: 2 Отмечено: 0

Приступить к управлению рекламным баннером или списком баннеров выбранной рекламной области можно непосредственно из публичной части сайта. Для этого

перейдите в режим **правки** и воспользуйтесь одной из кнопок управления рекламной областью:



### Управление показом рекламы с помощью ключевых слов

Управление показом баннеров с помощью ключевых слов является одной из форм таргетинга рекламы на сайте. Особенность данного метода заключается в том, что он позволяет организовать рекламную кампанию, направленную на четко определенную целевую группу пользователей сайта.

С помощью ключевых слов можно:

- организовать показ рекламы, нацеленный на конкретную группу пользователей сайта. Т.е. показывать рекламу на страницах, посещаемых преимущественно этими пользователями, или страницах, тематика которых может заинтересовать данную группу пользователей.
- ограничить показ рекламы на страницах сайта, например, исходя из того, насколько содержание рекламы связано с информацией, представленной на странице.

### Механизм управления

Управление показом рекламы на страницах сайта осуществляется с помощью желательных и обязательных ключевых слов страницы сайта и набора ключевых слов рекламного баннера. В целях управления показом рекламы для страниц используются два вида специальных ключевых слов:

- ключевые слова баннера;
- ключевые слова страниц сайта:
  - **желательные (desired)**: если для страницы сайта заданы желательные ключевые слова, то для показа на данной странице будут доступны все баннеры, в наборе ключевых слов которых содержится хотя бы одно желательное ключевое слово страницы.
  - **обязательные (required)**: если для страницы сайта заданы обязательные ключевые слова, то для показа на данной странице будут доступны все



баннеры, в наборе ключевых слов которых содержаться все обязательные ключевые слова страницы.

The screenshot shows the 'Properties of the page' dialog box. It includes fields for the page title ('Title: Каталог книг'), page description ('Description: 1С-Битрикс: Управление сайтом'), and page keywords ('Keywords: 1С-Битрикс, CMS, PHP, bitrix, система управления контентом'). A red box highlights the 'Desired key words for advertising' and 'Required key words for advertising' sections, which are part of the 'Key words for advertising' group. Below these, there are fields for 'ROBOTS' (set to 'index, follow') and 'NOT\_SHOW\_NAV\_CHAIN' (set to 'Y'). At the bottom are 'Save' and 'Cancel' buttons.

Если система не найдет баннеры удовлетворяющих каким-либо из ключевых слов, то на странице будут показаны баннеры, для которых ключевые слова не заданы. Отбор и показ данных баннеров осуществляется при этом на основе стандартного алгоритма системы (разрешенных/запрещенных страниц контрактов и баннеров, групп пользователей, типов баннеров и других).

Общий порядок действий по организации показа баннеров на нужных страницах такой:

- Определяется, какая реклама будет доступна для показа на тех или иных страницах сайта.
- В соответствии с поставленными задачами для страниц сайта задаются наборы специальных ключевых слов.
- В настройках рекламных баннеров задается необходимый набор ключевых слов.

Ключевые слова баннеров задаются в поле **Ключевые слова** на странице создания/редактирования баннера, закладка **Таргетинг** ([Сервисы > Реклама > Баннеры](#)):



Группы пользователей:  
Ctrl+ (выберите нужные)

Показывать баннер только для выбранных групп  
 Не показывать баннер для выбранных групп

Администраторы [1]  
Зарегистрированные пользователи [3]  
**Пользователи панели управления [4]**  
1С интеграция [5]  
Администраторы интернет-магазина [6]  
Администраторы техподдержки [7]  
Подписчики [8]  
Пользователи имеющие право голосовать за рейтинг [9]  
Пользователи имеющие право голосовать за авторитет [10]  
Администраторы клуба [11]

Ключевые слова:  
Битрикс  
веб-сайт

Ключевые слова должны быть разделены новой строкой.

Управление желательными ключевыми словами страницы выполняется с помощью специального свойства `adv_desired_target_keywords`. Данное свойство является предустановленным, т.е. оно описано в программном коде продукта.

Типы свойств:	Тип	Название
	description	Описание страницы
	keywords	Ключевые слова
	title	Заголовок окна браузера
	keywords_inner	Продвигаемые слова
	not_show_nav_chain	Не показывать навигационную
	<b>adv_desired_target_ke</b>	<b>Ключевые слова для рекламы</b>

Желательные ключевые слова страницы могут быть заданы с помощью функции `SetDesiredKeywords`. Данная функция содержит следующие параметры:

- `keywords` – одно или несколько желательных ключевых слов страницы;
- `TIP_SID` – тип рекламы, управление показом которой будет осуществляться с помощью указанных желательных ключевых слов. Если данный параметр оставить пустым, то ключевые слова будут заданы для всех типов рекламы.

```
CAdvBanner::SetDesiredKeywords (array ("Partners", "Cooperation", "Company", "Contacts"),
"RIGHT");
```



**⚠ Обратите внимание!** По умолчанию, если в коде страницы не установлено каких-либо ключевых слов для рекламы, считается что функция `SetDesiredKeywords` использует в качестве параметра значение свойства страницы `adv_desired_target_keywords`. Если оно не задано, то в качестве параметра функции используется значение свойства `keywords`.

Функция `SetDesiredKeywords` вызывается автоматически в момент сборки страницы, ее не нужно дополнительно вызывать в файле `header.php`, если нет необходимости переопределить ключевые слова для показа рекламы.

Обязательные ключевые слова страницы задаются с помощью предустановленной функции системы `SetRequiredKeywords`. Данная функция содержит следующие параметры:

- `keywords` – одно или несколько обязательных ключевых слов страницы;
- `TYPE_SID` – тип рекламы, управление показом которой осуществляется с помощью указанных желательных ключевых слов. Если данный параметр оставить пустым, то ключевые слова будут заданы для всех типов рекламы.

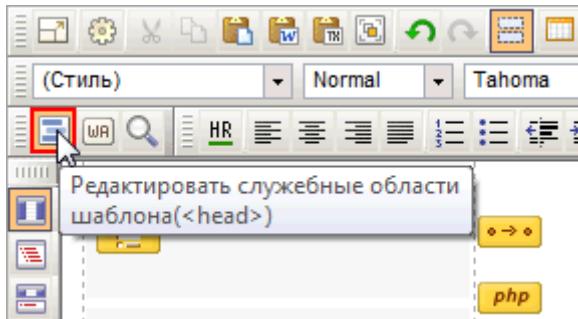
```
CAdvBanner::SetRequiredKeywords (array("Partners", "Cooperation"), "RIGHT");
```

В приведенном примере для показа на странице будет доступен баннер, в наборе ключевых слов которого содержаться все обязательные ключевые слова (`partners` и `cooperation`), заданные для страницы.

## Управление служебными данными шаблона

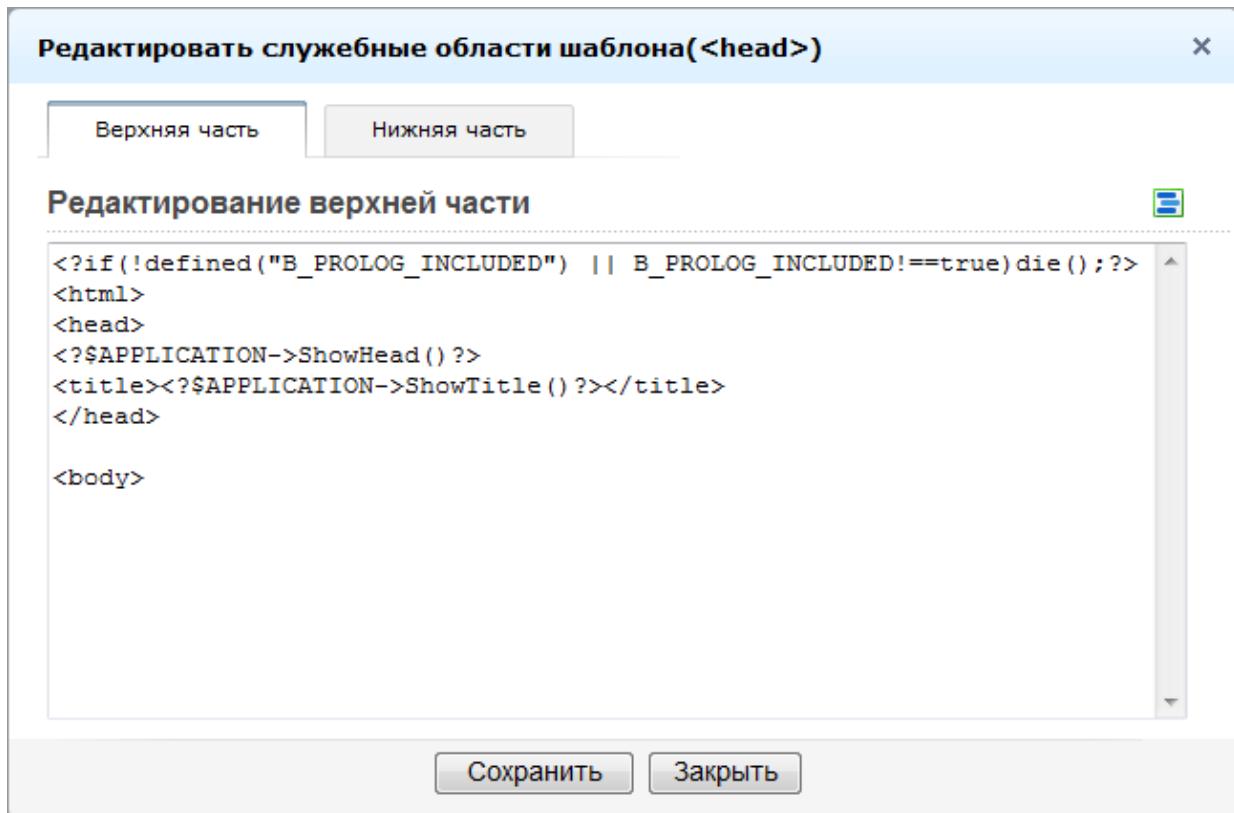
### Редактирование служебных областей

Если для создания (редактирования) шаблона вы используете визуальный редактор, то управление служебными областями можно выполнить в специальной форме. Переход к этой форме осуществляется с помощью кнопки **Редактировать служебные области шаблона**, расположенной в панели редактора.





Форма редактирования областей состоит из двух закладок: **Верхняя часть** и **Нижняя часть**.



На первой закладке выполняется редактирования верхней части шаблона до тега **<body>**. Вы можете задать содержимое области значениями по умолчанию. Для этого нажмите на кнопку и в форму будут вставлен набор стандартных функций, таких как установка кодировки страниц, вывод заголовка и метаданных страницы, подключение файлов стилей и др. Аналогично, на закладке **Нижняя часть** выполняется редактирования нижней части шаблона, содержимое которой также может быть задано значениями по умолчанию.

**⚠ Внимание!** Использование функций *ShowMeta()*, *ShowTitle()*, *ShowCSS()* и т.д. позволяет производить инициализацию отдельных элементов непосредственно из скрипта на странице или из компонента. Например, заголовок страницы может быть добавлен уже после вывода результатов работы скрипта. Таким образом, если ранее требовалось производить инициализацию заголовка страницы до подключения основного дизайна, то сейчас можно производить установку заголовка страницы непосредственно из кода в рабочей области страницы.



## Управление кодировкой страниц

Одной из важных особенностей **Bitrix Framework** является поддержка произвольного количества языков. Система позволяет:

- использовать многоязычный интерфейс в административном разделе;

The screenshot shows the 1C-Bitrix administrative interface. At the top, there are tabs for 'Меню' (Menu), 'Сайт' (Site), and 'Администрирование' (Administration). The 'Настройки' (Settings) tab is selected in the left sidebar. The main content area has three panels: 'Заказы' (Orders) showing statistics for orders created, paid, canceled, and allowed for delivery; 'Информация о системе' (System Information) displaying system status, last update, performance metrics, and user count; and 'Статистика' (Statistics) showing a line chart of data over time. The interface supports multiple languages, as indicated by the 'ru' and 'en' buttons at the top.



The screenshot shows the 1C-Bitrix Control Panel interface. On the left, there's a sidebar with various modules like Content, Services, e-Store, Web Analytics, and Settings. The main area has tabs for 'Orders' and 'Statistics'. Under 'Orders', there are tables for statistics by day, week, and month. Under 'Statistics', there's a line chart showing the number of hosts over time from February 2nd to March 13th. The chart shows several peaks, with the highest point reaching nearly 10 hosts around February 5th.

- создавать произвольное количество сайтов (в зависимости от лицензии) на различных языках в рамках одной системы.

**⚠ Примечание:** Количество используемых в системе языков не зависит от количества сайтов.

### Использование кодировок

Для корректного отображения национальных символов используются соответствующие кодировки. При показе страницы браузер распознает используемую кодировку и на ее основе выполняет отображение символов.

Ниже приводится список таблиц кодов, используемых для отображения символов русского, английского и немецкого языков:

Язык	Кодировка
Russian (ru)	windows-1251, koi8-r, iso-8859-5

English (en) windows-1252, iso-8859-1, latin1



German (de) windows-1252, iso-8859-1, latin1

Полный список кодировок, используемых для различных языков, приводится в [документации продукта](#).

**⚠ Примечание:** Начиная с версии 7.0, в продукте (для баз данных MySQL и Oracle) поддерживается универсальная кодировка **UTF-8**. С ее помощью содержимое сайта может быть одновременно представлено на разных языках. Если **UTF-8** не используется, но в системе необходимо сочетание разных языков, то для каждого языка нужно определить кодовую таблицу, с использованием которой будут отображаться текстовые сообщения.

**⚠ Внимание!** Кодировка страниц и кодировка таблиц базы данных должны совпадать.

Настройка кодировки выполняется отдельно для административного и публичного раздела:

- Настройка кодировки, используемой в публичном разделе, выполняется для каждого сайта ([Настройки > Настройки продукта > Сайты > Список сайтов](#)):

Кодировка задается исходя из языка, используемого на сайте. Также при настройке параметров языка можно задать формат времени и даты, что позволит правильно выводить эти данные в публичном разделе (например, при показе новостей, товаров каталога и т.д.).

Параметры:

*Язык:	[ru] Russian ▾
*Формат даты: (например: DD.MM.YYYY)	DD.MM.YYYY
*Формат даты и времени: (например: DD.MM.YYYY HH:MI:SS)	DD.MM.YYYY HH:MI:SS
*Кодировка:	UTF-8

Название веб-сайта: Мой сайт

URL сервера (без http://):

E-Mail адрес по умолчанию:

Путь к корневой папке веб-сервера для этого сайта:  
(оставьте пустым, если все сайты работают на одном веб-сервере)

[вставить текущий]



Параметры:

*Язык:	[en] English
*Формат даты: (например: DD.MM.YYYY)	MM/DD/YYYY
*Формат даты и времени: (например: DD.MM.YYYY HH:MI:SS)	MM/DD/YYYY HH:MI:SS
*Кодировка:	UTF-8

Название веб-сайта: My site

URL сервера (без http://):

E-Mail адрес по умолчанию:

Путь к корневой папке веб-сервера для этого сайта:  
(оставьте пустым, если все сайты работают на одном веб-сервере)

[\[вставить текущий\]](#)

- Настройка кодировки для административного раздела сайта выполняется через форму управление параметрами языков, используемых в системе ([Настройки > Настройки продукта > Языки интерфейса](#)).

Также при настройке параметров языка можно определить формат времени и даты.

Параметры

Параметры языка системы

*ID:	ru
Активен:	<input checked="" type="checkbox"/>
*Название:	Russian
По умолчанию:	<input checked="" type="checkbox"/>
*Сортировка:	1
*Формат даты: (например: DD.MM.YYYY)	DD.MM.YYYY
*Формат даты и времени: (например: DD.MM.YYYY HH:MI:SS)	DD.MM.YYYY HH:MI:SS
*Кодировка:	UTF-8
Направление текста:	Слева направо

[Сохранить](#) [Применить](#) [Отменить](#)

Указанный формат будет использоваться при отображении даты и времени в административном разделе сайта.



The screenshot shows the 'Catalog books' section of the 1C-Bitrix CMS. The 'Авторы' folder is selected and highlighted with a red box. The table lists the following items:

	Имя	Размер файла	Изменен	Тип	Права на доступ продукта
...	...				
...	Авторы		02.02.2012 11:57:38	Папка	Полный доступ
...	Рецензии		02.02.2012 11:57:38	Папка	Полный доступ
...	Меню типа «left»	206 Б	02.02.2012 11:57:38	Скрипт PHP	Полный доступ
...	Каталог книг	4 КБ	02.02.2012 11:57:38	Скрипт PHP	Полный доступ

### Определение текущей кодировки

Текущая кодировка, используемая в публичном разделе сайта, определяется с помощью php-константы `LANG_CHARSET`, подставленной в область заголовка шаблона сайта.

При применении шаблона к сайту запрашивается значение параметра кодировка, заданное в настройках сайта. Константе `LANG_CHARSET` присваивается значение, равное значению параметра кодировка.

Пример кода, с помощью которого выполняется установка кодировки страниц, приводится ниже:

```
<head>
...
<meta http-equiv="Content-Type" content="text/html; charset=< ?echo LANG_CHARSET? >">
...
</head>
```



## Управление метаданными

Как правило, основной целью использования метаданных является оптимизация сайта для поисковых систем. Поисковые системы используют метаданные для индексации документов сайта.

Примером управления метаданными в продукте может служить механизм задания ключевых слов и описаний для страниц и разделов сайта. По умолчанию в дистрибутиве продукта настроено управление именно этими двумя типами метаданных (по аналогичной схеме список возможных вариантов может быть дополнен).

В разделе описываются основные инструменты и способы управления метаданными страниц и разделов, представленные в **Bitrix Framework**.

### Управление значениями метаданных через визуальный интерфейс

Чтобы иметь возможность управлять значениями метаданных, предварительно необходимо создать соответствующие свойства в настройках модуля **Управление структурой** ([Настройки > Настройки продукта > Настройки модулей > Управление структурой](#)):

Типы свойств:	Тип	Название
	description	Описание страницы
	keywords	Ключевые слова
	title	Заголовок окна браузера
	keywords_inner	Продвигаемые слова

**⚠ Важно!** Названия **типов** свойств, используемых для управления метаданными страниц, должны совпадать с названиями мета-тегов в языке HTML. Например, типы свойств **ключевые слова** и **описание** должны совпадать именами (name) соответствующих мета-тегов: `keywords` и `description`.

**⚠ Примечание:** Набор свойств может быть задан отдельно для каждого сайта, работающего под управлением системы.



**Настройки для сайтов**

Использовать индивидуальные настройки для каждого сайта:

Настройки для сайта: Моя компания

Типы меню:	Тип	Название
	left	Меню раздела
	top	Главное меню

Количество дополнительных параметров меню: 2

Типы свойств:	Тип	Название
	description	Описание страницы
	keywords	Ключевые слова
	title	Заголовок окна браузера

Подробную информацию по настройке свойств страниц и разделов, а также по управлению значениями метаданных вы можете узнать в курсах [Контент-менеджер](#) и [Администратор. Базовый](#).

**⚠ Обратите внимание!** Значения свойств, заданные для папки, по умолчанию будут использоваться для всех страниц и подразделов соответствующего раздела сайта (если для них не заданы собственные значения этих свойств).

### Управление метаданными в коде

Для вывода значений метаданных в коде страницы нужно воспользоваться функцией `ShowMeta()`, размещаемой в прологе шаблона дизайна сайта:

```
<head>
...
<?$APPLICATION->ShowMeta("keywords")?>
<?$APPLICATION->ShowMeta("description")?>
...
</head>
```

Предположим, что для страницы заданы следующие значения свойств ключевые слова и описание:



Свойство	Значение
Описание:	Система управления сайтом Текущее значение из свойств раздела: 1С-Битрикс: Управление сайтом
Ключевые слова:	веб, разработка, программирование Текущее значение из свойств раздела: 1С-Битрикс, CMS, PHP, bitrix, система управления контентом
Дополнительный заголовок (заголовок окна браузера):	
Теги (введите слова или словосочетания, разделяя их запятыми):	

**Еще...**

**Сохранить    Применить    Отменить**

С помощью функции `SetPageProperty()` значения данных свойств будут применены к странице:

```
< ?
$APPLICATION->SetPageProperty("keywords", "веб, разработка, программирование");
$APPLICATION->SetPageProperty("description", "Система управления сайтом");
?>
```

**⚠ Примечание:** Настройка свойств раздела может быть выполнена с помощью функции `SetDirProperty()` (например, в коде файла `.section.php`):

```
< ?
...
$APPLICATION->SetDirProperty("keywords", "дизайн, веб, сайт");
...
?>
```

Тогда в результате работы функции `ShowMeta()` в код страницы будет подставлен следующий HTML-код:

```
<meta name="keywords" content="веб, разработка, программирование ">
<meta name="description" content="Система управления сайтом ">
```



Если для самой страницы значение свойства не задано, то будет взято значение свойства вышестоящего раздела (рекурсивно до корня сайта). Если значение свойства не определено, то значение соответствующего мета-тэга останется не заданным. Свойства страницы могут быть установлены динамически из кода компонентов, расположенных на странице. Например, для страниц показа информации каталога или новостей свойства страницы могут быть установлены в соответствии с определенными свойствами элементов инфоблоков:

```
$APPLICATION->SetPageProperty("description",$arIBlockElement["PROPERTIES"][$META_DESCRIPTION]["VALUE"]);
```

В данном случае в качестве значения свойства страницы `description` будет использовано значение свойства элемента информационного блока с кодом `meta_description`. Таким образом, можно создавать свойства `keywords` и `description` для элементов каталога и динамически подставлять их в код страницы.

## Управление заголовком документа

Использование заголовка помогает привлечь внимание пользователей к странице сайта, а также создать общее представление о ее содержании и назначении. Кроме того, для того чтобы пользователь мог безошибочно определить, в каком окне открыта интересующая его страница, рекомендуется задавать дополнительные заголовки страниц для отображения в качестве заголовка окна браузера.

Создание и изменение заголовка страницы, а также окна веб-браузера может быть выполнено как с помощью средств [административного](#), так и [публичного раздела](#).

**⚠ Примечание:** Установка дополнительного заголовка окна веб-браузера осуществляется с помощью зарезервированного в продукте свойства `title`.

Для удобного использования свойства следует задать его название (например, **Дополнительный заголовок (заголовок окна веб-браузера)**) в настройках модуля **Управление структурой**.

Подробнее про установку нескольких заголовков смотрите на странице [Примеры работы](#).

**⚠ Примечание:** Некоторые компоненты могут самостоятельно устанавливать заголовок, учитывайте это при работе.



## Управление заголовком в коде

### **Задание и показ заголовка страницы**

Заголовок страницы может быть установлен следующими способами.

- При редактировании документа с помощью встроенного редактора (в режиме «Текст», «PHP» или «HTML»). В этом случае заголовок страницы задается путем подстановки в код документа следующей функции:

```
<?
$APPLICATION->SetTitle("О компании");
?>
```

- Заголовок документа может устанавливаться динамически одним из компонентов, расположенным на странице. В приведенном ниже примере в качестве заголовка страницы используется имя текущего информационного блока:

```
<?
$arIBlock = GetIBlock($ID, "news")
$APPLICATION->SetTitle($arIBlock["NAME"]);
...
?>
```

Вывод заголовка документа выполняется с помощью размещения функции `ShowTitle()` в месте показа заголовка страницы:

```
<H1><?$APPLICATION->ShowTitle()?></H1>
```

Если при выводе заголовка страницы функция `ShowTitle()` использует параметр `false`, это означает, что для установки заголовка страницы не нужно проверять значение свойства `title` (например, **Дополнительный заголовок (заголовок окна веб-браузера)**).

```
<H1><?$APPLICATION->ShowTitle(false)?></H1>
```

Т.е. в качестве заголовка страницы будет использован заголовок, установленный функцией `SetTitle()`.

**⚠ Примечание:** Подробнее про работу функции `ShowTitle(false)` смотрите на странице [Примеры работы](#).

### **Задание и показ заголовка окна веб-браузера**

Вывод заголовка окна веб-браузера выполняется с помощью кода `ShowTitle()`, размещенного в области `<head>` шаблона дизайна сайта:

```
<head><title><?$_APPLICATION->ShowTitle()?></title></head>
```

Заголовок окна веб-браузера может быть установлен с использованием различных механизмов. По умолчанию заголовок устанавливается в свойстве страницы `title` (например, **Дополнительный заголовок (заголовок окна веб-браузера)**). Если значение данного свойства не указано, то заголовок окна браузера будет установлен равным текущему заголовку страницы.

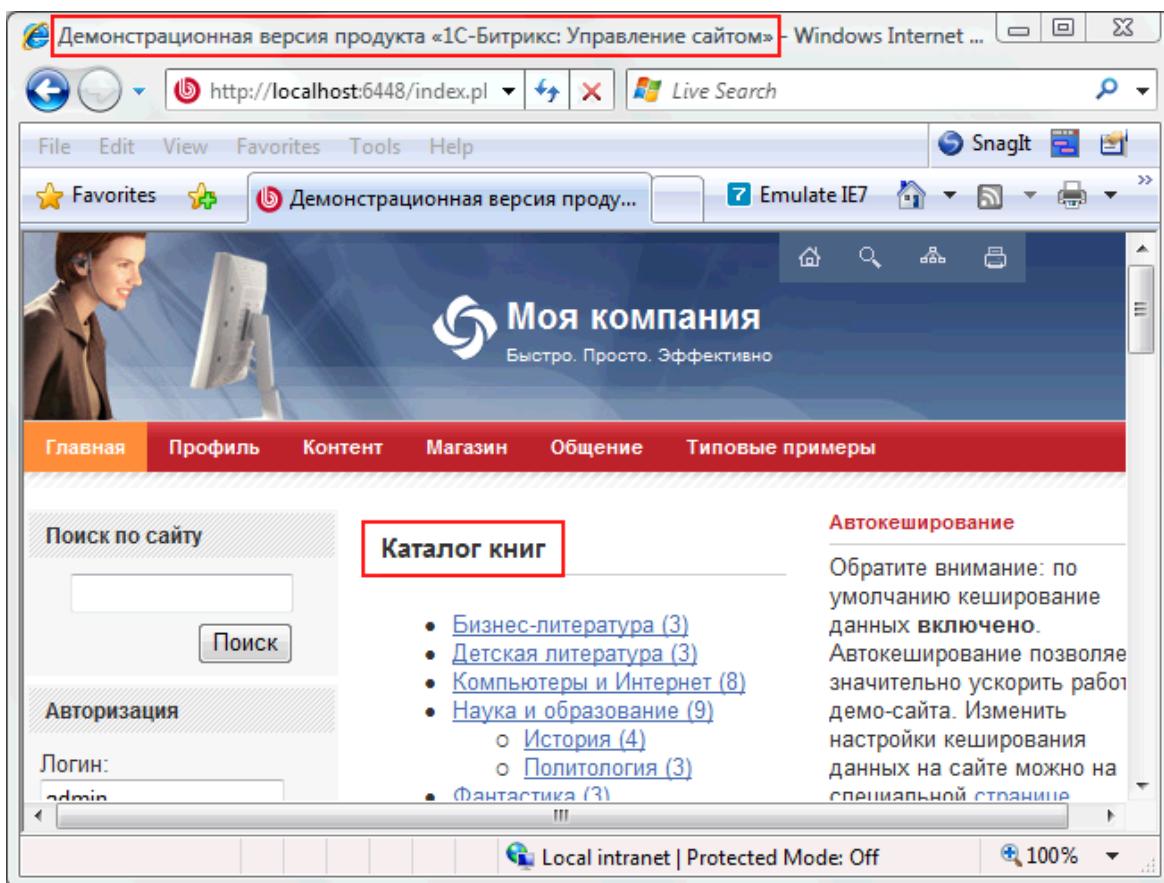
**⚠ Примечание:** Заголовок окна браузера может также устанавливаться с помощью функции `SetPageProperty()`; или задаваться в публичной части сайта. Подробнее смотрите на странице [Примеры работы](#).

**⚠ Внимание!** Если на странице располагается несколько одинаковых функций или компонентов, которые могут устанавливать заголовок, то используется будет тот заголовок, который задается в самой последней (самой нижней на странице) функции/компоненте.

### Примеры работы

В данном уроке мы поэтапно рассмотрим пример создания страницы, у которой заголовок окна браузера и сам заголовок страницы будут различаться. В ходе работы, возможно, потребуется изменить шаблон сайта.

В результате мы должны получить примерно следующий результат:

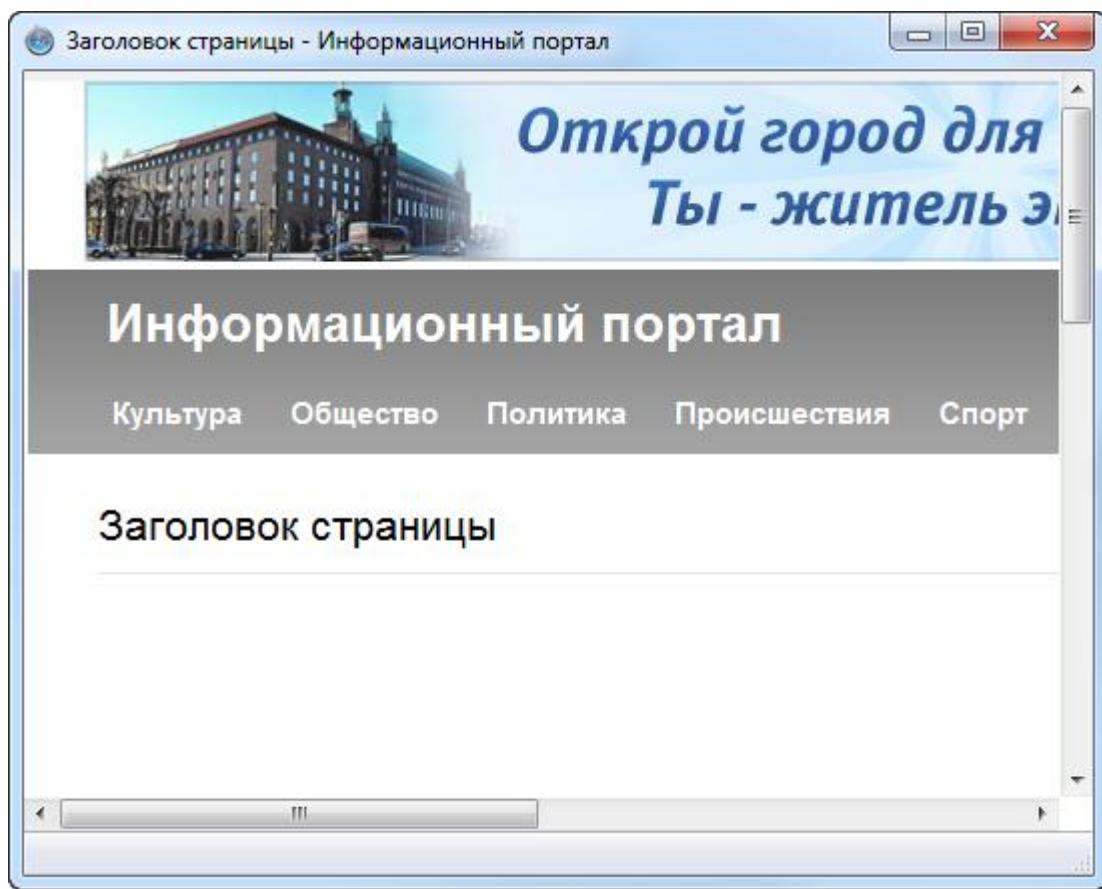


Выполним следующее:

- Зададим заголовок страницы из [интерфейса](#) или с помощью функции `$APPLICATION->SetTitle("Заголовок страницы")`:

```
<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Заголовок страницы"); ?>

<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php"); ?>
```



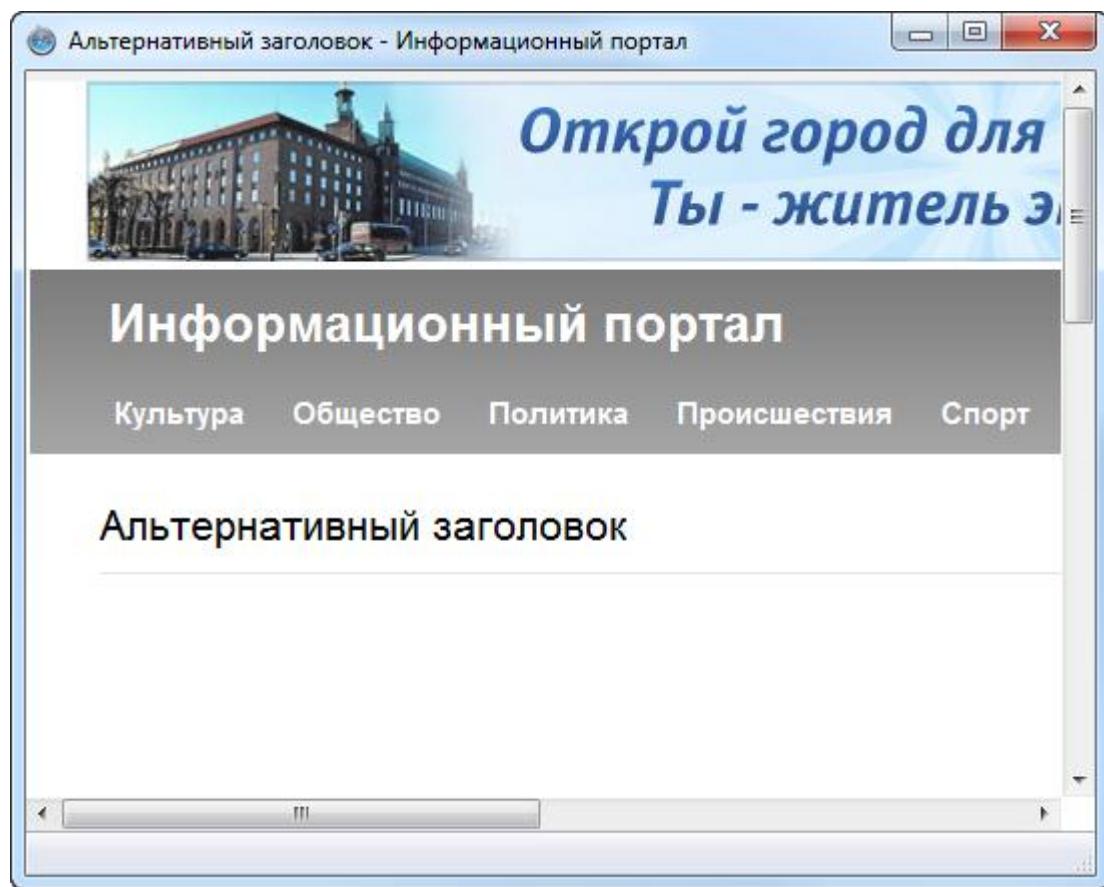
- Добавим дополнительный заголовок [через интерфейс](#) или с помощью функции `$APPLICATION->SetPageProperty('title', 'Альтернативный заголовок')`:

```
<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Заголовок страницы"); ?>

<? $APPLICATION->SetPageProperty('title','Альтернативный заголовок'); ?>

<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php"); ?>
```

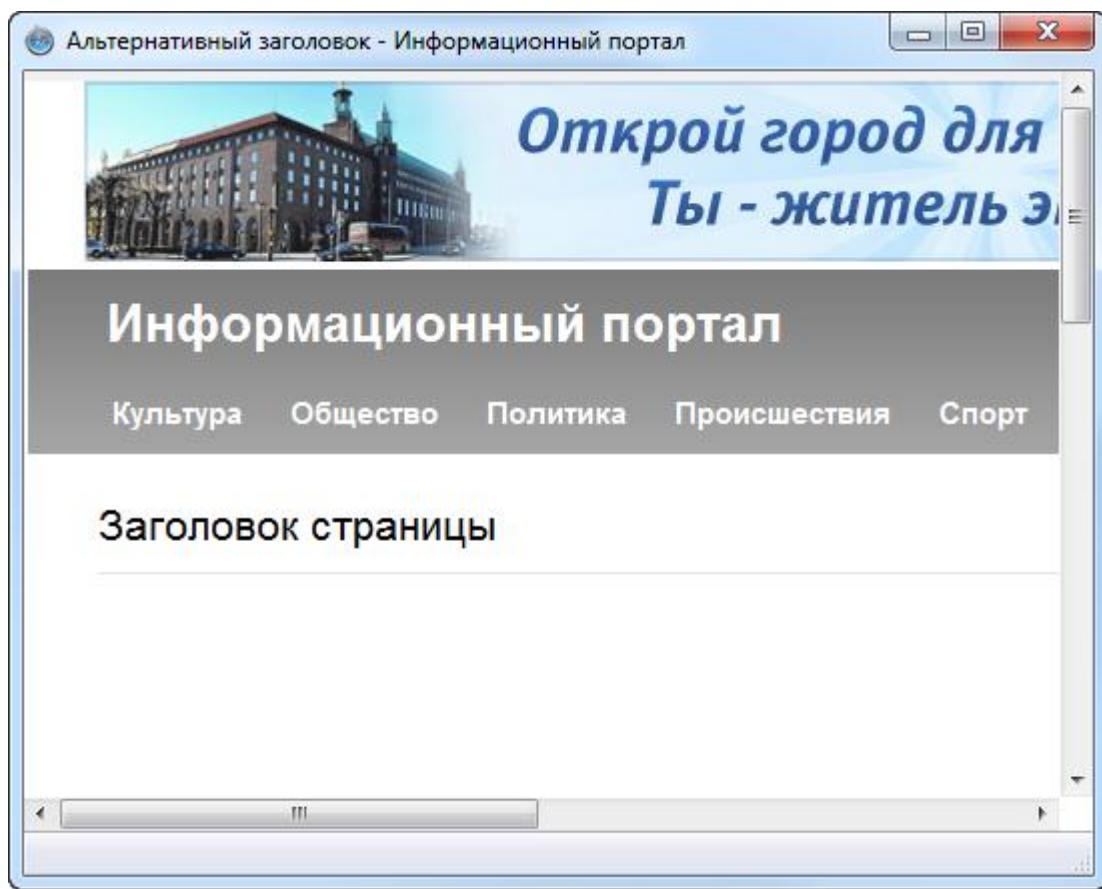
В системе функция `$APPLICATION->SetPageProperty()` имеет приоритет над функцией `$APPLICATION->SetTitle()`, поэтому в заголовке браузера и страницы будет отображено именно ее содержимое:



- Изменим шаблон страницы, если необходимо, так, чтобы вместо функции \$APPLICATION->ShowTitle() заголовок страницы (не браузера) задавала \$APPLICATION->ShowTitle(false).

В таком случае будет игнорироваться значение свойства страницы SetPageProperty('title', 'Альтернативный заголовок') и в качестве заголовка страницы будет использован заголовок, установленный функцией SetTitle().

Пример работы кода из предыдущего пункта, но с измененной функцией \$APPLICATION->ShowTitle() на \$APPLICATION->ShowTitle(false) в шаблоне сайта:

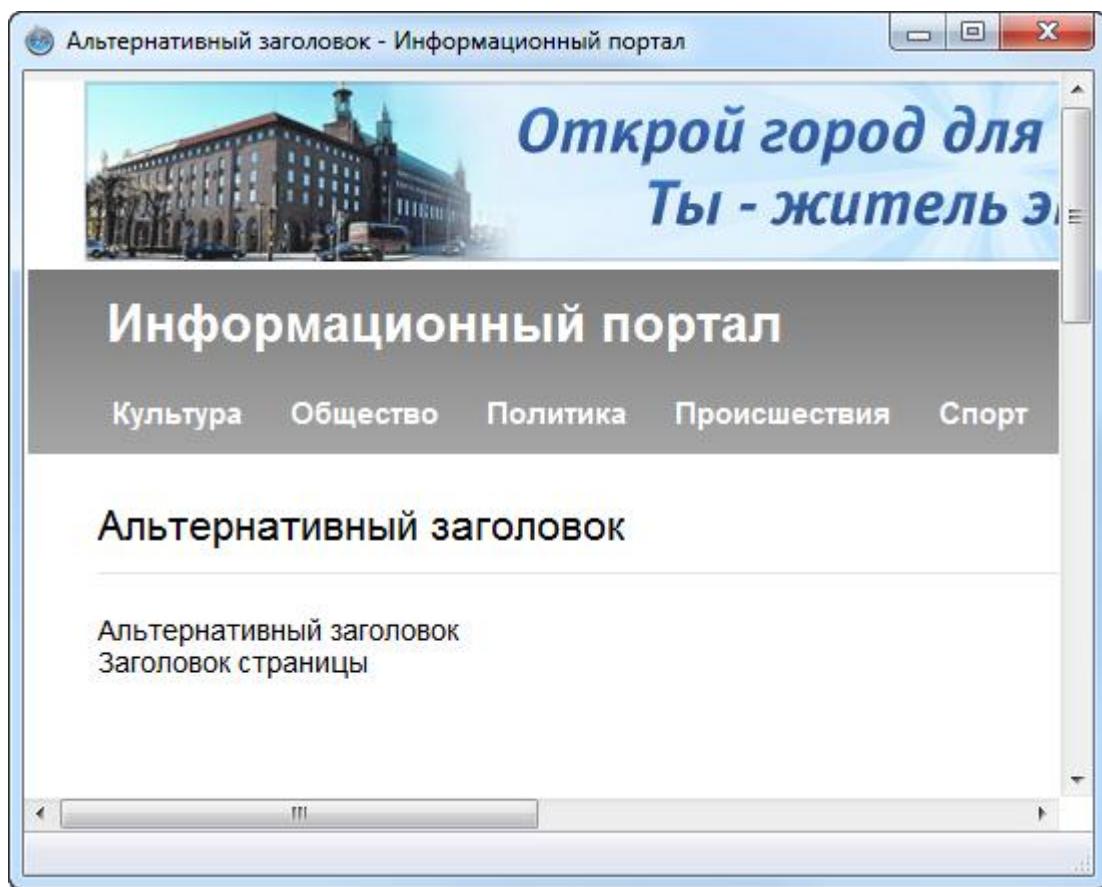


Отдельно продемонстрируем разницу работы функции `$APPLICATION->ShowTitle()` с параметром `false` на следующем примере, без изменения кода шаблона:

```
<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Заголовок страницы"); ?>
<? $APPLICATION->SetPageProperty('title','Альтернативный заголовок'); ?>

<? $APPLICATION->ShowTitle(); ?>
<br>
<? $APPLICATION->ShowTitle(false); ?>

<?require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");?>
```

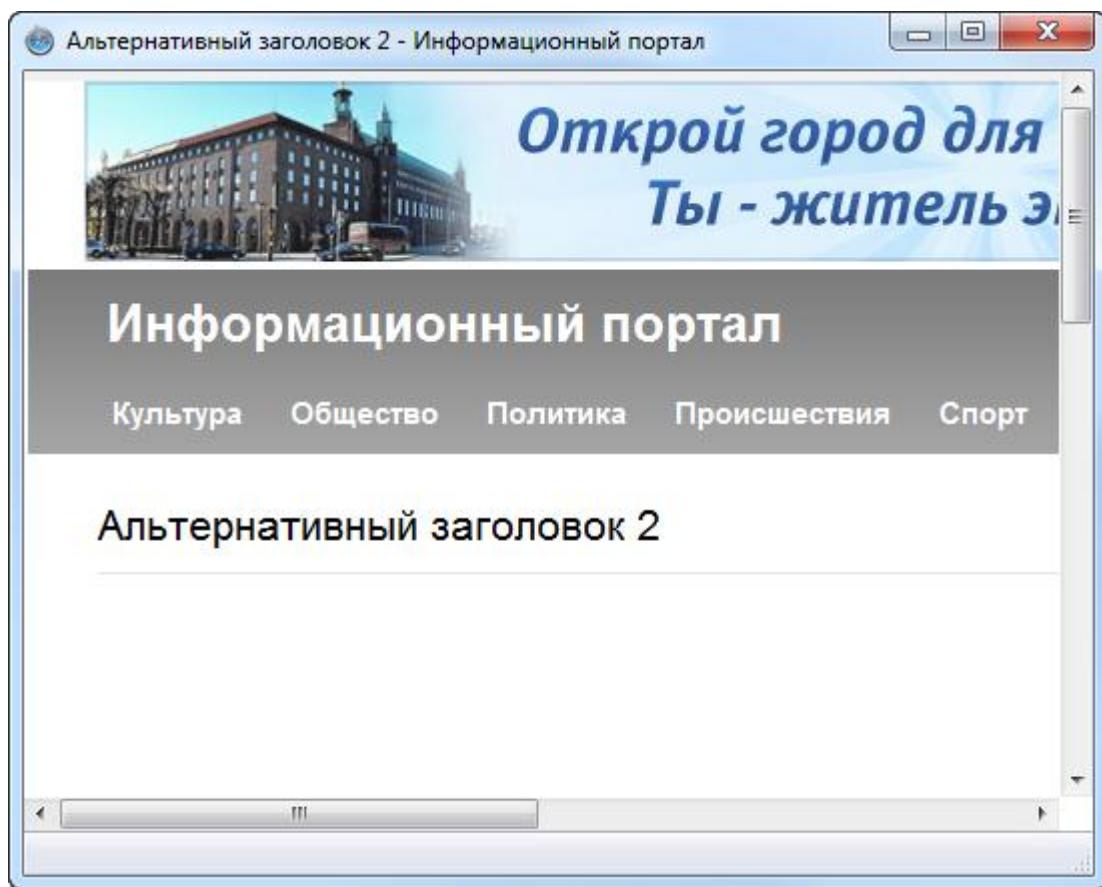


- Дополнительный пример работы нескольких функций по установке заголовка:

```
<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Заголовок страницы"); ?>

<?
$APPLICATION->SetPageProperty('title','Альтернативный заголовок');
//.....
$APPLICATION->SetTitle("Заголовок страницы 2"); // заголовок установлен не
будет т.к. у этой функции приоритет меньше
//.....
$APPLICATION->SetPageProperty('title','Альтернативный заголовок 2'); // будет
установлен именно этот заголовок т.к функция расположена ниже по коду
?>

<? require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php"); ?>
```



## Управление стилями

Таблицы стилей представляют собой совокупность правил, применяемых для оформления определенных элементов на страницах сайта. Технология каскадных стилей (CSS) позволяет хранить всю информацию о разметке страницы, используемых на ней шрифтах, цветах, стилях оформления меню и т.д. в определенном месте (одном или нескольких файлах).

Использование CSS упрощает задачу оформления и верстки страниц сайта. Кроме того, при изменении дизайна сайта или его отдельных элементов нет необходимости в редактировании каждой страницы: достаточно просто изменить соответствующую таблицу стилей.

Так, например, можно изменить оформление форума (шрифт, цвет элементов и др.) В приведенном примере стили форума задаются отдельно от стилей общего шаблона сайта.

			Заголовок темы	Автор темы	Ответов	Прочитано	Последнее сообщение
			Хотим стать партнером вашей компании!	admin	0	6	30.03.2007 10:02:59 Автор: admin



```
.forumborder {background-color:#CACBBD;}
.forumhead {background-color:#EAEBE2;}
.forumbody {background-color:#FBFBF9;}
.forumbodytext {font-family: Arial, Helvetica, sans-serif; font-size:smaller; color:#000000;}
.forumheadtext {font-family: Arial, Helvetica, sans-serif; font-size:smaller; color:#000000;}
.forumtitletext {font-family: Arial, Helvetica, sans-serif; font-size:smaller; color:#000000;}

.postsep {background-color: #9C9A9C; height: 1px}

.forumquote {font-family: Arial, Helvetica, sans-serif; font-size:8pt; color: #000000;
background-color: #FBFBF9; border : 1px solid Black;padding-top: 2px; padding-right: 2px;
padding-bottom: 2px; padding-left: 2px; text-indent: 2pt;}
.forumcode {font-family: Arial, Helvetica, sans-serif; font-size:8pt; color: #333333;
background-color: #FBFBF9; border : 1px solid Black; padding-top: 2px; padding-right: 2px;
padding-bottom: 2px; padding-left: 2px; text-indent: 2pt;}
...
```

	Заголовок темы	Автор темы	Ответов	Прочитано	Последнее сообщение
из	<a href="#">Добро пожаловать в форум партнеров</a>	Администратор Битрикс	0	9	22.12.2004 18:13:12 Автор: Администратор Битрикс

```
.forumborder {background-color:#96C0FA;}
.forumhead {background-color:#A9CAF7;}
.forumbody {background-color:#D7E6FB;}
.forumbodytext {
    font-family: Tahoma, Arial, Verdana, sans-serif;
    font-size: 80%;
    color:#042A69;
}
.forumheadtext {
    font-family: Tahoma, Arial, Verdana, sans-serif;
    font-size: 80%;
    color:#011B46;
}...
```

Таблицы стилей настраиваются отдельно для каждого шаблона сайта, используемого в системе, и размещаются в папках соответствующих шаблонов сайта. Таблицы стилей для

представления внутреннего содержания страниц сайта хранятся в файлах вида `styles.css` папки соответствующего шаблона. Дополнительные таблицы стилей, например, стили форума, могут иметь имена вида: `forum.css`.

 **Примечание:** В продукте добавлен механизм, позволяющий осуществлять раздельное хранение стилей:

- стили, используемые в **шаблоне дизайна**, хранятся отдельно в файле `template_styles.css`.
- стили, используемые при оформлении **контента страниц** (стили сайта), хранятся в файле `styles.css`. Стили из этого файла выводятся в выпадающем списке стилей при редактировании страниц в визуальном редакторе.

Формирование таблицы стилей сайта (файл `styles.css`) выполняется на странице редактирования шаблона дизайна на закладке **Стили сайта**. Важным элементом при создании таблицы стилей страниц является создание названий стилей. Названия следует создавать только для тех стилей, которые планируется использовать при редактировании страниц в режиме визуального HTML-редактора (секция **Описания стилей**).



Шаблон Стили сайта Стили шаблона Дополнительные файлы

Файл стилей сайта (styles.css)

```
}

code
{
    font-size:100%;
    font-weight:normal;
    display:block;
    padding:1.5em 1em 1em 1em;
    border-style:solid;
    border-width:1px;
    border-color:#F5F5F5;
    margin:1em 0;
    background-color:#F5F5F5;
    font-family:sans-serif;
}

.tablehead {background-color:#D8D9DA; }

.tablebody {background-color:#F8F8F8; }
```

Описания стилей

Имя стиля	Название стиля
information-block-head	Заголовок инфоблока
content-block-head	Заголовок на сером фоне
information-block-body	Текст элементов инфоблока
tablehead	Таблица - шапка
tablebody	Таблица - тело
Еще...	

Сохранить Применить Предпросмотр Отменить

Стили будут доступны в визуальном редакторе из выпадающего списка под именами, определенными в данной форме. Заданные здесь названия будут храниться в файле <идентификатор\_шаблона>/.styles.php (файл с именами стилей).

Создание таблицы стилей шаблона дизайна (файл template\_styles.css) выполняется на закладке **Стили шаблона** формы редактирования шаблона сайта

Шаблон Стили сайта Стили шаблона Дополнительные файлы

### Файл стилей шаблона (template\_styles.css)

```
html, body, form
{
    margin: 0;
    padding: 0;
}

html
{
    height:100%;
}

body
{
    height:auto !important;
    height:100%;
    min-height:100%;
    min-width:760px;
}

/*Font*/
body
{
    font-size: 80%;
    font-family: Tahoma, Verdana, Helvetica, sans-serif;
    color: #333;
}
```

## **Работа со стилями в визуальном HTML-редакторе**

Важным элементом при формировании таблицы стилей является добавление названий стилей. Названия стилей могут быть добавлены непосредственно в файл `.styles.php`. Данный файл размещается в папке шаблона сайта и имеет следующий формат.

```

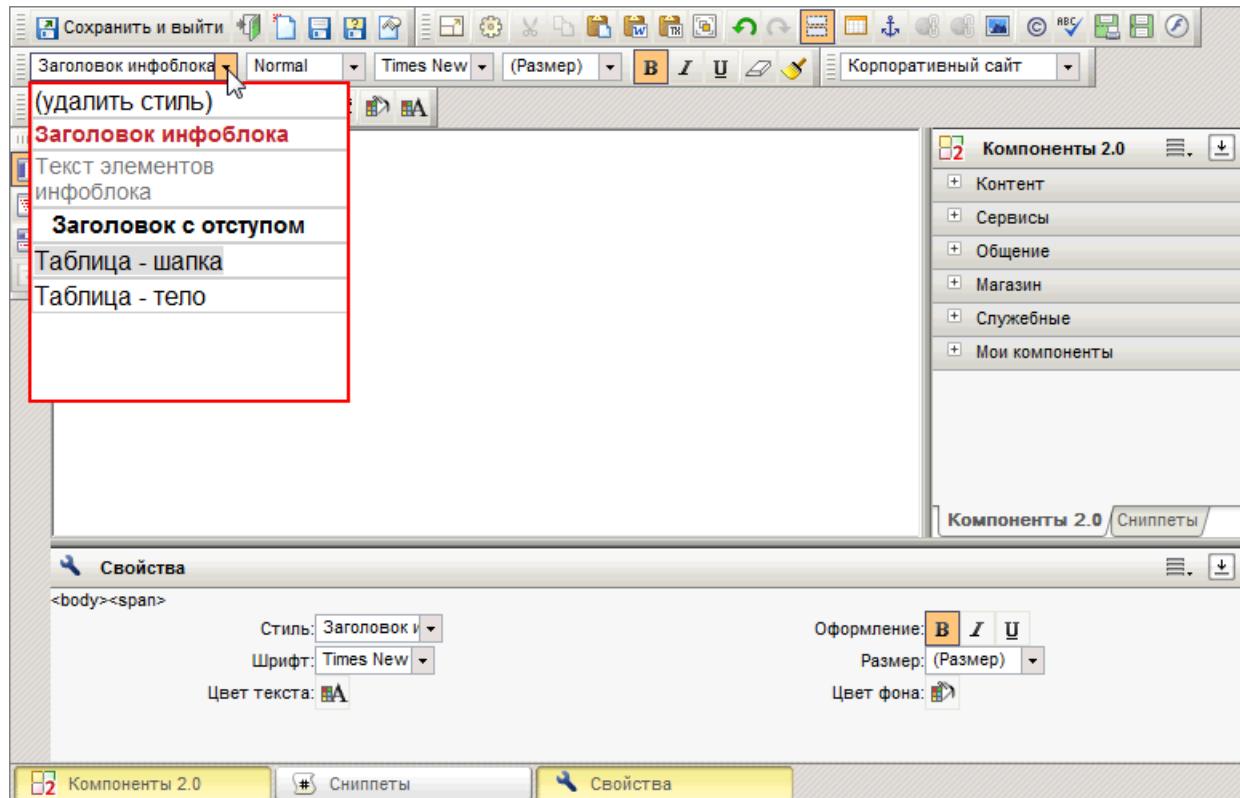
<?
$arStyles = [
    "information-block-head" => "Заголовок инфоблока",
    "content-block-head" => "Заголовок с отступом",
    "information-block-body" => "Текст элементов инфоблока",
    "tablehead" => "Таблица - шапка",
    "tbody" => "Таблица - тело",
];

```

```
return  
?> $arStyles;
```

Также названия стилей могут добавляться в форме редактирования шаблона на закладке **Стили сайта**. В этом случае файл .styles.php создается автоматически.

Названия стилей будут отображаться в визуальном редакторе в выпадающем списке.



Если в форме настроек модуля **Управление структурой** на закладке **Визуальный редактор** выбрать опцию **Разрешить в визуальном HTML редакторе выводить стили без названий**,



Настройки Визуальный редактор Медиабиблиотека Карты Доступ

### Настройки визуального редактора

Разрешить в визуальном HTML редакторе выводить стили без названий:

Отображать внешний вид стилей в выпадающем списке:

По умолчанию открывать HTML редактор в полноэкранном режиме:

Разрешить предварительный просмотр компонентов в визуальном редакторе:

Идентификатор элемента BODY области редактирования в визуальном редакторе:

Название CSS класса для элемента BODY области редактирования в визуальном редакторе:

то в списке будут представлены все доступные для страницы стили, в том числе те, для которых не заданы названия:

Сохранить и выйти

Заголовок инфоблока (удалить стиль) information-block

Заголовок инфоблока Текст элементов инфоблока content-block

Заголовок с отступом content-block-body

Корпоративный сайт

Свойства

Страница

Стиль: Заголовок инфоблока

Шрифт: Times New

Цвет текста: A

Оформление: B I U

Размер: (размер)

Цвет фона: A

Компоненты 2.0

+ Контент

+ Сервисы

+ Общение

+ Магазин

+ Служебные

+ Мои компоненты

Компоненты 2.0 / Сниппеты /

Для обозначений стиля с неустановленным названием будет использоваться имя, заданное при создании стиля: `information-block`, `content-block` и т.д.



Также в таблице стилей могут быть созданы стили, применяемые к определенным html-элементам страницы (например, границам таблицы, ячейкам, рисункам и др.). При выделении данных элементов в списке будут выводиться доступные для них стили.

### Пример таблицы стилей для шаблона сайта

#### **Стили для левого меню:**

```
.leftmenu, .leftmenuact {font-family: Arial, Helvetica, sans-serif;  
font-size:12px; font-weight: bold; text-decoration: none;}  
.leftmenu {color: #3F3F3F;}  
.leftmenuact {color:#FF5235;}
```

#### **Стили для верхнего меню:**

```
.topmenu {font-family: Arial, Helvetica, sans-serif; font-size:12px;  
font-weight: bold; color: #FFFFFF; text-decoration: none;}  
.topmenuact {font-family: Arial, Helvetica, sans-serif; font-size:12px;  
font-weight: bold; color: #FAC535; text-decoration: none;}
```

#### **Стили для таблиц:**

```
.tableborder {background-color:#9C9A9C;}  
.tablehead {background-color:#D8D9DA;}  
.tbody {background-color:#F8F8F8;}  
.tbodytext, .theadtext {font-family: Arial, Helvetica, sans-serif;  
font-size:12px;}  
.tbodytext {color:#000000;}  
.theadtext {color:#000066;}
```

### Механизм реализации

Таблицы стилей подключаются к шаблону сайта в области пролога с помощью функции ShowCSS().

```
<?  
$APPLICATION->ShowCSS();  
?>
```

Функция ShowCSS() выполняет подключение файла стилей из текущего шаблона сайта, а также всех дополнительных стилей определенных для данной страницы функцией SetAdditionalCSS().

```
<?
$APPLICATION->SetAdditionalCSS("/bitrix/templates/demo/additional.css");
?>
```

Дополнительные стили могут использоваться, например, для оформления форума, веб-форм, таблиц, некоторых типов меню и т.д.

При использовании функции `ShowCSS()` без параметров подключение стилей будет выполнено в виде ссылки на CSS файл:

```
<LINK href="/bitrix/templates/demo/styles.css" type="text/css" rel="STYLESHEET">
```

При этом стили, подключаемые с использованием `SetAdditionalCSS()`, будут включены в код страницы с использованием PHP функции `require()` (т.е. будут полностью включены в итоговый код страницы).

В случае использования функции `ShowCSS()` с параметром `false` файл стилей для текущего дизайна будет также включен в код страницы с использованием `require()`:

```
<?
$APPLICATION->ShowCSS(false);
?>
```

## **Настройка внешнего вида дополнительных элементов оформления сайта**

### **Настройка сообщений об ошибках**

Часто возникает необходимость выполнить настройку сообщений об ошибках, чтобы сообщение об ошибке было аккуратно показано в дизайне сайта.

Чтобы произвести настройку внешнего вида сообщения об ошибке соединения с базой данных, следует отредактировать файл: `/bitrix/php_interface/dbconn_error.php`.

Для того чтобы произвести настройку внешнего вида сообщения об ошибке в запросе к базе данных, следует отредактировать файл: `/bitrix/php_interface/dbquery_error.php`.



## Настройка файла, подключаемого при закрытии сайта

Чтобы произвести настройку внешнего вида файла, подключаемого при закрытии публичной части сайта, следует скопировать файл: /bitrix/modules/main/include/site\_closed.php и поместите его в /bitrix/php\_interface/<язык>/ или в /bitrix/php\_interface/include/.

## Настройка внешнего вида навигации постраничного просмотра

Постраничный показ информации организуется с использованием PHP функции NavPrint() – функции вывода ссылок для постраничной навигации. Для управления внешним видом постраничной навигации могут быть использованы следующие параметры: NavPrint(\$title, \$show\_allways=false, \$StyleText="text", \$template\_path) где:

- \$title – название выводимых элементов;
- \$show\_allways – если значение параметра **false**, то функция не будет выводить навигационные ссылки, если все записи умещаются на одну страницу. Если **true**, то ссылки для постраничной навигации будут выводиться всегда;
- \$StyleText – CSS класс шрифта для вывода навигационных ссылок;
- \$template\_path – путь к шаблону показа навигационных ссылок.

## Глава 5. Как создать простой сайт

**Цитатник веб-разработчиков.**

**Sergey Leshchenko:** Сайты разрабатываются не для того, чтобы их было удобно потом поддерживать веб-разработчикам.

Примерный алгоритм действий по созданию простого сайта следующий:

- Получение верстки и ознакомление с техническим заданием на сайт;
- Определение необходимого числа шаблонов и их структуры;
- Установка «1С-Битрикс»;
- Создание шаблонов и применение их к сайту;
- Создание и настройка необходимых элементов шаблона для SEO-продвижения сайта;
- Создание структуры сайта;
- Создание и настройка инфоблоков;
- Создание шаблонов визуальных компонентов;
- Тестирование.

Под простыми сайтами понимаются проекты, функционал которых можно реализовать штатными средствами **Bitrix Framework** с кастомизацией только шаблонов компонентов с целью изменения формы вывода данных. Кастомизация шаблона компонента - самый простой и несложный для освоения способ изменения вывода данных компонентов, не требующий серьезного программирования.

В этой главе будут рассмотрены основные шаги по созданию простого сайта.

### Техническое задание на сайт

**Цитатник веб-разработчиков.**

**Антипов Андрей:** Как правило, плохо составленное ТЗ приводит к расходам со стороны исполнителя, разочарованию заказчика и, в худшем случае, бесконечным доработкам (на основе логики «так это же очевидно»).

**Техническое задание** - исходный документ на проектирование технического объекта. ТЗ устанавливает основное назначение разрабатываемого объекта, его технические и тактико-технические характеристики, показатели качества и технико-экономические требования, предписание по выполнению необходимых стадий создания документации (конструкторской, технологической, программной и т. д.) и её состав, а также специальные требования..

Составление технического задания - обязательный шаг для разработки качественного, удовлетворяющего пользователя сайта. Обычно клиенты обращаются за разработкой сайта, не имея ни постановки задачи, ни, тем более, технического задания. А лишь имея некое формальное описание того, чего бы им хотелось получить в итоге. Такое формальное описание называется "бриф".

 **Пример брифа:**

*Необходимо разработать сайт автосервиса с Интернет-магазином автозапчастей, с формами заказа запасных частей, заявки на ремонт или ТО, отзыва о сайте, с фото-галереей.*

Бриф может быть таким как в примере, а может быть гораздо обширнее и походить на постановку задачи.

Чтобы начать разработку проекта, необходимо подготовить всю необходимую документацию:

- Постановка задачи, составляется после анализа брифа от заказчика;
- Техническое задание, составляется после того, как с постановкой задачи разобрались;
- ER-диаграмма - инфологическая или даталогическая (т.к. в 1С-Битрикс для хранения информации используются инфоблоки, то эти модели как бы совмещены) модель. На этой диаграмме отображаются все ваши сущности (инфоблоки) и связи между ними.
- Прототип будущего сайта. Для больших проектов рисуется подробный прототип в специализированных программах, например, [Axure](#).

Для небольших проектов острой необходимости в документации нет, есть несколько разработчиков, один заказчик и небольшой функционал. Все держится в голове. Фиксировать необходимо только основные договоренности с заказчиком.

Для небольших проектов достаточно "начертить" прототипы в графическом редакторе или на бумаге.

После готовности такого пакета документации можно смело приступать к реализации проекта:

- разработка дизайна;



- интеграция дизайна в систему 1С-Битрикс;
- создание структуры сайта;
- разработка функционала;
- тестирование;
- внедрение и сопровождение.

## Верстка базового шаблона

**Цитатник веб-разработчиков.**

*Иван Левый: Самая лучшая верстка - своя верстка. Сколько ни заказывали у внешних подрядчиков дизайн с версткой, все равно приходилось после них править.*

Когда дизайн готов, обычно применяется один из двух технологических процессов по интеграции дизайна в систему управления: либо разработчик сам верстает (т.е. переводит из графического эскиза в HTML) макет сайта, либо ему предоставляется уже готовая верстка, и он ее интегрирует в сайт. Вопросы создания верстки не входят в программу обучения **Bitrix Framework**, поэтому речь пойдет о готовом, сверстанном шаблоне.

## Шаблон сайта и визуальное редактирование

**Bitrix Framework** имеет возможность создания шаблона с помощью встроенного визуального редактора. Однако этот редактор разработан для работы с контентом. Поэтому этой возможностью пользоваться не рекомендуется. Редактирование шаблона дизайна сайта в визуальном режиме будет происходить некорректно, если:

- в атрибутах HTML-тегов содержится php-код;
- если строки и ячейки таблицы прерываются php-кодом при формировании таблицы.

Если в коде шаблона дизайна сайта есть такие особенности, то редактировать его следует только в режиме кода. Также не рекомендуется редактировать шаблон в визуальном редакторе **при наличии сложной верстки**.

## Определение количества необходимых шаблонов

Перед началом работы необходимо определить, сколько различных шаблонов сайта понадобится. Обычно при разработке сайта прорисовываются все различные страницы или основные элементы сайта.

**Bitrix Framework** позволяет использовать неограниченное число шаблонов и назначать их по разным условиям. Рассмотрим простейший вариант, что на всех этих страницах простого сайта фактически меняется только контентная часть, а дизайн – не изменяется. Исключение составляет главная страница, у которой контентная область устроена по-другому (не содержит заголовка страницы) и разделена на две части. Это можно реализовать как дополнительными условиями в шаблоне сайта, так и созданием двух шаблонов. Рекомендуется использовать дополнительные условия, в этом случае потребуется всего один шаблон сайта.

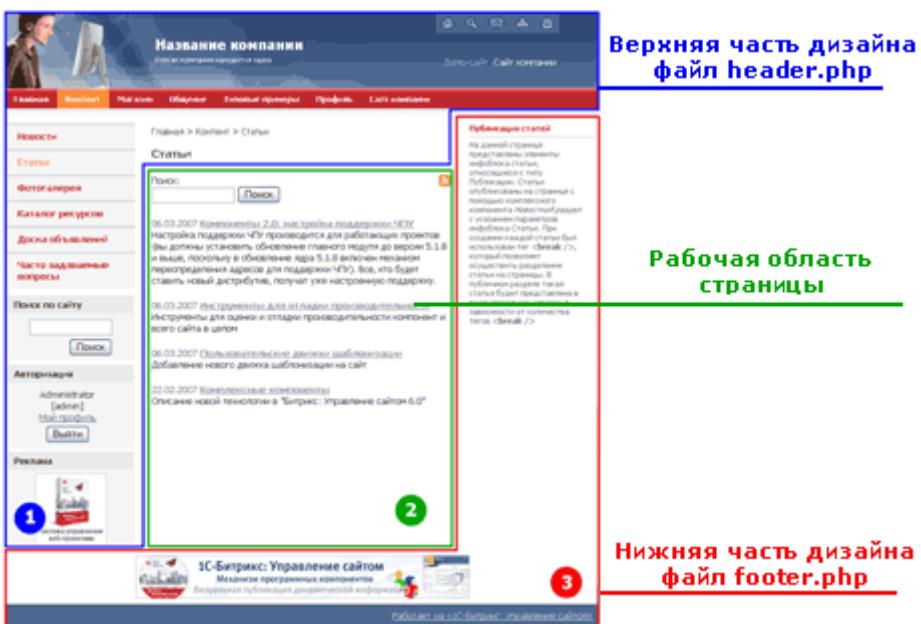
## Редактирование шаблона

Перейти к редактированию шаблона можно любым из способов:

- Создав (открыв для редактирования) нужный шаблон ([Настройки > Настройки продукта > Сайты > Шаблоны Сайтов](#));
  - Выбрав **Шаблон** в меню **Пуск** ([Настройки > Настройки продукта > Сайты > Шаблоны Сайтов](#));
  - С помощью кнопки **Шаблон сайта** на **Панели управления** ([Шаблон сайта > В Панели управления > Редактировать шаблон](#));
  - Прямое редактирование файлов **header.php** и **footer.php** в папке шаблона.

## Структурное деление шаблона

Проанализируйте шаблон и определите, какая часть кода должна относится к **Прологу** (файл `header.php`), какая к **Эпилогу** (файл `footer.php`), а какая часть - к **Рабочей области страницы**. Выбранные части кода должны быть размещены в соответствующих файлах, а рабочая область должна быть отмечена тегом `#WORK AREA#` в шаблоне сайта.



Добавьте соответствующий этим частям код в указанные файлы. Либо, если редактирование происходит в редакторе, разделите их тегом #WORK\_AREA#, удалив из шаблона контентную часть.

### Служебные директивы

Необходимо заменить некоторые части верстки на служебные директивы **Bitrix Framework** для создания шаблона:

- Заменить подключение стилей и, возможно, javascript файлов на директиву <?\$\_APPLICATION->ShowHead() ?>
- Заменить прописанный явно заголовок страницы на <title><?\$\_APPLICATION->ShowTitle() ?></title>
- Сразу после тэга <body> добавить <?\$\_APPLICATION->ShowPanel();?>. Если этого не сделать, **Панель управления** не появится.
- Перед всеми картинками добавить путь к ним /bitrix/templates/<? echo SITE\_TEMPLATE\_ID;?>/images/
- Заменить контент на специальный разделитель - #WORK\_AREA#

### Картинки и файлы стилей

Все изображения, относящиеся к шаблону размещаются в папке **/bitrix/templates/ID шаблона сайта/images/**.

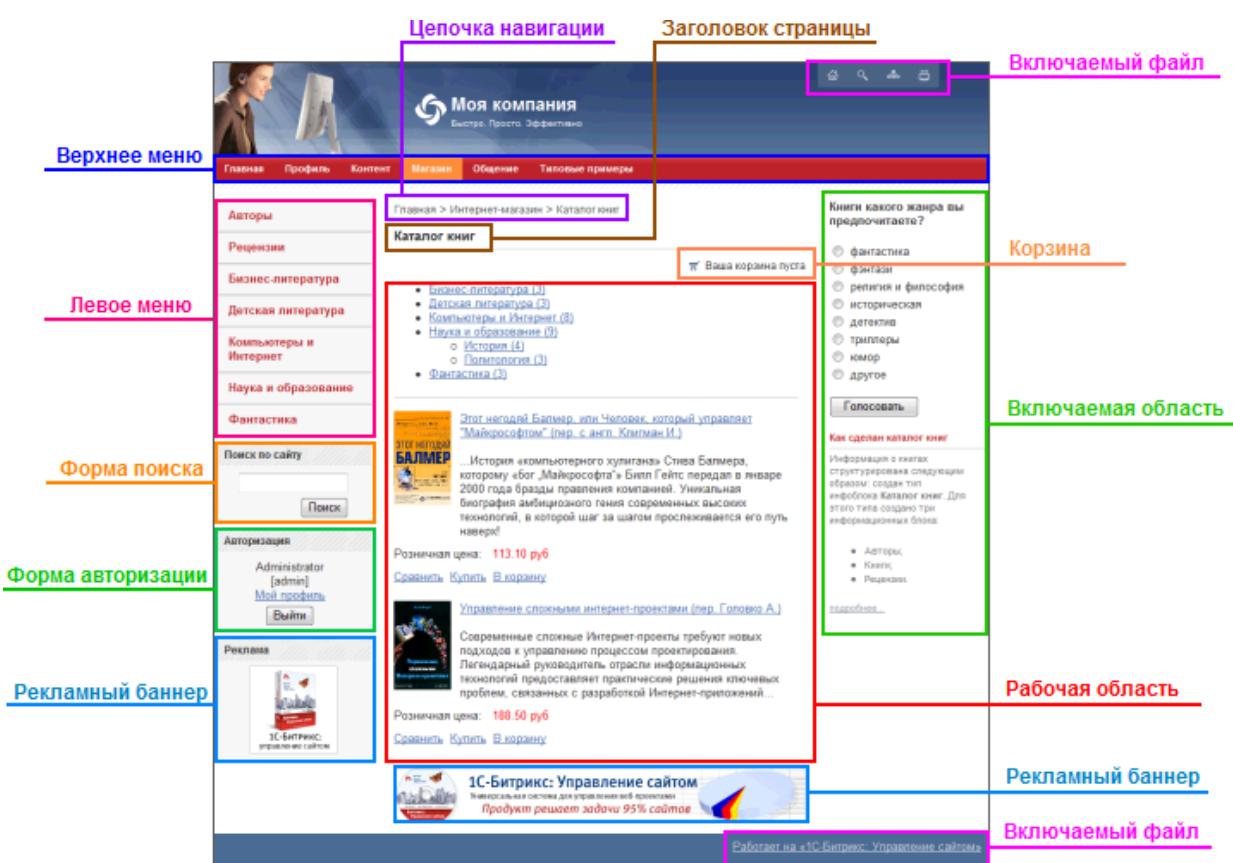
Описания стилей из представленной верстки переносятся в файл: **/bitrix/templates/ID шаблона сайта/styles.css**.

Описания стилей собственно шаблона переносятся в файл **/bitrix/templates/ID шаблона сайта/template\_styles.css**.

### Интеграция компонентов

Для создания полноценного сайта необходимо интегрировать в шаблон компоненты. Для этого необходимо выделить в дизайне сайта блоки, которые содержат динамическую информацию (вместо них будет размещен вызов компонентов) и блоки, информация в которых должна изменяться пользователем без изменения шаблона (это будут включаемые области).

Пример выделения функциональных областей:



Компоненты, которые должны выполнять функции в этих областях:





В зависимости от того, где должен быть расположен компонент в файле **header.php** или **footer.php**, удаляется html код, отображающий данную функциональную область и вставляется код компонента.

Если шаблон редактируется без визуального редактора системы (а рекомендуется именно такой вариант), то возможны 2 варианта вставки кода визуальных компонентов:

- Из **Пользовательской документации**. В ней описан формат вызова всех компонентов.
- Вставкой кода в визуальном редакторе на какой-нибудь странице или в новом окне браузера при редактировании любого другого шаблона. Полученный код копируется в нужное место шаблона.

 **Примечание:** Рекомендуется этот способ в силу того, что:

- документация может отставать от актуальной версии,
- заказчиком может использоваться устаревшая версия продукта.

## Примеры размещения компонентов

Компоненты версии 2.0 подключаются в коде страницы с помощью функции `IncludeComponent()`. В качестве параметров функции используется:

- название компонента в форме `<пространство_имен>:<название_компонента>`. Причем название компонента рекомендуется строить иерархически, начиная с общего понятия и заканчивая конкретным назначением компонента. Например, `«catalog»`, `«catalog.element»`, `«catalog.section.list»` и т.п.
- название шаблона. Шаблон компонента "по умолчанию" можно задавать пустой строкой, также можно явно определять `.default`
- массив параметров компонента `Array(...)`

## Добавление включаемых областей

Код для вставки компонента **Включаемая область (bitrix:main.include)**, с настройкой на вывод из файла, выглядит так:

```
<?$APPLICATION->IncludeComponent(
    "bitrix:main.include",
    "",
    Array(
        "AREA_FILE_SHOW" => "file",
        "PATH" => $APPLICATION->GetTemplatePath("include_areas/company_name.php"),
        "EDIT_TEMPLATE" => ""
    )
)
```



```
);?>
```

В этом примере `include_areas/company_name.php` - файл с включаемой областью:

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
```

*World Book*

*Все книги мира*

В файле включаемой области не требуется подключать пролог и эпилог (файлы **header.php** и **footer.php**). Необходимо лишь проверить, что файл включаемой области подключен из системы, а не вызван напрямую. Это делает строка:

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
```

## Меню

Для подключения меню необходимо вставить в нужное место шаблона вызов компонента:

```
<$APPLICATION->IncludeComponent(
    "bitrix:menu",
    "horizontal_multilevel",
    Array(
        "ROOT_MENU_TYPE" => "top",
        "MAX_LEVEL" => "3",
        "CHILD_MENU_TYPE" => "left",
        "USE_EXT" => "Y",
        "MENU_CACHE_TYPE" => "A",
        "MENU_CACHE_TIME" => "3600",
        "MENU_CACHE_USE_GROUPS" => "Y",
        "MENU_CACHE_GET_VARS" => Array()
    )
);?>
```

Это можно сделать как через визуальный редактор, так и через PHP-код шаблона.

## Кастомизация шаблонов компонентов

Кастомизация шаблона компонента, как правило, преследует две цели:

- Приведение формы вывода данных компонента в соответствие с дизайном сайта;
- Организация вывода данных компонента в виде, недоступном в стандартном варианте.

На простых сайтах, как правило, шаблоны требуют кастомизации только при решении первой задачи. Для изменения системного шаблона компонента под конкретный сайт, его необходимо сначала целиком скопировать в папку шаблона сайта. После этого можно перейти к редактированию скопированного шаблона.

Детально с кастомизацией шаблонов можно познакомиться в разделе [Кастомизация шаблона](#).

### Пример редактирования шаблона на основе компонента меню

Выделите в HTML-верстке код, отвечающий за показ верхнего меню. Например:

```
<div class="topmenu">
    <ul class="topmenu">
        <li><a href="#" class="first">На главную</a></li>
        <li><a href="#">Новости</a></li>
        <li><a href="#" class="selected">Магазины</a></li>
        <li><a href="#">Книги</a></li>
        <li><a href="#">Форум</a></li>
        <li><a href="#">О компании</a></li>
        <li><a href="#">О Контакты</a></li>
    </ul>
</div>
```

В этом коде пункты меню представлены в виде списка, который обладает следующими нюансами:

- У первого пункта меню должен быть указан стиль **first**;
- У выделенного пункта меню должен быть указан стиль **selected**;
- Меню является одноуровневым.

Модифицировать будем код шаблона **gray-tabs-menu**. Скопируйте шаблон в собственное пространство имен и откройте его для редактирования.

Редактирование шаблона можно проводить как в форме **Bitrix Framework**, так и копированием кода и правкой его в другом редакторе. Использовать другой редактор удобнее в случае объемных текстов, так как форма редактирования в **Bitrix Framework** не поддерживает цветовое выделение тегов. Для примера используйте **Notepad++**.



Код меню для этого шаблона выглядит так:

```
1 <?if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
2
3 <?if (!empty($arResult))?:?>
4 <div class="gray-tabs-menu">
5     <ul>
6 <?foreach($arResult as $arItem):?>
7
8     <if ($arItem["PERMISSION"] > "D")?:?>
9         <li><a href=<?= $arItem["LINK"]?>"><nobr><?=$arItem["TEXT"]?></nobr></a>
10    <endif?>
11
12 <?endforeach?>
13
14     </ul>
15</div>
16<div class="menu-clear-left"></div>
17<?endif?>
```

Рассмотрим каждую строчку этого шаблона:

#### Строки шаблона

1. Проверка, вызван ли этот файл напрямую или нет. Если напрямую – прекратить работу.
3. Если массив с пунктами меню \$arResult не пуст, то выполнять дальнейшие действия
- 4,5. Внешний блок и начало списка пунктов меню
6. Цикл по массиву с пунктами меню. В \$arItem – текущий элемент цикла.
- 8-10. Если текущий пользователь обладает правами на просмотр данной страницы, вывести элемент списка с ссылкой на эту страницу. В полях LINK и TEXT содержится адрес страницы и название пункта меню, соответственно.
12. Конец цикла по массиву с пунктами меню.

14,15. Конец списка пунктов меню и конец блока

16. Специальный HTML-тэг, специфичный для использованной верстки

17. Конец условия на наличие пунктов меню (см. строку 3)

Таким образом, шаблон меню содержит:

- область пролога шаблона меню;
- область тела шаблона меню (вывод повторяющихся элементов);
- область эпилога шаблона меню.

После адаптации шаблон примет вид (зеленым цветом выделены изменения):

```
1 <?if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
2
3 <?if (!empty($arResult)):?:>
4 <div class="topmenu">
5     <ul class="topmenu">
5a <? $cnt=0; ?>
6 <?foreach($arResult as $arItem):?:>
7
8 <?if ($arItem["PERMISSION"] > "D"):?:>
8a   <?if ($arItem["SELECTED"]): ?:>
8b     <li><a href=<?= $arItem["LINK"]?>" class="selected"><?= $arItem["TEXT"]?></a></li>
8c   <?else:>
8d     <?if ($cnt==0):?:>
8e     <li><a href=<?= $arItem["LINK"]?>" class="first"><?= $arItem["TEXT"]?></a></li>
8f   <?else:>
9     <li><a href=<?= $arItem["LINK"]?>"><?= $arItem["TEXT"]?></a></li>
10a   <?endif?>
10b   <?endif?>
10c <? $cnt++; ?>
10 <?endif?>
11
12 <?endforeach?>
13
14     </ul>
```



```
15</div>
16
17<?endif?>
```

Итак, что мы сделали?

- В строках 4,5 заменили стили у блока и у списка.
- В строке 5а ввели переменную \$cnt с единственной целью – отследить первый элемент списка – в верстке он задается другим стилем. Эта переменная используется в строке 8d b в строке 10с.
- В строках 8-10 добавили условие проверки активного пункта меню и первого пункта меню.
- И, наконец, удалили специфику верстки из 16 строки.
- Кроме того, у нас нет необходимости в специализированных стилях и картинках для этого шаблона. Поэтому нужно удалить из каталога /bitrix/templates/default/components/menu/top\_menu/ файл style.css и папку /images/.

**⚠ Внимание!** Вы можете использовать стили компонента и каталог шаблона компонента для хранения стилей и дополнительных файлов. Это позволит вам переносить шаблон компонента между проектами.

## Создание структуры сайта

Цитатник веб-разработчиков.

**Подшивалов Иван:** Если сайт от начала до конца делаете вы, то не обязательно давать пользователю права администратора. Можно создать отдельную группу и настроить права на чтение, у тех страниц, где информацию должен выводить компонент и на редактирование у статичных страниц.

[Создание структуры](#) сайта производится в соответствии с ТЗ на сайт.

На сайте должна быть представлена статическая (о компании, контакты) и динамическая (новости, каталоги, форум) [информация](#).

Создайте требуемую для вас структуру файлов и папок. При создании структуры нельзя забывать про служебные страницы - поиск и карту сайта.

**⚠ Примечание:** разработчик должен взять за правило: на проекте должны быть вменяемые тексты на страницах 404, 500 ошибок и сообщение о технических работах.

Пример структуры сайта:

Раздел сайта	Каталог/файл	Тип информации	Используемые компоненты
Главная	/index.php	Страница с динамической информацией	bitrix:news.list bitrix:catalog.top
Новости	/news	Страница с динамической информацией	bitrix:news
Магазины	/shops	Статическая страница	нет
Книги	/books/	Страница с динамической информацией	bitrix:catalog
Форум	/forum	Страница с динамической информацией	bitrix:forum
О компании	/about/	Статическая страница	нет
Контакты	/contacts/	Статическая страница	нет
Карта сайта (служебная страница)	/search/map.php	Страница с динамической информацией	bitrix:main.map
Результаты поиска (служебная страница)	/search/index.php	Страница с динамической информацией	bitrix:search.page
Обработчик ошибки 404	/404.php	Страница с динамической информацией	bitrix:main.map
обработчик ошибки 500-й	/500.html	Статическая страница (не Bitrix Framework)	HTML

## Настройка инфоблоков

Вывод динамичной [информации](#) из базы данных в **Bitrix Framework** осуществляется с помощью **информационных блоков**. Создавая сайт необходимо [продумать структуру](#) информационных блоков. Рассмотрим пример простого использования информационного блока на примере каталога.

Схема каталога товаров которую необходимо построить на сайте:

- Группа 1
  - Группа 1.1
    - Свой фильтр по свойствам
  - Группа 1.2
    - Свой фильтр по свойствам

### Возможные способы реализации

**Первый способ.** Все товары в одном инфоблоке. Информационный блок расположен на первом уровне (Группа 1).

Плюсы:

- **иерархия**, которой можно управлять из 1С;
- легко управляемая **структура каталогов** в рамках сайта;

Минусы:

- сложности со **свойствами товаров**, если товары разнородные;
- свойства будут храниться в **одной таблице**, что плохо повлияет на производительность.

**Второй способ.** Товары размещены в нескольких инфоблоках. Информационные блоки расположены на втором уровне (Группа 1.1; Группа 1.2 и так далее).

Плюсы:

- **индивидуальный фильтр** с возможностью хранения свойств в различных таблицах;
- **справочники** будут сразу разбиты по типам товаров и соответствующим свойствам;

Минусы:

- дополнительные усилия по настройке **импорта из 1С**: в настройках выгрузки необходимо указывать, какие разделы привязываются к какому инфоблоку;
- дополнительные усилия по созданию **структуре сайта**: необходимо вручную создать нужные подразделы, а в них на нужном уровне на страницах расположить простые или комплексные компоненты каталога для соответствующих инфоблоков;

После выбора схемы реализации нужно создать [тип](#) информационного блока, собственно [информационный блок](#), задать его [свойства](#) и наполнить контентом через импорт ([csv](#), [xml](#), [1С](#)) или вручную.

Последний шаг: настройка [параметров компонента](#) на созданный информационный блок.

## Управление служебными данными

### Кодировка страниц и формат отображения дат

**Bitrix Framework** поддерживает кодировку UTF-8, в которой на одной странице могут сочетаться различные языки – от русского до иероглифов, что позволяет не заботиться о кодировке страниц. Рекомендуем вам разработку сайтов именно в кодировке UTF-8, что позволит вам избежать проблем с настройками однобайтных кодировок.

Настройка кодировки выполняется отдельно для административного и публичного раздела.

## Управление метаданными

Основной целью использования метаданных является оптимизация сайта для поисковых систем. Поисковые системы используют метаданные для индексации документов сайта.

Примером управления метаданными в продукте может служить механизм задания ключевых слов и описаний для страниц и разделов сайта. По умолчанию в дистрибутиве продукта настроено управление именно этими двумя типами метаданных (по аналогичной схеме список возможных вариантов может быть дополнен).

Набор свойств может быть задан отдельно для каждого сайта, работающего под управлением системы.

## Управление заголовком документа

Одним из самых важных элементов сайта на сегодняшний день с позиции SEO-продвижения является заголовок окна браузера (тэг `<title>`). Практически повсеместно требуется, чтобы заголовок окна браузера отличался от заголовка страницы (тэг `<H1>`). В «1С-Битрикс» включена поддержка этого требования.



Установка заголовка может производиться как с помощью свойств страницы, так и из визуальных компонент, размещенных на странице. Установка дополнительного заголовка окна веб-браузера осуществляется с помощью зарезервированного в продукте свойства title.

## Глава 6. Инфоблоки

---

**Информационные блоки** - модуль, позволяющий каталогизировать и управлять различными типами (блоками) однородной информации. С помощью информационных блоков может быть реализована публикация различных типов динамической информации: каталоги товаров, блоки новостей, справочники и т.д.

Информационные блоки - ключевой момент **Bitrix Framework**. Практически всё, что делается в системе в той или иной мере завязано на этот модуль, даже если это и не отображается явно. Например, модуль **Обучение** не требует создания собственного типа информационного блока и самого инфоблока. Тем не менее, все что создается в его рамках работает на механизме информационных блоков.

Информационные блоки представляют собой очередной уровень абстракции над обычными таблицами СУБД, своеобразная "база данных в базе данных". Поэтому к ним частично применимы все те правила, которых придерживаются при проектировании БД.

Инфоблоки - сущность, которая в физической структуре БД создает 4 таблицы, не меняющиеся при изменении структуры данных: типы объектов, экземпляры объектов, свойства объектов и значения свойств объектов.

Плюсы такого подхода:

- удобный контроль над данными такой структуры из своего приложения,
- универсальность методов,
- общая структура данных для любого проекта,
- возможность многократно менять типы данных для полей без уничтожения самих данных.

Минусы такого подхода:

- повышенные требования к производительности,
- непрозрачность при прямом доступе к данным.

### Особенности упорядочивания элементов по разделам

Упорядочивание элементов инфоблоков по разделам может быть очень удобным для навигации по инфоблоку в административном интерфейсе. Фасетное упорядочивание делает навигацию ещё более удобной.

Также инфоблок с разделами уже сам в себе содержит описание того, каким образом следует его представлять пользователю. То есть достаточно обойти его самым обычным алгоритмом обхода дерева, чтобы вывести его в максимально удобном виде.



Особенности работы заключаются в некотором неудобстве работать отдельно именно с разделами. Так, например, если помимо инфоблока **Статьи** существует инфоблок **Книги**, то велика вероятность, что его элементы также будут нуждаться в классификации по дате публикации и по тематикам. В этом случае придётся ещё раз создавать такую же структуру разделов. Также весьма непросто будет вывести, например, список всех материалов (статей и книг) по одной тематике, упорядочив их по дате публикации. Также сложно будет вывести общий рубрикатор тематик в меню сайта.

В таком случае следует завести отдельный инфоблок **Тематики** и добавить в инфоблоки **Книги** и **Статьи** свойство ссылочного типа **Тематика**, а также свойство **Дата Публикации** типа **Дата**. Навигацию в административном интерфейсе тогда будет удобнее осуществлять при помощи установки фильтров по этим свойствам.

## Работа с инфоблоками штатными средствами

### Порядок работы

Создание любого раздела сайта с использованием информационных блоков необходимо проводить [в следующем порядке](#):

- Внимательное продумывание структуры инфоблоков.
- Создание нужного [типа инфоблоков](#) с настройкой параметров.
- [Создание самих инфоблоков](#) с настройкой параметров.
- Создание физической страницы (в случае использования комплексного компонента) или страниц (при использовании простых компонентов) и размещение на ней компонента (компонентов) с последующей настройкой его свойств.
- Кастомизация работы компонента под потребности ТЗ и дизайна сайта (кастомизация шаблона компонента, использование файлов `result_modifier.php` или `component_epilog`, кастомизация собственно компонента).
- Создание структуры внутри инфоблока.
- Создание элементов инфоблока

### Штатные возможности

Штатные средства модуля **Информационные блоки** достаточно обширны. Не ограничено ни количество типов инфоблоков, ни число самих инфоблоков, ни количество свойств каждого инфоблока, ни количество разделов или элементов.

### **Советы от веб-разработчиков.**

**Максим Месилов:** Для важных ИБ, которыми управляют несколько людей, желательно включать [журналирование](#). Так вы сможете



*быстро найти концы в случае непонятного удаления элементов или их редактирования.*

*Настройки журналирования - вкладка в настройках ИБ.*

## **Свойства инфоблоков**

Элементы каждого инфоблока имеют набор системных свойств, которые могут быть расширены пользовательскими свойствами. Свойства, задаваемые для инфоблока, различаются по своим типам:

- **Строка** - значение свойства задается в виде текстовой строки;
- **Число** - значение свойства задается в виде числа;
- **Список** - значение свойства выбирается из списка;
- **Файл** - в качестве значения свойства используется файл;
- **Привязка к разделам** - с помощью данного свойства можно задать связь между элементом данного инфоблока и разделами другого информационного блока;
- **Привязка к элементам** - задание связи между элементами информационных блоков «поштучно»;
- **HTML/текст** - значение свойства задается в виде текста с HTML-тегами;
- **Привязка к элементам по XML\_ID** - привязка хранится как строка и значением является XML\_ID привязанного элемента;
- **Привязка к карте Google Maps** - задается связь между элементом инфоблока и компонентом Google Map;
- **Привязка к Яндекс.Карте** - задается связь между элементом инфоблока и компонентом Яндекс.Карта;
- **Счетчик** - аналог **autoincrement** для БД. При добавлении элемента инфоблока значение будет больше на единицу, чем последнее. Стартовое значение задается произвольно. Можно использовать для журналов учета входящих документов и т.п., где должна быть непрерывная нумерация документов.
- **Привязка к пользователю** - с помощью данного свойства можно задать связь между элементом данного инфоблока и пользователями системы;
- **Дата/Время** - значение свойства задается в виде даты/времени;
- **Видео** - задается связь между элементом списка и медиафайлом;
- **Привязка к элементам в виде списка** - задание связи между элементами списком;
- **Привязка к теме форума** - с помощью данного свойства можно задать связь между элементом данного инфоблока и темами форума;

- **Привязка к файлу (на сервере)** - с помощью данного свойства можно задать связь между элементом инфоблока и файлом на удаленном сервере.

Каждый тип свойств характеризуется собственным набором параметров, настраиваемых в соответствующих формах. Свойства могут быть множественными, обязательными для заполнения.

 **Примечание.** Выносить в отдельные свойства рекомендуется только те характеристики, по которым будет происходить фильтрация. Остальные следует внести в описание товара в виде текста.

### Свойства разделов инфоблока

Имеется возможность задавать [пользовательские свойства](#) для разделов инфоблоков. Пользовательские поля в своем коде должны обязательно иметь приставку **UF\_**. Список типов полей несколько меньше, чем для самого инфоблока:

- Число;
- Да/Нет;
- Видео;
- Шаблон;
- Список;
- Страна;
- Дата/Время;
- Привязка к разделам инф. блоков;
- Файл;
- Целое число.

Как и свойства самого инфоблока, свойства разделов могут быть множественными и обязательными. Кроме этого можно задать будет ли участвовать в поиске и в фильтрации, может ли пользователь редактировать значение свойства и будет ли оно отображаться в общем списке свойств.

### Экспорт-импорт

Добавление большого числа элементов инфоблоков вручную - очень трудоемкое занятие. С целью облегчения добавления информации можно применять [импорт/экспорт данных](#) с использованием разных форматов файлов. Поддерживаются форматы:

- RSS
- CSV
- XML



Экспорт и импорт [в формате RSS](#) организуются с помощью специальных компонентов **RSS новости (экспорт)** (`bitrix:rss.out`) и **RSS новости (импорт)** (`bitrix:rss.show`) соответственно.

Экспорт данных из инфоблока в CSV файл выполняется с помощью формы **Выгрузка информационного блока** ([Контент > Информационные блоки > Экспорт > CSV](#)). Импорт данных, хранящихся в отдельном [CSV файле](#), в информационный блок выполняется в форме **Загрузка информационного блока** ([Контент > Информационные блоки > Импорт > CSV](#)).

**⚠ Примечание:** если нужно осуществить экспорт инфоблока как торгового каталога, то необходимо воспользоваться путем [Магазин > Торговый каталог > Экспорт данных](#). Возможен и импорт из файла формата CSV: в качестве торгового каталога. В этом случае необходимо воспользоваться путем [Магазин > Торговый каталог > Импорт данных](#).

Функционал экспорта\импорта инфоблоков в формат **XML** позволяет переносить не только содержимое инфоблоков, но и их свойства и изображения. Экспорт производится на странице **Экспорт XML** ([Контент > Информ. блоки > Экспорт > XML](#)). Импорт осуществляется на странице **Импорт XML** ([Контент > Информ. блоки > Импорт > XML](#)).

The screenshot shows the 'Настройки экспорта' (Export Settings) dialog box. It has the following fields:

- Файл для выгрузки: [File selection button] [Открыть...](#)
- Информационный блок: [Select dropdown] (выберите тип) (choose type)  
[Select dropdown] (выберите инфоблок) (choose infoblock)
- Длительность шага в секундах (0 - выполнять экспорт за один шаг): [Input field] 30
- Разделы: [Select dropdown] выгружать только активные (only active)
- Элементы: [Select dropdown] выгружать только активные (only active)
- Buttons at the bottom: [Экспортировать](#) (Export) and [Остановить экспорт](#) (Stop export)

## Настройка форм

Добавление/редактирование информационных блоков возможно как с административной, так и с публичной части. С публичной части это осуществляют контент-менеджеры. Формы добавления\редактирования инфоблоков желательно кастомизировать. В этом случае работа контент-менеджеров станет более легкой и удобной. Функция [настройки форм](#) - штатная и не требует программирования. Система позволяет:



- Задать значения полей формы по умолчанию.
- Задать автоматическую обработку фотографий по предварительно обозначенным параметрам.
- Задать порядок следования закладок формы и полей на них.

Если разработчику не удовлетворяют возможности штатной настройки форм, то он может [создать собственные](#).

### Типы хранения инфоблоков

При создании информационных блоков рекомендуется хранить свойства инфоблока в отдельной таблице, причем все значения свойств одного элемента хранятся в одной строке. Эта технология называется **Инфоблоки 2.0** и позволяет существенно ускорить работу системы, а также снять ряд ограничений в предыдущей версии инфоблоков. Например, теперь нет необходимости в дополнительном запросе [CIBlockElement::GetProperty](#) при выборе значений свойств функцией [CIBlockElement::GetList](#).

Возможности инфоблоков 2.0:

- При выборке элементов можно сразу получать значения свойств, т.к. количество присоединяемых таблиц в запросе не увеличивается с каждым свойством, а всегда равно единице.
- Фильтрация по значениям свойств происходит аналогично инфоблокам 1.0 (за исключением множественных).
- Выборка значений множественных свойств не приводит к декартовому произведению результата запроса - значения свойств передаются в виде массива.
- Для комбинированных фильтров по немножественным (единичным) свойствам появилась возможность ручного создания составных индексов БД для ускорения операций выборки.
- Для инфоблоков 2.0 нет возможности "сквозной" выборки элементов, когда в фильтре указывается тип инфоблока и символьный код свойства. В фильтре необходимо указывать **IBLOCK\_ID**.

Важным является полная совместимость API. Т.е. техника использования инфоблоков, свойств, элементов и их значений одинакова для обоих версий инфоблоков.

### Связь между инфоблоками

**Bitrix Framework** допускает создание взаимосвязей между информационными блоками с помощью свойств типа **Привязка к элементу** и **Привязка к разделу**.

Кроме этого вида взаимосвязи есть еще один вид, имеющий отношение к торговому каталогу: [SKU](#).



## Инфоблоки 2.0

При создании информационных блоков рекомендуется хранить свойства инфоблока в отдельной таблице, причем все значения свойств одного элемента хранятся в одной строке. Эта технология называется **Инфоблоки 2.0** и позволяет существенно ускорить работу системы, а также снять ряд ограничений в предыдущей версии инфоблоков. Например, теперь нет необходимости в дополнительном запросе [CIBlockElement::GetProperty](#) при выборе значений свойств функцией [CIBlockElement::GetList](#).

 **Примечание.** В документации к *Bitrix Framework*, в сообщениях форума на сайте компании и в других местах могут встречаться прежние названия технологии: **инфоблоки +**.

Возможности инфоблоков 2.0:

- При выборке элементов можно сразу получать значения свойств, т.к. количество присоединяемых таблиц в запросе не увеличивается с каждым свойством, а всегда равно единице.
- Фильтрация по значениям свойств происходит аналогично инфоблокам 1.0 (за исключением множественных).
- Выборка значений множественных свойств не приводит к декартовому произведению результата запроса - значения свойств передаются в виде массива.
- Для комбинированных фильтров по немножественным (единичным) свойствам появилась возможность ручного создания составных индексов БД для ускорения операций выборки.
- Для инфоблоков 2.0 нет возможности "сквозной" выборки элементов, когда в фильтре указывается тип инфоблока и символьный код свойства. В фильтре необходимо указывать IBLOCK\_ID.

Важным является полная совместимость API. Т.е. техника использования инфоблоков, свойств, элементов и их значений одинакова для обоих версий инфоблоков.

Для разработчика очень удобно и гибко то, что свойства хранятся в одной общей таблице и управляются информацией метаданных из **IBLOCK\_PROPERTY\_ID**. Так как всегда можно поправить какую-то метаинформацию и никакая другая информация при этом у вас не пропадет. Это свойственно простым инфоблокам.

Если информация в **инфоблоках 2.0** и меняется у свойства его тип, например с **Числа** на **Строку**, то при этом изменяется тип хранения в самой базе данных. То есть меняется не логика интерпретации продуктом значения этого свойства, а меняется само значение. То есть нельзя "играться" с данными.

С точки зрения производительности **инфоблоки 2.0** выигрывают на небольших справочниках с небольшим количеством (20-30) редко изменяемых свойств. Ленту

новостей нет никого смысла переводить на этот вид инфоблоков. Вы выигрываете в числе запросов, но проигрываете во времени их исполнения.

В инфоблоках 2.0 существует физическое ограничение БД на количество свойств инфоблока. На данный момент это не отслеживается в системе. При превышении этого физического ограничения вы уткнетесь в редкую для **Bitrix Framework** ошибку MySQL.

Большое преимущество инфоблоков 2.0 – возможность использования составных индексов. Однако ситуация, когда выполняется фильтр по = и по нескольким полям одновременно – ситуация достаточно редкая.

### Уровень информационного блока

У информационного блока есть признак **VERSION**, который при создании нового инфоблока определяет выбор хранения значений свойств информационного блока в общем хранилище или в выделенном. При выборе выделенного хранения для данного инфоблока в БД создаются две дополнительные таблицы, включающие в своё имя идентификатор инфоблока. В одной из них будут храниться множественные свойства, а в другой единичные и кэшированные значения множественных.

При редактировании инфоблока доступна ссылка на "конвертер" между типами хранения. Пользоваться им надо с большой осторожностью, т.к. продолжительность процесса конвертации зависит от общего объема значений свойств инфоблока. В течение всего процесса инфоблок находится в несогласованном состоянии (часть значений перенесена, а часть нет). На тестовой конфигурации для MySQL версии скорость была порядка 1500 элементов за 20-ти секундный шаг.

В классе [CIBlockResult](#) переопределён метод [Fetch](#). В нем происходит кэширование значений множественных свойств элемента, участвующих в выборке. Для этих же свойств типа **список** выбираются пары **ID=>VALUE** из справочника списков.

В API предусмотрен параметр **VERSION** в массиве полей **\$arFields** метода [CIBlock::Add](#). Его значения: 1 - для общего хранения и 2 - для выделенного (нового).

### Уровень свойств информационного блока

При редактировании свойств (смена признака множественности или типа свойства) для свойств в выделенном хранилище выполняются дополнительные операции по управлению таблицами. Такие как удаление/добавление колонок, вставка/обновление или удаление большого количества записей. Без настоятельной необходимости лучше избегать этого. Наилучшей методикой будет менять тип или множественность одного свойства за один раз. Причем для единичных свойств предпочтительнее сначала сделать его множественным а потом сменить тип, а для множественных наоборот - сначала тип, и уже потом делать его единичным.

## Уровень элементов информационного блока

При вставке элемента вставляется и соответствующая запись в таблицу хранения значений свойств элемента.

## Уровень значений свойств элементов информационного блока

Значения единичных свойств инфоблока с выделенным хранилищем ID - "синтетический" состоят из ID элемента и ID свойства, разделенных двоеточием. При обновлении множественных свойств происходит сброс кеша этих значений.

Значения свойств хранятся в 2-х таблицах (описание таблиц и их структуры имеет справочный характер и могут меняться в следующих версиях):

- **b\_iblock\_element\_prop\_mNN** - для множественных. Имеет ту же самую структуру, что и **b\_iblock\_element\_property**;
- **b\_iblock\_element\_prop\_sNN** - для единичных. Имеет поле **IBLOCK\_ELEMENT\_ID** - ID элемента инфоблока которому принадлежат свойства:
  - **PROPERTY\_XX** - хранит значения единичного свойства XX или кэш значений для множественного свойства;
  - **DESCRIPTION\_XX** - хранит описание для единичного свойства.

## Как достичь наибольшей эффективности при использовании новых инфоблоков?

Для использования преимуществ, создаваемых структурой хранения данных в новых инфоблоках, необходимо несколько модифицировать логику работы компонентов.

Например: если раньше шаблон кода был примерно таким:

```
<?
//Определяем массив нужных полей элемента
$arSelect = array(
    "ID",
    "IBLOCK_ID",
    "IBLOCK_SECTION_ID",
    "NAME",
    "PREVIEW_PICTURE",
    "DETAIL_PICTURE",
    "DETAIL_PAGE_URL",
);
//Получаем список элементов. (+1 запрос)
if($rsElements      =      GetIBlockElementListEx($IBLOCK_TYPE,      false,      false,
array($ELEMENT_SORT_FIELD => $ELEMENT_SORT_ORDER),
```



```
$ELEMENT_COUNT, $arFilter, $arSelect))  
{  
    //Инициализация постраничного вывода.  
    $rsElements->NavStart($ELEMENT_COUNT);  
    $count = intval($rsElements->SelectedRowsCount());  
    if ($count>0)  
    {  
        //Для каждого элемента:  
        while ($obElement = $rsElements->GetNextElement())  
        {  
            $arElement = $obElement->GetFields();  
            //Получаем его свойства. (+1 запрос)  
            $arProperty = $obElement->GetProperties();  
            //Ниже можно пользоваться значениями свойств.  
            //Например:  
            echo $arProperty["SOURCE"], "  
";  
            //и т.д. и т.п.  
        }  
    }  
}  
?>
```

Теперь, после преобразования в новый тип хранения, стало возможным избавится от запросов в цикле:

```
<?  
//Определяем массив нужных полей элемента  
$arSelect = array(  
    "ID",  
    "IBLOCK_ID",  
    "IBLOCK_SECTION_ID",  
    "NAME",  
    "PREVIEW_PICTURE",  
    "DETAIL_PICTURE",  
    "DETAIL_PAGE_URL",  
    "PROPERTY_SOURCE", //Выбираем нужное нам свойство  
    // И все другие которые могут понадобится
```



```
// непосредственно в списке
);

//Получаем список элементов. (+1 запрос)
if($rsElements      =      GetIBlockElementListEx($IBLOCK_TYPE,      false,      false,
array($ELEMENT_SORT_FIELD => $ELEMENT_SORT_ORDER),
Array("nPageSize"=>$ELEMENT_COUNT), $arFilter, $arSelect))
{
    //Инициализация постраничного вывода.
    $rsElements->NavStart($ELEMENT_COUNT);
    if($obElement = $rsElements->GetNextElement())
    {
        //Для каждого элемента:
        do
        {
            $arElement = $obElement->GetFields();
            //Ниже можно пользоваться значениями свойств.
            //Например:
            echo $arElement["PROPERTY_SOURCE"],"
";
            //и т.д. и т.п.
        }
        while ($obElement = $rsElements->GetNextElement())
    }
}
?>
```

## План действий при проблемах

При возникновении проблем в работе проекта для их устранения рекомендуем воспользоваться следующим алгоритмом:

- Ставьте перед собой конкретную цель. Так как процесс оптимизации, улучшения – бесконечный. Например: каждая страница с каталогом ваших товаров должна открываться за определенное время, скажем 0,2 секунды. Только установка таких конкретных целей и их решение может быть эффективна, в отличие от неясных и формализованных пожеланий, "что бы лучше работало".
- С помощью страницы [отладки](#) ([Настройки > Производительность > SQL Запросы](#)) найти и убрать лишние запросы.

Лишний запрос – это:

- Запрос в цикле (вынести из цикла в переменную).
  - Запросы, которые добирают данные в цикле. (Лучше собрать данные в фильтре, а потом одним запросом вывести данные, потом дополнительный цикл – разложить данные) в этом случае запросы не зависят линейно от количества элементов.
  - Убрать из запросов неиспользуемые данные (`$arSelect`). Точно указывать поля, которые впоследствии будут использоваться. Это в разы повышает производительность, потому что такое четкое указание используемых данных означает для базы данных сокращение объемов сортировки. База данных производит такую сортировку не на жестком диске, а в оперативной памяти.
  - Оценить возможность использования **PROPERTY\_\*** в сочетании с инфоблоками 2.0. Инфоблоки 2.0 хранят свои свойства в отдельной таблице. Эта таблица при упоминании **PROPERTY\_\*** присоединяется на этапе выборки. И до выборки значения свойств присоединение этих свойств не происходит.
- Когда это не применимо? Например, при малой выборке из большого числа элементов (10 новостей из нескольких тысяч записей). Момент этот не очень однозначный, и малозависим от разработчика. Причина в том, что результат выборки из инфоблоков и инфоблоков 2.0 – разный. В простом инфоблоке при выборке записи начинают размножаться. А инфоблоки 2.0 возвращают массив значения свойства. Если код в шаблоне не предусматривает эту ситуацию, то смена обычных инфоблоков на инфоблоки 2.0 приведет только к тому, что шаблон «развалится».
- Посмотреть план исполнения самых тяжелых запросов и добавить/удалить индексы.

## Фильтрация

Цитатник веб-разработчиков.

*Nikolay Ryzhonin:* Все таки более правильный и эффективный путь это анализ с `explain` медленных запросов и анализ кода их вызвавшего. А уже по результатам добавление необходимых индексов или изменение кода приложения. Так как, не всегда медленные запросы следствие проблем в API, часто встречаются случаи его неправильного использования.

Кеширование иногда - это зло. Если кешировать каталог с многовариативным фильтром при большой посещаемости, то генерация кеша может превышать 200 мегабайт в минуту. Любая квота на диске заполниться достаточно быстро. Возникают и проблемы с очисткой этого кеша. Отключение такого кеша разгрузит файловую систему на запись.

Не надо бояться создавать индексы. Хотя точно сказать какие индексы создавать в каждой конкретной ситуации заочно нельзя, надо всегда рассматривать конкретную ситуацию. В этом помогает инструмент [Индексы](#).



Один из часто используемых индексов - индекс по списочному свойству. Это индекс для простых инфоблоков. **b\_iblock\_element\_property** - в этой таблице хранятся значения свойств. Её и индексируем: значение свойства, его ID, ID элемента. Создание такого индекса при фильтре по списковому свойству фактически исключает обращение MySQL к указанной таблице, так как все имеющиеся в запросе поля присутствуют в индексе.

Индексы нужны для конкретных выборок на конкретных проектах. В зависимости от архитектуры и логики проекта медленные запросы получаются свои, и для них нужны свои индексы, часто составные.

### Совет от веб-разработчиков.

**Антон Долганин:** Если требуется обновить большое количество элементов, то имеет смысл обновление производить в обход индексации поиском:

```
$el->Update($arEls['ID'], array('PREVIEW_TEXT' => $preview,
'DETAIL_TEXT' => $detail), false, false);
```

4-й параметр в *false*. А уже после обработки запустить переиндексацию инфоблоков сайта. Запись в модуль поиска увеличивает конечное время исполнения в десятки-сотни раз.

## Типы фильтрации

В большинстве функций и методов модуля информационных блоков, выбирающих списки, в фильтрах возможно использование различных типов фильтрации. Символы типов фильтрации указываются непосредственно перед названием фильтруемого поля.

### Типы:

- (пустой) - для строковых полей означает поиск по маске (в маске: "%" - произвольное число любых символов, "\_" - один произвольный символ), для полей с нестроковыми типами поиск "равно".

```
<?
// найти элементы у которых название начинается на "#"
$res = CIBlockElement::GetList(Array(), Array("NAME"=>"#%"));

// найти элементы с идентификатором 100
$res = CIBlockElement::GetList(Array(), Array("ID"=>"100"));
?>
```

- "!" - для строк выражение не попадающее под маску, или не равно (для остальных типов полей).



```
<?
// найти элементы у которых название не начинается на "#"
$res = CIBlockElement::GetList(Array(), Array("!NAME=>#%"));
?>
```

- "?" - с применением логики, работает только для строковых свойств.

```
<?
// найти элементы у которых название содержит "One" или "Two"
$res = CIBlockElement::GetList(Array(), Array("?NAME=>One | Two"));
?>
```

- "<" - меньше;
- "<=" - меньше либо равно;
- ">" - больше;
- ">=" - больше либо равно.

```
<?
// найти элементы у которых название начинается на "A"
$res = CIBlockElement::GetList(Array(), Array(">NAME=>A", "<NAME=>B"));

// найти элементы с идентификатором большим 100
$res = CIBlockElement::GetList(Array(), Array(">ID=>100"));
?>
```

- "=" - равно;
- "!=" - не равно.

```
<?
// найти элементы у которых название равно "ELEMENT%1"
$res = CIBlockElement::GetList(Array(), Array("=NAME=>ELEMENT%1"));
?>
```

- "%" - подстрока;
- "!"% - не подстрока.

```
<?
// найти элементы у которых в названии есть символ процента "%"
$res = CIBlockElement::GetList(Array(), Array("%NAME=>%"));
?>
```

- "><" - между;
- "!"><" - не между.



В качестве аргумента данные типы фильтров принимает массив вида Array("значение ОТ", "значение ДО")

```
<?
// найти элементы у которых название начинается между "A" и "B" или между "D" и "E"
$res = CIBlockElement::GetList(Array(), Array("><NAME"=>Array(Array("A", "B"), Array("D", "E"))));

// найти элементы, у которых дата начала активности не в периоде 2003 года
$res = CIBlockElement::GetList(Array(),
    Array("!><DATE_ACTIVE_FROM"=>
        Array(date($DB->DateFormatToPHP(CLang::GetDateFormat("FULL")),
            mktime(0,0,0,1,1,2003)),
        date($DB->DateFormatToPHP(CLang::GetDateFormat("FULL")),
            mktime(0,0,0,1,1,2004)))));

?>
```

## Работа с инфоблоками через API

**Цитатник веб-разработчиков.**

***Ban Dmitry:** Ну да, при работе с битриксом часто надо думать, в какие запросы трансформируется твой вызов API. Но, думается мне, в любых других высоконивневых средах разработки то же самое.*

Потребности заказчиков сайтов очень разнообразны. Штатный функционал **Bitrix Framework** не может решать всех задач, которые могут быть поставлены перед разработчиком при создании интернет-проектов. Для реализации нестандартных задач необходимо использовать API. [API инфоблоков](#) рекомендуется особенно внимательно изучить. Они чаще всего используются при программировании.

**⚠ Внимание!** Прямые обращения к базе данных настоятельно не рекомендуются. В этом случае не гарантируется работа базовых функций системы. Кроме того, это может привести к нарушению целостности данных.

API модуля состоит из нескольких высоконивневых функций для выборки данных в публичном разделе сайта и набора классов с низкоуровневыми методами для более специализированной работы.

Перед использованием модуля необходимо проверить, установлен ли он, и подключить его при помощи конструкции:

```
<?
if(CModule::IncludeModule("iblock"))
{
    //здесь можно использовать функции и классы модуля
}
?>
```

Для получения данных при показе в публичном разделе сайта можно пользоваться функциями с простыми параметрами и предустановленными фильтрами. Эти функции выбирают по умолчанию те значения, которые подходят для места выборки, а именно только активные, привязанные к текущему сайту, подходящие по правам доступа и т.п.

Вся работа с датами через API (вставка, выборка, фильтры и т.п.) производится в формате текущего сайта или, если в административной части, в формате текущего языка.

Ряд функций API доступен всегда, т.е. описан в главном модуле, а ряд функций зависит от используемого модуля, и может присутствовать или отсутствовать в различных редакциях продукта. Например, функции для работы с социальной сетью присутствуют в редакциях **«1С-Битрикс: Управление сайтом - Бизнес»** и выше, а также в **«1С-Битрикс: Корпоративный портал»**.

Для большинства классов **Bitrix Framework** доступны функции:

- Выборка данных (`GetList`).
- Занесение нового элемента (`Add`).
- Обновление и удаление элемента (`Update`).
- Удаление элемента (`Delete`).
- И другие функции.

Для большинства модулей предлагается специализированная структура классов, механизм событий и дополнительные функции. В частности, для модуля Информационные блоки приводится описание:

- Всех используемых таблиц в базе данных, в том числе полей таблиц.
- Классов для работы с типами инфоблоков, инфоблоками, элементами, разделами, полями.
- Событий, происходящих при добавлении, изменении и удалении объектов модуля.
- Функций, расширяющих возможности ядра.
- Способов создать пользовательские формы редактирования и свои типы данных.
- Другая информация.



В уроках главы будут рассмотрены некоторые примеры использования API информационных блоков.

### Совет от веб-разработчиков

**Антон Долганин:** Всегда минимизируйте запросы, если в цикле идет запрос к элементу ИБ, к примеру, то уже думайте над минимизацией. Да, это займет больше времени, но и вам скажут спасибо.

Нельзя:

```
foreach($arResult["ORDERS"] as $key => $val)
{
    foreach($val["BASKET_ITEMS"] as $vvval)
    {
        $rsEls = CIBlockElement::GetByID();
    }
}
```

Следует:

```
foreach($arResult["ORDERS"] as $key => $val)
{
    foreach($val["BASKET_ITEMS"] as $vvval)
    {
        $arIDs[] = $vvval["PRODUCT_ID"];
    }
}

$rsEls = CIBlockElement::GetList(array(), array("ID" => $arIDs));
....
```

```
foreach($arResult["ORDERS"] as $key => $val)
{
    foreach($val["BASKET_ITEMS"] as $vvval)
    {
        //наполняем данные, налаживая соответствие ID-ков
    }
}
```

*Фактически, вы сводите порой десятки, если не сотни, запросов к одному. Разве это не круто?*

## Работа с пользовательскими свойствами инфоблоков

Примеры решения задач, возникающих при работе с элементами, разделами и свойствами инфоблоков.

**Задача 1:** Получить значения всех свойств элемента, зная его ID.

```
1      <? $db_props = CIBlockElement::GetProperty(IBLOCK_ID, ELEMENT_ID, "sort", "asc",
array());?
2      $PROPS = array();
3      while($ar_props = $db_props->Fetch())
4          $PROPS[$ar_props['CODE']] = $ar_props['VALUE'];?>
```

Теперь символьный код свойства является ключом ассоциативного массива **\$PROPS**, то есть, если вам нужно значение свойства с кодом **price**, то оно будет хранится в **\$PROPS['price']**.

**Задача 2:** Получить свойства элементов, используя метод [CIBlockElement::GetList](#)

```
1      <?     $arSelect      =      array("ID",      "NAME",      "PROPERTY_prop_code_1",
"PROPERTY_prop_code_2");
2      $res = CIBlockElement::GetList(array(), array(), false, array(), $arSelect);?>
```

Дальше использовать цикл и получить свойства с символьными кодами **prop\_code\_1** и **prop\_code\_2**.

### Советы веб-разработчиков.

**Антон Долганин:** Если для какого-либо изменения в БД предусмотрен специальный метод, следует использовать именно его, а не более общий метод изменения БД.

Хороший пример: модуль интернет-магазина и работа с заказом. Можно изменить флаг оплаты заказа путем [CSaleOrder::Update](#), а можно путем [CSaleOrder::PayOrder](#). Так вот, следует применять именно **PayOrder**, потому что в нем произойдет вызов соответствующих обработчиков.

**Задача 3:** Добавить свойство типа **TEXT/html** для элемента.



Если свойство не множественное:

```
01  <? $element = new CIBlockElement;
02  $PROP = array();
03  $PROP['символьный код свойства']['VALUE']['TYPE'] = 'text'; // или html
04  $PROP['символьный код свойства']['VALUE']['TEXT'] = 'значение, которое нужно
забыть';
05  $arLoadArray = array(
06      "IBLOCK_ID" => IBLOCK_ID,
07      "PROPERTY_VALUES"=> $PROP,
08      "NAME"      => "Название элемента"
09  );
10  $element->Add($arLoadArray);?>
```

Если свойство множественное:

```
01  <? // В $ITEMS хранятся значения множественного свойства, которое нужно
забыть
02  foreach($ITEMS as $item)
03  {
04      $VALUES[]['VALUE']['TYPE']= 'text'; // или html
05      $VALUES[]['VALUE']['TEXT']= $item;
06  }
07  $element = new CIBlockElement;
08  $PROPS = array();
09  $PROPS['символьный код свойства'] = $VALUES;
10  $arLoadArray = array(
11      "IBLOCK_ID" => IBLOCK_ID,
12      "PROPERTY_VALUES"=> $PROPS,
13      "NAME"      => "Название элемента"
14  );
15  $element->Add($arLoadArray);?>
```

**Задача 4:** Заполнить множественное свойство типа **Файл**. Довольно часто при добавлении элемента в инфоблок может понадобиться привязать к нему несколько файлов. Для этого удобно создать у инфоблока множественное свойство типа **Файл** и хранить файлы в нём. Пример заполнения свойства:

```
01  <?
02  $arFiles = array();
```



```
03  for($i = 1; $i < 10; $i++)
04  {
05      if(file_exists($_SERVER['DOCUMENT_ROOT'].'/images/image_.'.$i.'.jpg'))
06      {
07          $arFiles[] = array('VALUE' =>
08              CFile::MakeFileArray($_SERVER["DOCUMENT_ROOT"].'/images/image_.'.$i.'.jpg'),
09              'DESCRIPTION' => '');
10      }
11  ?>
```

После этого массив **\$arFiles** передается как значение свойства при добавлении элемента.

**Задача 5:** Заполнить множественное свойство типа **Список** с отображением в виде флагжков. В данном случае у каждого элемента списка значений есть свой **ID**. Посмотреть их можно, зайдя в детальное редактирование свойства. Заполняется свойство следующим образом:

```
1  <?
2      if($first_condition == true) $values[] = array('VALUE' => 1);
3      if($second_condition == true) $values[] = array('VALUE' => 2);
4      CIBlockElement::SetPropertyValuesEx($ELEMENT_ID, $IBLOCK_ID, array('property_code' => $values));
5  ?>
```

В данном случае при выполнении первого и второго условий мы отмечаем флагжками элементы списка с **ID =1** и **ID=2** соответственно. Заменить следует **\$ELEMENT\_ID**, **\$IBLOCK\_ID** и **property\_code** на нужные значения.

**Задача 6:** Получить пользовательское свойство раздела

```
1  <? $section_props = CIBlockSection::GetList(array(), array('IBLOCK_ID' => IBLOCK_ID, 'ID' => SECTION_ID), true, array("UF_ADDITIONAL_PRICE"));
2  $props_array = $section_props->GetNext(); ?>
```

Теперь в **\$props\_array['UF\_ADDITIONAL\_PRICE']** лежит значение свойства **UF\_ADDITIONAL\_PRICE** раздела инфоблока.

**Совет от веб-разработчиков.**



**Алексей Коваленко:** При работе с инфоблоками удобнее все коды свойств именовать заглавными буквами. В таком случае вы сможете избежать небольших несостыковок в своей работе.

Например, значение свойства с кодом `foto` при работе с компонентами часто доступно через `[PROPERTIES][foto][VALUE]`?, а при использовании метода `GetList` вы можете получить `PROPERTY_FOTO_VALUE`.

### Пример работы с пользовательскими свойствами

В качестве значения свойства попробуем завести картинку с превью. Это могут быть например фотографии гостиницы на туристическом сайте или что-то подобное. В варианте такого применения и рассмотрим решение задачи.

Один из вариантов реализации: хранить изображения в отдельном инфоблоке и показывать как привязку к элементу. Пример кода:

```
AddEventHandler("iblock", "OnIBlockPropertyBuildList", array("CIBlockPropertyPicture", "GetUserTypeDescription"));

AddEventHandler("iblock", "OnBeforeIBlockElementDelete", array("CIBlockPropertyPicture", "OnBeforeIBlockElementDelete"));

class CIBlockPropertyPicture
{
    function GetUserTypeDescription()
    {
        return array(
            "PROPERTY_TYPE" =>"E",
            "USER_TYPE" =>"Picture",
            "DESCRIPTION" =>"Картина",
            "GetPropertyFieldHtml" =>array("CIBlockPropertyPicture", "GetPropertyFieldHtml"),
            "GetPublicViewHTML" =>array("CIBlockPropertyPicture", "GetPublicViewHTML"),
            "ConvertToDB" =>array("CIBlockPropertyPicture", "ConvertToDB"),

            // "GetPublicEditHTML" =>array("CIBlockPropertyPicture", "GetPublicEditHTML"),
            // "GetAdminListViewHTML" =>array("CIBlockPropertyPicture", "GetAdminListViewHTML"),
            // "CheckFields" =>array("CIBlockPropertyPicture", "CheckFields"),
            // "ConvertFromDB" =>array("CIBlockPropertyPicture", "ConvertFromDB"),
            // "GetLength" =>array("CIBlockPropertyPicture", "GetLength"),
        );
}
```



```
}

function GetPropertyFieldHtml($arProperty, $value, $strHTMLControlName)
{
    $LINK_IBLOCK_ID = intval($arProperty["LINK_IBLOCK_ID"]);
    if($LINK_IBLOCK_ID)
    {
        $ELEMENT_ID = intval($value["VALUE"]);
        if($ELEMENT_ID)
        {
            $rsElement      = CIBlockElement::GetList(array(), array("IBLOCK_ID" =>
$arProperty["LINK_IBLOCK_ID"], "ID" => $value["VALUE"]), false, false, array("ID",
"PREVIEW_PICTURE", "DETAIL_PICTURE"));
            $arElement = $rsElement->Fetch();
            if(is_array($arElement))
                $file_id = $arElement["DETAIL_PICTURE"];
            else
                $file_id = 0;
        }
        else
        {
            $file_id = 0;
        }

        if($file_id)
        {
            $db_img = CFile::GetByID($file_id);
            $db_img_arr = $db_img->Fetch();
            if($db_img_arr)
            {
                $strImageStorePath = COption::GetOptionString("main", "upload_dir", "upload");
                $sImagePath
                = "/". $strImageStorePath ."/". $db_img_arr["SUBDIR"]. "/". $db_img_arr["FILE_NAME"];

                return ' Удалить файл '.$sImagePath."
                .";
            }
        }
    }
}
```

```

        return "";
    }
    else
    {
        return "Ошибка настройки свойства. Укажите инфоблок в котором будут
храниться картинки.";
    }
}

function GetPublicViewHTML($arProperty, $value, $strHTMLControlName)
{
    $LINK_IBLOCK_ID = intval($arProperty["LINK_IBLOCK_ID"]);
    if($LINK_IBLOCK_ID)
    {
        $ELEMENT_ID = intval($value["VALUE"]);
        if($ELEMENT_ID)
        {
            $rsElement = CIBlockElement::GetList(array(), array("IBLOCK_ID" =>
$arProperty["LINK_IBLOCK_ID"], "ID" => $value["VALUE"]), false, false, array("ID",
"PREVIEW_PICTURE", "DETAIL_PICTURE"));
            $arElement = $rsElement->Fetch();
            if(is_array($arElement))
                return CFile::Show2Images($arElement["PREVIEW_PICTURE"],
$arElement["DETAIL_PICTURE"]);
        }
    }
    return "";
}

function ConvertToDB($arProperty, $value)
{
    $arResult = array("VALUE" => "", "DESCRIPTION" => "");
    $LINK_IBLOCK_ID = intval($arProperty["LINK_IBLOCK_ID"]);
    if($LINK_IBLOCK_ID)
    {
        if(
            is_array($value["VALUE"])
            && is_array($value["VALUE"]["error"])
        )
    }
}

```



```
&& $value["VALUE"]["error"]["VALUE"] == 0
&& $value["VALUE"]["size"]["VALUE"] > 0
)
{
$arDetailPicture = array(
    "name" => $value["VALUE"]["name"]["VALUE"],
    "type" => $value["VALUE"]["type"]["VALUE"],
    "tmp_name" => $value["VALUE"]["tmp_name"]["VALUE"],
    "error" => $value["VALUE"]["error"]["VALUE"],
    "size" => $value["VALUE"]["size"]["VALUE"],
);
$obElement = new CIBlockElement;
$arResult["VALUE"] = $obElement->Add(array(
    "IBLOCK_ID" => $LINK_IBLOCK_ID,
    "NAME" => $arDetailPicture["name"],
    "DETAIL_PICTURE" => $arDetailPicture,
), false, false, true);
}
elseif(
    is_array($value["VALUE"])
    && isset($value["VALUE"]["size"])
    && !is_array($value["VALUE"]["size"])
    && $value["VALUE"]["size"] > 0
)
{
$arDetailPicture = array(
    "name" => $value["VALUE"]["name"],
    "type" => $value["VALUE"]["type"],
    "tmp_name" => $value["VALUE"]["tmp_name"],
    "error" => intval($value["VALUE"]["error"]),
    "size" => $value["VALUE"]["size"],
);
$obElement = new CIBlockElement;
$arResult["VALUE"] = $obElement->Add(array(
    "IBLOCK_ID" => $LINK_IBLOCK_ID,
    "NAME" => $arDetailPicture["name"],
    "DETAIL_PICTURE" => $arDetailPicture,
```



```
        ), false, false, true);
    }

    elseif($value["VALUE"]["del"])
    {
        $obElement = new CIBlockElement;
        $obElement->Delete($value["VALUE"]["old"]);
    }

    elseif($value["VALUE"]["old"])
    {
        $arResult["VALUE"] = $value["VALUE"]["old"];
    }

    elseif(!is_array($value["VALUE"]) && intval($value["VALUE"]))
    {
        $arResult["VALUE"] = $value["VALUE"];
    }

    return $arResult;
}

function OnBeforeIBlockElementDelete($ELEMENT_ID)
{
    $arProperties = array();
    $rsElement = CIBlockElement::GetList(array(), array("ID" => $ELEMENT_ID), false, false,
array("ID", "IBLOCK_ID"));
    $arElement = $rsElement->Fetch();
    if($arElement)
    {
        $rsProperties = CIBlockProperty::GetList(array(), array("IBLOCK_ID" =>
$arElement["IBLOCK_ID"], "USER_TYPE" => "Picture"));
        while($arProperty = $rsProperties->Fetch())
            $arProperties[] = $arProperty;
    }

    $arElements = array();
    foreach($arProperties as $arProperty)
    {
```



```
$rsPropValues = CIBlockElement::GetProperty($arElement["IBLOCK_ID"], $arElement["ID"],  
array(  
    "EMPTY" => "N",  
    "ID" => $arProperty["ID"],  
));  
while($arPropValue = $rsPropValues->Fetch())  
{  
    $ID = intval($arPropValue["VALUE"]);  
    if($ID > 0)  
        $arElements[$ID] = $ID;  
}  
}  
  
foreach($arElements as $to_delete)  
{  
    CIBlockElement::Delete($to_delete);  
}  
}  
}
```

Что мы в итоге имеем:

- Интерфейс редактирования элемента с возможностью добавления и удаления изображений.
- При удалении элемента связанная с ним информация удаляется.
- Поддержка компонент публичной части.

#### Инструкция по применению:

- Этот код разместите в файле /bitrix/php\_interface/init.php.
- Создайте инфоблок для хранения изображений и в его настройках укажите параметры генерации картинки предварительного просмотра из детальной (на вкладке **Поля**).
- В инфоблоке **Гостиницы** добавьте свойство типа **Картинка** и в дополнительных настройках этого свойства укажите созданный на первом шаге инфоблок. Не забудьте указать символьный код свойства.
- Создайте элемент и "поиграйтесь" со значениями этого свойства.
- В публичной части, например в компоненте news, в параметрах настройки списка элементов выбрать это свойство.



## Использование пользовательских свойств на примере дополнительных полей в подписке

**Задача:** подписчиком будет указываться пол, и в зависимости от этого выбора в письме рассылки она или он получит "Уважаемая" или "Уважаемый" в качестве обращения.

В решении используется демо дистрибутив с настроенным функционалом рассылки. Для использования описанного функционала на ваших проектах нужно провести работы:

- Размещение компонента **Форма подписки** (**bitrix:subscribe.form**) в шаблоне сайта.
- Настройка компонента на страницу редактирования подписки пользователя.
- Создание, при необходимости, рубрик подписки.
- Создать раздел управления подписками.
- Размещение компонента **Страница рассылок** (**bitrix:subscribe.index**) и настройка ее на страницу редактирования подписки пользователя
- Создание страницы редактирования подписки пользователя и размещение на ней компонента **Страница редактирования подписки** (**bitrix:subscribe.edit**).
- [Настройка](#) модуля **Подписка**

**Решение.** Зададим идентификатор сущности к которой будут привязываться значения дополнительных свойств: **MY\_SUBSCRIPTION**. В качестве уникального идентификатора объектов этой сущности будут выступать **b\_subscription.ID**.

На странице [Пользовательские поля](#) ([Настройки > Настройки продукта > Пользовательские поля](#)) откроем [форму создания](#) нового поля.

Заполните поля:

- Тип данных - Список
- Объект - **MY\_SUBSCRIPTION**
- Код поля - **UF\_GENDER**

Остальные поля не заполняем, нажимаем кнопку **Применить**.

На вкладке **Список** задаем возможные значения: **Женский** и **Мужской**. Применяем внесенные изменения.

### **Кастомизация компонента subscribe.edit**

После копирования компонента в свое пространство имён заменяем вызов на странице `/personal/subscribe/subscr_edit.php` на путь к копированному компоненту.



Для вывода значения пользовательских свойств подписки в файле **component.php** после

```
$arResult["ALLOW_REGISTER"] = $bAllowRegister?"Y":"N";
добавляем чтение значений из базы данных
$arResult["USER_PROPERTIES"] = $GLOBALS["USER_FIELD_MANAGER"]->GetUserFields(
    "MY_SUBSCRIPTION",
    $arResult["ID"],
    LANGUAGE_ID
);
```

В файле **setting.php** шаблона выводим примерно следующее:

```
<table>
<?foreach ($arResult["USER_PROPERTIES"] as $FIELD_NAME => $arUserField):?>
    <tr>
        <td><?echo $arUserField["EDIT_FORM_LABEL"]?>:</td>
        <td><?$APPLICATION->IncludeComponent(
            "bitrix:system.field.edit",
            $arUserField["USER_TYPE"]["USER_TYPE_ID"],
            array(
                "bVarsFromForm" => false,
                "arUserField" => $arUserField
            ),
            null,
            array("HIDE_ICONS"=>"Y"));?></td>
    </tr>
<?endforeach;?>
</table>
```

Для сохранения значений в базе данных в файле **component.php** после строк

```
if($ID>0)
{
    ...
    ...
    $res = $obSubscription->Update($ID, $arFields);
    ...
}
else
{
```

```
...
$ID = $obSubscription->Add($arFields);
...
}
```

добавляем код установки значений свойств

```
if($res && $ID > 0)
{
    global $USER_FIELD_MANAGER;
    $arUserFields = $USER_FIELD_MANAGER-> GetUserFields("MY_SUBSCRIPTION");
    $arFields = array();
    foreach($arUserFields as $FIELD_ID => $arField)
        $arFields[$FIELD_ID] = $_REQUEST[$FIELD_ID];
    $USER_FIELD_MANAGER->Update("MY_SUBSCRIPTION", $ID, $arFields);
}
```

Для полноты действий у данного поля в административной части указываем:

- Обязательное - Да
- Подпись - Пол

[Создайте](#) новую подписку (или отредактируйте уже существующую) и укажите пол подписчика.

Теперь надо сделать так, чтобы `#GENDER_HELLO#` будет заменяться на **Уважаемая/Уважаемый** в зависимости от пола. Создаем обработчик события [BeforePostingSendMail](#):

```
<?
// файл /bitrix/php_interface/init.php
// регистрируем обработчик
AddEventHandler("subscribe", "BeforePostingSendMail", Array("MyClass",
"BeforePostingSendMailHandler"));
class MyClass
{
    // создаем обработчик события "BeforePostingSendMail"
    function BeforePostingSendMailHandler($arFields)
    {
        $rs = CSubscription::GetByEmail($arFields["EMAIL"]);
        if($ar = $rs->Fetch())
            $ar["GENDER"] = "#GENDER_HELLO#";
        $arFields["GENDER"] = $ar["GENDER"];
    }
}
```



```
{  
    global $USER_FIELD_MANAGER;  
    $arUserFields = $USER_FIELD_MANAGER->GetUserFields("MY_SUBSCRIPTION", $ar["ID"]);  
  
    if($arUserFields["UF_GENDER"]["VALUE"] == 1)  
        $arFields["BODY"] = str_replace("#GENDER_HELLO#",  
"Уважаемая", $arFields["BODY"]);  
    elseif($arUserFields["UF_GENDER"]["VALUE"] == 2)  
        $arFields["BODY"] = str_replace("#GENDER_HELLO#",  
"Уважаемый", $arFields["BODY"]);  
    else  
        $arFields["BODY"] = str_replace("#GENDER_HELLO#", "",  
$arFields["BODY"]);  
    }  
    else  
    {  
        $arFields["BODY"] = str_replace("#GENDER_HELLO#", "",  
$arFields["BODY"]);  
    }  
    return $arFields;  
}  
}  
?>
```

## Выборка из нескольких инфоблоков

Пример реализации выборки из нескольких инфоблоков с постраничной навигацией и сортировкой.

### Задача

- Выбрать список элементов из нескольких инфоблоков в таблицу;
- Иметь возможность сортировки элементов;
- Должна работать постраничная навигация

### Решение

Этот пример можно использовать при небольшом количестве выбираемых элементов (максимум до 100), естественно с кэшированием.



Собираем все данные в ассоциативный массив, например вот такой:

```
[ITEAM] => Array
(
    [0] => Array
    (
        [CITY_NAME] => value
        [CITY_DETAIL_URL] => value
        [OBJECT_NAME] => value
        [OBJECT_ID] => 2487
        [DATE_CREATE] => 02.07.2006
        [STATUS] => Y
        [PAID_STATUS] => Y
        [DATEIL_OBJECT_URL] => value
    )

    [1] => Array
    (
        [CITY_NAME] => value
        [CITY_DETAIL_URL] => value
        [OBJECT_NAME] => value
        [OBJECT_ID] => 2489
        [DATE_CREATE] => 02.07.2006
        [STATUS] => Y
        [PAID_STATUS] => N
        [DATEIL_OBJECT_URL] => value
    )
)
```

Теперь нужно отсортировать массив `$arResult['ITEAM']`, для это пишем класс:

```
class CCabinet_SortObject {

    function __cmp_ValueOf($a, $b, $name, $order) {
        if(is_set($a[$name]) && is_set($b[$name])) {
            if($order == 'ASC')
                return ($a[$name]<$b[$name])?true:false;
            elseif($order == 'DESC')
                return ($b[$name]>$a[$name])?false:true;
        }
    }
}
```



```
        }

    }

    function cmp_STATUS_ASC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "STATUS", "ASC");
    }

    function cmp_STATUS_DESC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "STATUS", "DESC");
    }

    function cmp_NAME_ASC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "OBJECT_NAME", "ASC");
    }

    function cmp_NAME_DESC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "OBJECT_NAME", "DESC");
    }

    function cmp_CITY_ASC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "CITY_NAME", "ASC");
    }

    function cmp_CITY_DESC($a, $b) {
        return CCabinet_SortObject::__cmp_ValueOf($a, $b, "CITY_NAME", "DESC");
    }

    function cmp_DATE_DESC($a, $b) {
        if ($a["DATE_CREATE"] == $b["DATE_CREATE"]) {
            return 0;
        }
        return ($a["DATE_CREATE"] > $b["DATE_CREATE"]) ? -1 : 1;
    }

    function cmp_DATE_ASC($a, $b) {
        if ($a["DATE_CREATE"] == $b["DATE_CREATE"]) {
            return 0;
```



```
        }
        return ($a["DATE_CREATE"] < $b["DATE_CREATE"]) ? -1 : 1;
    }
}
```

Вот пример применения класса:

```
usort($arResult['ITEAM'],
array("CCabinet_SortObject",
"cmp_". $arParams['SORT_BY']. "_" . $arParams['SORT_ORDER']));
```

После чего нам нужно разбить массив постранично используя API:

```
$rs_ObjectList = new CDBResult;
$rs_ObjectList->InitFromArray($arResult['ITEAM']);
$rs_ObjectList->NavStart(10, false);
$arResult["NAV_STRING"] = $rs_ObjectList->GetPageNavString("", 'komka.cabinet');
$arResult["PAGE_START"] = $rs_ObjectList->SelectedRowsCount() - ($rs_ObjectList->NavPageNomer - 1) * $rs_ObjectList->NavPageSize;
while($ar_Field = $rs_ObjectList->Fetch())
{
$arResult['_ITEAM'][] = $ar_Field;
}
```

## Копирование инфоблока

В **Bitrix Framework** штатно не предусмотрена возможность копирования инфоблоков. Иногда возникает такая потребность и ее можно решить. Автоматизация этого процесса и будет примером использования API инфоблоков.

## Использование импорта XML

Копирование инфоблоков можно осуществить через функцию импорта/экспорта XML:

- Выгрузите необходимый инфоблок экспортом в XML.
- Откройте для редактирования файл XML и аккуратно в нужных местах скорректируйте ID инфоблока. В начале XML можно поменять узел **ИД** и узел **Наименование**:

```
<?xml version="1.0" encoding="UTF-8"?>
<КоммерческаяИнформация ВерсияСхемы="2.021" ДатаФормирования="2010-03-20T09:55:13">
    <Классификатор>
```



```
<Ид>2
<Наименование>ноутбуки
<Свойства>
  <Свойство>
    <Ид>CML2_CODE
    <Наименование>Символьный код
```

после описания инфоблока и его свойств найдите код:

```
<Каталог>
<Ид>2
<ИдКлассификатора>2
<Наименование>ноутбуки
```

установите данные в соответствии с внесенными изменениями выше в узлах ИД, ИД классификатора и Наименование.

### Автоматизация копирования

Есть небольшой инструмент для импорта метаданных с ранее созданного информационного блока при генерации нового:

**Главная страница**

**Копируем мета данные ИБ в новый ИБ**

Копируем ИБ:  
Клиенты [1] ▾

Копирую ИБ в тип:  
▾

Копирую в новый ИБ свойства другого ИБ: \*  
▾

Клиенты [1]  
new\_lists [2]

\* если значение не указано мета данные ИБ секции "Свойства" берутся из ИБ первого поля

копирую

Настройка копирования метаданных задается тремя полями:

- Копирую ИБ.** Поле обязательно для заполнения и всегда предустановлено. В данной секции указывается с какого ИБ будут импортироваться метаданные (за исключением описания свойств).
- Копирую в новый ИБ свойства другого ИБ.** Поле не обязательное, может быть использовано для импорта в новый информационный блок только метаданных свойств любого инфоблока. В случае если поле не заполнено, метаданные свойств берутся из инфоблока указанного в поле **Копирую ИБ**.



- **Копируем ИБ в тип.** Поле не обязательное и может быть указано, в случае если новый информационный блок необходимо сгенерировать в каком либо типе ИБ. Если настройка не указана, используется тип инфоблока, указанного в поле **Копируем ИБ**.

Новый инфоблок после копирования будет иметь имя старого с суффиксом **\_new**.

Код скрипта:

```
CModule::IncludeModule("iblock");
if(intval($_REQUEST["IBLOCK_ID_FIELDS"])>0){
    $bError = false;
    $IBLOCK_ID = intval($_REQUEST["IBLOCK_ID_FIELDS"]);
    $ib = new CIBlock;
    $arFields = CIBlock::GetArrayByID($IBLOCK_ID);
    $arFields["GROUP_ID"] = CIBlock::GetGroupPermissions($IBLOCK_ID);
    $arFields["NAME"] = $arFields["NAME"]."_new";
    unset($arFields["ID"]);
    if($_REQUEST["IBLOCK_TYPE_ID"]!="empty")
        $arFields["IBLOCK_TYPE_ID"]=$_REQUEST["IBLOCK_TYPE_ID"];
    $ID = $ib->Add($arFields);
    if(intval($ID)<=0)
        $bError = true;
    if($_REQUEST["IBLOCK_ID_PROPS"]!="empty")
        $iblock_prop=intval($_REQUEST["IBLOCK_ID_PROPS"]);
    else
        $iblock_prop=$IBLOCK_ID;

    $iblock_prop_new = $ID;
    $ibp = new CIBlockProperty;
    $properties      =      CIBlockProperty::GetList(Array("sort"=>"asc",      "name"=>"asc"),
Array("ACTIVE"=>"Y", "IBLOCK_ID"=>$iblock_prop));
    while ($prop_fields = $properties->GetNext()){
        if($prop_fields["PROPERTY_TYPE"] == "L"){
            $propertyEnums      =      CIBlockPropertyEnum::GetList(Array("DEF"=>"DESC",
"SORT"=>"ASC"), Array("IBLOCK_ID"=>$iblock_prop, "CODE"=>$prop_fields["CODE"]));
            while($enum_fields = $propertyEnums->GetNext()){
                $prop_fields["VALUES"][] = Array(
                    "VALUE" => $enum_fields["VALUE"],
                    "DEF" => $enum_fields["DEF"],
```



```
"SORT" => $enum_fields["SORT"]
);
}
}
$pProp_fields["IBLOCK_ID"] = $iblock_prop_new;
unset($prop_fields["ID"]);
foreach($prop_fields as $k=>$v){
    if(!is_array($v))$prop_fields[$k]=trim($v);
    if($k{0}=='~') unset($prop_fields[$k]);
}
$pPropID = $ibp->Add($prop_fields);
if(intval($PropID)<=0)
    $bError = true;
}
if(!$bError && $IBLOCK_ID>0)
    LocalRedirect($APPLICATION-
>GetCurPageParam("success=Y",array("success","IBLOCK_ID_FIELDS")));
else
    LocalRedirect($APPLICATION-
>GetCurPageParam("error=Y",array("success","IBLOCK_ID_FIELDS")));
}
$str .='
Начало формы





```



```
$str .= '[td]Копируем в новый ИБ свойства другого ИБ: *';
';
$str .= '';
$str .= '/td][/tr]';
$str .= '[tr][td]Копируем ИБ в тип:';
';
$str .= '';
}
$str .= '/td][/tr]';
$str .= '[tr][td][/td][/tr]';
$str .= '/table]
Конец формы
';
echo $str;
```

Скрипт может оказать неоценимую помощь например при копировании ИБ не прибегая к использованию механизмов XML экспорта и XML импорта информационных блоков.

Удобнее всего этот инструмент использовать для инфоблоков в которых много списочных свойств или вообще большое количество свойств требующих детальной настройки.

Скрипт должен быть размещен в корне сайта.

## Инфоблоки в Документообороте

При работе в режиме документооборота создаётся два элемента инфоблока: один "конечный" с пустым **WF\_PARENT\_ELEMENT\_ID** (который мы видим в административной части), второй - "промежуточный" с **WF\_PARENT\_ELEMENT\_ID** равным **ID** только что созданного конечного элемента. "Промежуточный" превращается в "конечный" когда в рамках документооборота он достигнет финального статуса **Опубликован**. Либо любого другого статуса с поднятым флагом **IS\_FINAL**. Этот флаг нельзя выставить методами API **Bitrix Framework**, только правкой БД. Соответственно до этого момента записи в инфоблок заноситься будут, однако для стандартных методов API с параметрами по умолчанию они будут не доступны.

Основным отличием "промежуточных" элементов от "конечных" является поле **WF\_PARENT\_ELEMENT\_ID**, которое пусто для "конечных" элементов и содержит идентификатор "конечного" элемента для промежуточных элементов. В случае создания



нового элемента в документообороте (при условии, что начальный статус документооборота не является финальным) будет создан элемент, в поле **WF\_PARENT\_ELEMENT\_ID** будет записан его собственный идентификатор. При последующем переводе элемента в любой другой статус документооборота, в том числе и в финальный, в инфоблоке будут создаваться новые элементы с полем **WF\_PARENT\_ELEMENT\_ID**.

При переводе элемента в финальный статус элемент, с которого всё началось, будет обновлён так, что поле **WF\_PARENT\_ELEMENT\_ID** станет пустым, а поле **WF\_STATUS\_ID** станет равным значению финального статуса (зачастую это 1). При последующем переводе элемента в какой-либо промежуточный статус круг повториться с той небольшой разницей, что в качестве отправной точки будет использоваться текущий опубликованный элемент.

По умолчанию API позволяет работать только с опубликованными элементами. Причём если перевести опубликованный элемент в какой либо другой статус, то API будет возвращать именно ту версию элемента, которая соответствует опубликованной.

Чтобы получить список всех элементов (в том числе и не опубликованных) у метода [CIBlockElement::GetList](#) в свойствах фильтра необходимо задать параметр "SHOW\_HISTORY" => "Y".

Кроме того, существует возможность: зная **ID** элемента получить его последнюю версию в документообороте с помощью функции [CIBlockElement::WF\\_GetLast](#), и наоборот - зная последнюю версию элемента получить его оригинальный **ID** с помощью функции [CIBlockElement::GetRealElement](#).

Отдельно стоит упомянуть обработчики событий, в том числе [OnAfterIBlockElementUpdate](#). Так как в случае работы с документооборотом непосредственный Update происходит только в случае перевода элемента в финальный статус, не следует ожидать вызова обработчиков этого события при переводе элемента в "промежуточные" статусы. Вместо этого стоит добавить обработчик на событие [OnAfterIBlockElementAdd](#) и ориентироваться на поля "WF" = "Y" (признак того, что элемент участвует в документообороте) и **WF\_PARENT\_ELEMENT\_ID** (идентификатор "конечного" элемента).

Выбрать всю историю изменений элемента можно с помощью функции [CIBlockElement::WF\\_GetHistoryList](#). Для получения детальной информации о промежуточных элементах полученных с помощью этой функции рекомендуется использовать функции [CIBlockElement::GetByID](#) и [CIBlockElement::GetProperty](#).

## Вывод свойств элемента инфоблока

**Задача:** Выбрать свойство(-а) элемента инфоблока и вывести его на экран.

## Решение.

Решение первой части банальна: метод [GetProperty](#) класса [CIBlockElement](#) подробно описаны в документации.

Решение второй части. Возьмём свойство типа **HTML\текст**. Для этого свойства нельзя просто вывести его значение (ключ `VALUE`), т.к. это — массив, содержащий «сырое» значение и его тип (HTML или текст). Всего один вызов метода `GetDisplayValue` класса `CIBlockFormatProperties`:

```
$arResult['DISPLAY_PROPERTIES'][$pid] = CIBlockFormatProperties::GetDisplayValue($arResult, $prop, 'news_out');
```

Теперь в шаблоне мы можем писать так:

```
echo $element['PROPERTY_CODE']['DISPLAY_VALUE'];
```

И любое свойство, тип которого предполагает форматирование значения перед выводом, будет соответствующим образом преобразовано.

## Некоторые ошибки при работе с инфоблоками

### Ошибка типа:

```
Fatal error: Class 'CIBlockElement' not found in /hosting/site.ru/www/index.php on line XX
```

Если используете модуль **Инфоблоки**, его нужно сначала подключить:

```
CModule::IncludeModule("iblock");
```

### Ошибка типа:

```
Fatal error: Call to a member function GetNextElement() on a non-object in /hosting/site.ru/www/index.php on line XX
```

Скорее всего вы передали неверные параметры какому-то методу. Например, так:

```
$res = CIBlockElement::GetList(array(), $arFilter, array(), array(), $arSelect);
```

Третий-то параметр должен быть **true/false**, а не **array**. Читайте внимательно описание используемого метода.

## Глава 7. Компоненты

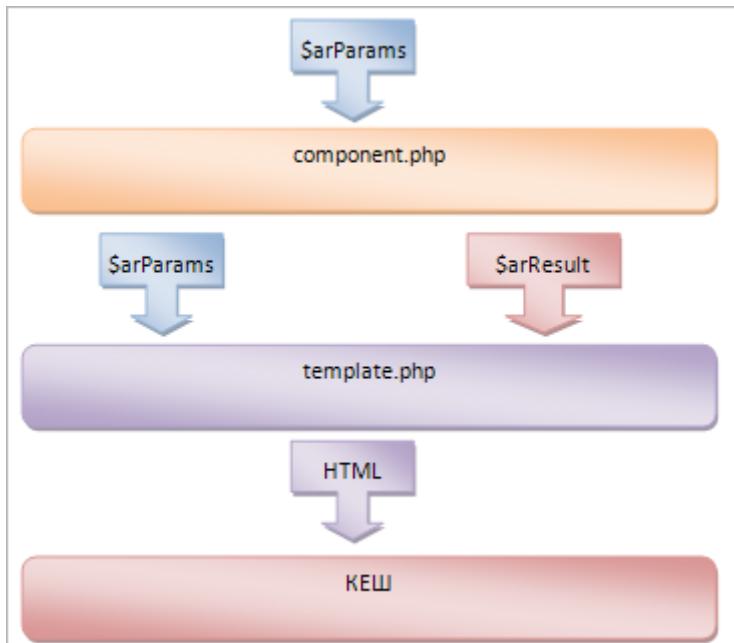
### Цитатник веб-разработчиков.

**Роман Петров:** Начиная работать с 1С-Битрикс, ребята говорили "здесь надо свой компонент", "свое свойство" и т.д. Сейчас познали DAO стандартных компонентов и счастливы.

Компоненты - это основной инструмент разработчика при работе с проектами, созданными на **Bitrix Framework**. От умения владеть этим инструментом во многом зависит профессионализм разработчика.

**Компонент** - это логически завершённый код, предназначенный для извлечения информации из инфоблоков и других источников и преобразования её в HTML-код для отображения в виде фрагментов web-страниц. Состоит из собственно компонента (контроллер) и шаблона (представление). Компонент, с помощью API одного или нескольких модулей, манипулирует данными. Шаблон компонента выводит данные на страницу.

Классическая схема работы компонента:



### Carrier Rider Mapper

Компоненты в полной мере реализуют паттерн проектирования **Carrier Rider Mapper**.



- **Carrier.** Носитель любой информации к которой могут иметь доступ несколько клиентов одновременно.
- **Rider** (Reader либо Writer) - объекты, посредством которых Carrier предоставляет доступ к хранимой в нём информации. Клиенты считывают и записывают информацию хранимую в Carrier исключительно только посредством объектов типа Reader и Writer. Таким образом, Reader и Writer - интерфейсы доступа к информации.
- **Mapper** (Scanner либо Formatter) - объекты обёртки над Reader либо Writer соответственно. Мапперы отвечают за преобразование форматов данных в удобные для клиентов форматы.

Поток информации от носителя к клиенту (считывание): *Carrier -> Reader -> Scanner -> Client.*

Поток информации от клиента к носителю (запись): *Carrier <- Writer <- Formatter <- Client.*

Введение прослойки мапперов между Carrier-Rider и клиентами позволяет соединять один и тот же Carrier-Rider с разными типами клиентов посредством соответствующих (разных) мапперов.

## Использование компонентов

Компоненты используются для:

- создания полнофункциональных разделов на сайте, например новостного раздела, фотогалереи, каталога товаров и т.д. Такие разделы создаются с помощью комплексных компонентов;
- создания часто используемых областей в шаблоне или на страницах сайта (например, формы авторизации, формы подписки);
- представления динамически обновляемой информации (например, ленты новостей, случайного фото);
- выполнения любых других операций с данными.

При размещении компонента на странице пользователь задаёт параметры, с которыми её программный модуль будет вызван на данной конкретной странице. Набор параметров (включая их типы) перечисляется в файле параметров компонента в виде специального хэш-массива.

На странице сайта может быть размещено несколько компонентов. Один компонент может отвечать за вывод собственно текста статьи, другой - за вывод баннеров, третий - за вывод новостей, относящихся к теме данной статьи и т.п. Один и тот же компонент может использоваться на разных страницах сайта и может использоваться на любом из сайтов внутри данной установки продукта.



Компоненты могут быть простыми и комплексными.

### **Основные особенности технологии компонентов**

- В компонентах разделена логика и визуальное представление. Логика - это сам компонент, представление - это шаблон вывода компонента. Для одной логики может быть создано несколько представлений, в том числе зависящих от шаблона текущего сайта. Визуальное представление (шаблон вывода) может быть написано на любом шаблонном языке, который можно подключить из PHP. Например, шаблоны могут быть на PHP, Smarty, XSL и т.д.
- В компонентах нет необходимости изменять логику компонента для изменения особенностей его показа. Поэтому управлять внешним видом информации, выводимой компонентом, стало значительно легче. Шаблон вывода существенно проще, чем компонент в целом.
- Компоненты централизованно хранятся в одной папке. Это обеспечивает большую целостность и понятность структуры сайта. Папка доступна для обращений, а значит компонент и его шаблоны, могут легко подключать свои дополнительные ресурсы.

**⚠ Примечание:** Эволюция развития *Bitrix Framework* через компоненты обычного стандарта привела к действующим сейчас компонентам в стандарте 2.0. Обычный стандарт (компоненты версии 1.0) в данное время не поддерживается. Но в некоторых случаях возможно встретить устаревшие компоненты в проектах, работающих на старых версиях 1С-Битрикс: Управление сайтом. Если встретился такой анахронизм, обратитесь к [документации](#).

### **Простые и комплексные**

Цитатник веб-разработчиков.

**Антон Долганин:** Советую даже опытным спецам посмотреть как сделаны (и которые будут сделаны) решения от самого Битрикс (магазин, инфопортал, к примеру). Встречаются довольно хитрые решения, новый взгляд на обычные компоненты.

Компоненты делятся на **простые** (одностраничные) и **комплексные** (многостраничные). Простой компонент реализует вывод на одной физической странице, доступной под конкретным URL. Комплексный же компонент заменяет собой набор простых компонентов. Например, создание новостного раздела можно реализовать несколькими простыми компонентами, размещаемыми каждый на отдельной физической странице, а можно - одним комплексным, размещенным на одной физической странице.



С точки зрения структуры и способов подключения простые и комплексные компоненты очень похожи. Но с точки зрения функционирования они сильно отличаются.

### Простые компоненты

**Простые** (обычные, одностораничные) компоненты создают какую-либо область на одной странице. Их удобно использовать, когда на одной странице требуется разместить данные из различных модулей (блоги и инфоблоки, например) или данные из разных инфоблоков (новости и каталог товаров). Для создания полного раздела новостей или каталога товаров пользоваться ими довольно неудобно: приходится создавать большое число статических страниц и следить за тем, чтобы они были корректно связаны друг с другом.

### Комплексные компоненты

**Комплексные** (сложные, многостраничные) компоненты создают **разделы сайта**. Например, компонент каталога создает на сайте весь раздел каталога: список каталогов, список групп и страницы товаров. То есть, комплексный компонент состоит из набора динамических страниц при просмотре сайта, но из одной статической страницы на физическом уровне. Комплексные компоненты строятся на основе простых компонентов, используя их логику.

Преимущество комплексных компонентов состоит в автоматической компоновке параметров одностораничных компонентов и отсутствии необходимости их связывать.

**Комплексные компоненты разрешают следующие проблемы:**

- Отпадает необходимость создания большого числа статических страниц для размещения всех требуемых подкомпонентов. Отпадает необходимость отдельно настраивать для каждого из подкомпонентов общие (пересекающиеся) свойства (например, тип инфоблока и инфоблок).
- Происходит установление сложных взаимосвязей между подкомпонентами. Например, нет необходимости для страницы со списком сообщений темы форума настраивать, как эта страница может указать на страницу списка тем форума, а как на страницу профиля посетителя.
- В компонент (даже с кастомизированными шаблонами вывода) можно добавить новую страницу. Например, если на форуме появится страница (подкомпонент) по выводу 10 посетителей с самым высоким рейтингом, то эта страница станет доступной и в публичной части.
- Можно сменить шаблон вывода всего комплексного компонента одним действием, а не настраивать вывод каждого из подкомпонентов.

Алгоритм работы комплексного компонента таков:

1. на основании действий посетителя сайта (например, переход по пунктам меню) комплексный компонент определяет, какая страница должна быть показана

пользователю, и подключает свой шаблон компонента для этой динамической страницы;

2. шаблон страницы подключает обычные компоненты, автоматически настраивая необходимым образом их свойства;
3. обычные компоненты выполняют свою работу: запрашивают данные у ядра системы, форматируют их и выводят посетителю, а также предоставляют пользователю различные элементы управления (ссылки, формы, кнопки и т.п.);
4. пользователь с помощью каких-либо элементов управления, посыпает новый запрос комплексному компоненту.

### **Пример**

Рассмотрим упрощённый пример работы комплексного компонента новостей. Пусть у нас есть обычные компоненты: список новостей и детальной новости (последний принимает во входных параметрах код новости, которую нужно показать).

Раздел новостей можно организовать, например, разместив на странице **index.php** компонент списка новостей, а на странице **news.php** - компонент детальной новости. При этом у компонента списка новостей нужно настроить входные параметры так, чтобы он мог формировать ссылки на страницу детальной новости (с кодом новости), а у компонента детальной новости нужно настроить входные параметры так, чтобы он мог формировать ссылку на страницу списка новостей.

Чтобы задать ссылку на страницу детальной новости, нужно задать путь к этой странице, а так же название параметра, в котором будет передаваться код новости для показа. То же название параметра нужно задать и во входных параметрах компонента детальной новости, чтобы он знал, где брать код новости для показа. Даже в данном максимально упрощённом случае настройки не так просты. А если это набор из десятков компонентов форума?

Более удобной альтернативой для сборщика сайта будет использование комплексного компонента новостей. Этот компонент, например, можно просто установить на страницу **index.php**. Согласованием ссылок и параметров будет заниматься сам комплексный компонент. От разработчика сайта никаких дополнительных действий не потребуется.

Страницы шаблона комплексного компонента будут содержать подключение соответствующих обычных компонентов с правильной настройкой их входных параметров. Обычные компоненты будут выполнять свои обычные функции: им все равно, кто их вызвал и зачем. Для обычных компонентов важна только правильная настройка их входных параметров.

Таким образом реализуется паттерн **MVC**:

- на комплексный компонент новостей (controller) приходит HTTP запрос (действия пользователя);

- комплексный компонент новостей (controller) проверяет, установлен ли через HTTP запрос код новости и подключает из своего шаблона страницу списка новостей или страницу детальной новости (view);
- подключенная страница, в свою очередь, подключает соответствующий обычный компонент, устанавливая при этом его входные параметры соответствующим образом;
- обычный компонент выполняет свою работу: запрашивает данные у ядра (model), форматирует их и выводит посетителю, а также отображает элементы управления (ссылки, формы, кнопки и т.п.);
- пользователь с помощью элементов управления посыпает новый HTTP запрос на комплексный компонент новостей (controller);
- процедура повторяется по мере надобности.

## Размещение компонента в системе и его подключение

Компоненты в **Bitrix Framework** должны храниться только в определенных местах:

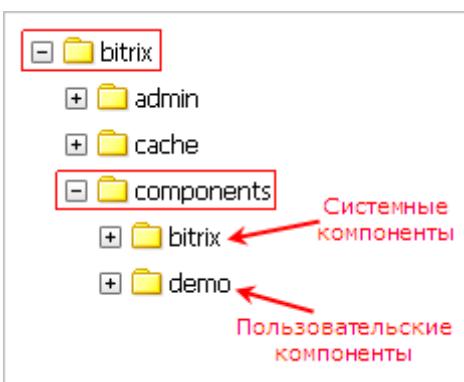
- в папке `/bitrix/components/bitrix` (по умолчанию);
- в папке `/bitrix/components/`собственное пространство имен.

Оптимальным будет размещение компонентов в папке компонентов в собственном пространстве имен.

Все компоненты находятся в папке `/bitrix/components/`. Системные компоненты находятся в папке `/bitrix/components/bitrix/`. Содержимое этой папки обновляется системой обновлений и не может изменяться пользователями.

 **Внимание!** Изменение содержимого папки системных компонентов `/bitrix/components/bitrix/` может привести к непредсказуемым последствиям.

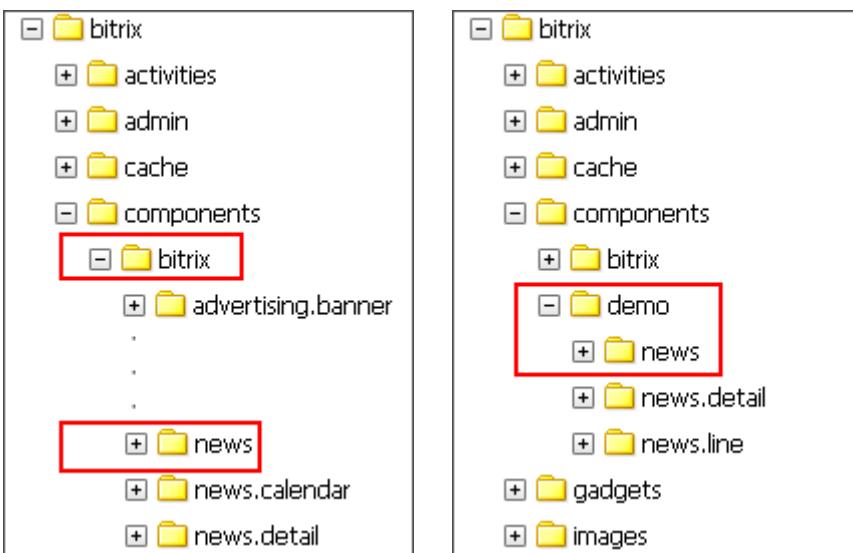
Пользовательские компоненты могут находиться в любых других подпапках папки `/bitrix/components/` или прямо в папке `/bitrix/components/`.



## Именование

Название подпапки директории `/bitrix/components/` образует пространство имен (**namespace**) компонентов. Например, все системные компоненты расположены в пространстве имен **bitrix**. При создании пользовательских компонентов рекомендуется создать какое-либо пространство имен и размещать пользовательские компоненты в нем.

Например, существует системный компонент **news**, который расположен в пространстве имен **bitrix**. Если необходим пользовательский компонент **news**, который имеет другую функциональность, не реализуемую с помощью шаблона системного компонента, то рекомендуется создать некоторое пространство имен (подпапку) в папке `/bitrix/components/` (например, **demo**) и расположить пользовательский компонент **news** в этом пространстве имен. Таким образом будут доступны два компонента **news**, но лежащие в разных пространствах имен: **bitrix:news** и **demo:news**.



Имена компонентов имеют вид **идентификатор1.идентификатор2....**. Например, **catalog**, **catalog.list**, **catalog.section.element** и т.п. Рекомендуется строить имена иерархически, начиная с общего понятия и заканчивая конкретным назначением компонента. Например, компонент, показывающий список товаров данной группы, может называться **catalog.section.elements**. Полное имя компонента - это имя компонента с указанием



пространства имен. Полное имя имеет вид **пространство\_имен:имя\_компоненты**. Например, **bitrix:catalog.list**. Если компонент лежит вне пространства имен, то пространство имен не указывается. Например, **catalog.section**.

### Подключение компонента

В самом общем виде подключение компонента осуществляется следующим образом:

```
<$APPLICATION->IncludeComponent(  
    componentName, // имя компонента  
    componentTemplate, // шаблон компонента, пустая строка если шаблон по умолчанию  
    arParams=array(), // параметры  
    parentComponent=null,  
    arFunctionParams=array()  
>
```

Внутри компонента (файл **component.php**) доступны следующие предопределенные переменные:

- `$componentName` – полное название компонента (например: **bitrix:iblock.list**).
- `$componentTemplate` – шаблон, с которым вызывается компонент.
- `$arParams` – входные параметры компонента (т.е. параметры с которыми вызывается компонент). Параметры доступны так же по их именам.
- `$componentPath` – путь к компоненту относительно корня сайта (пример: `/bitrix/components/bitrix/iblock.list`).
- `$parentComponentName` – название родительского компонента (пустое, если нет родителя).
- `$parentComponentPath` – путь к родительскому компоненту относительно корня сайта (пустой, если нет родителя).
- `$parentComponentTemplate` – шаблон родительского компонента (пустой, если нет родителя).
- `$arResult` — результат, чтение/изменение. Затрагивает одноименный член класса компонента.
- `$this` — естественно ссылка на текущий вызванный компонент (объект класса [CBitrixComponent](#)), можно использовать все методы класса. Кроме того, в компоненте объявлены глобальными переменные `$APPLICATION`, `$USER`, `$DB`.

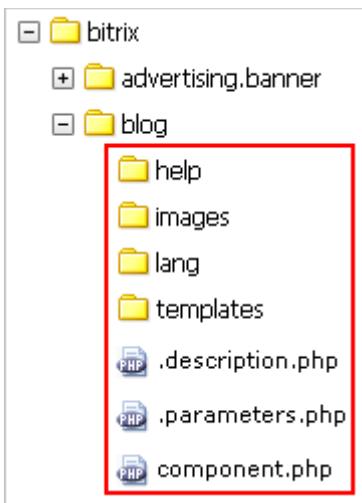


## Структура компонента

Компонент хранит все, что ему нужно для работы, в своей папке. Поэтому их можно легко переносить между проектами.

**⚠ Важно:** файлы компонентов нельзя использовать по отдельности. Компонент - это единое целое, он обладает свойством неделимости.

Папка компонента может содержать следующие подпапки и файлы:



- подпапку **help**, в которой расположены файлы помощи компонента. В ней, в свою очередь, находятся подпапки с именами, равными кодам языков. В каждой из подпапок есть файл **index.php**, который является главным файлом помощи данного компонента для данного языка, и, возможно, другие файлы. Например, путь к файлу помощи для английского языка будет иметь вид `/help/en/index.php` (относительно папки компонента). В файлах помощи должна быть описана структура массива, в котором компонент возвращает данные шаблону. Подпапка **help** может отсутствовать.
- подпапку **install**, в которой расположены скрипты для инсталляции и деинсталляции компонента. Скрипт инсталляции должен называться **install.php**, а скрипт для деинсталляции - **uninstall.php**. Подпапка **install** может отсутствовать.
- подпапку **lang**, в которой расположены файлы языковых сообщений (переводов) компонента.
- подпапку **templates**, в которой расположены шаблоны вывода (отображения) компонента. Подпапка **templates** может отсутствовать, если у компонента нет шаблонов вывода.
- файл **component.php**, который содержит логику (код) компонента. Задача этого файла - сформировать из полученных параметров (`$arParams`) массив `$arResult`, который впоследствии попадет в шаблон компонента. Этот файл должен всегда присутствовать в папке компонента.
- файл **.description.php**, который содержит название, описание компонента и его положение в дереве логического размещения (для редактора). Этот файл должен всегда присутствовать в папке компонента. Его отсутствие не скажется на работе компонента, но размещение компонента через визуальный редактор

станет невозможным.

- файл **.parameters.php**, который содержит описание входных параметров компонента для редактора. Если у компонента есть входные параметры, то этот файл должен присутствовать в папке компонента.
- любые другие папки и файлы с ресурсами, необходимыми компоненту, например, папка **images**.

### Общая структура компонента

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
<?
//Проверяем и инициализируем входящие параметры компонента
if(вывод компонента находится в валидном кеше)
{
//Вывод данных из кеша
}
else
{
//Запрос данных и формирование массива $arResult в соответствии со
//структурой, описанной в файле помощи компонента
$this->IncludeComponentTemplate();
//Кеширование вывода
}
?>
```

 **Примечание:** детально со структурой комплексного компонента можно познакомиться в [документации](#).

### Папки локализаций

Компоненты и их шаблоны поддерживают возможность вывода пользовательских сообщений на различных языках. Так, например, если компонент выводит содержимое инфоблока, это содержимое может понадобиться предварить строковой константой. Например, «Здесь вы видите содержимое инфоблока». Пользователь же может перейти, например, в англоязычную версию сайта - и в этом случае эта константа может быть для него автоматически выведена на английском языке.



Для реализации этой функции при выводе строковых констант выводится не сама константа, а вызывается спецфункция, а в качестве аргумента ей передаётся идентификатор этой константы.

**! Пример:**

*В файле /template.php без локализации:*

```
<?echo "Торговый каталог"?>
```

*В файле /template.php с локализацией:*

```
<?echo GetMessage("CATALOG")?>
```

*В этом случае в файле /lang/ru/template.php пишем:*

```
<?$MESS["CATALOG"] = "Торговый каталог";?>
```

В папке **lang** для каждой языковой версии создаётся папка с названием языка (например, **ru**, **en** и т.п.), в которой и размещаются файлы языковых сообщений. Рекомендуется называть файл языковых сообщений для какого-либо файла компонента так же, как называется этот файл. В этих файлах находятся массивы, ключами для которых являются идентификаторы констант, а значениями - сами константы, переведенными на соответствующий язык.

Рекомендуется располагать файлы в той же иерархии относительно папки **/lang/код\_языка/**, в которой файл располагается относительно папки компонента. Например, языковой файл с английскими фразами для файла **/install/uninstall.php** рекомендуется располагать по пути **/lang/en/install/uninstall.php**. Подпапка **lang** может отсутствовать, если в компоненте нет зависящих от языка фраз.

Папки локализаций создаются отдельно для компонентов и для каждого из шаблонов.

### Как поменять стандартные надписи в интерфейсе.

Скопировать шаблон компонента для того, чтобы его можно было кастомизировать (если он еще не кастомизирован). Затем либо:

- С помощью модуля **Перевод**: Настройки > Локализация.
- Либо же вручную редактируете языковые файлы **/bitrix/templates/имя\_шаблона\_сайта/components/имя\_компоненты/templates/имя\_шаблона/lang/ru/template.php**.

### Подключение языкового файла (файла перевода) компонента



Языковые файлы в компоненте и всех его стандартных файлах (**component.php**, **.description.php**, **.parameters.php**) подключаются автоматически. В других файлах компонента языковые файлы можно подключить командой:

```
$this->IncludeComponentLang($relativePath = "", $lang = False)
```

где:

**\$relativePath** - путь к файлу относительно папки компонента,

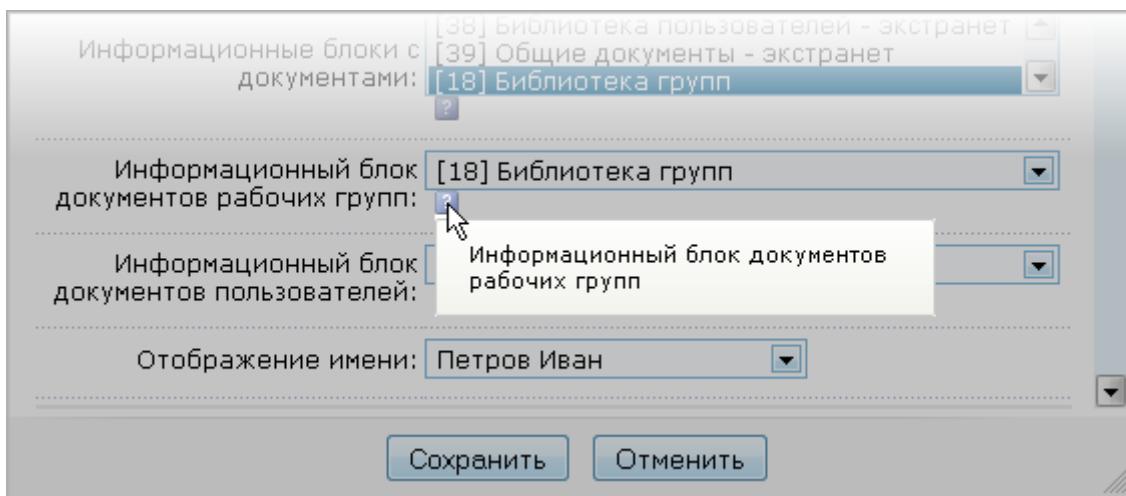
**\$lang** - язык. Если передается **False**, то используется текущий язык.

Пример:

```
$this->IncludeComponentLang("myfile.php");
```

## Папка help

С помощью этой папки можно создать подсказки к параметрам компонента:



Для этого в папке **help** необходимо создать файл с названием **.tooltips.php**. (Языковые файлы для него лежат соответственно в папке (от корневой папки компонента) - **/lang/ID\_языка/help/.tooltips.php**.)

В файле - массив **\$arTooltips**, в котором ключами являются параметры компонента, значениями - вызовы **GetMessage()**. В качестве примера:

```
$arTooltips = array(
    'IBLOCK_TYPE' => GetMessage('VWS_COMP_DVL_TIP_IBLOCK_TYPE'),
    'IBLOCK_ID' => GetMessage('VWS_COMP_DVL_TIP_IBLOCK_ID'),
    'SORT_BY1' => GetMessage('VWS_COMP_DVL_TIP_SORT_BY1'),
    'SORT_ORDER1' => GetMessage('VWS_COMP_DVL_TIP_SORT_ORDER1'),
```

);

Здесь заданы подсказки для параметров компонента **IBLOCK\_TYPE**, **IBLOCK\_ID**, **SORT\_BY1** и **SORT\_ORDER1**.

Ряд стандартных параметров (**CACHE\_TIME**, **AJAX\_MODE** и другие) имеет несколько подсказок. Для **CACHE\_TIME**:

- **CACHE\_TIME** - время кеширования;
- **CACHE\_TYPE** - тип кеширования.

Для страничной адресации:

- **DISPLAY\_TOP\_PAGER** - показывать постраничку над списком;
- **DISPLAY\_BOTTOM\_PAGER** - показывать постраничку под списком;
- **PAGER\_TITLE** - название элементов в постраничке;
- **PAGER\_SHOW\_ALWAYS** - показывать постраничку всегда;
- **PAGER\_TEMPLATE** - имя шаблона постранички;
- **PAGER\_DESC\_NUMBERING** - обратная адресация;
- **PAGER\_DESC\_NUMBERING\_CACHE\_TIME** - время кеширования обратной адресации.

 **Внимание!** Нельзя делать подсказки про запас (создать, но оставить пустыми): перестанут показываться настройки компонента.

## Описание компонента

В файле **.description.php** содержится описание компонента. Это описание применяется для работы с компонентом (например, в визуальном редакторе), а также при работе в режиме редактирования сайта. При работе самого компонента (при обращении к странице, на которой расположен компонент) описание не используется и файл **.description.php** не подключается.

Файл **.description.php** должен находиться в папке компонента. Языковой файл подключается автоматически (должен лежать в папке **/lang/<язык>/description.php** относительно папки компонента).

Структура типичного файла **.description.php** такова:

```
<?
$arComponentDescription = array(
```



```
"NAME" => GetMessage("COMP_NAME"),
"DESCRIPTION" => GetMessage("COMP_DESCR"),
"ICON" => "/images/icon.gif",
"PATH" => array(
    "ID" => "content",
    "CHILD" => array(
        "ID" => "catalog",
        "NAME" => "Каталог товаров"
    )
),
"AREA_BUTTONS" => array(
    array(
        'URL' => "javascript:alert('Это кнопка!!!');",
        'SRC' => '/images/button.jpg',
        'TITLE' => "Это кнопка!"
    ),
),
"CACHE_PATH" => "Y",
"COMPLEX" => "Y"
);
?>
```

Как видно, в файле определяется массив **\$arComponentDescription**, который описывает компонент. Этот массив может иметь следующие ключи:

- "**NAME**" - название компонента;
- "**DESCRIPTION**" - описание компонента;
- "**ICON**" - путь к пиктограмме компонента относительно папки компонента. Значок компонента используется в среде БУС (например, в визуальном редакторе);
- "**PATH**" - расположение компонента в виртуальном дереве компонента в визуальном редакторе. Значением этого элемента должен быть массив, имеющий ключи:
  - "**ID**" - код ветки дерева. ID узла должен быть уникальным в пределах всего дерева компонент (включая стандартные). Если у узлов будут два одинаковых ID, то оба не будут открываться. Например, для компонента собственной разработки выбран узел ID = "news", а такой ID уже есть для стандартных компонентов.
  - "**NAME**" - название ветки дерева. Необходимо обязательно указать. NAME берется из первого попавшегося компонента в узле. Если его не оказалось либо нет нужной языковой константы - в качестве NAME используется ID.



- "CHILD" - дочерняя или подчиненная ветка. В элементе с ключом "CHILD" может быть задана подчиненная ветка дерева с той же структурой, что и родительская ветка.

Дерево ограничено тремя уровнями. Как правило, строится двухуровневое дерево и компоненты располагаются на втором уровне. Следующие служебные названия первого уровня зарезервированы и не могут быть использованы: "content" (контент), "service" (сервисы), "communication" (общение), "e-store" (магазин), "utility" (служебные).

Если ключ "PATH" не задан, то компонент не будет присутствовать в визуальном редакторе;

- "AREA\_BUTTONS" - пользовательские кнопки, которые показываются для компонента в режиме редактирования сайта;
- "CACHE\_PATH" - если значение равно "Y", отображается кнопка очистки кэша компонента в режиме редактирования сайта (предполагается, что кэш лежит по стандартному пути: /<код сайта>/<относительный путь к компоненту>). Если равно не пустой отличной от "Y" строке, отображается кнопка очистки кэша компонента в режиме редактирования сайта (кэш лежит по пути, равному значению с ключом "CACHE\_PATH" - для нестандартных путей);
- "COMPLEX" - элемент должен иметь значение "Y" для комплексного компонента, для простых компонентов не имеет значения.

## Параметры компонента

В файле **.parameters.php** содержится описание входных параметров компонента. Данные файла нужны исключительно для создания формы ввода свойств компонента в среде **Bitrix Framework** (например, в визуальном редакторе). Это описание применяется для работы с компонентом, а также при работе в режиме редактирования сайта. При работе самого компонента (при обращении к странице, на которой расположен компонент) описание не используется и указанный файл не подключается.

Файл **.parameters.php** должен находиться в папке компонента. Языковой файл подключается автоматически (должен лежать в папке **/lang/<язык>/parameters.php**, относительно папки компонента).

В файле определяется массив **\$arComponentParameters**, который описывает входные параметры компонента. Если необходимо, производится выборка каких-либо дополнительных данных. Например, для формирования выпадающего списка типов информационных блоков (входной параметр **IBLOCK\_TYPE\_ID**) выбираются все активные типы.

Структура типичного файла **.parameters.php** такова (на примере компонентов, работающих с модулем **Информационные блоки**):



```
<?

CModule::IncludeModule("iblock");

$dblBlockType = CIBlockType::GetList(
    array("sort" => "asc"),
    array("ACTIVE" => "Y")
);
while ($arIBlockType = $dblBlockType->Fetch())
{
    if ($arIBlockTypeLang = CIBlockType::GetByIDLang($arIBlockType["ID"], LANGUAGE_ID))
        $arIBlockType[$arIBlockType["ID"]] = "[".$arIBlockType["ID"]."]"
        ".$arIBlockTypeLang["NAME"];
}

$arComponentParameters = array(
    "GROUPS" => array(
        "SETTINGS" => array(
            "NAME" => GetMessage("SETTINGS_PHR")
        ),
        "PARAMS" => array(
            "NAME" => GetMessage("PARAMS_PHR")
        ),
    ),
    "PARAMETERS" => array(
        "IBLOCK_TYPE_ID" => array(
            "PARENT" => "SETTINGS",
            "NAME" => GetMessage("INFOBLOCK_TYPE_PHR"),
            "TYPE" => "LIST",
            "ADDITIONAL_VALUES" => "Y",
            "VALUES" => $arIBlockType,
            "REFRESH" => "Y"
        ),
        "BASKET_PAGE_TEMPLATE" => array(
            "PARENT" => "PARAMS",
            "NAME" => GetMessage("BASKET_LINK_PHR"),
            "TYPE" => "STRING",
            "MULTIPLE" => "N",
        )
    )
);
```



```
"DEFAULT" => "/personal/basket.php",
"COLS" => 25
),
"SET_TITLE" => array(),
"CACHE_TIME" => array(),
"VARIABLE_ALIASES" => array(
    "IBLOCK_ID" => array(
        "NAME" => GetMessage("CATALOG_ID_VARIABLE_PHR"),
    ),
    "SECTION_ID" => array(
        "NAME" => GetMessage("SECTION_ID_VARIABLE_PHR"),
    ),
),
"SEF_MODE" => array(
    "list" => array(
        "NAME" => GetMessage("CATALOG_LIST_PATH_TEMPLATE_PHR"),
        "DEFAULT" => "index.php",
        "VARIABLES" => array()
    ),
    "section1" => array(
        "NAME" => GetMessage("SECTION_LIST_PATH_TEMPLATE_PHR"),
        "DEFAULT" => "#IBLOCK_ID#",
        "VARIABLES" => array("IBLOCK_ID")
    ),
    "section2" => array(
        "NAME" => GetMessage("SUB_SECTION_LIST_PATH_TEMPLATE_PHR"),
        "DEFAULT" => "#IBLOCK_ID#/SECTION_ID#",
        "VARIABLES" => array("IBLOCK_ID", "SECTION_ID")
    ),
),
)
);
?>
```

Опишем ключи массива **\$arComponentParameters** подробнее.



## GROUPS

Значением этого ключа является массив групп параметров компонента. Параметры в визуальных средствах среды **Bitrix Framework** (например, в визуальном редакторе) группируются. Группы в среде **Bitrix Framework** располагаются в том порядке, в котором заданы в файле. Массив групп параметров компонента состоит из элементов следующего вида:

```
"код группы" => array(  
    "NAME" => "название группы на текущем языке"  
    "SORT" => "сортировка",  
)
```

Перечень стандартных групп:

- **ADDITIONAL\_SETTINGS** (сортировка - 700). Эта группа появляется, например, при указании параметра **SET\_TITLE**.
- **CACHE\_SETTINGS** (сортировка - 600). Появляется при указании параметра **CACHE\_TIME**.
- **SEF\_MODE** (сортировка 500). Группа для всех параметров, связанных с использованием ЧПУ.
- **URL\_TEMPLATES** (сортировка 400). Шаблоны ссылок
- **VISUAL** (сортировка 300). Редко используемая группа. Сюда предполагается загонять параметры, отвечающие за внешний вид.
- **DATA\_SOURCE** (сортировка 200). Тип и ID инфоблока.
- **BASE** (сортировка 100). Основные параметры.
- **AJAX\_SETTINGS** (сортировка 550). Все, что касается ajax.

## PARAMETERS

Значением этого ключа является массив параметров компонента. В каждой группе параметров параметры располагаются в том порядке, в котором заданы в файле. Массив обычных параметров компонента состоит из элементов следующего вида:

```
"код параметра" => array(  
    "PARENT" => "код группы", // если нет - ставится ADDITIONAL_SETTINGS  
    "NAME" => "название параметра на текущем языке",  
    "TYPE" => "тип элемента управления, в котором будет устанавливаться параметр",  
    "REFRESH" => "перегружать настройки или нет после выбора (N/Y)",  
    "MULTIPLE" => "одиночное/множественное значение (N/Y)",  
    "VALUES" => "массив значений для списка (TYPE = LIST)",  
    "ADDITIONAL_VALUES" => "показывать поле для значений, вводимых вручную (Y/N)",
```



"SIZE" => "число строк для списка (если нужен не выпадающий список)",  
"DEFAULT" => "значение по умолчанию",  
"COLS" => "ширина поля в символах",  
,

Для типа элемента управления **TYPE** есть значения:

- **LIST** - выбор из списка значений. Для типа LIST ключ **VALUES** содержит массив значений следующего вида:

*VALUES => array(*

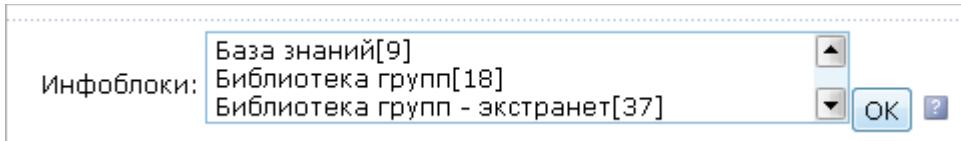
*"ID или код, сохраняемый в настройках компонента" => "языкозависимое описание",*

*),*

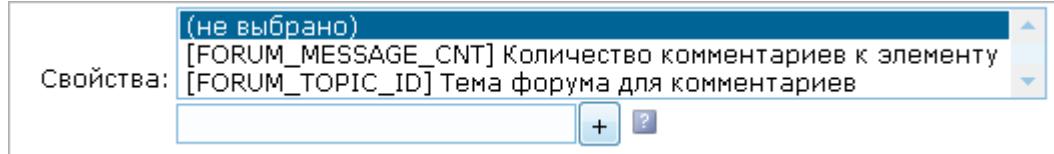
- **STRING** - текстовое поле ввода.
- **CHECKBOX** - да/нет.
- **CUSTOM** - позволяет создавать кастомные элементы управления.

Внешний вид списка меняется в зависимости от наличия/отсутствия ключей **MULTIPLE** и **ADDITIONAL\_VALUES**:

- Если **MULTIPLE** и **ADDITIONAL\_VALUES** отсутствуют или равны "N", то выводится просто список, никаких значений в список не добавляется:



- Если **ADDITIONAL\_VALUES** = "Y", **MULTIPLE** = "N", то в список добавляется значение **другое** и рядом доп.поле для ввода значения вручную:



- Если **ADDITIONAL\_VALUES** = "N", **MULTIPLE** = "Y", то в список ничего не добавляется, просто появляется возможность выбрать несколько элементов:



- Если **ADDITIONAL\_VALUES** = "Y", **MULTIPLE** = "Y", то список добавляется значение **не выбрано** и рядом множественное дополнительное поле для ввода значения вручную.

**⚠ Примечание:** Скриншоты делались для значения **COLS** = 8. Если этот ключ не указывать, список будет выпадающим.

Параметр **REFRESH** позволяет после выбора значения перегрузить всю форму с параметрами. Делается это, например, для выбора инфоблока конкретного типа. То есть имеем два параметра - тип инфоблока и код инфоблока. Исходное положение - в первом списке всех типов инфоблоков, во втором - список всех инфоблоков данного сайта, а после выбора нужного типа инфоблока параметры компонента перегружаются и мы видим только инфоблоки нужного типа.

Внешне для параметров типа **LIST** этот ключ проявляется как кнопка с надписью **OK** возле параметра (см. скриншоты выше).

Если нужно, чтобы некий параметр появлялся или нет в зависимости от другого, делается это так. Пусть нам необходимо показать список свойств инфоблока. Предположим, что ID инфоблока содержится в параметре компонента **IBLOCK\_ID**, а параметр, где будет список свойств назовем **PROP\_LIST**. У параметра **IBLOCK\_ID** должен быть выставлен ключ **REFRESH** = 'Y'. Код:

```
if (0 < intval($arCurrentValues['IBLOCK_ID']))
{
    $arPropList = array();
    $rsProps      = CIBlockProperty::GetList(array(),array('IBLOCK_ID' =>
    $arCurrentValues['IBLOCK_ID']));
    while ($arProp = $rsProps->Fetch())
    {
        $arPropList[$arProp['ID']] = $arProp['NAME'];
    }
    $arComponentParameters['PARAMETERS']['PROP_LIST'] => array(
        'NAME' => 'название параметра',
        'TYPE' => 'LIST'
    )
}
```



```
'VALUES' => $arPropList,  
);  
}
```

Существуют особые параметры, которые стандартизованы и которые нет необходимости описывать полностью. Достаточно указать, что они есть. Например,

```
"SET_TITLE" => array(),  
"CACHE_TIME" => array(),
```

Первый из указанных параметров указывает, следует ли компоненту установить заголовок страницы, а второй - все настройки, связанные с кэшированием.

Только комплексные компоненты могут работать в режиме ЧПУ или переопределять переменные, которые приходят из HTTP запроса. В этом случае необходимо среди параметров указать ещё два особых параметра:

- **"VARIABLE\_ALIASES"** - массив, описывающий переменные, которые компонент может получать из HTTP запроса. Каждый элемент массива имеет вид:

```
"внутреннее название переменной" => array(  
    "NAME" => "название переменной на текущем языке",  
)
```

- **"SEF\_MODE"** - массив, описывающий шаблоны путей в режиме ЧПУ. Каждый элемент массива имеет вид:

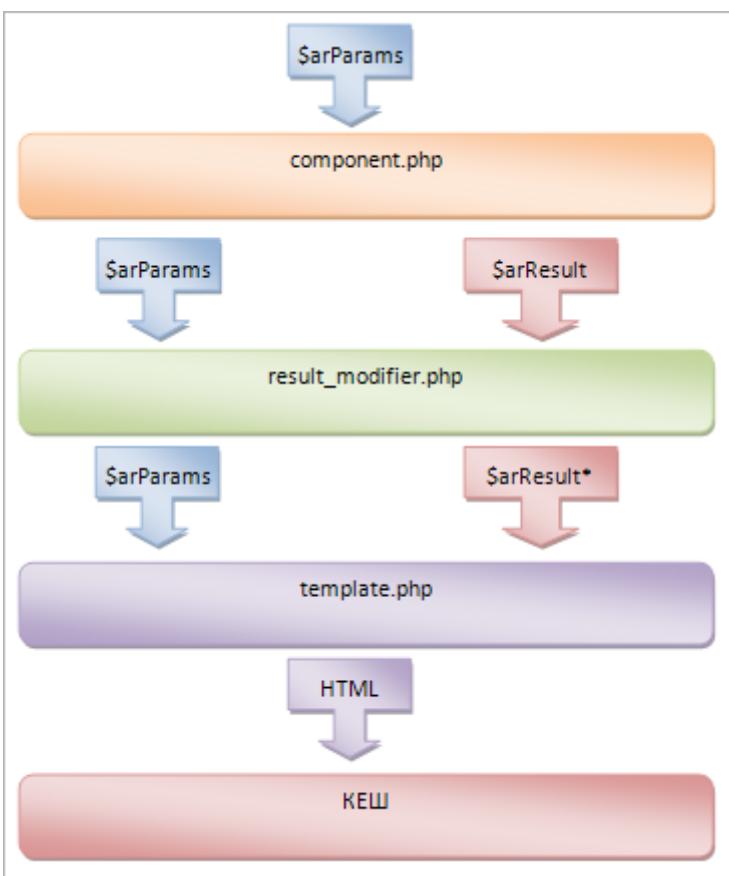
```
"код шаблона пути" => array(  
    "NAME" => "название шаблона пути на текущем языке",  
    "DEFAULT" => "шаблон пути по-умолчанию",  
    "VARIABLES" => "массив внутренних названий переменных, которые могут  
использоваться в шаблоне"  
)
```

## Файл result\_modifier.php

Файл **result\_modifier.php** - инструмент для модификации данных работы компонента произвольным образом. Создается разработчиком самостоятельно.

Если в папке шаблона есть файл **result\_modifier.php**, то он вызывается перед подключением шаблона.

Схема работы компонента с файлом **result\_modifier.php**:



В этом файле можно запросить дополнительные данные и занести их в массив результатов работы компонента `$arResult`. Это может оказаться полезным, если требуется вывести на экран какие-либо дополнительные данные, но не хочется кастомизировать компонент и отказываться от его поддержки и обновлений.

**⚠ Примечание:** файл `result_modifier.php` запускается только если шаблон НЕ кешируется. И через него не получится устанавливать динамические свойства типа: `title`, `keywords`, `descriptioin`.

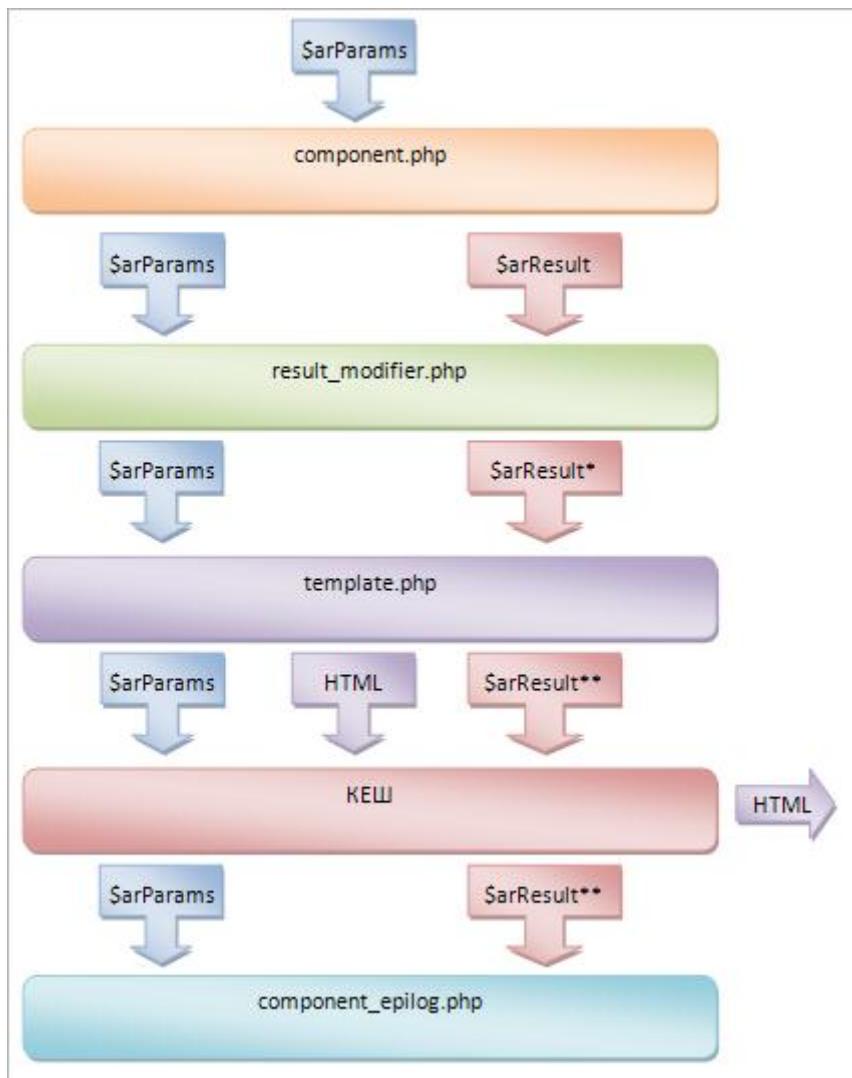
В файле доступны языковые фразы шаблона компонента и следующие переменные:

- **`$arParams`** - параметры, чтение, изменение. Не затрагивает одноименный член компонента, но изменения тут влияют на `$arParams` в файле `template.php`.
- **`$arResult`** — результат, чтение/изменение. Затрагивает одноименный член класса компонента.
- **`$APPLICATION`, `$USER`, `$DB`** - объявлять их как `$this` — ссылка на текущий шаблон (объект, описывающий шаблон, тип [CBitrixComponentTemplate](#))

## Файл component\_epilog.php

Файл **component\_epilog.php** - инструмент для модификации данных работы компонента с включенным кешированием. Создается разработчиком самостоятельно.

Схема работы компонента с файлами result\_modifier.php и component\_epilog.php:



Это файл подключается после выполнения шаблона. Аналогично файлам стилей, родительский компонент сохраняет в своем кеше список файлов эпилогов всех шаблонов дочерних компонентов (возможно вложенных), а при хите в кеш подключает эти файлы в том же порядке, как они исполнялись без кеширования. Точно так же при вызове дочерних компонентов в шаблоне нужно передавать значение \$component.

В файле **component\_epilog.php** доступны **\$arParams**, **\$arResult**, но эти значения берутся из кеша. Набор ключей массива **\$arResult**, попадающих в кеш, определяется в **component.php** вызовом вида:



```
$this->SetResultCacheKeys(array(
    "ID",
    "IBLOCK_TYPE_ID",
    "LIST_PAGE_URL",
    "NAV_CACHED_DATA",
    "NAME",
    "SECTION",
    "ELEMENTS",
));
});
```

При разработке своих компонентов обязательно используйте такую конструкцию чтобы ограничить размер кеша только необходимыми данными.

**⚠ Примечание** В файле **component\_epilog.php** может быть любой код - только следует учитывать, что исполняться он будет «мимо кеша» на каждом хите. До версии главного модуля 10.0 в шаблон компонента передавалась копия массива **arResult**. В силу этого модификация этого массива в файле **result\_modifier.php** не давала никаких результатов. Чтобы стало возможным изменение результатов вывода закешированных данных нужно разместить в файле **result\_modifier.php** следующий код:

```
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
global $APPLICATION;

$cp = $this->__component; // объект компонента

if (is_object($cp))
{
    // добавим в arResult компонента два поля - MY_TITLE и IS_OBJECT
    $cp->arResult['MY_TITLE'] = 'Мое название';
    $cp->arResult['IS_OBJECT'] = 'Y';
    $cp->SetResultCacheKeys(array('MY_TITLE','IS_OBJECT'));

    // сохраним их в копии arResult, с которой работает шаблон
    $arResult['MY_TITLE'] = $cp->arResult['MY_TITLE'];
    $arResult['IS_OBJECT'] = $cp->arResult['IS_OBJECT'];

    $APPLICATION->SetTitle($cp->arResult['MY_TITLE']); // не будет работать на
    // каждом хите:
    // отработает только первый раз, затем будет все браться из кеша и вызова
    $APPLICATION->SetTitle()
```



```
// не будет. Поэтому изменение title делается в component_epilog, который
выполняется на каждом хите.

}
```

```
?>
```

После чего изменения **arResult**, совершенные в шаблоне, станут доступны в **component\_epilog.php**.

### Пример файла component\_epilog.php

```
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();

global $APPLICATION;

if (isset($arResult['MY_TITLE']))
    $APPLICATION->SetTitle($arResult['MY_TITLE']);
?>
```

### Особенность использования

Файл **component\_epilog.php** подключается непосредственно после подключения и исполнения шаблона. Таким образом, если в компоненте идёт подключение шаблона, а затем в коде компонента следуют ещё операции, то они будут выполнены уже после выполнения файла **component\_epilog.php**.

Соответственно, в случае совпадения изменяемых данных в **component\_epilog.php** и в коде компонента после подключения шаблона выведены будут только последние данные, то есть из кода компонента.

### **Пример такой ситуации**

Используем файл **component\_epilog.php** из примера выше. А в коде компонента (файл **component.php**) есть такой код:

```
<?
$this->IncludeComponentTemplate();
if($arParams["SET_TITLE"])
{
    $APPLICATION->SetTitle($arResult["NAME"], $arTitleOptions);
}
?>
```

В этом случае вы не получите желаемого результата, выводятся данные компонента, а не **component\_epilog.php**.

В файле component\_epilog.php доступны:

- **\$arParams** - параметры, чтение/изменение не затрагивает одноименный член компонента.
- **\$arResult** — результат, чтение/изменение не затрагивает одноименный член класса компонента.
- **\$componentPath** — путь к папке с компонентом от DOCUMENT\_ROOT (например /bitrix/components/bitrix/iblock.list).
- **\$component** — ссылка на **\$this**.
- **\$this** — ссылка на текущий вызванный компонент (объект класса [CBitrixComponent](#)), можно использовать все методы класса.
- **\$epilogFile** — путь к файлу **component\_epilog.php** относительно DOCUMENT\_ROOT
- **\$templateName** - имя шаблона компонента (например: **.default**)
- **\$templateFile** — путь к файлу шаблона от DOCUMENT\_ROOT (напр. /bitrix/components/bitrix/iblock.list/templates/.default/template.php)
- **\$templateFolder** — путь к папке с шаблоном от DOCUMENT\_ROOT (напр. /bitrix/components/bitrix/iblock.list/templates/.default)
- **\$templateData** — обратите внимание, таким образом можно передать данные из **template.php** в файл **component\_epilog.php**, причем эти данные закешируются и будут доступны в **component\_epilog.php** на каждом хите/
- **\$APPLICATION, \$USER, \$DB** — глобальные переменные.

## Работа комплексного компонента в SEF режиме

В комплексные компоненты встроена функция генерации ЧПУ. У этих компонентов всегда есть входной параметр **SEF\_MODE**, который может принимать значения **Y** и **N**. Если параметр **SEF\_MODE** равен **N**, то компонент работает с физическими ссылками и все параметры передает через стандартные параметры HTTP запроса. Например:

```
/fld/cat.php?IBLOCK_ID=12&SECTION_ID=371
```

Если параметр **SEF\_MODE** равен **Y**, то компонент генерирует и обрабатывает ссылки на основании шаблонов. Например, он может понять и обработать ссылку:

```
/catalog/section/371.php?IBLOCK_ID=12
```



, даже если сам он лежит в файле **/fld/cat.php**.

Если параметр **SEF\_MODE** равен **Y**, то у компонента должен так же присутствовать параметр **SEF\_FOLDER**, который должен содержать путь до папки, с которой работает компонент. Этот путь может как совпадать с физическим путем, так и не совпадать. Например, в компоненте **bitrix:catalog**, подключенным в файле **/fld/cat.php**, могут быть установлены параметры **SEF\_MODE = Y** и **SEF\_FOLDER=/catalog/**. Тогда компонент будет отвечать на запросы по пути **/catalog/....** По умолчанию редактор должен устанавливать текущий физический путь к редактируемому файлу.

У комплексного компонента, который может работать в режиме SEF, должен быть определен набор шаблонов путей **по умолчанию**. Например, в комплексном компоненте **bitrix:catalog** может быть определен следующий массив:

```
$arDefaultUrlTemplatesSEF = array(
    "list" => "index.php",
    "section" => "section.php?IBLOCK_ID=#IBLOCK_ID#&SECTION_ID=#SECTION_ID#",
    "element" => "element.php?ELEMENT_ID=#ELEMENT_ID#"
);
```

Эти шаблоны путей могут быть переопределены с помощью входного параметра комплексного компонента **SEF\_URL\_TEMPLATES**, содержащего новый массив всех шаблонов путей или их части.

При сохранении страницы с компонентом, работающим в SEF режиме, в редакторе создается или обновляется запись в системе **urlrewrite**. Например, при сохранении файла **/fld/cat.php**, в котором лежит компонент **bitrix:catalog**, переключенный в SEF режиме с параметром **SEF\_FOLDER=/catalog/**, в системе **urlrewrite** создается или обновляется запись типа:

```
array(
"CONDITION" => "#^/catalog/#",
"ID" => "bitrix:catalog",
"PATH" => "/fld/cat.php"
),
```

- в **CONDITION** записывается значение параметра **SEF\_FOLDER**, обрамленное символами «**#^**» и «**#**»;
- в **ID** записывается название компонента;
- в **PATH** записывается физический путь к файлу, который сохраняется.

Если запись с таким **PATH** и **ID** уже есть, то она обновляется, если нет – добавляется.



В run-time при запросе физически не существующей страницы механизм urlrewrite производит поиск соответствующей записи по **CONDITION** и передает управление на страницу **PATH**.

Компонент на странице **PATH** на основании шаблонов путей выясняет запрашиваемую страницу и восстанавливает переменные, спрятанные в пути.

**⚠ Внимание!** Обязательное требование к набору шаблонов путей данного компонента - это уникальность каждого шаблона пути без учета параметров и переменных. Это должно проверяться при сохранении страницы в визуальном редакторе.

То есть, набор шаблонов путей

```
"section" => "section/#SECTION_ID#.php?IBLOCK_ID=#IBLOCK_ID#",
"element" => "element/#ELEMENT_ID#.php"
```

является допустимым, а набор шаблонов путей

```
"section" => "#SECTION_ID#.php?IBLOCK_ID=#IBLOCK_ID#",
"element" => "#ELEMENT_ID#.php"
```

допустимым не является.

## Типичные ошибки

Цитатник веб-разработчиков.

**Террорист:** И правда, все проблемы в руках! Ищите ошибки в коде!!!  
Все теги должны отвечать стандартам. А Битрикс, подхвачен!!!  
Который раз помогает очистка кода. Пусть и ручная работа, и  
кропотливая, но с достойным финалом!

### Не удалось обнаружить код вызова компонента

Ошибка может возникать из-за разных причин:

- Код вызова компонента не взят в отдельные <? ?>.

**Решение:** проверить отделенность кода компонента от другого php-кода на странице.

То есть, если у вас на странице php-код в таком виде:

```
<?
```



```
php-код
```

```
компонент
```

```
php-код
```

```
?>
```

то будет ошибка.

Необходимо, что бы было так:

```
<?
```

```
php-код
```

```
?>
```

```
<?
```

```
компонент
```

```
?>
```

```
<?
```

```
php-код
```

```
?>
```

- Ошибки в html коде на странице.

**Решение:** проверить валидность html кода.

- Несоответствие кодировки файла с проектом в целом.

**Решение:** проверить кодировку. Достаточно универсальное решение - в файле **.htaccess** прописать строки:

Для сайта с кодировкой windows-1251:

```
php_value mbstring.func_overload 0  
php_value mbstring.internal_encoding cp1251
```

Для сайта с кодировкой UTF-8:

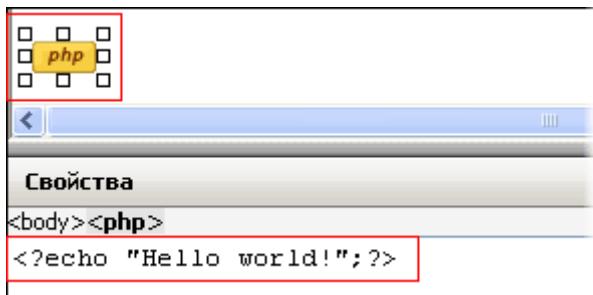
```
php_value mbstring.func_overload 2  
php_value mbstring.internal_encoding utf-8
```

- Несоответствие между владельцем файла и пользователем под которым система файлы редактирует.

**Решение:** проверить права пользователя.

## Добавление произвольного PHP кода

В рабочую область страницы также может быть добавлен произвольный PHP-код, который в режиме редактирования в визуальном редакторе будет отображаться в виде значка . Если установить курсор на этот значок, то в панели **Свойства** отобразится непосредственно сам код:



Таким образом, также можно реализовывать дополнительный функционал проекта. При этом рекомендуется использовать этот компонент в крайнем случае. За правило должно браться изменение логики работы за счет кастомизации [самого компонента](#), либо [его шаблона](#).

**Внимание!** PHP-код непосредственно в теле страницы не рекомендуется!

## Кэширование компонентов

Цитатник разработчиков.

**Денис Шаромов:** Периодически приходится наблюдать картину: хостер отключает аккаунт за большую нагрузку на сервер, клиент обращается к нам в техподдержку, мы видим, что кэширование компонентов не используется. Клиент объясняет это так: "мне сказали ваши партнёры-разработчики компонентов, что для этого компонента нельзя включать кэширование т.к. оно в битриксе криво работает".

На самом деле проблема в неумении работать с кэшированием.

Одним из видов кэширования в **Bitrix Framework** является кэширование компонентов.

Для ускорения обработки запроса клиента и уменьшения нагрузки на сервер компоненты должны использовать кэширование. Кэшировать, как правило, необходимо ту информацию, которая не зависит от конкретного обратившегося человека. Например,

список новостей сайта идентичен для всех посетителей. Поэтому нет смысла выбирать данные каждый раз из базы.

Все динамические компоненты, которые используются для создания веб-страниц, имеют встроенную поддержку управления кешированием. Для использования технологии достаточно включить автокеширование одной кнопкой на административной панели. Это удобно использовать на этапе разработки, когда автокеширование можно выключить, что облегчит работу, а перед сдачей проекта снова включить. При этом все компоненты, у которых в настройках был включен режим автокеширования, создадут кеши и полностью перейдут в режим работы без запросов к базе данных.

**⚠ Внимание!** При использовании режима **Автокеширования**, обновление информации, выводимой компонентами, происходит в соответствии с параметрами отдельных компонентов.

Управление автокешированием располагается на закладке **Кеширование компонентов** ([Настройки > Настройки продукта > Автокеширование](#)):

The screenshot shows the 'Component Caching' tab selected in the top navigation bar. Below it, the 'Cache component configuration' section is active. A large button labeled 'Disable component caching' is visible. A note below explains that the 'Auto-caching' mode speeds up site work. It also states that when this mode is enabled, components with the 'Auto + Managed' caching configuration will switch to a caching mode. At the bottom, there's a toolbar with icons for refresh, components management, and a pencil.

**⚠ Примечание:** При включении режима **автокеширования компонентов**, компоненты с настройкой кеширования **Авто + Управляемое** будут переведены в режим работы с кешированием.

Чтобы обновить содержимое закешированных объектов на странице, вы можете:

- Перейти на нужную страницу и обновить ее содержимое, используя кнопку **Сбросить кеш** на панели инструментов.
- В режиме **Правки сайта** использовать кнопки для очистки кеша в панели отдельных компонентов.

- Использовать автоматический сброс кеша по истечении времени кеширования, для чего в настройках компонента выбрать режим кеширования **Кешировать** или **Авто + Управляемое**.
- Использовать автоматический сброс кеша при изменении данных, для чего в настройках компонента выбрать режим кеширования **Авто + Управляемое**.
- Перейти к настройкам выбранных компонентов и перевести их в режим работы без кеширования.

**⚠ Примечание:** дополнительно о кешировании компонентов смотрите в уроке [Кеширование в собственных компонентах](#).

## Автокеширование

Автокеширование появилось в 6-й версии продукта для замены стандартного кеширования компонентов. Отличие в том, что автокеширование может выключаться глобально на весь сайт одной кнопкой в административной части на странице **Автокеширование** [Настройки > Настройки продукта > Автокеширование](#).

Как правило на этапе разработки оно выключено, и включается перед сдачей проекта. Не стоит питать иллюзий относительно того, что **Bitrix Framework** сам будет выбирать время кеширования и подходящий момент для очистки кеша. Это может делать только разработчик решения, основываясь на реальных потребностях конкретного проекта: необходимо указывать в настройках компонентов время кеширования, адекватное периодичности обновления информации.

## Устройство и место хранения

Кеш компонентов хранится в файлах в папке `/bitrix/cache`.

Идентификатор кеша компонента формируется на основе следующих параметров:

- ID текущего сайта, который определяет путь к файлу с кешем. (Есть возможность использовать альтернативные способы хранения, но от этого не зависит работа с компонентами).
- имени компонента,
- имени шаблона компонента,
- параметров компонента,
- внешних условий, которые определяются в компоненте (например, список групп, к которым привязан текущий пользователь).

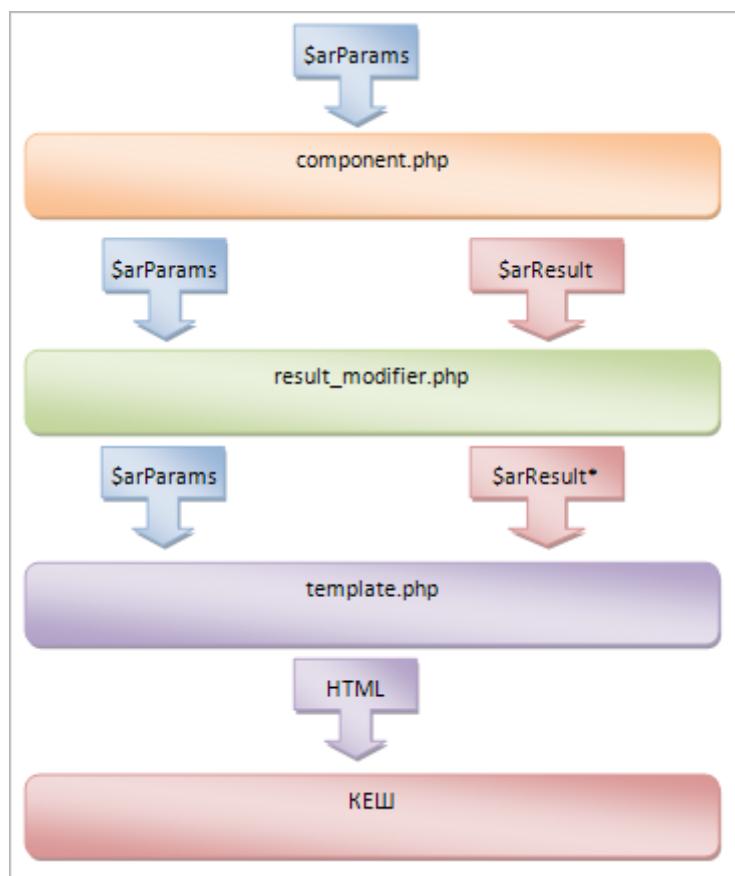
Зная все эти параметры, можно очистить кеш любого компонента. Иначе кеш будет сброшен по истечении времени кеширования, кнопкой обновить кеш на панели инструментов публичной части или полной очисткой кеша из административной части.



**⚠ Примечание:** Когда сбрасываете кеш страницы кнопкой , имейте ввиду, что компонент может использовать привязку к группам для хранения кеша, тогда незарегистрированные пользователи будут по-прежнему видеть не актуальную страницу.

Непосредственно после добавлении/изменении элемента инфоблока в административной части кеш публичных компонентов не сбрасывается. Образно это объясняется тем, что инфоблок **Новости** "не знает", где выводятся новости в публичной части и сколько компонентов их отображает. Это не проблема если время кеширования выставлено правильно.

### Структурная схема работы компонента



В **\$arParams** содержится набор параметров компонента, **component.php** работает с входными параметрами запроса и базой данных, формирует результат в массив **\$arResult**. Шаблон компонента преобразует результат в текст HTML.

При первом хите сформированный HTML попадает в кеш. При последующих хитах запросов в базу не делается (или делается мало), данные читаются из кеша.



**⚠ Внимание!** Учитите порядок выполнения, в этом случае код шаблона компонента и *result\_modifier.php* не исполняются.

Типичная ошибка: в шаблоне компонента вызываются отложенные функции: *\$APPLICATION->SetTitle()*, *\$APPLICATION->AddChainItem()* и т.д. В этом случае они работают только при выключенном кэшировании.

При разработке шаблонов собственных компонентов надо следовать простому правилу: задача шаблона - на основе входного массива **\$arResult** сформировать текст HTML на выходе.

В собственных компонентах надо формировать такой ID кеша, который однозначно определяет результирующий html. Но не допускайте попадание избыточных данных в ID кеша: это приводит к расходу места на диске и снижает попадания в кеш (а значит эффективность кэширования).

## Cache Dependencies (тегированный кеш)

С версии главного модуля 9.1.0 появилась поддержка тегов кеша. Кеш можно помечать тегами и сбрасывать по тегам же. Сброс кеша компонентов инфоблоков происходит при изменении информации в них.

Базовый код тегирования кеша:

```
01: $cache_id = md5(serialize($arParams));
02: $cache_dir = "/tagged_getlist";
03:
04: $obCache = new CPHPCache;
05: if($obCache->InitCache(36000, $cache_id, $cache_dir))
06: {
07:   $arElements = $obCache->GetVars();
08: }
09: elseif(CModule::IncludeModule("iblock") && $obCache->StartDataCache())
10: {
11:   $arElements = array();
12:   $rsElements = CIBlockElement::GetList($arParams["order"], $arParams["filter"]);
13:
14:   global $CACHE_MANAGER;
15:   $CACHE_MANAGER->StartTagCache($cache_dir);
16:   while($arElement = $rsElements->Fetch())
17:   {
```



```
18:     $CACHE_MANAGER->RegisterTag("iblock_id_".${arElement["ID"]});  
19:     ${arElements[]} = ${arElement};  
20: }  
21: $CACHE_MANAGER->RegisterTag("iblock_id_new");  
22: $CACHE_MANAGER->EndTagCache();  
23:  
24: $obCache->EndDataCache(${arIBlocks});  
25: }  
26: else  
27: {  
28:     ${arElements} = array();  
29: }
```

В строке 01 инициализируется уникальный идентификатор файла кеша. Далее определяется каталог относительно /bitrix/cache в котором будут сохранятся файлы кеша с разными значениями **\$arParams**. Важно, что этот путь начинается со слеша и им не заканчивается. При использовании в качестве кеша **memcached** или **APC** это будет критичным при сбросе кеша.

В строках 04-05 инициализируется объект кеша. Если время кеширования не истекло, то будет выполнена строка 07 и мы получим данные из файла кеша.

Условие в строке 09 фактически всегда будет **true**. Это подключение модуля и начало кеширования.

В строке 12 происходит обращение к базе данных. Важно, чтобы все параметры от которых зависит результат выборки "поучаствовали" в идентификаторе кеша (**\$cache\_id**).

В 14-й строчке объявляется доступ к переменной **\$CACHE\_MANAGER**. Этот объект будет управлять тегами.

Строка 15 - все последующие назначаемые теги будут привязаны к каталогу **\$cache\_dir**. При сбросе кеша по одному из них содержимое этого каталога будет удалено. **StartTagCache** - может использоваться рекурсивно. Например:

```
$CACHE_MANAGER->StartTagCache($cache_dir1);  
$CACHE_MANAGER->StartTagCache($cache_dir2);  
    $CACHE_MANAGER->StartTagCache($cache_dir3);  
    $CACHE_MANAGER->EndTagCache();  
$CACHE_MANAGER->EndTagCache();  
$CACHE_MANAGER->EndTagCache();
```



Важно чтобы вызовы **StartTagCache** и **EndTagCache** были сбалансированы. Объект **\$CACHE\_MANAGER** создает и отслеживает стек каталогов кеша. При этом теги назначенные на каталог **\$cache\_dir3** будут также связаны и с **\$cache\_dir2** и **\$cache\_dir1**. Теги назначенные на **cache\_dir2** будут связаны и с **\$cache\_dir1**.

В строке 18 происходит отметка кеша тегом с помощью метода **RegisterTag**. Длина тела может быть до 100 символов. В методе **RegisterTag** автоматически удаляются дубликаты тегов.

Строка 22 - запись тегов каталога в таблицу базы данных. Можно считать по одному **insert'у** на тег.

Сброс кеша:

```
$CACHE_MANAGER->ClearByTag("iblock_id_7");
```

### Компоненты инфоблоков

Для запуска механизма необходимо определить константу в файле **dbconn.php**.

```
define("BX_COMP_MANAGED_CACHE", true);
```

При этом в методе **StartResultCache** компонента будет вызываться **StartTagCache** с путем к кешу компонента (с учетом страницы). А в методе **EndResultCache** (который в свою очередь вызывается из **IncludeComponentTemplate**) - **EndTagCache**.

В модуле инфоблоков [CIBlockElement::GetList](#) и [CIBlockSection::GetList](#) возвращают объект класса [CIBlockResult](#).

В методе **Fetch/GetNext** этого объекта будут вызываться **\$CACHE\_MANAGER->RegisterTag("iblock\_id\_".\\$res["IBLOCK\_ID"]);**. Если выборка не содержит элементов, то значение идентификатора инфоблока будет взято из фильтра.

Сброс кеша вызывается из методов **Add/Update/Delete** для элементов, разделов и инфоблоков.

Не стоит использовать этот механизм при частом обновлении элементов или разделов. С другой стороны это должно быть удобным для редакторов сайтов: изменения моментально отображаются на сайте. Еще один плюс - можно задавать практически бесконечное время кеширования.

### **Советы от веб-разработчиков.**

**Антон Долганин:** Не забывайте, что управляемый кеш инфоблоков очищается только при вызове **Update**. При изменении, например, свойств с помощью [SetPropertyValueCode](#) очистки не произойдет.

*Делаем вручную после изменения:*

```
if(defined('BX_COMP_MANAGED_CACHE'))  
    $GLOBALS['CACHE_MANAGER']-  
>ClearByTag('iblock_id_'].$arParams['IBLOCK_ID']);
```

## Работа с компонентами

**Цитатник веб-разработчиков.**

**Антон Долганин:** Стражайше запрещено писать любую PHP-логику на обычной странице сайта. Любой php-код должен инкапсулироваться в компоненты. Максимум, что позволяет - использовать подключение скриптов с помощью [IncludeFile](#), но это если разработчик полностью отдает себе отчет в том, что этого делать нельзя и это будет исправлено в ближайшее время, или того действительно требует логика проекта (единичные случаи).

Компонент – основной способ вывода информации в **Bitrix Framework**. Соответственно именно работа с ним дает максимальные возможности по изменению условий вывода данных и изменению (добавлению) функционала системы.

Рекомендуемое соотношение задач и способов их решений:

- Для решения задач изменения формы вывода данных модифицируйте шаблон компонента.
- Для изменения и дополнения кешируемых данных, выводимых компонентом, используйте возможности файла **result\_modifier.php**.
- Для реализации логики, отрабатывающей при каждом вызове компонента независимо от кеширования, используйте возможности файла **component\_epilog.php**.
- Для дополнения и неявного изменения (без вмешательства в код) логики работы компонента можно использовать технологию Событий.
- Для дополнения логики работы компонента копируйте компонент в свое пространство имен и изменяйте его.
- Для создания новой логики и новых возможностей создавайте компонент заново.

Достаточно часто задачу приходится решать комбинацией методов. То есть, например, редактировать шаблон и добавлять код в **result\_modifier.php**.



В указанном порядке задач и способов решений и мы рассмотрим работу с компонентами в главах ниже.

**⚠ Внимание!** Осуществляя любые действия по работе с компонентами надо также учитывать кеширование. Внедрение тяжелых кодов в `component_epilog.php` под кеширование не попадает. И бывают случаи, когда все же правильнее кастомизировать компонент, что дает выигрыш в производительности (особенно, если какой-то тяжелый код используется на главной или наиболее часто посещаемой странице).

## Кастомизация шаблона

**Цитатник веб-разработчиков.**

**Рамиль Юналиев:** По себе знаю, что не хватало примеров работы именно с шаблонами, потому что с них начинал. Если программисты "въедут" в эти шаблоны то там уже пойдет...

В веб-программировании и программировании вообще есть популярный миф, что логику, данные и представления легко и очевидно можно отделить друг от друга, что решение этой задачи однозначно. Если немного подумать, становится очевидно что логика, данные и представления легко переходят между собой в зависимости от того, с какого уровня абстракции вы смотрите на задачу. Например, html-код шаблона для разработчика сайта это безусловно представление, для браузера это код, а для ядра CMS это данные.

Разработчики, впервые видящие код шаблона любого компонента, например списка новостей, ужасаются циклу, каше из php и HTML (и js изредка можно встретить). Это шаблон, он для вывода. Все данные уже собраны, запросы отработали, понятно что и куда мы выводим, даже отлов угроз уже прошел. Шаблон - последнее звено в цепочке кода и данных. Тут логика вывода действительно смешана с оформлением, и это правильно.

Начинающему разработчику часто очень непросто разобраться в том, где в шаблоне данные, где представление. В шаблоне можно писать любой php-код, можно написать прямое, не API, обращение к базе данных, можно написать десяток строк на API и решить задачу. Битрикс этим очень "развращает" разработчика: не запрещено писать бизнес-логику в шаблонах и HTML в компонентах, слишком много свободы. В собственном коде и по идеологии **Bitrix Framework** задача разделения данных решена нормально, а вот написание в шаблоне ерунды, не рекомендованной по идеологии, система не контролирует.

Приступая к кастомизации шаблона нужно помнить:

**⚠ Внимание!** вся логика должна быть в компоненте, в шаблоне - только представление вывода полученных данных!

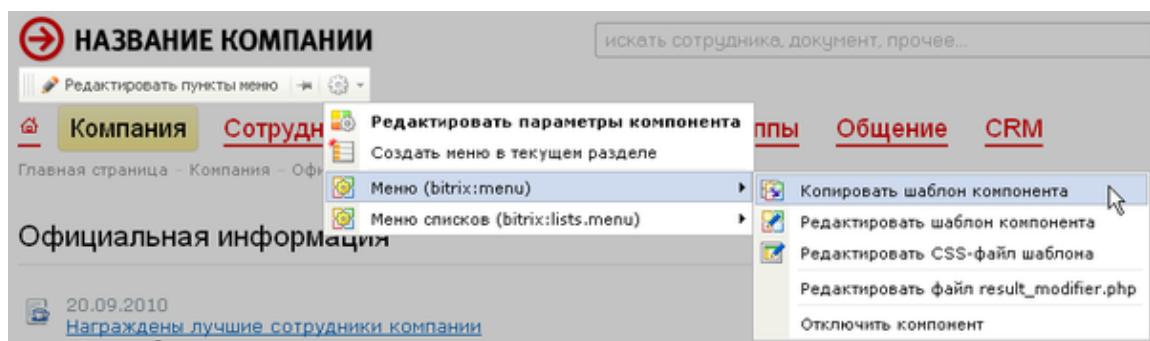
Кастомизация шаблона компонента, как правило, преследует две цели:

- Приведение формы вывода данных компонента в соответствие с дизайном сайта;
- Организация вывода данных компонента в виде, недоступном в стандартном варианте.

Пользовательские шаблоны компонента - шаблоны, которые изменены под нужды конкретного проекта. Они должны лежать в папках шаблонов портала (т.е. в /bitrix/templates/шаблон\_сайта/). При копировании шаблона компонента средствами системы, они будут расположены по следующему пути: /bitrix/templates/шаблон\_сайта/components/namespace/название\_компонента/название\_шаблона.

Копировать шаблон можно следующими способами:

- В рамках файловой системы копированием папки /bitrix/components/bitrix/\_нужный\_компонент\_/templates/ в папку /bitrix/templates/шаблон\_сайта/components/namespace/название\_компонента/\_название\_шаблона.
- Средствами интерфейса системы с помощью команды **Копировать шаблон компонента**:



После копирования шаблон можно изменять и, после изменения, применить его к компоненту в настройках компонента.

В главе приведены некоторые примеры кастомизации шаблонов.

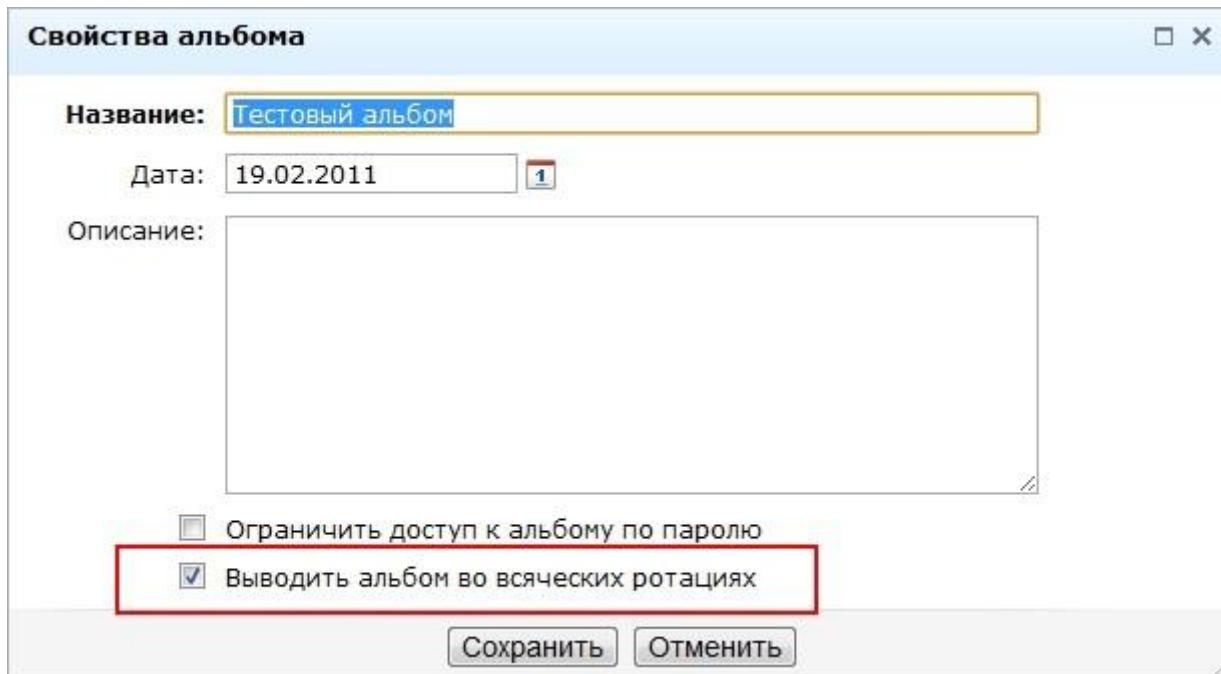
**⚠ Примечание:** Еще один пример редактирования шаблона (шаблона меню) представлен на странице [Кастомизация шаблонов компонентов](#) главы [Как создать простой сайт](#).



## Добавление функции

Рассмотрим на примере свойств фотоальбома добавление функции с помощью кастомизации шаблона компонента.

Добавим функцию вывода альбома при любых ротациях (и чтобы она сохранялась, конечно):



Какой первый порыв новичка? Вынести компонент **section.edit** в свое пространство имен (а то бывает и всю фотогалерею целиком выносят) и в коде прописать всю логику. Верное решение, задачу оно решит. Но завтра в компоненте появится некая вкусная функция, а вы её не получите.

## **Изменения в шаблоне**

Скопируйте шаблон данного компонента.

Код вставки галочки в шаблон мы пропустим, задача банальная, только рядом с этим кодом добавьте скрытое поле:

```
<input type="hidden" name="PHOTOALBUM_PUBLIC_EDIT" value="Y" />
```

Файл *result\_modifier.php*

А также создайте *result\_modifier.php*, где разместите такой код:

```
$arResult['SECTION']['ID'], 'IBLOCK_ID' => $arResult['SECTION']['IBLOCK_ID'],
false, array('UF_COOL_ALBUM'))->GetNext();
$arResult['SECTION']['UF_COOL_ALBUM'] = $arSection['UF_COOL_ALBUM'];
```



```
?>
```

**UF\_COOL\_ALBUM** - это код свойства добавляемой галочки.

### Обработчик событий

Осталось написать обработчик на обновление секции, вот и весь секрет:

```
function OnAfterIBlockSectionUpdateHandler(&$arFields)
{
    if ($arFields['RESULT'] &&
        $arFields['IBLOCK_ID']==68 &&
        $_POST['PHOTOALBUM_PUBLIC_EDIT']=='Y')
    {
        $_POST['PHOTOALBUM_PUBLIC_EDIT'] = 'N';
        $es = new CIBlockSection();
        $es->Update($arFields['ID'], array('UF_COOL_ALBUM' => $_POST['COOL_ALBUM']));
    }
}
```

Рассмотрим простой пример шаблона компонента **bitrix:news.detail** с выводом из папки `/images/`, расположенной в папке шаблона, изображения **bullet.gif**. Для получения HTML-кода изображения мы воспользовались функцией [CFile::ShowImage\(\)](#):

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!=true)die();?>
<?
//выводим изображение из папки шаблона компонента
$bullet = CFile::ShowImage($templateFolder .'/images/bullet.gif');
?>
<?=$bullet?>


<?if($arParams["DISPLAY_NAME"]!="N" && $arResult["NAME"]):?>
    <h2 class="alt"><?=$arResult["NAME"]?></h2>
    <?endif;?>
    <?if($arParams["DISPLAY_PICTURE"]!="N"
is_array($arResult["DETAIL_PICTURE"])):?>
        <img class="detail_picture" border="0"
src=<?=$arResult["DETAIL_PICTURE"]["SRC"]?>"


```



## Модификация шаблона простого компонента в составе комплексного

Комплексный компонент обеспечивает взаимодействие простых компонентов с общей тематикой. Простые компоненты содержат код непосредственной работы с данными. Например, компоненты соцсети настраивать по отдельности не удобно.

Но если надо лишь изменить внешний вид некоторых элементов, это можно легко сделать, не отказываясь от остальных стандартных шаблонов.

Желательно обходиться без кастомизации компонента там, где в этом нет особой необходимости. В этом случае:

- не теряются обновления;
- легче решать проблемы через техподдержку (ТП не занимается решением проблем, возникающих в работе кастомного кода если явно не обозначена ошибка в работе API);
- в комплексных компонентах стандартно реализован AJAX.

В качестве примера возьмем компонент **Социальная сеть**. Задача: заменить форму редактирования описания группы (textarea) на визуальный редактор с возможностью вставки html.

**⚠ Внимание!** После решения этой задачи нужно модифицировать шаблон компонента вывода описания, чтобы не экранировались теги html. Но решение аналогичное и выходит за рамки этого описания.

### Как сделать

- Скопируйте шаблон компонента
- Сделайте замену в шаблоне:

```
<textarea name="GROUP_DESCRIPTION" style="width:98%" rows="5"><?=$arResult["POST"]["DESCRIPTION"]; ?></textarea>
```

на

```
<input type=hidden name="GROUP_DESCRIPTION"><?
$GLOBALS['APPLICATION']->IncludeComponent(
    "bitrix:fileman.light_editor",
    ".default",
    Array(
        "CONTENT" => htmlspecialchars_decode($arResult["POST"]["DESCRIPTION"]),
        "INPUT_NAME" => "GROUP_DESCRIPTION",
        "WIDTH" => "98%",
        "HEIGHT" => "200px",
```

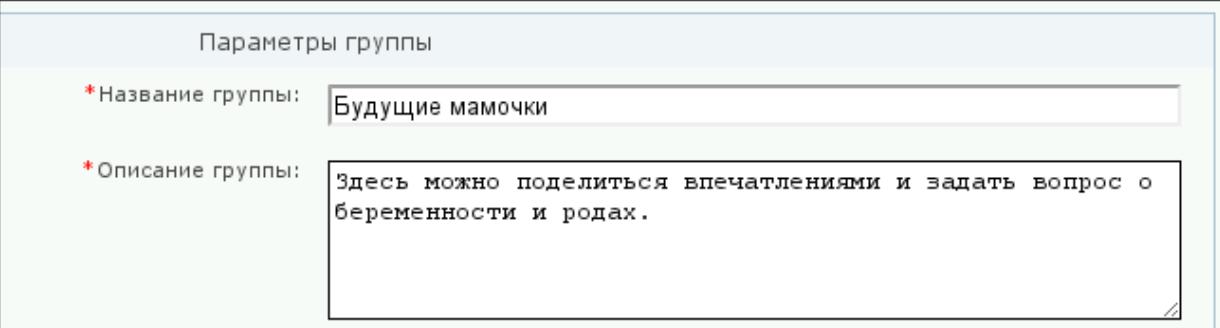
```
"USE_FILE_DIALOGS" => "N",
"FLOATING_TOOLBAR" => "N",
"ARISING_TOOLBAR" => "N",
"VIDEO_ALLOW_VIDEO" => "N",
)
);
?>
```

Получаем результат. Было:

Параметры группы

\* Название группы: Будущие мамочки

\* Описание группы: Здесь можно поделиться впечатлениями и задать вопрос о беременности и родах.



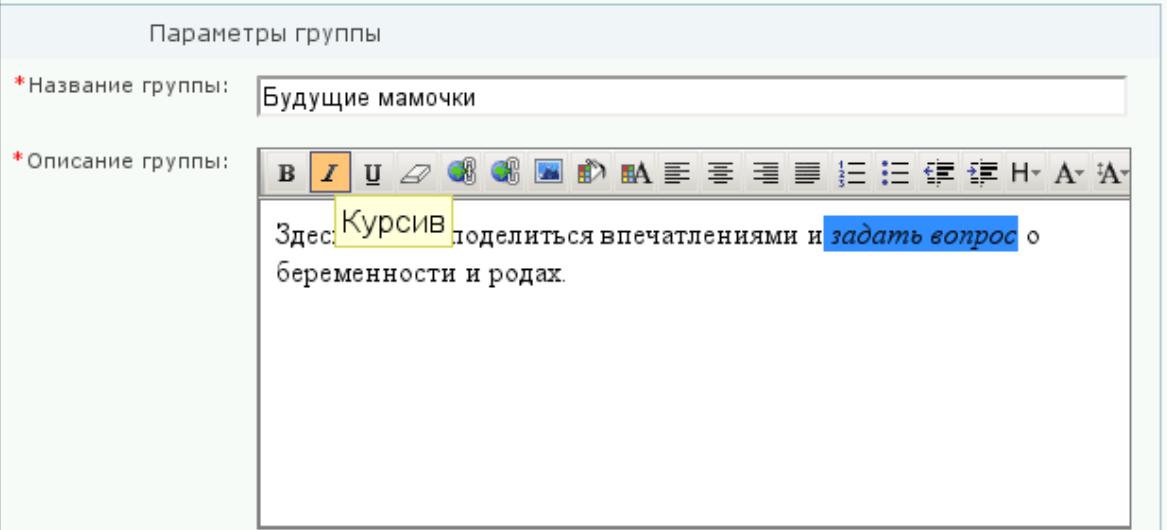
Стало:

Параметры группы

\* Название группы: Будущие мамочки

\* Описание группы:

Здесь **Курсив** поделиться впечатлениями и **задать вопрос** о беременности и родах.



При этом весь остальной код остаётся стандартный, т.е. будет обновляться и поддерживаться компанией "1С-Битрикс".

### Модификация шаблона или создание result modifier?

Достаточно часто можно решить одну и ту же задачу разными способами. Выбор конечного варианта в таких задачах остается за разработчиком, которое он принимает в



зависимости от конкретной задачи. Рассмотрим пример решений задачи, которую можно решить двумя способами.

Как сделать возможность вставки видео при публикации новостей на сайте? На первый взгляд это может показаться сложно. Но фактически все легко реализуемо. Идея состоит в том, чтобы для прикреплённого к новости файла подключать компонент **Медиа проигрыватель** (**bitrix:player**). Для отображения новости будет использоваться компонент **Новость детально** (**bitrix:news.detail**).

Какой бы способ вы не использовали создайте свойство типа **Файл** в инфоблоке новостей:

ID	Название	Тип	Множ.	Обяз.	Сорт.	Код	Изн.	Удал.
Видео ролик	Файл		<input type="checkbox"/>	<input type="checkbox"/>	500	movie		
	Строка		<input type="checkbox"/>	<input type="checkbox"/>	500			
	Строка		<input type="checkbox"/>	<input type="checkbox"/>	500			
	Строка		<input type="checkbox"/>	<input type="checkbox"/>	500			
	Строка		<input type="checkbox"/>	<input type="checkbox"/>	500			

**Еще**

**Сохранить** **Применить** **Отменить**

### Решение с редактированием шаблона

- Скопируйте шаблон компонента **news.detail** в шаблон сайта. Сам компонент менять не придётся.
- Создайте новую страницу в визуальном редакторе и разместите на ней компонент **Медиа проигрыватель** (**bitrix:player**). Укажите базовые настройки (путь к видео файлу пока не заполняйте). Скопируйте из полученного следующий код:

```
<?$_APPLICATION->IncludeComponent(
    "bitrix:player",
    "",
    Array(
        "PLAYER_TYPE" => "auto",
        "USE_PLAYLIST" => "N",
        "PATH" => "",
        "WIDTH" => "400",
        "HEIGHT" => "300",
        "FULLSCREEN" => "Y",
        "SKIN_PATH" => "/bitrix/components/bitrix/player/mediaplayer/skins",
    )
);?
```



```
"SKIN" => "bitrix.swf",
"CONTROLBAR" => "bottom",
"WMODE" => "transparent",
"HIDE_MENU" => "N",
"SHOW_CONTROLS" => "Y",
"SHOW_STOP" => "N",
"SHOW_DIGITS" => "Y",
"CONTROLS_BGCOLOR" => "FFFFFF",
"CONTROLS_COLOR" => "000000",
"CONTROLS_OVER_COLOR" => "000000",
"SCREEN_COLOR" => "000000",
"AUTOSTART" => "N",
"REPEAT" => "N",
"VOLUME" => "90",
"DISPLAY_CLICK" => "play",
"MUTE" => "N",
"HIGH_QUALITY" => "Y",
"ADVANCED_MODE_SETTINGS" => "N",
"BUFFER_LENGTH" => "10",
"DOWNLOAD_LINK_TARGET" => "_self"
)
);?>
```

- В шаблоне компонента вместо свойства movie настройте подключение медиа плеера. Найдите строки вывода свойств:

```
30    <?foreach($arResult["DISPLAY_PROPERTIES"] as $pid=>$arProperty):?>
31
32        <?=$arProperty["NAME"]?:>
33        <?if(is_array($arProperty["DISPLAY_VALUE"])):?:>
34            <?=implode(" / ", $arProperty["DISPLAY_VALUE"]);?:>
35        <?else:>
36            <?=$arProperty["DISPLAY_VALUE"];?:>
37        <?endif?>
38        <br />
39    <?endforeach;?>
```

- Вставьте проверку и замену, получается:

```
<?foreach($arResult["DISPLAY_PROPERTIES"] as $pid=>$arProperty):?>
```



```
<if ($arProperty["CODE"]=='movie' && $arProperty["DISPLAY_VALUE"]) {?>

<$APPLICATION->IncludeComponent(
    "bitrix:player",
    "",
    Array(
        "PLAYER_TYPE" => "auto",
        "USE_PLAYLIST" => "N",
        "PATH" => CFile::GetPath($arProperty["VALUE"]),
        "WIDTH" => "400",
        "HEIGHT" => "300",
        "FULLSCREEN" => "Y",
        "SKIN_PATH" => "/bitrix/components/bitrix/player/mediaplayer/skins",
        "SKIN" => "bitrix.swf",
        "CONTROLBAR" => "bottom",
        "WMODE" => "transparent",
        "HIDE_MENU" => "N",
        "SHOW_CONTROLS" => "Y",
        "SHOW_STOP" => "N",
        "SHOW_DIGITS" => "Y",
        "CONTROLS_BGCOLOR" => "FFFFFF",
        "CONTROLS_COLOR" => "000000",
        "CONTROLS_OVER_COLOR" => "000000",
        "SCREEN_COLOR" => "000000",
        "AUTOSTART" => "N",
        "REPEAT" => "N",
        "VOLUME" => "90",
        "DISPLAY_CLICK" => "play",
        "MUTE" => "N",
        "HIGH_QUALITY" => "Y",
        "ADVANCED_MODE_SETTINGS" => "N",
        "BUFFER_LENGTH" => "10",
        "DOWNLOAD_LINK_TARGET" => "_self"
    ),
    $component
);?>
<? } else {?>
```



```
<?=$arProperty["NAME"]?:>
<?if(is_array($arProperty["DISPLAY_VALUE"])):?:>
    <?=implode(" / ", $arProperty["DISPLAY_VALUE"]);?:>
<?else:>
    <?=$arProperty["DISPLAY_VALUE"];?:>
<?endif?>
<?}?>
<br />
<?endforeach;?:>
```

**⚠ Примечание:** Здесь следует обратить внимание на следующие моменты:

- Для получения пути к файлу из ID используется системный вызов [CFile::GetPath](#).
- При подключении компонентов указан четвёртый параметр **\$component** для того чтобы из публичной части случайно не изменить его параметры (см. класс [CMain::IncludeComponent](#)).

### Решение с помощью result\_modifier.php

Если вы хотите, чтобы решение не было похоже на "костыли", необходимо вынести замену свойства в **result\_modifier.php**. Тогда шаблон компонента будет стандартный.

- Создайте файл **result\_modifier.php** с кодом:

```
<?
// передадим значение свойства по ссылке:
$arProperty = &$arResult['DISPLAY_PROPERTIES'][$arParams['PROPERTY_VIDEO']];

if ($arProperty['DISPLAY_VALUE']) // проверим, установлено ли свойство
{
    global $APPLICATION;
    ob_start(); // включим буферизацию чтобы отловить вывод компонента
    $APPLICATION->IncludeComponent(
        "bitrix:player",
        "",
        Array(
            "PLAYER_TYPE" => "auto",
            "USE_PLAYLIST" => "N",
            "PATH" => CFile::GetPath($arProperty["VALUE"]),
            "WIDTH" => "400",
```



```
"HEIGHT" => "300",
"FULLSCREEN" => "Y",
"SKIN_PATH" => "/bitrix/components/bitrix/player/mediaplayer/skins",
"SKIN" => "bitrix.swf",
"CONTROLBAR" => "bottom",
"WMODE" => "transparent",
"HIDE_MENU" => "N",
"SHOW_CONTROLS" => "Y",
"SHOW_STOP" => "N",
"SHOW_DIGITS" => "Y",
"CONTROLS_BGCOLOR" => "FFFFFF",
"CONTROLS_COLOR" => "000000",
"CONTROLS_OVER_COLOR" => "000000",
"SCREEN_COLOR" => "000000",
"AUTOSTART" => "N",
"REPEAT" => "N",
"VOLUME" => "90",
"DISPLAY_CLICK" => "play",
"MUTE" => "N",
"HIGH_QUALITY" => "Y",
"ADVANCED_MODE_SETTINGS" => "N",
"BUFFER_LENGTH" => "10",
"DOWNLOAD_LINK_TARGET" => "_self"
)
);
$arResult['DISPLAY_VALUE'] = ob_get_contents(); // подменим $arResult
ob_clean(); // очистим наш буфер чтобы плеер не появился дважды
ob_end_clean(); // закроем буфер
}
?>
```

Символьный код свойства можно сделать параметром компонента, чтобы не привязываться жёстко к конкретному инфоблоку. Для этого нужно доработать файл .parameters.php компонента Новость детально, расположенный в скопированном шаблоне компонента.

- Измените код файла **.parameters.php**:

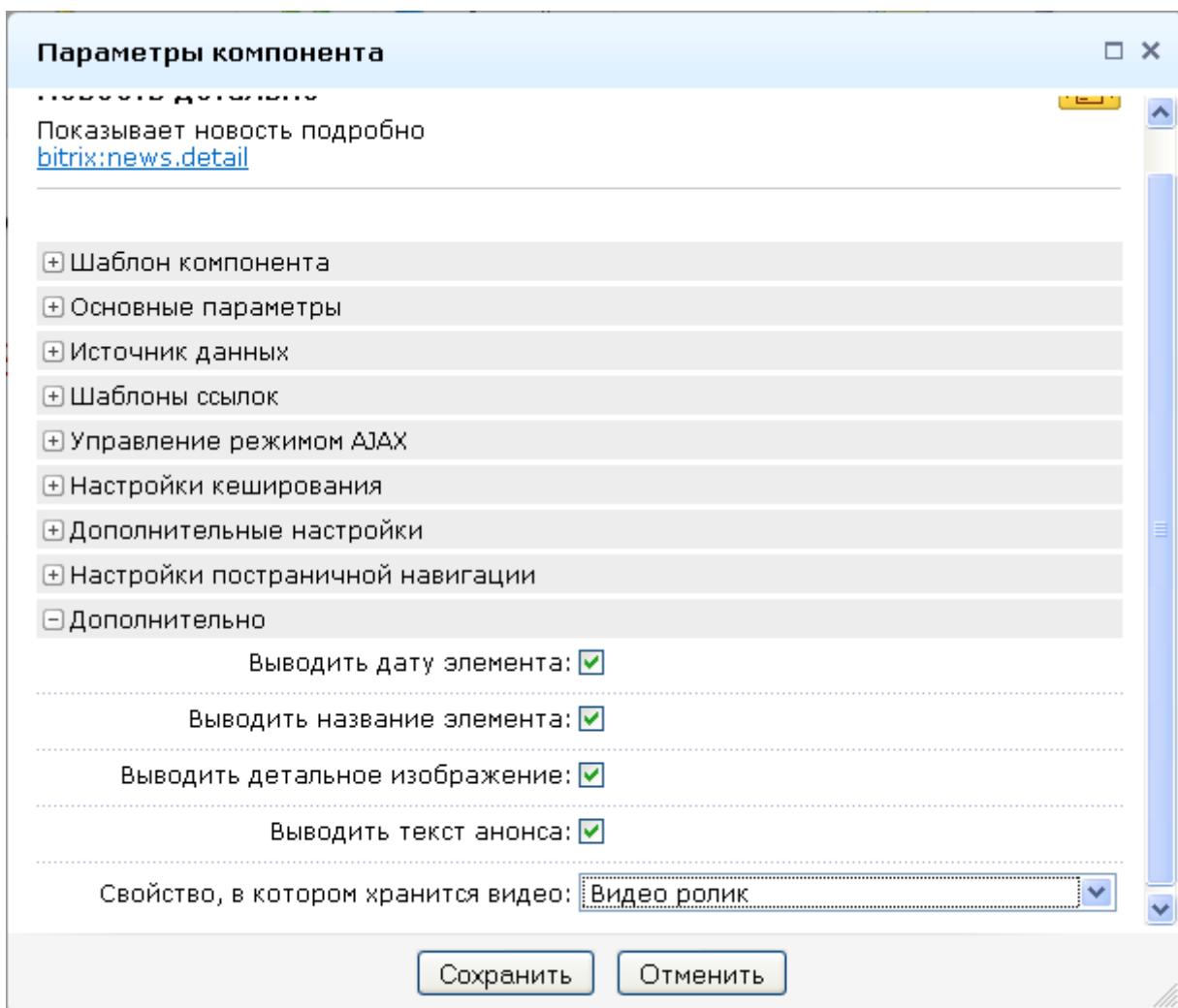
```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
```



```
$arProps = array();
$rs=CIBlockProperty::GetList(array(),array("IBLOCK_ID"=>$arCurrentValues['IBLOCK_ID'],
,"ACTIVE"=>"Y"));
while($f = $rs->Fetch())
    $arProps[$f['CODE']] = $f['NAME'];

$arTemplateParameters = array(
    "DISPLAY_DATE" => Array(
        "NAME" => GetMessage("T_IBLOCK_DESC_NEWS_DATE"),
        "TYPE" => "CHECKBOX",
        "DEFAULT" => "Y",
    ),
    "DISPLAY_NAME" => Array(
        "NAME" => GetMessage("T_IBLOCK_DESC_NEWS_NAME"),
        "TYPE" => "CHECKBOX",
        "DEFAULT" => "Y",
    ),
    "DISPLAY_PICTURE" => Array(
        "NAME" => GetMessage("T_IBLOCK_DESC_NEWS_PICTURE"),
        "TYPE" => "CHECKBOX",
        "DEFAULT" => "Y",
    ),
    "DISPLAY_PREVIEW_TEXT" => Array(
        "NAME" => GetMessage("T_IBLOCK_DESC_NEWS_TEXT"),
        "TYPE" => "CHECKBOX",
        "DEFAULT" => "Y",
    ),
    "PROPERTY_VIDEO" => Array(
        "NAME" => "Свойство, в котором хранится видео",
        "TYPE" => "LIST",
        "VALUES" => $arProps
    ),
);
?>
```

В результате в настройках параметра появляется новое поле:



Не забудьте в параметрах подключения компонента указать свойство, в котором хранится видео, иначе оно выводиться не будет.

### Вывод голосования

**Задача:**

1. Выдать голосование не в детальном товаре, а на странице со списком товаров.
2. Выдать ее на AJAX стандартными средствами

В каталоге нет голосования. Зато оно есть в комплексном компоненте **Новости**, в детальном просмотре элемента.

**Решение** (для среднеподготовленного человека, методом копи-паст).

### **Первая часть задачи**

В шаблоне комплексного компонента **Новости** находим такой код:



```
<?$_APPLICATION->IncludeComponent(
    "bitrix:iblock.vote",
    "",
    Array(
        "IBLOCK_TYPE" => $arParams["IBLOCK_TYPE"],
        "IBLOCK_ID" => $arParams["IBLOCK_ID"],
        "ELEMENT_ID" => $ElementID,
        "MAX_VOTE" => $arParams["MAX_VOTE"],
        "VOTE_NAMES" => $arParams["VOTE_NAMES"],
        "CACHE_TYPE" => $arParams["CACHE_TYPE"],
        "CACHE_TIME" => $arParams["CACHE_TIME"],
    ),
    $component
);?>
```

Вставляем этот код в шаблон компонента **bitrix:catalog.top** куда-нибудь, где он должен выводиться. Например, в таблицу после вывода <?=\$arElement ["PREVIEW\_TEXT"] ?>.

Теперь нужно сделать, чтобы голосование выводилось для нужного нам элемента. Меняем строку:

```
"ELEMENT_ID" => $ElementID,
```

на

```
"ELEMENT_ID" =>$arElement ["ID"],
```

## Вторая часть задачи

В папке с шаблонами компонента **bitrix:iblock.vote** есть два шаблона: .default (страшненький выпадающий список) и ajax. Применяем второй. Вторая проблема тоже решена. В итоге вызов компонента получился вот таким:

```
<?$_APPLICATION->IncludeComponent(
    "bitrix:iblock.vote",
    "ajax",
    Array(
        "IBLOCK_TYPE" => $arParams["IBLOCK_TYPE"],
        "IBLOCK_ID" => $arParams["IBLOCK_ID"],
        "ELEMENT_ID" =>$arElement["ID"],
        "MAX_VOTE" => $arParams["MAX_VOTE"],
        "VOTE_NAMES" => $arParams["VOTE_NAMES"],
```



```

"CACHE_TYPE" => $arParams["CACHE_TYPE"],
"CACHE_TIME" => $arParams["CACHE_TIME"],
),
$component
);?>

```

## Добавление типа отсутствия

В некоторых случаях для изменения вывода данных нужно в дополнение к модификации шаблона прибегать к изменению свойств информационных блоков, используемых компонентом.

В качестве примера рассмотрим такую задачу в рамках "1С-Битрикс: Корпоративный портал": необходимо добавить в Графике отсутствий ([Сотрудники > График отсутствий](#)) новый Тип отсутствий, например: отсутствие Отгул в счет отработанного времени.

### Задания свойств инфоблока

- В свойствах инфоблока ([Контент > Информ. блоки > Типы информ. блоков > Оргструктура > график отсутствий](#)) в закладке **Свойства** в строке Тип присутствия нажмите на кнопку с многоточием. Откроется форма Настройка свойства информационного блока:

ID XML_ID	Значение (нет значения по умолчанию)	Сорт.	Умолч.
8 VACATION	отпуск ежегодный	100	<input checked="" type="radio"/>
9 ASSIGNMENT	командировка	200	<input type="radio"/>
10 LEAVESICK	больничный	300	<input type="radio"/>
11 LEAVEMATERINIT	отпуск декретный	400	<input type="radio"/>
12 LEAVEUNPAYERD	отгул за свой счет	500	<input type="radio"/>
13 UNKNOWN	прогул	700	<input type="radio"/>
14 OTHER	другое	800	<input type="radio"/>
LEAVEPAYERD	отгул в счет отработанного времени	600	<input type="radio"/>
		500	<input type="radio"/>

**Сохранить**   **Отменить**



- В **Значениях списка** добавьте новое значение **Отгул за отработанное время**. И смените параметры сортировки, что бы расположить новый тип отсутствия в нужном месте.

### Изменение шаблона

- Скопируйте шаблон и откройте его для редактирования.
- Найдите строки:

```
GetMessage('INTR_ABSC_TPL_LEGEND_*****')?>'},
{NAME:'*****',TITLE:'
```

- Вместо звездочек вставьте **XML\_ID**, который вы использовали при добавлении значения свойства в форме настройки информационного блока.
- Сохраните внесенные изменения.
- Назначьте модифицированный шаблон для использования компонентом.

### Изменение файла CSS шаблона

- Откройте для редактирования файл стилей этого шаблона.
- Найдите строки:

```
div.bx-calendar-color-***** {background-image:
url(/bitrix/components/bitrix/intranet.absence.calendar/templates/.default/images/cale
ndar_grad_red.gif);}
```

- Добавьте свою аналогичную строку с файлом формата gif (или другого, это не важно) нужного вам цвета (предварительно загрузив файл в систему).
- Введите вместо звездочек **XML\_ID**, который вы задали при добавлении значения свойства в форме настройки информационного блока.
- Сохраните внесенные изменения.

### Добавление языкового сообщения

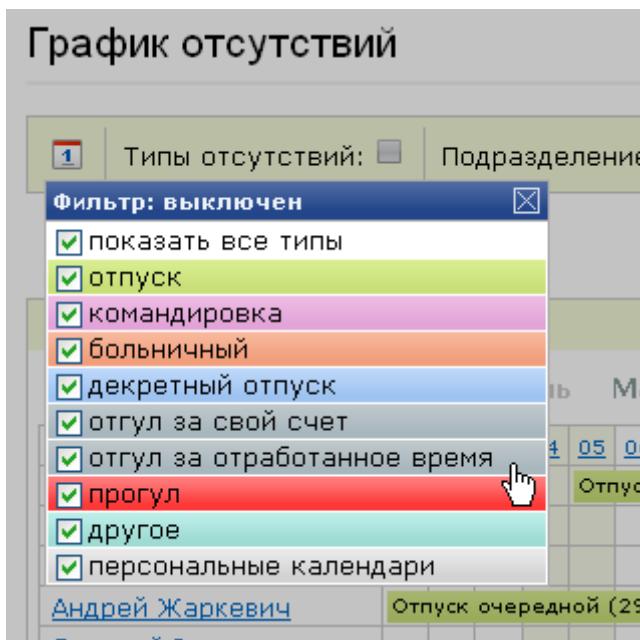
- Откройте для редактирования файл с языковыми сообщениями \bitrix\templates\.default\components\bitrix\intranet.absence.calendar\\_имя вашего шаблона\\_lang\ru\template.php.
- Добавьте строку:

```
$MESS["INTR_ABSC_TPL_LEGEND_*****"] = "отгул за отработанное
время";
```

Вместо звездочек используйте **XML\_ID**, который вы задали при добавлении значения свойства в форме настройки информационного блока.

- Сохраните внесенные изменения.

Если теперь открыть список типов отсутствий, что после отгула за свой счет появился новый тип события: отгул за отработанное время:



## Файлы result\_modifier и component\_epilog

Изменение логики работы компонентов с помощью этих файлов не требует модификации кода самого компонента. Примеры таких модификаций приведены в этой главе.

### С помощью файла result\_modifier

#### Ограничение срока показа новых сотрудников

**Задача:** показывать новых сотрудников на корпоративном портале только небольшой период времени, допустим, месяц с момента их приема на работу, а не до тех пор, пока не будет принят кто-то еще.

Чтобы решить задачу можно слегка изменить результаты работы компонента **intranet.structure.informer.new**, на основе которого работает гаджет **Новые сотрудники**.

- Копируем шаблон **.default** компонента в шаблон портала. В файл **result\_modifier.php** добавляем код:

```
//show only users hired in the last N days
$period_days = 30;
foreach ($arResult['ENTRIES'] as $key => $arEntry) {
```



```
$user_reg_timestamp = MakeTimeStamp($arEntry["DATE_ACTIVE_FROM"] ,  
"MM/DD/YYYY HH:MI:SS");  
$from = strtotime("-".$period_days." days");  
if ($user_reg_timestamp < $from) {  
    unset($arResult['ENTRIES'][$key]);  
}  
}
```

В коде задается количество дней: в данном случае гаджет будет показывать только сотрудников, принятых в течение месяца. Дата приема сотрудника будет сравниваться с текущей датой и, если сотрудник принят больше, чем месяц назад, то он выводиться не будет.

- Нужно предусмотреть ситуацию, если новых сотрудников за данный период нет. В таком случае можно выводить сообщение (добавляется в код шаблона):

```
<?  
if (count($arResult['ENTRIES']) == 0) {  
    echo GetMessage('INTR_NO_ENTRIES');  
}  
?>
```

Само текстовое сообщение будет находиться в файле шаблона /lang/ru/template.php

```
$MESS['INTR_NO_ENTRIES'] = "Новых сотрудников в последнее время не было";
```

После изменений гаджет **Новые сотрудники** сможет показывать новых сотрудников в соответствии с выбранными параметрами.

### Показ описания файлов

На тех страницах корпоративного портала, где используется комплексный компонент **Библиотека (bitrix:webdav)**, можно настроить список полей для показа. Но на вкладке Файлы на персональной странице или на вкладке Файлы в рабочих группах штатно список полей для показа настроить нельзя.

Если надо показывать описание файла на персональной странице или в рабочих группах, то в каталог /bitrix/components/bitrix/webdav.section.list/templates/.default/ добавляем файл **result\_modifier.php** со следующим кодом:

```
<?  
if (!defined('B_PROLOG_INCLUDED') || B_PROLOG_INCLUDED!==true)die();  
$arParams['COLUMNS'][] = 'PREVIEW_TEXT';  
?>
```



В результате имеем дополнительную колонку **Описание для анонса:**

Загрузить	Создать папку	Подписаться	Помощь
Путь: Файлы			
<input type="checkbox"/>		Название	Изменен
<input type="checkbox"/>		book.pdf	10.03.2010 16:15:45
			Администратор 221.34 КБ Какая-то книга

### Рекламный баннер внутри текста

Для размещения рекламного баннера внутри текста новости используйте разделитель **#BANNER\_BOTTOM#**, где **BOTTOM** - тип баннера, который будет показан.

В режиме визуального редактора код может выглядеть следующим образом:

The screenshot shows the Bitrix visual editor interface. At the top, there are tabs: Новость, Анонс, Подробно (which is selected), Группы, Дополнительно, and Документооборот. Below the tabs, the title "Детальная информация" is displayed. The main content area contains the following text:

С 25.04.2007 по 27.04.2007 в Берлине (Германия) пройдет торговая выставка книгопечатания.

Организаторы выставки: **Messe Berlin GmbH**

**#BANNER\_BOTTOM#**

Рубрика: **Рекламная и издательская деятельность**

Место проведения: **Messegelande**

The visual editor toolbar is visible at the top, featuring icons for image, text, HTML, and visual editor selection. Below the toolbar, there are dropdown menus for CSS class, format, font, size, and style options, along with standard rich text editing tools like bold, italic, underline, and horizontal rule.

Используйте **result\_modifier.php**, который следует поместить рядом с соответствующим шаблоном показа новости или статьи. Шаблон компонента предварительно скопирован в текущий шаблон сайта:

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
$arResult["DETAIL_TEXT"] = preg_replace(
```



```
"/#BANNER_([A-Za-z-0-9]+)#/e",
'CAdvBanner::GetHTML(CAdvBanner::GetRandom("\1")),
$arResult["DETAIL_TEXT"]
);
?>
```

В итоге на сайте это будет выглядеть следующим образом:

## Выставка "PostPrint - 2007"

С 25.04.2007 по 27.04.2007 в Берлине (Германия) пройдет торговая выставка книгопечатания.

Организаторы выставки: **Messe Berlin GmbH**



Рубрика: Рекламная и издательская деятельность

Место проведения: **Messegelände**

Выставка **PostPrint** сосредотачивается на печати и распределении. Преимущество состоит в том, что всестороннее понятие выставки теперь включает все стадии технологического процесса печати (планирование, цифровая печать, бумага, распределение).

Основные тематические разделы:

- Маркировка
- Обработка бумаги
- Печать
- Программное обеспечение

## Result\_modifier и component\_epilog одновременно

Задача не частая, но встречающаяся: разместить в теле элемента информационного блока какой-либо компонент. Например, опрос. Попробуем реализовать эту задачу.

- Обозначьте маркер для замены на текущий опрос. Пусть этот маркер будет #VOTE\_ID\_XX#, где XX это ID нужного нам опроса. Этот маркер вставьте в тело новости в нужное место:



Текст HTML Визуальный редактор

Normal (Шрифт) small В И У

В состав жюри входят редакторы журнала, писатели-фантасты. Каждый член жюри оценивает работу по десятибалльной системе. Победителями считаются работы, набравшие наибольшее количество баллов.

Победителям конкурса будут вручены призы от журнала "Самиздат".

А теперь небольшой опрос:

#VOTE\_ID\_2#

**Свойства**

<body><p>

Стиль: (Стиль) Оформление:

Шрифт: (Шрифт) Размер:

Цвет текста: Цвет фона:

- Настройте компонент (в нашем случае это **bitrix:voting.current**) на отдельной странице так, как вам нужно. Деталь: отключите работу в AJAX-режиме.
- Скопируйте шаблон новости в свой шаблон для редактирования. Откройте шаблон для детального просмотра и создайте два файла: **result\_modifier.php** и **component\_epilog.php**:

**result\_modifier.php** такого содержания:

```
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!=true)die();?>
<$this->__component->SetResultCacheKeys(array("CACHED_TPL"));?>
component_epilog.php такого:
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!=true)die();?>
<
echo preg_replace_callback(
    "/#VOTE_ID_(\d+)/#is".BX_UTF_PCRE_MODIFIER,
    create_function('$matches', 'ob_start();
/*component here*/
    $retrunStr = @ob_get_contents();
    ob_get_clean();
    return $retrunStr;');
    $arResult["CACHED_TPL"]);
?>
```



- Вместо `/*component here*/` в **component\_epilog.php** вставьте вызов нашего компонента (для большей наглядности этот код пишем отдельно от общего кода):

```
$GLOBALS["APPLICATION"]->IncludeComponent(
    "bitrix:voting.current",
    "main_page",
    Array(
        "CHANNEL_SID" => "ANKETA",
        "VOTE_ID" => $matches[1],
        "CACHE_TYPE" => "A",
        "CACHE_TIME" => "3600",
        "AJAX_MODE" => "N",
        "AJAX_OPTION_SHADOW" => "Y",
        "AJAX_OPTION_JUMP" => "Y",
        "AJAX_OPTION_STYLE" => "Y",
        "AJAX_OPTION_HISTORY" => "N",
    )
);
```

**⚠ Примечание:** Обратите внимание, что вместо обычного `$APPLICATION` написано `$GLOBALS["APPLICATION"]`. Так надо для видимости объекта внутри временной функции. В остальном это полностью код компонента `bitrix:voting.current`.

И обратите внимание на `$matches[1]`. Это единственный динамический параметр в вызываемом компоненте. Динамический в том плане, что он будет зависеть от того какой маркер мы меняем. Для `#VOTE_ID_1#` он будет равен 1, для `#VOTE_ID_2#` 2 и так далее.

- Теперь надо изменить **template.php**. На второй строчке впишите конструкцию:

```
<?ob_start();?>
```

а в самом конце:

```
<?
$this->__component->arResult["CACHED_TPL"] = @ob_get_contents();
ob_get_clean();
?>
```

Манипуляции с **component\_epilog.php** сделаны чтобы обойти кеширование.

Что получили в итоге:



Победителям конкурса будут вручены призы от журнала "Самииздат".

А теперь небольшой опрос:

**Книги какого жанра вы предпочитаете?**

- фантастика
- фэнтези
- религия и философия
- историческая
- детектив
- триллеры
- юмор
- другое

**Голосовать**

[Возврат к списку](#)

В этом примере от файла *result\_modifier.php* можно избавиться совсем (лишняя файловая операция все же). То есть \$component->SetResultCacheKeys(array("CACHED\_TPL")); можно добавить прямо в файл **template.php**. Здесь **result\_modifier.php** был создан только для следования академическим правилам написания кода в **Bitrix Framework**.

**Пример с пустым компонентом**

**Типичная ошибка**

Вот как выглядит типичная попытка начинающего разработчика осуществить выборку из Информационного блока в явном виде в теле страницы:

```
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Прямой вызов API Битрикса на странице");
?>
<h1><?=$APPLICATION->ShowTitle()?></h1>
<div class="box">
<?
$IBLOCK_ID = intval($arParams['IBLOCK_ID']);
if ($IBLOCK_ID <=0) $IBLOCK_ID = 1;
```



```
if(!CModule::IncludeModule("iblock"))
    die('iblock module is not included!');
//делаем выборку из Инфоблока
$arSort = Array("SORT"=>"ASC", "NAME"=>"ASC");
$arFilter = Array("IBLOCK_ID"=>$IBLOCK_ID, "ACTIVE"=>"Y");
$obIBlockResult = CIBlockElement::GetList($arSort, $arFilter);

//выводим результат выборки в виде списка
echo "<ol>";
while($arFields = $obIBlockResult->GetNext())
{
    echo "<li>{$arFields["NAME"]}</li>";
}
echo "</ol>";
?>
</div>
<?require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");?>
```

### Чем же плохо такое решение?

Основных проблем несколько:

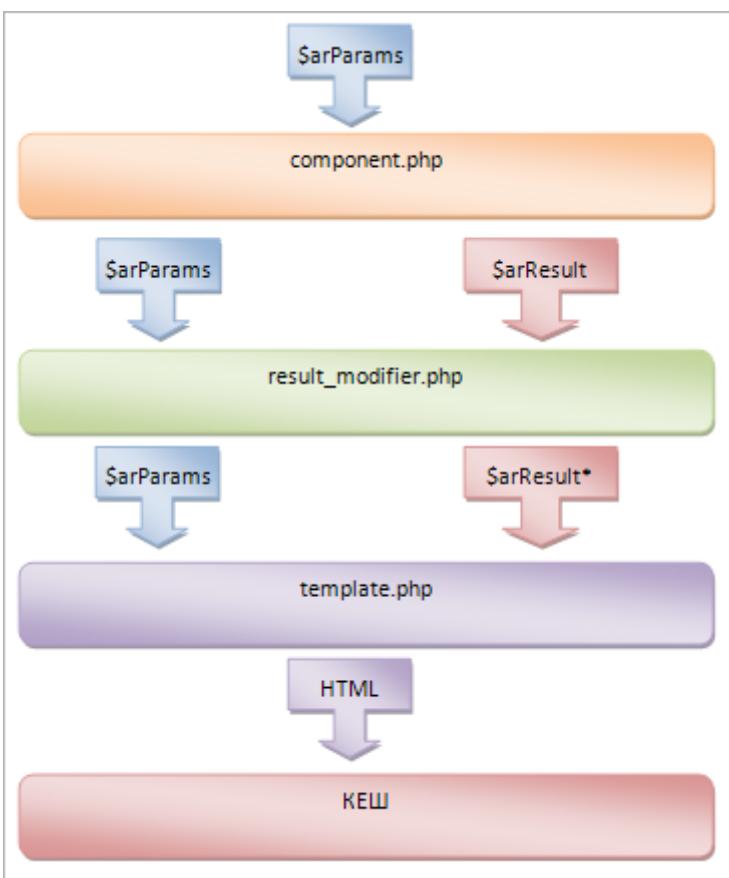
- отсутствие кеширования на такой странице (в отличие от компонента);
- отсутствие внятной и явной передачи параметров (в отличие от компонента);
- риск открытия страницы визуальным редактором с заменой спецсимволов, приводящей к неработающему коду на странице.

Эти проблемы можно решить поместив код в шаблон пустого компонента, не потеряв при этом в скорости разработки за счет использования правильных инструментов.

### Переход на использование компонента

Предлагается использовать **result\_modifier.php**, который находится в папке шаблона компонента и изначально предназначен для модификации массива **\$arResult** перед его выводом в файле шаблона **template.php**.

Рассмотрим классическую схему работы компонента с **result\_modifier.php** в шаблоне, представленную на рисунке:



В данном случае, в `result_modifier.php` будет помещен непосредственно код, который до этого планировалось разместить прямо на странице.

Общая схема решения выглядит следующим образом:

- На странице вместо кода с вызовами API помещается вызов пустого компонента `system.empty` со специальным шаблоном (например, `get_list`)
- В этом шаблоне компонента создается файл `result_modifier.php`, в который помещается необходимый код с вызовами API. Файл `template.php` оставляется пустым. (Но в дальнейшем возможно его использование стандартным образом – для этого в `result_modifier.php` нужно будет сформировать массив `$arResult`, который и будет выводиться в шаблоне.)
- В результате, на странице остается только вызов компонента, параметры которому можно передавать через массив параметров функции [IncludeComponent\(\)](#)

Пример кода страницы:

```
<?
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/header.php");
$APPLICATION->SetTitle("Компонентный подход: вызов API Битрикса в шаблоне компонента");
```

```

?>
<h1><?=$APPLICATION->ShowTitle()?></h1>
<div class="box">
<?
//задаем Инфоблок
$IBLOCK_ID = 1;

//подключаем пустой компонент, где логика лежит в шаблоне get_list в файле
result_modifier.php
$APPLICATION->IncludeComponent(
    "bitrixonrails:system.empty",
    "get_list",
    Array(
        "IBLOCK_ID" => $IBLOCK_ID,
        "CACHE_TYPE" => "A",
        "CACHE_TIME" => "3600"
    ),
false
);?>

</div>
<?require($_SERVER["DOCUMENT_ROOT"]."/bitrix/footer.php");?>

```

Перенесенный в **result\_modifier.php** исходный код:

```

<?
//получаем идентификатор Инфоблока из параметров компонента
$IBLOCK_ID = intval($arParams['IBLOCK_ID']);
if ($IBLOCK_ID <= 0) $IBLOCK_ID = 1;

if(!CModule::IncludeModule("iblock"))
    die('iblock module is not included!');
//делаем выборку из Инфоблока
$arSort = Array("SORT"=>"ASC", "NAME"=>"ASC");
$arFilter = Array("IBLOCK_ID"=>$IBLOCK_ID, "ACTIVE"=>"Y");
$obIBlockResult = CIBlockElement::GetList($arSort, $arFilter);

//выводим результат выборки в виде списка

```



```
echo "<ol>";
while($arFields = $obIBlockResult->GetNext())
{
    echo "<li>{$arFields["NAME"]}</li>";
}
echo "</ol>";
?>
```

Если необходимы и другие вставки кода, то просто создается новый шаблон (можно просто копированием шаблона **get\_list**) и в его **result\_modifier.php** вставляется нужный код. В результате получается множество шаблонов ("квази-компонентов"), каждый из которых соответствует исходной странице.

### С помощью файла component\_epilog

Часто возникают задачи, которым мешает кэширование шаблонов компонентов. Но только ради исключения шаблона из кэша не хочется кастомизировать компонент и от кэширования результата компонента отказываться тоже не хочется. Самый распространенный пример – голосование за элементы инфоблоков в списках или вывод рекламы.

Идея решения проста: переместить шаблон компонента в эпилог компонента:

- Скопировать шаблон компонента в адресное пространство шаблона сайта.
- В папке шаблона компонента создать файл **component\_epilog.php** и полностью скопировать в него код из файла шаблона **template.php**. Затем в самом верху **component\_epilog.php** сразу после проверки подключения эпилога ядра `<?if (!defined("B_PROLOG_INCLUDED")smile;)` || `B_PROLOG_INCLUDED!==true)die();?>` перед началом непосредственно самого кода шаблона нужно добавить такой код:

```
<?
// заменяем $arResult эпилога значением, сохраненным в шаблоне
if(isset($arResult['arResult'])) {
    $arResult =& $arResult['arResult'];
    // подключаем языковой файл
    global $MESS;
    include_once(GetLangFileName(dirname(__FILE__).'/lang/', '/template.php'));
} else {
    return;
}
```



```
?>  
Полностью очистить файл template.php и добавить такой код:  
<arResultCacheKeys)) {  
    $component->arResultCacheKeys = array();  
}  
$sVarName = 'arResult';  
$component->arResultCacheKeys[] = $sVarName;  
$component->arResult[$sVarName] = $$sVarName;  
}
```

Теперь результат компонента кэшируется, а шаблон – нет.

Недостатки этого способа:

- Увеличение размера кэша. С размером кэша можно бороться путем включения в кэш только необходимых данных
- Шаблон все же постоянно генерирует html-код, что медленнее, чем вывод уже готового, ранее сформированного html-кода.

## Использование событий

Для дополнения и неявного изменения (без вмешательство в код компонента) логики работы используйте технологию Событий. Рассмотрим пример использования событий.

### Прямое назначение ответственного в Техподдержке

Есть потребность напрямую назначать ответственного за обращение в техподдержку. Такое бывает, когда сотрудники компании хорошо знают друг друга, либо когда клиенты ТП желают общаться с конкретным сотрудником.

Есть два варианта решения данной задачи:

- Модифицируется код стандартного компонента **support.wizard** в файлах **component.php** и **.description.php**. Также, для большей гибкости, можно установить переменную-флаг в файле **.parameters.php**.

Такой подход для непосвященного в такие понятия, как События в **Bitrix Framework** наиболее очевиден. В результате, задача резко усложняется - нужно разбираться в логике кода компонента, чтобы понять, где нужно добавить свой функционал, а также, не стоит забывать о том, что **Bitrix Framework** постоянно обновляется. Решение довольно трудоёмкое.

- Достаточно будет использовать События и модифицировать шаблон.



Теория Событий вам знакома. Вкратце повторимся: в код ядра, обычно в начало и конец вызова системной функции, разработчики уже вставили вызов системной функции, в нашем случае это: **CTicket::ExecuteEvents**. Если мы хотим дополнить эту функцию своей, то в файле `/bitrix/php_interface/init.php` пишем:

```
AddEventHandler("<имя_модуля>","<событие>","<имя_функции>");
```

Если вызываемая функция принадлежит классу, то нужно вместо `<имя_функции>` написать `array("<имя_класса>","<имя_функции>")`. Код файла `init.php`:

```
<?
// В массиве $arFields передаются все параметры, которые были переданы созданному тикету + ID и MID тикета
// $_REQUEST["PERSONAL"] - Значение, которое мы получаем из нашего шаблона (тега select)
function AfterTicketAdd($arFields)
{
    if ($_REQUEST["PERSONAL"]>0)
    {
        //Добавляем к уже созданному тикету свойство "RESPONSIBLE_USER_ID"=>
        CTicket::Set(array("RESPONSIBLE_USER_ID"=>$_REQUEST["PERSONAL"]),$intMessage,$arFields['ID'], "N");
    }
}
?>
```

`$intMessage` - ID нового сообщения, нам не нужно, "N" - здесь указывается, что права на добавление нам проверять не надо.

### Модификация шаблона

- Скопируйте шаблон системного компонента
- Добавьте в шаблон следующий код:

```
<select name="PERSONAL" id="PERSONAL"/>
<option><?=GetMessage("Select_any")?></option>
<?
    // CTicket::GetSupportTeamList() - список всех сотрудников техподдержки
    $dbSupportTeamList=CTicket::GetSupportTeamList();
    while($arPersonal=$dbSupportTeamList->Fetch()):?>
```



```
<option  
value="<?=$arPersonal["REFERENCE_ID"]?>"><?=$arPersonal["REFERENCE"]?></option  
>  
<?endwhile?>  
</select>
```

- При необходимости допишите локализацию на вашем родном языке.
- Добавьте шаблон в файл **ticket\_edit.php**.

## Кастомизация компонентов

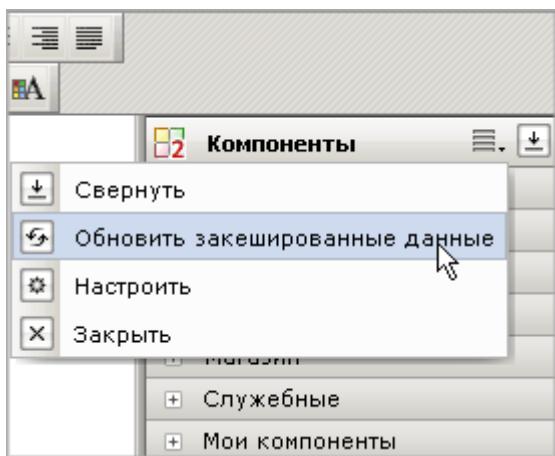
**Кастомизация стандартного компонента** - копирование стандартного компонента в собственное пространство имён и изменение его логики работы с целью изменения/добавления функционала.

Большинство задач в **Bitrix Framework** реализуется через компоненты, и в шаблоне компонента вы оперируете массивами **\$arResult** - это результат работы компонента (данные) и **\$arParams** - это входные параметры.

Чтобы кастомизировать стандартный компонент необходимо:

- Создать новое пространство имён компонентов в папке **/bitrix/components/**, например создать директорию **/bitrix/components/my\_components/**.
- В созданную папку необходимо скопировать папку с компонентом, который хотите изменить (копировать из папки **/bitrix/components/bitrix/**).
- Изменить компонент под текущие задачи.
  - изменить описание компонента на свое в файлах **.description.php** и **/lang/ru/.description.php**;
  - исправить файлы **.parameters.php** и **component.php**, модифицировав (добавив необходимый) функционал с помощью API продукта;
- Отредактировать шаблон компонента под текущие задачи.
- Очистите кеш визуального редактора. В результате в визуальном редакторе отобразится кастомизированный компонент.

**⚠ Примечание:** Обновление кеша визуального редактора делается на закладке **Компоненты**:



### Простой пример кастомизации компонента

Компонент **news.list** при большом числе элементов может существенно тормозить генерацию страницы. Задача – оптимизировать работу компонента. Одним из вариантов оптимизации может стать удаление ссылки на детальный текст новости в виде части текста (останется ссылка в виде названия новости).

- Скопируйте компонент в свое пространство имен.
- В коде скопированного компонента удалите строки:

```
"DETAIL_TEXT",
"DETAIL_TEXT_TYPE",
```

и

```
if($bGetProperty)
    $arSelect[]="PROPERTY_*";
```

- Сохраните внесенные изменения.
- Примените вместо стандартного свойственный компонент.

Мы получим в результате этих действий большее число запросов к БД, но меньшее время формирования страницы.

### Модификация простого компонента в составе сложного

При работе с комплексным компонентом можно модифицировать один или несколько простых, остальные остаются стандартные.

Например, необходимо в компоненте **socialnetwork.user\_groups**, который в составе комплексного компонента **socialnetwork\_group** выводит список групп, увеличить длину



выводимого описания группы с 50 до 100 символов (специально выберем простую модификацию чтобы акцентировать внимание на самом процессе).

- Скопируйте шаблон комплексного компонента.

Теперь в шаблоне сайта имеем шаблон комплексного компонента, перейдя в который увидим большой набор файлов в папке `/bitrix/templates/<шаблон сайта>/components/bitrix/socialnetwork_group/.default`.

	Имя	Размер файла	Изменен
	..		
	bitrix		14.03.2010 18:41:51
	images		14.03.2010 18:41:51
	lang		14.03.2010 18:40:55
	auth.php	100 Б	14.03.2010 18:41:51
	group.php	4 КБ	14.03.2010 18:41:51
	group_ban.php	4 КБ	14.03.2010 18:41:51
	group_blog.php	5 КБ	16.03.2010 01:38:21
	group_blog_draft.php	5 КБ	14.03.2010 18:41:51
	group_blog_post.php	6 КБ	14.03.2010 18:41:51
	group_blog_post_edit.php	4 КБ	14.03.2010 18:41:51
	group_blog_post_rss.php	876 Б	14.03.2010 18:41:51
	group_blog_rss.php	866 Б	14.03.2010 18:41:51
	group_calendar.php	5 КБ	14.03.2010 18:41:51
	group_create.php	1 КБ	14.03.2010 18:41:51
	group_delete.php	2 КБ	14.03.2010 18:41:51
	group_edit.php	2 КБ	14.03.2010 18:41:51
	group_features.php	2 КБ	14.03.2010 18:41:51
	group_files.php	3 КБ	14.03.2010 18:41:51
	group_files_element.php	4 КБ	14.03.2010 18:41:51
	group_files_element_edit.php	3 КБ	14.03.2010 18:41:51
Выбрано: 64		Отмечено: 0	

Каждый из файлов вызывается на определённой странице социальной сети и подключает требуемые простые компоненты.

Теперь надо найти файл, который подключает тот компонент, который нужно изменить. В нашем случае это `index.php`. Остальные файлы в шаблоне комплексного компонента, расположенного в шаблоне сайта, можно удалить. Комплексный компонент будет подключать эти файлы из ядра. А значит, они будут обновляться.



		Имя	Размер файла	Изменен
		..		
		images		16.03.2010 00:38:53
		lang		16.03.2010 00:38:53
		index.php	3 КБ	16.03.2010 00:46:33
		style.css	17 КБ	16.03.2010 00:38:53
Выбрано: 4		Отмечено: 0		

- Теперь в оставшемся файле заменяем

```
$APPLICATION->IncludeComponent(
    "bitrix:socialnetwork.user_groups",
```

на

```
$APPLICATION->IncludeComponent(
    "custom:socialnetwork.user_groups",
```

- Копируем папку /bitrix/components/bitrix/socialnetwork.user\_groups в /bitrix/components/custom/socialnetwork.user\_groups.
- в файле /bitrix/components/custom/socialnetwork.user\_groups/component.php заменяем

```
"GROUP_DESCRIPTION" => SubStr($arGroups["GROUP_DESCRIPTION"], 0, 50)."...",
```

на

```
"GROUP_DESCRIPTION" => SubStr($arGroups["GROUP_DESCRIPTION"], 0, 100)."...",
```

Теперь весь функционал социальной сети остаётся стандартный, кроме компонента **socialnetwork.user\_groups**.

## Создание компонентов

### Цитатник веб-разработчиков.

**Роман Петров:** Методы, которые были актуальны 2 года назад - сейчас могут быть неактуальны из-за появления готовых компонентов, делающих то же самое

Фактически, на сегодняшний момент свой компонент нужно писать лишь тогда, когда нужен абсолютно новый функционал для сайта. Если учесть тот факт, что состав



стандартных компонентов довольно большой, то в большинстве случаев написание компонентов и не требуется, достаточно расширить функционал уже имеющихся.

Тем не менее, приходит время, когда разработчик должен научиться создавать свои компоненты.

### Типовая последовательность действий

- В веб-проекте, при составлении ТЗ и проектировании, выявляют и описывают возможные виды собственных компонентов.
- Определяется пространство имен собственных компонентов, например, с использованием названия проекта. Системные компоненты **Bitrix Framework** размещены в пространстве имен **bitrix**, компоненты проекта могут размещаться в пространстве имен, к примеру, **citybank**.

**⚠ Внимание!** Названия создаваемых компонентов не должны пересекаться со стандартными названиями.

- Определяется, какой стандартный компонент можно взять за основу для создания собственного компонента. В коде стандартных компонентов много примеров типичного и правильного использования API и техник программирования, поэтому рекомендуется брать их за основу.
- К каждому компоненту 2.0 продумывается интерфейс - какие параметры компонента должны быть доступны администратору веб-сайта для редактирования. Например, для компонента, отображающего прогноз погоды, можно в настройки для администратора вынести свойство **Адрес веб-сервиса** и **Таймаут соединения с веб-сервисом** и т.п.
- Определяется, в каком разделе дерева компонентов в визуальном редакторе необходимо разместить данный компонент.
- Компонент кодируется. Особое внимание уделяется настройке автокеширования компонента и профилированию его работы - он не должен выполнять запросы к базе данных в режиме кэширования, выполняет минимальное количество запросов к базе данных при устаревании кэша, хранит в кэше только необходимые данные, использует минимально возможный объем оперативной памяти (не сортирует массивы размером с десятки-сотни мегабайт и т.п.).

### Порядок создания собственно компонента

Выделить необходимый php-код в отдельный файл для того, чтобы использовать его потом в виде вызываемого файла не сложно. Но компонент еще нужно подключить в систему с помощью файла описания, который опознается ядром **Bitrix Framework**, в результате чего пользователь видит в визуальном редакторе иконку с названием компонента и может настраивать его свойства.

Напомним, что компонент – это выделенный в отдельный файл php-код с законченной функциональностью, файл регистрации компонента в системе и описания его параметров, а также файлы локализации.

- Регистрация компонента
  - Выделение необходимого php-кода в отдельный файл.
  - Создание файла описания **.description.php**
  - Размещение файлов в папке в собственном пространстве имен.
- Задание параметров в коде компонента
- Локализация
  - Подготовка файлов с текстовыми константами для компонента и файла регистрации: /lang/ru/<имя\_компонента>/<имя\_компонента>.php и /lang/ru/<имя\_компонента>/.description.php
  - Внесение изменения в код обоих файлов компонента для использования этих констант (подключение файла локализации делается при помощи функции [IncludeTemplateLangFile](#)).

 **Совет от Максима Месилова:**

*Если на сайте используются компоненты собственной разработки и требуется их большое обновление, то рекомендуется использовать в префиксе номер версии: news.list.v2 так вы не поломаете старые компоненты по всему сайту и у вас будет возможность сделать новый компонент с потерей обратной совместимости.*

Созданные для веб-проекта собственные компоненты могут использоваться как основа для новых компонентов, а также, по причине модульной структуры и "отчуждаемости от проекта" - эффективно использоваться в других веб-решениях.

**Дополнительные методы, доступные в компонентах и шаблонах**

В компонентах и шаблонах можно использовать дополнительные методы из класса [CComponentEngine](#).

```
string CComponentEngine::MakePathFromTemplate ($pageTemplate, $arParams);
```

где:

**\$pageTemplate** - шаблон вида /catalog/#IBLOCK\_ID#/section/#SECTION\_ID#.php или catalog.php?BID=#IBLOCK\_ID#&SID=#SECTION\_ID#,

**\$arParams** - ассоциативный массив замен параметров, в котором ключ - это название параметра, а значение - это значение параметра. Возвращает путь на основании шаблона пути **\$pageTemplate** и массива замен.

Пример:

```
$url = CComponentEngine::MakePathFromTemplate
```



```
("/catalog/#IBLOCK_ID#/section/#SECTION_ID#.php",
array(
    "IBLOCK_ID" => 21,
    "SECTION_ID" => 452
)
);
```

## Организация явной связи между компонентами на одной странице комплексного компонента

Явную связь между компонентами можно организовывать через возвращаемые значения и входящие параметры этих компонентов.

Если из компонента **comp1** нужно передать данные в компонент **comp2**, то в конце кода компонента **comp1** нужно написать: `return` данные;

Подключить компонент **comp1** нужно следующим образом:

```
$result = $APPLICATION->IncludeComponent("comp1", ...);
```

Теперь данные находятся в переменной **\$result** и их можно передать входящими параметрами в другой компонент **comp2**.

### Переопределение входящих переменных

Каждый компонент имеет набор переменных, в которых он принимает извне коды или другие атрибуты запрашиваемых данных. Например, компонент **bitrix:catalog.section** имеет переменные **IBLOCK\_ID** и **SECTION\_ID**, в которых он принимает и обрабатывает коды каталога и группы товаров соответственно.

Все компоненты, которые входят в состав комплексного компонента, должны иметь единообразный набор переменных. Например, комплексный компонент **bitrix:catalog** и все обычные компоненты (**bitrix:catalog.list**, **bitrix:catalog.section** и т.д.), которыми он управляет, работают с переменными **IBLOCK\_ID**, **SECTION\_ID**, **ELEMENT\_ID** и другими.

Если разработчик при размещении комплексного компонента на странице хочет переопределить переменные компонента, то он среди входящих параметров компонента должен задать параметр **VARIABLE\_ALIASES**.

При подключении компонента в режиме **SEF** (ЧПУ) этот параметр должен иметь вид:

```
"VARIABLE_ALIASES" => array(
    "list" => array(),
    "section" => array(
        "IBLOCK_ID" => "BID",
    )
);
```



```
"SECTION_ID" => "ID"
),
"element" => array(
"SECTION_ID" => "SID",
"ELEMENT_ID" => "ID"
),
)
```

Здесь коды массива соответствуют кодам в массиве шаблонов путей. Для каждого пути могут быть заданы свои переопределения переменных.

При подключении компонента **не** в режиме **SEF** (ЧПУ) этот параметр должен иметь вид:

```
"VARIABLE_ALIASES" => array(
    "IBLOCK_ID" => "BID",
    "SECTION_ID" => "GID",
    "ELEMENT_ID" => "ID",
)
```

Пример №1:

Пусть требуется, чтобы компонент **bitrix:catalog**, лежащий в файле **/fld/cat.php**, работал с путями:

- /catalog/index.php – для списка каталогов,
- /catalog/section/код\_группы.php?ID=код\_каталога – для группы товаров,
- /catalog/element/код\_товара.php?ID=код\_группы – для детальной информации о товаре.

Во входящих параметрах подключения компонента должны быть установлены следующие параметры:

```
"SEF_MODE" => "Y",
"SEF_FOLDER" => "/catalog/",
"SEF_URL_TEMPLATES" => array(
    "list" => "index.php",
    "section" => "section/#SECTION_ID#.php?ID=#IBLOCK_ID#",
    "element" => "element/#ELEMENT_ID#.php?ID=#SECTION_ID#"
),
"VARIABLE_ALIASES" => array(
    "list" => array(),
```



```
"section" => array(
    "IBLOCK_ID" => "ID"),
"element" => array(
    "SECTION_ID" => "ID",),
```

### Пример №2:

Пусть требуется, чтобы компонент **bitrix:catalog**, лежащий в файле `/fld/cat.php`, работал с путями

- `/fld/cat.php` – для списка каталогов,
- `/fld/cat.php?BID=код_каталога&SID=код_группы` – для группы товаров,
- `/fld/cat.php?ID=код_товара&SID=код_группы` – для детальной информации о товаре.

Во входящих параметрах подключения компонента должны быть установлены следующие параметры:

```
"SEF_MODE" => "N",
"VARIABLE_ALIASES" => array(
    "IBLOCK_ID" => "BID",
    "SECTION_ID" => "SID",
    "ELEMENT_ID" => "ID",
),
```

### Пользовательские движки шаблонизации

Компоненты могут работать с любыми движками шаблонизации, которые могут быть подключены из PHP. Чтобы добавить новый движок шаблонизации на сайт необходимо определить (или дополнить) глобальную переменную **\$arCustomTemplateEngines** в файле `/bitrix/php_interface/init.php`. В этой переменной содержится ассоциативный массив, каждый элемент которого имеет вид:

```
"код_шаблонизатора" => array(
    "templateExt" => array("расширение1", "расширение2" ...),
    "function" => "имя_функции_подключения_движка"
)
```

где:

- **код\_шаблонизатора** - произвольное уникальное в рамках сайта слово,
- **расширениеN** - расширение файла, который должен обрабатываться этим движком шаблонизации,



- **имя\_функции\_подключения\_движка** - имя функции, которая будет вызываться, если шаблон компонента имеет указанное расширение. Функцию можно разместить в этом же файле /bitrix/php\_interface/init.php.

Например, если на сайте кроме стандартного движка шаблонизации (PHP) требуется использовать **Smarty**, то в файл /bitrix/php\_interface/init.php необходимо добавить следующий код:

```
global $arCustomTemplateEngines;  
$arCustomTemplateEngines = array(  
    "smarty" => array(  
        "templateExt" => array("tpl"),  
        "function" => "SmartyEngine"  
    ),  
);
```

Тогда при подключении шаблона с расширением **tpl** будет запускаться не стандартный движок PHP, а функция SmartyEngine, которая должна подключить движок **Smarty**.

Синтаксис функций подключения движков следующий:

```
function имя_функции_подключения_движка($templateFile, $arResult, $arParams,  
$arLangMessages, $templateFolder, $parentTemplateFolder, $template)
```

где:

- \$templateFile – путь к файлу шаблона относительно корня сайта,
- \$arResult – массив результатов работы компонента,
- \$arParams – массив входных параметров компонента,
- \$arLangMessages – массив языковых сообщений (переводов) шаблона,
- \$templateFolder – путь к папке шаблона относительно корня сайта (если шаблон лежит не в папке, то эта переменная пуста),
- \$parentTemplateFolder - путь относительно корня сайта к папке шаблона комплексного компонента, в составе которого подключается данный компонент (если компонент подключается самостоятельно, то эта переменная пуста),
- \$template – объект шаблона.

Код функции подключения движка шаблонизации зависит от подключаемого движка.

### Полный пример подключения движка Smarty

В файл /bitrix/php\_interface/init.php необходимо добавить код:



```
global $arCustomTemplateEngines;
$arCustomTemplateEngines = array(
    "smarty" => array(
        "templateExt" => array("tpl"),
        "function" => "SmartyEngine"
    )
);

function SmartyEngine($templateFile, $arResult, $arParams, $arLangMessages,
$templateFolder, $parentTemplateFolder, $template)
{
    if (!defined("SMARTY_DIR"))
        define("SMARTY_DIR", "<абсолютный путь к движку Smarty>/libs/");

    require_once('<абсолютный путь к движку Smarty>/libs/Smarty.class.php');

    $smarty = new Smarty;

    $smarty->compile_dir = "<абсолютный путь к движку Smarty>/templates_c/";
    $smarty->config_dir = "<абсолютный путь к движку Smarty>/configs/";
    $smarty->template_dir = "<абсолютный путь к движку Smarty>/templates/";
    $smarty->cache_dir = "<абсолютный путь к движку Smarty>/cache/";

    $smarty->compile_check = true;
    $smarty->debugging = false;

    $smarty->assign("arResult", $arResult);
    $smarty->assign("arParams", $arParams);
    $smarty->assign("MESS", $arLangMessages);
    $smarty->assign("templateFolder", $templateFolder);
    $smarty->assign("parentTemplateFolder", $parentTemplateFolder);

    $smarty->display($_SERVER["DOCUMENT_ROOT"].$templateFile);
}
```

Строчку <абсолютный путь к движку Smarty> нужно везде заменить на абсолютный путь к движку Smarty. Подробности по установке движка на сайт есть в системе помощи по Smarty.



В примере кода в массиве **\$arCustomTemplateEngines** регистрируется движок Smarty. В функции **SmartyEngine** инициализируются параметры движка в соответствии с требованиями системы (см. документацию [Smarty](#)). Далее в Smarty передаются переменные результатов работы компонента, входных параметров, языковых сообщений и т.д. И в конце вызывается метод обработки и показа шаблона Smarty.

### Полный пример подключения движка XML/XSLT

В файл `/bitrix/php_interface/init.php` необходимо добавить код:

```
global $arCustomTemplateEngines;
$arCustomTemplateEngines = array(
    "xslt" => array(
        "templateExt" => array("xsl"),
        "function" => "XSLTEngine"
    ),
);

function CreateXMLFromArray($xDoc, $xNode, $ar)
{
    foreach($ar as $key=>$val)
    {
        if(!is_string($key) || strlen($key)<=0)
            $key = "value";

        $xElem = $xDoc->createElement($key);
        if(is_array($val))
        {
            CreateXMLFromArray($xDoc, $xElem, $val);
        }
        else
        {
            $xElem->appendChild($xDoc->createTextNode(iconv(SITE_CHARSET, "utf-8", $val)));
        }
        $xNode->appendChild($xElem);
    }
    return $xNode;
}
```



```
function XSLTEngine($templateFile, $arResult, $arParams, $arLangMessages, $templateFolder,
$parentTemplateFolder, $template)
{
    $arResult["PARAMS"] = array(
        "templateFolder" => $templateFolder,
        "parentTemplateFolder" => $parentTemplateFolder,
        "arParams" => $arParams,
        "arLangMessages" => $arLangMessages
    );

    $xDoc = new DOMDocument("1.0", SITE_CHARSET);
    $xRoot = $xDoc->createElement('result');
    CreateXMLFromArray($xDoc, $xRoot, $arResult);
    $xDoc->appendChild($xRoot);

    $xXsl = new DOMDocument();
    $xXsl->load($_SERVER["DOCUMENT_ROOT"].$templateFile);

    $xProc = new XSLTProcessor;
    $xProc->importStyleSheet($xXsl);

    echo $xProc->transformToXML($xDoc);
}
```

### Работа комплексного компонента в SEF режиме

В комплексные компоненты встроена функция генерации ЧПУ. У этих компонентов всегда есть входной параметр **SEF\_MODE**, который может принимать значения **Y** и **N**. Если параметр **SEF\_MODE** равен **N**, то компонент работает с физическими ссылками и все параметры передает через стандартные параметры HTTP запроса. Например:

```
/fld/cat.php?IBLOCK_ID=12&SECTION_ID=371.
```

Если параметр **SEF\_MODE** равен **Y**, то компонент генерирует и обрабатывает ссылки на основании шаблонов. Например, он может понять и обработать ссылку:

```
/catalog/section/371.php?IBLOCK_ID=12, даже если сам он лежит в файле /fld/cat.php.
```

Если параметр **SEF\_MODE** равен **Y**, то у компонента должен так же присутствовать параметр **SEF\_FOLDER**, который должен содержать путь до папки, с которой работает компонент. Этот путь может как совпадать с физическим путем, так и не совпадать.



Например, в компоненте **bitrix:catalog**, подключенном в файле `/fld/cat.php`, могут быть установлены параметры **SEF\_MODE = Y** и **SEF\_FOLDER=/catalog/**. Тогда компонент будет отвечать на запросы по пути `/catalog/...`. По умолчанию редактор должен устанавливать текущий физический путь к редактируемому файлу.

У комплексного компонента, который может работать в режиме SEF, должен быть определен набор шаблонов путей **по умолчанию**. Например, в комплексном компоненте **bitrix:catalog** может быть определен следующий массив:

```
$arDefaultUrlTemplatesSEF = array(
    "list" => "index.php",
    "section" => "section.php?IBLOCK_ID=#IBLOCK_ID#&SECTION_ID=#SECTION_ID#",
    "element" => "element.php?ELEMENT_ID=#ELEMENT_ID#"
);
```

Эти шаблоны путей могут быть переопределены с помощью входного параметра комплексного компонента **SEF\_URL\_TEMPLATES**, содержащего новый массив всех шаблонов путей или их части.

При сохранении страницы с компонентом, работающим в SEF режиме, в редакторе создается или обновляется запись в системе **urlrewrite**. Например, при сохранении файла `/fld/cat.php`, в котором лежит компонент **bitrix:catalog**, переключенный в SEF режиме с параметром **SEF\_FOLDER=/catalog/**, в системе **urlrewrite** создается или обновляется запись типа:

```
array(
    "CONDITION" => "#^/catalog/#",
    "ID" => "bitrix:catalog",
    "PATH" => "/fld/cat.php"
),
```

- в **CONDITION** записывается значение параметра **SEF\_FOLDER**, обрамленное символами **#^** и **#**;
- в **ID** записывается название компонента;
- в **PATH** записывается физический путь к файлу, который сохраняется.

Если запись с таким **PATH** и **ID** уже есть, то она обновляется, если нет – добавляется.

В **run-time** при запросе физически не существующей страницы механизм **urlrewrite** производит поиск соответствующей записи по **CONDITION** и передает управление на страницу **PATH**.

Компонент на странице **PATH** на основании шаблонов путей выясняет запрашиваемую страницу и восстанавливает переменные, спрятанные в пути.

**⚠ Внимание!** Обязательное требование к набору шаблонов путей данного компонента - это уникальность каждого шаблона пути без учета параметров и переменных. Это должно проверяться при сохранении страницы в визуальном редакторе.

То есть, набор шаблонов путей:

```
"section" => "section/#SECTION_ID#.php?IBLOCK_ID=#IBLOCK_ID#",
"element" => "element/#ELEMENT_ID#.php"
```

является допустимым, а набор шаблонов путей:

```
"section" => "#SECTION_ID#.php?IBLOCK_ID=#IBLOCK_ID#",
"element" => "#ELEMENT_ID#.php"
```

допустимым не является.

### Простой пример создания компонента

В качестве примера сделаем компонент, который выводит текущую дату и время. Причем формат вывода даты и времени задается выбором в свойствах. В реальных условиях такой компонент будет неинтересен. Но нам важно понять, как выполняется разработка компонента. В более сложных случаях все реализуется похожим образом.

#### **Предварительные действия**

Готовим php-код компонента. Первым делом сразу напишем php-код, который выполняет, то, что нам нужно.

```
<?
echo date('Y-m-d');
?>
```

Правда код просто выводит дату и нельзя выбрать другой формат. Лучше поместить в переменную формат вывода даты:

```
<?
$format = 'Y-m-d';
echo date($format);
?>
```

И последний штрих — нужно разделить логику и представление:

```
<?
// параметры
```



```
$format = 'Y-m-d';
// логика
$d = date($format);
// представление
echo $d;
?>
```

### Создаем структуру папок и файлов компонента

Теперь нужно создать свое пространство имен, например: dv. Для этого надо создать папку /bitrix/components/dv. В ней делаем папку компонента — **date.current**. И в ней, в свою очередь, создаем два обязательных файла и папку для хранения шаблонов **templates**. В папке templates должна быть создана папка **.default** и в ней файл **template.php**.

Получаем такую структуру в папке /bitrix/components/dv/date.current:

- component.php
- .description.php
- templates/.default/template.php

### Реализуем компонент без входных параметров

Пока сделаем компонент без возможности задания входного параметра — формата даты.

Содержимое файлов:

- component.php

```
<? if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
$arResult['DATE'] = date('Y-m-d');
$this->IncludeComponentTemplate();
?>
```

- .description.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
$arComponentDescription = array(
    "NAME" => GetMessage("Текущая дата"),
    "DESCRIPTION" => GetMessage("Выводим текущую дату"),
);
?>
```

- templates/.default/template.php



```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
```

Как вы могли заметить в каждом из файлов компонента в начале пишется строка `if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();`. Она нужна для того, чтобы данные файлы нельзя было вызвать напрямую из окна браузера.

В простейшем виде компонент готов — его можно вызывать в коде страниц при помощи конструкции:

```
<? $APPLICATION->IncludeComponent(
    "dv:date.current",
    ".default",
    Array(
    ),
    false
);?>
```

### Реализуем компонент с входными параметрами

Теперь давайте сделаем, чтобы компонент можно было добавлять на страницу из визуального редактора, и чтобы можно было задавать шаблон выдачи даты в настройках компонента.

Чтобы наш компонент появился в визуальном редакторе нужно дополнить файл описания компонента.

.description.php:

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
$arComponentDescription = array(
    "NAME" => GetMessage("Текущая дата"),
    "DESCRIPTION" => GetMessage("Выvodim текущую дату"),
    "PATH" => array(
        "ID" => "dv_components",
        "CHILD" => array(
            "ID" => "curdate",
            "NAME" => "Текущая дата"
        )
    ),
    "ICON" => "/images/icon.gif",
);
?>
```



Для размещения компонента в дереве компонентов мы добавили элемент массива описания PATH. Таким образом, наш компонент будет показан в отдельной папке. Опционально, можно задать иконку компонента — она будет показываться в дереве и в визуальном редакторе.

Разберемся с настройками компонента. Будем считать, что опцию шаблон даты мы будем задавать строкой. Создаем файл **.parameters.php** с таким содержанием:

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
$arParams = array(
"GROUPS" => array(),
"PARAMETERS" => array(
"TEMPLATE_FOR_DATE" => array(
"PARENT" => "BASE",
"NAME" => "Шаблон для даты",
"TYPE" => "STRING",
"MULTIPLE" => "N",
"DEFAULT" => "Y-m-d",
"REFRESH" => "Y",
),
),
);
?>
```

И изменяем файл с логикой компонента, чтобы он мог использовать параметр, который мы задаем **component.php**:

```
<? if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
$arResult['DATE'] = date($arParams["TEMPLATE_FOR_DATE"]);
$this->IncludeComponentTemplate();
?>
```

### Что же мы сделали?

Мы создали компонент, в самом простом виде. Мы не учитывали мультиязычность, не учитывали возможность создания помощи к компоненту и прелести кеширования компонентов.

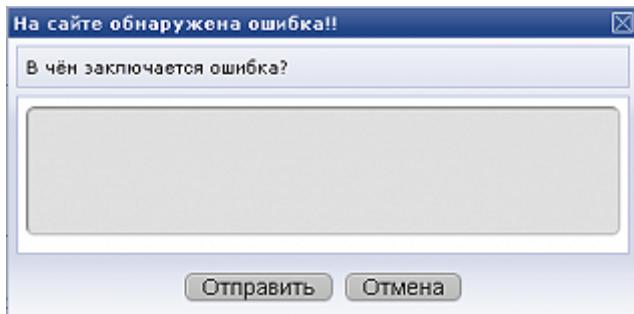
Большая часть заказных компонентов для **Bitrix Framework** создается путем изменения компонентов, идущих в поставке продуктов. Поэтому очень нужно хорошо изучить стандартные компоненты, перед тем как программировать новые. Скорее всего, задачу, которую вы хотите решить — уже решили разработчики компании 1С-Битрикс.



## Пример создания компонента

Рассмотрим пример создания компонента для сообщений администратору об ошибке.

С помощью этого компонента можно реализовать функционал, который бы позволял пользователям сообщать ответственным за контент о найденной на сайте ошибке. Ошибка будет высыпаться почтовым уведомлением. Алгоритм работы пользователя с компонентом очень простой: если пользователь находит на портале ошибку, то он выделяет текст, нажимает *Ctrl+Enter* и получает форму:



Форма заполняется и сообщение отправляется ответственному лицу.

## **Создание почтового шаблона**

Так как сообщение об ошибке будет отправлено на почту, то потребуется создать новый почтовый тип.

- Перейдите на страницу *Настройки > Настройки продукта > Почтовые события > Типы почтовых событий*.
- Заполните поля формы:



Тип события

Тип почтового события

\* Тип почтового события:

[ru] Russian

Сортировка:

Название:

Описание:  
#ERROR\_MESSAGE# – текст ошибки  
#ERROR\_DESCRIPTION# – описание ошибки  
#ERROR\_URL# – URL страницы с ошибкой

[en] English

- Перейдите на страницу [Настройки > Настройки продукта > Почтовые события > Почтовые шаблоны](#).
- Нажмите в контекстной панели на **Добавить шаблон**, откроется форма создания шаблона.
- Задайте шаблон для созданного типа почтового события (Рисунок 46):



**Параметры**

**Параметры почтового шаблона**

Активен:

\*Сайт:  [s1] Моя компания

Тип почтового события:

\*От кого:

\*Кому:

[показать дополнительные заголовки...](#)

Тема:

Тип тела сообщения:  Текст /  HTML

**Тело сообщения:**

Текст ошибки:  
`#ERROR_MESSAGE#`

Описание ошибки:  
`#ERROR_DESCRIPTION#`

URL страницы с ошибкой:  
`#ERROR_URL#`

**Доступные поля:**

[#ERROR\\_MESSAGE#](#) - текст ошибки  
[#ERROR\\_DESCRIPTION#](#) - описание ошибки  
[#ERROR\\_URL#](#) - URL страницы с ошибкой  
[#DEFAULT\\_EMAIL\\_FROM#](#) - E-Mail адрес по умолчанию  
[#SITE\\_NAME#](#) - Название сайта (устанавливается в настройках)  
[#SERVER\\_NAME#](#) - URL сервера (устанавливается в настройках)

## Создание компонента

Создайте в собственном пространстве имен папку **feedback.error** со следующей структурой:

- папка **images**
  - файл **feedback.gif**
- папка **templates**
  - папка **.default**
    - файл **script.js**



- файл template.php
- файл .description.php
- файл .parameters.php
- файл component.php

Файл **feedback.gif** - иконка, которая будет отображаться в визуальном редакторе.

код файла **script.js**:

```
BX.bind(document, "keypress", SendError);

function SendError(event, formElem)
{
    event = event || window.event;

    if((event.ctrlKey) && ((event.keyCode == 0xA) || (event.keyCode == 0xD)))
    {
        var Dialog = new BX.CDialog({
            title: "На сайте обнаружена
ошибка!!",
            head: "В чём заключается
ошибка?",
            content: '<form method="POST"
id="help_form">\n
<textarea
name="error_desc" style="height: 78px; width: 374px;"></textarea>\n
<input
type="hidden" name="error_message" value="'+getSelectedText()+'"\n
type="hidden" name="error_url" value="'+window.location+'\n
type="hidden" name="error_referer" value="'+document.referrer+'\n
type="hidden" name="error_useragent" value="'+navigator.userAgent+'\n
type="hidden" name="sessid" value="'+BX.bitrix_sessid()+'></form>',
            resizable: false,
            height: '198',
            width: '400'});
    }
}
```



```
Dialog.SetButtons([
{
    'title': 'Отправить',
    'id': 'action_send',
    'name': 'action_send',
    'action': function(){
        BX.ajax.submit(BX("help_form"));
        this.parentWindow.Close();
    }
},
{
    'title': 'Отмена',
    'id': 'cancel',
    'name': 'cancel',
    'action': function(){
        this.parentWindow.Close();
    }
]);
Dialog.Show();
}

function getSelectedText(){
if (window.getSelection){
    txt = window.getSelection();
}
else if (document.getSelection) {
    txt = document.getSelection();
}
else if (document.selection){
    txt = document.selection.createRange().text;
}
else return;
return txt;
}
```

код файла template.php:

```
<?
CUtil::InitJSCore(array('window', 'ajax'));
?>
```

код файла .description.php:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentDescription = array(
    "NAME" => "Send Error",
    "DESCRIPTION" => "Send Error",
    "ICON" => "/images/feedback.gif",
    "PATH" => array(
        "ID" => "utility",
    ),
);
?>
```

код файла .parameters.php:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arComponentParameters = array();
?>
```

код файла component.php:

```
<?
if (check_bitrix_sessid() && $_SERVER['REQUEST_METHOD'] == "POST" &&
!empty($_REQUEST["error_message"]) && !empty($_REQUEST["error_url"]))
{
    $arMailFields = Array();
    $arMailFields["ERROR_MESSAGE"] = trim($_REQUEST["error_message"]);
    $arMailFields["ERROR_DESCRIPTION"] = trim($_REQUEST["error_url"]);
    $arMailFields["ERROR_URL"] = $_REQUEST["error_desc"];
    $arMailFields["ERROR_REFERER"] = $_REQUEST["error_referer"];
}
```



```
$arMailFields["ERROR_USERAGENT"] = $_REQUEST["error_useragent"];  
  
        CEvent::Send("BX", SITE_ID, $arMailFields);  
}  
$this->IncludeComponentTemplate();  
?>
```

Теперь подробнее для разработчиков о том, что находится в файле feedback.error\templates\.default\script.js.

Объявление функции-обработчика, которая выводится на <body>:

```
function SendError(event, formElem)
```

Ожидание нажатия Ctrl+Enter:

```
if((event.ctrlKey) && ((event.keyCode == 0xA) || (event.keyCode == 0xD)))
```

Определение параметров будущего окна и его содержимого:

```
var Dialog = new BX.CDialog({  
    title: "На сайте обнаружена ошибка!!",  
    head: "В чём заключается ошибка?",  
    content: 'action="/bitrix/templates/.default/send_error.php">\n            <textarea name="error_desc" style="height: 78px; width:  
374px;"></textarea>\n            <input type="hidden" name="error_message" value="'+getSelectedText()+'"\n            <input type="hidden" name="error_url" value="'+window.location+'"\n            <input type="hidden" name="sessid" value="'+BX.bitrix_sessid()+'"\n        ',  
    resizable: false,  
    height: '198',  
    width: '400'});
```

Определение набора кнопок:

```
Dialog.SetButtons([  
{  
    'title': 'Отправить',  
    'id': 'action_send',  
    'name': 'action_send',
```



```
'action': function(){
    BX.ajax.submit(BX("help_form"));
    this.parentWindow.Close();
},
{
    'title': 'Отмена',
    'id': 'cancel',
    'name': 'cancel',
    'action': function(){
        this.parentWindow.Close();
    }
},
]);
});
```

Вывод окна:

```
Dialog.Show();
```

Функция **getSelectedText()** получает выделенный мышью текст. И далее идет отправка письма в тексте файла **component.php**:

```
if (check_bitrix_sessid() && $_SERVER['REQUEST_METHOD'] == "POST" &&
!empty($_REQUEST["error_message"]) && !empty($_REQUEST["error_url"]))
{
    $arMailFields = Array();
    $arMailFields["ERROR_MESSAGE"] = trim($_REQUEST["error_message"]);
    $arMailFields["ERROR_DESCRIPTION"] = trim($_REQUEST["error_desc"]);
    $arMailFields["ERROR_URL"] = trim($_REQUEST["error_url"]);
    CEvent::Send("BX", SITE_ID, $arMailFields);
};
```

### Компонент интеграции визуального редактора

Создадим компонент, интегрирующий популярный редактор **TinyMCE** в **Bitrix Framework**.

Загрузите [свежую версию](#) редактора с сайта производителя.

В своем пространстве имен создайте структуру папок и файлов:

- /bitrix/components/tools/;



- /bitrix/components/tools/editor.tiny.mce/;
  - /bitrix/components/tools/editor.tiny.mce/templates/;
    - /bitrix/components/tools/editor.tiny.mce/templates/.default/;
  - /bitrix/components/tools/editor.tiny.mce/tiny\_mce/ — папка для дистрибутива редактора;
  - **componet.php** — логика компонента;
  - **.parameters.php** — файл для описания входящих параметров.

Скопируйте скаченный дистрибутив в папку /tiny\_mce.

В файл **component.php** добавляем код:

```
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();  
$APPLICATION->AddHeadScript($this->_path .'/tiny_mce/tiny_mce.js');  
  
$sNameTextAria = (isset($arParams['TEXTARIA_NAME']) == false) ? 'content' :  
$arParams['TEXTARIA_NAME'];  
$sIdTextAria = (isset($arParams['TEXTARIA_ID']) == false) ? "" :  
$arParams['TEXTARIA_ID'];  
$sEditorID = (isset($arParams['INIT_ID']) == false) ? 'textareas' :  
$arParams['INIT_ID'];  
$iTextariaWidth = (isset($arParams['TEXTARIA_WIDTH']) == false) ? '100%' :  
$arParams['TEXTARIA_WIDTH'];  
$iTextariaHeight = (isset($arParams['TEXTARIA_HEIGHT']) == false) ? '300' :  
$arParams['TEXTARIA_HEIGHT'];  
$sText = (isset($arParams['TEXT']) == false) ? "" :  
$arParams['TEXT'];  
?  
  
<script type="text/javascript">  
<?  
if($arParams['TYPE_EDITOR'] == 'TYPE_1')  
{  
?>  
tinyMCE.init(  
{  
language : 'ru',  
mode : "textareas",  
//elements : "<?=$sEditorID?>",
```

```

        editor_selector : "<?=$sEditorID?>",
        theme   : "advanced",
        plugins  : "safari, spellchecker, upload.images.komka, wordcount,
fullscreen",
        theme_advanced_buttons1 :
"formatselect,fontselect,fontsizeselect,bold,italic,underline,link,justifyleft,justifycenter,justifyright,pasteword,pastetext,images,|,bullist,numlist,|,undo,redo,|,spellchecker,fullscreen",

        theme_advanced_buttons2 : "",
        theme_advanced_buttons3 : "",
        theme_advanced_toolbar_location : "top",
        theme_advanced_toolbar_align   : "left",
        theme_advanced_statusbar_location : "bottom",
        theme_advanced_resizing       : false,
        content_css                  : "<?=$this->__path?>/example.css",
        height : "<?=$iTextariaHeight?>",
        spellchecker_languages : '+Русский=ru,English=en',
        spellchecker_word_separator_chars : '\s!\"#$%&()/*,-./;<=>?@[\\]^_{}'
    }
);

<?
}

elseif($arParams['TYPE_EDITOR'] == 'TYPE_2')
{
    ?>
    tinyMCE.init({
        language : 'ru',
        mode   : "textareas",
        editor_selector : "<?=$sEditorID?>",
        theme   : "advanced",
        plugins
:
"safari,spellchecker,pagebreak,style,layer,table,save,advhr,advimage,advlink,emotions,iespell,inlinepopups,insertdatetime,preview,media,searchreplace,print,contextmenu,paste,directionality,fullscreen,noneditable,visualchars,nonbreaking,xhtmlxtras,template,imagemanager,filemanager",

```



```
        theme_advanced_buttons1 :  
"save,newdocument,|,bold,italic,underline,strikethrough,|,justifyleft,justifycenter,justifyright,ju  
stifyfull,|,styleselect,formatselect,fontsizeselect",  
        theme_advanced_buttons2 :  
"cut,copy,paste,pastetext,pasteword,|,search,replace,|,bullist,numlist,|,outdent,indent,blockqu  
ote,|,undo,redo,|,link,unlink,anchor,image,cleanup,help,code,|,insertdate,inserttime,preview,|,  
forecolor,backcolor",  
        theme_advanced_buttons3 :  
"tablecontrols,|,hr,removeformat,visualaid,|,sub,sup,|,charmap,emotions,iespell,media,advhr,|  
,print,|,ltr,rtl,|,fullscreen",  
        theme_advanced_buttons4 :  
"insertlayer,moveforward,movebackward,absolute,|,styleprops,spellchecker,|,cite,abbr,acrony  
m,del,ins,attribs,|,visualchars,nonbreaking,template,blockquote,pagebreak,|,insertfile,insertim  
age",  
        theme_advanced_toolbar_location : "top",  
        theme_advanced_toolbar_align : "left",  
        theme_advanced_statusbar_location : "bottom",  
        theme_advanced_resizing : true,  
  
content_css : "<?=$this->__path?>/example.css",  
height : "<?=$iTextariaHeight?>",  
  
template_external_list_url : "js/template_list.js",  
external_link_list_url : "js/link_list.js",  
external_image_list_url : "js/image_list.js",  
media_external_list_url : "js/media_list.js",  
template_replace_values : {username : "Some User", staffid :  
"991234"}  
}  
};  
IncludeComponentTemplate();?>
```

Файл .parameters.php

```
<? if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();  
  
$arComponentParameters = array(  
    "PARAMETERS" => array(  
        "TYPE_EDITOR" => Array(
```



```
"PARENT" => "SETTINGS",
"NAME" => "Режим редактора",
"TYPE" => "LIST",
"VALUES" => array('TYPE_1' => 'Упрощенные редактор', 'TYPE_2' => 'Полной
редактор'),
),

'INIT_ID' => array(
"PARENT" => "SETTINGS",
"NAME" => "ID редактора (уникальный)",
"TYPE" => "STRING",
"DEFAULT" => '',
),

'TEXT' => array(
"PARENT" => "SETTINGS",
"NAME" => "Контент который нужно вставить в редактор",
"TYPE" => "STRING",
"DEFAULT" => $_POST['content'],
),

'TEXTARIA_NAME' => array(
"PARENT" => "SETTINGS",
"NAME" => "Имя поля TEXTARIA",
"TYPE" => "STRING",
"DEFAULT" => 'content',
),

'TEXTARIA_ID' => array(
"PARENT" => "SETTINGS",
"NAME" => "ID поля TEXTARIA",
"TYPE" => "STRING",
"DEFAULT" => 'content',
),

'TEXTARIA_WIDTH' => array(
"PARENT" => "SETTINGS",
```



```
"NAME" => "Ширина редактора",
"TYPE" => "STRING",
"DEFAULT" => '100%',
),

'TEXTARIA_HEIGHT' => array(
"PARENT" => "SETTINGS",
"NAME" => "Высота редактора",
"TYPE" => "STRING",
"DEFAULT" => '300',
),
)

);
/*
array(
'TEXT' => $_POST['content'], # контент, в
html
'TEXTARIA_NAME' => 'content', # имя поля
'TEXTARIA_ID' => 'content', # ID поля
'INIT_ID' => 'textareas', # ID
редактора
'TEXTARIA_WIDTH' => '100%', # понятно
'TEXTARIA_HEIGHT' => '300' # можно
понятно
)
*/
?>
```

Важные моменты в коде файла **component.php**, которые надо пояснить. Подключение собственно редактора:

```
<? $APPLICATION->AddHeadScript($this->__path .'/tiny_mce/tiny_mce.js'); ?>
```

Подключение стилей, которые вынесены в папку с компонентом чтобы было удобнее, это настройка уже в js, в коде инициализации:

```
content_css : '<?=$this->__path?>/example.css',
```



**⚠ Внимание!** если на странице 2 или более редакторов то мы идентифицируем их по имени класса `editor_selector` : '`<?=$sEditorID?>`'.

Собственно сама область текста для которой это все и делается:

```
<textarea id='<?=$sIdTextAria?>' name='<?=$sNameTextAria?>'  
style='width:<?=$iTextariaWidth?>'><?=$sText?></textarea>
```

## Как использовать

Подключается вот в таком виде:

```
<? echo $_POST['content'] ?>  
<? echo $_POST['content2'] ?>  
<form action="" method="post" name="">  
<? $APPLICATION->IncludeComponent("tools:editor.tiny.mce", ".default", array(  
"TEXT" => $_POST["content"], // контент из запроса который нужно вставить  
"TEXTARIA_NAME" => "content", // имя поля  
"TEXTARIA_ID" => "content", // id поля  
"TEXTARIA_WIDTH" => "100%", // ширина  
"TEXTARIA_HEIGHT" => "300", // высота  
  
"INIT_ID" => "ID" // ID самого редактора  
),  
false  
);  
?>  
<input value="submit" name="sub" type="submit" />  
</form>
```

## Кэширование в собственных компонентах

Казалось бы, проще всего обращаться напрямую через API в базу данных, получать информацию, форматировать ее в шаблоне компонента и отображать пользователю.

Все дело в производительности веб-проекта при одновременной работе с ним множества пользователей. Если компонент отрабатывает без кэширования за 0.1 сек, выполняя, допустим, 100 запросов к базе данных, то, при одновременной работе 100 пользователей не только резко возрастет нагрузка на сервер базы данных, но и время отработки компонента может вырасти, к примеру, до 5-10 секунд.

Не менее важный момент, на который стоит обратить внимание – скорость отработки компонента при получении данных из кэша. Если без кэширования компонент отрабатывает за 2 сек. для каждого пользователя, то при использовании кэширования компонент для одного пользователя отработает за 2 сек., а для остальных 100 пользователей в ближайшие полчаса, допустим, будет отрабатывать 0.1 сек.

При использовании кэширования в собственных компонентах 2.0:

- резко увеличивается производительность веб-проекта и его устойчивость к нагрузкам, т.к. нагрузка на базу данных качественно минимизируется и веб-решение сможет обслужить уже, к примеру, не 50 000 пользователей в сутки, а 1 000 000 и больше
- веб-страницы загружаются в браузер пользователя значительно быстрее (десятые доли секунды), т.к. информация для их построения сохранена на сервере и не берется из базы данных

### Время кэширования

Период времени кэширования зависит от типа кэширования. Если используется **Авто+Управляемое** кэширование – информация будет отдаваться из кэша до тех пор, пока она не поменяется в базе данных и кэш сбросится автоматически. Время кэширования для этого режима должно быть большим, к примеру, 1 год.

Если используется **Авто** кэширование – рекомендуется устанавливать максимально допустимый с учетом бизнес-логики интервал кэширования. Время кэширования для этого режима зависит от частоты обновления информации - для некоторых компонентов устанавливается период в 24 часа, а для часто обновляемых либо рекомендуется использовать управляемое кэширование или установить значение, к примеру , в 10 минут.

### Встроенная поддержка кэширования

В компонентах 2.0 есть встроенная поддержка типичного алгоритма кэширования. Структура компонента с использованием встроенной поддержки кэширования будет примерно такова:

```
// Проверка и инициализация входных параметров
if ($arParams["ID"] <= 0)
    $arParams["ID"] = 10;

// Если нет валидного кеша (то есть нужно запросить
// данные и сделать валидный кеш)
if ($this->StartResultCache())
{
    // Запрос данных и заполнение $arResult
```



```
$arResult = array(
    "ID" => rand(1, 100)
);

for ($i = 0; $i < 5; $i++)
    $arResult["FIELDS"][] = rand(1, 100);

// Если выполнилось какое-то условие, то кешировать
// данные не надо
if ($arParams["ID"] < 10)
    $this->AbortResultCache();

// Подключить шаблон вывода
$this->IncludeComponentTemplate();
}

// Установить заголовок страницы с помощью отложенной
// функции
$APPLICATION->SetTitle($arResult["ID"]);
```

### Пояснения по коду

Метод [StartResultCache](#) имеет следующее описание:

```
bool $this->StartResultCache($cacheTime = False, $additionalCacheID = False, $cachePath = False)
```

где:

- **\$cacheTime** - время кеширования (если **False** - подставляется `intval($arParams["CACHE_TIME"]);`);
- **\$additionalCacheID** - от чего дополнительно зависит кеш кроме текущего сайта **SITE\_ID**, имени компонента, пути в файлу и входных параметров;
- **\$arParams** (если **False**, то кеш зависит только от указанных параметров);
- **\$cachePath** - путь к файлу кеша (если **False** - подставляется `"/".SITE_ID.<путь к компоненту относительно bitrix/components>".`).

Если есть валидный кеш, то метод отправляет на экран его содержимое, заполняет **\$arResult** и возвращает **False**; если нет валидного кеша, то он возвращает **True**.



Если кеш зависит не только от сайта, входных параметров, имени компонента и пути к текущему сайту, но и от других параметров, то эти параметры в виде строки нужно передать в метод вторым параметром. Например, если кеш зависит еще от групп пользователей, в которые входит текущий посетитель, то условие нужно написать следующим образом:

```
if ($this->StartResultCache(false, $USER->GetGroups()))  
{  
    // Валидного кеша нет. Выбираем данные из  
    // базы в $arResult  
}
```

Если в результате выборки данных (в случае отсутствия валидного кеша) выяснилось, что кешировать данные не надо, то нужно вызвать метод `$this->AbortResultCache();`. Например, если выяснилось, что новости с таким ID нет, то нужно прервать кеширование и выдать сообщение, что такой новости нет. Если кеширование не прерывать, то злоумышленники смогут забить кешем все отведенное сайту дисковое пространство вызывая страницу с произвольными (в том числе и не существующими) ID.

Метод `$this->IncludeComponentTemplate();` подключает шаблон компонента и сохраняет в кеш-файл вывод и массив результатов `$arResult`. Все изменения `$arResult` и вывод после вызова метода подключения шаблона не будут сохранены в кеш.

Если при исполнении кода компонента мы не вошли в тело условия `if ($this->StartResultCache())`, то значит для данного компонента, страницы и входных параметров есть валидный кеш. После вызова этого метода HTML из кеша отправлен на вывод и мы имеем заполненный массив `$arResult`. Здесь можно выполнить некоторые действия. Например, установить заголовок страницы с помощью отложенных функций.

Если при выполнении некоторых условий нужно очистить кеш компонента (например, компонент знает, что данные изменились), то можно воспользоваться методом `$this->ClearResultCache($additionalCacheID = False, $cachePath = False)`. Параметры этого метода соответствуют одноименным параметрам метода `StartResultCache`.

## Сложное кеширование

Если компоненту требуется какое-либо особое кеширование, которое не может быть выполнено с помощью встроенной поддержки кеширования, то можно использовать стандартный класс [CPHPCache](#). Структура компонента с использованием класса [CPHPCache](#) будет примерно такова:

```
// Проверка и инициализация входных параметров  
if ($arParams["ID"] <= 0)  
    $arParams["ID"] = 10;
```



```
$arParams["CACHE_TIME"] = IntVal($arParams["CACHE_TIME"]);
$CACHE_ID = SITE_ID."|".$APPLICATION->GetCurPage()."|";
// Кеш зависит только от подготовленных параметров без "~"
foreach ($this->arParams as $k => $v)
    if (strcmp("~", $k, 1))
        $CACHE_ID .= ".$k."=".$v;
$CACHE_ID .= "|".$USER->GetGroups();

$cache = new CPageCache;
if    ($cache->StartDataCache($arParams["CACHE_TIME"],      $CACHE_ID,      "/".SITE_ID.$this->GetRelativePath()))
{
    // Запрос данных и формирование массива $arResult
    $arResult = array("a" => 1, "b" => 2);

    // Подключение шаблона компонента
    $this->IncludeComponentTemplate();

    $templateCachedData = $this->GetTemplateCachedData();

    $cache->EndDataCache(
        array(
            "arResult" => $arResult,
            "templateCachedData" => $templateCachedData
        )
    );
}
else
{
    extract($cache->GetVars());
    $this->SetTemplateCachedData($templateCachedData);
}
```

### Пояснения по коду

Кеш должен зависеть только от подготовленных параметров. То есть от параметров, которые инициализированы нужным образом, приведены к нужному типу (например, с помощью `intval()`) и т.д. В массиве `$arParams` содержатся как подготовленные параметры, так и исходные параметры (с тем же ключем, но с префиксом "~"). Если кеш

будет зависеть от неподготовленных параметров, то злоумышленники смогут забить кешем все отведенное сайту дисковое пространство вызывая страницу с ID равными "8a", "8b", ... (которые после **intval()** дадут 8).

Метод `$this->IncludeComponentTemplate()` не запрашивает данные из базы. Но его тоже лучше включить в кешируемую область, так как этот метод производит определенные дисковые операции.

Перед вызовом метода завершения кеширования и сохранения кеша (метод **EndDataCache**) необходимо запросить у шаблона параметры, которые должны быть использованы даже в том случае, если сам шаблон не подключается и данные берутся из кеша. В текущей реализации такими параметрами являются стили css шаблона, которые подключаются отложенными функциями, а значит не попадают в кеш. Структура возвращаемых шаблоном данных не документирована и не имеет значения для компонента. Это просто какие-то данные, которые нужно положить в кеш, а затем взять из кеша и вернуть в шаблону.

Чтобы вернуть шаблону те данные, которые он просил сохранить в кеше, можно пользоваться методами `$this->SetTemplateCachedData($templateCachedData);` или `CBitrixComponentTemplate::ApplyCachedData($templateCachedData);`. Один из этих методов должен быть вызван в той области компонента, которая выполняется в случае наличия валидного кеша. В параметрах ему должны быть переданы те данные, которые шаблон просил сохранить.

## Ошибки при работе с компонентами

### Не удалось обнаружить код вызова компонента

Довольно распространенная ошибка, когда вы в режиме редактирования пытаетесь отредактировать параметры какого-то компонента на странице. Хоть в коде и присутствует строка `$APPLICATION->IncludeComponent()` (вызов компонента), всё равно иногда появляется ошибка **Не удалось обнаружить код вызова компонента**. К сожалению, универсального решения данной проблемы нет.

Возможные варианты решения:

- В **.htaccess** включить две строки:

для не-UTF:

```
php_value mbstring.func_overload 0  
php_value mbstring.internal_encoding latin1
```

для UTF:

```
php_value mbstring.func_overload 2  
php_value mbstring.internal_encoding UTF-8
```

- Возможно ошибка появляется из-за неправильной расстановки html-тегов (например, какой-то из тегов закрыт не в том месте, где надо);



- Убрать все html-комментарии со страницы;
- Заключить код вызова компонента в отдельные символы <? ?> (то есть, отделить от другого php-кода)
- Вставить такую конструкцию перед вызовом компонента: <?/\* \*/?>
- Удалить несколько аналогичных компонентов рядом с неработающим.

## Глава 8. Модули

**Bitrix Framework** имеет модульную структуру. Каждый модуль отвечает за управление определенными элементами и параметрами сайта: информационным наполнением и структурой сайта, форумами, рекламой, рассылкой, распределением прав между группами пользователей, сбором статистики посещений, оценкой эффективности рекламных кампаний и т.д.

*Модуль - это модель данных и API для доступа к этим данным. Статические методы классов модуля могут вызываться в компонентах, шаблонах, других модулях. Также внутри контекста **Bitrix Framework** могут создаваться экземпляры классов.*

Модули системы, главным образом, работают независимо друг от друга. Однако в целом ряде случаев функционал одних модулей основан на возможностях других. Например:

- Модуль **Торговый каталог** расширяет возможности модуля **Информационные блоки** и позволяет выполнять настройку цен товара в зависимости от различных условий, применять к товарам наценку и скидки и т.п.
- Модуль **Документооборот** позволяет организовать последовательную коллективную работу с содержимым модулей **Информационные блоки** и **Управление структурой**.

После установки системы список используемых модулей можно просмотреть на странице **Управление модулями** ([Настройки > Настройки продукта > Модули](#)) в административном разделе системы:



Управление модулями					
Название	Версия	Дата обновления	Статус	Действие	
<b>Главный модуль</b> Ядро продукта с технологией "SiteUpdate".	10.0.1	18.03.2011	Установлен		
<b>AD/LDAP интеграция</b> Модуль для работы с Active Directory и LDAP.	10.0.0	15.03.2011	Установлен	<button>Удалить</button>	
<b>Wiki</b> Модуль дает возможность ведения wiki-страниц на сайте.	10.0.0	09.03.2011	Установлен	<button>Удалить</button>	
<b>Бизнес-процессы</b> Модуль для создания и работы с бизнес-процессами	9.5.3	28.12.2010	Установлен	<button>Удалить</button>	
<b>Блоги</b> Модуль дает возможность ведения блогов на сайте.	10.0.1	09.03.2011	Установлен	<button>Удалить</button>	
<b>Валюты</b> Модуль управления валютами позволяет управлять валютами сайта и их курсами.	10.0.0	15.03.2011	Установлен	<button>Удалить</button>	
<b>Веб-аналитика</b> Модуль сбора и отображения статистики сайта.	10.0.1	23.03.2011	Установлен	<button>Удалить</button>	
<b>Веб-кластер</b> Модуль поддержки веб-кластера.	10.0.1	24.12.2010	Не установлен		<button>Установить</button>
<b>Веб-сервисы</b> Модуль позволяющий организовать систему веб-сервисов и SOAP.	10.0.0	15.03.2011	Установлен	<button>Удалить</button>	

Для экономии дискового пространства неиспользуемые модули рекомендуется удалить, при этом дистрибутив модуля остается в системе, и он в любое время может быть снова установлен. При деинсталляции некоторых модулей система предлагает сохранить накопленные модулем данные (таблицы модуля). Если вы в дальнейшем планируете использовать эти данные, то при удалении модуля необходимо отметить соответствующую опцию.

Управление уровнем прав пользователей на доступ к модулям системы осуществляется отдельно для каждого модуля на странице его настроек. На этой же странице выполняется управление общими параметрами работы модулей.

Страница настроек конкретного модуля может иметь различное число вкладок и полей, в зависимости от функционала модуля. Перейти к ней можно следующими способами:

- с помощью административного меню: [Настройки > Настройки продукта > Настройки модулей > имя\\_модуля](#);
- с помощью кнопки **Настройки**  , расположенной на административной панели. Данная кнопка позволяет перейти к настройкам модуля, страницы (формы) которого открыты в текущий момент в основной рабочей области.

**⚠ Примечание:** Перед использованием модуля необходимо проверить установлен ли он и подключить его при помощи конструкции:

```
<?
if(CModule::IncludeModule("*****"))
```

```
{  
//здесь можно использовать функции и классы модуля  
}  
?>
```

где \*\*\*\* - [идентификатор модуля](#).

## Модули и компоненты

Модули в **Bitrix Framework** представляют собой модели и контроллеры нижнего уровня (в понятиях **MVC**), а компоненты - контроллеры верхнего уровня, включающие представления, которые основаны на иерархии файловой структуры сайта. Весь функционал любого сайта, как правило, реализуется на стандартных модулях, но приходится кастомизировать компоненты (или написать свои) для формирования и вывода страниц и подключать их на соответствующих страницах сайта.

## Разработка модулей

Bitrix Framework допускает [разработку пользовательских модулей](#).

## Структура файлов

Структура файлов модуля:

- /bitrix/modules/*ID* модуля/ - корневой каталог модуля
  - /admin/ - каталог с административными скриптами модуля;
    - menu.php - файл с административным меню модуля;
  - /classes/ - скрипты с классами модуля;
    - /general/ - классы модуля, не зависящие от используемой базы данных;
    - /mysql/ - классы модуля, предназначенные для работы только с MySQL;
    - /mssql/ - классы модуля, предназначенные для работы только с MS SQL;
    - /oracle/ - классы модуля, предназначенные для работы только с Oracle;
  - /lang/*ID языка*/ - каталог с языковыми файлами скриптов модуля;
  - /install/ - каталог с файлами используемыми для инсталляции и деинсталляции модуля;
    - /admin/ - каталог со скриптами подключающими административные скрипты модуля (*вызывающие скрипты*);

- /js/ - каталог с js-скриптами модуля. Копируются в /bitrix/js/*ID\_модуля*/;
- /db/ - каталог с SQL скриптами для инсталляции/деинсталляции базы данных;
  - /mysql/ - SQL скрипты для инсталляции/деинсталляции таблиц в MySQL;
  - /mssql/ - SQL скрипты для инсталляции/деинсталляции таблиц в MS SQL;
  - /oracle/ - SQL скрипты для инсталляции/деинсталляции таблиц в Oracle;
- /images/ - каталог с изображениями используемыми модулем; после инсталляции модуля они должны быть скопированы в каталог /bitrix/images/*ID* модуля/;
- /templates/ - каталог с компонентами 1.0 модуля. (Каталог сохраняется только с целью совместимости версий.);
  - /*ID* модуля/ - каталог с основными файлами компонент;
  - /lang/*ID* языка/*ID* модуля/ - в данном каталоге находятся языковые файлы компонент модуля;
- /components/пространство имен/имя компонента/ - каталог с компонентами 2.0 модуля.
- **index.php** - файл с описанием модуля;
- **include.php** - данный файл подключается в тот момент, когда речь идет о подключении модуля в коде, в нем должны находиться включения всех файлов с библиотеками функций и классов модуля;
- **default\_option.php** - содержит массив с именем **\$ID модуля\_default\_option**, в котором заданы значения по умолчанию для параметров модуля;
- **options.php** - данный файл подключается на странице настройки параметров модулей в административном меню **Настройки**;
- **prolog.php** - файл может подключаться во всех административных скриптах модуля. Обычно в нем определяется константа **ADMIN\_MODULE\_NAME** (идентификатор модуля), используемая в панели управления.

## Описание и параметры

### Описание

Каждый модуль должен быть корректно описан в системе для того, чтобы система знала, как с этим модулем работать. Некорректно описанные модули могут привести к полной или частичной неработоспособности системы (например, может не работать система обновлений).



Основным файлом используемым системой для манипуляции модулем является **/bitrix/modules/*ID* модуля/install/index.php**. Основное назначение этого файла - это размещение в нем класса с именем, совпадающим с **ID модуля**.

Пример:

```
01  <?
02  Class mymodule extends CModule
03  {
04      var $MODULE_ID = "mymodule";
05      var $MODULE_NAME;
06
07      function DoInstall()
08      {
09          global $DB, $APPLICATION, $step;
10          $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/mymodule/install/step1.php");
11      }
12
13      function DoUninstall()
14      {
15          global $DB, $APPLICATION, $step;
16          $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/mymodule/install/unstep1.php");
17
18      }
19  }
20  ?>
```

Обязательные методы этого класса:

- `DoInstall` - запускается при нажатии кнопки **Установить** на странице **Модули** административного раздела, осуществляет инсталляцию модуля.
- `DoUninstall` - запускается при нажатии кнопки **Удалить** на странице **Модули** административного раздела, осуществляет deinсталляцию модуля.

Необязательный метод этого класса:

- `GetModuleRightList` - возвращает список уникальных прав (или ролей) модуля.

Обязательные свойства объекта этого класса:



- MODULE\_ID - хранит **ID модуля**;
- MODULE\_VERSION - текущая версия модуля в формате XX.XX.XX;
- MODULE\_VERSION\_DATE - строка содержащая дату версии модуля; дата должна быть задана в формате YYYY-MM-DD HH:MI:SS;
- MODULE\_NAME - **имя модуля**;
- MODULE\_DESCRIPTION - **описание модуля**;
- MODULE\_GROUP\_RIGHTS - если задан метод GetModuleRightList, то данное свойство должно содержать Y.

## Примеры

Пример файла с описанием модуля **Веб-формы**:

```
<?
global $MESS;
$PathInstall = str_replace("\\", "/", __FILE__);
$PathInstall = substr($PathInstall, 0, strlen($PathInstall)-strlen("/index.php"));
IncludeModuleLangFile($PathInstall."/install.php");
include($PathInstall."/version.php");
if(class_exists("form")) return;
Class form extends CModule
{
    var $MODULE_ID = "form";
    var $MODULE_VERSION;
    var $MODULE_VERSION_DATE;
    var $MODULE_NAME;
    var $MODULE_DESCRIPTION;
    var $MODULE_GROUP_RIGHTS = "Y";

    function form()
    {
        $this->MODULE_VERSION = FORM_VERSION;
        $this->MODULE_VERSION_DATE = FORM_VERSION_DATE;
        $this->MODULE_NAME = GetMessage("FORM_MODULE_NAME");
        $this->MODULE_DESCRIPTION = GetMessage("FORM_MODULE_DESCRIPTION");
    }

    function DoInstall()
```

```

{
global $DB, $APPLICATION, $step;
$FORM_RIGHT = $APPLICATION->GetGroupRight("form");
if ($FORM_RIGHT=="W")
{
    $step = IntVal($step);
    if($step<2)
        $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/step1.php");
    elseif($step==2)
        $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/step2.php");
}
}

function DoUninstall()
{
global $DB, $APPLICATION, $step;
$FORM_RIGHT = $APPLICATION->GetGroupRight("form");
if ($FORM_RIGHT=="W")
{
    $step = IntVal($step);
    if($step<2)
        $APPLICATION->IncludeAdminFile( GetMessage("FORM_UNINSTALL_TITLE"),
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/unstep1.php");
    elseif($step==2)
        $APPLICATION->IncludeAdminFile( GetMessage("FORM_UNINSTALL_TITLE"),
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/form/install/unstep2.php");
}
}

function GetModuleRightList()
{
global $MESS;
$arr = array(
    "reference_id" => array("D","R","W"),
    "reference" => array(

```



```
    GetMessage("FORM_DENIED"),
    GetMessage("FORM_OPENED"),
    GetMessage("FORM_FULL"))

};

return $arr;
}

?>
```

Пример файла с указанием версии модуля

```
<?
$arModuleVersion = array(
    "VERSION" => "11.0.4",
    "VERSION_DATE" => "2011-11-17 14:00:00"
);
?>
```

## Параметры

**Параметры модуля** доступны для изменения в административном интерфейсе на странице **Настройки модулей** ([Настройки > Настройки продукта > Настройки модулей](#)). При выборе модуля на данной странице, система подключает файл **/bitrix/modules/ID модуля/options.php**, предназначенный для управления параметрами модуля, назначения прав на модуль и т.п.

Параметры модуля хранятся в базе данных.

При получении параметров модуля, может использоваться значение по умолчанию, задаваемое в файле **/bitrix/modules/ID модуля/default\_option.php**. В данном файле определяется массив **\$ID модуля\_default\_option**, хранящий значения по умолчанию.

Пример файла **/bitrix/modules/ID модуля/default\_option.php**:

```
<?
$support_default_option = array(
    "SUPPORT_DIR"          => "#SITE_DIR#support/",
    "SUPPORT_MAX_FILESIZE" => "100",
    "ONLINE_INTERVAL"       => "900",
    "DEFAULT_VALUE_HIDDEN"  => "N",
    "NOT_IMAGE_EXTENSION_SUFFIX" => "_",
```



```
"NOT_IMAGE_UPLOAD_DIR" => "support/not_image",
"DEFAULT_AUTO_CLOSE_DAYS" => "7"
);
?>
```

Пример использования:

```
<?
// установим строковый параметр
COption::SetOptionString("my_module_id", "MY_PARAMETER_ID", "VALUE");

// получим строковый параметр
$value      =      COption::GetOptionString("my_module_id",      "MY_PARAMETER_ID",
"DEFAULT_VALUE");
?>
```

Для работы с параметрами модуля предназначен класс [COption](#).

## Административные скрипты

**Административные скрипты** - это скрипты используемые модулем в административной части системы. Они должны располагаться в каталоге `/bitrix/modules/ID модуля/admin/`.

Необходимо учитывать, что напрямую в браузере административные скрипты нельзя вызывать (как в общем-то и любые скрипты каталога `/bitrix/modules/`). Поэтому для их вызова используются дополнительные одноименные вызывающие скрипты находящиеся в каталоге `/bitrix/admin/`. Они, как правило, состоят только из подключения одноименного административного скрипта:

```
<?
require_once($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/ID      модуля/admin/имя
скрипта");
?>
```

При инсталляции модуля, вызывающие скрипты должны быть скопированы из каталога `/bitrix/modules/ID модуля/install/admin/` в каталог `/bitrix/admin/`. В момент deinсталляции модуля вызывающие скрипты должны быть удалены из этого каталога.



**!** *Примечание:* Необходимо учитывать, что вызывающие скрипты всех инсталлированных модулей находятся в одном каталоге /bitrix/admin/, поэтому во избежание дублирования, желательно давать им имена начинающиеся с какого-либо префикса характерного только для соответствующего модуля.

В каждом административном скрипте до подключения визуальной части административного пролога необходимо определять две константы, необходимые для формирования иконки над заголовком страницы:

- **ADMIN\_MODULE\_NAME** - идентификатор модуля (см. для примера [идентификаторы стандартных модулей](#));
- **ADMIN\_MODULE\_ICON** - HTML код для большой иконки модуля выводимой над заголовком страницы.

Пример определения таких констант:

```
define("ADMIN_MODULE_NAME", "statistic");
define("ADMIN_MODULE_ICON", "
");
```

### Языковые файлы для административных скриптов модуля

Языковые файлы должны располагаться в каталоге /bitrix/modules/*ID* модуля/lang/*ID* языка/. Особенностью их расположения внутри этого каталога является то, что они должны располагаться строго по тому же пути относительно каталога /bitrix/modules/*ID* модуля/, что одноименные файлы в которых они подключаются. В этом случае подключение языковых файлов может осуществляться только функцией [IncludeModuleLangFile](#).

К примеру для скрипта /bitrix/modules/*ID* модуля/admin/body/my\_script.php языковой файл должен быть расположен: /bitrix/modules/*ID* модуля/lang/*ID* языка/admin/body/my\_script.php.

И подключаться кодом:

```
IncludeModuleLangFile(__FILE__);
```

### Взаимодействие модулей

Модули могут взаимодействовать между собой двумя способами: явно (прямым вызовом) и скрыто (через систему событий).



## Явное взаимодействие

Явное взаимодействие подразумевает:

- Подключение модуля с помощью функции `CModule::IncludeModule`;
- Непосредственный вызов метода класса или функции модуля.

Пример явного взаимодействия:

```
<?
// подключаем модуль mymodule
if (CModule::IncludeModule("mymodule"))
{
    // выполним его метод
    CMyModuleClass::DoIt();
}
?>
```

## Взаимодействие через события

**Событие** - это какое либо произвольное действие, в момент выполнения которого (до или после) собираются все обработчики этого события и выполняются по одному.

Сущность **Событие** позволяет сделать модули максимально независимыми друг от друга. Модуль ничего не знает об особенностях функционирования другого модуля, но может взаимодействовать с ним через интерфейс событий.

Схема работы с событиями. Модуль, инициирующий событие, в том месте кода, где это событие происходит, должен выполнить следующее:

- Собрать все зарегистрированные обработчики с помощью функции `GetModuleEvents`.
- Выполнить их по одному с помощью функции `ExecuteModuleEvent`, обрабатывая соответствующим образом возвращаемые обработчиками значения.

В свою очередь, модуль, который хочет выполнить какие либо действия на это событие, должен:

- Зарегистрировать в момент инсталляции свой обработчик с помощью функции `RegisterModuleDependences`.
- Соответственно необходимо иметь эту функцию-обработчик и убедиться, что скрипт, в котором эта функция находится, подключается в файле `/bitrix/modules/ID модуля/include.php`.

## Пример взаимодействия

Ярким примером взаимодействия такого типа является взаимодействие модулей системы с модулем **Поиска**. Этот модуль не имеет никакой информации о данных других модулей, особенностях их хранения и обработки. Он только предоставляет интерфейс для индексации данных. Любой модуль системы, который должен индексироваться, при инсталляции регистрирует обработчик на событие `OnReindex`. Каждый такой обработчик в свою очередь возвращает данные для индексации, которые модуль **Поиска** использует для наполнения своей базы.

Коды примеров взаимодействия через события:

```
<?
// регистрация обработчика:
// когда в модуле init_module возникнет событие OnSomeEvent
// будет вызван метод CMyModuleClass::Handler модуля handler_module

RegisterModuleDependences(
    "init_module", "OnSomeEvent",
    "handler_module", "CMyModuleClass", "Handler"
);

?>
<?
// произвольная функция модуля init_module
// в которой генерируется событие

function MyFunction()
{
    // здесь располагается произвольный код
    // представляющий из себя событие

    // далее следует сбор зарегистрированных обработчиков
    $rsHandlers = GetModuleEvents("anothermodule", "OnSomeEvent");
    while($arHandler = $rsHandlers->Fetch())
    {
        // и их выполнение по одному
        if(!ExecuteModuleEvent($arHandler, $param1, $param2))
        {
            // если обработчик вернет false,
            // то например, вернем надпись "I can't do it..."
```



```
    return "I can't do it...";  
}  
}  
return "I have done it!";  
}  
?  
<?  
// обработчик  
  
class CMyModuleClass  
{  
    function Handler($param1, $param2)  
    {  
        if($param1=="delete all")  
            return false;  
        return true;  
    }  
}  
?>
```

## Установка и удаление

### Установка модуля

Инсталляция модуля осуществляется в административном интерфейсе на странице Настройки > Настройки продукта > Модули нажатием кнопки **Установить**. При этом будет вызван метод [DoInstall](#) класса с именем совпадающим с ID модуля. Этот класс должен быть описан в файле /bitrix/modules/*ID* модуля/install/index.php.

В процессе инсталляции модуля должны быть выполнены в обязательном порядке:

- Регистрация модуля, которая осуществляется с помощью функции [RegisterModule](#).
- Если модуль обладает административными скриптами, то для их вызова в каталог /bitrix/admin/ должны быть скопированы вызывающие скрипты.
- Все изображения, используемые модулем, должны быть скопированы в каталог /bitrix/images/*ID* модуля/.

## Удаление модуля

Деинсталляция модуля осуществляется нажатием кнопки **Удалить**. При этом будет вызван метод [DoUninstall](#) класса с именем совпадающим с ID модуля. Этот класс должен быть описан в файле `/bitrix/modules/ID модуля/install/index.php`.

В процессе деинсталляции модуля должны быть выполнены в обязательном порядке:

- Удаление регистрационной записи модуля, которая осуществляется с помощью функции [UnRegisterModule](#)
- Если модуль обладает административными скриптами, то вызывающие их скрипты должны быть удалены из каталога `/bitrix/admin/`.
- Все изображения, используемые модулем, должны быть удалены из каталога `/bitrix/images/ID модуля/`.

## Кастомизация и создание модулей

### Цитатник веб-разработчиков.

**Войтенко Андрей:** Скажу по своему опыту (3 года): проблемы, которые нельзя было решить без правки ядра, не встречал. Понятно, что не считая багов самого Битрикса. Они решались по принципу: ТП и ручная правка.

Создание нового модуля или изменение работы штатного модуля в **Bitrix Framework** требуется не часто, львиная доля проблем решается с помощью компонентов и их кастомизацией.

**⚠ Внимание!** Кастомизация модуля - это модификация ядра системы со всеми вытекающими отсюда последствиями: рисками получить неработоспособную систему после обновления, потерей права на ТП.

В силу этого кастомизация модуля - не рекомендуемая, чуть ли не запретная операция. Тем не менее, технически такая возможность имеется и в главе будет приведён пример не сложной кастомизации, если возникла такая потребность.

В Социальной сети разработчиков на **Bitrix Framework** есть [специальная группа](#), в которой можно пообщаться на тему создания собственных модулей.

## Пример изменения работы модуля

**⚠ Внимание!** Перед тем как начать модифицировать работу модуля (то есть модифицировать ядро системы) необходимо быть уверенными что иными средствами стоящую перед вами задачу не решить. При первом же обновлении продукта все добавленные вами изменения затрутся, и вам придется заново вносить их.

Решим задачу выгрузки контактов из корпоративного портала в **Outlook 2003**. Напомним, что в стандартной поставке «1С-Битрикс: Корпоративный портал» рассчитан на взаимодействие с **Outlook 2007**.

За формирование и подготовку данных для Outlook отвечает файл /bitrix/modules/intranet/classes/general/ws\_contacts.php.

Есть две стандартные проблемы:

- В **Outlook 2003** не выгружаются аватарки - вследствие чего сразу же возникает ошибка.
- Не для всех пользователей выгружается компания, к которой они принадлежат (поле в **Outlook - Организация**).

Решаем первую проблему с помощью функции \_\_getRow(\$arRes, \$listName, &\$last\_change). Комментируем строки установки атрибута картинки:

```
/*if ($this->bGetImages && $arRes['PERSONAL_PHOTO'] > 0)
{
    $arImage = CIntranetUtils::InitImage($arRes['PERSONAL_PHOTO'], 100, 100);

    $obRow->setAttribute('ows_Attachments', '#'.($APPLICATION->IsHTTPS() ? 'https://' :
'http://').$_SERVER['HTTP_HOST'].$arImage['CACHE']['src'].">#.CIntranetUtils::makeGUID(md5(
$arRes['PERSONAL_PHOTO'])).',1,#');

    $obRow->setAttribute('ows_MetaData_AttachProps', ".$arImage['FILE']['FILE_NAME'].");
}

else
{*/
    $obRow->setAttribute('ows_Attachments', 0);
//}
```

Решаем вторую проблему с помощью функции GetListItemChangesSinceToken(\$listName, \$viewFields = "", \$query = "", \$rowLimit = 0, \$changeToken = ""). В цикле

```
while ($arUser = $obUsers->NavNext())
```

комментируем все строки, начинающиеся с

```
$arUser['WORK_COMPANY']
```

(т.е. где просто меняется значение этого атрибута).

**⚠ Примечание:** В функции `GetList` происходит формирование схемы атрибутов. Если перед строчкой: `return array('GetListResult' => $data);` сделать вывод массива `$data`, то можно ознакомиться со схемой выгрузки.

## Создание собственных модулей

### Цитатник веб-разработчиков.

**Sergey Leshchenko:** Если код часто используется повторно, то его лучше вынести в модуль. Потратить на это лишние час-два, но зато не ловить потом фатальные баги сразу на всех проектах одновременно, когда кто-то, по случайности или не знанию, внес какие-то специфические изменения в этот код. И модуль вынести в маркетплейс, чтобы накатывать апдейты удобнее было.

Расширить функционал проектов на основе **Bitrix Framework** позволяют сторонние модули и решения.

## Создание модуля

Партнерские модули отличаются от стандартных модулей следующим:

- **код модуля** - полный код партнёрского модуля, который задается в формате **код\_партнера.код\_модуля**.  
Часть **код\_партнера** постоянна для партнёра (задается в карточке партнёра). Часть **код\_модуля** вводится партнёром при добавлении нового модуля. Эти коды должны быть алфавитно-цифровыми с первым алфавитным символом, и код некоторым образом должен соответствовать сути модуля. Например, для модуля форума желательно задать код **forum**. Тогда полный код будет **myscompany.forum**.
- В файле `/install/index.php` кроме той информации, которая задается в любом стандартном модуле, необходимо еще указать:

```
$this->PARTNER_NAME = "Имя партнера - автора модуля";  
$this->PARTNER_URI = "http://www.mysite.ru";
```

У клиента эта информация будет доступна в списке модулей.

**⚠ Внимание!** Модуль необходимо создавать в кодировке windows-1251, при установке его на сайт с кодировкой UTF-8 происходит автоматическая перекодировка.

Помните, что в **Bitrix Framework** принято, что версия не может быть равной 0, то есть 0.0.1 - минимальный номер версии.

### Инфоблоки или таблицы БД?

Цитатник веб-разработчиков.

**Максим Месилов:** Инфоблоки отлично подходят для прототипирования и макетирования функционала. На уровне своего приложения (модуля) делаете прослойку, которая отвечает за хранение данных и начинаете использовать инфоблоки.

*Если вы упрётесь в производительность или особенности работы ИБ, то просто смените самый нижний уровень. В моей практике такого пока не случалось.*

При создании собственных модулей у разработчиков часто возникает вопрос: При написании собственного модуля что целесообразнее использование инфоблоков или собственные таблицы? Ответ на этот вопрос зависит от решаемой задачи. Наличие в **Bitrix Framework** инфоблоков не означает обязательности их использования для реализации своих модулей.

**Инфоблоки** - это универсальность. По этой причине:

- Инфоблоки часто избыточны по своим возможностям;
- При использовании инфоблоков разработчик может работать с модулем как с обычным компонентом, не нужно дорабатывать АПИ (и описывать его).

**Собственные таблицы** - это прежде всего производительность. Используя свои таблицы разработчик:

- может сделать модуль куда более быстродействующим, чем если бы делал модуль со стандартным АПИ;
- разрабатывая собственное АПИ принимает на себя всю ответственность по безопасности своего модуля;
- должен подумать о том, что с его кодом будут работать другие разработчики: комментарии и документация.



## Структура полной сборки модуля

Полная сборка модуля предназначена для первоначальной установки модуля (когда этого модуля ещё нет у клиента или партнера).

Полная сборка должна содержать следующую структуру обязательных файлов модуля:

- /install/index.php - файл с описанием модуля, содержащий инсталлятор/деинсталлятор модуля.
- /install/version.php - файл с номером версии модуля. Версия не может быть равной нулю.
- /include.php – подключаемый файл (файл подключается при подключении модуля во время выполнения скриптов сайта), в нем должны находиться включения всех файлов с библиотеками функций и классов модуля.

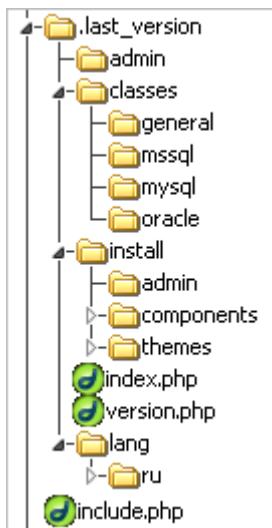
Все остальные файлы могут быть включены в модуль (могут отсутствовать), если это необходимо.

 **Примечание:** подробная информация о [структуре файлов](#) модуля.

Перед загрузкой модуля на сайт необходимо запаковать полную сборку модуля в архив. Для этого выполните следующее:

- Создайте папку .last\_version.
- Скопируйте в неё все файлы для полной сборки.
- Заархивируйте папку .last\_version в формат .zip или .tar.gz

В итоге должен получиться архив с именем .last\_version.zip (.last\_version.tar.gz). Для типичного модуля полная сборка может иметь следующую структуру каталогов и файлов:



При обнаружении и исправлении грубых ошибок партнерам следует обновить полную сборку модуля в системе обновлений. Причем это следует сделать в любом случае, вне зависимости от автоматической или не автоматической загрузки обновлений. Нужно всегда учитывать, что при любом типе установки обновления могут быть и не установлены.

Полная сборка не влияет на обновление модуля и вообще не используется для этого. Она нужна только для первичного скачивания и установки нового модуля. Если в силу каких-то причин необходимо запретить загрузку обновлений для каких-то пользователей (через год, например, или при новой версии), то достаточно отвязать клиента от модуля. Модуль у него останется, но обновляться он не сможет.

### Полезные методы

*WizardServices::SetFilePermission(\$path, \$permissions)*

по сигнатуре  
\$APPLICATION->SetFileAccessPermission(\$path, \$permissions), но с более правильной логикой добавления прав доступа (не затирает существующие права).

*WizardServices::AddMenuItem(\$menuFile, \$menuItem, \$siteID)*

добавление пункта меню

*WizardServices::IncludeServiceLang(\$relativePath, \$lang = false, \$bReturnArray = false)*

подключает произвольный языковой файл сервиса

*ImportIBlockFromXML(\$xmlFile, \$iblockXmIID, \$iblockType, \$siteID, \$permissions = Array())*

импорт инфоблока.

Если метод используется более чем в одном месте выносите его в класс **WizardServices**.

### Инсталлятор и деинсталлятор

Инсталлятор и деинсталлятор размещаются в файле /bitrix/modules/**ID** модуля/install/index.php. В нем должен быть описан класс, название которого совпадает с ID модуля. Например, так:

```
01  <?
02  Class mymodule extends CModule
03  {
04      var $MODULE_ID = "mymodule";
```



```
05      var $MODULE_NAME;
06
07      function DoInstall()
08      {
09          global $DB, $APPLICATION, $step;
10          $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/mymodule/install/step1.php");
11      }
12
13      function DoUninstall()
14      {
15          global $DB, $APPLICATION, $step;
16          $APPLICATION->IncludeAdminFile( GetMessage("FORM_INSTALL_TITLE"),
$_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/mymodule/install/unstep1.php");
17
18      }
19  }
20 ?>
```

Метод [DoInstall](#) будет вызываться при нажатии на кнопку **Установить** в списке модулей административной панели. Соответственно, [DoUninstall](#) – при нажатии на кнопку **Удалить**.

Также в папке `/install` находятся файлы `step1.php` и `unstep1.php`. Если нужен многошаговый установщик, то следует создать файлы `step1.php`, `step2.php` и т.д. Задача установщика – зарегистрировать модуль в системе. На самом деле, для этого достаточно вызвать всего лишь одну функцию:

```
1 RegisterModule("mymodule");
```

Однако скорее всего для модуля понадобятся свои таблицы в БД и файлы. В таком случае, следует создать их в шагах инсталлятора (`step1.php`, `step2.php` и т.д.). Аналогично, в деинсталляторе нужно сделать противоположные действия (удалить таблицы и файлы).

В качестве примера можно рассмотреть инсталляторы штатных модулей. Наиболее наглядный установщик у модуля **Веб-Формы**.

#### Примечание:

- Подчеркивание в ID модуля использовать нельзя. При добавлении модуля система проверяет допустимые символы в коде модуля. Можно использовать латинские буквы + цифры. Учитите, что если название

модуля начинается с цифр, то работать не будет (в PHP функция/класс не могут начинаться с цифр). Также не стоит использовать смешанный регистр в коде модуля.

- В **PARTNER\_URI** нельзя использовать языковое сообщение через `GetMessage`, только непосредственно строку.

### Сторонние библиотеки

Использование сторонних библиотек возможно если соблюдаются все лицензионные ограничения разработчиков библиотек. Если используется сторонняя библиотека, то необходимо проверить чтобы имена объектов/классов/функций не совпадали с системными.

Библиотеку не стоит размещать в папке `/bitrix/modules/`. Необходимые для работы модуля файлы лучше положить в ваш модуль, для удобства последующего обновления. Пользовательские же файлы с подобными файлами обычно кладутся в `/bitrix/php_interface/`.

### Пункт в меню админки

Если необходимо добавить в меню админки пункты вашего модуля, то необходимо использовать событие [OnBuildGlobalMenu](#).

Пример использования события для добавления собственного пункта в **Список пользователей**:

```
AddEventHandler("main", "OnBuildGlobalMenu", "MyOnBuildGlobalMenu");
function MyOnBuildGlobalMenu(&$aGlobalMenu, &$aModuleMenu)
{
    foreach($aModuleMenu as $k => $v)
    {
        if($v["parent_menu"] == "global_menu_settings" && $v["items_id"] == "menu_users")
        {
            $aModuleMenu[$k]["items"][] = Array(
                "text" => "Кастомный пункт пользователей",
                "url" => "user_custom.php?lang=".LANG,
                "title" => "Своя страница пользователей"
            );
        }
    }
}
```



## Задание демо-режима для модулей

Платный модуль может быть представлен в демо-режиме для изучения пользователем его возможностей перед приобретением.

В модуле могут быть заданы следующие ограничения:

- По новой технологии файлы **include.php** и **install/index.php** будут обфурцированы.
- В файл **include.php** будет добавлен код проверки триального режима и его срока.
- В файл **install/index.php** будет добавлен код, который будет устанавливать дату установки модуля для дальнейших проверок.

Кроме этого, для подключения модулей теперь можно использовать новую функцию **CModule::IncludeModuleEx(\$module\_name)**. Отличие ее от стандартной **CModule::IncludeModule** в том, что она в качестве результата может возвращать:

- **MODULE\_NOT\_FOUND** (0) - модуль не найден (например скопировали ваши компоненты из модуля, а модуль удалили);
- **MODULE\_INSTALLED** (1) - модуль установлен и подключен;
- **MODULE\_DEMO** (2) - модуль работает в демо-режиме (например можно вывести сообщение, что вы можете купить версию без ограничений);
- **MODULE\_DEMO\_EXPIRED** (3) - срок работы демо-режима модуля истек.

Если ваш модуль содержит только компоненты, то рекомендуется часть их функционала вынести в **include.php**, для того чтобы компоненты не работали без модуля.

## Обновления модулей

Обновления модуля ставятся друг за другом в строгом соответствии с версией. Каждое обновление содержит лишь изменение по сравнению с предыдущим. Настоятельно советуем при каждом выпуске обновления эмулировать самые различные ситуации для тестирования. Например, перед выпуском установить обновления и на чистые дистрибутивы разных версий, и на рабочие сайты.

Папка с обновлением должна содержать следующие файлы:

- **/install/version.php** - файл содержит номер версии обновления и дату его выпуска. Обязательный файл.
- **description.\*** - содержит описание обновления, где \* - идентификатор языка в системе. Например, описание обновления модуля на русском языке будет содержаться в файле **description.ru**, на английском - **description.en**. Необязательный файл.
- **updater.php** – файл запускается при установке обновления. С помощью этого файла выполняются действия для обновления на новую версию. Файл может

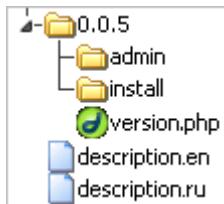
содержать произвольный PHP код, который выполняется в контексте сайта.  
Необязательный файл.

- **version\_control.txt** - служит для организации связи между версиями модулей. Файл содержит ссылки на версии модулей, от которых зависит данное обновление. Например, файл может содержать iblock,3.8.0. Это означает, что данное обновление будет установлено, если в системе установлен модуль Информационные блоки версии не ниже 3.8.0. Либо модуль Информационные блоки не установлен вообще. Необязательный файл.

Перед загрузкой модуля на сайт необходимо запаковать сборку обновления модуля в архив. Для этого выполните следующее:

- Создайте каталог с названием версии обновления. Например, 0.0.2, 0.0.5 и т.д.
- Скопируйте в созданную папку файлы и каталоги обновления модуля.
- Заархивируйте папку в формат .zip или .tar.gz.

В результате должно получиться, например, 0.0.2.zip, 0.0.5.zip. Например, папка с обновлением может иметь следующую структуру:



Все файлы модуля кроме **updater.php**, **description.ru** и прочих служебных копируются автоматически в папку вашего модуля при обновлении.

**⚠ Примечание:** Автоматически обновляется только ядро. Все остальные файлы (в том числе и компоненты модуля) - только по явному указанию. Если возникла необходимость скопировать файлы в обновлении самостоятельно, то нужно использовать:

```
$updater->CopyFiles("install/classes",
"modules/quintura.search/classes");
```

В этом случае файлы из папки *install/classes*, находящиеся в папке обновления, скопируются в папку *bitrix/modules/quintura.search/classes*.

Если при выпуске последующих обновлений возникла потребность установить зависимость его от новых модулей, то необходимо помнить, что обновление с зависимостью не будет требовать установки указанных модулей. В этом случае возможны два варианта:

1. Обновление всё равно устанавливать, проверять присутствие нужного модуля уже в функционале модуля.



- Добавить в код обновления проверку на нужный модуль, и при его отсутствии выводить ошибку пользователю. Обновление не будет установлено, если присвоить переменной **\$errorMessage** строку сообщения.

### Размещение модуля в партнерской системе обновлений

Чтобы выполнить установку модуля через партнерскую систему обновлений необходимо:

- партнеру предварительно подать запрос на включение функционала для вашей партнерской карточки. После одобрения в карточке партнера укажите **Код партнера**, который будет использоваться как код для собственных модулей, и **Лицензионный ключ**, который будет использоваться для тестирования альфа-версий обновлений, доступных только вам по этому ключу.

Прочие возможности	
Код партнера (используется как код для собственных модулей):	<input type="text" value="anniacompany"/>
Лицензионный ключ (может использоваться Вами для тестирования обновления, должен быть активным, в исходных кодах и привязан к вашей карточке партнера):	<input type="text"/>

- партнеру собрать и загрузить на сервер компании «1С-Битрикс» дистрибутив вашего модуля и обновления.
- клиенту установить модуль (либо его обновление).

### **Загрузка модуля**

Для загрузки модуля выполните следующее:

- Перейдите на страницу [Ваши решения](#).
- На странице перейдите по ссылке **Добавить решение**, и в открывшейся форме



## Создание решения

[Вернуться в список решений](#)

### Решение

*Код	anniacompany.
Активность	<input checked="" type="checkbox"/>
Бесплатное решение	<input type="checkbox"/> Да <input type="checkbox"/> Установите эту галочку, если хотите поставлять ваше решение клиенту в триальном режиме на <input type="text" value="30"/> дней Цена: <input type="text"/> руб. <input checked="" type="checkbox"/> Разрешить партнерские скидки
Изображение	<input type="text"/> Обзор... <input type="checkbox"/> Удалить
Кодировка	<input type="text" value="windows-1251"/>
*Продукты, с которыми работает решение	Управление сайтом Корпоративный портал Управление сайтом ASP.NET
*Решение включает в себя	Компоненты Мастера создания Мастера создания (установка) Модуль Переводы
*Категория	Типовой сайт (ru) eCommerce (ru) Сервисы (ru) Мультимедиа (ru)

- заполните следующие поля:

- Код - в поле обязательно указывается полный код партнёрского модуля в формате **код\_партнера.код\_модуля**. Часть **код\_партнера** постоянна для партнёра (задается в карточке партнёра). Часть **код\_модуля** вводится партнёром при добавлении нового модуля. Эти коды должны быть алфавитно-цифровыми с первым алфавитным символом, и код некоторым образом должен соответствовать сути модуля. Например, для модуля можно задать код **mусар**. Тогда полный код будет **alexey.mусар**. Пример класса модуля, в котором этот код модуля используется, содержится в приложении.
- Активность - при отмеченной опции модуль будет отображаться в списке модулей.
- Бесплатное решение - при отмеченной опции модуль будет бесплатным, то есть доступным всем. При снятом флаге появятся дополнительные поля в которых надо отобразить:
  - Наличие триального периода;



- Срок действия триального периода;
- Цену модуля.

**⚠ Примечание:** Если необходимо сменить текущую цену, то поменяв значение цены вы фактически отправляете уведомление в Партнерский отдел "1С-Битрикс", который активирует новую цену.

- Разрешить партнерские скидки. При установленной опции другим партнерам будет разрешено покупать ваш модуль со скидками соответствующими статусу партнера.
- **Доступен в каталоге для покупателей** - при отмеченной опции модуль будет добавлен в каталоге Marketplace на сайте компании «1С-Битрикс».
- **Изображение** - с помощью кнопки **Выберите файл** укажите путь к изображению, которое будет отображаться в списке модулей. (В разных браузерах выглядит по разному.)
- **Кодировка** - указывается кодировка файлов модуля.
- **Примечание:** система обновлений автоматически переводит языковые файлы из Win-1251 в UTF-8, если у клиента выбрана кодировка UTF-8. Если разница в версиях только в кодировке языковых файлов, то размещать надо только Win-1251 версию. Если разница в коде, то рекомендуется вынести эту разницу в языковые файлы.
- **Продукты, с которыми работает модуль (обязательный)** – в списке выберите продукт для которого предназначен модуль.
- **Модуль включает в себя** – в списке обязательно указываются сущности, которые имеются в модуле: Компоненты, Мастер создания, Модуль, Переводы или Шаблоны сайта.
- **Категория** - указывается категория, к которой относится модуль. Если вам не хватает категорий, то вы можете написать в техподдержку компании «1С-Битрикс» с просьбой добавить необходимую категорию.
- **Архив с полной сборкой модуля** – указывается путь к архиву с полной сборкой модуля (файл .last\_version.zip или .last\_version.tar.gz).
- **Описание модуля на Русском языке** – в данной секции задаются значения следующих полей:
  - **Название** – название модуля;
  - **Описание модуля** – описание модуля;
  - **Описание установки модуля** – описание установки модуля;
  - **Описание техподдержки и контактных данных** – контактные данные и техподдержка модуля;
  - **Ссылка на демо-версию** – ссылка на сайт с демо-версией модуля;
  - **Скриншоты** – можно загрузить скриншоты модуля. (5 скриншотов)
- Сохраните внесенные изменения.

В результате модуль будет добавлен в список ваших персональных модулей:



## Загрузка обновлений

Для загрузки обновлений выполните следующее:

- Перейдите на страницу Ваши решения
- Перейдите по ссылке **Обновления** в колонке действий. Откроется страница Обновления модуля:

**Обновления модуля "Моя машина"**

[Вернуться в список модулей](#) | [Загрузить новое обновление](#)

Версия	Тип	Действие
Нет еще ни одного обновления модуля		

- Чтобы загрузить обновление воспользуйтесь ссылкой **Загрузить новое обновление**. На странице **Загрузка обновления для решения «название\_решения»** с помощью кнопки **Выберите файл** укажите путь к архиву с обновлением:

**Загрузка обновления для модуля "Моя машина"**

[Вернуться в обновления модуля](#)

Архив с обновлением (в формате .zip или .tar.gz)  
Имя архива должно соответствовать номеру обновления, например 8.0.1.zip

Выберите файл 0.0.2.zip

**Загрузить**

- Нажмите кнопку **Загрузить**.
- После загрузки архива на странице Обновления модуля «название\_модуля» в поле **Тип** укажите тип обновления:



## Обновления модуля "Моя машина"

[Вернуться в список модулей](#) | [Загрузить новое обновление](#)

Версия	Тип	Действие
0.0.5	<input type="radio"/> Альфа <input checked="" type="radio"/> Бета <input type="radio"/> Стабильное	<a href="#">Изменить</a> <a href="#">Удалить</a>
0.0.2	<input checked="" type="radio"/> Альфа <input type="radio"/> Бета <input type="radio"/> Стабильное	<a href="#">Изменить</a> <a href="#">Удалить</a>

[Сохранить](#)

- **Альфа** – альфа-версия обновления, доступная для загрузки только копии продукта с ключом, который указан в карточке партнера, предназначена для тестирования обновления;
- **Бета** - бета-версия обновления, которая может быть установлена, если включена соответствующая опция в настройках Главного модуля;
- **Стабильное** - окончательная стабильная версия обновления:

- Нажмите кнопку **Сохранить**.

### Расшифровка ошибок

При загрузке модуля (или обновления) в Marketplace выдается сообщение: Неверное содержимое архива с обновлением. Возможные причины:

- В файле `install/version.php` не задана версия модуля;
- В файле `install/index.php` не указан `$MODULE_ID`;
- В файле `install/index.php` не указан `PARTNER_NAME` (Название партнера, разработчика модуля);
- В файле `install/index.php` не указан `PARTNER_URI` (Адрес партнера, разработчика модуля);
- В файле `install/index.php` не верно указано имя класса.

### Работа с клиентами модуля

Marketplace предоставляет возможность автоматической работы с клиентами модуля. На данный момент добавлять клиентов модуля можно только для платных модулей.

#### Скрипт для автоматической работы с клиентами модуля

[Скачать](#)

### Входные параметры

**partner\_id** ID партнера на нашем сайте (обязательный)

**module\_id** код модуля (обязательный)

**key** хэш ключа (передается вам клиентом) (обязательный)

**name** название клиента

**email** email клиента

**site\_url** адрес сайта клиента

**contact\_person** контактное лицо

**phone** телефон клиента

**comments** произвольные комментарии (метод оплаты, описание клиента и т.п.)

**action** действие, принимает значение add|delete|update, если не задано, считаем что это добавление нового клиента

подпись запроса, формируется следующим образом:

**hash**

```
$md5
md5($partner_id."|".$module_id."|".$key."|".$action."|".$salt
);
```

**\$salt** "Пароль для подписи данных", задается в карточке партнера

**is\_utf** если значение "Y", то все входные параметры будут перекодированы из UTF-8 в кодировку нашего сайта (windows-1251)

### Результат работы

Скрипт возвращает два типа ответа:

**OK**

*сообщение об успешно совершенном действие*

**ERROR**

текст ошибки

### Примеры использования

```
//Добавим нового клиента
http://partners.1c-
bitrix.ru/add_client.php?partner_id=46844&module_id=alexey.mycar&key=f6f90c0f4a0b6f70ef
1ed1a4beaefefe&action=add&name=Anton%20Ezhkov&email=anton@bitrix.ru&site_url=www.
1c-bitrix.ru&contact_person=Anton%20Ezhkov&phone=123123123&comments=Оплатил
безналом&is_utf=Y&hash=34a36dfee17bc66bd87f036c8af07223
```

```
//Изменим информацию о клиенте
http://partners.1c-
bitrix.ru/add_client.php?partner_id=46844&module_id=alexey.mycar&key=f6f90c0f4a0b6f70ef
1ed1a4beaefefe&action=update&name=Антон
Ежков&email=anton@bitrix.ru&site_url=www.1c-bitrix.ru&contact_person=Антон
Ежков&phone=123123123&comments=Оплатил безналом, помогал в
техподдержке&is_utf=Y&hash=2422d9cf365017a452842f7047ba0662
```

```
//Удалим ключ из клиентов модуля
http://partners.1c-
bitrix.ru/add_client.php?partner_id=46844&module_id=alexey.mycar&key=f6f90c0f4a0b6f70ef
1ed1a4beaefefe&action=delete&hash=2422d9cf365017a452842f7047ba0662
```

### Решения типовых сайтов

Кроме модулей в Marketplace возможна реализация решений типовых сайтов партнеров.

### **Примерный порядок создания мастера установки решения**

Чтобы сделать мастер:

- Создайте папку мастера, и его базовые файлы с папками. Они почти одинаковые для каждого мастера, можно взять готовые и внести в них.
- Скопируйте файлы публички в папку /public, шаблоны в папку /templates. Основная логика установки должна быть описана в файле services/.services.php. Массив **\$arServices** содержит описание последовательности запуска файлов из соответствующих подкаталогов services.

На сами файлы никаких ограничений нет, Вы можете, например, создать файл **events.php** для установки почтовых событий и написать там вызов API для их создания в случае, если они уже не были созданы.

В качестве примера можно рассмотреть уже готовые мастера установки решений, имеющиеся в дистрибутивах продуктов на **Bitrix Framework**. Например: Мастер создания интернет-магазина в "1С-Битрикс: Управление сайтом", расположенный в папке: /bitrix/wizards/bitrix/store.

### Особенности, которые необходимо учитывать

В публичной части инсталлятора файл **index.php** должен именоваться как **\_index.php** - иначе будет ошибка при инсталляции. Такого именования достаточно, что бы мастер автоматом в конце установки заменил текущий **index.php** на ваш.

Все файлы с кодировкозависимыми символами должны располагаться внутри папки **~/lang/[ru|en]/** и должны быть созданы в языковых сообщениях, когда это требуется. Система обновлений для партнерских модулей перекодирует все файлы, путь к которым содержит **/lang/**. Например, для файла **./portal/site/services/.services.php** путь должен быть таким: **./portal/lang/[ru|en]/site/services/.services.php**. Для установщика сервиса (подключается автоматически) **./portal/site/services/папка\_сервиса/установщик.php** путь такой: **./portal/site/services/папка\_сервиса/lang/[ru|en]/установщик.php**.

### Переустановка сайта

При создании мастера важно также учитывать момент "переустановки сайта", когда пользователь будет запускать мастер повторно. Нужно разложить все сущности на те, которые следует переустановить (шаблоны, например) и те, которые следует сохранить (включаемые области, инфоблоки и т.д.)

В установочных скриптах необходимо проверять на существование создаваемую сущность. Если сущность уже установлена – пропускать установку. В установочных файлах можно делать return. Пример:

```
<?
if (!CModule::IncludeModule("iblock"))
return;
$typeResult = CIBlockType::GetByID("news");
if ($typeResult->Fetch()) //Установлен тип Новости? Пропускаем установку
return;
//Создание типа инфоблока
?>
```

### Инфоблоки, каталоги, блоги и прочее



Чтобы прописать ID инфоблоков в файлы публичной части, надо запускать после копирования файлов публички и установки инфоблока метод [CWizardUtil::ReplaceMacros](#), где первый параметр - это путь к файлу, а второй - массив подстановок, например, чтобы заменить #NEWS\_IBLOCK\_ID# на число, нужно сделать так:

```
CWizardUtil::ReplaceMacros(WIZARD_SITE_PATH."/news/index.php",
array("NEWS_IBLOCK_ID" => $iblockID));
```

Пример вывода настроек инфоблока:

```
$iblockID = 2;
$result = CUserOptions::GetOption("form", "form_element_". $iblockID); var_export($result);
echo
";
CModule::IncludeModule("iblock");
$arFields = CIBlock::getarraybyid($iblockID);
foreach ($arFields["FIELDS"] as $id => $arField) unset($arFields["FIELDS"][$id]["NAME"]);
var_export($arFields["FIELDS"]);
```

## Многосайтовость

При создании решения, если он должен быть установлен в директорию первого сайта (многосайтовость на одном домене), то ссылки должны учитывать директорию (поле SITE\_DIR в настройках сайта).

## Включаемые области

Файлы с включаемыми областями, которые подключаются в шаблоне и хранятся в папке с шаблоном сайта (/bitrix/templates/имя\_шаблона/includes/), при установке сайта мастером, не перекодируются в UTF-8. Правильно будет разместить файлы с шаблонами включаемых областей в папке /includes/ в корне.

**⚠ Примечание:** Включаемые области в мастере могут отсутствовать вообще, а значения по умолчанию для них хранятся в LANG файле, соответственно, перекодируются и заносятся в форму настроек решения (или берутся из областей, если решение уже было установлено раньше и файлы областей существуют в папке целевого сайта). При установке файлы включаемых областей создаются мастером на основе данных из формы настроек решения.

## Другие моменты

- Чтобы мастер установки решения запускался сразу из Marketplace, после загрузки модуля, необходимо чтобы он лежал по адресу: /bitrix/modules/модуль/install/wizards/партнер/мастер/.
- Во всех исполняемых файлах, не находящихся в публичной части, первая строка должна выглядеть так:



```
if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
```

- В установочных файлах можно пользоваться глобальными объектами \$DB, \$DBType, \$APPLICATION, \$USER;
- Пользователь, запускающий мастер, будет авторизован как администратор (ID=1).

#### Некоторые константы

<b>WIZARD_SITE_ID</b>	идентификатор сайта
<b>WIZARD_TEMPLATE_ID</b>	идентификатор выбранного шаблона
<b>WIZARD_TEMPLATE_RELATIVE_PATH</b>	путь к выбранному шаблону в мастере относительно корня сайта
<b>WIZARD_TEMPLATE_ABSOLUTE_PATH</b>	полный путь к выбранному шаблону в мастере
<b>WIZARD_THEME_ID</b>	идентификатор выбранной темы
<b>WIZARD_THEME_RELATIVE_PATH</b>	путь к выбранной цветовой схеме в мастере относительно корня сайта
<b>WIZARD_THEME_ABSOLUTE_PATH</b>	полный путь к выбранной цветовой схеме в мастере
<b>WIZARD_ABSOLUTE_PATH</b>	полный путь к мастеру
<b>WIZARD_RELATIVE_PATH</b>	путь к мастеру относительно корня сайта
<b>WIZARD_SERVICE_ABSOLUTE_PATH</b>	полный путь к директории, где находится установщик сервиса
<b>WIZARD_SERVICE_RELATIVE_PATH</b>	путь к директории установщика сервиса относительно корня сайта
<b>WIZARD_SITE_PATH</b>	абсолютный путь к корню сайта
<b>WIZARD_REINSTALL_DATA</b>	путь к мастеру удаления демо-данных



## Установка модуля

### Бесплатный модуль

Для его установки необходимо выполнить следующее:

- В административной части сайта перейти на страницу **Сторонние обновления** ([Настройки > Сторонние обновления](#)).
- На закладке **Добавить модуль** в строке поиска указать код модуля (в формате **код\_партнера.код\_модуля**) и нажать кнопку **Найти модуль**.
- После того как модуль будет найден необходимо нажать кнопку **Загрузить**.

### Платный модуль

Для его установки необходимо выполнить следующее:

- Партнеру (владельцу модуля) на закладке **Служебное** страницы **Карточка партнера** необходимо указать **Код партнера**, который будет использоваться как код для собственных модулей, и **Лицензионный ключ**, который будет использоваться для тестирования альфа-версий обновлений, доступных только вам по этому ключу:

Код партнера (будет использоваться как код для собственных модулей):	<input type="text" value="alexey"/>
Лицензионный ключ (Будет использоваться Вами для тестирования обновления, должен быть активным, в исходных кодах и привязан к вашей карточке партнера):	<input type="text" value="GT6-UY-IUYTYUIJMIOQ"/>

- Клиенту скопировать **Код лицензионного ключа** на странице **Сторонние модули** и передать данный код партнеру после продажи лицензии на использование модуля.



Установка обновлений Список обновлений Активация купона

**Установите обновления на ваш сайт**

Система обновлена

**Рекомендуемые обновления:** нет

Установить рекомендуемые обновления

[Посмотреть список обновлений](#)

На вашем сайте включена установка бета-версий обновлений продукта. Если вы хотите установить только стабильные обновления, отключите, пожалуйста, установку бета-версий в настройках системы обновлений.

Для того чтобы получить новый функционал, ускорить работу сайта и усилить безопасность, всегда устанавливайте обновления модулей.

Ответ сервера обновлений

Зарегистрировано на имя: Татьяна Старкова  
**Код лицензионного ключа:** a48a2fdfe5a2f076fec80c8b1809fa5  
Обновления доступны: с 01.04.2010 по 31.12.2012  
Сервер обновлений: www.bitrixsoft.com

- Партнеру перейти по ссылке **Клиенты** в колонке **Действия** и в поле указать **Код лицензионного ключа** клиента, нажать кнопку **Добавить** и заполнить поля формы:

Клиент

Информация о клиенте

Код ключа: f383f2e65d6e0c722bab31bb0f4f9b8e

Название: ООО "ГРАТЭКС"

Email:

Адрес сайта: gratwest.ru; trans-roller.ru

Контактное лицо:

Телефон:

Комментарий: Добавлен через систему обновлений

**Сохранить** **Применить** **Отменить**



В результате клиенты смогут установить и получить обновления партнерского модуля.

## Установка решения и обновлений

Установка стороннего решения и получение обновлений по нему описана в курсе [Администратор. Базовый](#)

### Частые ошибки

**Цитатник веб-разработчиков.**

**Максим Месилов:** Всегда проверяйте свои разработки на последних стабильных [версиях PHP](#).

### Ошибки при создании модулей

**Проблема:** После установки продукта не происходит запуск мастера.

**Решение:** Проверить наличие строк в файле **index.php**:

```
10 "START_TYPE" => "WINDOW",
11 "WIZARD_TYPE" => "INSTALL",
```

**Проблема:** В состав модуля не добавляется файл с указанием версии

Примерное содержание файла /install/version.php:

```
<?
$arModuleVersion = array(
    "VERSION" => "11.0.4",
    "VERSION_DATE" => "2011-11-17 14:00:00"
);
?>
```

### Ошибки при создании решений

**Проблема:** Не импортируется часть свойств инфоблоков мастером установки сайта, при установке UTF-8 версии, при установке в Windows-1251 проблем нет.

**Решение:** Если файл располагается в директории ~/ru/ то система обновлений его автоматически перекодирует. Т.е. он должен быть загружен в windows-1251, в UTF-8 он при необходимости сам перекодируется. Однако при таком подходе необходимо убирать из заголовка файла указание кодировки, чтобы не было двойной\некорректной перекодировки файла. В противном случае его необходимо выносить из ~/ru/ и производить ручную перекодировку с помощью \$APPLICATION->ConvertCharset().



## Пример класса для модуля

Пример класса для модуля alexey.mycar

```
Class alexey_mycar extends CModule
{
    var $MODULE_ID = "alexey.mycar";
    var $MODULE_VERSION;
    var $MODULE_VERSION_DATE;
    var $MODULE_NAME;
    var $MODULE_DESCRIPTION;
    var $MODULE_CSS;
    var $MODULE_GROUP_RIGHTS = "Y";

    function alexey_mycar()
    {
        $arModuleVersion = array();

        $path = str_replace("\\", "/", __FILE__);
        $path = substr($path, 0, strlen($path) - strlen("/index.php"));
        include($path."/version.php");

        $this->MODULE_VERSION = $arModuleVersion["VERSION"];
        $this->MODULE_VERSION_DATE = $arModuleVersion["VERSION_DATE"];
        $this->PARTNER_NAME = "Partner name";
        $this->PARTNER_URI = "http://www.1c-bitrix.ru/";

        $this->MODULE_NAME = GetMessage("MYCAR_MODULE_NAME");
        $this->MODULE_DESCRIPTION
GetMessage("MYCAR_MODULE_DESCRIPTION");
    }

    function InstallDB()
    {
        RegisterModule("alexey.mycar");
        //RegisterModuleDependences("search",      "OnReindex",      "alexey.mycar",
"CMycarSearch", "OnSearchReindex");
        return true;
    }
}
```

```
}

function UnInstallDB()
{
    //UnRegisterModuleDependences("search",      "OnReindex",      "alexey.mycar",
"CMyCarSearch", "OnSearchReindex");
    UnRegisterModule("alexey.mycar");
    return true;
}

function InstallEvents()
{
    return true;
}

function UnInstallEvents()
{
    return true;
}

function InstallFiles()
{
    CopyDirFiles(
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/alexey.mycar/install/components/",
        $_SERVER["DOCUMENT_ROOT"]."/alexey/components",
        true, true
    );
    CopyDirFiles(
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/alexey.mycar/install/admin/",
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/admin",
        true, true
    );
    CopyDirFiles(
        $_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/alexey.mycar/install/themes",
```



```
$_SERVER["DOCUMENT_ROOT"]."/bitrix/themes", true, true
};

return true;
}

function UnInstallFiles()
{

DeleteDirFiles($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/alexey.mycar/install/admin",
$_SERVER["DOCUMENT_ROOT"]."/bitrix/admin");

DeleteDirFiles($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/alexey.mycar/install/themes/.default/",
$_SERVER["DOCUMENT_ROOT"]."/bitrix/themes/.default");//css
DeleteDirFilesEx("/bitrix/themes/.default/icons/alexey.mycar/");//icons

return true;
}

function DoInstall()
{
global $APPLICATION;

if (!IsModuleInstalled("alexey.mycar"))
{
$this->InstallDB();
$this->InstallEvents();
$this->InstallFiles();
}
}

function DoUninstall()
{
$this->UnInstallDB();
$this->UnInstallEvents();
$this->UnInstallFiles();
}
}
```



## Пример создания модуля

В реальности разработчики сторонних модулей используют чаще всего модуль как инсталлятор для своих, либо кастомизированных компонентов.

Визуально модули в **Bitrix Framework** для пользователя представляют собой лишь список на странице, где можно инсталлировать или деинсталлировать модуль в системе.

Поставим себе задачу сделать модуль, который бы просто при инсталляции ставил компонент, который в качестве [примера создания компонента](#): компонент, выводящий текущую дату и время. Имя компонента было **dv:date.current**.

Начнем с определения тех файлов и папок, которые нам нужно создать. Очевидно, что нам не нужно инсталлировать картинки, шаблоны, для решения нашей задачи не нужны административные скрипты и скрипты, которые вызывают административные, также не нужны.

Таким образом, структура модуля будет такой:

- /install/
  - /components/
    - /dv/
      - /date.current/
        - /templates/
          - /.default/
          - **template.php**
      - component.php
      - .description.php
      - .parameters.php
    - index.php
    - step.php
    - unstep.php
    - version.php

Вся эта структура должна располагаться в папке `/bitrix/modules/dv_module`, таким образом **dv\_module** будет являться папкой создаваемого модуля.

Файлы в папах `/install/components/` и `/install/components/` - это компонент, который мы хотим установить и поэтому писать эти файлы уже не надо. Основной файл, код которого отвечает собственно за инсталляцию/деинсталляцию модуля — это `/install/index.php`. Код его приведен ниже:

```
<?
```



```
Class dv_module extends CModule
{
var $MODULE_ID = "dv_module";
var $MODULE_VERSION;
var $MODULE_VERSION_DATE;
var $MODULE_NAME;
var $MODULE_DESCRIPTION;
var $MODULE_CSS;

function dv_module()
{
$arModuleVersion = array();

$path = str_replace("\\", "/", __FILE__);
$path = substr($path, 0, strlen($path) - strlen("/index.php"));
include($path."/version.php");

if (is_array($arModuleVersion) && array_key_exists("VERSION", $arModuleVersion))
{
$this->MODULE_VERSION = $arModuleVersion["VERSION"];
$this->MODULE_VERSION_DATE = $arModuleVersion["VERSION_DATE"];
}

$this->MODULE_NAME = "dv_module – модуль с компонентом";
$this->MODULE_DESCRIPTION = "После установки вы сможете пользоваться
компонентом dv:date.current";
}

function InstallFiles($arParams = array())
{
CopyDirFiles($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/dv_module/install/components"
, $_SERVER["DOCUMENT_ROOT"]."/bitrix/components", true, true);
return true;
}

function UnInstallFiles()
```

```

{
DeleteDirFilesEx("/bitrix/components/dv");
return true;
}

function DoInstall()
{
global $DOCUMENT_ROOT, $APPLICATION;
$this->InstallFiles();
RegisterModule("dv_module");
$APPLICATION->IncludeAdminFile("Установка модуля dv_module",
$DOCUMENT_ROOT."/bitrix/modules/dv_module/install/step.php");
}

function DoUninstall()
{
global $DOCUMENT_ROOT, $APPLICATION;
$this->UnInstallFiles();
UnRegisterModule("dv_module");
$APPLICATION->IncludeAdminFile("Деинсталляция модуля dv_module",
$DOCUMENT_ROOT."/bitrix/modules/dv_module/install/unstep.php");
}
}

?>
```

В этом файле декларируется новый класс — класс нашего модуля **dv\_module** как потомок **CModule**. Далее идет определение конструктора **dv\_module()**, в которой происходит определение переменных для вывода информации о модуле в списке модулей **Bitrix Framework**.

Метод **DoInstall()** вызывается при установке модуля из Панели управления, метод **DoUninstall()**, соответственно, при деинсталляции модуля. В методах этого класса вызываются так или иначе файлы `/install/step.php`, `/install/unstep.php` и `/install/version.php`. Первые два файла — это файлы, которые показываются при установке и деинсталляции модуля, соответственно. Вспомогательный файл **version.php** содержит информацию о версии модуля. Коды файлов:

- `/install/step.php`

```

<if(!check_bitrix_sessid()) return;?>
< ?
```



```
echo CAdminMessage::ShowNote("Модуль dv_module установлен");
?>
```

- /install/unstep.php

```
<?if(!check_bitrix_sessid()) return;?>
< ?
echo CAdminMessage::ShowNote("Модуль успешно удален из системы");
?>
```

- /install/version.php

```
<?
$arModuleVersion = array(
"VERSION" => "1.0.0",
"VERSION_DATE" => "2010-06-03 17:00:00";
?>
```

Из кода файла /install/index.php видно, что последних трех файлов может и не быть. Однако, в таком случае не будет четкого разделения логики и представления.

**Совет от Коваленко Алексея:** Если вы осуществляете в модулях какие то операции с обработкой данных или действия, на которые готовы доверить "допуск" к корректировке данных, заложите свои события в системе. И те специалисты, которые будут внедрять ваши решения будут вам за это благодарны.

### Сборка обновлений собственного модуля

Предлагаемый вашему вниманию скрипт является больше примером для оптимизации разработки и экономии времени, чем готовым решением. Скрипт разработан сотрудниками ТП "1С-Битрикс" для облегчения рутинных операций по сборке обновлений модулей.

**⚠ Примечание:** Предлагаемый вариант не единственный, но он обладает большими возможностями в плане "тонкой" настройки сборки обновлений. Если не нужна избыточная гибкость в настройках, то лучше воспользоваться более удобным [методом сборки](#) обновлений.

[Скачать скрипт.](#)

Предположим, что вы уже создали свой модуль и выложили в [Marketplace](#). Теперь у вас стоит задача совершенствовать его и выкладывать обновления.

Возможно для кого-то это покажется простой и интересной задачей, но для других она кажется рутинной и постоянно бегать по директориям в поисках нужного файла и так пофайлово собирать обновления довольно утомительно.

## Настройки

Код модуля:

```
$arSettings["MODULE_ID"] = 'mail';
```

Код модуля директория относительно корня сайта куда будут складываться обновления:

```
$arSettings["U_PATH"] = '/iupdate/';
```

Массив директорий содержимое которой будет прочитано, но глубже чтение не будет проводиться:

```
$arSettings["DIR_READ_NOFOLLOW"] = array(  
    $arSettings["MODULE_PATH"].'lang/',  
);
```

Массив директорий, имя которой будет добавлено но не будет чтение ее содержимого:

```
$arSettings["DIR_NOFOLLOW"] = array(  
    $arSettings["MODULE_PATH"].'install/db/',  
    $arSettings["MODULE_PATH"].'install/images/',  
    $arSettings["MODULE_PATH"].'install/themes/',  
);
```

Массив директорий, которые будут пропущены при чтении (построении списка):

```
$arSettings["DIR_SKIP"] = array();
```

Массив игнорируемых имен:

```
$arSettings["FILE_NAME_SKIP"] = array(  
    '.',  
    '..',  
    '.hg',  
    '.hgignore',  
    '.svn',  
    '.csv',  
);
```



Массив игнорируемых файлов:

```
$arSettings["FILE_SKIP"] = array  
    $arSettings["MODULE_PATH"].'install/version.php',  
    $arSettings["MODULE_PATH"].'version_control.php',  
);
```

Массив для **updater.php** при наличии необходимых директории на основании этих данных будет построен файл обновления:

```
$arSettings["UPDATER_COPY"] = array  
    "install/js" => "js/main/core",  
    "install/components" => "components",  
);
```

### Создание Updater'ов для модуля mail

Версия: 10.0.0

Описание:

- Миграция на версию 10.0.0
- Незначительные изменения

**Собрать**

- Очистить директорию сборки  
 Создать архив

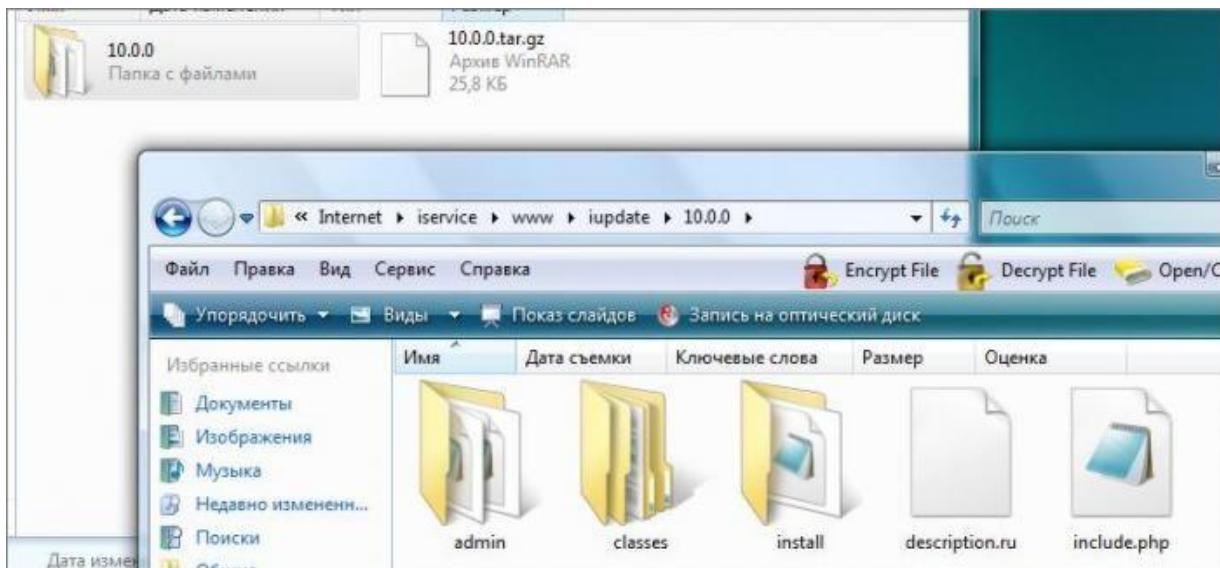
/  
 constants.php  
 include.php  
 options.php  
 prolog.php  
 smtpd.php  
 /admin/  
 mail\_attachment\_view.php  
 mail\_check\_mailbox.php  
 mail\_check\_new\_messages.php  
 mail\_filter\_admin.php  
 mail\_filter\_edit.php  
 mail\_index.php  
 mail\_log.php  
 mail\_mailbox\_admin.php  
 mail\_mailbox\_edit.php  
 mail\_message\_admin.php  
 mail\_message\_view.php  
 mail\_smtpd\_manager.php  
 menu.php  
 /classes/  
 /classes/general/  
 mail.php

Что же скрипт умеет?

- Собирать обновления с необходимым набором файлов;
- Создавать файл описания на основании введенного текста;

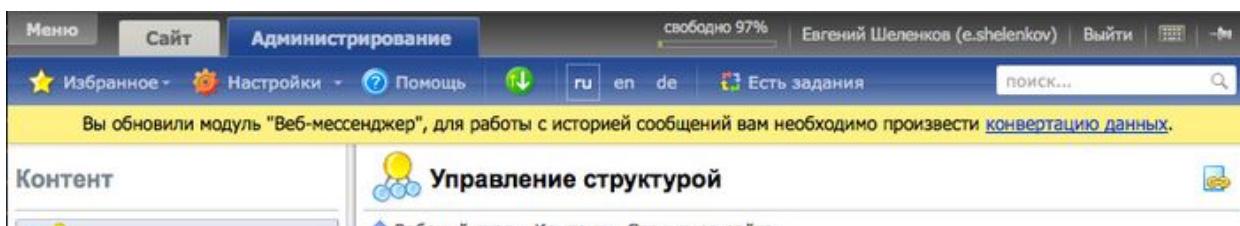


- Создавать файл **version.php** с номером версии указанной в поле ввода и время создания обновления;
- Создание **updater.php** на основании файлов обновления и настроек
- Создание архива собранного обновления



## Дополнительно

Начиная с версии ядра 11.5.6 в **Bitrix Framework** появляется возможность уведомлять администраторов о важных действиях, например что необходимо произвести конвертацию данных. Администратор увидит стикер под панелью инструментов, как в административном интерфейсе, так и в публичной части. Если одновременно должно вывестись несколько уведомлений, то они выведутся одно над другим.



Уведомления есть двух типов: первые администратор может закрыть, вторые уйдут только после выполнения действия (например полной конвертации).

Для использования этой возможности в собственных модулях используйте методы класса [CAdminNotify](#).



## Конструктор модулей для Marketplace

**Mpbuilder** - модуль для создания простого модуля из любого скрипта или компонентов с целью дальнейшей публикации в каталоге решений Marketplace. Автоматически конвертирует файлы в нужную кодировку, извлекает языковые фразы и собирает обновления.

Скачать и установить Конструктор модулей для Marketplace можно с сайта или в Административной части сайта ([Настройки > Marketplace > Каталог решений](#)) в разделе Служебные:

The screenshot shows the 'Catalog Marketplace' section of the 1C-Bitrix administration interface. The 'Mpbuilder' module is listed as a free module (Бесплатный) with a rating of 5 stars and 8 votes. It includes a large red icon with a white 'b' logo, a 'Install' button, and details about its purpose: simplifying module creation, extracting language phrases, and creating an archive for publication in the Marketplace. Below this, there are tabs for Screenshots, Updates, Technical Support, and Installation, and a detailed configuration window for creating a module structure.



Доступные решения

Рабочий стол > Настройки > Marketplace > Установленные решения

Решение "Конструктор модуля" успешно установлено.

### Доступные решения

Решения, доступные в системе.  
Вы можете устанавливать, удалять и стирать решения. Для того чтобы стереть решение из системы его необходимо сначала удалить.

Название	Разработчик	Версия	Дата обновления	Статус
<b>Конструктор модуля (bitrix.mpbuilder)</b> Модуль для создания простого модуля из любого скрипта или компонентов с целью дальнейшей публикации в каталоге решений: <a href="http://marketplace.1c-bitrix.ru">http://marketplace.1c-bitrix.ru</a> . Автоматически конвертирует файлы в нужную кодировку, извлекает языковые фразы и собирает обновления.	<a href="#">1С-Битрикс</a>	1.0.3	16.02.2012	Установлен

Выбрано: 1

## Работа с конструктором

Модуль **Mpbuilder** позволяет начинать разработку модуля как «с нуля», так и выпускать обновление уже готового модуля для **Marketplace**.

Итак, приступим.

### Шаг 1. Создание модуля из скрипта.

На первом шаге заполняются необходимые информации:



**Шаг первый: Создание структуры модуля**

Рабочий стол > Настройки > bitrix.mpbuilder > step1.php

Данная форма поможет создать модуль из служебных скриптов с целью дальнейшей публикации в каталоге решений [marketplace.1c-bitrix.ru](#). Подробную документацию по созданию своего модуля можно найти на сайте для разработчиков [dev.1c-bitrix.ru](#).

**Шаг 1**

**Создание модуля из скрипта**

**Данные разработчика**

Название вашей компании: Битрикс

Адрес вашего сайта: <http://www.1c-bitrix.ru>

Код партнера: bitrixtest

Код партнера постоянен для партнера, задается в карточке партнера (например: *mycompany*).  
Код модуля вводится при добавлении модуля в marketplace (например: *testmodule*). Папка модуля состоит из кода партнера и кода модуля, например: *mycompany/testmodule*.

**Данные нового модуля**

Переписать существующие файлы:

Код модуля: bitrixtest

Название модуля: Тестовый модуль для МР

Описание модуля: Тестовый модуль для МР

**Компоненты модуля**

Выберите компоненты: (из папки /bitrix/components/ кроме системных в bitrix)

video:images  
video:lang  
video:templates

**Скрипты административных страниц модуля**

Файлы: `it\admin\test_admin.php`  `it\admin\test_admin.php`

После создания модуля новые скрипты можно положить вручную в папку admin.

**Продолжить**

- Заполняются группы полей данные о разработчике и данные о новом модуле;

**⚠ Внимание!** Код партнера должен строго соответствовать тому, что выводится в карточке партнера. Код модуля заполняется латиницей. Название и описание будут выводиться в списке модулей. Все это подробно описано в [документации](#) для разработчиков.

- Опция **Переписать существующие файлы** означает, что необходимо перезаписать информацию, если такой модуль уже установлен в системе;
- При необходимости в группе настроек **Компоненты модуля** выбираются компоненты для включения в дистрибутив создаваемого модуля, т.е они должны быть установлены на текущей установке в папке `/bitrix/components/`. Эти компоненты будут скопированы в папку `/install/` нового модуля. При установке автоматически скопируются из этой папки в `/bitrix/components/`, и соответственно удалятся при деинсталляции.



**⚠ Внимание!** Все пользовательские компоненты должны находиться в папке `/bitrix/components/`. Системные компоненты находятся в папке `/bitrix/components/bitrix/`, содержимое которой обновляется системой обновлений и не должно изменяться пользователями. Об это подробно описано в документации по [созданию компонентов](#).

- Если у модуля имеются административные страницы, то с помощью кнопки **Обзор...** можно их включить в дистрибутив, они попадут в папку `/admin/` нового модуля. При установке будут созданы ссылки на эти страницы в папке `/bitrix/admin/`. Кроме того, они появятся в основном административном меню.
- Для перехода к следующему шагу служит кнопка **Продолжить**, после нажатия которой модуль появится в списке решений, и можно продолжать его разработку уже в ядре:

**Доступные решения**

Решения, доступные в системе.  
Вы можете устанавливать, удалять и стирать решения. Для того чтобы стереть решение из системы его необходимо сначала удалить.

	Название	Разработчик	Версия	Дата обновления	Статус
	<b>Конструктор модуля</b> (bitrix.mpbuilder)	<a href="#">1С-Битрикс</a>	1.0.5	19.03.2012	Установлен
	<b>Тестовый модуль для MP</b> (bitrixtest.bitrixtest)	<a href="#">Битрикс</a>	1.0.0	22.03.2012	Установлен

Выбрано: 2

## Шаг 2. Выделение языковых фраз.

На данном шаге происходит:

- Сканирование всех php-файлов модуля, если в коде нового модуля имеются кириллические символы, то выводится список этих страниц:



Шаг 2

### Выделение языковых фраз

Выбор модуля

Текущий модуль: bitrixtest.bitrixtest

Выбор файлов

Включить обработку файла	Текущая кодировка файла
<input checked="" type="checkbox"/> /admin/test_admin.php	utf8

Продолжить

Система автоматически определяет кодировку **utf8** или **cp1251**. В дальнейшем скрипты будут автоматически перекодированы в кодировку сайта (а перед архивацией - в **cp1251**, как требует документация).

- Языковые фразы сохраняются в соответствующий **lang**-файл, а вместо фраз в код страниц модуля вставляется вызов [GetMessage](#). В качестве ключа используется транслитерация исходной фразы - это позволяет в автоматическом режиме создавать понятные ключи:

```

6 $B = GetMessage("BITRIXTEST_BITRIXTEST_PROVERKA_KIRILLICY");
7
8 IncludeModuleLangFile(__FILE__);
9 if ($_REQUEST['file'])
10     $APPLICATION->SetTitle(GetMessage("BITRIXTEST_BITRIXTEST_ISHODNYY_KOD_FAYLA"));
11 elseif ($_REQUEST['scan'])
12     $APPLICATION->SetTitle(GetMessage("BITRIXTEST_BITRIXTEST_SKANIROVANIE"));
13 elseif ($_REQUEST['module'])
14     $APPLICATION->SetTitle('API '.GetMessage("BITRIXTEST_BITRIXTEST_MODULA").htmlspecialchar);
15 elseif ($_REQUEST['search'])
16     $APPLICATION->SetTitle(GetMessage("BITRIXTEST_BITRIXTEST_POISK_PO"));
17 elseif ($_REQUEST['show_diff'])
18     $APPLICATION->SetTitle(GetMessage("BITRIXTEST_BITRIXTEST_IZMENENIA_API_POSLE"));
19 else
20     $APPLICATION->SetTitle(GetMessage("BITRIXTEST_BITRIXTEST_AKTUALNOE_OPISANIE"));

```

**⚠ Будьте осторожны!** В процессе такой работы исходный файл переписывается. При разборе учитывается **html, php, js, css, \**

- Языковой файл создается автоматически:

```

1 <?
2 $MESS["BITRIXTEST_BITRIXTEST_PROVERKA_KIRILLICY"] = "Проверка кириллицы";
3 $MESS["BITRIXTEST_BITRIXTEST_ISHODNYY_KOD_FAYLA"] = "Исходный код файла";
4 $MESS["BITRIXTEST_BITRIXTEST_SKANIROVANIE"] = "Сканирование API";
5 $MESS["BITRIXTEST_BITRIXTEST_POISK_PO"] = "Поиск по API";
6 $MESS["BITRIXTEST_BITRIXTEST_IZMENENIA_API_POSLE"] = "Изменения API после обновления";
7 $MESS["BITRIXTEST_BITRIXTEST_AKTUALNOE_OPISANIE"] = "Актуальное описание API";
8 $MESS["BITRIXTEST_BITRIXTEST_MODULA"] = "модуля ";
9 ?>

```

- Если файл был редактирован, то новые фразы дописываются в конец.

- Если фразы были в исходном файле (или ранее по коду), то они используются повторно.
- Если ключи совпадают, в конце дописывается цифра.
- Для скриптов папки **/admin/** в самое начало скрипта дописывается код подключения ядра, если не был найден. Это необходимо для работы языковых функций.

Таким образом, скрипт можно постоянно дорабатывать и применять выделение фраз повторно.

### Шаг 3. Создание архива

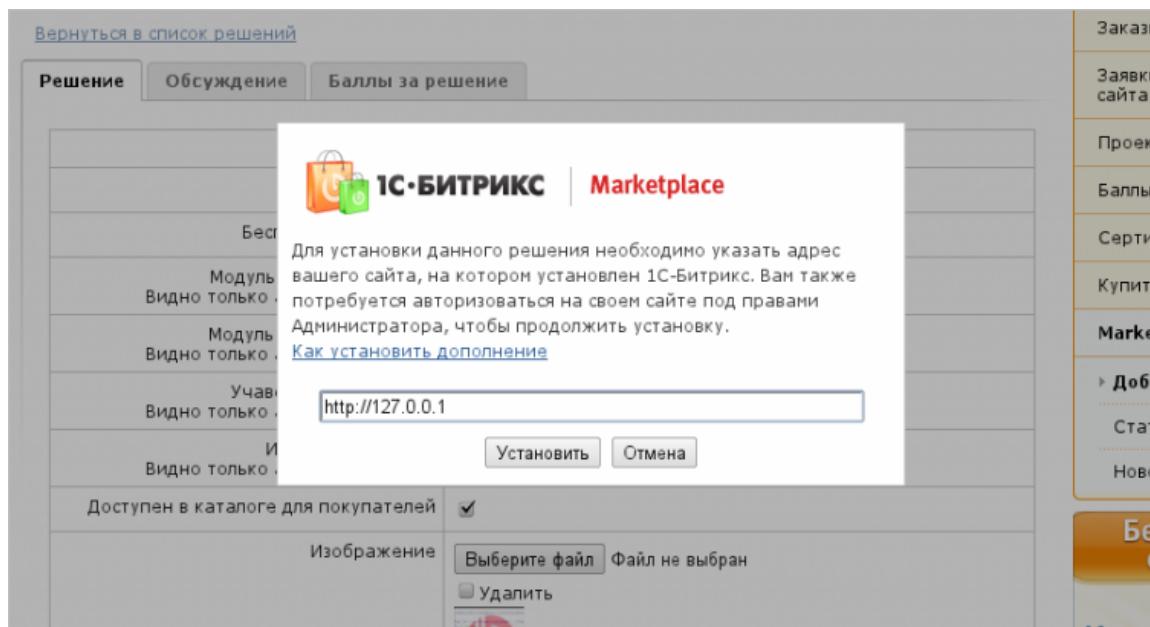
На данном шаге:

- Все скрипты модуля будут сконвертированы в кодировку **cp1251**, затем будет создан архив **.last\_version.tar.gz** в папке **/bitrix/tmp/<название.модуля>/**:

The screenshot shows the 'Шаг третий: создание архива' (Step 3: Create archive) screen. At the top, there's a breadcrumb navigation: Рабочий стол > Настройки > bitrix.mpbuilder > step3.php. Below it is a note: 'Все скрипты модуля будут сконвертированы в cp1251, затем будет создан архив .last\_version.tar.gz, который надо отправить в marketplace.' A green box highlights a success message: 'Архив создан успешно' (Archive created successfully) with the link '/bitrix/tmp/bitrixtest.bitrixtest/.last\_version.tar.gz'. The main form is titled 'Шаг 3' and 'Создание архива'. It has a 'Выбор модуля' (Module selection) section with fields for 'Текущий модуль:' (Current module: bitrixtest.bitrixtest) and 'Версия модуля:' (Module version: 1.0.1). There's also a checkbox 'обновить версию' (Update version). A 'Продолжить' (Continue) button is at the bottom.

Здесь можно изменить версию модуля в одноименном поле.

- Далее файл **.last\_version.tar.gz** необходимо загрузить в карточке партнера [Marketplace](#).



- После пробуем установить себе созданный модуль и дожидаемся, когда решение будет отмодерировано.

#### Шаг 4. Сборка обновления

Данный шаг автоматизирует сборку обновлений любого модуля: все файлы модуля, измененные после даты, которая хранится в `install/version.php`, попадают в архив обновления. Также сборка обновлений проверяет дату изменения файлов в установленных компонентах, копирует их в папку модуля, затем помещает в архив с обновлением. В обновлении лежит включается файл `updater.php`, который обновляет компоненты на сайтах клиентов. Перекодировка файлов из cp1251 в utf8 и обратно происходит автоматически.



Шаг 4

Сборка обновления

Выбор модуля

Текущий модуль: bitrixtst.bitrixtst ▾

Версия обновления: 1.0.1  обновить дату и версию в ядре модуля  
(следующие обновления будут собираться от текущего момента)

Обработка установленных компонентов: Модуль не содержит компонентов в папке /install

Описание обновления: Исправлены некоторые ошибки в коде

При передаче исполняемого кода через браузер у вас должны быть права на обход проактивного фильтра модуля проактивной защиты.

Скрипт обновления updater.php:

```
<?
if(IsModuleInstalled('bitrixtst.bitrixtst'))
{
    if (is_dir(dirname(__FILE__).'/install/components'))
        $updater->CopyFiles("install/components",
"components/");
}

/*
```

**Продолжить**

- Если включить опцию **Обновить дату и версию в ядре модуля**, то исходный файл модуля будет переписан и следующие обновления будут собираться от текущей версии.
- В поле **Описание обновления** добавляется текстовая информация по изменениям в обновлении.
- В поле **Скрипт обновления updater.php** добавляется сам пользовательский скрипт обновления. По умолчанию в этом поле дан пример скрипта.
- После нажатия кнопки **Продолжить** будет создан архив с обновлением **/bitrix/tmp/<модуль>/<версия\_модуля>.tar.gz**, который необходимо будет загрузить в карточке партнера [Marketplace](#).

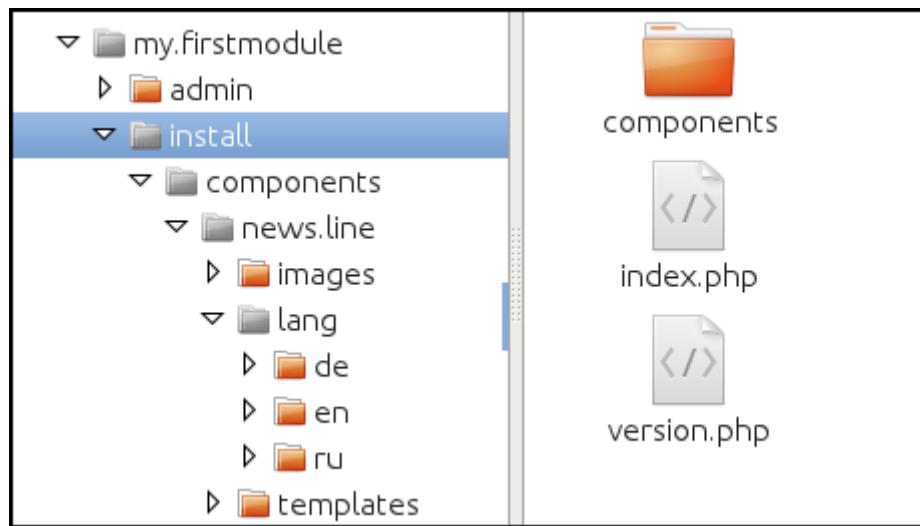
**⚠ Примечание:** Шаги 2-4 можно повторять сколько угодно раз в процессе разработки и обновлении модулей.



## Сборка обновлений модуля

У **Конструктора модулей** (**bitrix.mpbuilder**) для Marketplace есть возможность собирать обновления для своих модулей. При этом не важно, созданы они были при помощи Конструктора или "руками".

Например, имеем некий модуль, который предоставляет компоненты.



После его выпуска поправили шаблон, при этом добавилось несколько языковых фраз.



Выделенный серым фрагмент кода был добавлен. Обратите внимание, что слово **Да** уже присутствовало ранее в языковом файле.

Перед сборкой обновления надо выделить языковые фразы. Можно это сделать вручную, а можно при помощи **Конструктора модулей**.

Для этого открываем *Шаг 2 "Выделение языковых фраз"* и выбираем свой модуль в списке. Он показывает список файлов (за пределами `/lang`), которые содержат кириллические символы.

Шаг второй: выделение языковых фраз

Рабочий стол > Настройки > bitrix.mpbuilder > step2.php

Перед публикацией модуля необходимо выделить языковые фразы чтобы при установке клиенту они конвертировались в кодировку сайта.  
Исходные файлы будут переписаны, убедитесь, что у вас сохранены резервные копии файлов.

Шаг 2

Выделение языковых фраз

Выбор модуля  
Текущий модуль: my.firstmodule

Выбор файлов  
Включить обработку файла Текущая кодировка файла  
 /install/components/news.line/templates/.default/template.php cp1251

Продолжить

Нажмите **Продолжить**. Теперь языковые фразы оказались в языковом файле шаблона. Старые фразы были использованы, а новые дописаны в конец.



**⚠ Примечание:** Ваши компоненты могут лежать или непосредственно в /install/components, или в /install/components/<мое пространство имен>. Поиск языкового файла идет автоматически на основе стандартной структуры компонента 2.0 (комплексные компоненты также поддерживаются).

Перейдите к Шагу 4 через меню (третий шаг сборки архива модуля пропустите). Он использует дату и версию модуля из файла /install/version.php для сборки обновления.

```
1 <?
2 $arModuleVersion = array(
3     "VERSION" => "3.0.4",
4     "VERSION_DATE" => "2012-03-10 10:20:06"
5 );
6 ?>
```

Если включить соответствующую опцию **Версия обновления**, эти данные будут обновлены автоматически.

Шаг 4

Сборка обновления

Выбор модуля

Текущий модуль: my.firstmodule

Версия обновления: 3.0.5  обновить дату и версию в ядре модуля (следующие обновления будут собираться от текущего момента)

Обработка установленных компонентов:  Скопировать измененные файлы компонентов из /bitrix/components/ в ядро модуля

Пространство имен компонентов: /bitrix/components/my

Компоненты модуля лежат непосредственно в папке install, чтобы проверить изменения в публичных компонентах, должно быть указано, где они лежат.  
Также убедитесь что код updater.php правильно копирует компоненты.

Описание обновления:  
Добавлена возможность прервать выполнение

При передаче исполняемого кода через браузер у вас должны быть права на обход проактивного фильтра модуля проактивной защиты.

Скрипт обновления updater.php:

```
<?
if(IsModuleInstalled('my.firstmodule'))
{
    if (is_dir(dirname(__FILE__).'/install/components'))
        $updater->CopyFiles("install/components", "components/my");
}

/*
```

Продолжить



Если были изменены уже установленные компоненты модуля в `/bitrix/components`, на этом этапе можно автоматически скопировать изменения в модуль. Тут есть два варианта:

- если компоненты модуля лежат без указания пространства имен (как в моем примере), то надо указать пространство имен для синхронизации изменений (оно же подставится в код `updater.php`).
- в противном случае ничего указывать не надо (например, `/install/components/bestpartner/news.line`).



#### Обновление собрано

Архив обновления можно скачать по ссылке: </bitrix/tmp/my.firstmodule/3.0.5.tar.gz>.  
[Загрузить в marketplace](#)

#### Список файлов в архиве:

`/3.0.5/install/version.php`  
`/3.0.5/install/components/news.line/templates/.default/template.php`  
`/3.0.5/install/components/news.line/templates/.default/lang/ru/template.php`  
`/3.0.5/description.ru`  
`/3.0.5/updater.php`

**⚠ Обратите внимание**, если разработка компонентов велась в публичной части, а языковые фразы не выделены, то порядок немного другой. Надо сначала перейти на шаг 4, собрать обновление без изменения файла `version.php`, а потом выделить языковые фразы на шаге 2. И уже после этого вернуться на шаг 4 для окончательной сборки обновления. Если что-то пошло не так, руками измените дату и версию модуля в файле `/install/version.php`.

## Глава 9. Программирование в Bitrix Framework

### Цитатник веб-разработчиков.

**Хан Эрли:** В конечном счёте качество сайта определяется не платформой, а мастерством разработчика.

Программирование в **Bitrix Framework** не представляет особой сложности. Это обычное программирование на PHP. Особенности, конечно, есть, но это не те особенности, которые делают из "программистов на Битрикс" какую-то особую касту. Всё в пределах досягаемости для неленивого ума.

В этой главе освещаются вопросы особенности программирования на **Bitrix Framework**.

### Золотые правила

#### Цитатник веб-разработчиков.

**Степан Овечинников:** Неописуемы безобразия, которые можно увидеть в коде человека, знающего мало, а делающего много.

Перед тем как начать работать в **Bitrix Framework**, необходимо понять основные правила, следование которым поможет избежать многих и многих ошибок:

- **Если вы сопровождаете живой сайт и заказчик просит провести работы по доделке-расширению функционала**, то сначала проверьте, а пользуются ли вообще этим функционалом. Проверить можно так:

- Статистика посещаемости - [Внутренняя статистика](#), [Яндекс.Метрика](#), [Google Analytics](#) или [Liveinternet](#)
- Артефакты внутри Битрикса: результаты веб-форм, заказы, скачивания (в Яндекс.Метрике есть нужный отчёт и отлично работает) и т.д.
- Карты кликов в Яндекс.Метрике.

Если блок мёртвый, то лучше его вообще убрать или переформулировать ТЗ.

- **Если вы хотите внести какие-то изменения в работе сайта**, то:

- Сначала формализуйте свои требования на листе бумаги, а не кидайтесь править код.
- После этого заново просмотрите все случаи использования на сайте модифицируемого вами блока. Убедитесь, что всё логично. Очень часто делают небольшие, казалось бы, изменения, а потом оказывается, что страдает связанный функционал, о котором не подумали или забыли.



- После того, как вы формализовали потребности в изменениях, посмотрите, какие сущности эти требования затрагивают.
  - Только после этого подумайте, какие средства использовать для достижения своих целей.
- **Способы внесения изменений и желательный порядок их применения:**
- сначала попробуйте сделать это [редактированием шаблона](#) самого сайта и файлов CSS;
  - если предыдущее невозможно, то попробуйте сделать это средствами [редактирования страницы](#) сайта;
- ⚠ Примечание:** К этому пункту нужно подходить осторожно. Бездумное добавление кода может привести к тому, что компоненты будут обрамляться кодом, который должен быть в шаблоне компонента, а не на странице.
- при невозможности реализации задачи с помощью первых вариантов переходите к [редактированию шаблонов компонента](#) и файлов CSS компонента, либо изменяйте вывод данных с помощью файлов [result\\_modifier.php](#) и [component\\_epilog.php](#).
  - используйте [обработчики событий](#), которые позволяют решать очень широкий спектр задач.
  - [кастомизация компонента](#) или разработка [собственного компонента](#) или [модуля](#) - последний из возможных вариантов получения нештатного функционала.
- **Не рекомендуется писать код HTML в код PHP для изменения представления данных.** В компонентах 2.0 разделены логика и представление. Логика - это сам компонент, представление - это шаблон вывода компонента. Шаблон существенно проще, чем компонент в целом. Нет необходимости изменять логику компонента для изменения особенностей показа его данных. Для одной логики может быть несколько представлений, в том числе зависящих от шаблона текущего сайта. Представление (шаблон вывода) может быть написано [на любом шаблонном языке](#), который можно подключить из PHP. Например, шаблоны могут быть на PHP, Smarty, XSL и т.д.
- **Собственные компоненты и шаблоны - в собственном пространстве имен.** Кастомизируя штатные компоненты и шаблоны, разрабатывая собственные, размещайте их в собственном пространстве имен. При обновлении системы все внесенные изменения в пространстве **bitrix** затираются.
- **При работе с компонентами не надо обращаться к базе напрямую.** Концепция работы с продуктом предполагает работу с данными через [функции API](#). Структура данных может меняться от версии к версии, а функции сохраняют обратную совместимость. Мы настоятельно не рекомендуем использовать прямые запросы к БД, т.к. это может нарушить целостность данных и привести к неработоспособности сайта. В силу вышесказанного структура таблиц не афишируется.

- **Не рекомендуется править код ядра** в силу нескольких причин:
  - при обновлении системы внесенные изменения затрутся;
  - при изменении ядра владелец лицензии теряет право на техническую поддержку;
  - при изменении ядра разработчиком сайта возможна некорректная работа системы, так как ядро - сложная система, требующая учета работы всех модулей.

**Ядро продукта** - файлы, находящиеся в директории */bitrix/modules/* а также файлы системных компонентов: */bitrix/components/bitrix/*.

**⚠ Внимание!** Файлы, к которым нельзя обращаться напрямую (они не должны выполняться, будучи вызванными напрямую по адресу в браузере), должны содержать в начале следующий код:

```
<if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
```

К таким файлам относится все, что работает внутри продукта, например: шаблоны сайтов, шаблоны компонента, файлы *.parameters.php* и *.description.php*.

## **Файл init.php**

**Цитатник веб-разработчиков.**

**Максим Месилов:** С течением времени на проекте накапливается ворох «общих» функций, обработчиков событий, специфических библиотек и т.д. Очень часто всё это валят в *init.php* и там чёрт ногу сломит.

***init.php*** - необязательный файл в рамках структуры файлов **Bitrix Framework**. Он автоматически подключается в прологе.

Файл может содержать в себе инициализацию обработчиков событий, подключение дополнительных функций - общие для всех сайтов. В этом случае он располагается по пути */bitrix/php\_interface/init.php*. Для каждого отдельного сайта может быть свой аналогичный файл. В этом случае он располагается по пути */bitrix/php\_interface/ID сайта/init.php*. Если есть оба файла, то система подключит оба, но первым при этом будет файл */bitrix/php\_interface/init.php*.

Чтобы **init.php** не превращался в свалку непонятного кода следует код размещать логически группируя по файлам и классам.

Рекомендуется придерживаться следующих самых общих правил:



1. **init.php** содержит только подключения файлов. Причем подключать эти файлы желательно через **\_\_autoload**. Штатными средствами это делается так:

```
CModule::AddAutoloadClasses(  
    '', // не указываем имя модуля  
    array(  
        // ключ - имя класса, значение - путь относительно корня сайта к файлу с  
        // классом  
        'CMyClassName1' => '/path/cmyclassname1file.php',  
        'CMyClassName2' => '/path/cmyclassname2file.php',  
    )  
);
```

5. Если функционал используется только на одном из сайтов в системе, то он выносится в свой **init.php**;
6. Обработчики событий лучше группировать в одном файле и тщательно аннотировать где они используются и какая задача перед ними стоит.

### Как избежать проблем при редактировании init.php без ftp/ssh доступа

Ошибка в файле **init.php** приводит к полной потере работоспособности сайта и невозможности что-то исправить без доступа к файлу через ftp/ssh напрямую с диска. Такое возможно, например, в случаях:

- У клиента хостинг под Windows, а у вас "серый" IP и ftp не работает.
- Хостинг под Linux, но php и ftp работают из-под разных пользователей и файл недоступен для редактирования по ftp.
- У клиента свой сервер и доступ по ftp он не дает.

Если доступ есть только через веб, то один из наиболее простых способов - вынос всего вашего кода во внешний файл и подключение его примерно так:

```
if (isset($_GET['noinit']) && !empty($_GET['noinit']))  
{  
    $strNoInit = strval($_GET['noinit']);  
    if ($strNoInit == 'N')  
    {  
        if (isset($_SESSION['NO_INIT']))  
            unset($_SESSION['NO_INIT']);  
    }  
    elseif ($strNoInit == 'Y')  
    {
```

```
    $_SESSION['NO_INIT'] = 'Y';
}

if (!isset($_SESSION['NO_INIT']) && $_SESSION['NO_INIT'] == 'Y')
{
    if (file_exists($_SERVER["DOCUMENT_ROOT"]."/bitrix/php_interface/functions.php"))

        require_once($_SERVER["DOCUMENT_ROOT"]."/bitrix/php_interface/functions.php");
}
```

 **Примечание:** Параметр в адресной строке `noinit=Y` - отключает подключение, `noinit=N` - включает подключение. Свой функционал должен быть размещен (в примере) в `/bitrix/php_interface/functions.php`.

При таком подходе вы сможете безболезненно редактировать и допускать ошибки в файле **functions.php**, не боясь оказаться у разбитого корыта неработающего сайта без возможности как-то повлиять на ситуацию.

### init.php или собственный модуль?

У разработчика проектов есть два способа постоянного использования уже созданных наработок: собственный модуль или файл **init.php**. Оба варианта имеют свои плюсы и минусы.

**init.php.** Когда у вас есть классы, единые для нескольких сайтов (при многосайтности, или на одном сервере), то для удобства сами файлы с классами располагаются в одной папке, далее создаются символические ссылки.

Либо, при многосайтности, есть и ещё один способ: воспользоваться уже готовыми папками. Например, папками шаблонов. (Теоретически, с точки зрения системы, это "шаблон", для нас - просто общее хранилище файлов.)

И при втором и первом способах код `include $_SERVER["DOCUMENT_ROOT"]."/bitrix/templates/..."` ведет в одно и то же место для всех сайтов, где можно разместить общие для всех проектов файлы. В том числе никто не мешает создать свою папку для своих классов и уже оттуда их подключать, обращаясь `K /bitrix/templates/my_classes/....`

Использование симлинков на папки с кастомными библиотеками в точно такой же степени обязаны поддерживать совместимость как и модули. (Не важно где именно, при этом, они будут располагаться). Нет никакой особой разницы в подходах при поддержке и разработке, только охват проектов в первом случае ограничен рамками одного сервера.



**Модули.** Использование **init.php** будет предпочтительным если создаются проекты, которые точно всю жизнь проживут на одном сервере, и вы хотите максимально минимизировать затраты на поддержку кастомных библиотек. Желательно также чтобы эти библиотеки поддерживал один и тот же человек. Но если планируется эти наработки также использовать для заказных проектов, то дальнейшее делать модуль.

Модуль больше подходит для распространяемых API. При этом придется поддерживать и их интерфейс и формат ответа и другие параметры. Иначе: случайное обновление на одном из сайтов, использующих модуль, может оказаться роковым в случае изменения интерфейсов или форматов ответов API. Выигрыш спорный. С одной стороны - выигрываете, с другой - должны обеспечить пожизненные гарантии для всех использующих API модуля сайтов.

## UTF-8 или Windows-1251 ?

**Цитатник веб-разработчиков.**

*Denis Sharomov:* Если известно, что сайт не будет многоязычным (в обозримом будущем), не вижу причин не использовать эту кодировку (Win-1251).

И альтернативное мнение:

**Цитатник веб-разработчиков.**

*Зайцев Артемий:* Если есть возможность делать в UTF, надо делать в UTF.

Перед создателем сайтов всегда встает проблема: в какой кодировке создавать проект. В русскоязычном интернете используются две кодировки:

**UTF-8** (от англ. **Unicode Transformation Format**) — в настоящее время распространённая кодировка, реализующая представление Юникода, совместимое с 8-битным кодированием текста.

и

**Windows-1251** (или **cp1251**) — набор символов и кодировка, являющаяся стандартной 8-битной кодировкой для всех русских версий Microsoft Windows.

Считается, что UTF-8 - более перспективна. Но у любой вещи есть недостатки. И решение об использовании какой-то кодировки только потому, что она перспективна, без учета многих других факторов, не представляется правильным. Выбор будет



оптимальным только тогда, когда он полностью учитывает все нюансы конкретного проекта. Другое дело, что предусмотреть все нюансы - само по себе весьма не просто.

Мы не даём рекомендаций по использованию той или иной кодировки. Мы просто составили сравнительную таблицу их особенностей. А решать что выбрать - это дело разработчика проекта.

Свойство	UTF-8	Windows 1251
<b>Общего характера</b>		<ul style="list-style-type: none"> <li>Смена кодировки действующего крупного сайта с Windows-1251 на UTF-8 может вызвать серьёзные дополнительные трудовые и финансовые издержки.</li> </ul>
<b>Многоязычность</b>	Кодировка позволяет использовать разные языки как в публичной, так и в административной части сайта.	<ul style="list-style-type: none"> <li>Русский и английский без проблем работают с Windows-1251, если точно не будет потребности в других языках, то и нет потребности в UTF-8.</li> </ul>
<b>Большое число символов. Возможность использования спецсимволов.</b>	Есть. Но надо учитывать возможности браузеров.	Штатно нет. Есть возможность замены спецсимволов на "костыли", например, © на &copy; или × (знак умножения) на &times;. Однако это повышает требования к уровню подготовки контент-менеджера и создаёт проблемы при переносе данных из другой базы данных. Кроме того, в Bitrix Framework есть поля, которые не используют визуальный редактор, например, название страницы или название элемента инфоблока. Это также усложняет поддержку проекта силами низкоквалифицированных сотрудников.
<b>Скорость работы</b>		<ul style="list-style-type: none"> <li>При работе сайта идет подмена всех функций работы со строками на mb_*. Это значит, что весь текст</li> </ul>

будет перекодироваться в кодировку сайта.

- utf\_strlen зависит от длины строки, соответственно обычный strlen работает в 4 раза быстрее мультибайтового: 0.0004 против 0.0013 на тысяче итераций. По замерам на [мониторе производительности](#)

[Производительности](#) это выливается в 10-15% разницу в скорости работы реального сайта.

<b>Минимизация объема проекта.</b>	Проект на UTF-8 будет заведомо "тяжелее", в силу того что строки в этой кодировке занимают в два раза больше места, чем строки в однобайтной Windows-1251. Размер сайта и базы данных будет в 1,2 - 1,5 раз больше.
<b>Поддержка большинством js-фреймворков</b>	Поддерживается без проблем. Сложности в реализации.
<b>Поддержка MS SQL</b>	По техническим причинам, данные в MS SQL должны храниться и хранятся в Windows-1251. Требуется дополнительная настройка. Нет проблем.
<b>Импорт CSV</b>	Excel не сохраняет в UTF-8. Требуется пересохранение созданного файла в этой кодировке с помощью другого редактора. Нет проблем.
<b>Импорт из 1С</b>	Сайты на UTF-8 работают без проблем при интеграции через SOAP с такими

системами как, например, 1С.

## **Связанные с Bitrix Framework**

### **Возможность**

**сделать сайты в разной кодировке по системе многосайтности.**

Невозможно. Все сайты на одном ядре должны быть в одной кодировке.

**Поддержка различных хостингах**

**на**

При работе с Bitrix Framework необходимо подключение опции mbstring.func\_overload в значении большем или равном 2. Это [разрешается не на всех хостингах](#).

Работает на любых хостингах.

**Размещение продуктов на виртуальной машине BitrixVM.**

**на**

По умолчанию.

Требует дополнительных действий по настройке.

**Корректное отображение пунктов сайта меню**

При использовании данной кодировки такая проблема возможна. Решается пересохранением каждого файла в UTF-8.

**Импорт исходников в IDE, например, в [eclipse](#) [pdt](#)**

При выставленном в настройках проекта UTF-8, в коде ядра Битрикса портятся комментарии.

Нет проблем.

## **Разные мелочи**

**Взаимодействие с WordPress (блог-клиенты, trackback и ping'и)**

**Есть**

**Нет**

**Редактирование файлов по FTP**

Нет, FAR не поддерживает UTF.

**Возможно**

**через FAR**

**Поддержка  
большинством  
редакторов** Требуется редактор, который поддерживает кодировку Нет проблем. UTF-8 без [BOM](#).

## Работа с базами данных

**Цитатник веб-разработчиков.**

**Антон Долганин:**

*Новичкам надо поклясться на мануале, что они забудут про прямые запросы, вообще про БД забудут. Когда появится немного опыта и будет казаться что вы зверь в Битриксе - тогда можно почитать о том как происходит общение с БД, но все равно не трогать, подождать еще год-два активной работы с системой. И вот только потом уже можно подружиться с БД полностью, но помнить, что в первую очередь надо всеми правдами и неправдами сделать стандартными методами Битрикс. К БД прибегать только в случае:*

- *Невозможно сделать стандартно без большой нагрузки на PHP;*
- *Невозможно сделать стандартно без большой нагрузки на MySQL.*

*Разработчик должен точно понимать БД. Понимать что изменится, а что нет при ближайшем обновлении.*

Одним из первых впечатлений, возникающих у начинающего осваивать **Bitrix Framework** программиста, бывает непонимание запросов к базе данных. Код **Bitrix Framework** порождает очень большие и не сразу понятные запросы. Несколько экранов не очень понятного SQL традиционно пугают людей, которые редко пишут запросы сложнее `select * from ... where id=...` и уверенных, что объединение таблиц по условиям делается через **where** так же производительно, как и через **on**.

Но всё не так просто. Тема работы с БД включает в себя вопросы:

- узкотехнологические, вроде производительности конкретных выборок;
- вопросы удобного применения технологий работы с БД в больших развивающихся проектах на растущей CMS;



- вопросы стоимости владения и поддержки.

### **Почему в Bitrix Framework запрос получается неудобочитаемым?**

В системе есть инфоблоки, где быстро и комфортно создается структура данных. Есть стандартные компоненты, реализующие наиболее распространенные вещи. Есть API, позволяющее отправлять достаточно произвольные запросы. Вы работаете со всем этим, в итоге ваша достаточно высокоуровневая логика выборки превращается в запрос. Генератор запроса, несмотря на свою сложность, выполняет довольно однообразную работу по переводу логики, изложенной на высоком уровне, в обращения к конкретным полям конкретных таблиц по условиям.

В мощных СУБД есть умные и очень эффективные оптимизаторы запросов, а совсем плохие запросы такие СУБД даже не дают исполнять. Подавляющее большинство установок продуктов на **Bitrix Framework** работают на **MySQL**, чей оптимизатор достаточно слаб. В итоге мы имеем большие запросы, ограниченные средства их изменения с сохранением логики, и механизмы кеширования поверх всего этого.

### **Минус такого положения дел**

Сложные запросы очень велики и вы почти не можете влиять на их структуру. Даже если есть желание заняться их профилировкой и отладкой, вы фактически ограничены теми не слишком богатыми средствами изменения запроса, которые предлагает **Bitrix Framework**. Вы не можете:

- изменять порядок объединений;
- задавать специфические условия на выборки;
- создавать локальные кеширующие временные таблицы.

### **Плюсы такого положения дел**

Плюсы оказываются важнее, чем минусы:

- **Запросы не надо писать руками.** Создание проекта на **Bitrix Framework** обычно предусматривает визуальную настройку готовых компонентов, создания структуры сайта и интеграцию дизайна. Не в каждом проекте приходится кастомизировать компоненты, но даже эта работа не заставляет писать запросы.
- **Безопасность.** Обертки над запросами решают задачу защиты от атак или глупостей разработчика.

## Квалификация разработчика

Как и в любом другом деле, всё решает профессионализм работы с **Bitrix Framework**. И мастерство работы с БД не играет существенной роли. Объективно в большинстве проектов нагрузка на sql (время исполнения) несущественна по сравнению с нагрузкой на процессор и затратами на интерпретацию скриптов. **Тормозит не MySQL, а PHP.**

В связи с этим огромную роль играет правильность проектирования структуры данных, выбор связей и их реализация средствами системы инфоблоков или таблиц. **Устранить "тормоза" на проекте гораздо правильнее и эффективнее грамотным проектированием, чем оптимизацией запроса**, который **Bitrix Framework** генерирует по вашим указаниям.

Использование кеширования позволяет сэкономить на времени исполнения запросов. Грамотное распределение логики по компонентам позволяет вообще их не вызывать. Есть правило: **главная страница сайта (обычно не самая простая) не должна отправлять запросы к БД при включенном кеше.**

Разработчик должен учитывать специфику проекта и правильно выбирать тип таблиц: **InnoDB** или **MyISAM**.

Разработчик должен уметь оценивать уровень задач. Для действительно серьезных проектов есть **Oracle**, **MS SQL** а так же кластеризация **MySQL**.

## Что есть в Bitrix Framework для облегчения жизни простого программиста?

Перед началом работ по оптимизации проектов воспользуйтесь [Монитором производительности](#) для поиска узких мест на сайте.

В составе административной части системы есть инструменты [проверки БД](#) и [её оптимизации](#). Эти действия автоматизированы и вмешательство разработчика в них исключено.

Используйте [инфоблоки 2.0](#). Это включаемый галочкой в административном разделе режим работы инфоблока, когда его поля переносятся в отдельную таблицу. Но у них есть свои особенности, которые надо знать и учитывать.

Используйте настраиваемую из административной части [кластеризацию БД](#) на стандартных механизмах MySQL.

Используйте специфические для вашего проекта [индексы](#). Большой запрос не всегда синоним долгого исполнения. Возьмите тяжелый запрос из системы и сделайте **explain** в консоли. Вы удивитесь как хорошо он покрыт индексами и как мало там будет циклических переборов записей.

## Цитатник веб-разработчиков.

**Денис Шаромов:** Есть миф, что длинный запрос долго исполняется. В действительности, всё зависит от того, какой объем данных приходится обработать базе для исполнения запроса. Разбор собственно запроса занимает ничтожно малую часть времени. Основное время занимает фильтрация и сортировка. Время фильтрации зависит от параметров фильтра и индексов, а сортировка - от числа записей в выборке.

## Работа с БД

Перед тем как начать работать с БД усвойте, что:

**⚠ Внимание!** Прямое обращение к базе данных в рамках **Bitrix Framework** не приветствуется. Более того, если речь идет о системных таблицах самого **Bitrix Framework**, это не просто не приветствуется, это не поддерживается. Необходимо с ними работать через API системы, так как физическая структура БД может измениться, а работа даже самого древнего API гарантирована.

В силу этого предупреждения названия таблиц не афишируются. Если, все же, вы возьмёте на себя ответственность прямой работы с базой данных, то учтите, что все таблицы от **Bitrix Framework** начинаются с **b\_**. Соответственно ваши префиксы должны быть другими: необходимо чтобы имена ваших таблиц не пересеклись с битриксовыми. Вполне возможна ситуация, когда в новых обновлениях появится таблица с таким же именем, как и созданная вами (если вы используете префикс **b\_**). В лучшем случае обновление не установится.

Для работы с собственными таблицами используйте методы глобальной переменной **\$DB** (класс [CDatabase](#)).

Детально об оптимизации БД читайте в главе [Оптимизация базы данных](#) Курса для хостеров.

## Языковые файлы

### Цитатник веб-разработчиков.

**Антон Долганин:** В компонентах языковые фразы выношу в лангфайлы, просто потому что это системная часть и там хотелось бы видеть порядок.



**Языковой файл** - представляет из себя PHP скрипт, хранящий переводы языковых фраз на тот или иной язык. Данный скрипт состоит из массива **\$MESS**, ключи которого - идентификаторы языковых фраз, а значения - переводы на соответствующий язык.

Языковые файлы не являются обязательными.

#### Пример языкового файла для русского языка:

```
<?
$MESS ['SUP_SAVE'] = "Сохранить";
$MESS ['SUP_APPLY'] = "Применить";
$MESS ['SUP_RESET'] = "Сбросить";
$MESS ['SUP_EDIT'] = "Изменить";
$MESS ['SUP_DELETE'] = "Удалить";
?>
```

#### Пример языкового файла для английского языка:

```
<?
$MESS ['SUP_SAVE'] = "Save";
$MESS ['SUP_APPLY'] = "Apply";
$MESS ['SUP_RESET'] = "Reset";
$MESS ['SUP_EDIT'] = "Change";
$MESS ['SUP_DELETE'] = "Delete";
?>
```

#### Примеры работы

просмотр всего массива словаря:

```
<? echo'<pre>';print_r($MESS);echo'</pre>'; ?>
```

получение названия месяца в двух падежах:

```
<?
echo $MESS['MONTH_'.date('n')]; // Июнь
echo $MESS['MONTH_'.date('n').'_S']; // Июня
?>
```

Для каждого языка существует свой набор языковых файлов, хранящихся в подкаталогах **/lang/** (подробнее смотрите на страницах документации [структуре файлов системы](#), [структуре файлов модуля](#)).

Языковые файлы, как правило, используются в [административных скриптах](#) модулей или в [компонентах](#) и в зависимости от этого подключаются одной из следующих функций:

- [IncludeModuleLangFile](#) - в административных скриптах;
- [IncludeTemplateLangFile](#) - в компонентах;

Для удобства поиска и дальнейшей модификации языковых фраз можно пользоваться параметром страницы [show\\_lang\\_files=Y](#), позволяющим быстро найти и исправить ту или иную языковую фразу в модуле **Перевод**.

### **Языковые файлы в собственных компонентах**

При создании собственного компонента языковой путь должен выглядеть следующим образом:

```
/bitrix/templates/[шаблон_сайта].default/components/[пространство_имен]/[имя_компонента]/[имя_шаблона_компонента]/lang/[код_языка]/template.php
```

Где код языка, к примеру = **ru**.

В таком случае языковые файлы подключаются автоматически.

Для подключения из компонента языковых сообщений другого компонента можно использовать следующую функцию:

```
function IncludeComponentLangFile ($abs_path, $lang = false)
{
if ($lang === false) $lang = LANGUAGE_ID;

global $BX_DOC_ROOT;

$filepath = rtrim (preg_replace ("\"[\\\\V]+\"", "/", $abs_path), "/");

if (strpos ($filepath, $BX_DOC_ROOT) !== 0)
{
return;
}

$relative_path = substr ($filepath, strlen ($BX_DOC_ROOT));
```



```
if (preg_match ("~^/bitrix/components/([-a-zA-Z0-9_.%]+)/([-a-zA-Z0-9_.%]+)/templates/([-a-zA-Z0-9_.%]+)/(.*$~", $relative_path, $matches))
{
    $lang_path = $BX_DOC_ROOT."/bitrix/components/$matches[1]/$matches[2]/templates/$matches[3]/lang/$lang/$matches[4]";
    __IncludeLang ($lang_path);
    return;
}

if (preg_match ("~^/bitrix/components/([-a-zA-Z0-9_.%]+)/([-a-zA-Z0-9_.%]+)/(.*$~", $relative_path, $matches))
{
    $lang_path = $BX_DOC_ROOT."/bitrix/components/$matches[1]/$matches[2]/lang/$lang/$matches[3]";
    __IncludeLang ($lang_path);
    return;
}
```

## Замена языковых фраз продукта

Иногда при разработке требуется заменить какие-то слова или фразы в компонентах или модулях.

Коснемся этой технологии, суть которой в том, что после подключения языкового файла фразы продукта заменяются на определенные разработчиком.

Путь к файлу замен:

```
/bitrix/php_interface/user_lang/<код языка>/lang.php
```

**⚠ Примечание:** если в **php\_interface** нет нужных папок, их следует создать.

В файле должны определяться элементы массива **\$MESS** в виде **\$MESS['языковой файл'][‘код фразы’] = ‘новая фраза’**, например:

```
<?
$MESS["/bitrix/components/bitrix/system.auth.form/templates/.default/lang/ru/template.php"]
["AUTH_PROFILE"] = "Мой любимый профиль";
$MESS["/bitrix/modules/main/lang/ru/public/top_panel.php"]['top_panel_tab_view'] = "Смотрим";
```



```
$MESS["/bitrix/modules/main/lang/ru/interface/index.php"]['admin_index_sec'] =  
"Проактивка";  
?>
```

Первая строка меняет текст ссылки в компоненте формы авторизации; вторая строка меняет название вкладки публичной панели; третья меняет строку для индексной страницы панели управления.

 **Важно!** Возможны проблемы сопровождения этого файла при изменении кода фразы или расположения языкового файла.

## Интерфейс "Эрмитаж" с точки зрения разработчика

### Подключение панели

После авторизации на сайте для пользователя с соответствующими правами становится доступна панель **Панель управления** в верхней части страницы. С ее помощью можно:

- управлять параметрами текущего раздела;
- перейти к редактированию текущей страницы и включаемых областей;
- добавить и изменить меню текущего раздела;
- настроить параметры компонентов;
- быстро перейти в административный раздел сайта;
- и многое другое.

 **Примечание:** С подробным описанием интерфейса панели вы можете ознакомиться в курсе [Контент-менеджер](#).

Код для подключения административной панели задается в служебной области шаблона дизайна сайта сразу после тега <body> перед началом первой таблицы.

```
<?  
$APPLICATION->ShowPanel();  
?>
```

 **Примечание:** Если пользователь не обладает достаточными правами, то панель для него не будет отображена.



Панель также можно и принудительно отображать для требуемой группы или отдельных пользователей. Для этого необходимо воспользоваться опцией **Всегда показывать панель для пользователей** в настройках **Главного модуля**, закладка **Настройки**.

В данном случае панель будет отображаться, но набор кнопок на ней будет зависеть от прав пользователя.

**⚠ Примечание:** в концепции Эрмитажа основные действия выносятся в toolbar, остальные остаются в контекстном меню.

## Добавление кнопок на панель управления

При создании собственных проектов может возникнуть потребность в создании новых кнопок на **Панели управления**. Добавление кнопок на панель управления можно осуществить следующим образом:

```
$APPLICATION->AddPanelButton(  
    Array(  
        "ID" => "ID кнопки", //определяет уникальность кнопки  
        "TEXT" => "Название кнопки",  
        "TYPE" => "BIG", //BIG - большая кнопка, иначе маленькая  
        "MAIN_SORT" => 100, //индекс сортировки для групп кнопок  
        "SORT" => 10, //сортировка внутри группы  
        "HREF" => "URL для перехода", //или javascript:MyJSFunction()  
        "ICON" => "icon-class", //название CSS-класса с иконкой кнопки  
        "SRC" => "путь к иконке кнопки",  
        "ALT" => "Текст всплывающей подсказки", //старый вариант  
        "HINT" => array( //тултип кнопки  
            "TITLE" => "Заголовок тултипа",  
            "TEXT" => "Текст тултипа" //HTML допускается  
        ),  
        "HINT_MENU" => array( //тултип кнопки контекстного меню  
            "TITLE" => "Заголовок тултипа",  
            "TEXT" => "Текст тултипа" //HTML допускается  
        ),  
        "MENU" => Array(  
            Array( //массив пунктов контекстного меню  
                "TEXT" => "название пункта",  
                "TITLE" => "всплывающая подсказка над пунктом",  
            ),  
        ),  
    ),  
)
```

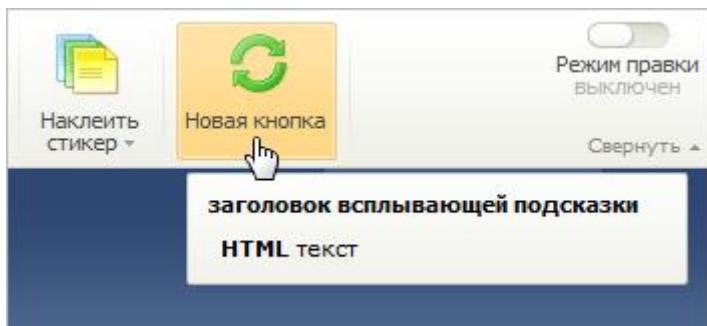


```

    "SORT" => 10, //индекс сортировки пункта
    "ICON" => "", //иконка пункта
    "ACTION" => "Javascript-код",
    "SEPARATOR" => true, //определяет пункт-разделитель
    "DEFAULT" => true, //пункт по умолчанию?
    "MENU" => Array() //массив подменю
)
),
$bReplace = false //перетереть существующую кнопку новыми данными?
);

```

Результат:



## Добавление контекстного меню

Для добавления пунктов контекстного меню к любой кнопке панели используйте следующий код:

```
$APPLICATION -> AddPanelButtonMenu($btnId, $arMenuItem)
```

где:

- **\$btnId** – идентификатор кнопки;
- **\$arMenuItem** – массив пунктов.

**⚠ Примечание:** пересортировку пунктов согласно индексу сортировки можно выполнить с помощью следующей конструкции:

```
"RESORT_MENU" => true
```



## Toolbar компонента

Для вывода toolbar'а компонента используйте следующий код:

```
$this->AddIncludeAreaIcons(  
    Array( //массив кнопок toolbar'a  
        Array(  
            "ID" => "Идентификатор кнопки",  
            "TEXT" => "Название кнопки toolbar'a",  
            "URL" => "ссылка для перехода" //или javascript^MyJSFunction ()  
            "ICON" => "menu-delete", //CSS-класс с иконкой  
            "MENU" => Array(  
                //массив пунктов контекстного меню  
            ),  
            "HINT" => array( //тултип кнопки  
                "TITLE" => "Заголовок тултипа",  
                "TEXT" => "Текст тултипа" //HTML допускается  
            ),  
            "HINT_MENU" => array ( //тултип кнопки контекстного меню  
                "TITLE" => "Заголовок тултипа",  
                "TEXT" => "Текст тултипа" //HTML допускается  
            ),  
            "IN_PARAMS_MENU" => true //показать в контекстном меню  
            "IN_MENU" => true //показать в подменю компонента  
        )  
    )  
);
```

```
//Режим редактирования включён?  
if ($APPLICATION->GetShowIncludeAreas())  
{  
    $this->AddIncludeAreaIcons(Array(  
        //массивы кнопок toolbar'a  
    ));  
}
```



## Контекстное меню элементов списка

- Установить HTML-атрибут **id** для блочного тега:

```
<div id="<?=$this->GetEditAreaID("идентификатор_области")?>">
    <!-- контент блока -->
</div>
```

- В **compote\_epilog.php** определить кнопки контекстного меню с помощью метода:

```
$APPLICATION->SetEditArea($areaid, $arIcons);
```

где:

- **\$areaid** – идентификатор области с контекстным меню;
- **\$arIcons** – массив иконок контекстного меню.

- Метод добавляет кнопку, которая открывает указанный URL в popup-окне:

```
$this->AddEditAction(
    "Идентификатор_области",
    "URL страницы, которая откроется в popup-окне",
    "Название кнопки в toolbar",
    Array(
        "WINDOW" => array("width"=>780, "height"=>500),
        "ICON" => "bx-content-toolbar-edit-icon",
        "SRC" => "/bitrix/images/myicon.gif"
    )
);
```

- Метод добавляет кнопку удаления элемента:

```
$this->AddDeleteAction(
    "Идентификатор_области",
    "URL страницы, удаляющая указанный элемент",
    "Название кнопки",
    Array(
        "CONFIRM" => "Вы действительно хотите удалить этот
элемент?",
        ...
    )
);
```



## Административные страницы в публичке

- Метод генерирует Javascript, открывающий URL в попур-окне:

```
$APPLICATION->GetPopupLink(Array(  
    "URL"=> "URL страницы, которая открывается в попур-окне",  
    "PARAMS" => Array(  
        "width" => 780,  
        "height" => 570,  
        "resizable" => true,  
        "min_width" => 780,  
        "min_height" => 400  
    )  
);
```

- Метод генерирует кнопки управления элементами и разделами инфоблока:

```
CIBlock::GetPanelButtons(  
    $IBLOCK_ID = 0, //ID инфоблока  
    $ELEMENT_ID = 0, //ID элемента инфоблока  
    $SECTION_ID = 0, //ID раздела инфоблока  
    $arOptions = Array(  
        "SECTION_BUTTONS" => true, //генерировать кнопки для управления  
разделами  
        "SESSID" => false, //добавлять ссылку вавторизованный token  
        "RETURN_URL" => "",  
        "LABELS" => Array() //надписи кнопок, по умолчанию берутся из  
настроек инфоблока  
    )  
);
```

## Новые методы буферизации

Усовершенствованные методы буферизации в шаблоне позволяют более не использовать **EndViewTarget()** ввиду того, что конец шаблона вызывает завершение буферизации автоматически.

Теперь есть поддержка стандартного кеширования в компонентах.

- **template.php:**

```
<?>$this->SetViewTarget("sidebar");?>
```



```
<div class="element-filter">
    <!--вывод фильтра -->
</div>

<?{$this->}EndViewTarget();?>

<div class="element-list">
    <!--вывод списка -->
</div>
```

- **header.php:**

```
<div id="sidebar">
    <?{$APPLICATION->}ShowViewContent("sidebar")?>
</div>
```

#### **Методы, доступные в шаблоне (через &this)**

- CBitrixComponentTemplate::SetViewTarget(\$view, \$pos)
- CBitrixComponentTemplate::EndViewTarget()

#### **Методы глобального объекта \$APPLICATION**

- [Cmain::AddViewContent\(\\$view, \\$content, \\$pos\)](#)
- [Cmain::ShowViewContent\(\\$view\)](#)

где:

**\$view** – идентификатор буферизируемой области;

**\$content** – буферизируемый контент;

**\$pos** – сортировка вывода контента.

**⚠ Примечание:** одному идентификатору **\$view** может соответствовать несколько буферов. Последовательность вывода контента определяется сортировкой **\$pos**.



## Немного теории

### Замечания по \$arParams и \$arResult

#### \$arParams

**\$arParams** - это предопределенная для компонента переменная, представляющая собой массив входных параметров компонента. Ключами в этом массиве являются названия параметров, а значениями - их значения.

Перед подключением компонента ко всем значениям параметров применяется функция [htmlspecialcharsEx](#). Исходные значения параметров сохраняются в этом же массиве с теми же ключами, но с префиксом ~. Например, **\$arParams["NAME"]** - входной параметр, к которому применена функция **htmlspecialcharsEx**, а **\$arParams["~NAME"]** - исходный входной параметр.

Переменная **\$arParams** является псевдонимом для члена класса компонента, поэтому все изменения этой переменной отражаются и на этом члене класса. В начале кода компонента должна быть произведена проверка входных параметров, инициализация не установленных параметров, приведение к нужному типу (например, **IntVal()**). Все эти изменения входных параметров будут доступны и в шаблоне. То есть параметры будут там уже проверенными и максимально безопасными. Дублирование подготовки параметров в шаблоне компонента не требуется.

#### \$arResult

**\$arResult** - это предопределенная для компонента переменная, в которую собирается результат работы компонента для передачи в шаблон. Перед подключением файла компонента эта переменная инициализируется пустым массивом **array()**.

Переменная **\$arResult** является псевдонимом для члена класса компонента, поэтому все изменения этой переменной отражаются и на этом члене класса. Значит явно передавать в шаблон эту переменную не нужно, это сделают внутренние механизмы класса компонента.

## Псевдонимы в PHP

Псевдонимы (references) в PHP служат для того, чтобы к одним и тем же данным можно было обратиться по разным именам. Если переменные **\$arParams** и **\$arResult** изменены некоторым образом в коде компонента, то они будут доступны измененными и в шаблоне.

При этом нужно учитывать следующие нюансы:

- Если в компоненте написать код:



```
$arParams = & $arSomeArray;
```

то переменная **\$arParams** будет отвязана от члена класса компонента и привязана к массиву **\$arSomeArray**. В этом случае дальнейшие изменения **\$arParams** не попадут в шаблон компонента.

- Если в компоненте написать код: **unset(\$arParams)**; то это так же разорвет связь между **\$arParams** и соответствующим членом класса компонента.

## Псевдонимы входящих переменных

Каждый компонент имеет набор переменных, в которые он принимает извне коды или другие атрибуты запрашиваемых данных. Например, компонент **bitrix:catalog.section** имеет переменные **IBLOCK\_ID** и **SECTION\_ID**, в которых он принимает и обрабатывает коды каталога и группы товаров соответственно.

Все компоненты, которые входят в состав комплексного компонента, должны иметь единообразный набор переменных. Например, комплексный компонент **bitrix:catalog** и все обычные компоненты, которыми он управляет, работают с переменными **IBLOCK\_ID**, **SECTION\_ID**, **ELEMENT\_ID** и пр.

Если разработчик при размещении комплексного компонента на странице хочет переопределить переменные компонента, то он среди входящих параметров компонента должен задать параметр **VARIABLE\_ALIASES**.

При подключении компонента в режиме SEF этот параметр должен иметь вид:

```
"VARIABLE_ALIASES" => array(
    "list" => array(),
    "section" => array(
        "IBLOCK_ID" => "BID",
        "SECTION_ID" => "ID"
    ),
    "element" => array(
        "SECTION_ID" => "SID",
        "ELEMENT_ID" => "ID"
    )
)
```

Здесь коды массива соответствуют кодам в массиве шаблонов путей. Для каждого пути могут быть заданы свои переопределения переменных.

При подключении компонента не в режиме SEF этот параметр должен иметь вид:



```
"VARIABLE_ALIASES" => array(
    "IBLOCK_ID" => "BID",
    "SECTION_ID" => "GID",
    "ELEMENT_ID" => "ID",
)
```

## Пример 1

Пусть требуется, чтобы компонент **bitrix:catalog**, лежащий в файле /fld/cat.php, работал с путями:

- /catalog/index.php (для списка каталогов)
- /catalog/section/код\_группы.php?ID=код\_каталога (для группы товаров)
- /catalog/product/код\_товара.php?ID=код\_группы (для детальной информации о товаре).

Во входящих параметрах подключения компонента должны быть установлены следующие параметры:

```
"SEF_MODE" => "Y",
"SEF_FOLDER" => "/catalog/",
"SEF_URL_TEMPLATES" => array(
    "list" => "index.php",
    "section" => "section/#SECTION_ID#.php?ID=#IBLOCK_ID#",
    "element" => "element/#ELEMENT_ID#.php?ID=#SECTION_ID#"
),
"VARIABLE_ALIASES" => array(
    "list" => array(),
    "section" => array(
        "IBLOCK_ID" => "ID"
    ),
    "element" => array(
        "SECTION_ID" => "ID",
    ),
)
```

## Пример 2

Пусть требуется, чтобы компонент bitrix:catalog, лежащий в файле /fld/cat.php, работал с путями:

- /fld/cat.php (для списка каталогов)
- /fld/cat.php?BID=код\_каталога&SID=код\_группы (для группы товаров)
- /fld/cat.php?ID=код\_товара&SID=код\_группы (для детальной информации о товаре)

Во входящих параметрах подключения компонента должны быть установлены следующие параметры:

```
"SEF_MODE" => "N",
"VARIABLE_ALIASES" => array(
    "IBLOCK_ID" => "BID",
    "SECTION_ID" => "SID",
    "ELEMENT_ID" => "ID",
)
```

### HTTP POST запросы

Если на сайте не существующие страницы обрабатываются с помощью опции **ErrorDocument** (ErrorDocument 404 /404.php), то при отправке HTTP POST запроса на несуществующую страницу данные POST запроса будут потеряны. Поэтому, такие запросы нужно направлять на скрипты, которые физически существуют.

Один из способов организации ЧПУ в компонентах 2.0 предполагает использование опции ErrorDocument. Для компонентов 2.0 существует унифицированное решение, которое при написании компонентов и шаблонов компонентов позволяет не заботиться о том, осуществляется ли поддержка ЧПУ с помощью опции **ErrorDocument** или **mod\_rewrite**.

При написании в шаблонах компонентов форм, отправляющих данные методом POST, в качестве action необходимо указывать константу POST\_FORM\_ACTION\_URI:

```
<form method="post" action=<?=POST_FORM_ACTION_URI?>>
* * *
</form>
```

В этом случае при работе с использованием опции ErrorDocument POST-запрос будет направлен на физически существующий скрипт, а в остальных случаях - по текущему адресу. При этом для компонента не будет разницы, вызван ли он POST- или GET-запросом. Все необходимые переменные будут соответствующим образом установлены.

## Правила написания исходного кода на PHP

### **Цитатник веб-разработчиков.**

**Сергей Талызёнков:** От чистоты кода зависит очень многое, это быстрая и качественная поддержка проекта разработчиком, читабельность и легкость восприятия для других программистов.

Качество программы начинается с качества ее исходного кода. Основным фактором качества исходного кода программы является его читаемость и понятность. Необходимо формализовать правила написания кода, которые сделают чужой код читабельнее и понятнее.

Правила форматирования кода должны быть едиными во всем проекте. И крайне желательно, чтобы они были очень похожими между проектами.

### **1. Форматирование кода**

#### **1.1. Структурирование текста**

1.1.1. Длина строки Нужно стараться избегать строк длинной более 120 символов. Если строка превышает этот размер, то нужно использовать правила переноса строки.

1.1.2. Правила переноса строки. Если длина строки превышает 120 символов, то необходимо пользоваться следующими правилами переноса:

- переносить можно после запятой или перед оператором;
- переносимая строка должна быть сдвинута относительно верхней на один символ табуляции;
- переносы должны быть в стиле UNIX.

#### **Пример 1:** для строки

```
$arAuthResult = $USER->ChangePassword($USER_LOGIN, $USER_CHECKWORD,  
$USER_PASSWORD, $USER_CONFIRM_PASSWORD, $USER_LID);
```

допустимым будет следующий вариант переноса

```
$arAuthResult = $USER->ChangePassword($USER_LOGIN, $USER_CHECKWORD,  
$USER_PASSWORD, $USER_CONFIRM_PASSWORD, $USER_LID);
```

#### **Пример 2:** для строки



```
if(COption::GetOptionString("main", "new_user_registration", "N")=="Y" &&
$_SERVER['REQUEST_METHOD']=='POST' &&$TYPE=="REGISTRATION" &&
(!defined("ADMIN_SECTION") || ADMIN_SECTION!=true))
```

допустимым будет следующий вариант переноса

```
if (COption::GetOptionString("main", "new_user_registration", "N") == "Y"
&& $_SERVER['REQUEST_METHOD'] == 'POST' && $TYPE == "REGISTRATION"
&& (!defined("ADMIN_SECTION") || ADMIN_SECTION != true))
```

1.1.3. Пробелы и табуляция Для форматирования отступов в коде нужно использовать табуляцию. Использование пробелов запрещено. Причины:

- в случае использования табуляции каждый может настроить в своем редакторе желаемый отступ;
- используется один символ вместо нескольких;
- в случае смещивания пробелов и табуляции текст будет **прыгать**, разрушая форматирование.

1.1.4. Форматирование подчиненности Подчиненный код должен быть сдвинут от главного ровно на один символ табуляции. Подчиненный код не может находиться на той же строке, что и главный.

**Пример.** Не правильно писать так:

```
if ($a == 0) {$a = 10;$b = 1;}
```

Правильно писать так

```
if ($a == 0)
{
    $a = 10;
    $b = 1;
}
```

## 1.2. Инструкции, выражения

### 1.2.1. Выражения

Желательно, чтобы в каждой строчке присутствовало только одно выражение.

**Пример.** Не правильно писать так:



```
$a = $b; $b = $c; $c = $a;
```

Правильно писать так

```
$a = $b;  
$b = $c;  
$c = $a;
```

1.2.2. Инструкции if, else, while и т.п. Допустимы два вида написания инструкций:

- если тела всех частей инструкции состоят не более чем из одного выражения, то инструкция может записываться в виде:

```
if (условие)  
    действие1;  
else  
    действие2;
```

- если тело хотя бы одной из частей состоит более чем из одного выражения, то инструкция должна записываться в виде:

```
if (условие)  
{  
    действие1;  
}  
else  
{  
    действие2;  
    действие3;  
}
```

При написании инструкций должно строго применяться правило **1.1.4 Форматирование подчиненности**: тело инструкции должно быть сдвинуто на один символ табуляции вправо от самой инструкции. Фигурные скобки должны находиться на отдельных строках и должны быть на одном уровне с инструкцией.

**Пример.** Не правильно писать так:

```
if ($a == 0) $a = 10;  
else{  
    $a = 5;  
    $b = 10;}
```



Правильно писать так:

```
if ($a == 0)
{
    $a = 10;
}
else
{
    $a = 5;
    $b = 10;
}
```

1.2.3. Сложные инструкции. Сложные инструкции следует разбивать по строкам в соответствии с правилами пункта 1.2.2. **Например**,

```
if(COption::GetOptionString("main", "new_user_registration", "N")=="Y" &&
$_SERVER['REQUEST_METHOD']=='POST' &&
$TYPE=="REGISTRATION" && (!defined("ADMIN_SECTION")) // 
ADMIN_SECTION!=true))
```

Можно записать как:

```
if (COption::GetOptionString("main", "new_user_registration", "N") == "Y"
&& $_SERVER['REQUEST_METHOD'] == 'POST' && $TYPE == "REGISTRATION"
&& (!defined("ADMIN_SECTION")) || ADMIN_SECTION != true))
{
```

Очень сложные инструкции рекомендуется разбивать на несколько более простых. Например,

```
if(!!(defined("STATISTIC_ONLY") && STATISTIC_ONLY && substr($APPLICATION->GetCurPage(), 0,
strlen(BX_ROOT."/admin/"))!=BX_ROOT."/admin/")
COption::GetOptionString("main", "include_charset", "Y")=="Y"
&& strlen(LANG_CHARSET)>0)
```

Можно записать так:

```
$publicStatisticOnly = False;
if (defined("STATISTIC_ONLY"))
```



```
&& STATISTIC_ONLY  
&& substr($APPLICATION->GetCurPage(), 0, strlen(BX_ROOT."/admin/")) !=  
BX_ROOT."/admin/")  
{  
$publicStatisticOnly = True;  
}  
if (!$publicStatisticOnly && strlen(LANG_CHARSET) > 0  
&& COption::GetOptionString("main", "include_charset", "Y") == "Y")  
{  
}
```

или так:

```
if (!defined("STATISTIC_ONLY") || !STATISTIC_ONLY  
|| substr($APPLICATION->GetCurPage(), 0, strlen(BX_ROOT."/admin/")) ==  
BX_ROOT."/admin/")  
{  
if (strlen(LANG_CHARSET) > 0 && COption::GetOptionString("main", "include_charset",  
"Y") == "Y")  
{  
}  
}
```

1.2.4. Форматирование массивов Массивы, которые записываются в несколько строк, следует форматировать следующим образом:

```
$arFilter = array(  
    "key1" => "value1",  
    "key2" => array(  
        "key21" => "value21",  
        "key22" => "value22",  
    ),  
);
```

### 1.3. Пустые строки и пробелы

1.3.1. Пустые строки Пустые строки помогают разбивать код приложения на логические сегменты. Несколько строками могут отделяться секции в исходном



файле. Одной пустой строкой отделяются друг от друга методы и логические секции внутри метода для более удобного чтения.

Перед логической секцией рекомендуется вставить комментарий, в котором будет указано назначение этой секции (см. пункт 3).

1.3.2. Пробелы После запятой должен быть пробел. После точки с запятой, если она не последняя в строке (например в инструкции **for**), должен быть пробел. Перед запятой или точкой с запятой пробелы не ставятся. Все операторы должны быть отделены пробелом от операндов с обеих сторон. Замена пробела символом табуляции не допускается.

Пояснения:

- Один пробел используется в объявлении методов после запятой, но не перед скобками: `TestMethod($a, $b, $c);`

Примеры неправильного использования:

- `TestMethod($a,$b,$c);`
- `TestMethod( $a, $b, $c );`
- Так же одиночный пробел используют для выделения операторов: `$a = $b * $c / $d;`

Пример неправильного использования: `$a=$b*$c/$d;`

- Тоже пробелы используются при формировании циклов: `for ($i = 0; $i < 10; $i++).`

Пример неправильного использования: `for($i=0;$i<10;$i++)`

- Табличное форматирование с помощью символов табуляции не должно использоваться.

Пример неправильного форматирования:

```
$arArray = array(  
    "key1" => "value1",  
    "key2" => "value2",  
);
```

 **Примечание:** присутствие или отсутствие пробела после **if** правилами не регламентируется.



**1.4. Прочее.** В сложных выражениях рекомендуется группировать операции с помощью скобок вне зависимости от того, требует это приоритет операций или нет.  
**Пример:** `$r = $a + ($b * $c);`

## 2. Соглашение об именовании

### 2.1. Общие понятия

Не используйте подчеркивание для отделения слов внутри идентификаторов, это удлиняет идентификаторы и затрудняет чтение.

Старайтесь давать переменным, методам и пр. "говорящие" названия. Предпочтительно использовать имена, которые ясно и четко описывают предназначение и/или смысл сущности.

Старайтесь делать имена идентификаторов как можно короче (но не в ущерб читабельности).

### 2.2. Именование переменных

Первое логическое слово должно начинаться с маленькой буквы, остальные логические слова - с большой (стиль Кэмел). Например: `$testCounter`, `$userPassword`.

### 2.3. Именование методов и функций

Каждое логическое слово должно начинаться с заглавной буквы (стиль Паскаль). Например: `CountVariable`, `ChangeUserPassword`.

### 2.4. Префиксы переменных

PHP - не особо типизированный язык и в нем различаются по смыслу только три группы типов: скалярные, массивы и объекты.

Массивы следует именовать с префиксом `аг`, при этом следующее логическое слово в названии начинается с большой буквы. Например, `$arResult`, `$arModifiedUsers`.

Объекты следует именовать с префиксом "`об`", при этом следующее логическое слово в названии начинается с большой буквы. Например, `$obElement`, `$obUser`.

Объект класса [CDBResult](#) следует начинать с префикса `db`, при этом следующее логическое слово в названии начинается с большой буквы. Например, `$dbResult`.

Скалярные типы следует начинать с префиксов только в том случае, если точно известно, что они имеют заданный тип. Например в коде:

```
$userID = $_REQUEST["var"];
$(userID = IntVal($userID);
```



переменная идет без префикса, так как ее тип меняется по ходу выполнения программы. А в коде:

```
$bFlag = ($aaa > 0) ? True : False;
```

переменная идет с префиксом, так как ее тип в общем известен и не меняется.

## 2.5. Именование классов

Имя класса должно начинаться с буквы С. Если класс принадлежит модулю, то дальше должно идти "фирменное" название модуля. Каждое логическое слово должно начинаться с заглавной буквы. Пример: CIBlockElement, CIBlockType, CSaleAffiliate.

Если класс различается для разных СУБД и соответственно имеет базовый класс с общими для всех СУБД методами, то этот базовый класс должен в своем имени после символа С содержать символы All. Пример: CSaleAffiliate.

## 2.6. Доступность членов-переменных и методов класса

Так как у нас нет других методов организации видимости и доступности членов-переменных и методов классов, то следует применять следующие правила:

- члены-переменные и методы, которые являются приватными и к которым не может обращаться никто, кроме самого модуля (т.е. ни публичная часть, ни другие модули), должны начинаться с двух символов подчеркивания. Например, \_\_CheckEmail, \_\_arData. Эти методы не описываются в документации и могут быть изменены без обеспечения совместимости;
- члены-переменные и методы, которые являются внутренними и к которым могут обращаться только модули продукта (т.е. публичная часть не может), должны начинаться с одного символа подчеркивания. Например, \_CheckEmail, \_arData. Эти методы не описываются в публичной документации и могут быть изменены без обеспечения совместимости;
- остальные методы и члены-переменные считаются публичными, должны быть описаны в документации и не могут быть изменены без обеспечения совместимости.

## 2.7. Именование констант

Константы должны писаться большими буквами и иметь префикс BX\_. Например, BX\_ROOT, BX\_FILE\_PERMISSIONS.

## 3. Комментарии



Для пояснения назначения класса или метода необходимо размещать комментарий перед объявлением этого класса или метода. Следует добавлять комментарий о назначении классов и методов к каждому публичному классу или методу. Перед логическими секциями кода желательно добавлять комментарии о том, что данная секция будет делать. Комментарии должны быть только на английском языке.

Необходимо избегать очевидных комментариев типа:

```
$i = $i + 1; // добавить к i единицу
```

## 4. Идиомы программирования

### 4.1. Общее понятие

В любом языке программирования существуют так называемые идиомы, то есть повсеместно применяемые способы использования тех или иных конструкций. Например, в языке PHP к таким идиомам можно отнести форму записи цикла по элементам массива:

```
for ($i = 0, $cnt = count($arArray); $i < $cnt; $i++)  
{  
}
```

или

```
foreach ($arArray as $key => $value)  
{  
}
```

Использование идиом позволяет читающему пропускать очевидные фрагменты кода и сосредоточиться на содержательных вещах, а также находить в коде нужные фрагменты по характерным (идиоматическим) конструкциям.

Необходимо стараться использовать общепринятые конструкции, а не изобретать свои собственные. Пример: код

```
reset($arHashLink);  
while(list($hash, $arData)=each($arHashLink))  
{  
}
```

лучше переписывать в таком виде

```
foreach ($arHashLink as $hash => $arData)
```

```
{  
}
```

## 4.2. Примеры идиом

Идиома оператора "?"

```
$res = ($bTrue? "True" : "False");
```

## 5. SQL запросы

Каждая операция **SELECT, FROM, WHERE, ORDER BY, GROUP BY, HAVING** должна начинаться с новой строки.

Правило переноса длинной строки такое же как в PHP - новая строка с табуляции.

## Безопасность

Вопросы безопасности при программировании в **Bitrix Framework**.

### Санитайзер

**Санитайзер** - инструмент, анализирующий введённый пользователем html код. Основная задача санитайзера - предотвратить внедрение/вывод на экран потенциально опасного кода в HTML.

Санитайзер удобно использовать там, где пользователь вводит произвольный html. Например, в визуальном редакторе или при копировании текста из **MS Word**. Кроме функций контроля введённого кода санитайзер частично отслеживает валидность вёрстки, в частности закрывает незакрытые теги.

### Как фильтровать текст

Если необходимо отфильтровать текст (содержащий HTML - тэги) введенный пользователем от нежелательных тэгов HTML с помощью санитайзера, то можно это сделать так:

```
$Sanitizer = new CBXSanitizer;  
$Sanitizer->AddTags( array (  
    'a' => array('href','id','style','alt'...),  
    'br' => array(),
```



```
.... );  
$pureHtml = $Sanitizer->SanitizeHtml($html);
```

Санитайзер отфильтрует все тэги и атрибуты, которые не содержатся в "белом" списке, сформированном функцией [AddTags\(\)](#).

В санитайзер включены 3 преднастроенных уровня фильтрации:

SECURE\_LEVEL\_HIGH (высокий уровень) включает следующий список:

```
$arTags = array(  
    'b'      => array(),  
    'br'     => array(),  
    'big'    => array(),  
    'code'   => array(),  
    'del'    => array(),  
    'dt'     => array(),  
    'dd'     => array(),  
    'font'   => array(),  
    'h1'     => array(),  
    'h2'     => array(),  
    'h3'     => array(),  
    'h4'     => array(),  
    'h5'     => array(),  
    'h6'     => array(),  
    'hr'     => array(),  
    'i'      => array(),  
    'ins'   => array(),  
    'li'     => array(),  
    'ol'     => array(),  
    'p'      => array(),  
    'small'  => array(),  
    's'      => array(),  
    'sub'    => array(),  
    'sup'    => array(),  
    'strong' => array(),  
    'pre'    => array(),  
    'u'      => array(),  
    'ul'    => array()
```



);

SECURE\_LEVEL\_MIDDLE (средний уровень) включает в себя:

```
$arTags = array(
    'a'    => array('href', 'title', 'name', 'alt'),
    'b'    => array(),
    'br'   => array(),
    'big'  => array(),
    'code' => array(),
    'caption' => array(),
    'del'   => array('title'),
    'dt'    => array(),
    'dd'    => array(),
    'font'  => array('color', 'size'),
    'color' => array(),
    'h1'    => array(),
    'h2'    => array(),
    'h3'    => array(),
    'h4'    => array(),
    'h5'    => array(),
    'h6'    => array(),
    'hr'    => array(),
    'i'     => array(),
    'img'   => array('src', 'alt', 'height', 'width', 'title'),
    'ins'   => array('title'),
    'li'    => array(),
    'ol'    => array(),
    'p'     => array(),
    'pre'   => array(),
    's'     => array(),
    'small' => array(),
    'strong' => array(),
    'sub'   => array(),
    'sup'   => array(),
    'table' => array('border', 'width'),
    'tbody' => array('align', 'valign'),
    'td'    => array('width', 'height', 'align', 'valign'),
```



```
'tfoot' => array('align','valign'),
'th'    => array('width','height'),
'thead' => array('align','valign'),
'tr'    => array('align','valign'),
'u'     => array(),
'ul'    => array()
);
```

SECURE\_LEVEL\_LOW (низкий уровень) включает в себя:

```
$arTags = array(
    'a'    => array('href', 'title', 'name', 'style', 'id', 'class', 'shape', 'coords', 'alt', 'target'),
    'b'    => array('style', 'id', 'class'),
    'br'   => array('style', 'id', 'class'),
    'big'  => array('style', 'id', 'class'),
    'caption' => array('style', 'id', 'class'),
    'code'  => array('style', 'id', 'class'),
    'del'   => array('title', 'style', 'id', 'class'),
    'div'   => array('title', 'style', 'id', 'class', 'align'),
    'dt'    => array('style', 'id', 'class'),
    'dd'    => array('style', 'id', 'class'),
    'font'  => array('color', 'size', 'face', 'style', 'id', 'class'),
    'h1'    => array('style', 'id', 'class', 'align'),
    'h2'    => array('style', 'id', 'class', 'align'),
    'h3'    => array('style', 'id', 'class', 'align'),
    'h4'    => array('style', 'id', 'class', 'align'),
    'h5'    => array('style', 'id', 'class', 'align'),
    'h6'    => array('style', 'id', 'class', 'align'),
    'hr'    => array('style', 'id', 'class'),
    'i'     => array('style', 'id', 'class'),
    'img'   => array('src', 'alt', 'height', 'width', 'title'),
    'ins'   => array('title', 'style', 'id', 'class'),
    'li'    => array('style', 'id', 'class'),
    'map'   => array('shape', 'coords', 'href', 'alt', 'title', 'style', 'id', 'class', 'name'),
    'ol'    => array('style', 'id', 'class'),
    'p'     => array('style', 'id', 'class', 'align'),
    'pre'   => array('style', 'id', 'class'),
    's'     => array('style', 'id', 'class'),
```



```
'small' => array('style','id','class'),
'strong' => array('style','id','class'),
'span' => array('title','style','id','class','align'),
'sub' => array('style','id','class'),
'sup' => array('style','id','class'),
'table' => array('border','width','style','id','class','cellspacing','cellpadding'),
'tbody' => array('align','valign','style','id','class'),
'td' => array('width','height','style','id','class','align','valign','colspan','rowspan'),
'tfoot' => array('align','valign','style','id','class','align','valign'),
'th' => array('width','height','style','id','class','colspan','rowspan'),
'thead' => array('align','valign','style','id','class'),
'tr' => array('align','valign','style','id','class'),
'u' => array('style','id','class'),
'ul' => array('style','id','class')
);
```

Воспользоваться санитайзером с одним из преднастроенных уровней можно так:

```
$Sanitizer = new CBXSanitizer;
$Sanitizer->SetLevel(CBXSanitizer::SECURE_LEVEL_MIDDLE);
$pureHtml = $Sanitizer->SanitizeHtml($html);
```

Для работы с санитайзером доступны функции класса [CBXSanitizer](#).

## Защита от фреймов

На странице **Защита от фреймов** ([Настройки > Проактивная защита > Защита от фреймов](#)) можно включить/отключить ограничение работы во фрейме.

Запрет на использование кросс-доменных фреймов ссылающихся на страницы ресурса задается установкой заголовка **X-Frame-Options** в значение **SAMEORIGIN**.

### **X-Frame-Options**

Данный заголовок указывает браузеру, можно ли загружать страницы сайта через `<frame>/<iframe>`.

Значение **DENY** запретит загрузку через фреймы, значение **SAMEORIGIN** разрешит загрузку через фреймы, но только если и фрейм, и страница, его загружающая, находятся на одном домене (*Same Origin Policy*).

Основная функция данной защиты — предотвращение кликджекинга; в качестве дополнительного бонуса это позволит предотвратить атаку, описанную Ben Schmidt.

При необходимости, вы можете добавить свою страницу в исключения путем определения константы **B\_SECURITY\_FRAME** в значение *false*, до подключения ядра.

## Примеры, хитрости и советы

### Цитатник веб-разработчиков.

**Антон Долганин:** Советую даже опытным спецам посмотреть как сделаны (и которые будут сделаны) решения от самого Битрикс (магазин, инфопортал, к примеру). Встречаются довольно хитрые решения, новый взгляд на обычные компоненты.

В этой главе собраны некоторые советы, примеры, которые могут помочь разработчикам в освоении и работе с **Bitrix Framework**.

## Командная PHP-строка

Иногда бывают ситуации, когда нужно быстро выполнить некоторый код, вызывающий функции API **Bitrix Framework** без создания новых страниц на сайте для этого. В этом случае поможет удобный и простой инструмент — командная PHP-строка, позволяющий запускать произвольный код на PHP с вызовами функций.

Инструмент расположен в административной части сайта по следующему пути: [Настройки > Инструменты > Командная PHP-строка](#) и имеет адрес [/bitrix/admin/php\\_command\\_line.php](/bitrix/admin/php_command_line.php).

Вот как выглядит результат выполнения кода, использующего функции класса [CUser](#) Главного модуля:



Командная PHP-строка

Рабочий стол > Настройки > Инструменты > Командная PHP-строка

**PHP-строка**

Произвольный PHP-скрипт для выполнения на сервере

```
1 $rs = CUser::Getlist();
2 while($f = $rs->Fetch())
3     echo $f['ID'] . '' . $f['LOGIN'] . '<br>';
```

3 строка: 3 символ: 43 Всего строк: 3 подсветка синтаксиса

Выполнить Очистить

**Результат выполнения команды**

```
2telega
1admin
```

**Примечание:** Аналогичный [инструмент](#) есть и для работы с БД.

## Мелкая оптимизация при разработке

Цитатник веб-разработчиков.

**Востриков Сергей:** Кэширование компонентов не решает задачу производительности как таковую, а помогает при масштабировании, увеличении нагрузки и т.д. То есть, если в коде есть мелкие косяки - лишние запросы, которых можно было бы при разработке избежать, то кэширование поверх этих запросов проблему производительности решает не всегда.

Один из партнеров "1С-Битрикс" разработал небольшой бесплатный модуль для облегчения текущей работы над проектами.

 **Примечание.** Поскольку модуль не является разработкой компании "1С-Битрикс", все претензии по его работе предъявляйте разработчику.

## Описание

Мелкие запросы, на первый взгляд, не сильно влияют (в относительной оценке) на работу сайта, если при этом на том же сайте есть какой-нибудь явно тяжелый кусок кода, связанный, например, с фильтрацией товаров, который по определению кэшированию не поддается.

Достаточно часто в проектах многие используют вызов [CIBlockElement::GetById](#). Простой, удобный метод когда надо вытащить какое-то поле для элемента инфоблока. Но этот метод тянет все поля и все свойства элемента. В случае инфоблока с большим количеством свойств и большого числа посетителей на сайте этот простой запрос превращается в небольшую дырку в производительности. А если таких дырок несколько десятков в различных **result\_modifier** у разных компонентов? Оказывается, что в совокупности, несмотря на кэширование, эти вещи создают пиковые нагрузки в момент обновления кэша.

Если уж надо получить название элемента по ID, то лучше воспользоваться [GetList](#) с указанием конкретного вытаскиваемого поля элемента.

 **Примечание:** *GetById* внутри вызывает *GetList*. Разница только в том, что *GetById* в любом случае тянет все поля и свойства, а в *GetList* этот перечень можно контролировать.

Соответственно, разница в скорости выполнения будет сильно зависеть от того, что именно надо вытащить, а потому сравнивать надо не просто два метода, а конкретную бизнес-логику конкретного сайта.

Единственное преимущество **GetById** перед **GetList** с точки зрения разработчика состоит в том, что GetById можно просто вызвать и все. Прямое использование GetList требует определенной работы с кодом (создание массива параметров).

В базовом модуле [сп. codenails](#) (который содержит, кстати говоря, ряд некоторых мелких полезняшек) есть метод **IBElement.GetById**, которому передается идентификатор элемента и перечень полей, которые нам надо получить в результате. Фактически это - обертка специально для этого случая, которая получилась даже удобнее, чем базовый GetById, поскольку сразу возвращает массив значений и не надо вызывать никакой **GetNext**.

Простое сравнение со включенным монитором запроса показывает значительную разницу в выполнении стандартного метода **GetById** и метода **IBElement.GetById**:



```
CModule::IncludeModule("cn_codenails");
CModule::IncludeModule("iblock");

$res = CIBlockElement::GetByID(329);
if ($ar_res = $res->GetNext())
    CNCodenails::Debug($ar_res);

CNCodenails::Debug(CNCodenailsBase::IBElementGetById(329, array("NAME",
"PROPERTY_MODERATORS")));
В первом случае мы получается в отладчике многострочный запрос, а во-втором -
простой и короткий select.
```

### Пример использования

Обычная ситуация - есть инфоблок, элементы которого через свойство ссылаются на элементы второго. Например, товары ссылаются на справочник брендов. Выводится список товаров, но при этом нам нужно, чтобы в этом списке также отображались и названия (логотипы) брендов.

Как принято решать такую задачу? Нужный компонент для списка товаров размещается на странице, проверяется, как товары отобразились. Потом копируется стандартный шаблон в свое пространство имен, чтобы его подогнать под дизайн сайта. Затем пишется **result\_modifier.php** для этого шаблона, в котором делается примерно такой цикл:

```
foreach ($arResult["ITEMS"] as $key => $arItem):
    // А не хапнуть ли нам данные о брендах? Конечно, хапнуть!

    // Мы модные пацаны, поэтому вместо стандартного GetByID будем
    // использовать реальный CNCodenails::IBElementGetById !

    $arResult["ITEMS"][$key]["BRAND"] =
    CNCodenails::IBElementGetById($arItem["PROPERTIES"]["BRAND"]["VALUE"], array("NAME",
"PROPERTY_LOGO")); endforeach;
```

За счет **CNCodenails::IBElementGetById** это уже лучше, чем могло бы быть (и чем было у нас до того, как мы осознали проблему с GetByID), но это не самое лучшее в некоторых ситуациях. Код неоптимальный.

А более оптимальный код в данном случае означал бы сведение цикла из запросов к одному запросу при помощи метода **GetList**, который умеет вытаскивать элементы по массиву значений, в частности, по массиву идентификаторов. Код, соответственно, будет выглядеть несколько иначе:



```
$arBrandIDs = array();  
  
foreach ($arResult["ITEMS"] as $arItem)  
    $arBrandIDs[] = $arItem["PROPERTIES"]["BRAND"]["VALUE"];  
  
$arBrandSelect = Array("ID", "NAME", "PROPERTY_LOGO");  
$arBrandFilter = Array("ID" => $arBrandIDs);  
$obBrandItems = CIBlockElement::GetList(Array(), $arBrandFilter, false, false, $arBrandSelect);  
while($arBrandItem = $obBrandItems->GetNext())  
{  
    $arResult["ITEMS"]["BRANDS"][$arBrandItem["ID"]] = $arBrandItem;  
}
```

Если посмотреть в режиме **Отладки** на получившиеся запросы, то мы увидим, что запрос и правда будет один и выглядит наподобие следующего:

```
SELECT BE.ID as ID,BE.NAME as NAME, FPV0.VALUE as PROPERTY_ISBN_VALUE, FPV0.ID as  
PROPERTY_ISBN_VALUE_ID  
FROM  
b_iblock B  
INNER JOIN b_lang L ON B.LID=L.LID  
INNER JOIN b_iblock_element BE ON BE.IBLOCK_ID = B.ID  
LEFT JOIN b_iblock_property FP0 ON FP0.IBLOCK_ID = B.ID AND FP0.CODE='BRAND'  
LEFT JOIN b_iblock_element_property FPV0 ON FPV0.IBLOCK_PROPERTY_ID = FP0.ID AND  
FPV0.IBLOCK_ELEMENT_ID = BE.ID  
WHERE  
1=1 AND ( ((BE.ID IN ('53', '54', '57')))) ) AND (((BE.WF_STATUS_ID=1 AND  
BE.WF_PARENT_ELEMENT_ID IS NULL)))
```

Вывод:

- Если нужны поля/свойства одного элемента, лучше использовать **CNCodetails::IBElementGetById** вместо **CIBlockElement::GetByID**.
- Если нужны поля/свойства нескольких элементов по известным идентификаторам, лучше использовать **GetList** и массив в качестве значений для поля "ID".



## Сортировка в компонентах news.list и catalog.section

Для выполнения сортировки в компоненте **news.list** или **catalog.section** компоненту необходимо передать параметры **ELEMENT\_SORT\_FIELD** и **ELEMENT\_SORT\_ORDER**.

Сортировку можно произвести по стандартным полям, для чего можно воспользоваться приведенным ниже списком:

- **id** - ID элемента;
- **sort** - индекс сортировки;
- **timestamp\_x** - дата изменения;
- **name** - название;
- **active\_from** или **date\_active\_from** - начало периода действия элемента;
- **active\_to** или **date\_active\_to** - окончание периода действия элемента;
- **status** - код статуса элемента в документообороте;
- **code** - мнемонический код элемента;
- **iblock\_id** - числовой код информационного блока;
- **modified\_by** - код последнего изменившего пользователя;
- **active** - признак активности элемента;
- **show\_counter** - количество показов элемента (учитывается функцией **CIBlockElement::CounterInc**);
- **show\_counter\_start** - время первого показа элемента (учитывается функцией **CIBlockElement::CounterInc**);
- **shows** - усредненное количество показов (количество показов / продолжительность показа);
- **rand** - случайный порядок;
- **xml\_id** или **external\_id** - внешний код;
- **tags** - теги;
- **created** - время создания;
- **created\_date** - дата создания без учета времени;
- **cnt** - количество элементов (только при заданной группировке).

 **Примечание:** Поля **active\_from** и **active\_to** - устаревшие.

Также сортировать можно по созданным вами свойствам элемента информационного блока:



- **property\_<PROPERTY\_CODE>** - по значению свойства с числовым или мнемоническим кодом **PROPERTY\_CODE** (например, PROPERTY\_123 или PROPERTY\_NEWS\_SOURCE).
- **propertysort\_<PROPERTY\_CODE>** - по индексу сортировки варианта значения свойства. Только для свойств типа **Список**.
- **catalog\_<CATALOG\_FIELD>\_<PRICE\_TYPE>** - по полю **CATALOG\_FIELD** (может быть PRICE - цена или CURRENCY - валюта) из цены с типом **PRICE\_TYPE** (например, catalog\_PRICE\_1 или CATALOG\_CURRENCY\_3). Сортировка должна иметь формат: **CATALOG\_(PRICE или CURRENCY)\_ID-типа-цены**.
- **catalog\_QUANTITY** - сортировка по количеству.
- **PROPERTY\_<PROPERTY\_CODE>.<FIELD>** - по значению поля элемента указанного в качестве привязки. **PROPERTY\_CODE** - мнемонический или символьный код свойства типа **привязка к элементам**. **FIELD** может принимать значения:
  - ID
  - TIMESTAMP\_X
  - MODIFIED\_BY
  - CREATED
  - CREATED\_DATE
  - CREATED\_BY
  - IBLOCK\_ID
  - ACTIVE
  - ACTIVE\_FROM
  - ACTIVE\_TO
  - SORT
  - NAME
  - SHOW\_COUNTER
  - SHOW\_COUNTER\_START
  - CODE
  - TAGS
  - XML\_ID
  - STATUS
- **PROPERTY\_<PROPERTY\_CODE>.PROPERTY\_<PROPERTY\_CODE2>** - по значению свойства элемента указанного в качестве привязки. **PROPERTY\_CODE** - мнемонический или символьный код свойства типа привязки к элементам. **PROPERTY\_CODE2** - код свойства связанных элементов.
- **HAS\_PREVIEW\_PICTURE** и **HAS\_DETAIL\_PICTURE** - сортировка по наличию и отсутствию картинок.



**⚠ Примечание:** Свойства **catalog** доступны только при наличии модуля **Торговый каталог**.

Тип сортировки указывается в соответствии со списком:

- **asc** - по возрастанию;
- **nulls,asc** - по возрастанию с пустыми значениями в начале выборки;
- **asc,nulls** - по возрастанию с пустыми значениями в конце выборки;
- **desc** - по убыванию;
- **nulls,desc** - по убыванию с пустыми значениями в начале выборки;
- **desc,nulls** - по убыванию с пустыми значениями в конце выборки.

Самый простой способ передать новые параметры для сортировки в компонент - это использовать **\$\_GET** запрос и передать соответствующие переменные.

Также можно воспользоваться **\$\_SESSION** и записать переменные в массив переменных сессии. Предположим нам необходимо сделать ссылки или кнопки(название, цена, лидер продаж, дата поступления) для сортировки товаров в разделе каталога(используем комплексный компонент **catalog**). После того как мы скопировали шаблон, необходимо открыть файл **section.php** и внести в него следующие модификации перед подключением компонента **bitrix:catalog.section**:

```
<?if ($_GET["sort"] == "name" ||  
      $_GET["sort"] == "catalog_PRICE_3" ||  
      $_GET["sort"] == "property_PRODUCT_TYPE" ||  
      $_GET["sort"] == "timestamp_x"){  
    $arParams["ELEMENT_SORT_FIELD"] = $_GET["sort"];  
    $arParams["ELEMENT_SORT_ORDER"] = $_GET["method"];  
}else{}?>
```

Этот код необходим для изменения параметров сортировки в компоненте. Далее откроем файл **template.php** компонента **catalog.section** и добавим ссылки управления сортировками:

```
<p class="sort">Сортировка:  
    <a <?if ($_GET["sort"] == "name")?:?> class="active" <?endif;?>  
    href=<?= $arResult["SECTION_PAGE_URL"]?>?sort=name&method=asc">название</a>  
    <a <?if ($_GET["sort"] == "catalog_PRICE_3")?:?> class="active" <?endif;?>  
    href=<?= $arResult["SECTION_PAGE_URL"]?>?sort=catalog_PRICE_3&method=asc">цена</a>  
    <a <?if ($_GET["sort"] == "property_PRODUCT_TYPE")?:?> class="active" <?endif;?>  
    href=<?= $arResult["SECTION_PAGE_URL"]?>?sort=property_PRODUCT_TYPE&method=desc">  
    лидер продаж</a>
```

```
<a <?if ($_GET["sort"] == "timestamp_x"):> class="active" <?endif;?>
href=<?= $arResult["SECTION_PAGE_URL"]?>?sort=timestamp_x&method=desc">дата
поступления<</a>
</p>
```

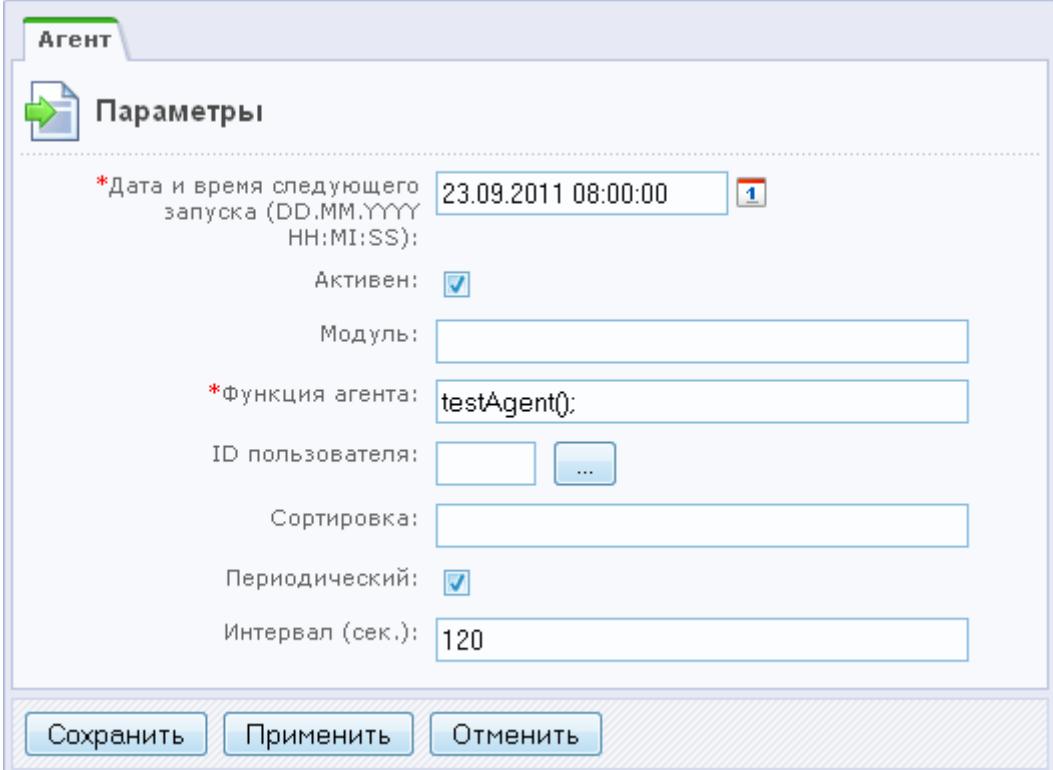
Данную сортировку можно выполнить без перезагрузки страницы с использованием **jQuery**.

## Использование агентов

### Примеры агентов

Если необходимо динамически добавлять агентов, то используйте [API](#) агентов. Если вам просто нужно прикрутить один или два агента, то это проще сделать вручную.

Агента создаем на странице *Настройки > Инструменты > Агенты* по команде **Добавить агента** на контекстной панели:



Агент

Параметры

\*Дата и время следующего запуска (DD.MM.YYYY HH:MI:SS): 23.09.2011 08:00:00

Активен:

Модуль:

\*Функция агента: testAgent()

ID пользователя:

Сортировка:

Периодический:

Интервал (сек.): 120

Сохранить Применить Отменить

О параметрах, значение которых может быть неясно из названия:

- **дата последнего запуска** - если агент периодичный, то будет выведено время последнего запуска при редактировании агента;

- **дата и время следующего запуска** – время старта работы агента, если он не переодический то выполнится 1 раз в это время;
- **модуль** - этот модуль будет автоматически подключаться, а именно будет подключаться файл /bitrix/modules/*ID* модуля/include.php, в этом случае необходимо убедиться, что функция-агент будет доступна после подключения этого файла;
- **функция агента** - это основное поле, у нас функция называется **testAgent()**;
- **ID пользователя** – это фильтр выполнения на хите для определенного пользователя;

Сама функция будет выглядеть так:

```
function testAgent()
{
    mail('mail@gmail.com', 'Агент', 'Агент');
    return "testAgent();";
}
```

Функцию добавить в файл /bitrix/php\_interface/init.php.

Если письмо пришло, то агент работает и можно писать свой функционал.

### Простые примеры агентов

```
<?
// добавим агент модуля "Статистика"
CAgent::AddAgent(
    "CStatistic::CleanUpStatistics_2()", // имя функции
    "statistic", // идентификатор модуля
    "N", // агент не критичен к кол-ву запусков
    86400, // интервал запуска - 1 сутки
    "07.04.2005 20:03:26", // дата первой проверки на запуск
    "Y", // агент активен
    "07.04.2005 20:03:26", // дата первого запуска
    30);
?>
```

```
<?
// добавим агент модуля "Техподдержка"
```

**CAgent::AddAgent(**

```
"CTicket::AutoClose();", // имя функции
"support",           // идентификатор модуля
"N",                 // агент не критичен к кол-ву запусков
86400,               // интервал запуска - 1 сутки
"",                  // дата первой проверки - текущее
"Y",                 // агент активен
"",                  // дата первого запуска - текущее
30);
?>
```

&lt;?

// добавим произвольный агент не принадлежащий ни одному модулю

**CAgent::AddAgent("My\_Agent\_Function());**

?&gt;

&lt;?

// файл /bitrix/php\_interface/init.php

```
function My_Agent_Function()
```

{

// выполняем какие-либо действия

```
return "My_Agent_Function();";
```

}

?&gt;

&lt;?

// добавим произвольный агент принадлежащий модулю

// с идентификатором my\_module

**CAgent::AddAgent(**

```
"CMyModule::Agent007(1)",
"my_module",
"Y",
86400);
```



```
?>

<?

// данный агент будет запущен ровно 7 раз с периодичностью раз в сутки,
// после чего будет удален из таблицы агентов.

Class CMyModule
{
    function Agent007($cnt=1)
    {
        echo "Hello!";
        if($cnt>=7)
            return "";
        return "CMyModule::Agent007(\".$( $cnt+1 ).\")";
    }
}
?>
```

### Практичный пример: Обновление курса валют на сайте

Запуск данного агента рекомендуется повесить на [cron](#).

```
// Обновление к

function AgentGetCurrencyRate()
{
    global $DB;

    // подключаем модуль "валют"
    if(!CModule::IncludeModule('currency'))
        return "AgentGetCurrencyRate();";

    $arCurList = array('USD', 'EUR');
    $bWarning = False;
    $rateDay = GetTime(time(), "SHORT", LANGUAGE_ID);
    $QUERY_STR = "date_req=". $DB->FormatDate($rateDay, CLang::GetDateFormat("SHORT",
SITE_ID), "D.M.Y");
    $strQueryText = QueryGetData("www.cbr.ru", 80, "/scripts/XML_daily.asp", $QUERY_STR,
$errno, $errstr);
```

```
// данная строка нужна только если у вас сайт в кодировке utf8
$strQueryText = iconv('windows-1251', 'utf-8', $strQueryText);

if (strlen($strQueryText) <= 0)
$bWarning = True;

if (!$bWarning)
{

require_once($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/classes/general/xml.php");
}

$strQueryText = eregi_replace("]{1,>", "", $strQueryText);
$strQueryText = eregi_replace("<.". "?XML[^>]{1,}\?". ">", "", $strQueryText);
$objXML = new CDataXML();
$objXML->LoadString($strQueryText);
$arData = $objXML->GetArray();
$arFields = array();
$arCurRate["CURRENCY_CBRF"] = array();

if (is_array($arData) && count($arData["ValCurs"]["#"]["Valute"])>0)
{
    for ($j1 = 0; $j1 < $arData["ValCurs"]["#"]["Valute"][$j1]["#"]["CharCode"][0]["#"],
        'DATE_RATE' => $rateDay,
        'RATE'           => DoubleVal(str_replace(",",".",$arData["ValCurs"]["#"]["Valute"][$j1]["#"]["Value"][0]["#"])),
        'RATE_CNT' => IntVal($arData["ValCurs"]["#"]["Valute"][$j1]["#"]["Nominal"][0]["#"]),
    );
}

return "AgentGetCurrencyRate();";
}
```



Указанный код добавляется в файл /bitrix/php\_interface/init.php. Не забудьте добавить агента **AgentGetCurrencyRate()**; на странице [Настройки > Инструменты > Агенты](#).

### Запуск агентов из cron

Достаточно часто возникает необходимость переноса исполнения некоторых особо тяжелых агентов на **cron**. И это правильно. Зачем посетителя сайта томить ожиданием окончания очистки таблиц статистики или пересчета форумов, если есть возможность выполнять периодические задания на **cron'e**?

#### **Механизм запуска**

Перейдите на страницу [Настройки > Инструменты > Командная PHP-строка](#) и исполните следующий код:

```
COption::SetOptionString("main", "agents_use_crontab", "Y");
echo COption::GetOptionString("main", "agents_use_crontab", "N");
```

Увидели "Y". С этой секунды на хитах будут исполняться только периодические агенты.

Перейдите на страницу [Настройки > Инструменты > Агенты](#) и настройте показ колонки **Периодический** (для контроля). И отредактируйте нужные вам агенты выставив галочку **Периодический**.

В **cron** добавьте на выполнение команду:

```
/usr/bin/php -f /var/www/bitrix/modules/main/tools/cron_events.php
```

Выставьте периодичность, например: \*/10 \* \* \* \* - что означает раз в десять минут.

**⚠ Примечание:** Непосредственно перед выполнением задания процедура запуска агентов пытается отменить ограничение:

```
@set_time_limit(0);
ignore_user_abort(true);
```

Если **set\_time\_limit** разрешен, то время выполнения может превышать то, что стоит в настройках файла **php.ini**.

Но необходимо помнить, что есть ограничения со стороны хостера: на объем памяти, время выполнения, периодичность запуска и т.д.

#### **Обобщенное решение для выполнения всех агентов из под cron**

Для начала полностью отключим выполнение агентов на хите. Для этого выполним следующую команду в php консоли:

```
COption::SetOptionString("main", "agents_use_crontab", "N");
echo COption::GetOptionString("main", "agents_use_crontab", "N");

COption::SetOptionString("main", "check_agents", "N");
echo COption::GetOptionString("main", "check_agents", "Y");
```

В результате выполнения должно быть "NN".

После этого убираем из файла `/bitrix/php_interface/dbconn.php` определение следующих констант:

```
define("BX_CRONTAB_SUPPORT", true);
define("BX_CRONTAB", true);
```

И добавляем в этот файл:

```
if(!defined("CHK_EVENT") && CHK_EVENT==true)
    define("BX_CRONTAB_SUPPORT", true);
```

Создаем файл проверки агентов и рассылки системных сообщений `/bitrix/php_interface/cron_events.php`:

```
<?
$_SERVER["DOCUMENT_ROOT"] = realpath(dirname(__FILE__)."/../../");
$DOCUMENT_ROOT = $_SERVER["DOCUMENT_ROOT"];

define("NO_KEEP_STATISTIC", true);
define("NOT_CHECK_PERMISSIONS",true);
define('CHK_EVENT', true);

require($_SERVER["DOCUMENT_ROOT"]. "/bitrix/modules/main/include/prolog_before.php");

@set_time_limit(0);
@ignore_user_abort(true);

CAgent::CheckAgents();
define("BX_CRONTAB_SUPPORT", true);
define("BX_CRONTAB", true);
CEvent::CheckEvents();
?>
```

И добавляем данный скрипт в **cron**:

```
*/5 * * * * /usr/bin/php -f /home/bitrix/www/bitrix/php_interface/cron_events.php
```

После этого все агенты и отправка системных событий будут обрабатываться из под cron, раз в 5 минут.

**⚠ Примечание:** Время выполнения можно скорректировать в соответствие с проектом. Кроме того, есть возможность через установку большого значения **mail\_event\_bulk** сделать более "быстрой" доставку почтовых уведомлений. Установка проверки раз в минуту вместе с отправкой за раз 100 сообщений, сделает для пользователей незаметным данную задержку.

Чтобы не увеличивалась очередь отправки почтовых сообщений, рекомендуется изменить параметр отвечающий за количество почтовых событий обрабатываемых за раз. Для этого выполняем в php консоли следующую команду:

```
COption::SetOptionString("main", "mail_event_bulk", "20");
echo COption::GetOptionString("main", "mail_event_bulk", "5");
```

Если очередной запуск **cron\_events.php** произошёл до завершения работы ранее запущенного скрипта, то запуска агентов не произойдет и скрипт завершит свою работу. (Так как агенты блокируются на время выполнения.) В данном случае обработка ничем не отличается от обработки на хите, новый хит может произойти в тот момент когда еще не обработали агенты на предыдущем.

### Еще пример

Вот типичный код скрипта, запускаемого из-под cron:

```
#!/usr/bin/php
<?php
$_SERVER["DOCUMENT_ROOT"] = "/home/hosting/www";
$DOCUMENT_ROOT = $_SERVER["DOCUMENT_ROOT"];
define("NO_KEEP_STATISTIC", true);
define("NOT_CHECK_PERMISSIONS", true);
set_time_limit(0);
define("LANG", "ru");
require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/prolog_before.php");

// ваш код...

require($_SERVER["DOCUMENT_ROOT"]."/bitrix/modules/main/include/epilog_after.php");
```



?>

## Ещё об агентах

На проектах с низкой посещаемостью возможна ситуация, когда большое количество агентов может тормозить работу сайта. Монитор производительности показывает низкую (в районе 1) оценку. Статистика выполнения страниц показывает время генерации от 1 секунды и больше, а некоторые страницы вообще отказываются открываться.

При таких симптомах посмотрите внимательнее статистику выполнения страницы. Если обнаружится что 90% времени генерации страницы занимает пролог сайта, то вероятно проблема в агентах. Достаточно 5-и агентов, у которых выставлено свойство **Периодический**. При выставлении этого свойства система при пропуске выполнения агента в следующие разы пытается компенсировать пропуски, что приводит к перегрузке сервера.

Решение: Выключить периодичность агентов.

## Использование событий

В главе приводятся примеры работы с событиями.

В большинстве случаев действия в системе имеют три рода событий:

- OnBefore;
- On;
- OnAfter.

Важно понимать какой из типов лучше использовать в вашем конкретном случае. События типа **OnBefore** отрабатывают в любом случае. А вот события типа **OnAfter** только тогда, когда будет проверена правильность введенных данных (соответствие пароля, e-mail и т.п.). Соответственно, если что-то в данных не так, то событие не отработает.

## Как написать обработчик события

Поставим перед собой абстрактную задачу: пусть для каждой создаваемой группы в соцсети (или в корпоративе) в описание добавляется информация о том, что ругаться нельзя и некоторому пользователю вообще нельзя создавать группы.

## Анализ вызова события

Для начала разберемся в событиях как таковых. Теорию вы уже знаете, рассмотрим на практике.



Вызов обработчиков всегда одинаковый, меняются только переменные и логика обработки ответа. Для анализа необходимо обратиться к исходным кодам. Это можно сделать просмотрев файлы системы, а лучше с использованием модуля [Живое АПИ](#).

Событие	Вызывается
<a href="#">OnAfterSocNetLogCommentAdd</a>	CSocNetLogComments::Add
<a href="#">OnBeforeSocNetFeatures</a>	CSocNetFeatures::Delete
<a href="#">OnBeforeSocNetFeaturesAdd</a>	CSocNetFeatures::Add
<a href="#">OnBeforeSocNetFeaturesPermsAdd</a>	CSocNetFeaturesPerms::Add
<a href="#">OnBeforeSocNetFeaturesPermsDelete</a>	CSocNetFeaturesPerms::Delete
<a href="#">OnBeforeSocNetFeaturesPermsUpdate</a>	CSocNetFeaturesPerms::Update
<a href="#">OnBeforeSocNetFeaturesUpdate</a>	CSocNetFeatures::Update
<a href="#">OnBeforeSocNetGroupAdd</a>	CSocNetGroup::Add
<a href="#">OnBeforeSocNetGroupDelete</a>	CSocNetGroup::Delete

Выберем для примера событие **OnBeforeSocNetGroupAdd**:

```
$db_events = GetModuleEvents("socialnetwork", "OnBeforeSocNetGroupAdd");
while ($arEvent = $db_events->Fetch())
    if (ExecuteModuleEventEx($arEvent, array(&$arFields)) === false)
        return false;
```

**Количество переменных.** В этом событии переменных всего одна (**\$arFields**). Именно столько же переменных нам надо будет вызвать в нашем обработчике. Переменных также может быть две или больше, например в событии **OnSocNetGroupAdd**:

```
$events = GetModuleEvents("socialnetwork", "OnSocNetGroupAdd");
while ($arEvent = $events->Fetch())
    ExecuteModuleEventEx($arEvent, array($ID, &$arFields));
```

**Переопределение переменных.** Если перед одной из переменных стоит **&**, значит ее можно переопределить (это называется передача по ссылке).

**Отмена действия.** В нашем случае для события **OnBeforeSocNetGroupAdd** есть такая возможность:, если мы в нашем обработчике сделаем *return false*, группа создана не



будет. А, к примеру, в **OnSocNetGroupAdd** возможности отмены действия нет. Ибо действие уже произведено.

### Создание обработчика события

Напомним теорию: для обработки событий в ваших модулях вам надо использовать [RegisterModuleDependences](#). А для обработки в иных случаях вам надо использовать [AddEventHandler](#).

Имя модуля нам известно (**socialnetwork**), имя события известно (**OnBeforeSocNetGroupAdd**), пишем функцию/метод по правилам из теории и не забываем про:

- количество переменных
- возможность переопределения
- отмену действия

### Как узнать что содержится в переменных, какие ключи массива и так далее?

Делаем вывод на экран переменных с завершением работы в теле функции:

```
echo '  
'; print_r($arFields); echo '  
'; die();
```

### Отмена действий

Отмена действия с передачей ошибки в систему:

```
if ($GLOBALS['USER']->GetID() == 2) {  
    $GLOBALS['APPLICATION']->throwException('Вы не можете создавать группы.');//  
    return false;  
}
```

### Результат

Мы собрали абстрактный обработчик, который добавляет к описанию группы правило, и запрещает пользователю с ID=2 создавать группы в принципе.

```
AddEventHandler('socialnetwork', 'OnBeforeSocNetGroupAdd', 'TestHandler');  
  
function TestHandler(&$arFields) {  
    $arFields['DESCRIPTION'] .= ' Ругаться матом запрещено!';  
    if ($GLOBALS['USER']->GetID() == 2) {  
        $GLOBALS['APPLICATION']->throwException('Вы не можете создавать группы.');//
```

```
    return false;  
}  
}
```

**Совет от Антона Долганина:** Лучше не плодить функции, а создать класс ваших обработчиков (в идеале по одному классу на каждый модуль), и писать обработчики внутри классов. Например, CForumHandlers::onBeforeTopicAdd();

### Добавление закладки в социальную сеть

Как добавить произвольную закладку в социальную сеть без изменения стандартных компонентов? Для этого необходимо написать обработчики специальных событий.

Обработчики можно расположить в файле /bitrix/php\_interface/init.php.

```
// Событие происходит при формировании списка дополнительного  
// функционала соц.сети  
// В обработчике можно изменить или дополнить список  
AddEventHandler("socialnetwork", "OnFillSocNetFeaturesList", "__AddSocNetFeature");  
  
// Событие происходит при формировании списка закладок  
// В обработчике можно изменить список закладок  
AddEventHandler("socialnetwork", "OnFillSocNetMenu", "__AddSocNetMenu");  
  
// Событие происходит в комплексном компоненте при работе в ЧПУ  
// режиме при формировании списка шаблонов адресов страниц  
// комплексного компонента  
AddEventHandler("socialnetwork", "OnParseSocNetComponentPath",  
    "__OnParseSocNetComponentPath");  
  
// Событие происходит в комплексном компоненте при работе в  
// не ЧПУ режиме при формировании списка псевдонимов переменных  
AddEventHandler("socialnetwork", "OnInitSocNetComponentVariables",  
    "__OnInitSocNetComponentVariables");  
  
// При формировании списка дополнительного функционала  
// добавим дополнительную запись superficha  
function __AddSocNetFeature(&$arSocNetFeaturesSettings)  
{
```

```

$arResult["superficha"] = array(
    "allowed" => array(SONET_ENTITY_USER, SONET_ENTITY_GROUP),
    "operations" => array(
        "write" => array(SONET_ENTITY_USER => SONET_RELATIONS_TYPE_NONE,
SONET_ENTITY_GROUP => SONET_ROLES_MODERATOR),
        "view" => array(SONET_ENTITY_USER => SONET_RELATIONS_TYPE_ALL,
SONET_ENTITY_GROUP => SONET_ROLES_USER),
    ),
    "minoperation" => "view",
);
}

// При формировании списка закладок добавим дополнительную
// закладку для функционала superficha
function __AddSocNetMenu(&$arResult)
{
    // Доступна для показа
    $arResult["CanView"]["superficha"] = true;
    // Ссылка закладки
    $arResult["Urls"]["superficha"] =
CComponentEngine::MakePathFromTemplate("/workgroups/group/#group_id#/superficha/",
array("group_id" => $arResult["Group"]["ID"]));
    // Название закладки
    $arResult["Title"]["superficha"] = "Моя фича";
}

// При формировании списка шаблонов адресов страниц
// комплексного компонента в режиме ЧПУ добавим шаблон
// для superficha
function __OnParseSocNetComponentPath(&$arResultTemplates, &$arResultCustomPagesPath)
{
    // Шаблон адреса страницы
    $arResultTemplates["superficha"] = "group/#group_id#/superficha/";
    // Путь относительно корня сайта,
    // по которому лежит страница
    $arResultCustomPagesPath["superficha"] = "/bitrix/php_interface/1/";
}

```



```
// Если компонент соц.сети работает в режиме
// ЧПУ, то этот обработчик не нужен
function __OnInitSocNetComponentVariables(&$arVariableAliases, &$arCustomPagesPath)
{
}
```

По пути /bitrix/php\_interface/1/ должен лежать файл **superficha.php**, который содержит код страницы:

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
<?
$APPLICATION->IncludeComponent(
    "bitrix:socialnetwork.group_menu",
    "",
    Array(
        "GROUP_VAR" => $arResult["ALIASES"]["group_id"],
        "PAGE_VAR" => $arResult["ALIASES"]["page"],
        "PATH_TO_GROUP" => $arResult["PATH_TO_GROUP"],
        "PATH_TO_GROUP_MODS" => $arResult["PATH_TO_GROUP_MODS"],
        "PATH_TO_GROUP_USERS" => $arResult["PATH_TO_GROUP_USERS"],
        "PATH_TO_GROUP_EDIT" => $arResult["PATH_TO_GROUP_EDIT"],
        "PATH_TO_GROUP_REQUEST_SEARCH" =>
$arResult["PATH_TO_GROUP_REQUEST_SEARCH"],
        "PATH_TO_GROUP_REQUESTS" => $arResult["PATH_TO_GROUP_REQUESTS"],
        "PATH_TO_GROUP_REQUESTS_OUT" => $arResult["PATH_TO_GROUP_REQUESTS_OUT"],
        "PATH_TO_GROUP_BAN" => $arResult["PATH_TO_GROUP_BAN"],
        "PATH_TO_GROUP_BLOG" => $arResult["PATH_TO_GROUP_BLOG"],
        "PATH_TO_GROUP_PHOTO" => $arResult["PATH_TO_GROUP_PHOTO"],
        "PATH_TO_GROUP_FORUM" => $arResult["PATH_TO_GROUP_FORUM"],
        "PATH_TO_GROUP_CALENDAR" => $arResult["PATH_TO_GROUP_CALENDAR"],
        "PATH_TO_GROUP_FILES" => $arResult["PATH_TO_GROUP_FILES"],
        "PATH_TO_GROUP_TASKS" => $arResult["PATH_TO_GROUP_TASKS"],
        "GROUP_ID" => $arResult["VARIABLES"]["group_id"],
        "PAGE_ID" => "group_superficha",
    ),
    $component
);
?>
```

Полезный код страницы...

### Учет регистрации нового пользователя в статистике

Для многих проектов важно отслеживать все новые регистрации на сайте в статистике для дальнейшего подробного анализа (например, откуда приходят пользователи, которые регистрируются). Отслеживать лучше всего через механизм Событий. При использовании Событий, появляется возможность смотреть отчеты по числу регистраций за день и строить график регистраций по времени.

Для решения задачи используется обработчик события **OnAfterUserRegister**. Код обработчика будет таким:

```
AddEventHandler("main", "OnAfterUserRegister", "OnUserEmailLoginRegisterHandler");

function OnUserEmailLoginRegisterHandler(&$arFields)
{
    if(CModule::IncludeModule("statistic") && intval($_SESSION["SESS_SEARCHER_ID"]) <= 0)
    {
        $event1 = "register";
        $event2 = "new_user";
        $event3 = $arFields["EMAIL"];
        CStatistic::Set_Event($event1, $event2, $event3);
    }
    return $arFields;
}
```

В результате в отчетах модуля статистики появятся данные о регистрациях:

event1	event2	event3	Дата	≡	Сессия	≡	Посет.	≡	Страна
register	new_user	vitaliy.panarin@gmail.com	27.04.2010 23:40:53		3871		3053	[RU] RUSSIAN FEDERATION	

## Дополнительно

**Вопрос:** Можно ли вызывать функцию [AddEventHandler](#) несколько раз для одного и того же события?

**Ответ:** Вызывать несколько раз функцию AddEventHandler с одинаковыми первыми двумя параметрами можно. Случай, когда это не так (возможен лишь один обработчик на событие) очень редки. При повторном вызове желательно указывать четвёртый параметр, который отвечает за очерёдность вызова обработчиков. Если не указать этот параметр, то обработчики будут вызваны в порядке добавления.

**Как обработчику события узнать, какое событие он обрабатывает?**

Функция является обработчиком событий модулей (функция не знает каких модулей и каких событий). Но она должна, учитывая какое произошло событие, совершать различные действия.

**Вопрос:** как обработчику события узнать, какое событие он обрабатывает? Как следует инициализировать такой обработчик функциями [RegisterModuleDependences](#) и [AddEventHandler](#)?

**Решение:** сделайте прослойку.

```
function OnAdd()
{
    RealHandler("add");
}

function OnUpdate()
{
    RealHandler("update");
}
```

## Пользовательские поля

**Пользовательское поле** - функционал системы, позволяющий добавлять к объектам системы поля, не предусмотренные штатным функционалом.

Необходимо отличать **Пользовательские поля** в модулях системы и [свойства используемые в рамках инфоблоков](#), хотя в формах системы (форма

создания/редактирования пользователя, форме создания/редактирования раздела инфоблока и другие) используется термин **пользовательские свойства**.

Пользовательские поля это сущность:

- более универсальная, так как их можно задать для разных объектов системы, в отличие от **свойств инфоблока**,
- более ограниченная по возможностям, так как имеет небольшое число типов данных.

Пользовательские поля могут создаваться в неограниченном количестве для каждого объекта. При выборе того или иного типа пользовательского поля становятся доступными дополнительные поля настройки для соответствующего типа. Детально об этом можно узнать [в документации](#).

Применение пользовательских полей в системе к тем или иным модулям задаётся с помощью объектов, которые необходимо указать при создании поля. Не все модули имеют объекты для пользовательских полей по умолчанию. Разработчик может создавать собственные объекты, но надо понимать, что в методах **GetList** поддерживаются только системные объекты:

Штатные объекты пользовательских полей		
Модуль	Объект	Описание
Главный модуль	<b>USER</b>	для пользователя
	<b>BLOG_BLOG</b>	для блога
Блоги	<b>BLOG_POST</b>	для сообщения в блоге
	<b>BLOG_COMMENT</b>	для комментария сообщения
Задачи	<b>TASKS_TASK</b>	для задач
Информационные блоки	<b>IBLOCK_N_SECTION</b>	для секций инфоблока с ID = N
	<b>IBLOCK_N</b>	для инфоблока с ID = N
Календарь	<b>CALENDAR_EVENT</b>	для событий календаря
Обучение	<b>LEARN_ATTEMPT</b>	для попыток теста

Социальная сеть	<b>SONET_GROUP</b>	для групп соцсети
Библиотека документов	<b>WEBDAV</b>	для библиотек документов

 **Примечание:** Модули, использующие информационные блоки, могут работать с объектами пользовательских полей модуля **Информационные блоки**.

## Создание полей

Создание пользовательских полей из Административной части выполняется на странице *Настройки > Настройки продукта > Пользовательские поля* либо, что предпочтительнее, с использованием ссылки Добавить пользовательское свойство в тех формах системы, в которых предусмотрено штатное добавление пользовательских свойств:

- форма добавления/редактирования пользователя;
- форма добавления/редактирования раздела информационного блока;
- форма добавления/редактирования блога

Использовать страницу **Пользовательские поля** можно в случае, если разработчик точно знает, какой идентификатор типа объектов ему нужен.

Работа со [списком](#) и [формой создания](#) не должна вызвать затруднений, но есть несколько нюансов.

Установка флажка в поле **Не разрешать редактирование пользователем** исключит возможность редактирования свойства не только пользователем, но и администратором через административный интерфейс. Значение таких свойств нужно устанавливать используя API. Это нужно для служебных полей, которые не должны использовать пользователи.

Пользовательские поля могут создаваться с разными типами данных. По умолчанию в системе предусмотрены следующие типы:

- Число;
- Да/Нет;
- Видео;
- Шаблон;
- Список;
- Страна;
- Дата/Время;



- Привязка к разделам инф. блоков;
- Привязка к элементам инф. блоков;
- Привязка к элементам CRM;
- Привязка к справочникам CRM;
- Файл;
- Целое число.

Как правило, этих типов вполне хватает для работы. Если есть необходимость создания собственных типов данных, то это можно сделать самостоятельно. Пример добавления типов данных ["Связь с элементом"](#) и ["Связь с элементом в виде списка"](#) (блог).

Для работы с пользовательскими полями можно использовать **События**.

События **Главного модуля** при работе с пользовательскими полями:

Событие	Вызывается	Метод
OnUserTypeBuildList	при построении списка пользовательских полей	CUserTypeManager::GetUserType
OnUserTypeRightsCheck	при проверке прав доступа на пользовательские поля	GetRights.

## Примеры работы

### Фильтрация

Пользовательские поля разделов могут принимать участие в фильтрации.

```
$sec_Filter= array(
    "IBLOCK_ID" => $IBLOCK_ID,
    "DEPTH_LEVEL" => "2",
    "!UF_ARC_PAGES" => ""
);
```

**⚠ Примечание:** Фильтрация по пользовательским полям работает только при наличии фильтра по **IBLOCK\_ID**.

Будут отобраны все разделы, у которых установлено значение свойства **UF\_ARC\_PAGES**.



Фильтрация по значению пользовательского свойства:

```
$arSFilter ['=UF_USERS_PROPERTY'] = $users_property_value;
```

### Сортировка

Сортировать по пользовательским полям разделов:

```
$arSort = array(  
    "UF_RATING"=>"asc",  
    "sort"=>"asc"  
  
);
```

### Получение значений

Получить значение пользовательского поля можно с помощью метода [GetList](#) соответствующего класса.

Значение пользовательского поля для пользователя с ID=2 можно таким образом:

```
$rsUser = CUser::GetByID($user);  
$arUser = $rsUser->Fetch();  
$нужное значение = $arUser['код пользовательского поля'];
```

Чтобы получить значение пользовательского поля определенного пользователя, тип поля – строка, необходимо воспользоваться методом [GetList](#) класса **CUser**. При этом в качестве четвертого аргумента данному методу необходимо передать массив с ключом **SELECT**, значениями которого являются список кодов пользовательских свойств, которые необходимо получить.

```
global $USER;  
$arFilter = array("ID" => $USER->GetID());  
$arParams["SELECT"] = array("UF_USER_CARD_CODE");  
$arRes = CUser::GetList($by,$desc,$arFilter,$arParams);  
if ($res = $arRes->Fetch()) {  
    echo $res["UF_USER_CARD_CODE"];  
}
```

Если тип пользовательского поля список, то для получения значения (или значений, если возможен множественный выбор) нужно воспользоваться методом [GetList](#) класса **CUserFieldEnum**.



```
global $USER;
$arFilter = array("ID" => $USER->GetID());
$arParams["SELECT"] = array("UF_LIST_TASK");
$arRes = CUser::GetList($by,$desc,$arFilter,$arParams);
if ($res = $arRes->Fetch()) {
    foreach ($res["UF_LIST_TASK "] as $id) {
        $rsRes= CUserFieldEnum::GetList(array(), array(
            "ID" => $id,
        ));
        if($arResult = $rsRes->GetNext())
            echo $arGender["VALUE"];
    }
}
```

Если необходимо получить список всех значений пользовательского поля объекта **USER** типа список, то следует воспользоваться следующим кодом:

```
global $USER_FIELD_MANAGER;
$arFields = $USER_FIELD_MANAGER-> GetUserFields("USER");
$obEnum = new CUserFieldEnum;
$rsEnum = $obEnum->GetList(array(), array("USER_FIELD_ID" => $arFields["UF_LIST_TASK "
""]["ID"]));
while($arEnum = $rsEnum->GetNext()){
    echo $arEnum["VALUE"];
}
```

Для выбора значения пользовательского поля у секции информационного блока можно воспользоваться методом [CIBlockSection::GetList](#):

```
$aSection = CIBlockSection::GetList( array(), array(
    'IBLOCK_ID' => 3,
    'CODE' => 'test_section',
), false, array( 'UF_DEV2DAY_FIELD' ) )->Fetch();
```

**⚠ Примечание:** Передача идентификатора инфоблока (*IBLOCK\_ID*) обязательна, иначе выборка пользовательских свойств не будет осуществлена.

Получение значения пользовательского поля типа файл конкретной секции инфоблока:

```
$rsResult = CIBlockSection::GetList(array("SORT" => "ASC"), array("IBLOCK_ID" => "1"), false,
$arSelect = array("UF_*"));
```



```
while ($arResult = $rsResult -> GetNext())
{
print "
". print_r($arResult, true) .
";
}
```

Так как пользовательские поля можно использовать не только с разделами информационного блока, но и с любыми другими сущностями, то для выбора значений по идентификатору сущности используется класс **CUserTypeManager**. Экземпляр данного класса уже находится в глобальной переменной **\$USER\_FIELD\_MANAGER**.

```
global $USER_FIELD_MANAGER;

$aSection = CIBlockSection::GetList( array(), array(
    'IBLOCK_CODE' => 'shop_news',
    'CODE'        => 'test_section',
) )->Fetch();

if( !$aSection ) {
    throw new Exception( 'Секция не найдена' );
}

$aUserField = $USER_FIELD_MANAGER-> GetUserFields(
    'IBLOCK_3_SECTION',
    $aSection['ID']
); // array
```

В результате мы получим массив содержащий в себе всю информацию о поле и его значении для конкретного объекта.

**⚠ Примечание:** Чтобы получить все значения пользовательских полей в параметре **arSelect** достаточно указать **Array("UF\_ \*")**.

## Добавление, редактирование, удаление пользовательских свойств и их значений

За работу с пользовательскими полями отвечает класс [CUserTypeEntity](#).



### Пример добавления пользовательского свойства типа Стока

```
/**  
 * Добавление пользовательского свойства  
 */  
$oUserTypeEntity = new CUserTypeEntity();  
  
$aUserFields = array(  
/*  
 * Идентификатор сущности, к которой будет привязано свойство.  
 * Для секция формат следующий - IBLOCK_{IBLOCK_ID}_SECTION  
*/  
    'ENTITY_ID' => 'IBLOCK_3_SECTION',  
/* Код поля. Всегда должно начинаться с UF_ */  
    'FIELD_NAME' => 'UF_DEV2DAY_FIELD',  
    /* Указываем, что тип нового пользовательского свойства строка */  
    'USER_TYPE_ID' => 'string',  
/*  
 * XML_ID пользовательского свойства.  
 * Используется при выгрузке в качестве названия поля  
*/  
    'XML_ID' => 'XML_ID_DEV2DAY_FIELD',  
/* Сортировка */  
    'SORT' => 500,  
/* Является поле множественным или нет */  
    'MULTIPLE' => 'N',  
/* Обязательное или нет свойство */  
    'MANDATORY' => 'N',  
/*  
 * Показывать в фильтре списка. Возможные значения:  
 * не показывать = N, точное совпадение = I,  
 * поиск по маске = E, поиск по подстроке = S  
*/  
    'SHOW_FILTER' => 'N',  
/*  
 * Не показывать в списке. Если передать какое-либо значение,  
 * то будет считаться, что флаг выставлен (недоработка разработчиков битрикс).  
*/
```



```
'SHOW_IN_LIST' => '',
/*
* Не разрешать редактирование пользователем.
* Если передать какое-либо значение, то будет считаться,
* что флаг выставлен (недоработка разработчиков битрикс).
*/
'EDIT_IN_LIST' => '',
/* Значения поля участвуют в поиске */
'IS_SEARCHABLE' => 'N',
/*
* Дополнительные настройки поля ( зависят от типа).
* В нашем случае для типа string
*/
'SETTINGS' => array(
    /* Значение по умолчанию */
    'DEFAULT_VALUE' => '',
    /* Размер поля ввода для отображения */
    'SIZE' => '20',
    /* Количество строчек поля ввода */
    'ROWS' => '1',
    /* Минимальная длина строки (0 - не проверять) */
    'MIN_LENGTH' => '0',
    /* Максимальная длина строки (0 - не проверять) */
    'MAX_LENGTH' => '0',
    /* Регулярное выражение для проверки */
    'REGEXP' => '',
),
/* Подпись в форме редактирования */
'EDIT_FORM_LABEL' => array(
    'ru' => 'Пользовательское свойство',
    'en' => 'User field',
),
/* Заголовок в списке */
'LIST_COLUMN_LABEL' => array(
    'ru' => 'Пользовательское свойство',
    'en' => 'User field',
),
```



```
/* Подпись фильтра в списке */
'LIST_FILTER_LABEL' => array(
    'ru' => 'Пользовательское свойство',
    'en' => 'User field',
),

/* Сообщение об ошибке (не обязательное) */
'ERROR_MESSAGE' => array(
    'ru' => 'Ошибка при заполнении пользовательского свойства',
    'en' => 'An error in completing the user field',
),
/* Помощь */
'HELP_MESSAGE' => array(
    'ru' => '',
    'en' => '',
),
);

$iUserFieldId = $oUserTypeEntity->Add( $aUserFields ); // int
```

При удачном добавлении свойства в переменную **\$iUserFieldId** будет возвращен идентификатор нового пользовательского свойства.

Для создания пользовательских полей других типов замените значение **USER\_TYPE\_ID**:

- **enumeration** - Список
- **double** - Число
- **integer** - Целое число
- **boolean** - Да/Нет
- **string** - Стока
- **file** - Файл
- **video** - Видео
- **datetime** - Дата/Время
- **iblock\_section** - Привязка к разделам инф. блоков
- **iblock\_element** - Привязка к элементам инф. блоков
- **string\_formatted** - Шаблон
- **crm** - Привязка к элементам CRM
- **crm\_status** - Привязка к справочникам CRM



## Обновление пользовательского свойства

При обновлении пользовательского свойства накладываются ограничения на изменение его типа (**USER\_TYPE\_ID**), объекта привязки (**ENTITY\_ID**), кода поля (**FIELD\_NAME**). Это связано с возможными ошибками связей значений и сущностей. Если необходимо изменить одно из этих полей, то нужно сначала создать новое пользовательское свойство, привязать все значения к нему, а затем удалить старое свойство.

Пример обновления пользовательского свойства:

```
$oUserTypeEntity->Update( $iUserFieldId, array(
    'MANDATORY' => 'Y',
) ); // boolean;
```

В примере установлено требование обязательности заполнения поля.

## Удаление пользовательского поля

Необходимо передать идентификатор пользовательского поля:

```
$oUserTypeEntity->Delete( $iUserFieldId ); // CDBResult
```

## Добавление и обновление значений пользовательских полей

Добавление и обновление реализовано также через класс **CUserTypeManager** и метод **Update**.

```
global $USER_FIELD_MANAGER;

$aSection = CIBlockSection::GetList( array(), array(
    'IBLOCK_CODE' => 'shop_news',
    'CODE'        => 'test_section',
) )->Fetch();

if( !$aSection ) {
    throw new Exception( 'Секция не найдена' );
}

$USER_FIELD_MANAGER->Update( 'IBLOCK_3_SECTION', $aSection['ID'], array(
    'UF_DEV2DAY_FIELD' => 'updated value'
) ); // boolean
```

В случае успешного обновления метод вернет *true*.



## Поля к нештатным объектам и новые объекты

### Создание пользовательского поля к нештатным объектам

Иногда возникает необходимость создавать пользовательские поля к объектам, у которых нет поддержки пользовательских полей по умолчанию. В таком случае, можно самостоятельно создать пользовательское свойство для этого объекта.

Рассмотрим это на примере комментариев блога. Например, у каждого комментария необходимо создавать свойство **Рейтинг**. Создаем в административной части пользовательское свойство нужного типа (Настройки > Настройки продукта > Пользовательские поля). Заполняем все поля, в поле **Объект** указываем любое придуманное имя объекта, главное, чтобы оно было уникально. В нашем случае, напишем **BLOG\_RATING**. Для считывания и записи значений пользовательских свойств можно использовать следующие функции:

```
function SetUserField ($entity_id, $value_id, $uf_id, $uf_value) //запись значения
{
    return $GLOBALS["USER_FIELD_MANAGER"]->Update ($entity_id, $value_id,
        Array ($uf_id => $uf_value));
}

function GetUserField ($entity_id, $value_id, $uf_id) //считывание значения
{
    $arUF = $GLOBALS["USER_FIELD_MANAGER"]->GetUserFields ($entity_id, $value_id);
    return $arUF[$uf_id]["VALUE"];
}

// $entity_id - имя объекта (у нас "BLOG_RATING")
// $value_id - идентификатор элемента (вероятно, ID элемента, свойство которого мы сохраняем или получаем. в нашем случае, это ID комментария)
// $uf_id - имя пользовательского свойства (в нашем случае UF_RATING)
// $uf_value - значение, которое сохраняем
```

### Пример использования:

```
SetUserField ("BLOG_RATING", $CommentID, "UF_RATING", $Rating);
echo "Рейтинг комментария: ". GetUserField ("BLOG_RATING", $CommentID, "UF_RATING");
```

Создание пользовательских полей вручную не так удобно, как использование функций **GetList** для объектов с поддержкой пользовательских свойств по умолчанию. Однако, он



позволяет максимально быстро и просто использовать в самописных компонентах и модулях пользовательские свойства для произвольных объектов.

### Создание собственного объекта

Можно создать любой объект и работать с ним как вам удобно. Пример:

```
$GLOBALS["USER_FIELD_MANAGER"]->Update("GRADEBOOK_RESULT", $ID,
Array("UF_TEACHERS"=>$arValue));
$arUserFields = $GLOBALS["USER_FIELD_MANAGER"]-> GetUserFields("GRADEBOOK_RESULT",
$ID);
```

### Гаджеты

**Гаджет** – особый программный элемент, выполняющий функцию вывода определенных данных.

Об имеющихся гаджетах, их настройке и управлении ими, вы можете прочитать в курсах:

- [Контент-менеджер](#);
- [Администратор Корпоративного портала](#).

Для отображения гаджетов используется компонент **Рабочий стол (Desktop)**. Этот одностораничный компонент позволяет создать настраиваемый рабочий стол с использованием гаджетов. В компоненте гаджеты инсталлируются с главным модулем системы. Разработчики могут создавать собственные гаджеты. Компонент их увидит, если они будут размещаться в папке `/bitrix/gadgets/`. Системные гаджеты располагаются во вложенной папке `/bitrix/`.

 **Внимание!** Категорически не рекомендуется без крайней необходимости трогать структуру системных гаджетов.

### Структура гаджета

- `.description.php` - файл описания ;
- `.parameters.php` - файл с настройками ;
- файл `index.php`, который содержит исполняемый код, реализующий задачу гаджета;
- языковые файлы в папке `/lang/`.

Гаджеты и компоненты имеют похожую структуру и назначение, однако:

- Гаджеты не используют шаблоны. HTML код зашифтован в файле `index.php`, в отличие от компонентов, где представление и логика разнесены.

- Гаджеты могут запоминать настройки для каждого пользователя, в отличие от компонентов, которые могут только выводить или не выводить информацию в зависимости от прав доступа.
- Настройки у гаджетов разделены на 2 группы: общие настройки, для гаджетов одного типа (например, для всех гаджетов **Новости** в рамках одного **Рабочего стола**), а также настройки каждого конкретного гаджета. Общие настройки задаются в компоненте **Рабочий стол**. Индивидуальные настройки задаются в контекстном меню каждого конкретного гаджета.

Превосходя визуальные компоненты в мобильности, гаджеты являются очень интересным инструментом как для разработчиков, так и для пользователей сайтов.

## Тестирование проектов

Для тестирования сайтов оптимально подходит метод **чеклиста** - списка операций, которые нужно обязательно выполнить.

**Контрольный список** (*перечень, таблица, карта; англ. checklist*) — список факторов, свойств, параметров, аспектов, компонентов, критериев или задач, структурированных особым образом с целью достижения поставленных задач.

В процессе тестирования много важных мелочей, которые повторяются из проекта в проект. Программисты часто говорят, что помнят их - на самом деле опыт показывает, что не помнят, пропускают важные моменты и занимаются "наступанием на грабли". Не зря опытные руководители проектов закрывают этапы/релизы только при наличии оформленных ответственными сотрудниками чеклистов-отчетов: начиная от верстальщиков, заканчивая системными администраторами.

Чеклисты обычно составляются опытными сотрудниками. По мере накопления интеллектуального багажа компании в базах знаний, развиваются и совершенствуются и чеклисты. Суть проста — сэкономить время и деньги, не позволив сотрудникам непроизвольно (а иногда и произвольно) наступать на грабли 2 и более раз.

Рекомендуется и вам составить собственный **чеклист** для сдачи проектов. Список может быть разным, но как показывает опыт сообщества в такой лист обязательно должны быть включены следующие моменты:

- **Поиск.** Если на сайте используется форма поиска, то её следует протестировать в первую очередь. Обязательно надо проверить, чтобы ссылки в результатах не былибитые. (Битые ссылки присутствуют в анонсах в подробных описаниях просто в статическом контенте в настройках инфоблоков.) Если на сайте используется несколько инфоблоков с динамической информацией, то нужно сделать поисковые запросы, чтобы найти по отдельности элементы различных инфоблоков и проверить, корректно ли работают ссылки в результатах поиска.



- **F5.** Во всех формах, в которых пользователь может оставлять какие-либо данные (например, комментарии), нужно проверить, не отправляются ли данные снова при нажатии кнопки F5 в браузере.
- **Пользователи и права доступа.** Следует протестировать сайт в трёх состояниях: неавторизованным, авторизованным и авторизованным под администратором. Часто бывает, что создавая инфоблок из-под администратора, ему забывают поставить нужные права доступа, и он остается доступным только для администраторов.
- **Карта сайта** Не всегда для этой страницы подойдёт стандартный компонент **Карта сайта**. Если основная информация на сайте – каталог товаров (сделанный, естественно, с помощью модуля инфоблоков), то в карте сайта логично вывести разделы каталога, а не только разделы сайта.
- **Дефолтные шаблоны.** Необходимо проверить, не используются ли где-нибудь на сайте дефолтные шаблоны. Может получиться не очень хорошо, если, например, пользователь пытается восстановить пароль, ему на почту уходит письмо со ссылкой на сайт, он нажимает на ссылку и видит не привычный ему дизайн сайта, а пример дефолтного «корпоративного» сайта "1С-Битрикс: Управление сайтом".
- **Стили в визуальном редакторе.** Хорошо, когда контент-менеджер, пользуясь визуальным редактором, видит в нём текст, максимально похожий на текст, который отобразится на сайте. То есть для абзацев, заголовков и т.д. свой кегль шрифта, цвет и т.д.
- **Динамическая информация во включаемых областях** Если во включаемой области содержится какая-либо динамическая информация, то она может закешироваться, что скорее всего приведет к нежелательному результату. Лучше вообще не использовать динамическую информацию во включаемой области.
- **Резервная копия, обновления.** Необходимо обновить Битрикс до актуальной версии, сделать резервную копию сайта и сохранить её на локальный компьютер.
- **Кеш.** Следует проверить, включено ли автокеширование. Если оно не включено, то нужно его включить и повторить всё тестирование с самого начала
- **Права доступа.** Проверка на корректность прав доступа различных групп пользователей генерация отчета по правам групп в папках сайта.
- **Ядро системы.** Проверка работы ядра и соответствие его оригиналу с генерацией отчета о папках компонентах и модулях не соответствующих оригиналу и генерации отчета о несоответствии версий. Иногда шаловливые ручки позволяют себе править непосредственно компоненты **Bitrix Framework** и, чтобы не порушить функционал сайта, очень полезно перед тем как обновлять проект узнать что наделано не так и что наделано дополнительно.
- **Неиспользуемые файлы.** Часто при разработке или изменениях делают резервные копии и хорошо если делают приставку old а иногда просто весь этот мусор остается, плюс неиспользуемые картинки и т.п.
- **Сторонние подключения.** Часто к сайту подключают различные партнерские программы: баннеры, скрипты и т.п., вещи без которых невозможна монетизация.



Но при этом может пострадать безопасность. Необходимо проверять подобный функционал на предмет безопасности и наличия уязвимостей (доступ на редактирование, доступ к ядру, возможность внедрения кода, сохранения через подключаемый модуль файлов)

### Из опыта веб-разработчиков.

**Роман Петров:** Один из простых способов протестировать на ошибки разработки ваш сайт на "1С-Битрикс: Управление сайтом" - это перед запуском проекта, включить [тест производительности](#) на нужное время и на том же сервере запустить 3-4 одновременных команды `wget` на зеркалирование сайта. Первое обращение создаст кэш страницы, а остальные - обратятся уже к закэшированным файлам. Это позволит очень легко выявить все проблемные странички (мелкие ошибки "по забывчивости", ошибки верстки (пропущенные файлы и др.)) и исправить ошибки.

Если у вас еще не достаточно опыта для составления собственного чеклиста, то воспользуйтесь штатным инструментом **Монитор качества**.

### Монитор качества

Веб-проект – сложный и комплексный продукт. Он отличается от традиционного софта тем, что веб-сайт – это результат взаимодействия трех участников: клиента, партнера и разработчика платформы. В результате непонимания этих трех сторон возникает проблема качества внедрения.

Для разрешения этой проблемы создан инструмент **Монитор качества**. Он позволяет решить задачу обеспечения прозрачного и гибкого процесса сдачи веб-проекта клиенту, повышая уровень гарантированного результата и снижая общие риски.

**Монитор качества** - инструмент для проверки качества выполненного проекта перед сдачей его заказчику.

Монитор качества это:

- Структурированная методика управления качеством внедрения;
- Система тестов для веб-разработчиков, набор рекомендаций для клиентов;
- Состоит из обязательных тестов и необязательных;
- Включает автоматических проверок.

Монитор качества дает дополнительные возможности разработчикам и клиентам:

**Разработчикам:**



- Систематизация процедуры тестирования;
- Повышение качества создания интернет-проектов за счет систематизации производства;
- Формализация отношений с клиентом как на этапе сдачи, так и на этапе поддержки.

**Клиентам:**

- Снижение рисков: чем раньше найдена проблема, тем дешевле ее устранить;
- Систематизация приемки проекта и запуска его в эксплуатацию: шаги расписаны, можно сосредоточиться на деталях;
- Формализация и упрощение взаимодействия с разработчиком на этапе поддержки и развития проекта;
- Снижение затрат на получение качественного результата;
- Высокая производительность и безопасность веб-решения.

Тесты представлены в виде дерева, организованного согласно этапам типичного внедрения. Однако можно выполнять тесты в любом удобном порядке. Тесты состоят из **обязательных** и **необязательных** тестов. Некоторые тесты могут быть **автоматизированными**. Автоматизированы сложные и рутинные проверки.

**Обязательный тест** - критичный тест для качества веб-решения. Может быть пропущен при условии наличия комментария разработчика. В общем списке помечаются черным цветом.

**Необязательный тест** - некритичный, но рекомендуемый для прохождения для нагруженных, сложных, больших проектов. В общем списке помечаются серым цветом.

Для сдачи проекта по чеклисту необходимо добиться успешного прохождения обязательных тестов. Необязательные тесты – призваны существенно улучшить качество решения и снизить риски.

**Автоматизированный тест** - тест, данные по которому собирает система. Конечное решение в любом случае за разработчиком. Автоматизированный тест можно запустить повторно.

**Сдача проекта**

Сдача проекта производится на странице **Монитор качества** ([Настройки > Инструменты > Монитор качества](#)). При первом открытии отобразится страница с вводной информацией. Запуск теста произойдет после нажатия на кнопку

[Сдать проект](#), после чего откроется дерево тестов с предложением запуска автотестирования. Автотестирование можно отложить и запустить его позже.

## Автоматизированный тест

После запуска автоматизированного теста система соберет данные и предложит отобразить тесты, которые по ее мнению пройдены, а какие - нет:

 Монитор качества

Рабочий стол > Настройки > Инструменты > Монитор качества

Выполнено тестов: 10 из 65

Всего тестов: 65  
Обязательных: 26  
Пройдено успешно: 5  
Не пройдено: 5  
Необязательных: 39  
В ожидании: 55

Тест: Настройки компонентов - отделены и описаны

**Завершить...**

**+ Интеграция дизайна и разработка (3/38)**

**- Безопасность (2/1/7)**

- ✓ 1. Используется проактивная защита. Минимальный уровень - «Стандартный» 1
- 2. Административные учетные записи - защищены
- 3. Группам пользователей предоставлены минимально необходимые права
- 4. Удалены тестовые данные
- ✗ 5. Настроены политики безопасности по работе с БД 2
- ✗ 6. Отключен вывод ошибок и предупреждений посетителям проекта 1
- 7. Настроен журнал системных событий

По результатам одного только автоматического тестирования тест сдать не получится, даже если все автоматические тесты будут пройдены:



9. Используется HTML-кэширование

10. Настроен веб-клUSTER в соответствии с требованиями

**[-] Размещение на хостинге (1/0/6)**

- 1. Введена информация о хостинге
- 2. Настройки хостинга - оптимизированы
- 3. Тесты окружения платформы - пройдены
- 4. Настроена передача IP-адреса
- ✗ 5. Выполняется резервное копирование**
- 6. Настроена почтовая подсистема

**[+] Сдача проекта (0/4)**

Вы не можете закрыть проект. Все **обязательные** тесты должны быть в статусе "Пропущен" либо в "Пройден". Так же в дереве не должно быть не пройденных тестов.

Сейчас пройдено обязательных: **4 из 26**  
Тестов в статусе "Не пройден": 5

**Закрыть**

## Ручное прохождение тестов

Необходимо вручную открыть непройденные или ручные тесты и вручную перевести их в тот или иной статус и дать, при необходимости описание. Тест открывается кликом по его названию. Откроется форма теста:

**Настроены политики безопасности по работе с БД – QSEC0050 (обязательный)**

**Тест**    **Описание**

Название: Настроены политики безопасности по работе с БД (QSEC0050)

Статус теста

Ожидает     Пройден успешно     Не пройден     Пропущен

Результат автотеста:  
Найдены ошибки  
[Подробный отчет](#)

**Запустить автотест**

Тестирующий

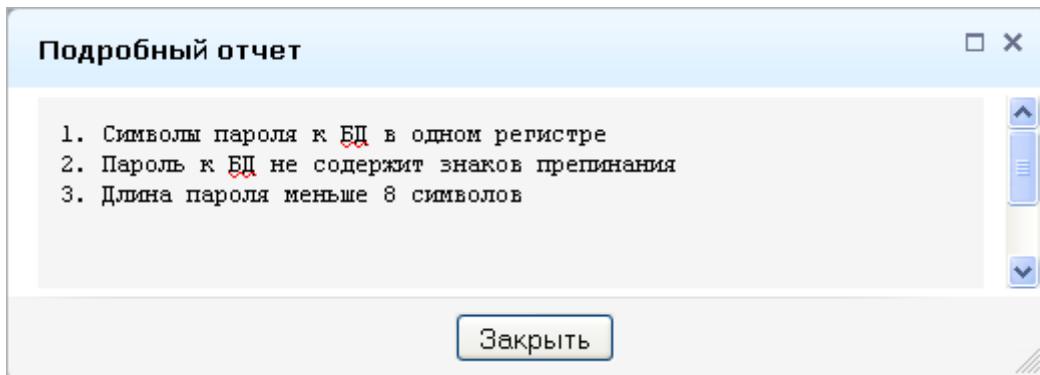
Разработчик

**Сохранить комментарий**

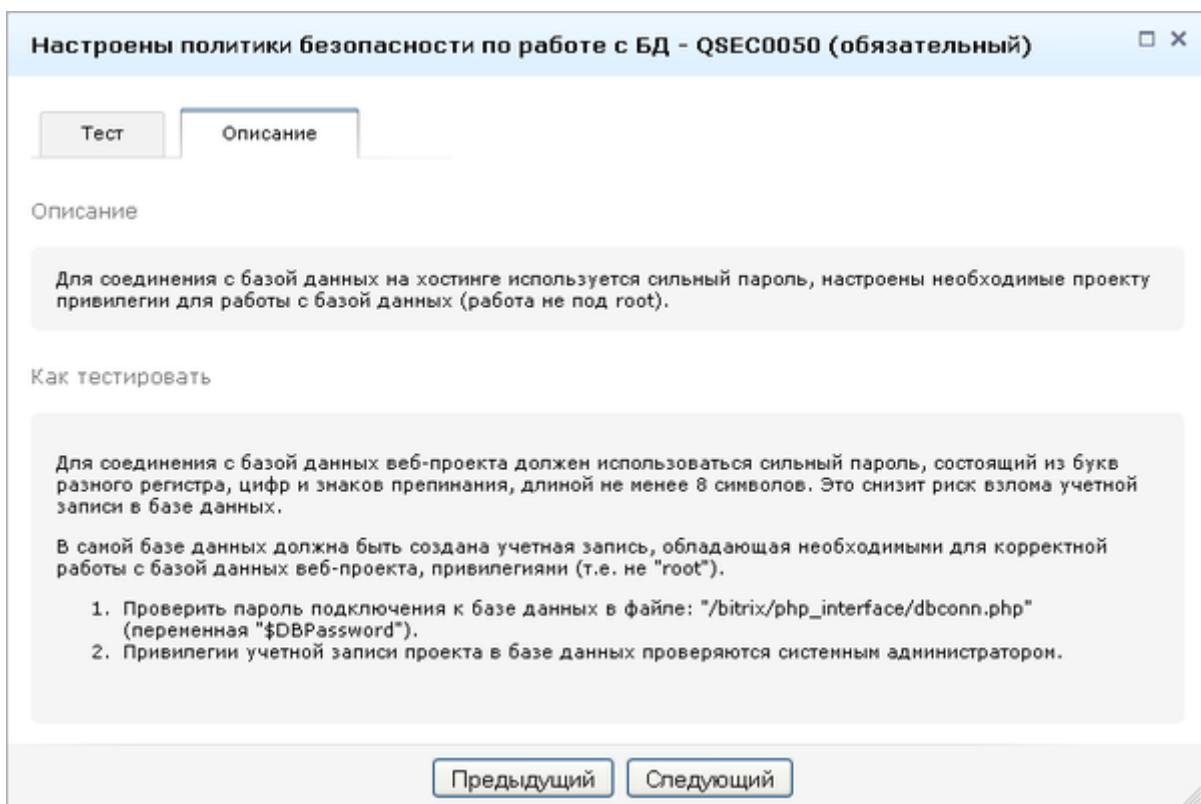
**Предыдущий**    **Следующий**



Если тест не пройден, то по ссылке Подробный отчет можно просмотреть причины непрохождения:



В закладке **Описание** можно посмотреть условия теста и какие параметры в системе нужно проверить и исправить для правильного прохождения теста:



После исправления ошибок можно:

- запустить повторно автоматический тест (для автоматизированных тестов);
- вручную сменить статус теста.

**Примечание:** ручной перевод обязательного теста в статус **Пропущен** требует обязательного добавления комментария.



После прохождения всех обязательных тестов веб-проект можно успешно сдать, после чего отчет по тестированию попадает в архив.

Результат тестирования: **28 из 28** тестов пройдено успешно  [сдать проект](#)

Дата: 12.10.2011

<b>Компания</b> <input type="text" value="ООО \" интегратор\""=""/>	<b>ФИО сотрудника</b> <input type="text" value="Иванов Иван Иванович"/>
<input style="margin-right: 5px;" type="button" value="Выберите файл"/> <input type="text" value="tnefgw...036.jpg"/> <a href="#">загрузить изображение</a>	
Подробный комментарий - что дорабатывалось, какие допущения указывались при тестировании и т.д.	

[Сохранить отчет](#)

## Архив тестов

**Монитор качества**

Рабочий стол > Настройки > Инструменты > Монитор качества

Архив отчетов

Дата	Кто проводил	Всего тестов	Пройдено успешно	Не пройдено	
12.10.2011 02:57:08	Иванов Иван Иванович	65	28	0	<a href="#">подробности</a>

[Сдать проект](#)

Отчеты можно просмотреть в любое время, получив подробную информацию по каждому тесту (в том числе и системные сообщения автотестов).

## Способы использования

Монитор качества можно использовать в нескольких вариантах.

### Базовое тестирование по чеклистику

Партнер/Разработчик организует тестирование выполненной интеграции по чеклистику и выступает в роли тестировщика. Затем предъявляет клиенту успешно сданный отчет, доступный в административном интерфейсе в разделе Настройки > Инструменты > Контроль качества – в котором все обязательные тесты успешно пройдены.

## Углубленное тестирование по чеклиству

Разработчик и клиент договариваются о необходимости углубленного тестирования по чеклиству качества высоконагруженного проекта. Партнер выступает в роли тестировщика и добивается успешного прохождения всех (большинства) тестов чеклиста + своих тестов. Клиент обращает внимание на число доступных тестов и число пройденных успешно, просматривая отчет по тестированию в архиве отчетов.

## Внутренняя разработка

Клиент силами собственной команды разработки осуществляет интеграцию решения. В роли тестировщика выступает команда тестирования клиента. Одно структурное подразделение клиента сдает проект другому подразделению - работу координирует менеджер проекта.

## Итерационное развитие с минимальными рисками

Партнер оказывает клиенту услугу по доработке функционала действующего веб-проекта, первоначальная интеграция которого была выполнена с использованием **Монитора качества**. Партнер сдает работу, формируя отчет по тестированию. Клиент проверяет отчет по тестированию и следит, чтобы все доработки веб-проекта регистрировались в архиве отчетов.

## Модификация тестов

Система допускает модификацию тестов под нужды разработчика используя штатный механизм Событий Bitrix Framework.

Разработчики, при необходимости, могут сами добавить свои тесты и разделы в **Монитор качества**.

Так же инструмент можно адаптировать под нужды конкретной задачи, создав собственные разделы и тесты. Например:

- Тесты по SEO-оптимизации;
- Тест на CodeStyle;
- Тест на корректность работы биллинга под нагрузкой;
- и другие.

## Как дополнить штатные тесты

Сначала нам необходимо описать свои тесты и их разделы. Для этого вешаем обработчик на событие Главного модуля **onCheckListGet**. Событие вызывается в конструкторе **CCheckList** с аргументом **\$arCheckList** следующего вида:

```
array(2) {
```



```
["CATEGORIES"]=>
array(10) {
    ["QDESIGN"]=>
        array(0) {
            []
        }
    ["DESIGN"]=>
        array(1) {
            ["PARENT"]=>
                string(7) "QDESIGN"
            []
        }
    ["MODEL"]=>
        array(1) {
            ["PARENT"]=>
                string(7) "QDESIGN"
            []
        }
    ["STANDART"]=>
        array(1) {
            ["PARENT"]=>
                string(7) "QDESIGN"
            []
        }
    []
}
["POINTS"]=>
array(65) {
    ["QD0010"]=>
        array(2) {
            ["PARENT"]=>
                string(6) "DESIGN"
            ["REQUIRE"]=>
                string(1) "Y"
            []
        }
    ["QD0020"]=>
        array(5) {
            ["REQUIRE"]=>
                string(1) "Y"
            ["PARENT"]=>
                string(6) "DESIGN"
            ["AUTO"]=>
                []
        }
    []
}
```



```
string(1) "Y"
["CLASS_NAME"]=>
string(10) "CAutoCheck"
["METHOD_NAME"]=>
string(14) "CheckTemplates"
}
)
```

Нетрудно заметить, что **CATEGORIES** содержит список разделов чеклиста, которые могут быть вложенными, а **POINTS** - сами тесты.

Ключ массива **CATEGORIES** — ID раздела, значение — массив параметров:

- **NAME** - наименование раздела;
- **LINKS**

Пример описания раздела:

```
$checkList['CATEGORIES']['ITC_QC'] = array(
'NAME' => 'Корпоративный тест качества ITConstruct',
'LINKS' => ''
);
```

Ключом элементов **POINTS** является символьный идентификатор теста, а значение представляет собой массив следующих параметров:

- **NAME** - наименование теста;
- **DESC** - краткое описание теста (вкладка **Описание**, блок **Описание**);
- **HOWTO** - длинный текст о том, что будет проверяться (вкладка **Описание**, блок **Как тестировать**);
- **LINKS** - аналогично разделам;
- **PARENT** - ID раздела, обязательное;
- **REQUIRE** - флаг обязательности теста (Y/N);
- **AUTO** - "Y", если является автотестом;
- **CLASS\_NAME** - имя класса теста (для автотеста);
- **METHOD\_NAME** - имя метода теста (для автотеста);
- **FILE\_PATH** - подключение файла теста, если оный вынесен в отдельный скрипт (для автотеста). Путь - относительно **DOCUMENT\_ROOT**;
- **PARAMS** - массив дополнительных параметров, передаваемых первым аргументом при вызове метода автотеста.



**⚠ Примечание:** Такие ключи как **NAME**, **DESC** и **LINKS** можно не определять массиве описания теста, они являются языковозависимые и жесткое определение их прямо в обработчике лишает возможности локализации.

Достаточно подключить свой языковой файл примерно с таким содержанием:

```
$MESS["CL_ITC_QC_FAICON_NAME"] = 'Наличие favicon';
$MESS["CL_ITC_QC_FAICON_DESC"] = 'Проверка наличия favicon - иконки сайта';
$MESS["CL_ITC_QC_FAICON_LINKS"] = 'блаблабла';
```

И эти языковые фразы будут подтягиваться в поля **NAME**, **DESC** и **LINKS** теста с кодом **ITC\_QC\_FAICON**. К **HOWTO** это тоже будет относится, но пока его можно определить только в массиве описания пункта.

Пример:

```
$checkList['POINTS']['ITC_QC_FAICON'] = array(
    'PARENT' => 'ITC_QC',
    'REQUIRE' => 'Y',
    'AUTO' => 'Y',
    'CLASS_NAME' => __CLASS__,
    'METHOD_NAME' => 'checkFavicon',
    'NAME' => 'Наличие favicon',
    'DESC' => 'Проверка наличия favicon - иконки сайта, отображаемой в заголовке вкладки и поисковых системах',
    'HOWTO' => 'Производится проверка главной страницы сайта на наличие соответствующего мета-тэга. Если тэг объявлен - проверяется наличие иконки по указанному урлу. Если не указан - наличие favicon.ico в корне сайта',
    'LINKS' => 'links'
);
```

Обработчик события может возвращать как изменённый **\$arCheckList**, так и новый массив с разделами и тестами - если категория/тест с неким ID уже существует, то он не заменится, т.е. подкорректировать системные тесты не получится.

Теперь необходимо объявить метод для автотеста. Для альфа-версии логика теста будет тривиальна:

```
$check = file_exists($_SERVER['DOCUMENT_ROOT'] . '/favicon.ico')
```

По результатам проверки мы должны вернуть массив. Автотест может быть одно- или многошаговым. Если тест многошаговый и текущая итерация не является конечной, необходимо вернуть массив следующего вида:



```
$arResult = array(
    'IN_PROGRESS' => 'Y',
    'PERCENT' => '42',
);
```

**PERCENT** служит лишь для визуализации прогресса на странице и никуда не сохраняется для последующего использования. Промежуточные данные для идентификации прогресса шага надо сохранять самим - в сессию, временный файл, базу (в зависимости от объема данных и прочих условий).

Если же тест закончен, сообщаем статус массивом, содержащим следующие ключи:

- **STATUS**- результат теста. **true**, если успешно, нечто иное, если тест провалился. В коде проверяется так:

```
if ($result['STATUS'] == "true")
```

- **MESSAGE** - разъяснения результата:

- **PREVIEW** - краткий текст результата;
- **DETAIL** - расширенное объяснение, открываемое во всплывающем окне.

Готовый метод теста фавиконки имеет вид:

```
static public function checkFavicon($arParams)
{
    $arResult = array('STATUS' => 'F');
    $check = file_exists($_SERVER['DOCUMENT_ROOT'] . '/favicon.ico');

    if ($check === true) {
        $arResult = array(
            'STATUS' => true,
            'MESSAGE' => array(
                'PREVIEW' => 'Favicon найдена - ' . '/favicon.ico',
            ),
        );
    } else {
        $arResult = array(
            'STATUS' => false,
            'MESSAGE' => array(
                'PREVIEW' => 'Favicon не найдена',
                'DETAIL' => 'Тест очень старался, но так и не смог найти фавиконку. Ну и чёрт с ней',
            ),
        );
    }
}
```

```
        ),  
    );  
}  
  
return $arResult;  
}
```

## Результат

Тест в мониторе качества:

<a href="#">+</a>	Интеграция дизайна и разработка (1/2/38)
<a href="#">+</a>	Безопасность (2/1/7)
<a href="#">+</a>	Производительность (2/1/10)
<a href="#">+</a>	Размещение на хостинге (1/0/6)
<a href="#">+</a>	Сдача проекта (1/4)
<a href="#">-</a>	Корпоративный тест качества ITConstruct (1/2) ✓ 1. Наличие favicon  1 2. Закрытие доступа извне к dev-серверу

Справочная информация о teste:



**Наличие favicon - ITC\_QC\_FAVIDICON (обязательный)**

Описание

Проверка наличия favicon - иконки сайта, отображаемой в заголовке вкладки и поисковых системах

Как тестировать

Производится проверка главной страницы сайта на наличие соответствующего мета-тэга. Если тэг объявлен - проверяется наличие иконки по указанному урлу. Если не указан - наличие favicon.ico в корне сайта

Тест пройден:

**Наличие favicon - ITC\_QC\_FAVIDICON (обязательный)**

Название: Наличие favicon (ITC\_QC\_FAVIDICON)

Статус теста

Ожидает  Пройден успешно  Не пройден  Пропущен

Результат автотеста:  
Favicon найдена - /favicon.ico

Автотест завершен

Тестирующий

Разработчик



## Код

Код обработчика и теста целиком:

```
AddEventHandler('main', 'OnCheckListGet', array('CltcCheckListTests', 'onCheckListGet'));

class CltcCheckListTests
{
    static public function onCheckListGet($arCheckList)
    {
        $checkList = array('CATEGORIES' => array(), 'POINTS' => array());

        $checkList['CATEGORIES'][['ITC_QC']] = array(
            'NAME' => 'Корпоративный тест качества ITConstruct',
            'LINKS' => ''
        );

        $checkList['POINTS'][['ITC_QC_FAVICON']] = array(
            'PARENT' => 'ITC_QC',
            'REQUIRE' => 'Y',
            'AUTO' => 'Y',
            'CLASS_NAME' => __CLASS__,
            'METHOD_NAME' => 'checkFavicon',
            'NAME' => 'Наличие favicon',
            'DESC' => 'Проверка наличия favicon - иконки сайта, отображаемой в заголовке вкладки и поисковых системах',
            'HOWTO' => 'Производится проверка главной страницы сайта на наличие соответствующего мета-тэга. Если тэг объявлен - проверяется наличие иконки по указанному урлу. Если не указан - наличие favicon.ico в корне сайта',
            'LINKS' => 'links'
        );

        $checkList['POINTS'][['ITC_QC_DENY_DEV']] = array(
            'PARENT' => 'ITC_QC',
            'REQUIRE' => 'N',
            'AUTO' => 'N',
            'NAME' => 'Закрытие доступа извне к dev-серверу',
            'DESC' => 'Согласовать с менеджером закрытие доступа ко внутреннему серверу разработок из внешнего мира',
        );
    }
}
```



```
'HOWTO' => 'Попинговать с телефона после апдейта днса',
);

return $checkList;
}

static public function checkFavicon($arParams)
{
    $arResult = array('STATUS' => 'F');
    $check = file_exists($_SERVER['DOCUMENT_ROOT'] . '/favicon.ico');

    if ($check === true) {
        $arResult = array(
            'STATUS' => true,
            'MESSAGE' => array(
                'PREVIEW' => 'Favicon найдена - ' . '/favicon.ico',
            ),
        );
    } else {
        $arResult = array(
            'STATUS' => false,
            'MESSAGE' => array(
                'PREVIEW' => 'Favicon не найдена',
                'DETAIL' => 'Тест очень старался, но так и не смог найти фавиконку. Ну и чёрт с ней',
            ),
        );
    }

    return $arResult;
}
}
```

## Несколько советов

### **Цитатник веб-разработчиков.**

**Максим Месилов:** Если на текущем проекте сбоит штатный функционал, то следует попробовать смоделировать аналогичную ситуацию в [демо-лаборатории](#) от 1С-Битрикс. Эта ссылка должна быть всегда под рукой.

## 2 способа отладки веб-приложений

### **var\_dump – способ**

Самый простой вариант – использование оператора **var\_dump()**: получение состава переменной, даже будь это объект или массив. Если обернуть вывод этого оператора в `<pre>`, то будет удобочитаемо.

### **irePHP – способ**

Есть более технологичный и в конечном итоге удобный способ просмотра содержимого переменных. Для этого нам понадобиться установленный браузер FireFox, установленное расширение FireBug и установленное расширение [FirePHP](#).

Загрузите с этого сайта последнюю версию класса для работы с расширением FirePHP и подключите этот класс к своему движку:

1. Скопируйте файл **fb.php** в папку `/bitrix/php_interface/`
2. Добавьте в файл `/bitrix/php_interface/init.php` строку:

```
require_once('FirePHPCore/fb.php');
```

Теперь можно использовать логгирование в консоль FireBug. В простейшем варианте это делается так: `fb($var)`, если нужно поставить метку, то `fb($var, 'Label');`



## Глава 10. Многосайтовость

**Многосайтовость** - это возможность системы «1С-Битрикс: Управление сайтом» управлять разными сайтами из единой Панели управления.

Особенностями системы многосайтовости являются:

- единые права на управление модулями сайта;
- единый набор бюджетов пользователей на все сайты;
- единая система ведения статистики на все сайты.

**⚠ Внимание!** По лицензионному соглашению на одной копии продукта нельзя создавать независимые сайты, физически размещенные на разных серверах, имеющие отдельную копию ядра продукта и отдельную базу данных.

В **Bitrix Framework** имеется возможность на базе одного экземпляра продукта создавать и поддерживать неограниченное количество сайтов. Для создания дополнительных сайтов необходимо приобрести дополнительные лицензии на необходимое количество сайтов.

В рамках системы **сайт** - это совокупность:

- **Учетной записи в базе данных.** Создается в Административном разделе ([Настройки > Настройки продукта > Сайты > Список сайтов](#)), включает в себя следующие основные параметры:
  - **Идентификатор** - набор символов, идентифицирующих сайт;
  - **Доменное имя** - одно или несколько доменных имен сайта;
  - **Папка сайта** - путь к каталогу, в котором будет храниться публичная часть сайта;
  - **Язык сайта;**
  - **Формат даты;**
  - **Формат времени;**
  - **URL** - протокол и доменное имя по умолчанию (например, <http://www.site.ru>);
  - **DocumentRoot.** Если многосайтовость реализуется на разных доменах, то в данном параметре должен храниться путь к корню сайта в файловой системе сервера;
  - **Условия подключения шаблонов.** Каждый сайт может иметь более одного шаблона для отображения своей публичной части, каждый такой шаблон может быть подключен по тому или иному условию.
- **Публичная часть** - совокупность страниц, лежащих в "папке сайта" и принадлежащих этому сайту.



- **Настройки** - каждый модуль системы может иметь ряд настроек, связанных с сайтом.

Например, у модуля **Информационные блоки** эти настройки представляют собой привязку информационного блока к тому или иному сайту, а у модуля **Техподдержка** - привязку к сайту статуса, категории обращений и т.п.

 **Важно!** В публичной части ID текущего сайта хранится в константе **SITE\_ID**.

## Использование многосайтовой версии

При создании сайта CMS устанавливается только в корневой каталог веб-сервера. После установки системы статические страницы и файлы, предназначенные для вывода в разных зеркалах сайта (например, языковых), размещаются в соответствующих подпапках корневого каталога. Если на одном и том же хостинге вами создается несколько сайтов, вы должны были проинсталлировать систему в каждый корневой каталог соответствующего сайта. Очевидно, что такой подход приводит к полной децентрализации управления вашими проектами. Более того, дисковое пространство хостинга расходуется не оптимально.

Система **Bitrix Framework** разработана с учетом требований компаний, желающих поддерживать несколько сайтов (например, языковых зеркал или тематических секций). Концепция **многосайтости** требует существенно меньших ресурсов для поддержки проектов и позволяет управлять всеми сайтами из одной точки.

Ядро системы позволяет использовать следующие типы URL для идентификации сайтов, привязываемых к системе:

- **относительные, по папкам** (/ru/, /en/) - для работы зеркал сайта на одном домене под управлением одной системы;
- **абсолютные, по домену** (www.site.ru) - для работы различных сайтов под управлением одной системы;

Для создания нескольких сайтов без установки системы в каждой корневой папке каждого сайта или для создания нескольких сайтов, не располагая их в подпапках корневой папки, следует создать два веб-сервера (или виртуальных хоста одного веб-сервера) с разными корневыми папками и в каждой из них, используя средства операционной системы, создать **symbolic links** (символьные ссылки) папок /bitrix и /upload.

Итак, общие сущности всех проектов на одном хостинге - папки /bitrix, /upload и база данных. Разделение сайтов осуществляется по статическим страницам сервера и по полю **Сайт** объектов базы данных (новостей, опросов и т.п.).

 **Важно:**

*В любом из режимов многосайтности используется единое ядро и единая база данных. В результате:*

- *невозможно использование многосайтности на разных редакциях программы;*
- *невозможно создание раздельных администраторов для разных сайтов;*
- *оба сайта должны использовать одну и ту же кодировку.*

Возможна ситуация, когда требуется перенести два сайта с разных установок на одну с многосайтовой конфигурацией. Такая возможность технически реализуема, но штатно не предусмотрена.

## Многоязычность

Система **1С-Битрикс: Управление сайтом** позволяет представлять информационное наполнение сайтов на различных языках:

- *создавать неограниченное число сайтов на разных языках. Например, на русском, английском или немецком языке.*

The image shows two side-by-side screenshots of website product pages. On the left is the 1C-Bitrix website, featuring a banner for the 'Corporate Portal' product, which is described as a system for managing corporate websites, e-commerce sites, social networks, and communities. It highlights features like integrated communication, collaboration, and document sharing. On the right is the Bitrix website, also featuring a banner for the 'Bitrix Intranet Portal Solution 8.0', described as enhancing workforce productivity. Both pages include navigation menus, search bars, and other typical website elements.

- Выбор и настройка параметров языка публичного раздела выполняется отдельно для каждого сайта в форме создания и редактирования сайта ([Настройки > Настройки продукта > Сайты > Список сайтов](#)):



**Параметры**

**Параметры сайта**

\*ID: s1  
Активен:   
\*Название: Моя компания

**Параметры для определения сайта в публичном разделе:**

По умолчанию:

Доменное имя:  
(список доменных имен, каждое в новой строке)

\*Папка сайта: /

\*Сортировка: 1

**Параметры:**

\*Язык: [ru] Russian ▾

\*Формат даты:  
(например: DD.MM.YYYY) DD.MM.YYYY

\*Формат даты и времени:  
(например: DD.MM.YYYY HH:MI:SS) DD.MM.YYYY HH:MI:SS

\*Кодировка: UTF-8

**Параметры**

**Параметры сайта**

\*ID: s1  
Активен:   
\*Название: My Company

**Параметры для определения сайта в публичном разделе:**

По умолчанию:

Доменное имя:  
(список доменных имен, каждое в новой строке)

\*Папка сайта: /en/

\*Сортировка: 2

**Параметры:**

\*Язык: [en] English ▾

\*Формат даты:  
(например: DD.MM.YYYY) DD.MM.YYYY

\*Формат даты и времени:  
(например: DD.MM.YYYY HH:MI:SS) DD.MM.YYYY HH:MI:SS

\*Кодировка: UTF-8



**⚠ Примечание:** дополнительно в настройках сайта можно определить формат показа даты и времени для используемого на сайте языка.

- публиковать на разных языках как статическую, так и динамическую информацию в рамках одного сайта.

**⚠ Примечание:** Использование различных сайтов для представления информации на разных языках позволяет осуществлять более гибкое управление контентом сайта. Например, настраивать формат показа даты, валюту и т.д. Для отдельных сайтов можно эффективно анализировать статистику, управлять валютами, разграничивать доступ к заказам Интернет-магазина.

Однако прежде чем создавать несколько сайтов для каждого из языков, следует предварительно проанализировать, какая именно информация должна быть представлена на различных языках и какие действия с объектами сайта предполагается производить. Например, будет ли необходимо анализировать статистику посещений раздельно по языковым разделам, потребуется ли разделять валюты для различных языков, права на заказы, созданные в различных языковых разделах, есть ли необходимость гибко позиционировать рекламу или веб-формы на сайте.

Если информационное наполнение будет представлено в основном статическими страницами и новостными блоками или каталогом, то для реализации такой модели можно просто разделить контент по соответствующим разделам на сервере, например для каждого языка выделить свою директорию. Аналогичным образом могут быть созданы отдельные информационные блоки для различных языков или заведены веб-формы.

Соответственно в различных языковых разделах сайта могут быть представлены свои новостные блоки и каталоги. Для этого достаточно выбрать информационный блок в настройках компонентов.

**⚠ Обратите внимание:** для различных разделов на сайте могут быть установлены различные шаблоны сайта. Для каждого шаблона можно задать отдельную кодовую страницу. Таким образом, можно представить каждый языковой раздел в отдельном дизайне.

## Чем определяется количество сайтов в системе

Система «1С-Битрикс: Управление сайтом» позволяет создавать несколько сайтов с применением одной копии (лицензии) продукта, размещая ядро и базу данных системы в единственном экземпляре на сервере.

Количество сайтов в системе определяется лицензионным соглашением и лицензией на дополнительные сайты. Если необходимо создать дополнительные сайты сверх текущего максимального количества, то следует приобрести лицензии на дополнительные сайты.

Согласно лицензионному соглашению на каждой копии продукта может быть создано два сайта. Это подразумевает, что оба сайта будут использовать общее программное ядро и общую базу данных.

**!** *Важно:* по лицензионному соглашению на одной копии продукта нельзя создавать независимые сайты, физически размещенные на разных серверах, имеющие отдельную копию ядра продукта и/или отдельную базу данных.

### Дополнительные сайты

Для каждой копии продукта может быть дополнительно приобретено неограниченное количество сайтов. Цена дополнительных сайтов зависит от редакции используемого продукта.

При приобретении лицензий на дополнительные сайты вам выдаются купоны на дополнительные сайты (один купон - одна дополнительная лицензия). Дополнительные лицензии (и купоны соответственно) зависят от редакции продукта. Например, для создания дополнительного сайта в продукте редакции **Эксперт** необходимо приобрести лицензию (купон) на дополнительный сайт для редакции **Эксперт**.

После получения купона вам необходимо перейти на закладку **Дополнительно** на главной странице системы обновлений ([Настройки > Marketplace > Обновление платформы](#)). В поле **Введите купон** введите полученный купон и нажмите на кнопку **Активировать купон**.

Если купон действителен, принадлежит соответствующей редакции продукта и не был уже применен ранее, то максимальное количество сайтов для данной копии продукта будет увеличено на одну штуку.

Подробную информацию по цене и условиям приобретения дополнительных сайтов можно найти на сайте компании [1С-Битрикс](#).

### Неограниченная лицензия

Лицензия на сайты «без ограничения» дает возможность создавать неограниченное количество сайтов, работающих на общем программном ядре и использующем единую базу данных.

Использование неограниченной лицензии может быть выгодно для компаний, имеющим в составе множество филиалов, отделений или департаментов, для каждого из которых предполагается наличие собственного ресурса.



## Языковые версии сайта

Очень часто возникает вопрос: для проекта необходимо представить на сайте материалы на разных языках, потребуется ли для этого покупать дополнительные лицензии на сайты?

Многосайтовость и языковые версии сайта – это разные понятия, хотя иногда взаимосвязанные. При необходимости языковые версии можно реализовать и как отдельные языковые папки (разделы) в составе одного сайта и как отдельные сайты.

В данном случае следует предварительно проанализировать, какая именно информация должна быть представлена на различных языках и какие действия с объектами сайта предполагается производить.

### Разные языковые папки

Если функционал сайта исчерпывается статическими страницами, новостными блоками и каталогом, то вполне можно использовать в качестве языковой версии сайта одну из директорий.

Вариант с языковыми версиями в виде разных языковых папок допускает создания любого количества версий сайта, однако он имеет свои ограничения. При выборе такого способа возникнут проблемы с анализом статистики посещений раздельно по языковым разделам, с разделением валют для различных языков, с правами на заказы, созданные в различных языковых разделах, с гибким позиционированием рекламы или веб-форм на сайте.

**⚠ Внимание!** Создание языковых версий сайтов как отдельных языковых папок требует хорошего знания системы и программирования высокого уровня, связанные с корректным распознанием необходимой пользователю языковой версии проекта. Поэтому мы рекомендуем реализовывать языковые версии именно как многосайтовость, то есть в виде отдельного сайта на другом языке.

**⚠ Примечание:** для различных разделов на сайте могут быть установлены различные шаблоны сайта. Для каждого шаблона можно задать отдельную кодовую страницу. Таким образом, можно представить каждый языковой раздел в отдельном дизайне.

Для задания разных шаблонов для разных папок в этом случае в настройках сайта нужно выбрать условие для отображения шаблона "для файла или папки" и ввести в поле название папки. Например: /ru/



## Языковые версии как отдельные сайты

Если функционал сайта требует раздельного ведения интернет-магазина, рекламы, статистики и разных валют, то рекомендуется создать языковые версии как отдельные сайты.

**⚠ Внимание!** Использование различных сайтов дает более гибкие возможности по настройке, дополнительно можно указывать формат вывода даты для показа новостей и прочее. Для отдельных сайтов можно эффективно анализировать статистику, управлять валютами, разграничивать доступ к заказам Интернет-магазина.

## Технология переноса посетителей между сайтами

Особенностями многосайтовой системы являются:

- единые права на все сайты
- единый набор бюджетов пользователей на все сайты
- единая система ведения статистики на все сайты
- и т.п.

Исходя из этого, становится актуальной задача распознать одного и того же посетителя, приходящего на разные сайты с разными доменными именами в рамках одного портала.

Распознавание посетителей осуществляется с помощью файлов **cookie** (куков), представляющих из себя информацию, передаваемую между веб-сервером и браузером и хранимую только на локальном диске посетителя.

Теперь постараемся объяснить на примере суть проблемы:

- При первом заходе посетителя на **сайт А** ему выдаются ряд идентификаторов, используемых разными модулями (например, идентификатор посетителя в модуле статистики или идентификатор покупателя в модуле интернет-магазина и т.д.), которые запоминаются в хранимых **cookie** принадлежащих **сайту А**.
- Когда посетитель в следующий раз возвращается на этот же **сайт А**, он будет "узнан" благодаря информации хранимой в **cookie**, принадлежащих **сайту А**.
- Теперь представим, что этот же посетитель пришел на **сайт В**. Возникает задача "узнать" его как посетителя в недавнем прошлом **сайта А**. Под термином "узнать" здесь понимается - получить идентификаторы, выданные ему на **сайте А**. Проблема осложняется тем, что если доменное имя **сайта В** отличается от доменного имени **сайта А**, то информация хранимая в **cookie** принадлежащих **сайту А** не может быть получена при заходе посетителя на **сайт В**. Также есть обратная проблема - **cookie** устанавливаемые с **сайта А** (и на этот же **сайт А**) не могут быть установлены на **сайт В**. Такова политика безопасности браузеров.



Для решения вышеописанных проблем используется технология переноса **cookie** посетителя между разными сайтами с разными доменными именами и принадлежащих одному порталу. Она имеет также название **UserMultiSiteTransfer**.

Алгоритм работы технологии можно описать так:

- Когда посетитель заходит на **сайт А**, идентификаторы, выдаваемые ему, будут сохраняться в **cookie** с помощью функции **CMain::set\_cookie**, основная задача которой не только установить **cookie** для текущего **сайта А**, но и запомнить данные этого **cookie** для дальнейшего распространения его на другие **сайты В, С, D**.
- В конце визуальной части эпилога вызывается функция **CMain::ShowSpreadCookieHTML**. Эта функция выводит набор IMG'ов, в каждом из которых вызывается скрипт **spread.php** с того домена на который необходимо установить **cookie**. Таким образом для **сайтов В, С, D** будет создано три IMG'a, в каждом из которых будет вызван скрипт <http://доменное имя сайта/bitrix/spread.php>. В параметрах этого скрипта будет передана необходимая информация для установки **cookie**. Эта информация передается в зашифрованном виде и подписана зашифрованным лицензионным ключом этого портала. В результате получится, что **cookie**, установленный на **сайте А**, будет скопирован (перенесен) на другие **сайты - В, С, D**.
- Аналогично происходит и для других сайтов. Если посетитель, зайдя на **сайт В**, получит какой либо идентификатор который необходимо сохранить в **cookie**, то этот идентификатор будет также сохранен и для других **сайтов А, С, D**. Таким образом мы добиваемся единого набора **cookie** для всех сайтов одного портала.

Использование данной технологии позволяет:

- В модуле **Веб-аналитика** подсчитывать уникальных посетителей для всего портала.
- В модуле **Реклама, баннеры** позволяет корректно учитывать количество показов одного баннера одному посетителю.

Другие модули также активно используют эту методику.

**⚠ Примечание:** Технология **UserMultiSiteTransfer** будет использоваться для сайтов многосайтовой конфигурации, если активирована опция: **Распространять куки на все домены в настройках Главного модуля**.



Настройки    Авторизация    Журнал событий    Система обновлений    Доступ

### Настройка параметров модуля

**Системные настройки**

Язык по умолчанию для административной части: [ru] Russian

Название сайта: Моя компания

URL сайта (без http://): Например: www.mysite.com  
localhost:6448

Имя префикса для названия cookies (без точек и пробелов): BITRIX\_SM

Распространять куки на все домены:  (This field is highlighted with a red rectangle)

Посыпать в заголовке статус 200 на 404 ошибку:

Режим вывода ошибок (error\_reporting): Только ошибки

Использовать визуальный редактор для редактирования шаблонов сайта:

## Конфигурирование многосайтности

Технически многосайтовая версия продукта может быть реализована в двух конфигурационных режимах:

- **Многосайтность на одном домене.** (Старое название: Многосайтность по первому способу.) Продукт и все сайты работают под управлением одной копии веб-сервера Apache.
- **Многосайтность на разных доменах.** (Старое название: Многосайтность по второму способу.) Каждый сайт работает под управлением отдельной копии веб-сервера Apache или отдельного виртуального веб-сервера.

С точки зрения программного продукта, оба способа равнозначны. Выбор того или иного способа многосайтности зависит от целей, преследуемых разработчиком.

**Многосайтность на одном домене** рекомендуется:

- если вам необходимо создание сайтов на разных языках или, например, региональных сайтов одной компании;
- если планируется, что оба сайта будут использовать общее доменное имя;
- если вы используете виртуальный хостинг, не позволяющий создать несколько виртуальных веб серверов с общим доступом к файлам.

В этом случае URL будут представлены как:



*http://example.com/s1 (http://example1.com/s1)*

*http://example.com/s2 (http://example.com/s)*

или как:

*http://example.com/ (http://example1.com/)*

*http://example.com/s2 (http://example.com/s2)*

**Многосайтовость на разных доменах** рекомендуется:

- если тематика создаваемых сайтов разная или требуется создание сайтов с уникальными URL;
- если вы используете выделенный сервер или на разделяемом хостинге можете настроить раздельные веб-сервера.

Использование многосайтовости на разных доменах так же позволяет исключить из URL-ов на сайте лишние подкаталоги /s1/ или /s2/ и начинать формирование URL адресов прямо от каталога "/". То есть адреса сайтов будут иметь вид:

*http://www.example.ru/*

*http://www.example.com/*

*http://www.second.example.ru/*

Дистрибутив программного продукта поставляется сконфигурированным для многосайтовости на одном домене. Для использования многосайтовости на разных доменах необходима дополнительная настройка.

Обе конфигурации будут рассмотрены на примере создания многосайтовой системы, состоящей из двух сайтов:

- [www.site1.com](http://www.site1.com) - корпоративный сайт компании
- [www.site2.com](http://www.site2.com) - Интернет-магазин компании

**⚠ Примечание.** При последующей необходимости выделить сайты из многосайтовости и сделать их полностью автономными помните, что системных механизмов разделения сайтов и их баз данных не существует. Такое решение возможно, но зависит от уровня подготовки программиста и индивидуально в каждом случае.

## Создание и настройка сайта

Перед настройкой системы на работу с несколькими сайтами необходимо создать новый сайт. Это осуществляется в Административном разделе на странице **Список сайтов**



([Настройки > Настройки продукта > Сайты > Список сайтов](#)) В момент добавления записи о новом сайте в таблицу сайтов необходимо указать следующие параметры:

- **идентификатор сайта** – двухсимвольная комбинация, например: **ru, en, de, s1, s2** и т.п.
- **название** – произвольное название сайта, наряду с идентификатором сайта используется в различных административных формах для указания привязки к тому или иному сайту.
- **доменное имя** – указываются доменные имена, которые соответствуют данному сайту.

**⚠ Примечание:** Доменные имена задаются в отдельной строке (одно имя на строку).

Каждое доменное имя является «маской», т.е. может включать субдомены. Например, по имени site.ru могут быть выбраны www.site.ru или www1.site.ru, а также my-site.ru. В данном случае site.ru частично входит в состав всех трех доменных имен.

Доменное имя: (список доменных имен, каждое в новой строке)	mysite.com mysite.info	При определении сайта каждый элемент этого списка интерпретируется как корневой домен (например, значение site.com будет означать www.site.com, my.site.com, ...)
*Папка сайта:	/	
*Сортировка:	100	

**⚠ Важно!**

Будьте внимательны при указании доменных имен сайта. Если на сайте используется технология cookies, то в случае некорректного указания доменных имен при просмотре пользователем страниц вашего сайта также будет выполняться обращение и к другому сайту (доменное имя которого указано по ошибке).

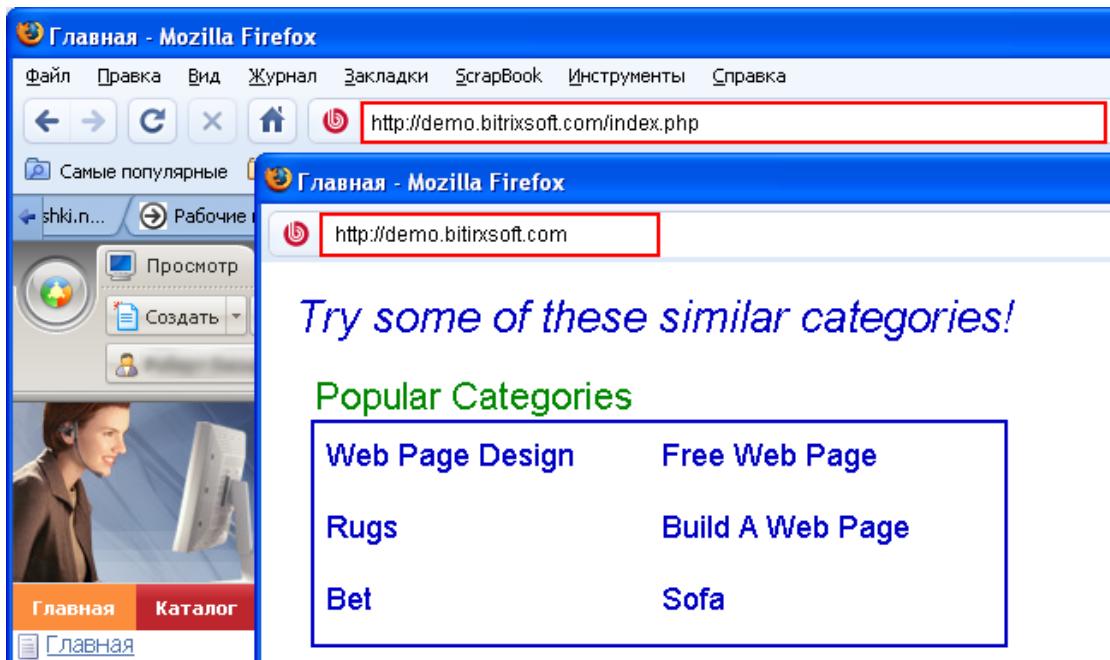


Если на сайте, доменное имя которого указано по ошибке, предусмотрен показ всплывающих окон, то посетителям вашего сайта также будет выполняться показ этих окон.

Например, если для сайта <http://demo.bitrixsoft.com/> вместо [demo.bitrixsoft.com](http://demo.bitrixsoft.com) указано [demo.bitirxsoft.com](http://demo.bitirxsoft.com) (ошибка в написании второго i), то в код страницы сайта <http://demo.bitrixsoft.com/> будет добавлен код вида:

```
<IMG style="width:0px; height:0px; border: 0px" src="http://demo.bitirxsoft.com/bitrix/spread.php? s=QkIUUkIYX1NNX0dVRVNUX0IEATk5NDgBMTE5MTU5NDU0OAEvAQECQkIUUkI YX1NNX0xBU1RfVkJTSVQBMTAuMTAuMjAwNiAxODoyOTowOAEExMTkxNTk0NTQ4AS8BAQ I%3D& k=9e8de698c64709edf2e76202279ce889"></IMG>
```

В результате посетителям сайта <http://demo.bitrixsoft.com/> будет выполняться показ всплывающих окон сайта с доменным именем [demo.bitirxsoft.com](http://demo.bitirxsoft.com):



**⚠ Примечание:** Чтобы избежать проблем в случае, когда доменные имена различных сайтов частично совпадают, следует использовать индекс сортировки сайтов. Тогда при выборе сайта по доменному имени произойдет сравнение индекса сортировки: будет использован сайт с меньшим значением индекса.

Если значение индекса сортировки совпадает, то проверка будет производиться по длине доменного имени.



- **Папка сайта** – задается папка, в которой расположено информационное содержимое сайта, его разделы и страницы.

Указанное значение используется как опорный уровень для построения логической и физической структуры в модуле управления структурой.

- **Сортировка** - указывается численный параметр, определяющий порядок сайта в общем списке.

Но не только. В ряде случаев при добавлении второго сайта могут не применяться указанные ему шаблоны. Эта проблема вызвана особым механизмом выбора сайтов. Так, при содержании в имени второго сайта (к примеру, test.site.org) имени первого сайта (site.org), и значения сортировки второго сайта больше, чем первого, происходит применение шаблона ко второму сайту от первого. При использовании таких имен необходимо указывать значение сортировки у второго сайта меньше, чем у первого.

В секции **Параметры** производится задание языковых настроек сайта и, кроме того, задаются следующие значения:

- **Название сайта** – произвольное название сайта, которое может быть использовано в почтовых шаблонах. Если значение не задается в параметрах сайта, будет использовано значение одноименного параметра из настроек главного модуля.
- **URL сервера** – заданный адрес сервера будет использован при формировании почтовых сообщений на основе шаблонов. Задание адреса производится без `http://`. В случае, если значение не задается в настройках сайта, будет использовано одноименное значение из настроек главного модуля.
- **E-mail адрес по умолчанию** – задается электронный адрес, который будет использован в качестве значения макроса **DEFAULT\_EMAIL\_FROM** при формировании почтовых сообщений для каждого сервера. В случае, если данное поле не заполнено для сайта, в шаблоне будет использован почтовый адрес, указанный в настройках главного модуля.
- **Путь к корневой папке веб-сервера для данного сайта** – указывается полный путь к папке, на которую настроена переменная **DOCUMENT\_ROOT** для данного сайта. Для упрощения вставки полного пути в данное поле, можно воспользоваться ссылкой «*вставить текущий*». В таком случае будет автоматически подставлен полный путь к файлам текущего сайта на сервере.

 **Обратите внимание:**

*Использовать данную возможность вставки текущего пути следует, только находясь непосредственно на сайте, для которого производится вставка значения.*

*Например, находясь в административном разделе при заходе с сайта `site1.ru/bitrix/admin` будет подставлено значение пути именно для этого сайта:*



- /home/public\_html/site1/

Чтобы установить текущее значение для site2.ru необходимо зайти в административную часть site2.ru/bitrix/admin.

Тогда будет подставлено значение пути:

- /home/public\_html/site2/

При создании нового сайта дополнительно предлагается опция по созданию почтовых шаблонов. Доступны следующие варианты:

- **Не создавать** - шаблоны для сайта не будут созданы;
- **Привязать к имеющимся шаблонам** - предлагается возможность привязать сайт к существующим шаблонам для одного из сайтов;
- **Скопировать из шаблонов сайта** - шаблоны сайта будут скопированы для указанного сайта из соответствующих шаблонов для выбранного сайта.

**Обратите внимание:**

*Выбор опции по созданию или привязке почтовых шаблонов доступен только при создании нового сайта.*

Создать почтовые шаблоны:

- не создавать
- привязать к имеющимся шаблонам сайта  
[ru] Русский
- скопировать из шаблонов сайта  
[ru] Русский

## Настройки сайта и настройки языков

В Административном разделе системы настройки языков и сайтов выполняются раздельно.

Несмотря на то, что многосайтовая конфигурация часто используется для представления языковых копий одного ресурса, настройки языков и языковых свойств сайтов выполняют разные задачи.

**Примечание:** Настройки языков предназначены для задания параметров языка интерфейса в Административном разделе.

Настройка осуществляется в разделе [Настройки > Настройки продукта > Языки интерфейса](#).

## Языки

Рабочий стол > Настройки > Настройки продукта > Языки интерфейса

ID	Акт.	Сорт.	Название	По умолчан.
ru	Да	100	Russian	Да
en	Да	150	English	Нет

Выбрано: 2 | Отмечено: 0

Для всех | | - действия - | Применить

Параметры языков влияют на отображение информации в Административном разделе сайта. Так, например, формат даты, заданный в настройках языка, будет определять формат даты при показе записей в административном интерфейсе.

**Параметры языка системы**

*ID:	ru
Активен:	<input checked="" type="checkbox"/>
*Название:	Russian
По умолчанию:	<input checked="" type="checkbox"/>
*Сортировка:	100
*Формат даты: (например: DD.MM.YYYY)	DD.MM.YYYY
*Формат даты и времени: (например: DD.MM.YYYY HH:MI:SS)	DD.MM.YYYY HH:MI:SS
*Кодировка:	windows-1251
Направление текста:	Слева направо

**Сохранить** **Применить** **Отменить**

ID	Название	Акт.	Сорт.	Дата изменения	Статус	Блокировка
860	Бесплатная гарнитура для всех покупателей LG	Да	500	30.01.2006 18:48:37	Published	



Параметр **Направление текста** также влияет только на отображение административного раздела сайта.

**⚠ Примечание:** количество языков интерфейса никак не влияет на количество сайтов в системе.

Для каждого сайта можно задать определенные языковые настройки. Это делается в Административном разделе сайта на странице настроек параметров сайта ([Настройки > Настройки продукта > Сайты > Список сайтов](#)) в разделе **Параметры**. Так в настройках каждого сайта можно выполнить привязку к определенному языку интерфейса, формат даты, формат даты и времени (при совместном отображении), кодировку.

Параметры:	
*Язык:	[ru] Russian
*Формат даты: (например: DD.MM.YYYY)	DD-MM-YYYY
*Формат даты и времени: (например: DD.MM.YYYY HH:MI:SS)	DD.MM.YYYY HH:MI:SS
*Кодировка:	windows-1251

Заданные языковые параметры будут использованы для показа записей в публичной части сайта. Например, формат даты будет использован при показе даты новостей, а кодировка может быть использована в коде шаблонов сайта.

18-01-2006

#### [Бесплатная гарнитура для всех покупателей LG](#)

Всем нашим покупателям, приобретающим мобильные телефоны LG с 20 по 31 января - подарок!

За выбор языка интерфейса в кодах сайта отвечает функция.

```
<meta http-equiv="Content-Type" content="text/html; charset=<?= LANG_CHARSET;?>" />
```

\*Внешний вид шаблона сайта (рабочую область заменить #WORK\_AREA#):

```
<?if(!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=<?echo LANG_CHARSET;?>">
<META NAME="ROBOTS" content="ALL">
```

**⚠ Примечание:** Выбор языка интерфейса определяет, например, язык сообщений публичных компонентов и сообщений об ошибке, которые выдаются в публичной части сайта.



## Как система различает сайты

В плане многосайтности при создании сайтов в системе «1С-Битрикс: Управление сайтом» нас интересуют поля: **Доменное имя** и **Папка сайта**. Именно они определяют, какой из сайтов откроет система по запросу пользователя. Эти поля вы найдете в форме создания (редактирования) сайта на странице [Настройки > Настройки продукта > Сайты > Список сайтов](#).

Рабочий стол > Настройки > Настройки продукта > Сайты > Список сайтов

Список сайтов | Добавить сайт | Копировать сайт | Удалить сайт

Параметры

Параметры сайта

\*ID: ru  
Активен:   
\*Название: Демо-сайт

Параметры для определения сайта в публичном разделе:

По умолчанию:

Доменное имя:  
(список доменных имен, каждое в новой строке)

\*Папка сайта: /

\*Сортировка: 100

## Доменное имя

По получении запроса от пользователя система сначала проверяет текущий домен: из настроек всех сайтов выбираются домены и сопоставляются с доменом, на котором находится пользователь. При этом сравнивается только правая часть до точки (т.е. все поддомены автоматически относятся к этому домену). Например, если в настройках указано: example.com, а пользователь открыл www.example.com, то условие будет считаться выполненным. Но если он откроет my-example.com - это уже другой домен, его надо отдельно указывать в поле **Доменное имя**.

**⚠ Примечание:** Важно не указывать в списке доменов сайты, которые не работают на данном экземпляре продукта. Указанный неправильно или несуществующий домен может замедлить работу системы. К тому же это фактически не позволит перенести данные в сайты, работающие не на общем экземпляре продукта.

Доменное имя желательно указывать без **www**. Можно перечислить в этом поле с новой строки любое число доменных имен, по которым вы хотите, чтобы отвечал сайт. Все домены третьего или более низких уровней продукт будет считать принадлежащими



данному сайту и будет открывать сайт #1 как по имени [www.site1.com](http://www.site1.com) так же, как и без [www](http://www).

### Папка сайта

Если доменные имена не указаны или на разных сайтах указаны одинаковые домены, то определение происходит по полю **Папка сайта**. Обратите внимание, что здесь указывается папка относительно корня сайта (т.е. путь в URL), а не путь в файловой системе на сервере.

Важно иметь в виду, что значения, указанные в поле **Доменные имена** используется продуктом для распространения в указанные домены информации о пользователях по технологии **UserMultiSiteTransfer** (перенос пользователей в многосайтовой системе). Желательно указывать полный список доменов, по которым может ответить сайт.

**Пример:** В настройках сайтов в поле **Папка сайта** для первого указанна папка - */*, другого - */ru*. При открытии страницы [example.com/forum/messages/](http://example.com/forum/messages/) попадаем на первый сайт, при открытии [example.com/ru/forum/messages/](http://example.com/ru/forum/messages/) - на второй. При этом папка **bitrix** (содержащая ядро продукта) лежит в корне, никуда не копируется и никакие другие настройки на сервере не делаются.

При создании сайтов по многосайтовости на одном домене:

- поле **Путь к корневой папке веб-сервера** для этого сайта должно оставаться пустым,
- в полях **Папка сайта** должны быть указаны разные папки.

При создании сайтов по многосайтовости на разных доменах:

- в поле **Путь к корневой папке веб-сервера** для этого сайта должны быть указаны разные пути.
- поле **Папка сайта** должна быть указана корневая папка «*/*» для обоих сайтов.

 **Важно знать:** Для определения текущего сайта не используется порт, т.е. нельзя настроить многосайтовость на одном домене и разных портах.

### Многосайтовость на одном домене

Принципиальная необходимость для многосайтовости на одном домене - разделение сайтов по подкаталогам, так, чтобы структура файлов не пересекалась. Каждый сайт должен быть размещен в отдельном подкаталоге внутри корневого каталога.

При этом возможно как создание равнозначных папок в структуре корневого каталога, так и создание папок второго сайта внутри директории первого сайта. То есть, допустимы комбинации как вида:

- /www/s1/ - первый сайт,
- /www/s2/ - второй сайт.

Так и комбинации вида:

- /www/s1/ - первый сайт,
- /www/s1/s2 – второй сайт.

В учебном курсе рассматривается вариант с сайтами расположеными на одном уровне. При размещении сайтов на разных уровнях вложенности действия аналогичны, просто меняются пути до второго сайта.

Система поставляется настроенной на многосайтность на одном домене. Дополнительной настройки в файле **httpd.conf** веб-сервера Apache не требуются, достаточно создать нужные папки и заполнить правильно поля в настройках сайтов.

### Настройка на многосайтность на одном домене

При настройке многосайтности на одном домене мы располагаем одним веб-сервером Apache, **DocumentRoot** которого настроен на каталог /home/www/allsites/.

 **Примечание:** Путь к корню сайта в файловой системе сервера задается в настройках веб-сервера, например:

- для Apache - в файле **httpd.conf** параметр **DocumentRoot**;
- для IIS - в свойствах сайта, закладка *Home Directory > Local Path*.

Установим программный продукт «1С-Битрикс: Управление сайтом» в этот каталог.

Каждый сайт в первом способе конфигурации должен быть размещен в отдельном подкаталоге внутри единого каталога, например:

- /home/www/allsites/s1/
- /home/www/allsites/s2/

Имена каталогов **s1**, **s2** можно выбирать любыми, например, **shop** и **company**, или **en** и **de** соответственно. Возможен так же вариант, когда один из сайтов располагается в корневом каталоге (например, /home/www/allsites/), а второй сайт в подкаталоге (например, /home/www/allsites/s2/).

### Конфигурирование сайтов

Настройка сайтов выполняется в Административном разделе системы ([Настройки > Настройки продукта > Сайты > Список сайтов](#)).

Выбираем "Изменить" параметры сайта №1 (<http://www.site1.com>) и указываем в них:

- Название: site1
- Доменное имя: оставить пустым
- Папка сайта: /s1/
- Название сайта: Корпоративный сайт компании "*Название компании*"
- URL сервера: www.site1.com
- Путь к корневой папке веб-сервера для этого сайта: оставить пустым

Важно иметь в виду, что значения, указанные в поле **Доменное имя** используется продуктом для распространения в указанные домены информации о посетителях по технологии *переноса посетителей* (см. [Технология переноса посетителей между сайтами](#)). Поэтому крайне желательно указывать полный список доменов, по которым может ответить сайт.

В параметре **Папка сайта** необходимо указывать путь относительно корня к каталогу в котором расположена *публичная часть* сайта. А **Путь к корневой папке веб-сервера для этого сайта** не используется в данном способе настройки многосайтовости и должен быть пустым для всех сайтов.

Аналогично настроим параметры сайта №2 (<http://www.site2.com>):

- Название: site2
- Доменное имя: оставить пустым
- Папка сайта: /s2/
- Название сайта: Интернет-магазин компании "*Название компании*"
- URL сервера: www.site2.com
- Путь к корневой папке веб-сервера для этого сайта: оставить пустым

 **Примечание:** Многосайтовость на одном домене на вебсервере IIS реализуется подобно тому, как это делается для сервера Apache.

## Многосайтовость на разных доменах

Для работы многосайтовости на разных доменах нам потребуется произвести настройку программного продукта. Настройку веб-сервера Apache, как и с случае с многосайтовостью на одном домене, должна произвести хостинговая компания.

Будем использовать для примера конфигурацию из двух сайтов:

- [www.site1.com](http://www.site1.com) - корпоративный сайт компании
- [www.site2.com](http://www.site2.com) - интернет-магазин компании



## Настройка многосайтности на разных доменах

Каждый сайт надо разместить в соответствующем каталоге, например:

- /home/www/site1/
- /home/www/site2/

## Установка продукта и настройка символьных ссылок

Продукт устанавливается в один из сайтов. Чтобы ядро могло работать для обоих сайтов, необходимо создать символьные ссылки для сайта, в котором нет установленного ядра. Ссылки потребуются для папок /bitrix и /upload.

**⚠ Примечание:** Есть возможность простого копирования указанных папок из первого сайта во второй. При таком копировании получится две копии ядра, которые работают с одной базой данных. Такой вариант будет работать, но есть два отрицательных момента: технический и юридический. Техническая проблема заключается в том, что после обновления одного из ядер обновится база данных и второй сайт перестанет работать. Юридическая проблема заключается в том, что, копирование ядра противоречит лицензии на продукт.

**⚠ Примечание:** Технически возможно (но не рекомендуется) копирование указанных папок в некоторую внешнюю папку, на которую для всех сайтов настраиваются символические ссылки.

**Символьная ссылка:** (также симлинк от англ. *Symbolic link*, символическая ссылка)

Специальный файл, для которого в файловой системе не хранится никакой информации, кроме одной текстовой строки. Эта строка трактуется как путь к файлу, который должен быть открыт при попытке обратиться к данной ссылке.

Практически символьные ссылки используются для более удобной организации структуры файлов на компьютере, так как позволяют одному файлу или каталогу иметь несколько имён и свободны от некоторых ограничений, присущих жёстким ссылкам (последние действуют только в пределах одного раздела и не могут ссылаться на каталоги).

Ссылки можно создать двумя способами. Первый - классический, который рекомендовался компанией с самого начала. Второй - более поздний, считается более "красивым и изящным". В нем отсутствует шаг создания отдельной папки и переноса в нее ядра системы.

**Первый вариант** (зеленым цветом приведены примеры для установки на UNIX системы):



1. установите программный продукт "1С-Битрикс: Управление сайтом" сначала в каталог первого сайта `/home/www/site1/`
2. создайте каталог `/home/www/shared/`, в котором будут располагаться общие для всех сайтов файлы:

```
mkdir /home/www/shared
```

3. перенесите весь каталог `/home/www/site1/bitrix/` в `/home/www/shared/bitrix/`:  
`mv /home/www/site1/bitrix /home/www/shared/bitrix`
4. перенесите весь каталог `/home/www/site1/upload/` в `/home/www/shared/upload/`:  
`mv /home/www/site1/upload /home/www/shared/upload`
5. создайте символическую связь для каталога `/bitrix/` в каждом из сайтов:
  - `ln -s /home/www/shared/bitrix /home/www/site1/`
  - `ln -s /home/www/shared/upload /home/www/site1/`
  - `ln -s /home/www/shared/bitrix /home/www/site2/`
  - `ln -s /home/www/shared/upload /home/www/site2/`
6. убедитесь, что веб-сервер (Apache, IIS) имеет право на запись в каталог `/home/www/shared/` (это необходимо будет для работы системы обновлений и загрузки графических файлов)
7. разместите публичную часть второго сайта в каталог `/home/www/site2/`

**⚠ Примечание:** Для создания символьных связей в Windows необходимо воспользоваться дополнительными программами, например, Far Manager или Junction от Sysinternals.

**⚠ Важно!** Файловая система FAT32 не поддерживает создание символьных ссылок.

При настройке многосайтовой конфигурации на UNIX, можно воспользоваться программным методом создания символьных ссылок:

```
<?
symlink("/virt/homes/forinsured/bitrix", "/virt/homes/forinsured/htdocs/bitrix");
symlink("/virt/homes/forinsured/upload", "/virt/homes/forinsured/htdocs/upload");
?>
```

**⚠ Примечание:** В ряде случаев, например если web сервер работает в chroot, необходимо делать относительные ссылки.

**Пример:**

`/var/www/s1` - первый сайт



/var/www/s2 - второй сайт

/var/www/shared - папка с ядром системы

Задаем в /var/www/s1 и создаем ссылки:

```
In -s ../shared/bitrix bitrix  
In -s ../shared/upload upload
```

Переходим в /var/www/s2 и выполняем те же команды.

**Второй вариант.** В этом варианте символьные ссылки создаются непосредственно в папке второго сайта.

1. Установите программный продукт «1С-Битрикс: Управление сайтом» сначала в каталог первого сайта /home/www/site1/
2. Создайте в корневой папке второго сайта (/home/www/site2/) скрипт, например, под именем **symlink.php**:

```
<html>  
<head><title>Создание ссылок на папки bitrix и upload</title></head>  
<body>  
<?  
error_reporting(E_ALL & ~E_NOTICE);  
@ini_set("display_errors",1);  
if  
    $path = rtrim($_POST['path'], "/\\");  
else  
    $path = '..site2/www';  
if  
{  
    if  
        $full_path = $path;  
    else  
        $full_path = realpath($_SERVER['DOCUMENT_ROOT'].'/.$path);  
    if  
        file_exists($_SERVER['DOCUMENT_ROOT']."/bitrix")  
        $strError = "В текущей папке уже существует папка bitrix";  
    elseif  
        (is_dir($full_path))  
    {  
        if  
        {  
            if  
                (is_dir($full_path."/bitrix"))  
                (symlink($path."/bitrix", $_SERVER['DOCUMENT_ROOT']."/bitrix"))  
            {  
                if(symlink($path."/upload", $_SERVER['DOCUMENT_ROOT']."/upload"))  
                    echo "<font color=green>Символические ссылки успешно созданы</font>";  
            }  
        }  
    }  
}
```



```
        else
            $strError = 'Не удалось создать ссылку на папку upload, обратитесь к
администратору сервера';

    }
    else
        $strError = 'Не удалось создать ссылку на папку bitrix, обратитесь к
администратору сервера';

}
else
    $strError = 'Указанный путь не содержит папку bitrix';
}
else
    $strError = 'Неверно указан путь или ошибка прав доступа';
if
    echo '<font color=red>'.$strError.'</font><br>Исходный путь: '.$full_path;
?
<form
Путь к папке, содержащей папки bitrix и upload: <input name=path
value=<?=htmlspecialchars($path)?>><br>
<input type=submit value='Создать' name=create>
</form>
</body>
</html>
```

3. Запустите скрипт и укажите путь к корневой папке первого сайта, в нашем случае /home/www/site1/.
4. После того как символьные ссылки созданы наберите в адресной строке браузера site1/bitrix/admin. Откроется панель авторизации.
5. Вводите данные администратора, которые указывали при установке продукта на первый сайт и попадете в административную панель «1С-Битрикс: Управление сайтом».

После завершения работы скрипта наличие символьических ссылок на папки /bitrix и /upload в папке второго сайта можно проверить по появлению одноименных папок.

На этапе создания могут возникнуть проблемы:

- отсутствия прав на запись в текущую папку;
- ограничение безопасности (**open\_basedir**), которое не позволяет пользователям разделяемого хостинга обращаться к другим сайтам.



В случае возникновения проблем с этим скриптом следует обратиться за помощью к хостеру.

### Конфигурирование сайтов

Настройка сайтов выполняется в административном разделе системы на странице [Настройки > Настройки продукта > Сайты > Список сайтов](#).

В строке первого сайта ([www.site1.com](http://www.site1.com)), в колонке действий выбираем команду **Изменить** и указываем в них:

- Название: site1
- Доменное имя: www.site1.com
- Папка сайта: /
- Название сайта: Корпоративный сайт компании "*Название компании*"
- URL сервера: www.site1.com
- Путь к корневой папке веб-сервера для этого сайта: /home/www/site1/

Если DNS настроен таким образом что ваш сайт отвечает на адрес <http://site1.com>, то в поле **Доменное имя** желательно указывать без www. Можно перечислить в этом поле с новой строки любое число доменных имен, по которым вы хотите, чтобы отвечал сайт (или уже отвечает).

Важно иметь в виду, что значения, указанные в поле **Доменное имя**, используются продуктом для распространения в указанные домены информации о посетителях по технологии *переноса посетителей*. Поэтому крайне желательно указывать полный список доменов, по которым может ответить сайт.

Очень важно не указывать в списке доменов сайты, которые не работают на данном экземпляре продукта. Указанный неправильно или несуществующий домен может не только замедлить работу пользователей, но и фактически не позволит перенести данные в сайты, работающие не на общем экземпляре продукта.

Аналогично настроим параметры второго сайта ([www.site2.com/](http://www.site2.com/)):

- Название: site2
- Доменное имя: site2.com
- Папка сайта: /
- Название сайта: Интернет-магазин компании "*Название компании*"
- URL сервера: www.site2.com
- Путь к корневой папке веб-сервера для этого сайта: /home/www/site2/



Обратите внимание, что для двух сайтов в параметре **Папка сайта** указано одинаковое значение: "/". Это связано с тем, что сайты обслуживаются разными "виртуальными серверами" (в терминологии Apache) у которых для размещения файлов использован разный каталог.

Также необходимо обратить на параметр **Путь к корневой папке веб-сервера для этого сайта**. Для разных сайтов у него свое значение, взятое из параметра **DocumentRoot** настроек соответствующего "виртуального сервера" (см. ниже пример части файла **httpd.conf** настроек Apache).

**⚠ Примечание:** Необходимо иметь в виду, что при организации многосайтности по данному способу, вы можете использовать как виртуальные сервера одной установки Apache, так и просто разные установки Apache. Это справедливо для других веб-серверов: IIS, EServ и т.д.

**⚠ Важно:** при создании второго сайта необходимо скопировать с основного сайта или заново создать файлы `/.htaccess` и `/404.php`.

### Файл .access.php

Создайте файл **.access.php** с таким содержанием в корне второго сайта:

```
<? $PERM["/"]["*"]="R"; ?>
```

**⚠ Примечание:** Для данного способа организации многосайтности не требуется настраивать на индексной странице алгоритм выбора сайтов, как это делается при [Псевдомногосайтности](#), т.к. сайт будет однозначно определяться по полю **Доменное имя**.

Конфигурация готова к работе.

### Вход в систему

- Наберите в адресной строке браузера `http://site1/bitrix/admin` (или `http://site2/bitrix/admin`). Откроется панель авторизации.
- Введите данные администратора, которые указывали при установке продукта на первый сайт и попадете в административную панель «1С-Битрикс: Управление сайтом».

Поскольку ядро одно и база одна - административная панель для обоих сайтов будет одинаковая.

Технически система допускает создание произвольного числа сайтов работающих по этой схеме. В каждом из вновь создаваемых сайтов необходимо настроить веб-сервер и создать символическую ссылку. Юридически для создания каждого нового сайта (кроме первых двух) необходимо приобрести дополнительный купон.



**!** *Примечание:* при включенном **HTML кешировании** должна быть настроена серверная переменная **BX\_PERSONAL\_ROOT**, иначе ко второму сайту будет "примешиваться" кеш страниц первого сайта.

```
<VirtualHost *>
    DocumentRoot "/var/www/site1/"
    ServerName www.site1.ru
    SetEnv BX_PERSONAL_ROOT "/bitrix_personal"
</VirtualHost>
```

Для каждого сайта надо создать каталог **/bitrix\_personal** в корне для "не общих" данных (в том числе и кеша).

Эта настройка позволяет разделить по сайтам бывшие ранее общими такие вещи как:

```
root@slamp:~# ls -l /var/www2/sites/s5/bitrix_personal/
итого 12
drwxrwxr-x 4 www-data www-data 2048 2007-12-12 15:44 cache
drwxrwxr-x 3 www-data www-data 2048 2007-12-12 10:46 managed_cache
drwxrwxr-x 3 www-data www-data 2048 2007-12-13 13:02 html_pages
drwxrwxr-x 2 www-data www-data 2048 2007-12-12 10:45 php_interface
drwxrwxr-x 3 www-data www-data 2048 2007-12-12 12:32 stack_cache
drwxrwxr-x 5 www-data www-data 2048 2007-12-12 10:51 templates
```

Далее необходимо в новой папке создать символьную ссылку на **php\_interface** в старой папке.

Кроме этого необходимо обязательно создать папку **/templates** в каждом из каталогов **/bitrix\_personal/**. Эта папка должна содержать в себе или шаблоны сайтов или символьные ссылки на системную папку с шаблонами.

## Многосайтость на разных доменах на IIS

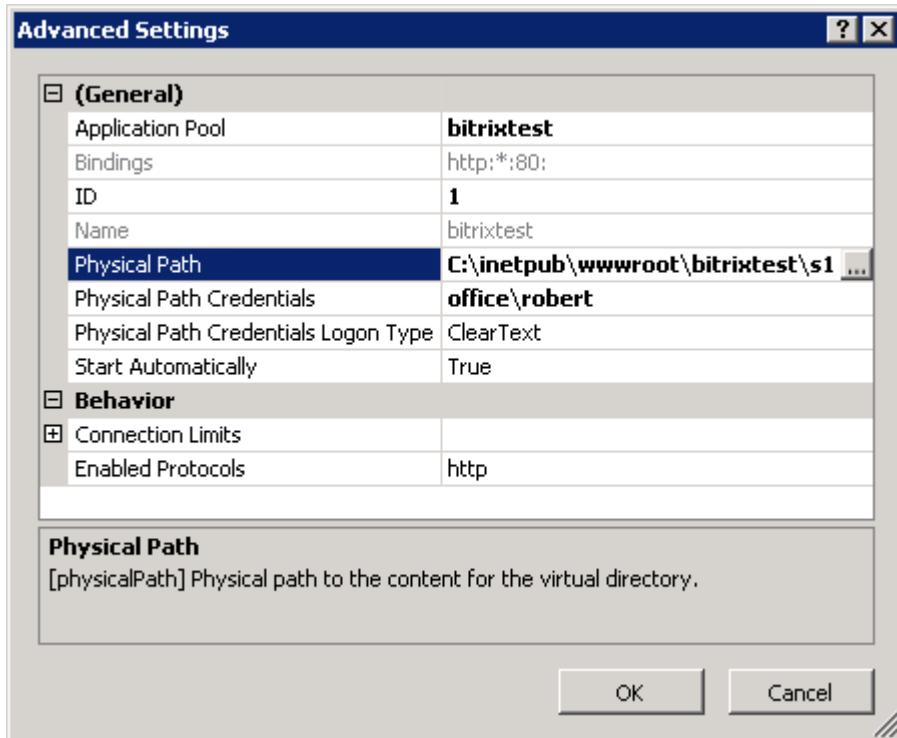
Описание настройки многосайтости на разных доменах на сервере IIS выполнены из расчета, что «1С-Битрикс: Управление сайтом» уже установлен.

### Изменение настроек основного сайта

- Создайте в папке, где установлен «1С-Битрикс: Управление сайтом» две папки, например **s1** и **s2**.
- Перенесите в любую из папок, пусть это будет **s1**, все системные папки и файлы из папки, где установлен «1С-Битрикс: Управление сайтом».



- Запустите **Internet Information Services Manager**.
- Перейдите в **IISM** на веб-сайт, в панели **Action** вызовите диалог **Advanced settings**.



- В строке **Physical Path** смените путь до новой папки с дистрибутивом «1С-Битрикс: Управление сайтом».
- Проверьте открытие сайта в браузере.

### Создание символьных ссылок

Необходимо создать символьные ссылки на системные папки **bitrix** и **upload** из папки **s2**. Это можно сделать с помощью специальной служебной утилиты [Junction](#) или с помощью файлового менеджера.

**⚠ Примечание:** Утилита **Junction** работает только на 32-хбитных системах.  
На 64-хбитных системах лучше пользоваться файловыми менеджерами.

- Чтобы создать или удалить точку соединения, запустите программу **Junction** командой:

```
junction [-d] <каталог с точкой соединения> [<объект соединения>]
```

Где в нашем случае каталог с точкой соединения – **s2**, объект соединения – указанные системные папки в **s1**.

- Чтобы удалить точку соединения, используйте параметр **-d** и укажите имя этой точки.



Использование файлового менеджера предпочтительнее по удобству. Рассмотрим создание файловых ссылок на примере **FAR**.

- Запустите файловый менеджер **FAR**.
- Откройте в одном окне папку **s1**, в другом - **s2**.
- С помощью команды **Alt+F6** создайте символьные ссылки на указанные системные папки в папке **s2**.

### Создание и настройка второго сайта в IIS

Создайте и настройте второй сайт в общем списке сайтов IIS, так как вы создавали основной сайт. При создании нового сайта учтите следующие отличия:

- Физический путь до папки должен указывать на папку **s2**.
- **Application pool** для этого сайта должен быть указан тот же, что и для основного, а не быть созданным заново.
- Для второго сайта должен быть назначен другой порт.

После создания сайта в IIS добавьте в папку **s2** файл **index.php**.

### Конфигурирование сайтов

Следующий шаг в настройке многосайтности на разных доменах на IIS – правильное конфигурирование созданных вами ранее сайтов в программном продукте. Настройка конфигурации сайтов идентична как для Apache, так и для IIS.

Настройка сайтов выполняется в административном разделе любого из сайтов, например в [www.site1.com/bitrix/](http://www.site1.com/bitrix/)

- Перейдите на страницу *Настройки > Настройки продукта > Сайты > Список сайтов*.
- Выберите команду **Изменить** в меню действий последовательно для первого и второго сайтов.
- Проверьте параметры сайтов.

Параметры первого сайта должны быть следующими:

- **Название:** site1
- **Доменное имя:** site1.com
- **Папка сайта:** /
- **URL сервера:** [www.site1.com](http://www.site1.com)
- **Название сайта:** site1
- **Путь к корневой папке веб-сервера для этого сайта:** /home/www/site1/

Параметры второго сайта должны быть следующими:

- **Название:** site2
- **Доменное имя:** site2.com
- **Папка сайта:** /
- **URL сервера:** www.site2.com
- **Название сайта:** site2
- **Путь к корневой папке веб-сервера для этого сайта:** /home/www/site2/

Обратите внимание, что для обоих сайтов папка сайтов указана одинаковая: "/". Это возможно потому, что сайты обслуживаются разными веб-серверами у которых разный каталог на диске использован для размещения файлов.

Доменное имя желательно указывать без **www**. Можно перечислить в этом поле с новой строки любое число доменных имен, по которым вы хотите, чтобы отвечал сайт.

Важно иметь в виду, что значения, указанные в поле **Доменные имена** используется продуктом для распространения в указанные домены информации о пользователях по технологии **UserMultiSiteTransfer**. Желательно указывать полный список доменов, по которым может ответить сайт.

 **Примечание:** Важно не указывать в списке доменов сайты, которые не работают на данном экземпляре продукта. Указанный неправильно или несуществующий домен может замедлить работу системы. К тому же это фактически не позволит перенести данные в сайты, работающие не на общем экземпляре продукта.

Конфигурация готова к работе.

### Псевдомногосайтость на разных доменах

Технически возможен (но не рекомендуется к использованию) вариант многосайтости, реализованный внешне как на разных доменах, но фактически обслуживаемый одним сервером Apache. Многосайтость реализуется в этом случае за счет кода индексной страницы основного сайта.

Особенностью этого метода является то, что в случае ошибочных указаний адресов возможны ситуации, когда, например, будет отображен контент одного сайта в шаблоне другого.



## Настройка сервера Apache

Настройка сервера производится аналогично настройкам многосайтовости на одном домене.

## Настройки сайта

Настройки сайтов производятся аналогично настройкам многосайтовости на одном домене с той лишь разницей что в поле **Доменное имя** каждого сайта указываются собственные доменные имена для каждого сайта.

## Настройка индексной страницы

Посетитель каждого из сайтов, заходя по адресу <http://www.site1.com> или <http://www.site2.com>, попадает фактически на страницу `/index.php`, лежащую в каталоге, указанном в параметре **DocumentRoot** настроек веб-сервера. В многосайтовой конфигурации роль этого файла немного меняется, и в нем необходимо разместить уже не содержимое **индексной страницы** корня одного из сайтов, а PHP код, осуществляющий выбор одного из сайтов в зависимости от текущего **доменного имени**.

При решении данной задачи могут быть использованы следующие функции класса **C MainPage**:

- **C MainPage::GetSiteByHost** - возвращает **ID** сайта, определяя его по текущему **доменному имени**.
- **C MainPage::GetSiteByAcceptLanguage** - возвращает **ID** сайта, определяя его по переменной **Accept-Language** в настройках браузера пользователя.
- **C MainPage::GetIncludeSitePage** - возвращает **абсолютный путь** на индексную страницу папки указанного сайта, для дальнейшего его подключения.
- **C MainPage::RedirectToSite** - перенаправляет на **индексную страницу** папки указанного сайта.

## Пример индексной страницы портала,

когда сайт определяется по текущему доменному имени:

```
<?
//      подключим      файл      с      классом      C MainPage
require($_SERVER['DOCUMENT_ROOT']."/bitrix/modules/main/include/mainpage.php");

//  получим  идентификатор  текущего  сайта  по  доменному  имени
$site_id = C MainPage::GetSiteByHost();

//  получим  абсолютный  путь  к  индексной  странице  папки  сайта
$page = C MainPage::GetIncludeSitePage($site_id);
```



```
// если сайт определен и определена индексная страница то
if(strlen($site_id)>0 && strlen($page)>0)
{
    // подключим страницу
    require_once($page);
}
else // иначе если сайт не определен то
{
    require($_SERVER['DOCUMENT_ROOT']."/bitrix/header.php");
    // далее можно разместить код который будет отображаться если
    // сайт ранее не был определен
?>

<?require($_SERVER['DOCUMENT_ROOT']."/bitrix/footer.php");
?>
?>
```

Этот пример кода определяет доменное имя, по которому пришел посетитель, используя функцию **CMainPage::GetSiteByHost**, сверяет это доменное имя с именами, указанными в настройках сайтов в поле **Доменное имя** для определения **ID** сайта и производит включение в тело документа индексной страницы из папки соответствующего сайта, используя функцию **CMainPage::GetIncludeSitePage**.

В нашем примере это будет означать, что посетителю, пришедшему по адресу <http://www.site1.com> прямо в теле текущей страницы без редиректа будет представлена страница `/s1/index.php`. А посетителю, пришедшему по адресу <http://www.site2.com> - страница `/s2/index.php`.

Использование приведенного алгоритма позволяет избежать редиректов для пользователей и поисковых роботов, обеспечивает удобную работу с многосайтовой конфигурацией. Данный алгоритм является рекомендуемым, но не единственным при работе с многосайтовой версией.

### Пример индексной страницы портала,

когда сайт определяется по установленным в браузере посетителя языкам:

```
<?
// подключим файл с классом CMainPage
require($_SERVER['DOCUMENT_ROOT']."/bitrix/modules/main/include/mainpage.php");

// получим идентификатор сайта по Accept-Language
$site_id = CMainPage::GetSiteByAcceptLanguage();
```

```

// если сайт определен, то
if(strlen($site_id)>0)
{
    // перенаправим на индексную страницу сайта
    CMainPage::RedirectToSite($site_id);
}
else // иначе если сайт не определен то
{
    require($_SERVER['DOCUMENT_ROOT']."/bitrix/header.php");
    // далее можно разместить код который будет отображаться если
    // сайт ранее не был определен
    ?>

    <?require($_SERVER['DOCUMENT_ROOT']."/bitrix/footer.php");
}
?>

```

В этом примере кода функция **CMainPage::GetSiteByAcceptLanguage** проверяет, какие языки установлены в настройках браузера посетителя, сравнивает с **ID** языка сайта и возвращает наиболее подходящий сайт.

После того, как сайт будет определен, функция **CMainPage::RedirectToSite** выполнит редирект (302 ответ веб-сервера) и переведет пользователя на *индексную страницу* папки указанного сайта, например, по адресу <http://www.site1.com/s1/> или <http://www.site2.com/s2/>.

### Пример индексной страницы портала,

когда один из сайтов расположен в корне, остальные - в папках, но при этом хосты у сайтов одинаковые:

```

<?
// подключим файл с классом CMainPage
require($_SERVER['DOCUMENT_ROOT']."/bitrix/modules/main/include/mainpage.php");

// получим идентификатор текущего сайта по доменному имени
$mainpage_siteid = CMainPage::GetSiteByHost();

//если текущим сайтом является s2, то получим абсолютный путь к индексной
//странице папки сайта
if ($mainpage_siteid != "s1" && $page = CMainPage::GetIncludeSitePage($mainpage_siteid)):
// подключим страницу
    require_once($page);

```

```
die();  
endif;  
  
require($_SERVER['DOCUMENT_ROOT']."/bitrix/header.php");  
// далее можно разместить обычный текст индексной страницы,  
//которая будет подключаться, если активен тот сайт, который в корне ?>  
  
<?require($_SERVER['DOCUMENT_ROOT']."/bitrix/footer.php");  
?>
```

В нашем примере если сайтом "по умолчанию" (из настроек сайтов) является **s2**, то его индексная страница и будет вызвана.

Если ваши сайты являются, по сути, разными международными языковыми зеркалами, то вы можете использовать вариант определения сайта по установленным в браузере посетителя языкам.

## Примеры настроек сервера Apache

Настройки веб-сервера Apache выполняет, как правило, техническая служба хостера. Приведенные ниже примеры настроек сервера Apache даны для ознакомления и понимания механизма настройки.

### Многосайтовость на одном домене

В конфигурационном файле **httpd.conf** веб-сервера Apache должна присутствовать примерно такая запись:

```
<VirtualHost *:80>  
    ServerAdmin admin@site1.com  
    DocumentRoot "/home/www/allsites/"  
    ServerName www.site1.com  
    ErrorLog logs/allsite.log  
    CustomLog logs/allsite.log common  
</VirtualHost>
```

Обратите внимание, что параметр **DocumentRoot** имеет значение `/home/www/allsites/` и явно указывает на каталог, в котором установлен продукт. Для двух сайтов параметр **DocumentRoot** будет иметь одно и тоже значение.

Строка **<VirtualHost \*:80>** указывает на то, что веб-сервер будет отвечать на любое доменное имя по любому IP адресу. Т.е. при соответствующей настройке DNS сервера, веб-сервер будет отвечать по любому из имен `www.site1.com` или `www.site2.com`.



## Многосайтовость на разных доменах

В конфигурационном файле **httpd.conf** веб-сервера Apache должны присутствовать две записи, каждая из которых описывает свой "виртуальный сервер" (в терминологии, принятой в Apache):

```
<VirtualHost *:80>
    ServerAdmin admin@site1.com
    DocumentRoot "/home/www/site1/"
    ServerName site1.com
    ServerAlias *.site1.com
    ErrorLog logs/site1.log
    CustomLog common
</VirtualHost>
```

```
<VirtualHost *:80>
    ServerAdmin admin@site2.com
    DocumentRoot "/home/www/site2/"
    ServerName site2.com
    ServerAlias *.site2.com
    ErrorLog logs/site2.log
    CustomLog common
</VirtualHost>
```

Обратите внимание, что параметр **DocumentRoot** для каждого сайта указывает в разный каталог на диске, в котором должен быть размещен соответствующий сайт.

Строки **<VirtualHost \*:80>** указывают на то, что веб-сервер будет отвечать на любом IP адресе, но переменная **ServerAlias** говорит о том, что каждый из сайтов будет отвечать только по определенному доменному имени.

Т.е. доменное имя `www.site1.com` будет обрабатываться одним веб-сервером Apache, который работает с каталогом `/home/www/site1/`, а `www.site2.com` - другим веб-сервером, работающим с каталогом `/home/www/site2/`.

Возможен так же вариант конфигурирования для разных IP адресов. Ниже приведен пример конфигурации Apache для двух разных IP адресов:

```
<VirtualHost 192.168.0.1:80>
    ServerAdmin admin@site1.com
    DocumentRoot "/home/www/site1/"
    ServerName site1.com
    ErrorLog logs/site1.log
    CustomLog common
</VirtualHost>
```

<i>Options &lt;/VirtualHost&gt;</i>	<i>+FollowSymLinks</i>
---	------------------------

<i>&lt;VirtualHost ServerAdmin admin@site2.com DocumentRoot "/home/www/site2/" ServerName site2.com ErrorLog logs/site2.log CustomLog logs/site2.log common Options +FollowSymLinks &lt;/VirtualHost&gt;</i>	<i>192.168.0.2:80&gt;</i>
--	---------------------------

В этом случае при соответствующей настройке DNS для разных доменных имен, каждый "виртуальный сервер" (в терминологии Apache) будет работать на отдельном IP адресе и отвечать только по определенному доменному имени.

**⚠ Примечание:** В силу некоторых причин, например: ограничения хостинга, Администратор сайта может не иметь доступа к файлу **httpd.conf**. В этом случае осуществить разделение информации для сайтов можно следующим образом:

- В корне второго сайта создать папку **/bitrix\_personal**
- В этой папке сделать две символьные ссылки на **/bitrix/php\_interface/** и **/bitrix/templates/** первого сайта.
- В файле **.htaccess** в корне каждого сайта в самом начале необходимо прописать:

<i>SetEnv BX_PERSONAL_ROOT "/bitrix_personal"</i>
---

## Выделение разделов сайта в поддомены

В некоторых случаях бывает необходимо организовать выделение некоторых разделов сайта в виде поддомена основного сайта. Например, организовать форум не по адресу <http://www.mysite.ru/forum/>, а по адресу <http://forum.mysite.ru>. Задача решается с использованием методов многосайтности на разных доменах. При этом сайт остается один и лицензия не нарушается.

**⚠ Примечание:** Необходимо помнить, что для такой настройки необходимо иметь зарегистрированный DNS третьего уровня.

- Создайте раздел, который будет выноситься в поддомен.

- Сконфигурируйте отдельный виртуальный сервер Apache для многосайтовости на разных доменах (см. [Примеры настроек сервера Apache](#)) с привязкой домена к созданной папке.
- Создайте в этой папке символьные ссылки на папки **bitrix** и **upload**.
- Создайте шаблон для сайта и примените его.

**⚠ Примечание:** Для задания разных шаблонов для разных папок в этом случае в настройках сайта нужно выбрать условие для отображения шаблона "выражение *php*" и вставить код:

```
$_SERVER['HTTP_HOST'] == 'site2'
```

где *site2.ru* - имя домена, для которого применяется собственный шаблон сайта.

**⚠ Примечание:** На таком сайте-поддомене нельзя использовать меню, созданные для основного сайта, так как меню основного сайта имеет пути относительно основного сайта. Необходимо либо использовать абсолютные пути в меню, либо создать собственное меню для сайта-поддомена.

## Работа с данными в многосайтовой конфигурации

Работа с системой в многосайтовой конфигурации имеет свои особенности. Это касается как настроек системы, учета статистики, так и работы с данными.

### Работа со структурой сайта

Для удобства работы с информационным наполнением сайтов файловая структура в модуле управления структурой представлена с разделением по сайтам. В разделе **Структура сайта** представлена структура сайтов в двух вариантах: как логическая структура разных сайтов и как физическая структура файлов и папок на сервере.

### Логическая структура

Логическая структура сайтов при этом разделяется следующим образом:



The screenshot shows the 1C-Bitrix CMS interface. On the left, there's a sidebar with icons for editing, creating, deleting, and managing content. Below it is a tree view of the site structure:

- Структура сайта
  - Русский
    - e-Learning
    - Блоги
    - Каталог
    - О компании
    - Партнеры
    - Персональный раздел
    - Поддержка
    - Поиск
  - English
    - About Us
    - Blogs
    - Catalog
    - e-Learning
    - Partners
    - Personal section
    - Search
    - Support
  - Файлы и папки

On the right, there's a table showing files and folders:

	Имя	Имя
		...
		Скидки и акции
		actions
		Фотогалерея
		gallery
		Меню типа «left»
		Меню типа «left»
		Контакты
		contacts.php
		О компании
		index.php

Логическая структура сайта создается на основе заголовков, которые задаются в диалоге управления свойствами папок.

The dialog box is titled "Свойства каталога". It contains a "Заголовок:" field with the value "О компании" highlighted by a red rectangle. Below it is a "Свойства папки" section with a table:

Код	Значение
Фото по разделам	about_company.jpg

Эти же заголовки используются при построении цепочки навигации по сайту:

A simple navigation bar with two items: "Главная" and "О компании" separated by a slash.

Название для файлов в логической структуре создается на базе заголовка страницы, который задается при редактировании страницы.

### Физическая структура

Физическая структура или структура файлов и папок будет представлена с разбивкой по сайтам (аналогично логической структуре) в случае использования многосайтности на разных доменах.

В случае использования многосайтности на одном домене структура обоих сайтов одновременно доступна при просмотре структуры файлов и папок, как показано на рисунке.

The screenshot shows a file structure viewer interface. On the left, there is a tree view of the logical structure:

- Структура сайта
  - + Русский
  - + English
  - Файлы и папки
    - + about
    - + admin
    - + bitrix
    - + blog
    - + catalog
    - + doc
    - + download
    - + en (highlighted with a red border)
    - + images
    - + img
    - + learning
    - + partners
    - + personal
    - + search
    - + support
    - + upload

On the right, there is a list view of the physical structure:

	Имя
	..
	actions
	anketa
	gallery
	news
	Меню типа «left»
	contacts.php
	contacts_inc.php
	index.php
	index_inc.php

Рамкой выделена папка, которая является корнем второго сайта.

**Совет.** Настраивая колонки для показа в списке, можно включить одновременное отображение физической и логической структуры. Рядом с реальным именем файла будет выводится его заголовок.

	Имя	Имя
<input type="checkbox"/>	..	..
<input type="checkbox"/>	Скидки и акции	actions
<input type="checkbox"/>	Фотогалерея	gallery
<input type="checkbox"/>	Меню типа «left»	Меню типа «left»
<input type="checkbox"/>	Контакты	contacts.php
<input type="checkbox"/>	О компании	index.php

## Какие объекты можно позиционировать по сайтам

Система позволяет осуществлять позиционирование информационного содержимого по сайтам.

Возможна привязка следующих объектов к сайтам:

### Модуль Блогов

- Группы блогов могут быть подключены к разным сайтам.

**Группа блогов**

**Параметры группы блогов**

ID: 2

\*Название группы: Авто и Мото

\*Сайт группы: [ru] Русский

**Сохранить      Применить      Отменить**

### Веб-формы

- Формы могут быть подключены к разным сайтам.

Сайт  [ru] Русский  
 [en] English

### Интернет-магазин

- Для различных сайтов могут быть настроены разные валюты (настройки модуля);

- Для различных сайтов могут быть настроены разные параметры напоминаний о неоплаченных заказах (настройки модуля);
- Для различных сайтов могут быть настроены разные параметры веса (настройки модуля);
- Для различных сайтов могут быть настроены группы, имеющие доступ к заказам сайта (настройки модуля);
- Заказы формируются с привязкой к сайту;
- Скидки на заказ привязываются к сайту;
- Службы доставки привязываются к сайтам;
- Платежные системы привязаны к сайтам;
- Типы плательщиков привязаны к сайтам;
- Налоги задаются с привязкой к сайту.

### Настройка прав на работу с заказами

Дополнительные параметры сайтов		
Сайт	Валюта	Группы, имеющие доступ к заказам сайта:
[ru] Русский	RUR (Рубль)	Зарегистрированные пользователи Партнеры Подписчики Редакторы сайта Администраторы техподдержки
[en] English	USD (Доллар США)	Зарегистрированные пользователи Партнеры Подписчики Редакторы сайта Администраторы техподдержки

### Информационные блоки

- Каждый информационный блок можно привязать к одному или нескольким сайтам.

### Валюта

- Курс валюты можно привязать к одному или нескольким сайтам.

### Опросы

- Группы опросов могут быть привязаны к одному или нескольким сайтам.

### Рассылки

- Категории рассылок привязываются к сайту.

### Поиск

- Переиндексация может быть выполнена по сайтам;



**Переиндексация**

**Параметры переиндексации**

Переиндексировать только измененные:

Максимальный размер индексируемого документа:  кб

Индексировать по шагам:

Шаг:  секунд

Сайт: [ru] Русский

Модуль: (все)

**Переиндексировать**    **Остановить**

- Создание Google Sitemap выполняется для каждого сайта;

**Google Sitemap**

**Параметры создания**

Сайт: [ru] Русский

Создавать по шагам:

Шаг:  секунд

**Создать**    **Остановить**    **Продолжить**

- Правила сортировки для результатов поиска задаются для каждого сайта.

**Правило**

**Параметры правила сортировки**

Сайт: [ru] Русский

Модуль: Статические файлы

## Почта

- Для почтового ящика выполняется привязка к сайту.



### Реклама

- Для контракта может быть установлена привязка к одному или нескольким сайтам;
- Для баннеров может быть выполнен таргетинг по сайтам, которые перечислены в контракте.

### Техподдержка

- SLA к одному или нескольким сайтам;
- Записи в справочнике могут быть привязаны к одному или нескольким сайтам;
- Обращения в техподдержку создаются с привязкой к сайту.

### Торговый каталог

- Скидки и купоны задаются с привязкой к сайту.

### Управление структурой

- Логическая и физическая структура может быть просмотрена по сайтам.



## Форум

- Для каждого форума может быть выполнена привязка к одному или нескольким сайтам и указан шаблон пути к сообщениям для соответствующих сайтов.

## Какие настройки модулей разделяются по сайтам

Некоторые модули дают возможность произвести раздельную настройку параметров для различных сайтов. Это может быть путь к публичным файлам или, как в случае с модулем Интернет-магазина, права на доступ к управлению заказами отдельных сайтов.

## Модуль блогов

- Пути к публичной части блогов

## Интернет магазин

- Валюта для сайта
- Права на заказы

## Управление структурой

- Можно индивидуально задавать следующие настройки для сайтов:
- типы меню
- количество дополнительных параметров меню
- типы свойств



**Настройки для сайтов**

Использовать индивидуальные настройки  для каждого сайта:

Настройки для сайта: **Русский**

Типы меню:	Тип	Название
	left	Левое меню
	top	Верхнее меню

Количество дополнительных параметров меню: **1**

Типы свойств:	Тип	Название
	description	Описание
	keywords	Ключевые слова
	title	Дополнительный
	adv_desired_target_keywords	Ключевые слова,
	work_pic	Фото по раздела
	NOT_SHOW_NAV_CHAIN	Не показывать на

### Социальная сеть

- Для каждого сайта могут быть заданы индивидуальные настройки функционала друзей, шаблонов для страниц и групп, собственные пути для персональных страниц.
- Для каждого сайта могут быть заданы индивидуальные настройки для форумов, блогов и фотогалерей пользователей.
- Для каждого сайта могут быть заданы индивидуальные настройки для форумов, блогов и фотогалерей групп.

### Какую статистику можно анализировать в разрезе по сайтам

В разрезе по сайтам можно просматривать следующие статистические данные:

- Сводная статистика
- Посещаемость: динамика
- Посещаемость: разделы и страницы
- Посещаемость: точки входа



- Посещаемость: точки выхода
- Пути по сайту
- Внимательность: длительность сессии
- Внимательность: активность
- Список событий
- Переходы с поисковиков: список переходов
- Переходы с поисковиков: по поисковикам
- Поисковые фразы: список фраз
- Поисковые фразы: внутренний поиск
- Индексация: хиты поисковиков
- Ссылающиеся сайты: сайты
- Ссылающиеся сайты: страницы
- Ссылающиеся сайты: переходы
- Посетители: список посетителей
- Посетители: сессии посетителей
- Посетители: хиты посетителей
- Посетители: стоп лист
- Кто на сайте

Разделение статистики по сайтам возможно с использованием различных условий фильтрации на страницах отчетов. Ниже приводятся примеры фильтров, которые используются на страницах различных отчетов.

Дополнительно

Период (DD.MM.YYYY):

Сайт:

(все)

[ru] Русский

[en] English

Найти Отменить



The screenshot shows the 'Additional' search dialog. In the 'Найти:' (Find) field, the text 'event1' is entered. In the 'Где произошло:' (Where it happened) dropdown, '(сайт)' (Site) is selected. Below the dropdown, a tooltip shows '(сайт)' highlighted. At the bottom left are 'Найти' (Find) and 'Отменить' (Cancel) buttons.

The screenshot shows the 'Additional' search dialog. In the 'ID:' field, there is no text. In the 'Сайт:' (Site) dropdown, '[ru] Русский' (Russian) is selected. Below the dropdown, a tooltip shows '[ru] Русский' and '[en] English'. At the bottom left are 'Найти' (Find) and 'Отменить' (Cancel) buttons.

**⚠️ Обратите внимание!** Разделение прав на просмотр статистики разных сайтов в продукте невозможно. Пользователи группы, которые имеют доступ к просмотру статистики, смогут просматривать статистику по всем сайтам.

## Типовые вопросы возникающие при работе с многосайтовой конфигурацией

При использовании многосайтости возникают некоторые технические моменты, разрешение которых не всегда видно сразу. В этой главе собраны часто встречающиеся проблемы, на которые просим обратить ваше внимание.

### Удаление сайта и связанных объектов

При удалении сайта из системы часто возникает ситуация, при которой сайт не может быть удален из-за наличия в системе привязанных к данному сайту объектов.

Поэтому перед удалением сайта необходимо предварительно удалить или отвязать все объекты от данного сайта.

На данный момент процедура удаления сайта не совсем проста для обычного пользователя, т.к. требуется самостоятельно просмотреть все объекты, которые могут быть привязаны к сайту.



Это могут быть заказы, информационные блоки, форумы и прочие объекты. При этом объекты могут иметь множественную привязку по сайтам, поэтому требуется просто отвязать их от сайта без удаления.

Помощь в поиске объектов могут оказать сообщения об ошибке, которые выдаются на экран в процессе удаления сайта.

В коде ошибке может быть указан класс или метод, по которому можно определить, в каком модуле производить поиск объекта для удаления или отвязывания. Приведенная ниже ошибка показывает, что сайт невозможно удалить, т.к. есть тип плательщика, привязанный к данному сайту.

ID	Акт.	Сортировка	Название	Папка	По умолчанию
ru	Да	100	Русский	/	Да
en	Да	150	English	/en/	Нет

### Как закрыть только один из сайтов для посещения пользователей

В настройках системы есть возможность закрыть сайт для публичного посещения, например, на время каких-либо технических работ. Это делается на странице настроек главного модуля ([Настройки > Настройки продукта > Настройки модулей > Главный модуль](#)):

## Служебные процедуры

The screenshot shows a dialog box titled 'Временное закрытие публичной части сайта' (Temporary closure of the public part of the site). It has two tabs: 'Публичная часть' (Public part) and 'Контроллер' (Controller), with 'Публичная часть' selected. Below the tabs, a green message says 'Доступ к публичной части открыт' (Access to the public part is open). At the bottom is a button labeled 'Закрыть доступ для посетителей' (Close access for visitors).

Однако, указанная возможность позволяет выполнить действие только для всех сайтов системы одновременно. Для того чтобы закрыть отдельный сайт в рамках многосайтовой конфигурации, необходимо разместить специальный программный код в файле:

/bitrix/php\_interface/ru/init.php

```
include($_SERVER["DOCUMENT_ROOT"]."/ru/underconstruction.html");
die();
```

где, **ru** - идентификатор сайта, который нужно закрыть.

В файле **underconstruction.html** следует поместить информационное сообщение, которое будет показано посетителям вместо стандартных страниц сайта.

## Компонент для переключения сайтов

В составе дистрибутива продукта имеется компонент **Выбор сайта** (**bitrix:main.site.selector**), который выполняет функцию переключения между сайтами в режиме многосайтности. Компонент внедряется в шаблон сайта. С его помощью посетитель сайта осуществит переход на корневую папку каждого сайта.

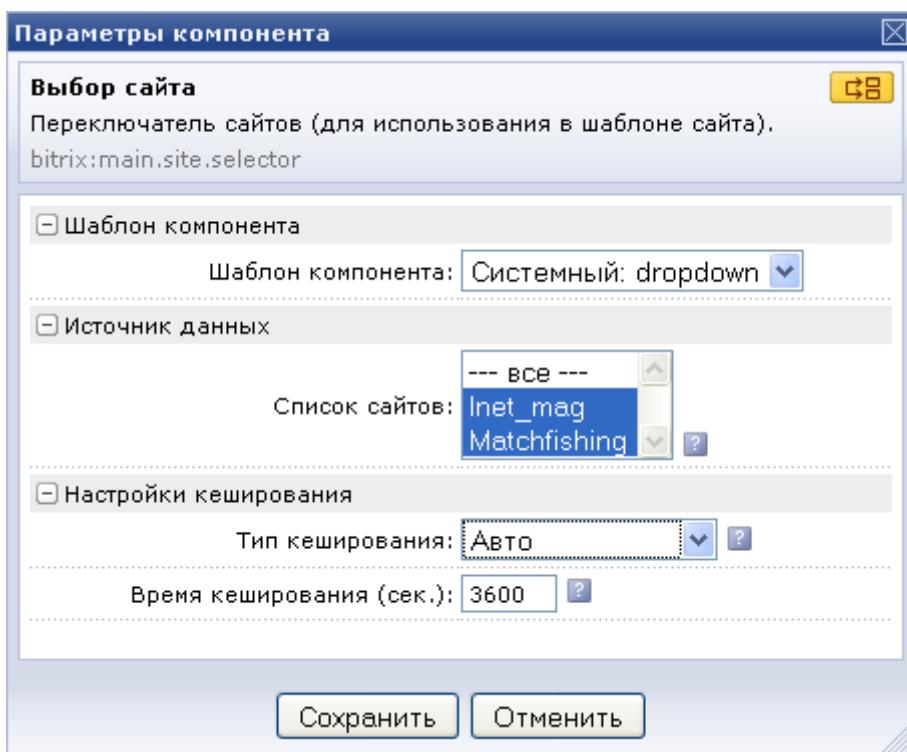
Компонент расположен:

- В рамках визуального редактора в группе компонентов [Служебные > Навигация](#);
- В рамках файловой структуры в папке \www\bitrix\components\bitrix\.

Настройка вывода данных осуществляется типовым способом: копированием шаблона компонента с последующей модификацией шаблона. Пример оформления:

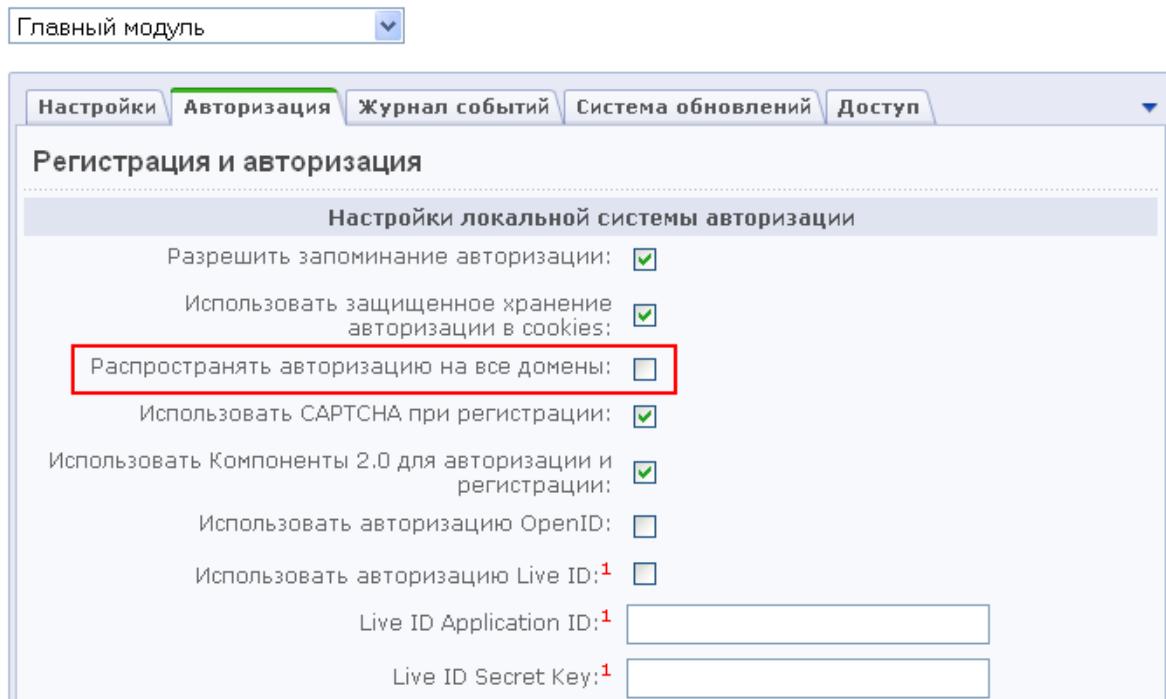
Русский > English >

Настройка параметров компонента так же выполняется стандартно для компонентов системы через диалог **Настройка параметров**:



## Сохранение авторизации пользователя при переходе по различным сайтам

Для удобства работы с различными сайтами в многосайтовой версии, в настройках **Главного модуля** можно использовать специальную опцию для сохранения авторизации при переходе по различным доменам системы.





## Глава 11. Бизнес-процессы для разработчика

### Цитатник веб-разработчиков.

**Максим Месилов:** Не нужно стесняться задавать вопросы которые вертятся на языке в момент проектирования.

**Бизнес-процесс** - это процесс обработки документа, для которого задана одна точка входа и несколько точек выхода и последовательность действий (шагов, этапов, функций), совершаемых в заданном порядке и в определенных условиях.

Модуль **Бизнес-процессы**:

- предназначен для организации как последовательной обработки элементов инфоблоков в виде отдельного последовательного процесса, так и для создания статусных схем сложных процессов с неопределенным периодом действия.
- это инструмент, с помощью которого другие части продукта (модули, компоненты и т.п.) могут предоставлять пользователям возможность определять, выполнять и управлять бизнес-процессами (рабочими потоками). Он дает возможность визуально программировать произвольную функциональность, а так же запускать и управлять подобными программами. Здесь под словом "произвольная" понимается возможность создания пользовательских действий, которые расширяют стандартную функциональность.
- предусматривает универсальный механизм программирования бизнес-процессов, доступный не программисту. Такой универсальный механизм реализован за счет визуального программирования по технологии drag&drop (бери и тащи), понятной и известной любому пользователю компьютера. Шаблон бизнес процессов создается в особом визуальном конструкторе.

**⚠ Важно!** Модуль Бизнес-процессов работает только на **PHP 5** и выше. Перед его установкой проверьте соответствие вашей установки этому требованию.

Каждый экземпляр бизнес-процесса (далее – БП) представляет собой программу. Входящие параметры БП являются параметрами, с которыми запускается программа. Переменные БП являются переменными программы.

Соответственно время жизни параметров и переменных БП ограничено временем жизни самого БП. Чтобы переменные, параметры или любые другие значения БП были доступны после завершения БП и/или вне БП, необходимо сохранить их куда-либо в постоянную память. Наиболее удобно сохранять их в документ БП.

Бизнес-процесс одновременно может работать только в одной копии. Если была попытка запустить вторую копию до завершения работы первой, то возникает ошибка: **Бизнес-**



**процесс заблокирован другим процессом..** Как правило к этому приводит ошибка и некорректное завершение другого БП.

Каждый экземпляр БП работает над документом. Документы физически могут представлять собой различные сущности и определять различный функционал работы. Например, компонент Бизнес-процесс, пример которого находится в меню **Сервисы "1С-Битрикс: Корпоративный портал"**, в качестве документа использует в конечном итоге элементы инфоблока.

БП с помощью действий предоставляет возможность манипулировать документом, над которым он работает. Например, изменить документ или опубликовать его.

**⚠ Внимание!** Внедрение бизнес-процессов требует особой квалификации специалистов. Эти специалисты должны понимать предметную суть бизнес-процессов в компании и уметь выражать эту суть логическими схемами.

При автоматизации бизнес-процессов не надо бросаться кодировать пока не будет нарисована схема бизнес-процесса и не будет понимания как эта схема будет реализовываться.

## Шаблон бизнес-процесса

**Шаблон бизнес-процесса** – это схема (программа), в которой задана одна точка входа, последовательность действий (шагов, этапов, функций), совершаемых в заданном порядке и направленных на достижение некоторой цели, а так же одна или несколько точек выхода, определяющих завершение выполнения.

Шаблон создается в специальном модуле **Дизайнер бизнес-процессов** с помощью визуального конструктора. Конструктор позволяет перетаскивать действия из панели инструментов на Основную рабочую область конструктора, создавая шаблон бизнес-процесса визуальным образом. Шаблон создается в виде блок-схемы, которая наглядно отображает логику работы бизнес-процесса.

В дистрибутив заложен ряд встроенных действий (англ. activities), которые могут быть использованы для выполнения работ общего назначения. Таких действий - несколько десятков. Есть действия, допускающие использование собственного скрипта или php-кода. Кроме того, при необходимости есть возможность создавать собственные действия и подключать их к бизнес-процессу.

Созданный шаблон бизнес-процесса может быть выполнен автоматически или вручную, в зависимости от настроек. В любое время может одновременно выполняться несколько экземпляров бизнес-процесса и система занимается управлением выполнения этих экземпляров, сохраняя и восстанавливая их состояние по требованию. Ведется лог каждого экземпляра бизнес-процесса для дальнейшего анализа работы схемы и корректировки ее под новые условия.

Во внутренней архитектуре бизнес-процесса шаблон бизнес-процесса представляется в виде многомерного массива, содержащего иерархию действий и значения их свойств. Именно с таким представлением шаблона бизнес-процесса работает API модуля **Бизнес-процессы**.

Пример простого массива, представляющего шаблон бизнес-процесса:

```
array(
    array(
        "Type" => "SequentialWorkflowActivity",
        "Name" => "SequentialWorkflowActivity1",
        "Properties" => array(),
        "Children" => array(
            array(
                "Type" => "SetFieldActivity",
                "Name" => "SetFieldActivity1",
                "Properties" => array("Field" => "XML_ID", "Value" => "В рассмотрении"),
            ),
            array(
                "Type" => "IfElseActivity",
                "Name" => "IfElseActivity1",
                "Properties" => array(),
                "Children" => array(
                    array(
                        "Type" => "IfElseBranchActivity",
                        "Name" => "IfElseBranchActivity1",
                        "Properties" => array("FieldCondition" => array("CREATED_BY", "=", 1)),
                    ),
                    array(
                        "Type" => "SetFieldActivity",
                        "Name" => "SetFieldActivity2",
                        "Properties" => array("Field" => "XML_ID", "Value" => "Принят"),
                    ),
                ),
            ),
        ),
        "array(
            "Type" => "IfElseBranchActivity",
            "Name" => "IfElseBranchActivity2",
            "Properties" => array(),
        ),
    ),
)
```



## Бизнес-процесс

**Бизнес-процесс** – это конкретный экземпляр шаблона бизнес-процесса. Он создается по требованию, запускается с точки входа и заканчивает работу по достижении одной из точек выхода. Для одного шаблона может быть одновременно запущено неограниченное число бизнес-процессов.

После завершения бизнес-процесс перестает существовать. Но его статус сохраняется и доступен для использования.

Бизнес-процесс всегда выполняется над определенным документом, определяющимся его кодом. При этом документ может не иметь физического представления (т.е. быть виртуальным). Бизнес-процесс может быть настроен на автоматический запуск при добавлении или изменении документа.

Каждый бизнес-процесс уникально идентифицирован с помощью его кода, который может быть назначен исполняющей средой или же задан программистом. По коду можно обратиться к определенному бизнес-процессу.

Бизнес-процессу может быть отправлено событие с помощью методов исполняющей среды. Сообщение отправляется бизнес-процессу по его уникальному коду.

При запуске бизнес-процесс может принимать на вход значения параметров, список которых задается при конструировании шаблона бизнес-процесса. Например, это могут быть идентификатор заказа или код текущего пользователя. Любое действие бизнес-процесса будет иметь доступ к этим параметрам.

Бизнес-процесс может не выполняться постоянно. Например, если в бизнес-процессе встречается действие **CBPDelayActivity** (реализует ожидание, откладывая выполнение на определенный срок), то бизнес-процесс входит в состояние ожидания, сохраняется в базе данных и удаляется из памяти. По истечении заданного времени бизнес-процесс считывается из базы данных, восстанавливается в памяти и продолжает выполняться с места остановки.

### **Совет**

Чтобы для отладки увидеть все внутренности, например, чтобы понять - какими переменными можно оперировать, можно вставить в интересующее место в схеме php код:

```
echo "<pre>", print_r( $_REQUEST ), "</pre>";
echo "----<br />";
echo "<pre>", print_r( $this ), "</pre>";
exit;
```

Затем запустить бизнес процесс на выполнение. Например,

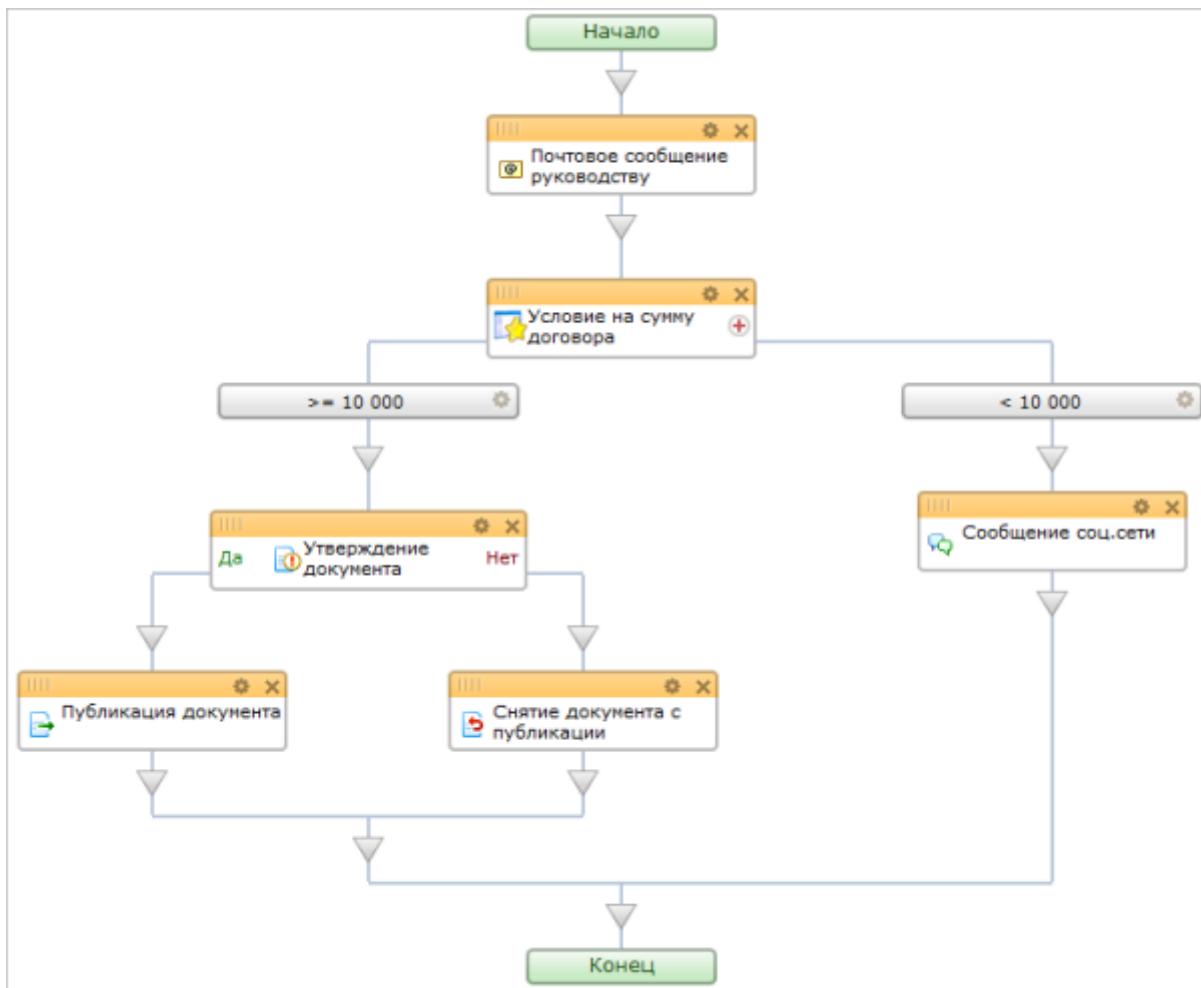
создать новый элемент в инфоблоке или отредактировать. И спокойно сидеть разбираться.

## Типы бизнес-процессов

При помощи модуля Бизнес-процессы могут быть описаны два типа бизнес-процессов: последовательный и со статусами.

### Последовательный бизнес-процесс

**Последовательный бизнес-процесс** — действия выполняются одно за другим от точки входа до точки выхода.



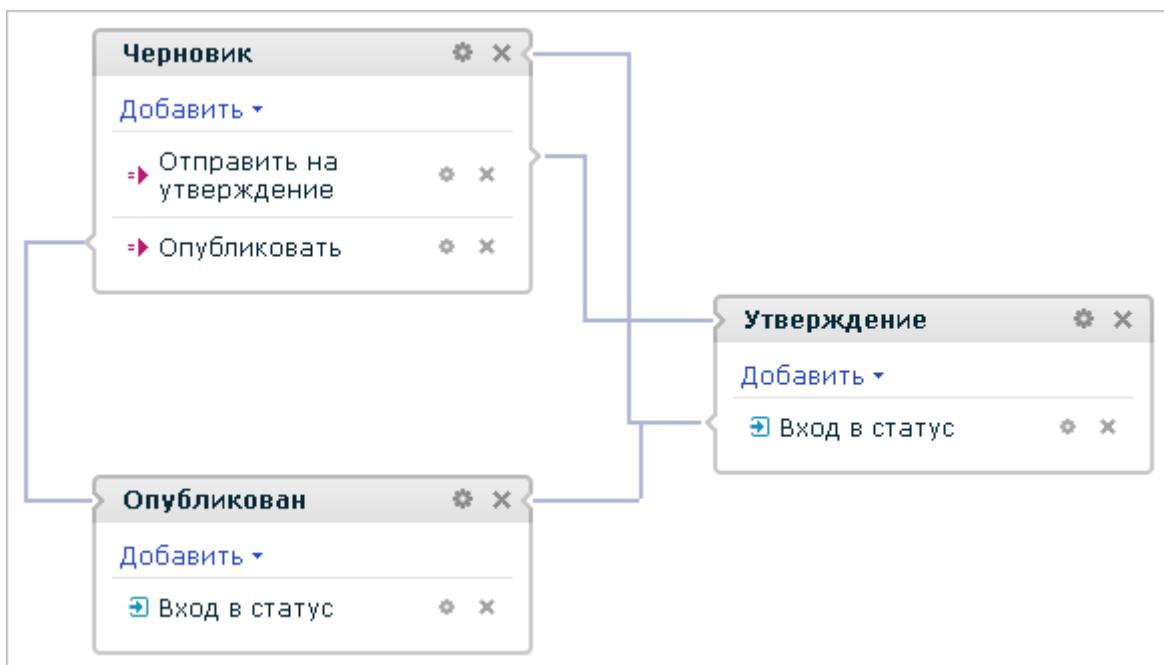
Последовательный бизнес-процесс представляет собой бизнес-процесс как набор шагов, которые следует выполнять по порядку до тех пор, пока они все не завершатся. Он похож на обычную блок-схему, описывающую алгоритм решения задачи.

Последовательный бизнес-процесс начинает работу, выполняя находящееся в нем первое дочернее действие, и продолжается до тех пор, пока не выполнит все остальные дочерние действия.

### Бизнес-процесс со статусами

**Бизнес-процесс со статусами** (так же известный как **машина состояний** или **автомат на состояниях**) — начала и конца не имеет, в процессе работы происходит переход из одного состояния в другое.

Бизнес-процесс со статусами представляет собой набор состояний, переходов и действий. Одно состояние обозначается как начальное состояние. В процессе выполнения процесса переходит из одного состояния в другое, основываясь на событиях.



При переходе бизнес-процесса в любой из статусов выполняется последовательный подпроцесс для инициализации этого статуса. После чего бизнес-процесс становится в ожидание одного из определенных в данном статусе событий. Список возможных событий и их названия могут быть получены для отображения в пользовательском интерфейсе. На каждое событие может быть получен список групп пользователей, которые имеют право отправлять данное событие.

При возникновении события выполняется последовательный подпроцесс, соответствующий принятому событию (связанный с ним).

Среди действий подпроцесса может быть действие по установке нового статуса. В этом случае бизнес-процесс переводится в новый статус. Если такого действия нет, то текущий статус считается конечным и бизнес-процесс завершается.

Если для данного статуса определен последовательный подпроцесс для финализации этого статуса, то он выполняется непосредственно перед выходом из данного статуса (перед переходом в другой статус).

### **Выбор типа бизнес-процесса**

Практически любая возникающая задача может быть решена как с помощью последовательного бизнес-процесса, так и с помощью бизнес-процесса со статусами. Но выбор не соответствующего типа бизнес-процесса существенно усложнит шаблон бизнес-процесса.

Тип **последовательный бизнес-процесс** выбирается в том случае, если просто необходимо выполнить определенную последовательность действий.

Тип **бизнес-процесс со статусами** выбирается в том случае, если бизнес-процесс (документ) может находиться в разных состояниях, переходы между которыми осуществляются по определенным правилам. При этом появляется возможность автоматически контролировать права на доступ к документу в разных состояниях и на выполнение переходов между состояниями. Для каждого состояния может быть задан набор событий, при возникновении которых выполняются соответствующие подпроцессы.

### **Действия**

Все, что происходит в бизнес-процессе — это действия. Сам бизнес-процесс представляет собой составное действие, которое позволяет определять внутри себя дочерние действия. Каждое действие в рамках бизнес-процесса должно иметь уникальное имя.

**Действие** — это класс, который наследуется от абстрактного класса **CBPActivity** или его потомков. Название класса должно начинаться с подстроки "CBP" и может состоять из латинских букв и цифр.

Пример:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();

class CBPMyActivity1
    extends CBPActivity
{
    ...
}

?>
>
```

Непосредственно от класса **CBPActivity** наследуются действия, которые не могут содержать внутри себя другие действия. Этот класс определяет набор базовых методов, которые необходимы любому действию. Некоторые методы, определенные в классе **CBPActivity** могут или должны быть переопределены в классе-наследнике.

В дистрибутиве по умолчанию есть несколько десятков действий (**Activity**) для создания бизнес-процессов. Тем не менее, иногда возникает потребность в создании собственного действия.

Рассмотрим несколько примеров созданий собственных действий.

### Основные стандартные действия

Действие	Описание
<a href="#">CBPActivity</a>	Абстрактный базовый класс всех действий.
CBPCompositeActivity	Абстрактный базовый класс составных действий, т.е. действий, которые могут содержать в себе дочерние действия.
CBPCodeActivity	Запускает на выполнение произвольный PHP-код.
CBPSetVariableActivity	Устанавливает значения переменных бизнес-процесса.
CBPDelayActivity	Реализует ожидание, откладывая выполнение на определенный срок.
CBPHandleExternalEventActivity	Реализует действие, которое ожидает внешнее событие. Бизнес-процесс останавливается до получения данного внешнего события.
CBPEventDrivenActivity	Служит контейнером для действий, выполнение которых происходит по событию.
CBPIfElseActivity	Реализует функционал условия.
CBPIfElseBranchActivity	Реализует функционал ветки условия.
CBPWhileActivity	Реализует функционал цикла.
CBPListenActivity	Реализует ожидание одного из нескольких возможных событий. Когда одно из событий происходит,

остальные перестают ожидать событий и отменяются.

CBPParallelActivity	Параллельно запускает набор дочерних действий.
CBPSequenceActivity	Последовательно запускает набор дочерних действий.
CBPSequentialWorkflowActivity	Представляет собой бизнес-процесс, который выполняет действия последовательно (последовательный бизнес-процесс).
CBPSetStateActivity	Устанавливает статус в бизнес-процессе со статусами.
CBPStateMachineWorkflowActivity	Представляет собой бизнес-процесс со статусами.

Любое действие наследуется от базового класса действий или одного из его наследников.

### Свойства действий

Действие может иметь свойства, значения которых настраиваются при добавлении действия в шаблон бизнес-процесса. Значениями свойств могут быть как константы, так и ссылки на свойства других действий бизнес-процесса.

Свойства действия описываются в конструкторе класса действия определением массива в члене класса **arProperties**:

```
public function __construct($name)
{
    parent::__construct($name);
    $this->arProperties = array("Title" => "", "MyProperty" => "");
}
```

Ключами в массиве определения свойств являются названия свойств, а значениями – значения по умолчанию.

При выполнении действия свойства доступны как члены класса:



```
$this->MyProperty
```

Входные параметры (свойства) бизнес-процесса доступны как свойства корневого действия бизнес-процесса. Любое действие бизнес-процесса может обратиться к входным параметрам бизнес-процесса.

Например, если бизнес-процесс был запущен с помощью кода:

```
// Код шаблона бизнес-процесса
$workflowTemplateId = 12;

// Бизнес-процесс запускается для документа - элемента инфоблока с кодом 358
$documentId = array("iblock", "CIBlockDocument", 358);

// Входные параметры бизнес-процесса
$arParameters = array("MyProperty" => "Красный");

$runtime = CBPRuntime::GetRuntime();
$wi = $runtime->CreateWorkflow($workflowTemplateId, $documentId, $arParameters);
$wi->Start();
```

то в любом действии этого бизнес-процесса значение параметра можно будет получить с помощью кода:

```
$rootActivity = $this->GetRootActivity();
if ($rootActivity->IsPropertyExists("MyProperty"))
    $val = $rootActivity->MyProperty;
// $val == "Красный"
```

Свойства действий описываются разработчиком при написании кода действия. Входящие параметры бизнес-процесса (они же свойства корневого действия бизнес-процесса) описываются пользователем при создании шаблона бизнес-процесса.

Значениями свойств могут быть как константы, так и ссылки на свойства других действий бизнес-процесса при условии, что эти действия выполнились раньше.

Чтобы в момент выполнения бизнес-процесса значением свойства одного действия являлось значение свойства выполненного выше другого действия, необходимо при создании шаблона бизнес-процесса в качестве значения свойства первого действия задать массив вида:

```
array("название действия, на свойство которого ссылаются", "название свойства")
```



В случае простых типов данных свойств в качестве значения свойства первого действия можно задать строку вида:

```
"{=название_действия, название_свойства}"
```

Чтобы сослаться на входной параметр (свойство) бизнес-процесса, которое доступно как свойство корневого действия бизнес-процесса, следует в качестве названия действия использовать слово **Template**:

```
array("Template", "название_свойства") "{=Template, название_свойства}"
```

Вообще в качестве названия действия можно использовать слова:

- **Document** – для обращения к произвольному полю документа, над которым запущен бизнес-процесс;
- **Template** – для обращения к входному параметру (свойству) бизнес-процесса (корневого действия);
- **Variable** – для обращения к переменной бизнес-процесса;
- **User** – для получения кода текущего пользователя (в качестве названия свойства должно быть указано "ID");
- **System** – обращение к системным переменным, в настоящее время доступно только свойство **Now** – текущая дата в формате сайта;
- любое другое имя – обращение к свойству действия с этим именем.

Например, если во время разработки в качестве значения свойства установить строку:

```
"Документ [url={=Template:PathTemplate}]{=Document:NAME}[/url] был одобрен"
```

и при этом рабочий поток запустится с входящим параметром **PathTemplate** равным **file.php** над документом с названием **План счетов**, то при выполнении действия значением свойства будет строка:

```
"Документ [url=file.php]План счетов[/url] был одобрен"
```

## Составные действия

Составные действия наследуются от абстрактного класса **CBPCompositeActivity**, который, в свою очередь, наследуется от класса **CBPActivity**. Класс **CBPCompositeActivity** обеспечивает поддержку возможности включать внутрь действия дочерние действия. Например, составным действием является стандартное действие **CBPParallelActivity** (параллельное выполнение), которое содержит в себе дочерние действия, соответствующие веткам параллельного выполнения.

Класс **CBPCompositeActivity** содержит член **arActivities**, с помощью которого можно обращаться к дочерним действиям.



Например, при запуске действия необходимо запустить первое дочернее действие и дождаться его завершения. Для этого можно использовать следующий код:

```
class CBPMyActivity
    extends CBPCompositeActivity // наследуем, так как составное действие
    implements IBPEventActivity // обработка события завершения дочернего
//действия
{
    // Исполняемый метод действия
    public function Execute()
    {
        // Возьмем первое дочернее действие
        $activity = $this->arActivities[0];
        // Подпишемся на событие изменения статуса дочернего действия
        // (завершение)
        $activity->AddStatusChangeHandler(self::ClosedEvent, $this);
        // Отправим дочернее действие исполняющей среде на выполнение
        $this->workflow->ExecuteActivity($activity);
        // Вернем указание исполняющей среде, что действие еще выполняется
        return CBPActivityExecutionStatus::Executing;
    }

    // Обработчик события изменения статуса интерфейса IBPEventActivity
    // Параметром передается действие, изменившее статус
    protected function OnEvent(CBPActivity $sender)
    {
        // Отпишемся от события изменения статуса дочернего действия
        // (завершения)
        $sender->RemoveStatusChangeHandler(self::ClosedEvent, $this);
        // Дочернее действие завершено, выполняем другой необходимый нам код
        // Например завершаем действие
        $this->workflow->CloseActivity($this);
    }
}
```



## Создание собственных действий

Пользовательские действия создаются в папке /bitrix/activities/custom относительно корня сайта. Каждое действие располагается в отдельной папке. Название папки действия должно совпадать с именем класса действия, но без первых символов "СВР". Кроме того имя папки должно быть записано строчными буквами (в нижнем регистре).

В папке действия должен располагаться файл класса действия. Название файла класса действия должно совпадать с названием папки действия и иметь расширение **php**. Кроме того в папке действия могут располагаться другие необходимые действию файлы. Например, файл с описанием действия, файлы с локализацией действия, изображения, файлы с ресурсами и т.п.

Файл с описанием действия располагается в папке действия и имеет имя **.description.php**. В этом файле содержится описание действия, которое необходимо для корректной работы системы. В файле описания действия должен содержаться код типа:

```
<if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arActivityDescription = array(
    "NAME" => GetMessage("MYACTIVITY_DESCR_NAME"),
    "DESCRIPTION" => GetMessage("MYACTIVITY_DESCR_DESCR"),
    "TYPE" => "activity",
    "CLASS" => "MyActivity",
    "JSCLASS" => "BizProcActivity",
    "CATEGORY" => array(
        "ID" => "other",
    ),
);
?>
```

Здесь определен тип действия в элементе **TYPE**, который имеет два возможных значения: **activity** для действий и **condition** для условий. Кроме того, задаются название и описание действия, Java-скриптовый класс для отрисовки в визуальном редакторе, категория и т.п.

В подпапке lang папки действия располагаются файлы с локализацией фраз действия на различные языки.

Файл с классом действия имеет вид типа:

```
<if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
```



```
class CBPMyActivity
    extends CBPAcitivity
{
    public function __construct($name)
    {
        parent::__construct($name);
        // Определим свойство действия MyText
        // Оно может быть задано в визуальном редакторе при
        // помещении действия в шаблон бизнес-процесса
        $this->arProperties = array("Title" => "", "MyText" => "");
    }

    // Исполняющийся метод действия
    public function Execute()
    {
        // Суть действия – запись значения свойства в файл
        if (strlen($this->MyText) > 0)
        {
            $f = fopen($_SERVER["DOCUMENT_ROOT"]."/dump.txt", "a");
            fwrite($f, $this->MyText);
            fclose($f);
        }

        // Возвратим исполняющей системе указание, что действие завершено
        return CBPAcitivityExecutionStatus::Closed;
    }

    // Статический метод возвращает HTML-код диалога настройки
    // свойств действия в визуальном редакторе. Если действие не имеет
    // свойств, то этот метод не нужен
    public static function GetPropertiesDialog($documentType, $activityName,
        $arWorkflowTemplate, $arWorkflowParameters, $arWorkflowVariables,
        $arCurrentValues = null, $formName = "")
    {
        $runtime = CBPRuntime::GetRuntime();

        if (!is_array($arWorkflowParameters))
```



```
$arWorkflowParameters = array();
if (!is_array($arWorkflowVariables))
    $arWorkflowVariables = array();

// Если диалог открывается первый раз, то подгружаем значение
// свойства, которое было сохранено в шаблоне бизнес-процесса
if (!is_array($arCurrentValues))
{
    $arCurrentValues = array("my_text" => "");

$arCurrentActivity= &CBPWorkflowTemplateLoader::FindActivityByName(
    $arWorkflowTemplate,
    $activityName
);
if (is_array($arCurrentActivity["Properties"]))
    $arCurrentValues["my_text "] =
$arCurrentActivity["Properties"]["MyText"];
}

// Код, формирующий диалог, расположен в отдельном файле
// properties_dialog.php в папке действия.
// Возвращаем этот код.
return $runtime->ExecuteResourceFile(
    __FILE__,
    "properties_dialog.php",
    array(
        "arCurrentValues" => $arCurrentValues,
        "formName" => $formName,
    )
);
}

// Статический метод получает введенные в диалоге настройки свойств
// значения и сохраняет их в шаблоне бизнес-процесса. Если действие не
// имеет свойств, то этот метод не нужен.
public static function GetPropertiesDialogValues($documentType, $activityName,
    &$arWorkflowTemplate, &$arWorkflowParameters, &$arWorkflowVariables,
```



```
$arCurrentValues, &$arErrors)
{
    $arErrors = array();

    $runtime = CBPRuntime::GetRuntime();

    if (strlen($arCurrentValues["my_text "]) <= 0)
    {
        $arErrors[] = array(
            "code" => "emptyCode",
            "message" => GetMessage("MYACTIVITY_EMPTY_TEXT"),
        );
        return false;
    }

    $arProperties = array("MyText" => $arCurrentValues["my_text "]);

    $arCurrentActivity = &CBPWorkflowTemplateLoader::FindActivityByName(
        $arWorkflowTemplate,
        $activityName
    );
    $arCurrentActivity["Properties"] = $arProperties;

    return true;
}
?>
```

Код в файле **properties\_dialog.php**, формирующий диалог настройки свойств действия в визуальном редакторе, может выглядеть примерно так:

```
<?if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
?>
<tr>
    <td align="right" width="40%"><span style="color:#FF0000;">*</span> :</td>
    <td width="60%">
        <textarea name="my_text" id="id_my_text" rows="5" cols="40"><?=htmlspecialchars($arCurrentValues["my_text"]) ?></textarea>
```



```
<input type="button" value="..." onclick="BPAShowSelector('id_my_text',  
'string');">  
</td>  
</tr>
```

Пользователь может ввести в поле **my\_text** явное значение или выбрать одно из значений с помощью диалога, открывающегося по кнопке . Во втором случае пользователь может установить, что значением свойства будет являться значение свойства корневого действия, которое задается как входящий параметр при запуске бизнес-процесса.

Далее рассмотрим несколько примеров создания собственных действий.

### Действие Запись в лог

Создадим действие, которое будет писать в лог БП или текстовый файл, произвольные значения из шаблона бизнес-процесса. Пусть имя у действия будет **write2logactivity**.

Создайте структуру файлов будущего **Activity**. Папка с Activity должна быть расположена в `\bitrix\activities\custom\`. Название папки с действием должно совпадать с именем файла, в котором находится класс с Activity, в данном случае папка должна называться `/write2logactivity`.

Структура в общем виде подобна структуре компонентов:

- **.description.php** - описание будущего действия;
- **properties\_dialog.php** - форма настроек действия;
- **write2logactivity.php** – код действия;
- **icon.gif** – иконка, которая будет отображать действие в общем списке;
- `/lang.ru` – содержит папки с языковыми сообщениями. Папки должны называться в соответствии с идентификаторами языков. Например, `ru`, `en`. В каждой папке содержатся файлы с фразами, одноименные соответствующим файлам действия:
  - **.description.php** - описание языковых сообщений;
  - **write2logactivity.php** – сами языковые сообщения.

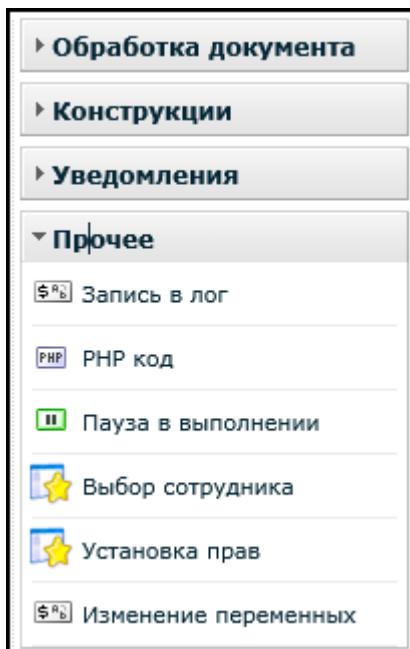
Задайте описание будущего действия в файле **.description.php**:

```
<?  
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();  
  
$arActivityDescription = array(  
    "NAME" => GetMessage("BPDDA_DESCR_NAME"),  
    "DESCRIPTION" => GetMessage("BPDDA_DESCR_DESCR"),
```



```
"TYPE" => "activity", // Тип - действие
"CLASS" => "Write2LogActivity", //Класс с Activity
"JSCLASS" => "BizProcActivity", //Стандартная JS библиотека, которая будет
рисовать
Activity "CATEGORY" => array(
    "ID" => "other", // Activity будет располагаться в категории "Прочее"
),
);
?>
```

В визуальном редакторе получаем иконку своего действия:



Запрограммируйте форму настроек действия в файле **properties\_dialog.php**:

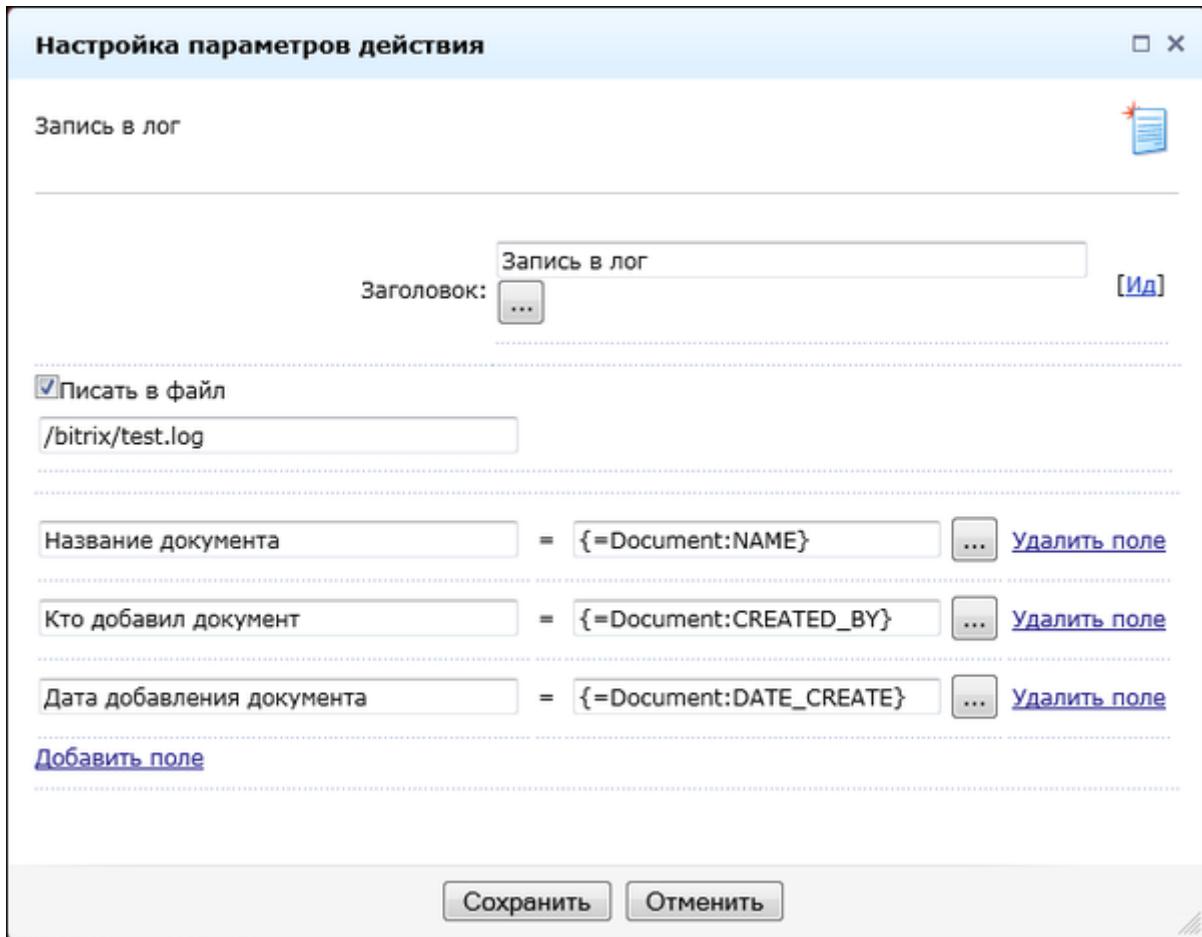
```
<script>
<?

foreach ($arCurrentValues["MapFields"] as $fieldKey => $documentFieldValue)
{
    ?>
    AddCondition('AddCondition("", "");<?
}
?>
var check = '<?= $arCurrentValues["write2file"];?>';
```



```
if (check == "Y")
{
    document.getElementById("write2file").checked = "checked";
    Write2File();
}
</script>
```

В результате на экране получим:



В зависимости от положения опции **Писать файл**, вывод мы получим либо в логе БП:



The screenshot shows a list of business process logs. The search bar at the top indicates no results found ('(нет)'). The table has columns: Date, Name, and Note. The data is as follows:

Дата	Название	Примечание
25.03.2011 02:14:23	Запись в лог	Дата добавления документа = 25.03.2011 02:14:23
25.03.2011 02:14:23	Запись в лог	Кто добавил документ = user_1
25.03.2011 02:14:23	Запись в лог	Название документа = Заявка

Всего: 3

либо в файле:

```

1 =====
2 dsfsdfsdf = 103092
3 выфывыфв = фвыфв
4
5
6
7 =====
8 dsfsdfsdf = 103093
9 выфывыфв = фвыфв
10
11 |
12
13 =====
14 Название документа = Заявка
15 Кто добавил документ = user_1
16 Дата добавления документа = 25.03.2011 02:17:28
17

```

Задайте код собственно действия в файле **write2logactivity.php**.

```

<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!=true)die();

class CBPWrite2LogActivity
    extends CBPActivity
{
    public function __construct($name)
    {
        parent::__construct($name);
        $this->arProperties = array(
            "Title" => "",
            "MapFields" => null,

```



```
"write2file"=> null,
"path2file"=> null
);
}

public function Execute()
{
if (is_array($this->arProperties["MapFields"])) && count($this->arProperties["MapFields"]))
{
$printVal = $this->__get("MapFields");
$bFirst = true;

foreach($printVal as $key => $val)
{
if ($this->arProperties["write2file"] == "Y")
{

    $f = fopen ($_SERVER["DOCUMENT_ROOT"].$this->arProperties["path2file"],
"a+");
if ($bFirst)
    fwrite ($f, print_r ("$key." = ".$val."\n",true));
fclose($f);
$bFirst = false;
}
else
    $this->WriteToTrackingService($key." = ".$val);
}
}

return CBPActivityExecutionStatus::Closed;
}

public static function GetPropertiesDialog($documentType, $activityName,
$arWorkflowTemplate, $arWorkflowParameters, $arWorkflowVariables, $arCurrentValues =
null, $formName = "")
{
```



```
$runtime = CBPRuntime::GetRuntime();  
  
if (!is_array($arCurrentValues))  
{  
    $arCurrentValues = array();  
  
    $arCurrentActivity  
    =&CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate, $activityName);  
  
    if (is_array($arCurrentActivity["Properties"]))  
        && array_key_exists("MapFields", $arCurrentActivity["Properties"])  
        && is_array($arCurrentActivity["Properties"]["MapFields"]))  
    {  
        foreach ($arCurrentActivity["Properties"]["MapFields"] as $k => $v)  
        {  
            $arCurrentValues["MapFields"][$k] = $v;  
        }  
        $arCurrentValues["write2file"] = $arCurrentActivity["Properties"]["write2file"];  
        $arCurrentValues["path2file"] = $arCurrentActivity["Properties"]["path2file"];  
    }  
}  
  
$runtime = CBPRuntime::GetRuntime();  
return $runtime->ExecuteResourceFile(  
    __FILE__,  
    "properties_dialog.php",  
    array(  
        "arCurrentValues" => $arCurrentValues,  
        "formName" => $formName,  
    )  
);  
}  
  
public static function GetPropertiesDialogValues($documentType, $activityName,  
    &$arWorkflowTemplate, &$arWorkflowParameters, &$arWorkflowVariables,  
    $arCurrentValues, &$arErrors)  
{
```



```
$runtime = CBPRuntime::GetRuntime();  
$arProperties = array("MapFields" => array());  
  
if (is_array($arCurrentValues) && count($arCurrentValues)>0)  
{  
    if (is_array($arCurrentValues["fields"]) && count($arCurrentValues["fields"]) > 0  
        && is_array($arCurrentValues["values"]) && count($arCurrentValues["values"]) > 0)  
    {  
        foreach($arCurrentValues["fields"] as $key => $value)  
        if (strlen($value) > 0 && strlen($arCurrentValues["values"][$key]) > 0)  
            $arProperties["MapFields"][$value] = $arCurrentValues["values"][$key];  
    }  
    $arProperties ["write2file"] = $arCurrentValues["write2file"] == "Y" ? "Y" : "N";  
    $arProperties ["path2file"] = $arCurrentValues["path2file"];  
}  
  
$arCurrentActivity  
=&CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate, $activityName);  
$arCurrentActivity["Properties"] = $arProperties;  
  
return true;  
}  
public static function ValidateProperties($arTestProperties = array(),  
CBPWorkflowTemplateUser $user = null)  
{  
    $arErrors = array();  
    $path = "";  
    if ($arTestProperties["path2file"])  
    {  
        $path = $_SERVER["DOCUMENT_ROOT"].$arTestProperties["path2file"];  
        if (!file_exists($path) && !is_writable(GetDirPath($path)))  
            $arErrors[] = array("code" => "DirNotWritable", "parameter" => "path2file",  
"message" => GetMessage("DIR_NOT_WRITABLE"));  
        if (file_exists($path) && !is_writable(GetDirPath($path)))  
            $arErrors[] = array("code" => "FileNotWritable", "parameter" => "path2file",  
"message" => GetMessage("FILE_NOT_WRITABLE"));  
    }  
}
```



```
        return array_merge($arErrors, parent::ValidateProperties($arTestProperties,
$user));
    }
}
?
>
```

Разъясним несколько моментов, на которые надо обратить внимание разработчикам.

- Зададим заголовок класса и конструктор. В конструкторе определяются параметры действия:

```
class CBPWrite2LogActivity
    extends CBPActivity
{
    public function __construct($name)
    {
        parent::__construct($name);
        $this->arProperties = array(
            "Title" => "",
            "MapFields" => null,
            "write2file" => null,
            "path2file"=> null
        );
    }
}
```

- Выводим диалог настройки параметров действия. Если параметры уже были заданы, то они инициализируются:

```
public static function GetPropertiesDialog($documentType, $activityName,
$arWorkflowTemplate, $arWorkflowParameters, $arWorkflowVariables,
$arCurrentValues = null, $formName = "")
{
    $runtime = CBPRuntime::GetRuntime();

    if (!is_array($arCurrentValues))
    {
        $arCurrentValues = array();

        $arCurrentActivity
        &CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate,
$activityName);
```



```
.....  
}  
  
$runtime = CBPRuntime::GetRuntime();  
return $runtime->ExecuteResourceFile(  
    __FILE__,  
    "properties_dialog.php",  
    array(  
        "arCurrentValues" => $arCurrentValues,  
        "formName" => $formName,  
    )  
);  
}  
}
```

- Сохранение настроек, сделанных в диалоге настройки действия:

```
public static function GetPropertiesDialogValues($documentType, $activityName,  
&$arWorkflowTemplate, &$arWorkflowParameters, &$arWorkflowVariables,  
$arCurrentValues, &$arErrors)  
{  
    $runtime = CBPRuntime::GetRuntime();  
    $arProperties = array("MapFields" => array());  
    //Получили массив значений $arCurrentValues  
    if (is_array($arCurrentValues) && count($arCurrentValues)>0)  
    {  
        .....  
        $arProperties ["write2file"] = $arCurrentValues["write2file"] == "Y" ? "Y" : "N";  
        $arProperties ["path2file"] = $arCurrentValues["path2file"];  
    }  
    //Получаем переменные текущего действия по ссылке и заменяем их  
    $arCurrentActivity  
    =&CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate,  
    $activityName);  
    $arCurrentActivity["Properties"] = $arProperties;  
  
    return true;  
}
```

- Проверка параметров действия:



```
public static function ValidateProperties($arTestProperties = array(),
CBPWorkflowTemplateUser $user = null)
{
    $arErrors = array();
    $path = "";
    if ($arTestProperties["path2file"])
    {
        $path = $_SERVER["DOCUMENT_ROOT"].$arTestProperties["path2file"];
        if (!file_exists($path) && !is_writable(GetDirPath($path)))
            $arErrors[] = array("code" => "DirNotWritable", "parameter" => "path2file",
"message" => GetMessage("DIR_NOT_WRITABLE"));
        if (file_exists($path) && !is_writable(GetDirPath($path)))
            $arErrors[] = array("code" => "FileNotWritable", "parameter" => "path2file",
"message" => GetMessage("FILE_NOT_WRITABLE"));
    }

    return array_merge($arErrors, parent::ValidateProperties($arTestProperties,
$user));
}
```

### Действие Создать задачу

Действие для создания задачи 2.0 (**task2activity**) создано по алгоритму, описанному выше. Поэтому просто приведем структуру действия и коды файлов. Функционал полностью повторяет стандартное действие для создания задачи.

### Структура файлов Activity

- /task2activity/
  - /lang/
    - /en/
      - .description.php
      - properties\_dialog.php
      - task2activity.php
    - /ru/
      - .description.php
      - properties\_dialog.php
      - task2activity.php
  - icon.gif
  - .description.php
  - properties\_dialog.php
  - task2activity.php

### Код файлов



task2activity\description.php:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arActivityDescription = array(
    "NAME" => GetMessage("BPTA2_DESCR_NAME"),
    "DESCRIPTION" => GetMessage("BPTA2_DESCR_DESCR"),
    "TYPE" => "activity",
    "CLASS" => "Task2Activity",
    "JSCLASS" => "BizProcActivity",
    "CATEGORY" => array(
        "ID" => "interaction",
    ),
);
?>
```

task2activity\properties\_dialog.php:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();
?>

<tr>
    <td align="right" width="40%"><span style="color:#FF0000;">*<?=GetMessage("BPTA1A_TASKNAME") ?>:</td>
    <td width="60%">
        <input type="text" name="task_name" id="id_task_name" value="<?=_htmlspecialchars($arCurrentValues["task_name"]); ?>" size="50">
        <input type="button" value="..." onclick="BPAShowSelector('id_task_name', 'string');">
    </td>
</tr>
<tr>
    <td align="right" width="40%"><span style="color:#FF0000;">*</span><?=GetMessage("BPTA1A_TASKCREATEDBY") ?>:</td>
    <td width="60%">
        <input type="text" name="task_created_by" id="id_task_created_by" value="<?=_htmlspecialchars($arCurrentValues["task_created_by"]); ?>" size="50">
```



```
<input type="button" value="..." onclick="BPAShowSelector('id_task_created_by',  
'user');">  
    </td>  
</tr>  
<tr>  
    <td align="right" width="40%"><span style="color:#FF0000;">*</span> <?=GetMessage("BPTA1A_TASKASSIGNEDTO") ?>:</td>  
    <td width="60%">  
        <input type="text" name="task_assigned_to" id="id_task_assigned_to" value="<?= htmlspecialchars($arCurrentValues["task_assigned_to"]) ?>" size="50">  
        <input type="button" value="..." onclick="BPAShowSelector('id_task_assigned_to', 'user');">  
    </td>  
</tr>  
<tr>  
    <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKACTIVEFROM") ?>:</td>  
    <td width="60%">  
        <span style="white-space:nowrap;"><input type="text" name="task_active_from" id="id_task_active_from" size="30" value="<?= htmlspecialchars($arCurrentValues["task_active_from"]) ?>"><?= CAdminCalendar::Calendar("task_active_from", "", "", true) ?></span>  
        <input type="button" value="..." onclick="BPAShowSelector('id_task_active_from', 'datetime');">  
    </td>  
</tr>  
<tr>  
    <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKACTIVEVETO") ?>:</td>  
    <td width="60%">  
        <span style="white-space:nowrap;"><input type="text" name="task_active_to" id="id_task_active_to" size="30" value="<?= htmlspecialchars($arCurrentValues["task_active_to"]) ?>"><?= CAdminCalendar::Calendar("task_active_to", "", "", true) ?></span>  
        <input type="button" value="..." onclick="BPAShowSelector('id_task_active_to', 'datetime');">  
    </td>  
</tr>  
<tr>  
    <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKDETAILTEXT") ?>:</td>
```



```
<td width="60%">
    <textarea name="task_detail_text" id="id_task_detail_text" rows="7"
cols="40"><?= htmlspecialchars($arCurrentValues["task_detail_text"]) ?></textarea>
    <input type="button" value="..." onclick="BPAShowSelector('id_task_detail_text',
'string');">
</td>
</tr>
<tr>
    <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKPRIORITY") ?></td>
    <td width="60%">
        <select name="task_priority">
            <?
            foreach ($arTaskPriority as $key => $value)
            {
                ?><option value=<?=$key ?>"<?= $arCurrentValues["task_priority"] == $key ? " selected" : "" ?>><?= $value ?></option><?
            }
            ?
        </select>
    </td>
</tr>
<tr>
    <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKGROUPID") ?></td>
    <td width="60%">
        <select name="task_group_id" id="id_task_group_id">
            <?
            foreach ($arGroups as $key => $value)
            {
                ?><option value=<?=$key ?>"<?= $arCurrentValues["task_group_id"] == $key ? " selected" : "" ?>><?= $value ?></option><?
            }
            ?
        </select>
    </td>
</tr>
<tr>
    <td align="right" width="40%"><?= GetMessage("BPTA1A_CHANGE_DEADLINE") ?></td>
```



```
<td width="60%">
    <input type="checkbox" name="task_change_deadline"
id="id_task_change_deadline" <?= ($arCurrentValues["task_change_deadline"] == "Y")?
"checked":""?>>
</td>
</tr>
<tr>
    <td align="right" width="40%"><?= GetMessage("BPTA1A_CHECK_RESULT") ?>:</td>
    <td width="60%">
        <input type="checkbox" name="task_check_result" id="id_task_check_result" <?=
($arCurrentValues["task_check_result"] == "Y")? "checked":""?>>
        </td>
    </tr>
    <tr>
        <td align="right" width="40%"><?= GetMessage("BPTA1A_ADD_TO_REPORT") ?>:</td>
        <td width="60%">
            <input type="checkbox" name="task_report" id="id_task_report" <?=
($arCurrentValues["task_report"] == "Y")? "checked":""?>>
            </td>
        </tr>
        <tr>
            <td align="right" width="40%"><?= GetMessage("BPTA1A_TASKTRACKERS") ?>:</td>
            <td width="60%">
                <input type="text" name="task_trackers" id="id_task_trackers" value=<?=
htmlspecialchars($arCurrentValues["task_trackers"]) ?>" size="50">
                <input type="button" value="..." onclick="BPAShowSelector('id_task_trackers',
'user');">
            </td>
        </tr>
    
```

task2activity\task2activity.php:

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!=true)die();

class CBPTask2Activity
    extends CBPAActivity
{
    public function __construct($name)
```



```
{  
    parent::__construct($name);  
    $this->arProperties = array(  
        "Title" => "", //ПСП°П·PIP°PSPëPμ PrPμPNøCЃC, PIPëCј  
        "TaskGroupId" => "", //P»PëC‡PSP°Cј PëP»Pë PiCЂC‡PíPiP°  
        "TaskOwnerId" => "", //  
        "TaskCreatedBy" => "", //P°PIC, PsCЂ  
        "TaskActiveFrom" => "", //PsC,  
        "TaskActiveTo" => "", //PrPs  
        "TaskName" => "", //PSP°P·PIP°PSPëPμ C,P°CЃPëPë  
        "TaskDetailText" => "", //PsPiPëCЃP°PSPëPμ  
        "TaskPriority" => "", //PїCЂPëPsCЂPëC, PμC,  
        "TaskAssignedTo" => "", //PsC, PIPμC, CЃC, PIPμPSPSC<PNø  
        "TaskTrackers" => "", //CЃP»PμPrCјC‰PëPμ  
        "TaskCheckResult" => "", //  
        "TaskReport" => "", //  
        "TaskChangeDeadline" => "",  
    );  
}  
  
private function __GetUsers($arUsersDraft)  
{  
  
    $arUsers = array();  
  
    $rootActivity = $this->GetRootActivity();  
    $documentId = $rootActivity->GetDocumentId();  
  
    $documentService = $this->workflow->GetService("DocumentService");  
  
    $arUsersDraft = (is_array($arUsersDraft) ? $arUsersDraft : array($arUsersDraft));  
    $l = strlen("user_");  
    foreach ($arUsersDraft as $user)  
    {  
        if (substr($user, 0, $l) == "user_")  
        {  
            $user = intval(substr($user, $l));  
        }  
    }  
}
```

```
        if ($user > 0)
            $arUsers[] = $user;
        }
    else
    {
        $arDSUsers = $documentService->GetUsersFromUserGroup($user,
$documentId);
        foreach ($arDSUsers as $v)
        {
            $user = intval($v);
            if ($user > 0)
                $arUsers[] = $user;
        }
    }

    return $arUsers;
}

public function Execute()
{
    if (!CModule::IncludeModule("tasks"))
        return CBPActivityExecutionStatus::Closed;

    $arTaskCreatedBy = $this->__GetUsers($this->TaskCreatedBy);
    $arTaskAssignedTo = $this->__GetUsers($this->TaskAssignedTo);

    if (count($arTaskCreatedBy) <= 0 || count($arTaskAssignedTo) <= 0)
        return CBPActivityExecutionStatus::Closed;

    $arTaskTrackers = $this->__GetUsers($this->TaskTrackers);

    $bFirst = true;
    $ACCOMPLICES = array();
    foreach($arTaskAssignedTo as $respUser)
    {
```



```
if ($bFirst)
{
    $RESPONSIBLE_ID = $respUser;
    $bFirst = false;
}
else
    $ACCOMPLICES[] = $respUser;
}

$arFields = array(
    "MODIFIED_BY" => $arTaskCreatedBy[0],
    "CREATED_BY" => $arTaskCreatedBy[0],
    "SITE_ID" => SITE_ID,
    "STATUS" => "1",
    "DATE_CREATE" => date($GLOBALS["DB"]-
>DateFormatToPHP(FORMAT_DATETIME)),
    "START_DATE_PLAN" => $this->TaskActiveFrom,
    "END_DATE_PLAN" => $this->TaskActiveTo,
    "DEADLINE" => $this->TaskActiveTo,
    "TITLE" => $this->TaskName,
    "DESCRIPTION" => $this->TaskDetailText,
    "PRIORITY" => $this->TaskPriority,
    "RESPONSIBLE_ID" => $RESPONSIBLE_ID,
    "AUDITORS" => $arTaskTrackers,
    "ADD_IN_REPORT" => $this->TaskReport,
    "TASK_CONTROL" => $this->TaskCheckResult,
    "ALLOW_CHANGE_DEADLINE" => $this->TaskChangeDeadline,
);
if ($this->TaskGroupId && $this->TaskGroupId != 0)
    $arFields["GROUP_ID"] = $this->TaskGroupId;

if (count ($ACCOMPLICES) > 0)
    $arFields["ACCOMPLICES"] = $ACCOMPLICES;

$task = new CTasks;
$result = $task->Add($arFields);
```



```
if ($result)
    $this->WriteToTrackingService(str_replace("#VAL#", $result,
GetMessage("BPSA_TRACK_OK")));

$arErrors = $task->GetErrors();
if (count($arErrors) > 0)
    $this->WriteToTrackingService(GetMessage("BPSA_TRACK_ERROR"));

return CBPActivityExecutionStatus::Closed;
}

public static function ValidateProperties($arTestProperties = array(),
CBPWorkflowTemplateUser $user = null)
{
    $arErrors = array();

    if (!array_key_exists("TaskAssignedTo", $arTestProperties) ||
count($arTestProperties["TaskAssignedTo"]) <= 0)
        $arErrors[] = array("code" => "NotExist", "parameter" =>
"TaskAssignedTo", "message" => GetMessage("BPSNMA_EMPTY_TASKASSIGNEDTO"));

    if (!array_key_exists("TaskName", $arTestProperties) ||
count($arTestProperties["TaskName"]) <= 0)
        $arErrors[] = array("code" => "NotExist", "parameter" => "TaskName",
"message" => GetMessage("BPSNMA_EMPTY_TASKNAME"));

    return array_merge($arErrors, parent::ValidateProperties($arTestProperties,
$user));
}

public static function GetPropertiesDialog($documentType, $activityName,
$arWorkflowTemplate, $arWorkflowParameters, $arWorkflowVariables, $arCurrentValues =
null, $formName = "")
{
    $runtime = CBPRuntime::GetRuntime();

    if (!CModule::IncludeModule("socialnetwork"))
        return;
```



```
$arMap = array(
    "TaskGroupId" => "task_group_id",
    "TaskOwnerId" => "task_owner_id",
    "TaskCreatedBy" => "task_created_by",
    "TaskActiveFrom" => "task_active_from",
    "TaskActiveTo" => "task_active_to",
    "TaskName" => "task_name",
    "TaskDetailText" => "task_detail_text",
    "TaskPriority" => "task_priority",
    "TaskAssignedTo" => "task_assigned_to",
    "TaskTrackers" => "task_trackers",
    "TaskCheckResult" => "task_check_result",
    "TaskReport" => "task_report",
    "TaskChangeDeadline" => "task_change_deadline",
);

if (!is_array($arWorkflowParameters))
    $arWorkflowParameters = array();
if (!is_array($arWorkflowVariables))
    $arWorkflowVariables = array();

if (!is_array($arCurrentValues))
{
    $arCurrentActivity =
&CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate, $activityName);
    if (is_array($arCurrentActivity["Properties"]))
    {
        foreach ($arMap as $k => $v)
        {
            if (array_key_exists($k, $arCurrentActivity["Properties"]))
            {
                if      ($k      ==      "TaskCreatedBy"      ||      $k      ==
"TaskAssignedTo"      ||      $k      ==      "TaskTrackers")
                    $arCurrentValues[$arMap[$k]]      =
CBPHelper::UsersArrayToString($arCurrentActivity["Properties"][$k],      $arWorkflowTemplate,
$documentType);
            }
        }
    }
}
```

```
else
```



```
$arCurrentValues[$arMap[$k]] =  
$arCurrentActivity["Properties"][$k];  
}  
elseif ($k == "TaskPriority")  
{  
    $arCurrentValues[$arMap[$k]] = "1";  
}  
else  
{  
    $arCurrentValues[$arMap[$k]] = "";  
}  
}  
}  
else  
{  
    foreach ($arMap as $k => $v)  
        $arCurrentValues[$arMap[$k]] = "";  
}  
  
}  
  
$arGroups = array( GetMessage("TASK_EMPTY_GROUP"));  
$db = CSocNetGroup::GetList(array("NAME" => "ASC"), array("ACTIVE" => "Y"),  
false, false, array("ID", "NAME"));  
while ($ar = $db->GetNext())  
    $arGroups[$ar["ID"]] = "[".$ar["ID"]."].".$ar["NAME"];  
  
$arTaskPriority = array(0, 1, 2);  
foreach($arTaskPriority as $k => $v)  
    $arTaskPriority[$v] = GetMessage("TASK_PRIORITY_". $v);  
  
return $runtime->ExecuteResourceFile(  
    __FILE__,  
    "properties_dialog.php",  
    array(  
        "arCurrentValues" => $arCurrentValues,
```



```
        "formName" => $formName,
        "arGroups" => $arGroups,
        "arTaskPriority" => $arTaskPriority,
    )
);
}

public static function GetPropertiesDialogValues($documentType, $activityName,
&$arWorkflowTemplate, &$arWorkflowParameters, &$arWorkflowVariables,
$arCurrentValues, &$arErrors)
{
    $arErrors = array();

    $runtime = CBPRuntime::GetRuntime();

    $arMap = array(
        "task_group_id" => "TaskGroupId",
        "task_owner_id" => "TaskOwnerId",
        "task_created_by" => "TaskCreatedBy",
        "task_active_from" => "TaskActiveFrom",
        "task_active_to" => "TaskActiveTo",
        "task_name" => "TaskName",
        "task_detail_text" => "TaskDetailText",
        "task_priority" => "TaskPriority",
        "task_assigned_to" => "TaskAssignedTo",
        "task_trackers" => "TaskTrackers",
        "task_forum_id" => "TaskForumId",
        "task_check_result" => "TaskCheckResult",
        "task_report" => "TaskReport",
        "task_change_deadline" => "TaskChangeDeadline",
    );

    $arProperties = array();
    foreach ($arMap as $key => $value)
    {
        if ($key == "task_created_by" || $key == "task_assigned_to" || $key ==
"task_trackers")
```

```

        continue;
        $arProperties[$value] = $arCurrentValues[$key];
    }

    $arProperties["TaskCreatedBy"] = CBPHelper::UsersStringToArray($arCurrentValues["task_created_by"], $documentType, $arErrors);
    if (count($arErrors) > 0)
        return false;

    $arProperties["TaskAssignedTo"] = CBPHelper::UsersStringToArray($arCurrentValues["task_assigned_to"], $documentType, $arErrors);
    if (count($arErrors) > 0)
        return false;

    $arProperties["TaskTrackers"] = CBPHelper::UsersStringToArray($arCurrentValues["task_trackers"], $documentType, $arErrors);
    if (count($arErrors) > 0)
        return false;

    $arErrors = self::ValidateProperties($arProperties, new CBPWorkflowTemplateUser(CBPWorkflowTemplateUser::CurrentUser));
    if (count($arErrors) > 0)
        return false;

    $arCurrentActivity = &CBPWorkflowTemplateLoader::FindActivityByName($arWorkflowTemplate, $activityName);
    $arCurrentActivity["Properties"] = $arProperties;

    return true;
}
?

```

task2activity\lang\ru\description.php:

```

<?
$MESS ['BPTA2_DESCR_DESCR'] = "Добавление задачи 2.0";

```



```
$MESS ['BPTA2_DESCR_NAME'] = "Задача 2.0";
?>
```

task2activity\lang\ru\properties\_dialog.php:

```
<?
$MESS ['BPTA1A_TASKGROUPID'] = "Группа соц. сети";
$MESS ['BPTA1A_TASKCREATEDBY'] = "Задача создается от имени";
$MESS ['BPTA1A_TASKASSIGNEDTO'] = "Ответственный";
$MESS ['BPTA1A_TASKACTIVEFROM'] = "Начало";
$MESS ['BPTA1A_TASKACTIVETO'] = "Окончание";
$MESS ['BPTA1A_TASKNAME'] = "Название задачи";
$MESS ['BPTA1A_TASKDETAILTEXT'] = "Описание задачи";
$MESS ['BPTA1A_TASKTRACKERS'] = "Следящие";
$MESS ['BPTA1A_TASKPRIORITY'] = "Важность";
$MESS ['BPTA1A_TASKFORUM'] = "Форум для комментариев";
$MESS ['BPTA1A_ADD_TO_REPORT'] = "Проконтролировать результат выполнения";
$MESS ['BPTA1A_CHECK_RESULT'] = "Включить задачу в отчет по эффективности";
$MESS ['BPTA1A_CHANGE_DEADLINE'] = "Разрешить ответственному менять сроки";
?>
```

task2activity\lang\ru\task2activity.php:

```
<?
$MESS ['BPSNMA_EMPTY_TASKASSIGNEDTO'] = "Свойство 'Ответственный' не указано.";
$MESS ['BPSNMA_EMPTY_TASKNAME'] = "Свойство 'Название задачи' не указано.";
$MESS ['TASK_PRIORITY_0'] = "Низкая";
$MESS ['TASK_PRIORITY_1'] = "Средняя";
$MESS ['TASK_PRIORITY_2'] = "Высокая";
$MESS ['TASK_EMPTY_GROUP'] = "Персональная задача";
$MESS ['BPSA_TRACK_OK'] = "Создана задача с ID ##VAL##";
$MESS ['BPSA_TRACK_ERROR'] = "При создании задачи произошла ошибка.";
?>
```

## Произвольный PHP код в бизнес-процессе

В некоторых случаях для успешного решения задачи в рамках бизнес-процессов можно обойтись без создания собственных действий. Для этого достаточно использовать собственный PHP код в рамках штатного действия PHP код.

Перед тем как рассмотреть примеры дадим несколько общих советов по созданию кода:

- Код будет выполняться в своем пространстве, то есть нужно учитывать, что модули заранее не подключались. Поэтому нужно вызывать стандартное API **Bitrix Framework**.
- Символы <??> вносить не нужно.
- Внимательно следите за типами переменных - приведение типа переменных организуйте в обязательном порядке (в некоторых примерах главы это пропущено для упрощения восприятия).

Рассмотрим примеры использования кода в рамках действия **PHP код**.

### Как запустить из одного бизнес процесса другой?

Через интерфейс дизайнера бизнес процессов это сделать нельзя, придется делать API запросы.

- Перейдите в административную часть системы на страницу с документами, для которых нужно создать новый шаблон бизнес-процесса.
- Перейдите на страницу с бизнес-процессами и создайте последовательный бизнес процесс при помощи дизайнера бизнес процессов.
- Добавьте в шаблон действие **PHP код** (раздел **Прочие**).

В данном действии можно вызывать более сложный бизнес-процесс, например **Двухэтапное утверждение** из стандартных шаблонов бизнес-процессов.

- Откройте диалог настроек параметров действия с помощью иконки .
- Введите код:

```
CBPDocument::StartWorkflow(  
    6,  
    array("iblock", "CIBlockDocument", "{=Document:ID}"),  
    array("Voters"=>array("user_1")));
```

Что передается в метод, который создает бизнес процесс:

- Первый аргумент - это ID запускаемого шаблона бизнес процесса (в нашем примере это **Двухэтапное утверждение**)

- второй параметр это параметры документа, для которого создается бизнес-процесс. Здесь задается то, что бизнес-процесс будет доступен в административной части и переменная {=Document:ID}.
- Последний, третий параметр, это массив параметров запускаемого бизнес-процесса. В нашем случае, это тот пользователь, которого необходимо ознакомить с документом. В примере, это администратор сайта.

## Вывод в лог

Если разработчик поддается желанию использовать в полную силу функционал бизнес-процессов, то ему потребуются дополнительные возможности функционала. Например, вывод любых сообщений в лог бизнес-процесса.

Допустим, используется определение руководителя из предыдущего примера или просто необходимо вывести в лог текст заявки.

Есть два варианта решения: просто вывод в лог из действия PHP код или создание собственного действия Запись в лог. Рассмотрим оба варианта.

### PHP код

Допустим, что при создании бизнес-процесса пользователь вводит текст какой-то заявки, который сохраняется в переменной **Text**. Чтобы вывести в лог значение этой переменной нужно просто в действии **PHP код** добавить вызов:

```
$rootActivity = $this->GetRootActivity();
$this->WriteToTrackingService($rootActivity->GetVariable("Text"));

Или же просто вывести тестовое сообщение:
$this->WriteToTrackingService("Это тест");
```

### Создание действия

Создайте недостающие папки относительно корня сайта:

- bitrix\activities\custom
- bitrix\activities\custom\logactivity
- bitrix\activities\custom\logactivity\lang\ru

В папке /logactivity/ создайте файлы:

- logactivity.php**

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true)die();
```



```
class CBPLogActivity
    extends CBPAcitivity
{
    public function __construct($name)
    {
        parent::__construct($name);
        $this->arProperties = array(
            "Title" => "",
        );
    }

    public function Execute()
    {
        $rootActivity = $this->GetRootActivity();
        $this->WriteToTrackingService($rootActivity->GetVariable("Text"));

        return CBPAcitivityExecutionStatus::Closed;
    }
}

?>
```

- **.description.php**

```
<?
if (!defined("B_PROLOG_INCLUDED") || B_PROLOG_INCLUDED!==true) die();

$arActivityDescription = array(
    "NAME" => GetMessage("BPMA_DESCR_NAME"),
    "DESCRIPTION" => GetMessage("BPMA_DESCR_DESCR"),
    "TYPE" => "activity",
    "CLASS" => "LogActivity",
    "JSCLASS" => "BizProcActivity",
    "CATEGORY" => array(
        "ID" => "other",
    ),
);?>
```

Создайте файл `logactivity\lang\ru\.description.php`:

```
<?
$MESS ['BPMA_DESCR_NAME'] = "Запись в лог";
$MESS ['BPMA_DESCR_DESCR'] = "Запись сообщения в лог";
?>
```

Теперь в дизайнере бизнес-процессов появилось новое действие **Запись в лог** в разделе **Прочее** и действие просто записывает в лог значение переменной **Text**.

## Вывод в лог. Переменные

Рассмотрим примеры как вывести в лог некоторые переменные с помощью PHP-кода.

### Переменная типа Список

Допустим, что используется переменная в бизнес-процессе DB типа **Список** следующего содержания:

```
[База данных 1]DB1
[База данных 2]DB2
[База данных 3]DB3
```

И необходимо вывести в лог значение этой переменной (при этом она может принимать множественные значения). Вывод нужно сделать перебором значений:

```
$rootActivity = $this->GetRootActivity();
$list = $rootActivity->GetVariable("DB");
foreach ($list as $k => $v)
{
    $str = $str." ".$v;
}
$this->WriteToTrackingService("Выбраны следующие БД: ".$str);
```

### Переменная типа Привязка к пользователю

Допустим, что используется переменная **Manager**, которая имеет тип **Привязка к пользователю** и представляет собой строку типа `user_145`, где число – это ID пользователя. Выведем ее значение в виде фамилии и имени.

```
$str = $rootActivity->GetVariable("Manager");
$str = str_replace("user_", "", $str);
$buf = CUser::GetByID(intval($str))->Fetch();
$this->WriteToTrackingService(" Руководитель:". $buf['NAME']. " [". $buf['ID']. "]);
```



## Арифметические действия в бизнес-процессе

Необходимость сложить две переменные бизнес-процесса в ходе выполнения его контекста может быть востребована когда вы анализируете расходы/доходы. Например: как пересчитать стоимость работы программиста.

Задайте параметры и переменные в шаблоне бизнес-процесса:

- параметр бизнес-процесса: `{=Template:integrator_USD}` - стоимость работы программиста в долларах;
- переменная бизнес процесса `{=Variable:kurs_usd}` - курс валюты для пересчета.

В действии **PHP код** используйте следующий код:

```
// получить текущий бизнес-процесс
$rootActivity = $this->GetRootActivity();

// получить значение переменной бизнес-процесса {=Variable:kurs_usd}
$ курсUSD = $rootActivity->GetVariable("kurs_usd");

// получить значение параметра бизнес-процесса {=Template:integrator_USD}
$ integrator = $rootActivity->integrator_USD;

// пересчет валюты
$ integrator = $ integrator*$ курсUSD;

// установка значения параметра бизнес-процесса {=Template:integrator_USD}
$rootActivity->integrator_USD = $ integrator;

// установка значения переменной бизнес-процесса {=Template:ttl}

// Вы можете не только считать, но и делать что хотите с переменными и
// параметрами бизнес-процесса
$rootActivity->SetVariable("ttl",
    'Минимально возможная цена:'.number_format($min_ttl,0,',','')." руб\n".
    'Прибыль от минимальной цены:'.number_format($min_ttl_plus,0,',','')." руб\n".
    'Налог:'.number_format($min_ttl*$nalog,0,',','')." руб\n\n".
    'Средняя цена:'.number_format($ttl,0,',','')." руб\n".
    'Прибыль от средней цены:'.number_format($ttl_plus,0,',','')." руб\n".
    'Налог:'.number_format($ttl*$nalog,0,',','')." руб\n"
);
```

## Вычисление ID начальника

Чтобы вычислить начальника автора документа с помощью действия **PHP-код** (т.е. без использования действия выбора пользователя), можно использовать приведенный ниже код.

В этом коде переменная **\$num** отвечает за уровень начальника (1 - непосредственный начальник, 2 - начальник начальника,...). ID начальника в формате БП (т.е. вида **user\_X**) записывается в переменную с именем **var5**. Эта переменная должна быть создана в параметрах БП и должна иметь тип привязки к пользователю.

```
$num = 1;
$userId = substr("{=Document:CREATED_BY}", 5);
$userId = intval($userId);
if ($userId > 0)
{
    CModule::IncludeModule("intranet");
    $dbUser = CUser::GetList(($by="id"), ($order="asc"), array("ID_EQUAL_EXACT"=>$userId),
array("SELECT" => array("UF_*")));
    $arUser = $dbUser->GetNext();
    $i = 0;
    while ($i < $num)
    {
        $i++;
        $arManagers = CIntranetUtils::GetDepartmentManager($arUser["UF_DEPARTMENT"],
$arUser["ID"], true);
        foreach ($arManagers as $key => $value)
        {
            $arUser = $value;
            break;
        }
    }
    $rootActivity = $this->GetRootActivity();
    $rootActivity->SetVariable("var5", "user_". $arUser["ID"]);
}
```

## Глава 12. Дополнительные сведения и примеры

В главе даются некоторые дополнительные сведения, которые могут быть полезными разработчикам. Кроме этого: примеры не типовых задач и их решений.

### Если нет описания API

Цитатник веб-разработчиков.

*Дмитрий Яковенко: Метод как всегда прекрасен и нездокументирован.*

При работе с **Bitrix Framework** очень большое значение имеет описание API. К сожалению составление описаний API нового функционала никогда не выходит одновременно с функционалом. Причин этого несколько:

- Писать API и сам функционал одновременно невозможно;
- Писать API сразу после выхода функционала – не практично, так как в первое время после выхода обнаруживается много багов, которые требуют исправления и, соответственно учета этих исправлений в документации.

Как правило, API пишется через некоторое время (обычно два-три месяца) после выхода релиза. К сожалению, бывает что и старый функционал не всегда полностью и верно описан: внесли изменения и забыли отметить в документации, например.

Для работы в таких условиях единственный выход у разработчика – смотреть сам код. В помощь для таких случаев есть специальный бесплатный модуль [Живое описание АПИ](#), который сканирует текущие файлы ядра и выводит список доступных API функций и событий всех модулей.

Возможности модуля:

- Модуль доступен только пользователю с правами администратора системы.
- Все модули сканируются последовательно один раз, после этого рядом с `live_api.php` появляется файл `live_api.data.php`, который содержит данные о функциях;
- Можно выбрать не только модуль, но и интересующий класс;
- В исходном коде имеющиеся функции и методы **Bitrix Framework** отображены в виде ссылок, которые ведут на их исходный код этих функций и методов.



<a href="#">Сканировать модули</a>
Выберите модуль: <input type="text" value="currency"/> класс: <input type="text"/> <input type="button" value="Поиск"/>
Искать метод или класс: <input type="text"/> <input type="button" value="Поиск"/>

## События модуля currency

Событие	Вызывается
<a href="#">CurrencyFormat</a>	CurrencyFormat
<a href="#">OnBeforeCurrencyDelete</a>	CCurrency::Delete
<a href="#">OnCurrencyDelete</a>	CCurrency::Delete

## Константы модуля currency

Константа	Проверяется
<a href="#">CURRENCY CACHE TIME</a>	CCurrencyRates::GetConvertFactor
<a href="#">CURRENCY SKIP CACHE</a>	CCurrencyRates::GetConvertFactor

## Список функций и методов модуля currency

Метод
<a href="#">CCurrency::Add(\$arFields)</a>
<a href="#">CCurrency::Delete(\$currency)</a>
<a href="#">CCurrency::GetBaseCurrency()</a>
<a href="#">CCurrency::GetByID(\$currency)</a>
<a href="#">CCurrency::GetCurrency(\$currency)</a>
<a href="#">CCurrency::GetList(&amp;\$by, &amp;\$order, \$lang = LANGUAGE_ID)</a>
<a href="#">CCurrency::SelectBox(\$sFieldName, \$sValue, \$sDefaultValue = "",</a>

### Как работать?

- С помощью кнопки **Сканировать модули** отсканируйте текущее состояние.
- В полях формы выберите модуль, класс, или воспользуйтесь поиском.

**⚠ Примечание:** Если будет выбран только модуль, то в форме отобразится все, относящееся к этому модулю.

Система автоматически отобразит все найденный сущности. По списку аргументов можно легко догадаться, что они означают.



Например, метод **CCurrency::GetList** (см. иллюстрацию выше) имеет два обязательных параметра: поле сортировки и порядок сортировки. Оба передаются по ссылке. Третий параметр язык, по умолчанию принимает значение текущего языка.

По клику на функцию открывается её описание в новом окне. Код функции читается прямо из файла, при этом уже скрипты не парсятся, вся необходимая информация передаётся в URL.

```
<?php
//    CCurrency::GetList
//    /bitrix/modules/currency/mysql/currency.php:36

function GetList(&$by, &$order, $lang = LANGUAGE_ID)
{
    global $DB;

    if (defined("CURRENCY_SKIP_CACHE") && CURRENCY_SKIP_CACHE
        || StrToLower($by) == "name"
        || StrToLower($by) == "currency"
        || StrToLower($order) == "desc")
    {
        $dbCurrencyList = CCurrency:: GetList($by, $order, $lang);
    }
    else
    {
        $by = "sort";
        $order = "asc";

        $cacheTime = CURRENCY_CACHE_DEFAULT_TIME;
        if (defined("CURRENCY_CACHE_TIME"))
            $cacheTime = IntVal(CURRENCY_CACHE_TIME);

        if ($GLOBALS["CACHE_MANAGER"]->Read($cacheTime, "currency_currency_list"))
        {
            $arCurrencyList = $GLOBALS["CACHE_MANAGER"]->Get("currency_currency_list");
            $dbCurrencyList = new CDBResult;
            $dbCurrencyList->InitFromArray($arCurrencyList);
        }
        else
        {
            $arCurrencyList = array();
            $dbCurrencyList = CCurrency:: GetList($by, $order, $lang);
            while ($arCurrency = $dbCurrencyList->Fetch())
                $arCurrencyList[] = $arCurrency;

            $GLOBALS["CACHE_MANAGER"]->Set("currency_currency_list", $arCurrencyList);
            $dbCurrencyList = new CDBResult;
        }
    }
}
```

По клику на событие или константу открывается метод, где оно инициируется. Вызов события (константы) подсвечивается.



```
<?php
// CurrencyFormat
// /bitrix/modules/currency/include.php:19

function CurrencyFormat ($fSum, $strCurrency)
{
    $result = "";
    $db_events = GetModuleEvents("currency", "CurrencyFormat");
    while ($arEvent = $db_events->Fetch())
        $result = ExecuteModuleEventEx($arEvent, Array($fSum, $strCurrency));

    if(strlen($result) > 0)
        return $result;

    if (!isset($fSum) || strlen($fSum)<=0)
        return "";

    $arCurFormat = CurrencyFormat&line=69>CCurrencyLang::GetCurrencyFormat ($strCurrency);

    if (!isset($arCurFormat["DECIMALS"]))
        $arCurFormat["DECIMALS"] = 2;
    $arCurFormat["DECIMALS"] = IntVal($arCurFormat["DECIMALS"]);

    if (!isset($arCurFormat["DEC_POINT"]))
        $arCurFormat["DEC_POINT"] = ".";
    if(!empty($arCurFormat["THOUSANDS_VARIANT"]))
    {
        if($arCurFormat["THOUSANDS_VARIANT"] == "N")
            $arCurFormat["THOUSANDS_SEP"] = "";
        elseif($arCurFormat["THOUSANDS_VARIANT"] == "D")
            $arCurFormat["THOUSANDS_SEP"] = ",";
    }
}
```

## Веб-сервисы

**Веб-служба, веб-сервис** (англ. *web service*) — программная система, идентифицируемая строкой *URI*, чьи публичные интерфейсы и привязки определены и описаны языком XML. Описание этой программной системы может быть найдено другими программными системами, которые могут взаимодействовать с ней согласно этому описанию посредством сообщений, основанных на XML, и передаваемых с помощью интернет-протоколов.

Достоинства веб-сервисов:

Веб-службы обеспечивают взаимодействие программных систем независимо от платформы. Веб-службы основаны на базе открытых стандартов и протоколов. Благодаря использованию XML достигается простота разработки и отладки веб-служб. Использование интернет-протокола HTTP обеспечивает взаимодействие программных систем через межсетевой экран.

Модуль Веб-сервисов служит для облегчения создания и интеграции с существующими веб-сервисами. Основное свое применение модуль найдет при разработке интеграций с действующими приложениями как внутри компании, так и с внешними уже работающими веб-сервисами.



## Пример создания windows-приложение для добавления новостей

Рассмотрим, как на его основе модуля **Веб-сервисы** можно сделать простейшее Windows-приложение для добавления новостей при помощи Visual Studio 2005 и .NET Framework 2.0.

### Создание информационного блока, в который будут добавляться новости

В нашем случае необходим информационный блок с некоторыми свойствами. Для простоты возьмем уже готовый информационный блок **Новости магазина** из демо-версии, его ID=33.

Название	Сорт.	Акт.	Элементов	Разделов	Сайт	Дата изм.	ID	Шаблоны бизнес-процессов
Новости магазина	100	Да	10	4	s1	23.11.2011 15:27:54	33	

### Создание компонента веб-сервиса для добавления новостей

При установке модуля **Веб-сервисы** создается новый компонент **bitrix:webservice.server**. Он предназначен для простого создания, тестирования и вывода в читабельном виде информации о ваших веб-сервисах. Для нашего веб-сервиса он будет являться «сервером».

Сначала создадим свой новый компонент. Для этого создадим новую папку в **/bitrix/components/demo**, назовем ее **webservice.addnews**. Как и любой другой компонент 2.0, компонент веб-сервиса должен содержать стандартные файлы описаний:

- Файл **.description.php**:

```
"Веб сервис добавления новостей",
"DESCRIPTION" => "Веб сервис добавления новостей",
"CACHE_PATH" => "Y",
"PATH" => array(
    "ID" => "service",
```



```
"CHILD" => array(
    "ID" => "webservice",
    "NAME" => "Веб-сервис добавления новостей."
)
),
);
?>
```

- Файл .parameters.php:

```
array(),
"PARAMETERS" => array(),
);
?>
```

- И исполняемый файл **component.php**. Создадим первоначально его в следующем виде:

```
wsname = "bitrix.webservice.addnews"; // название сервиса
$wsdesc->wsclassname = "CAddNewsWS"; // название класса
$wsdesc->wsdlauto = true;
$wsdesc->wsendpoint = CWebService::GetDefaultEndpoint();
$wsdesc->wstargetns = CWebService::GetDefaultTargetNS();

$wsdesc->classTypes = array();
$wsdesc->structTypes = Array();
$wsdesc->classes = array();

return $wsdesc;
}
}

$arParams["WEBSERVICE_NAME"] = "bitrix.webservice.addnews";
$arParams["WEBSERVICE_CLASS"] = "CAddNewsWS";
$arParams["WEBSERVICE_MODULE"] = "";

// передаем в компонент описание веб-сервиса
$APPLICATION->IncludeComponent(
    "bitrix:webservice.server",
    "",
```

```

$arParams
);

die();
?>

```

Это минимальное содержимое для определения веб-сервиса. Как видно он содержит наследованный от **IWebService** класс **CAddNewsWS**, переопределенный метод **GetWebServiceDesc**, возвращающий описание сервиса в формате **CWebServiceDesc** и вызов компонента **bitrix:webservice.server**, которому в качестве параметров передается описание нашего зарождающегося веб-сервиса.

Для проверки работоспособности создадим новую страницу например **ws\_addnews.php**, разместим наш новый компонент и сохраним.

 **Примечание:** При использовании **MS Visual Studio 2010** используйте адрес: [http://\\_ваш\\_домен\\_/ws\\_addnews.php?wsdl](http://_ваш_домен_/ws_addnews.php?wsdl).

После сохранения страница откроется в браузере:

Веб-сервис: bitrix.webservice.addnews

Пространство имен:	<a href="http://192.168.0.63:7710/">http://192.168.0.63:7710/</a>
Точка доступа:	<a href="http://192.168.0.63:7710/ws_addnews.php">http://192.168.0.63:7710/ws_addnews.php</a>
Стиль привязки:	document/literal only

Следующие операции поддерживаются. Формальное определение см. в [Описание службы](#).

Как мы видим, появилось описание нашего веб-сервиса. Но у него нет ни одного метода. Для создания метода нам потребуется добавить в класс новый метод, который принимает на вход в качестве параметров некоторые поля новости, а в качестве результата возвращает ID добавленной новости или ошибку:

```

function AddNews($NAME, $DATE, $PREVIEW_TEXT, $DETAIL_TEXT, $KEYWORDS, $SOURCE)
{
    $iblock_permission = CIBlock::GetPermission(33);
    if ($iblock_permission < "W")
    {
        $GLOBALS["USER"]->RequiredHTTPAuthBasic();
        return new CSOAPFault('Server Error', 'Unable to authorize user.');
    }
}

```



```
$arFields = Array(
    "IBLOCK_ID"=>33, // инфоблок "Новости магазина"
    "NAME"=>$NAME,
    "DATE_ACTIVE_FROM"=>$DATE,
    "PREVIEW_TEXT"=>$PREVIEW_TEXT,
    "DETAIL_TEXT"=>$DETAIL_TEXT,
    "PROPERTY_VALUES" => Array(
        "KEYWORDS"=>$KEYWORDS,
        "SOURCE"=>$SOURCE,
    )
);
$ib_element = new CIBlockElement();
$result = $ib_element->Add($arFields);
if($result>0)
    return Array("id"=>$result);

return new CSOAPFault( 'Server Error', 'Error: '.$ib_element->LAST_ERROR );
}
```

Зарегистрируем новый метод в массиве \$wsdesc->classes:

```
$wsdesc->classes = array(
    "CAddNewsWS"=> array(
        "AddNews" => array(
            "type"    => "public",
            "input"   => array(
                "NAME" => array("varType" => "string"),
                "DATE" => array("varType" => "string"),
                "PREVIEW_TEXT" => array("varType" => "string"),
                "DETAIL_TEXT" => array("varType" => "string"),
                "KEYWORDS" => array("varType" => "string"),
                "SOURCE" => array("varType" => "string"),
            ),
            "output"  => array(
                "id" => array("varType" => "integer")
            ),
            "httpauth" => "Y"
        ),
    ),
)
```

```
)  
);
```

В массиве содержится название класса и названия методов, с описанием входных и выходных параметров.

Вот и все, теперь если обновить страницу, на которой расположен компонент, то появится новый метод и также можно протестировать его работу непосредственно из браузера:

## Веб-сервис: bitrix.webservice.addnews

Для получения полного списка операций щелкните [здесь](#).

Пространство имен:	<a href="http://192.168.0.63:7710/">http://192.168.0.63:7710/</a>
Точка доступа:	<a href="http://192.168.0.63:7710/ws_addnews.php">http://192.168.0.63:7710/ws_addnews.php</a>
Стиль привязки:	document/literal only

***id:integer AddNews(string NAME, string DATE, string PREVIEW\_TEXT, string DETAIL\_TEXT, string KEYWORDS, string SOURCE);***

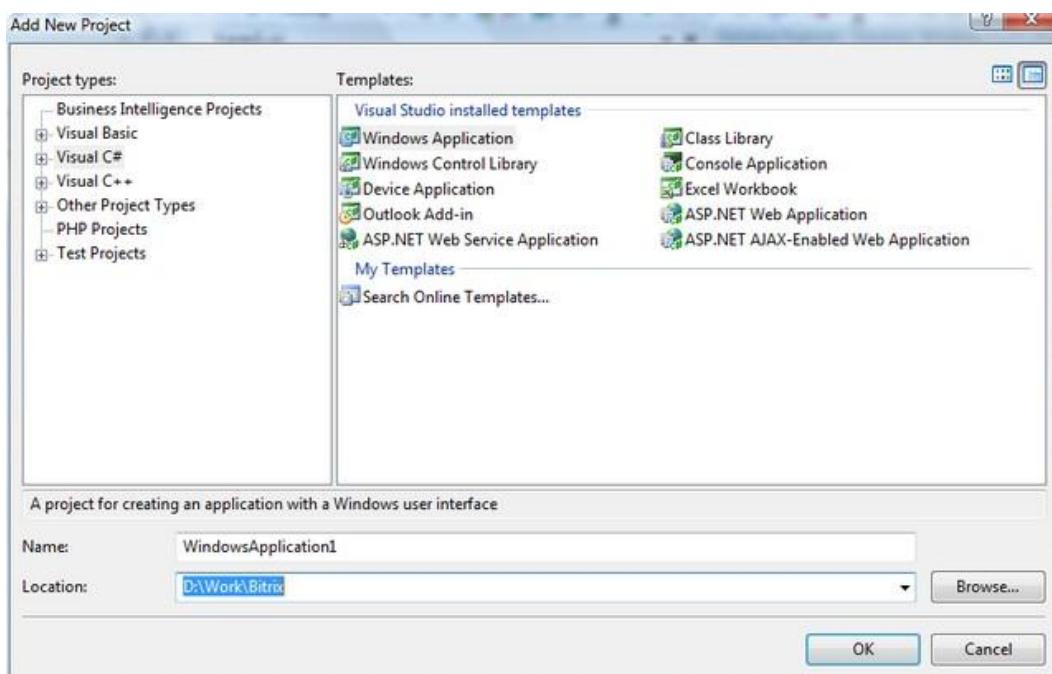
**Тест:**

Параметр	Значение
NAME:	<input type="text"/>
DATE:	<input type="text"/>
PREVIEW_TEXT:	<input type="text"/>
DETAIL_TEXT:	<input type="text"/>
KEYWORDS:	<input type="text"/>
SOURCE:	<input type="text"/>

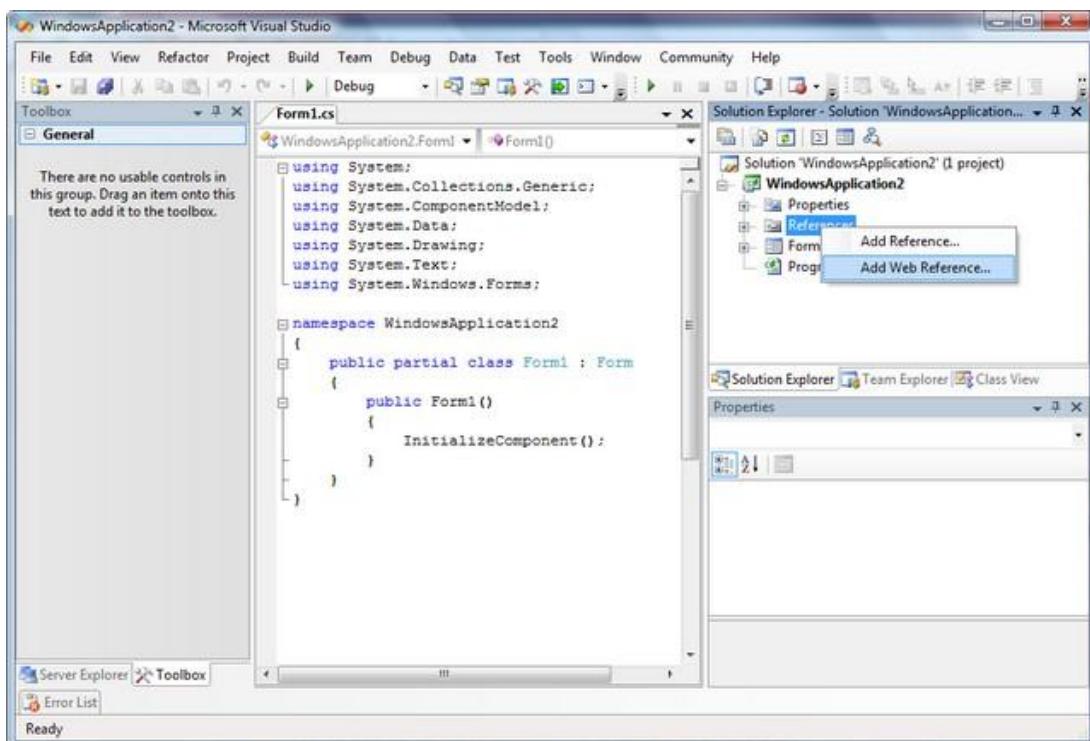
### Создание приложения

Теперь можно приступить к завершающему этапу - созданию в Visual Studio простого Windows-приложения. Для этого можно скачать и установить одну из бесплатных Express версий, например для C#.

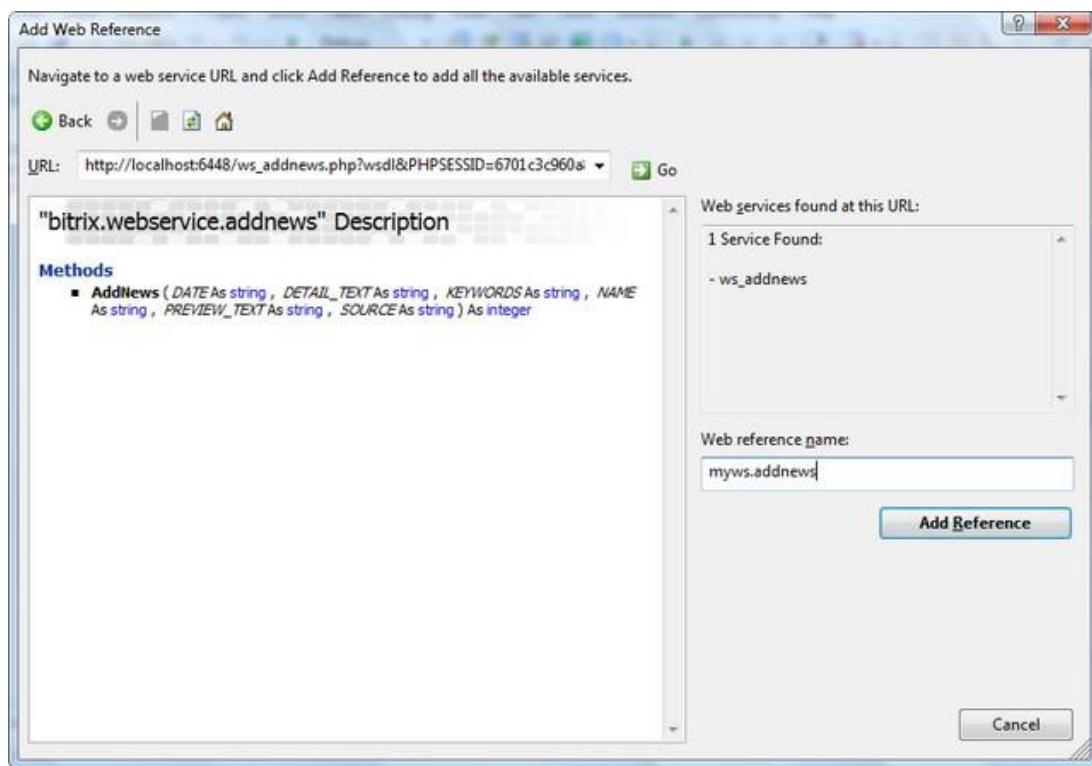
- Создаем новый проект в Visual Studio:



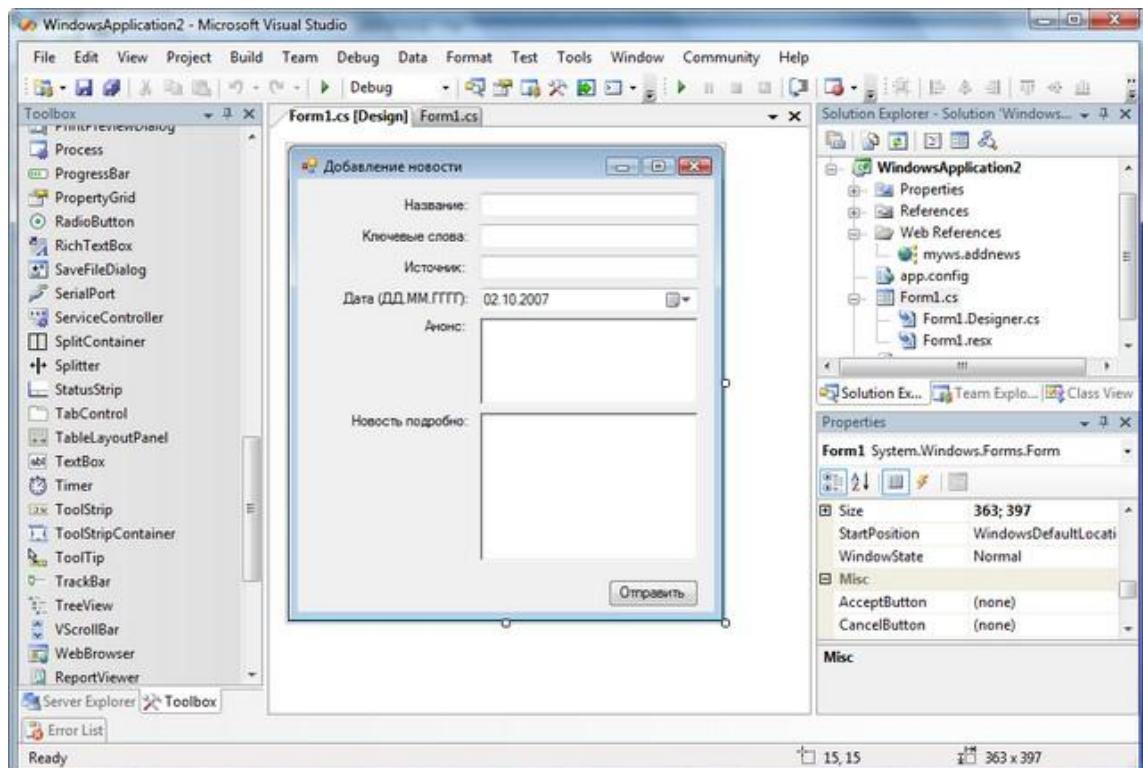
- На вкладке Solution Explorer создаем Web reference:



- Указываем ссылку на страницу с компонентом, нажимаем кнопку **Go**. В окне открывается уже знакомая нам страница с описанием сервиса, нажимаем на ней ссылку описание службы, Visual Studio считывает доступные методы и предлагает нам создать для них прокси-классы, переименовываем название в **myws.addnews** и нажимаем кнопку **Add reference**.



- Далее приступаем к созданию непосредственно самого окна ввода новости, для этого разместим на форме необходимые поля и кнопку **Отправить**:



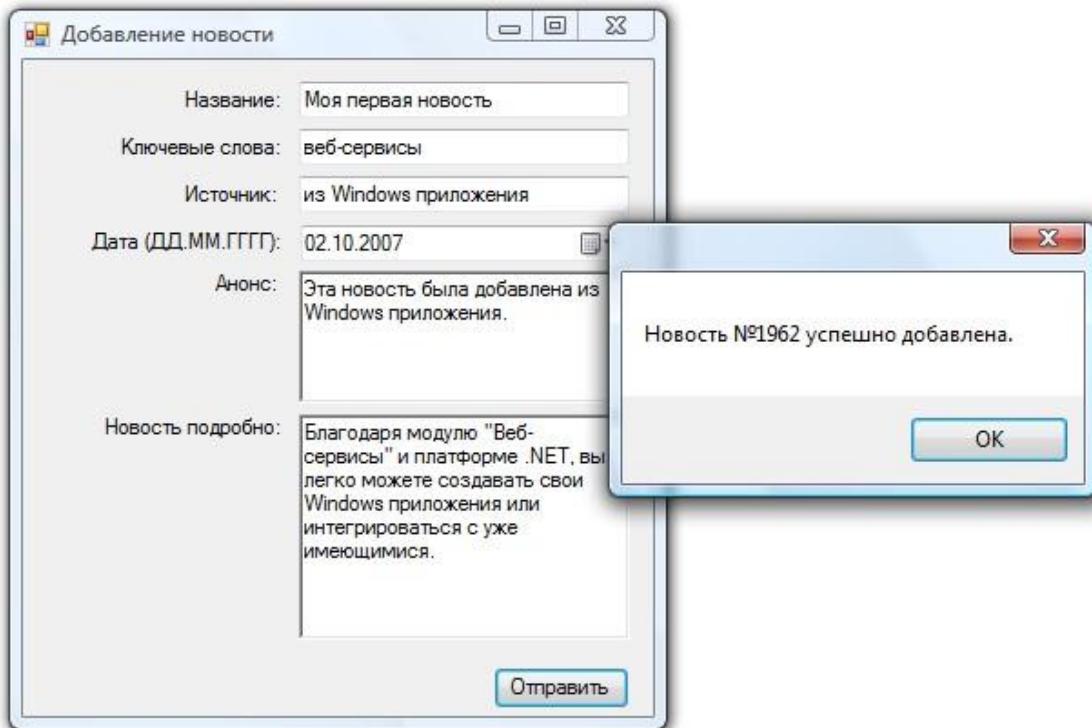
- Двойным кликом по кнопке открываем обработчик события нажатия на кнопку и размещаем код сохраняющий новость на сайте:

```
private void button1_Click(object sender, EventArgs e)
```



```
{  
    bitrixwebserviceaddnews news = new bitrixwebserviceaddnews();  
    news.Credentials = new NetworkCredential("admin", "password");  
    try  
    {  
        string result = news.AddNews(DATE.Text, DETAIL_TEXT.Text, KEYWORDS.Text,  
NAME.Text, PREVIEW_TEXT.Text, SOURCE.Text);  
        MessageBox.Show("Новость №" + result + " успешно добавлена.");  
    }  
    catch (System.Web.Services.Protocols.SoapHeaderException exception)  
    {  
        MessageBox.Show("Ошибка добавления новости [" + exception.Message + "]");  
    }  
}
```

- Компилируем приложение, запускаем, заполняем поля, нажимаем кнопку **Отправить**:



В данном уроке показан очень простой пример. Конечно, данный веб-сервис можно дорабатывать, например, чтобы в параметрах компонента можно было управлять в какой инфоблок, какую информацию добавлять, поддержку произвольных свойств и т.п. Также можно доработать и Windows-приложение, например, чтобы оно выгружало новости на сайт «пачками», содержало больше полей и настроек.



Исходный код компонента: [скачать](#).

Исходный код приложения: [скачать](#).

**!** *Примечание:* Для того, чтобы добавлять новости с картинкой, придется на стороне Windows-приложения читать файл, конвертировать его при помощи `System.Convert.ToString` в `BASE64`, а на стороне компонента конвертировать назад функцией `base64_decode`, сохранять его во временный файл и передавать на вход методу `CIBlockElement::Add()`, как одно из полей. Помимо этого, нам необходимо знать на сервере как минимум расширение (тип) файла, поэтому вместе с содержимым будем передавать оригинальное имя файла.

Таким образом вот так будет выглядеть наш класс веб-сервиса в компоненте:

```
class CAddNewsWS extends IWebService
{
    function AddNews($NAME, $DATE, $PREVIEW_TEXT, $DETAIL_TEXT, $KEYWORDS,
$SOURCE, $IMAGE_NAME, $IMAGE_CONTENT)
    {
        $iblock_permission = CIBlock::GetPermission(33);
        if ($iblock_permission < "W")
        {
            $GLOBALS["USER"]->>RequiredHTTPAuthBasic();
            return new CSOAPFault('Server Error', 'Unable to authorize user.');
        }

        $arFields = Array(
            "IBLOCK_ID"=>33, // инфоблок "Новости магазина"
            "NAME"=>$NAME,
            "DATE_ACTIVE_FROM"=>$DATE,
            "PREVIEW_TEXT"=>$PREVIEW_TEXT,
            "DETAIL_TEXT"=>$DETAIL_TEXT,
            "PROPERTY_VALUES" => Array(
                "KEYWORDS"=>$KEYWORDS,
                "SOURCE"=>$SOURCE,
            )
        );
        if(strlen($IMAGE_NAME)>0 && strlen($IMAGE_CONTENT)>0)
        {
            $IMAGE_CONTENT = base64_decode($IMAGE_CONTENT);
```



```
if(strlen($IMAGE_CONTENT)>0)
{
    $tmp_name = $_SERVER['DOCUMENT_ROOT'].'/bitrix/tmp/'.md5(uniqid(rand(), true)).".tmp";
    CheckDirPath($tmp_name);
    $f=fopen($tmp_name, "wb");
    fwrite($f, $IMAGE_CONTENT);
    fclose($f);
    $arFields["DETAIL_PICTURE"] = Array("name"=>$IMAGE_NAME,
    "tmp_name"=>$tmp_name, "size"=>strlen($IMAGE_CONTENT), "type"=>"image/jpeg");
}
}

$ib_element = new CIBlockElement();
$result = $ib_element->Add($arFields);
if($tmp_name)
    @unlink($tmp_name);
if($result>0)
    return Array("id"=>$result);

return new CSOAPFault( 'Server Error', 'Error: '.$ib_element->LAST_ERROR );
}

// метод GetWebServiceDesc возвращает описание сервиса и его методов
function GetWebServiceDesc()
{
    $wsdesc = new CWebServiceDesc();
    $wsdesc->wsname = "bitrix.webservice.addnews";
    $wsdesc->wsclassname = "CAddNewsWS";
    $wsdesc->wsdlauto = true;
    $wsdesc->wsendpoint = CWebService::GetDefaultEndpoint();
    $wsdesc->wstargetns = CWebService::GetDefaultTargetNS();

    $wsdesc->classTypes = array();
    $wsdesc->structTypes = Array();

    $wsdesc->classes = array(
        "CAddNewsWS"=> array(

```



```
"AddNews" => array(
    "type"      => "public",
    "input"     => array(
        "NAME" => array("varType" => "string"),
        "DATE" => array("varType" => "string"),
        "PREVIEW_TEXT" => array("varType" => "string"),
        "DETAIL_TEXT" => array("varType" => "string"),
        "KEYWORDS" => array("varType" => "string"),
        "SOURCE" => array("varType" => "string"),
        "IMAGE_NAME" => array("varType" => "string"),
        "IMAGE_CONTENT" => array("varType" => "string"),
        ),
    "output"   => array(
        "id" => array("varType" => "integer")
    ),
    "httpauth" => "Y"
),
),
);
);

return $wsdesc;
}
}
```

А вот так обработчик нажатия на кнопку в Windows-приложении (*IMAGE* - это новый контрол на форме, в нем должен быть путь к файлу на диске):

```
private void button1_Click(object sender, EventArgs e)
{
    Byte[] binaryData;
    string base64String = "";
    if(IMAGE.Text.Length>0)
    {
        try
        {
            System.IO.FileStream imageFile = new System.IO.FileStream(IMAGE.Text,
System.IO.FileMode.Open, System.IO.FileAccess.Read);
            binaryData = new Byte[imageFile.Length];
        }
    }
}
```



```
imageFile.Read(binaryData, 0, (int)imageFile.Length);
imageFile.Close();
base64String = System.Convert.ToBase64String(binaryData, 0, binaryData.Length);
}

catch (System.Exception exp)
{
    MessageBox.Show("Ошибка чтения картинки [" + exp.Message + "]");
    return;
}

bitrixwebserviceaddnews news = new bitrixwebserviceaddnews();
news.Credentials = new NetworkCredential("admin", "password");
try
{
    string result = news.AddNews(DATE.Text, DETAIL_TEXT.Text, base64String,
IMAGE.Text, KEYWORDS.Text, NAME.Text, PREVIEW_TEXT.Text, SOURCE.Text);
    MessageBox.Show("Новость №" + result + " успешно добавлена.");
}
catch (System.Web.Services.Protocols.SoapHeaderException exception)
{
    MessageBox.Show("Ошибка добавления новости [" + exception.Message + "]");
}
}
```

## Пример миграции базы сайта

**Пример** разработан для описания технологии миграции базы проекта на **Bitrix Framework** с **MySQL** на **SQL Express** без использования продуктов сторонних разработчиков.

Для своей работы система использует базу данных, в которой хранит контент и настройки сайта. В качестве СУБД используются свободно распространяемые *MySQL*, *SQL Express* или *Oracle*. В последнее время от клиентов, выбравших инсталляцию с *MySQL*, стали поступать вопросы о переходе на *SQL Express*. Документ предназначен для разработчиков и администраторов сайта, перед которыми стоит задача провести



миграцию действующего сайта. Специалисты, осуществляющие миграцию должны уметь работать с указанными базами данных.

Для обеспечения доступа к БД MySQL из внешних приложений были выбраны [коннекторы MySQL Connector/ODBC 5.1](#) и [MySQL Connector/Net 6.0](#). Первый коннектор используется при сравнении структур баз MySQL и SQL Express-инсталляций, второй – непосредственно при переносе данных.

При миграции не требуется создавать структуры таблиц, индексы, представления, ограничения, триггеры на стороне SQL Express по образу того, как это было в MySQL. Продукты **Bitrix Framework** имеют отдельные инсталляции для MySQL и SQL Express, т.е. структура базы и там, и там создается самостоятельно в процессе установки продукта и изменению не подлежит. В **Примере** приводится сравнение структур таблиц в случае MySQL и SQL Express, соотнесение таблиц и колонок, процесс копирования данных из БД MySQL в SQL Express на основе выстроенной схемы отображения.

## Предварительные операции

Согласно постановке задачи первоначально клиент имеет *1С-Битрикс: Управление сайтом* в варианте с базой MySQL. При создании **Примера** использовалась редакция **Большой бизнес** как имеющая наиболее сложную и полную структуру базы, установленная согласно разделу [Установка и настройка](#) курса **Администратор. Базовый**.

## Установка SQL Express

Для переноса данных необходимо иметь установку *1С-Битрикс: Управление сайтом* с использованием SQL Express. В отличие от MySQL SQL Express не входит в установку окружения *1С-Битрикс: Управление сайтом*. Его необходимо [скачать](#) и установить самостоятельно. Рекомендуется использовать последнюю версию продукта. Дистрибутив SQL Express распространяется в виде пакета Windows *Installer*.

SQL Express доступен в трех комплектациях:

1. SQL Server 2008 Express (Runtime Only)
2. SQL Server 2008 Express with Tools (Это п.1 + Management Studio (графическая утилита для разработки и администрирования)).
3. SQL Server 2008 Express with Advanced Services. (Это п.2 + полнотекстовый поиск и службы отчетности.)

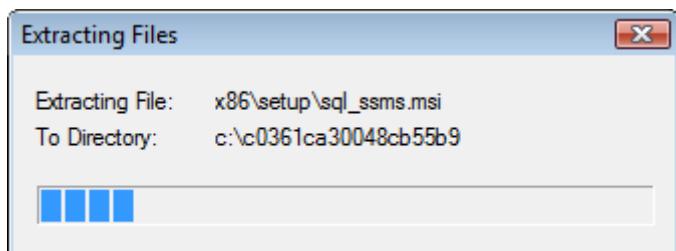
Рекомендуется использовать п.2 или 3. Пунктом 1 обычно пользуются независимые производители программного обеспечения, чтобы распространять его в составе своего приложения. Если планируется в дальнейшем прибегнуть к возможности полнотекстового поиска или разработке отчетов, выбирайте п.3. Если достаточно графической утилиты



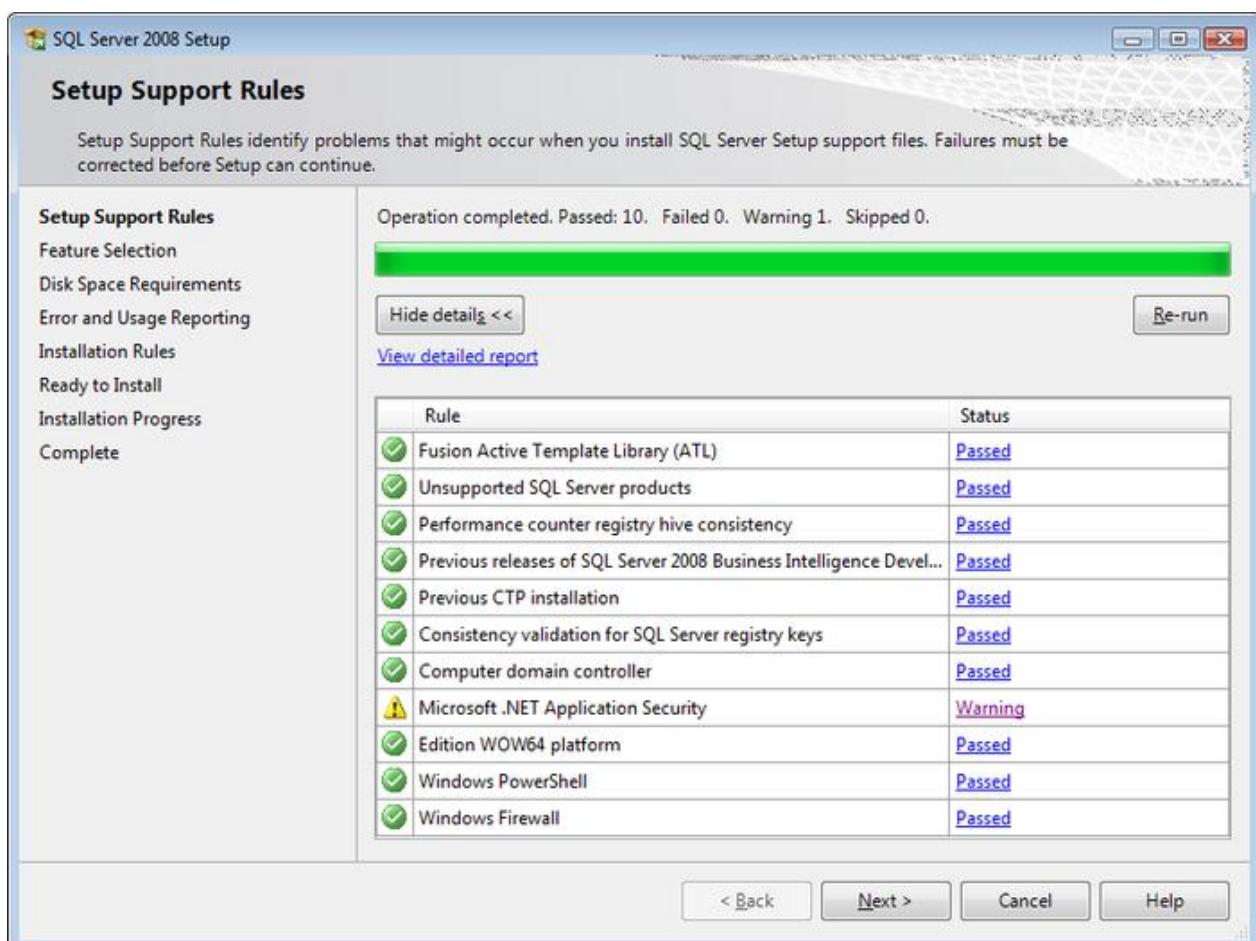
для разработки и администрирования, то используйте п.2. В **Примере** установка SQL Express иллюстрируется в варианте п.3.

## Установка

После запуска установки начинается процесс распаковки:



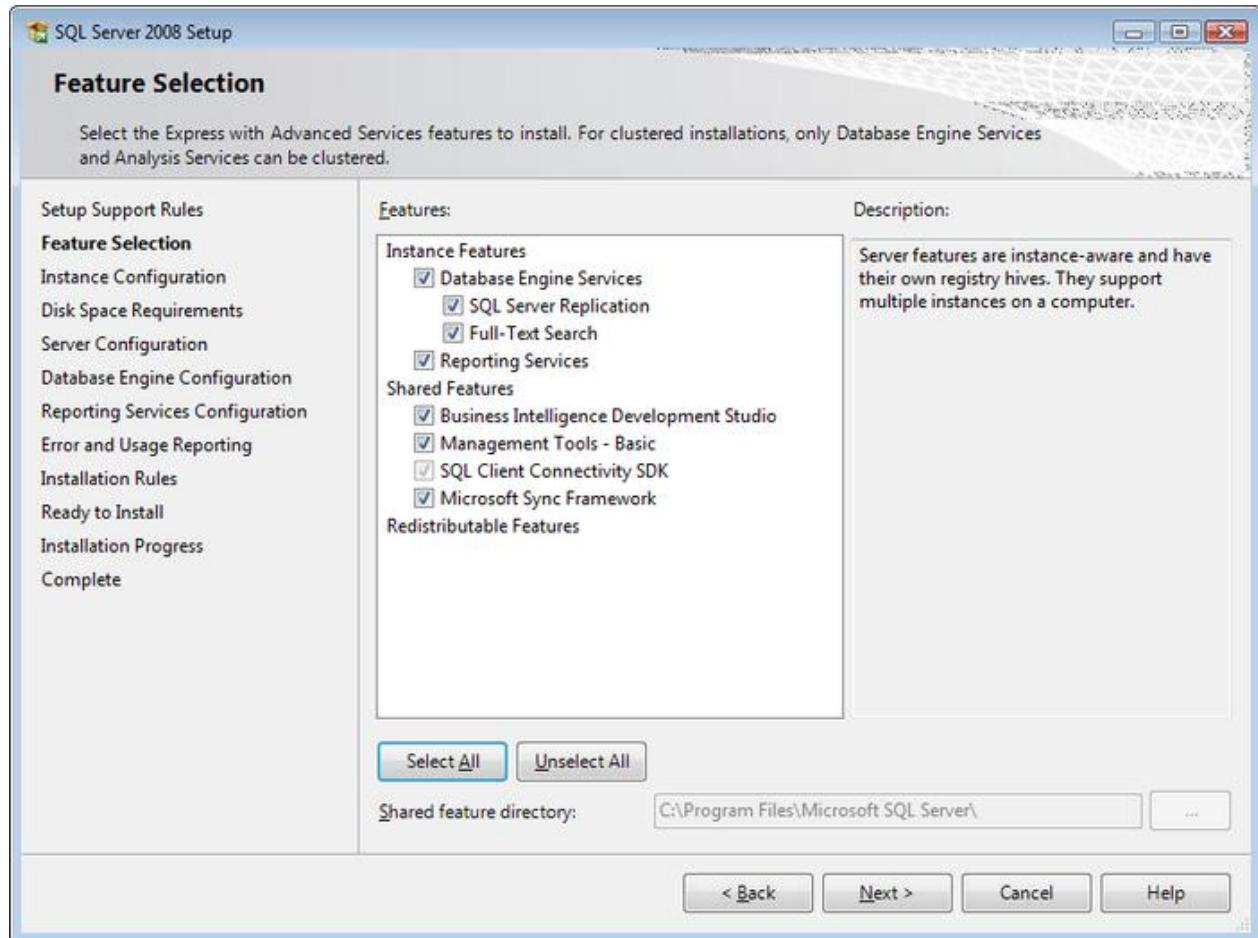
В начале установки происходит проверка характеристик аппаратной платформы и установленного ПО на предмет совместимости с *SQL Express*. Если обнаружено несовместимое ПО, либо какого-то ПО не хватает в качестве предварительно установленного, то программа установки выдаст об этом предупреждение:



После устранения неисправностей необходимо запустить процесс установки заново. По окончании проверки, если характеристики соблюдены, установщик переходит к

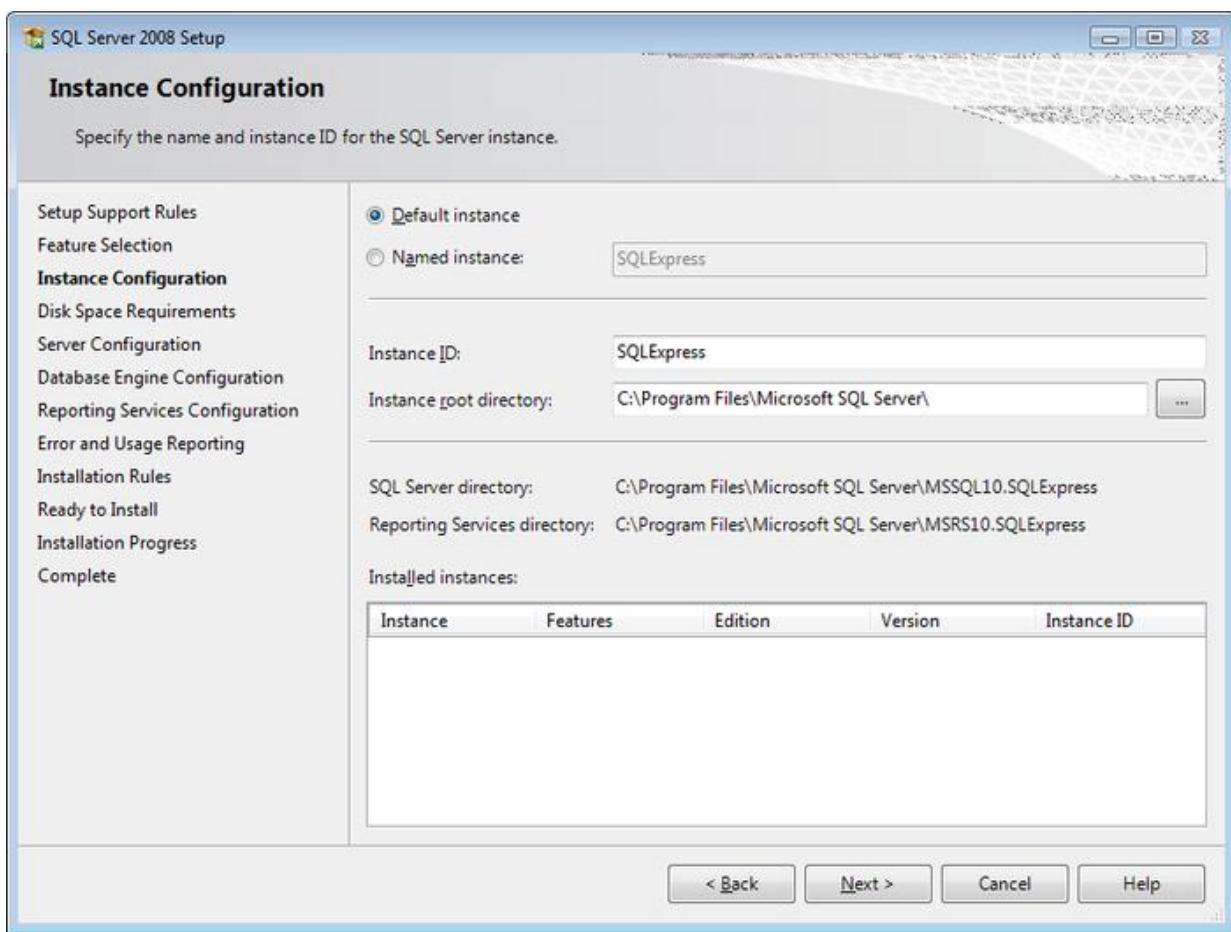


собственно установке SQL Express. Вначале устанавливаются вспомогательные файлы для процесса установки, затем в обычном формате Windows Installer будет предложено выбрать устанавливаемые опции:



**⚠ Примечание:** Исполнительный механизм SQL Express, обслуживающий базы данных и выполняющий запросы, оформлен как сервис операционной системы. На одну машину можно поставить до 16-ти экземпляров SQL Express. Это будет 16 независимых сервисов, каждый из которых ведает своими базами данных. Каждый сервис должен иметь свое уникальное имя.

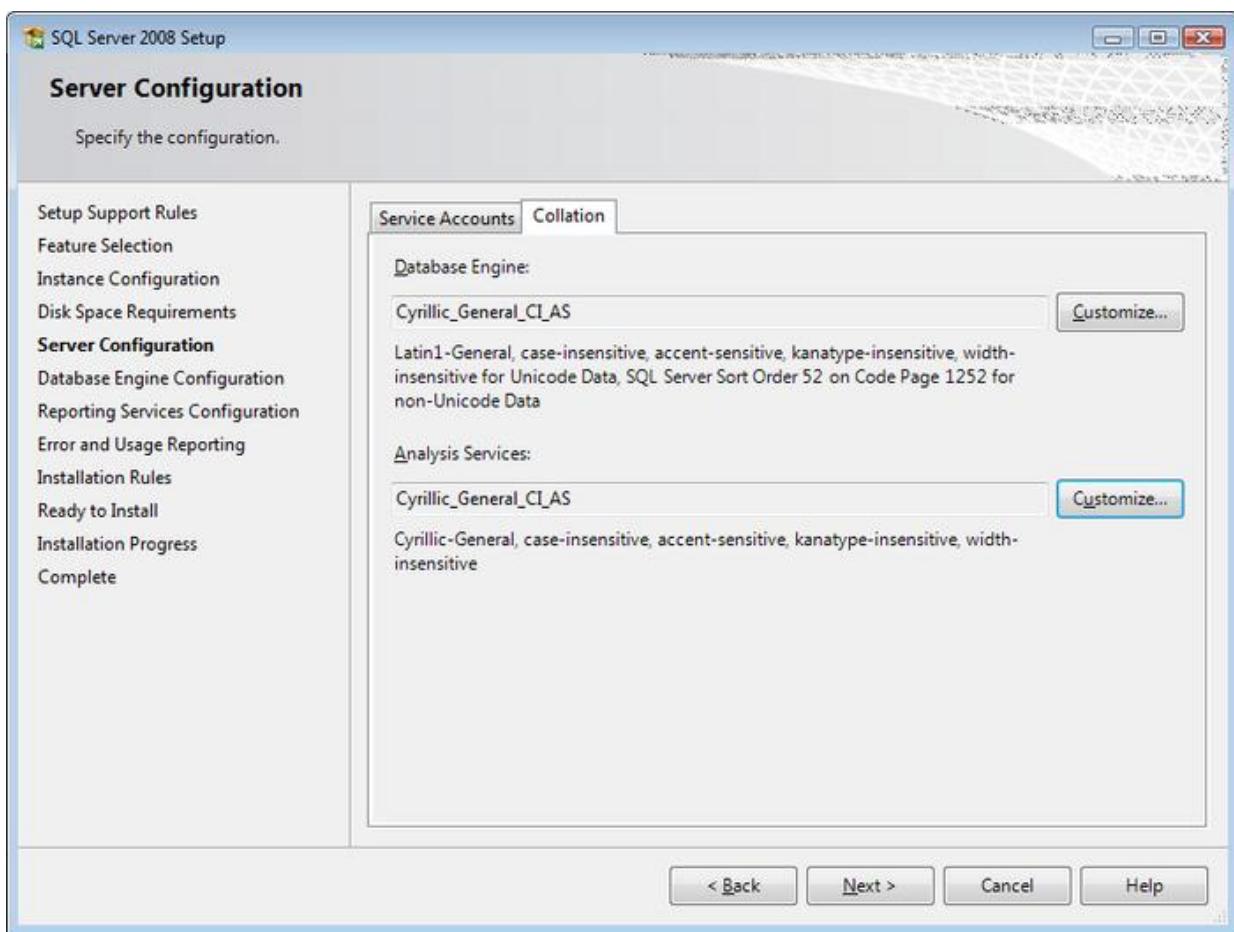
Если на машине не установлены SQL Server'a, то экземпляр можно ставить по умолчанию – поле **Default Instance**:



Тогда его имя будет совпадать с именем машины. Последующие экземпляры надо будет именовать, это осуществляется в поле **Named instance**. К именованному экземпляру обращаются как <имя машины>\<имя экземпляра>.

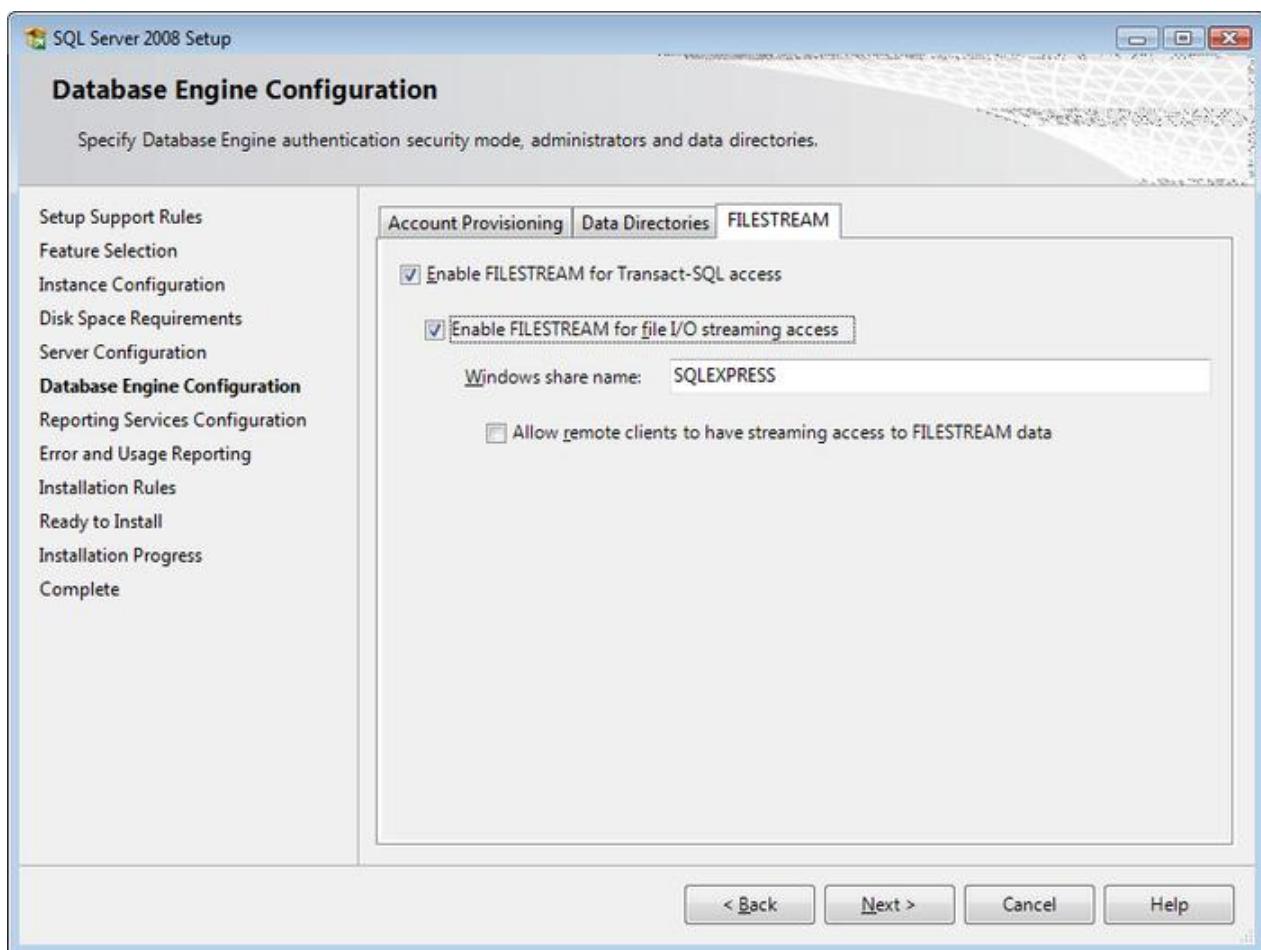
**! Примечание:** Размер одной БД в SQL Express ограничен 4 гигабайтами. SQL Express понимает только 1 процессор (под процессором понимается исполнительная сущность, например, ядро) и умеет работать не более, чем с 1 гигабайтом памяти. О других его ограничениях можно прочитать [здесь](#).

На приведенном ниже экране производится выбор юникодовской коллации, применяемой по умолчанию:

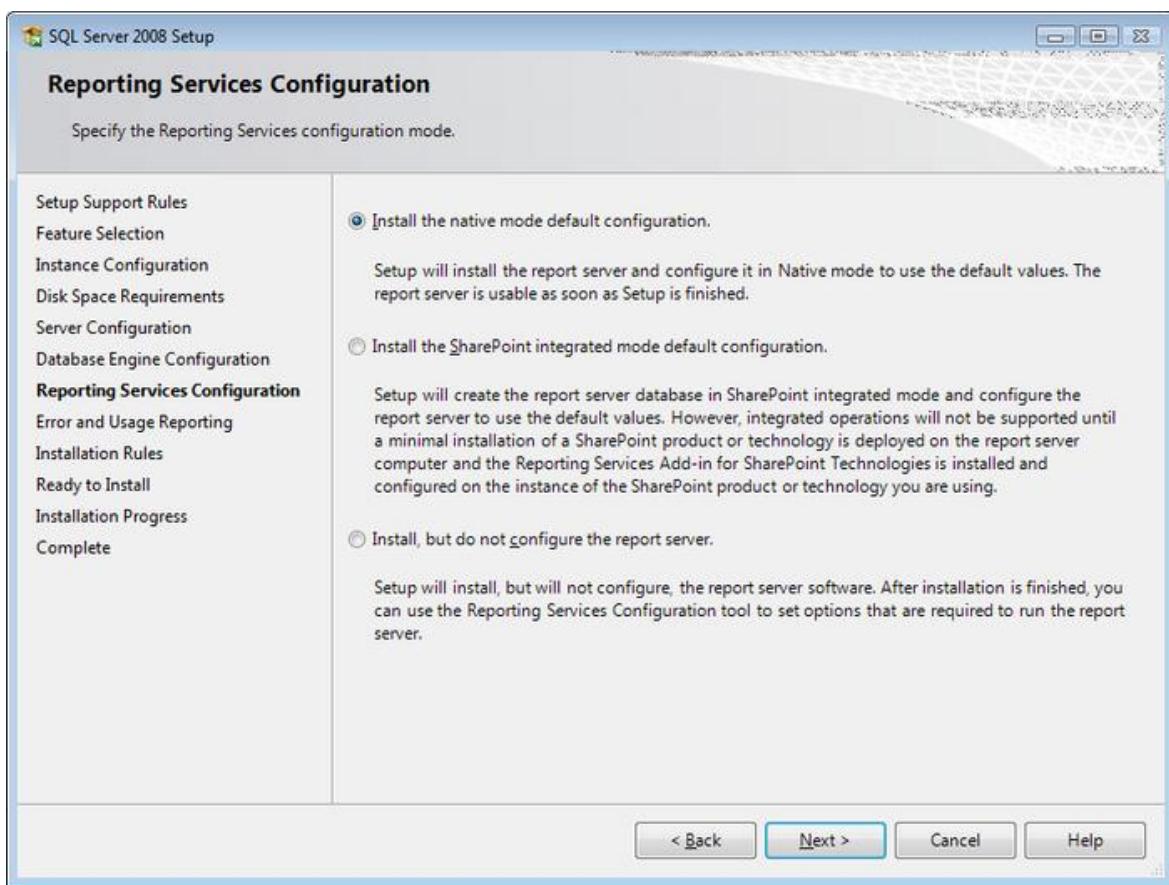


**Collation** (сопоставление) определяет ряд правил, согласно которым сортируются и сравниваются данные. Символьные данные сортируются, используя правила, которые определяют правильную последовательность символов, в зависимости от регистра, надстрочных знаков, символьных типов *kana* и ширины символов.

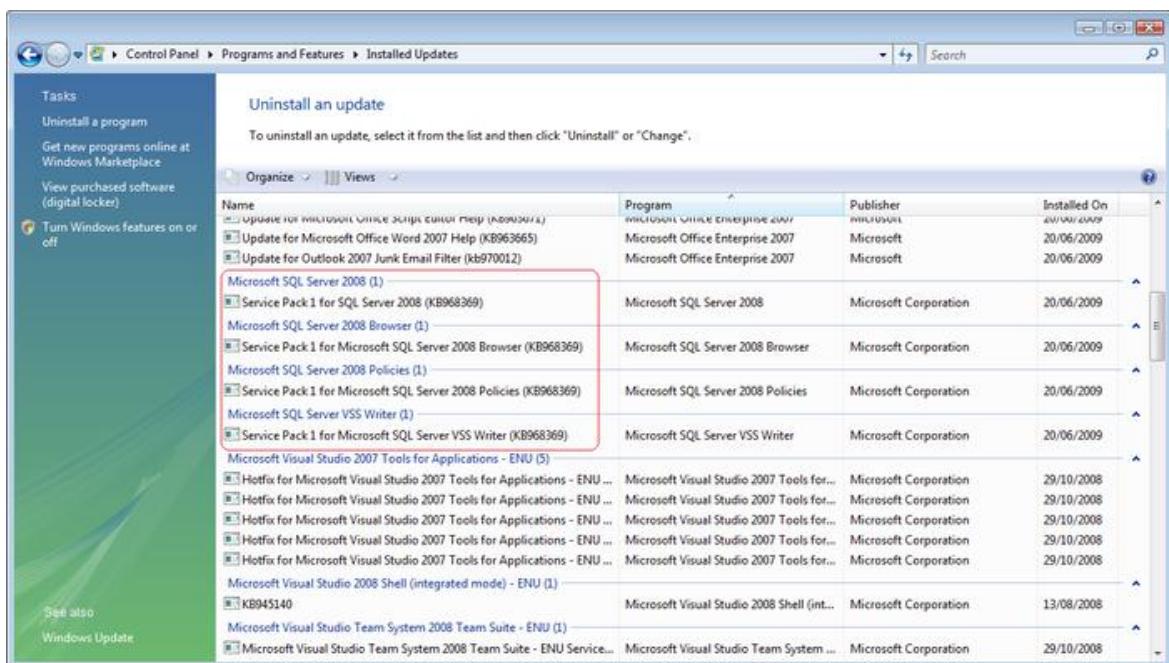
На приведенном ниже экране выполняется настройка атрибута **filestream**. Этот атрибут применяется к большим бинарным объектам и означает, что физически они будут храниться не в БД, а в файловой системе. Колонки таблиц **filestream** не попадают под действие упомянутого выше ограничения на размер БД в 4 гигабайта, их размер ограничен только размерами тома файловой системы.



Экран **Reporing Services Configuration** появляется в том случае, если происходит установка *SQL Express* в комплектации №3. На нем предлагается задать, будут ли службы отчетности установлены в режиме интеграции с *Sharepoint*. Оставьте на этом экране все без изменений.



После этого мастер установки начнет собственно процесс установки SQL Express. По его завершении обратитесь к службе Windows Update ([Пуск > Панель управления > Windows Update](#)) чтобы проверить, какие последние обновления доступны для SQL Server 2008 Express. Если такие обновления имеются, то рекомендуется их установить. После установки обновлений Microsoft SQL Server 2008 Express готов к работе.





## Установка 1С-Битрикс в варианте SQL Express

После установки базы *SQL Express* необходимо установить «1С-Битрикс: Управление сайтом» на установленную базу. Инсталляцию на *MySQL* при этом убирать не нужно, т.к. эта база данных содержит текущий контент сайта, который требуется перенести в *SQL Express*. В связи с постановкой задачи установка «1С-Битрикс: Управление сайтом» имеет свои особенности.

### **Начало установки**

В любом файловом менеджере перейдите в папку, где был установлен *Bitrix Environment* (в рамках локальной установки по умолчанию это будет - C:\Program Files\Bitrix Environment).

- Переименуйте подпапку /www в любую другую.
- Запустите заново процесс установки «1С-Битрикс: Управление сайтом».
- На этапе установки *Bitrix Environment* откажитесь от его установки.
- На этапе выбора места установки самого «1С-Битрикс: Управление сайтом» укажите путь C:\Program Files\Bitrix Environment\www.
- Продолжите установку программы до шага выбора базы данных. На этом шаге выберите базу *Microsoft SQL Server*.

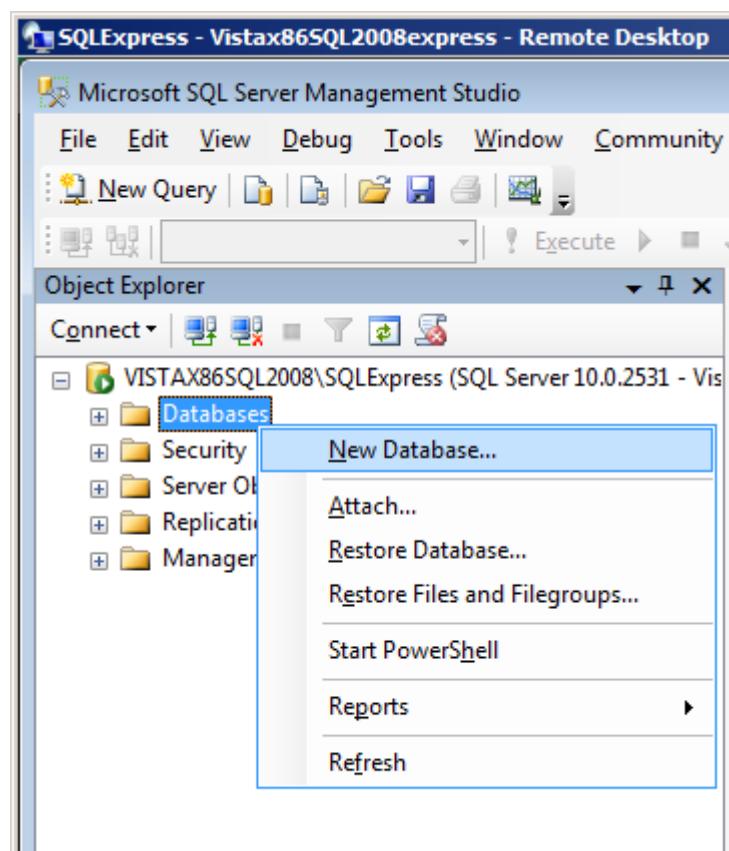
### **Создание базы данных**

Теперь нам потребуется создать для «1С-Битрикс: Управление сайтом» базу данных на *SQL Express*.

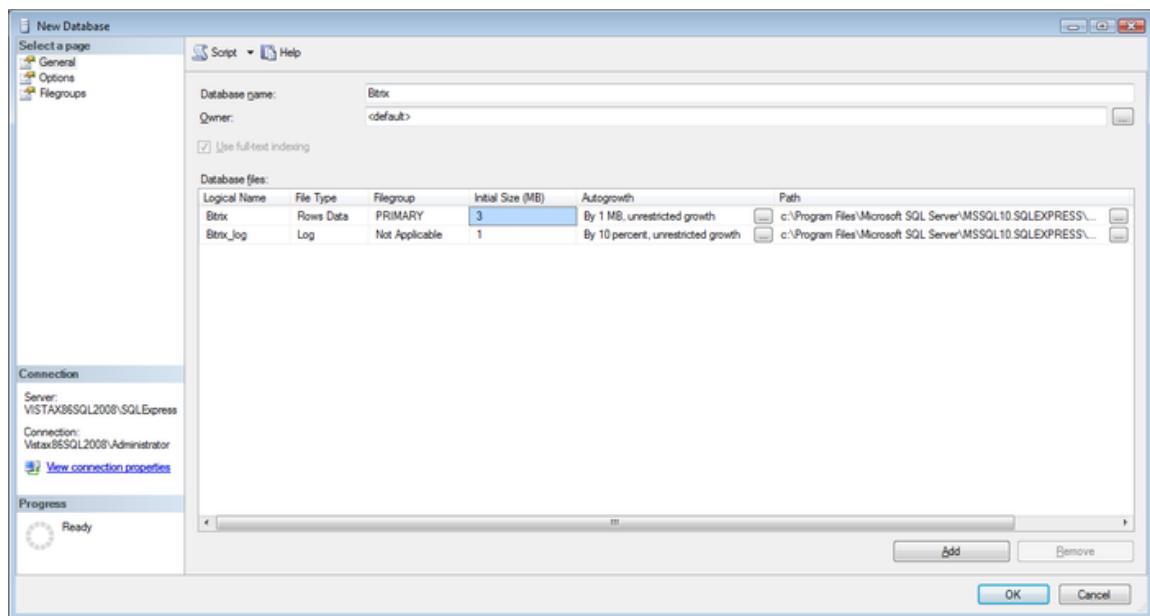
- Выполните команду *Пуск > Все программы > SQL Server 2008 > SQL Server Management Studio*.
- Соединитесь с экземпляром *SQL Express*:



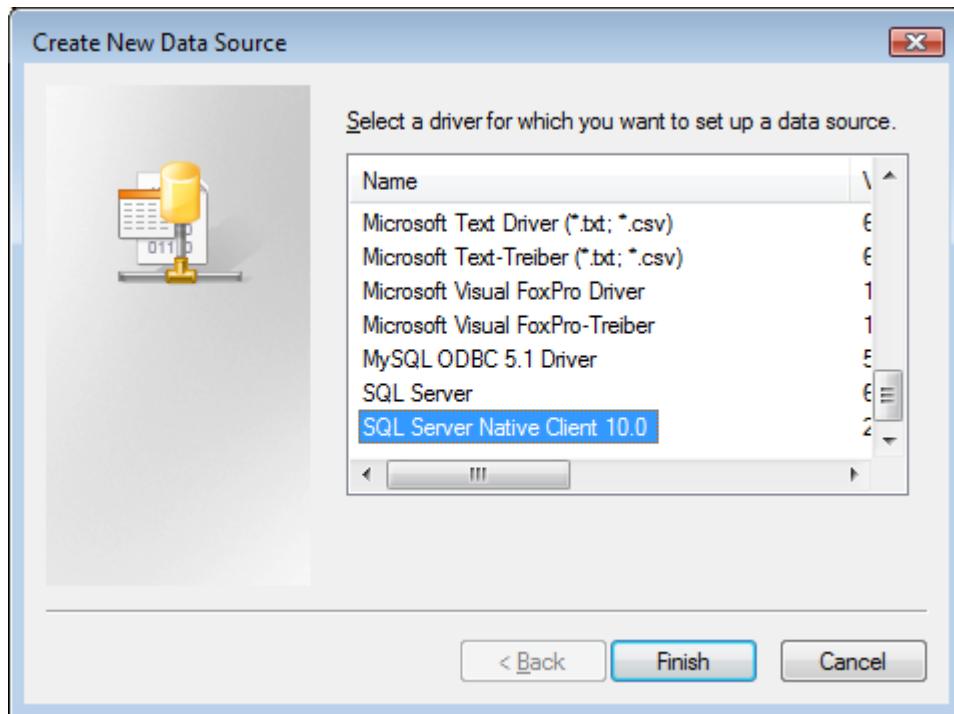
- Кликните правой кнопкой по папке **Databases** в Object Explorer и выберите пункт **New Database**:



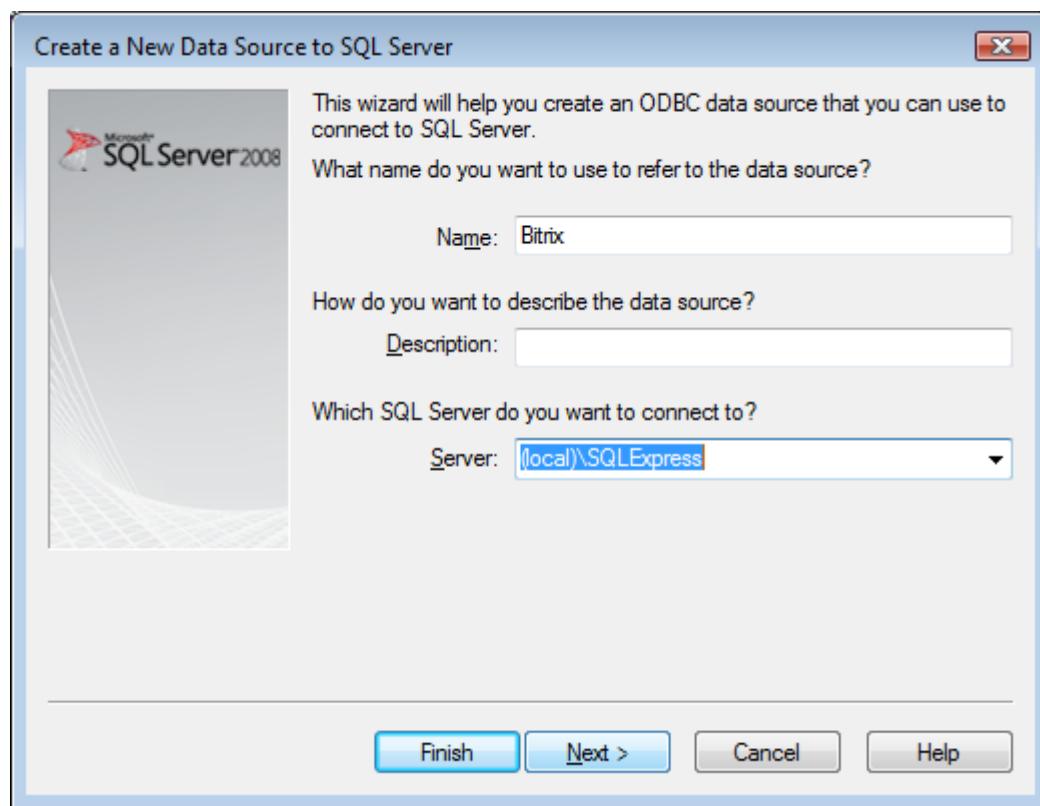
- В диалоге создания базы введите ее имя, оставив все остальное так, как оно предлагается по умолчанию. (В нашем Примере это имя – **Bitrix**). Нажмите внизу **OK**.



- Выполните команду Control Panel > Administrative Tools > Data Sources (ODBC).
- Перейдите на закладку **System DSN** и нажмите справа кнопку **Add**. Появится список драйверов. Из списка драйверов выбираем **SQL Server Native Client 10.0** как наиболее свежий на момент написания **Примера**:



- Введите название источника и имя сервера, к которому устанавливать соединение:



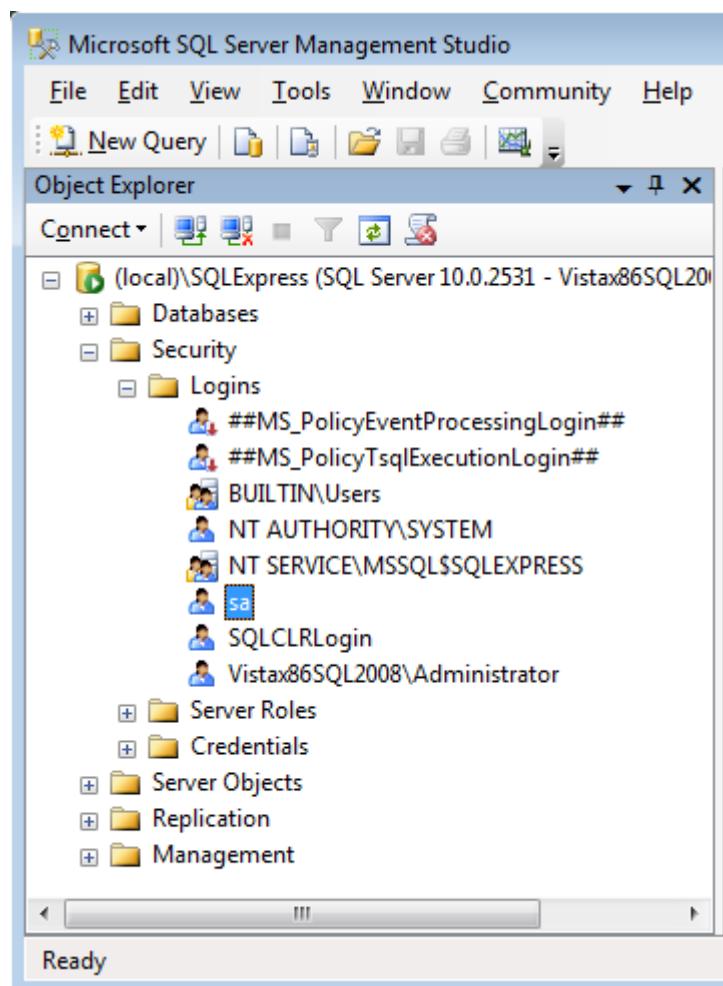
**⚠ Примечание:** Напомним, что имя сервера состоит из имени\_машины\имени\_экземпляра, которое задается SQL Server при установке. Если SQL Server ставился как экземпляр по умолчанию, достаточно просто имени машины. Если **SQL Browser Service** выключен, а порт используется нестандартный, то нужно в имени указать и порт.

**⚠ Примечание:** Создать соединение можно и по этому [примеру](#). На остальных шагах создания базы данных сохраняются настройки по умолчанию.

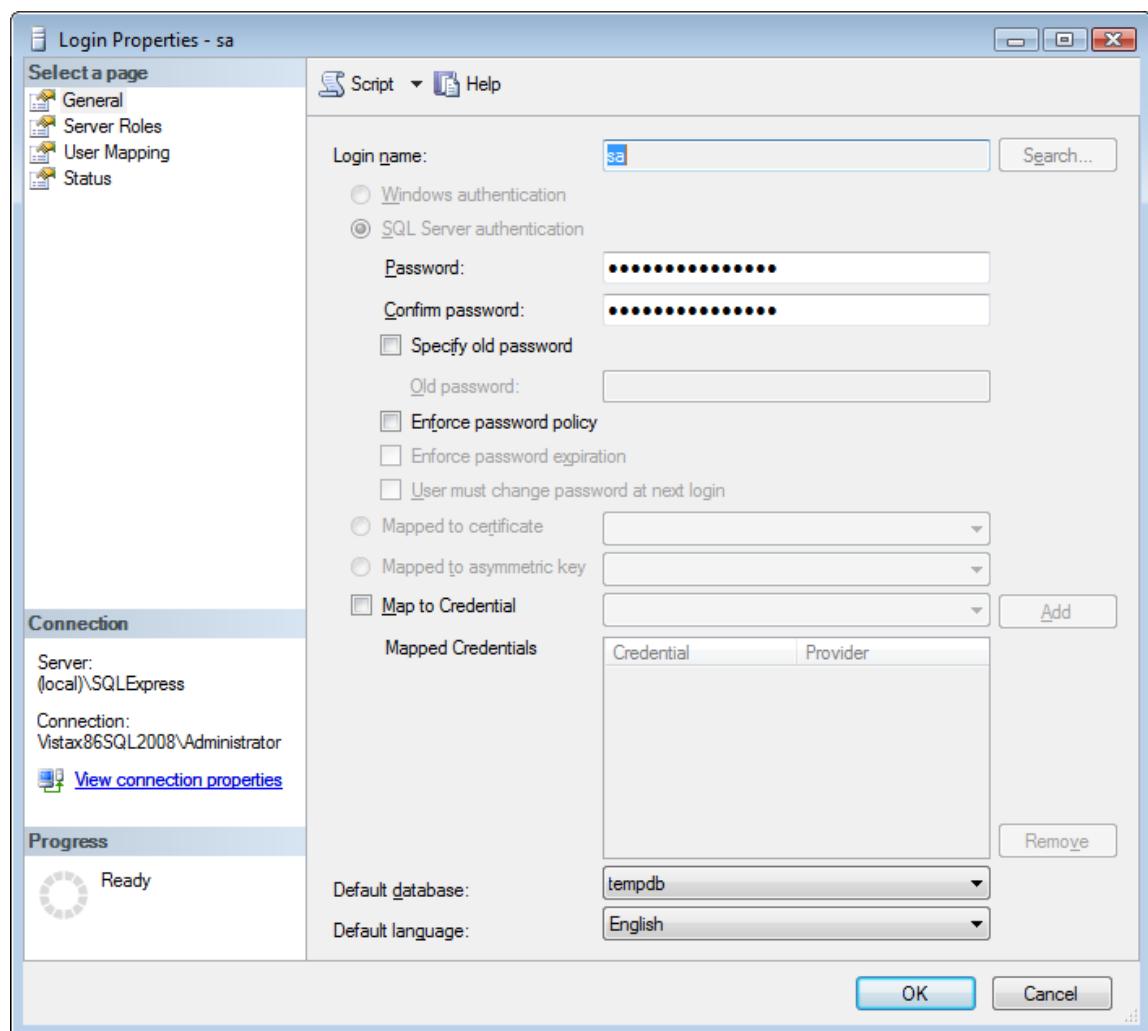
## Задание логина и пароля

Необходимо задать логин и пароль администратора SQL Server.

- Выполните команду [Пуск > Все программы > SQL Server 2008 > SQL Server Management Studio](#).
- В **Object Explorer** раскройте папку **Security\Logins**, выберите логин **sa**



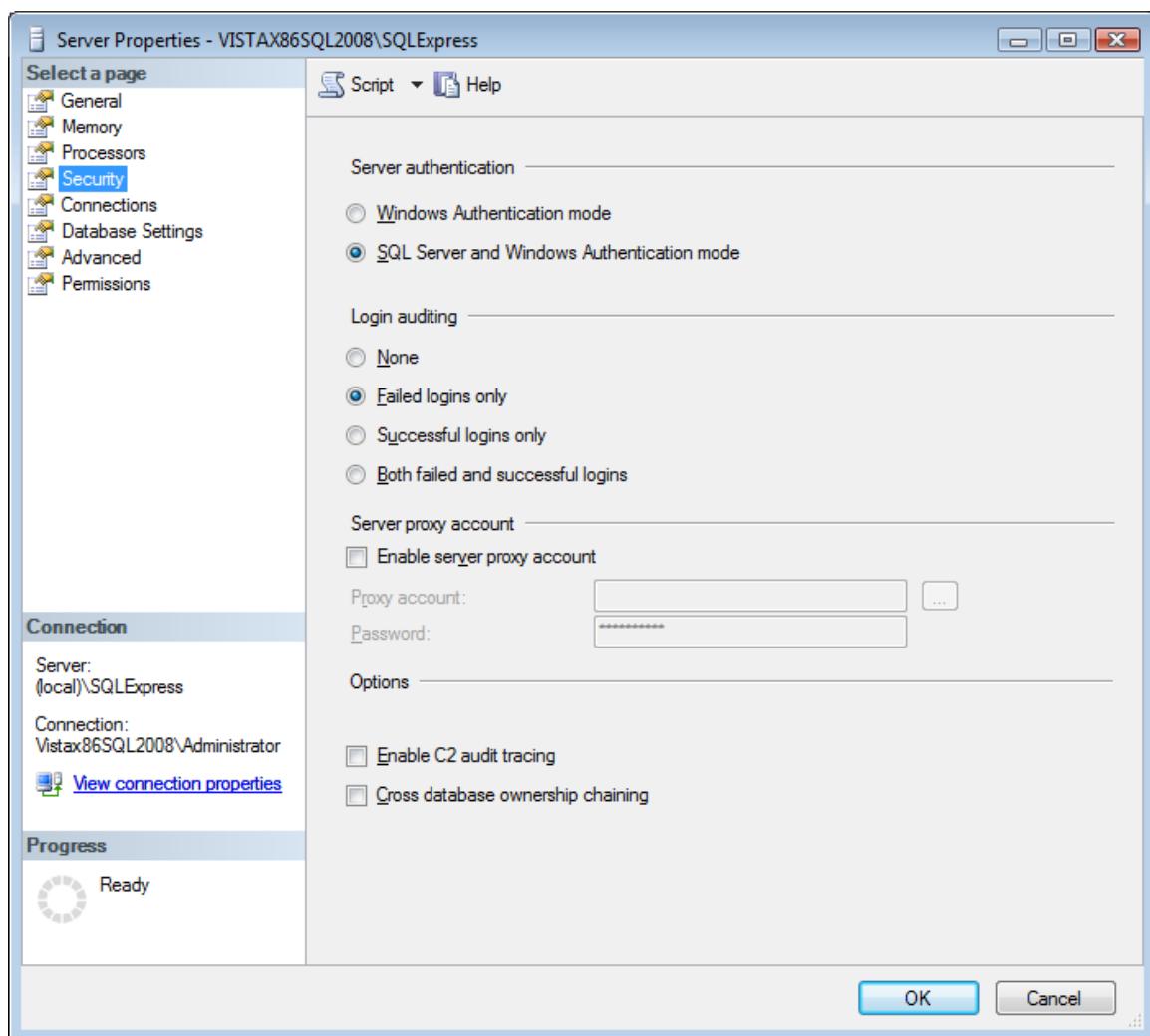
- Дважды кликните на нем. В открывшемся окне введите пароль и подтвердите его:



- Нажмите **OK**. Диалог закроется.

Если логин **sa** не отображается в общем списке, то:

- В верхней строчке **Object Explorer** вызовите через контекстное меню свойства созданного сервера.
- Перейдите на раздел **Security** и отметьте поле **SQL Server and Windows Authentication mode**:



- Логин **sa** появится.

### Завершение установки

Продолжите установку «1С-Битрикс: Управление сайтом». В поле **DSN** используйте название созданного вами источника **ODBC**. В качестве **Базы данных** используем созданную только что базу.

Создайте нового пользователя для указанной базы, используя логин и пароль администратора и завершите установку продукта.

### Сравнение структур баз

На данный момент имеется старый контент сайта в БД **bsm\_demo** на *MySQL* и новый, но пустой сайт в БД **bitrix** на *SQL Express*. Перед переносом данных необходимо выяснить тождественность структуры баз в случае *MySQL* и *SQL Express*.



## Создание сервера

В примере используется прилинкованный сервер со стороны SQL Express на MySQL, который использует MSDASQL (OLE DB поверх ODBC).

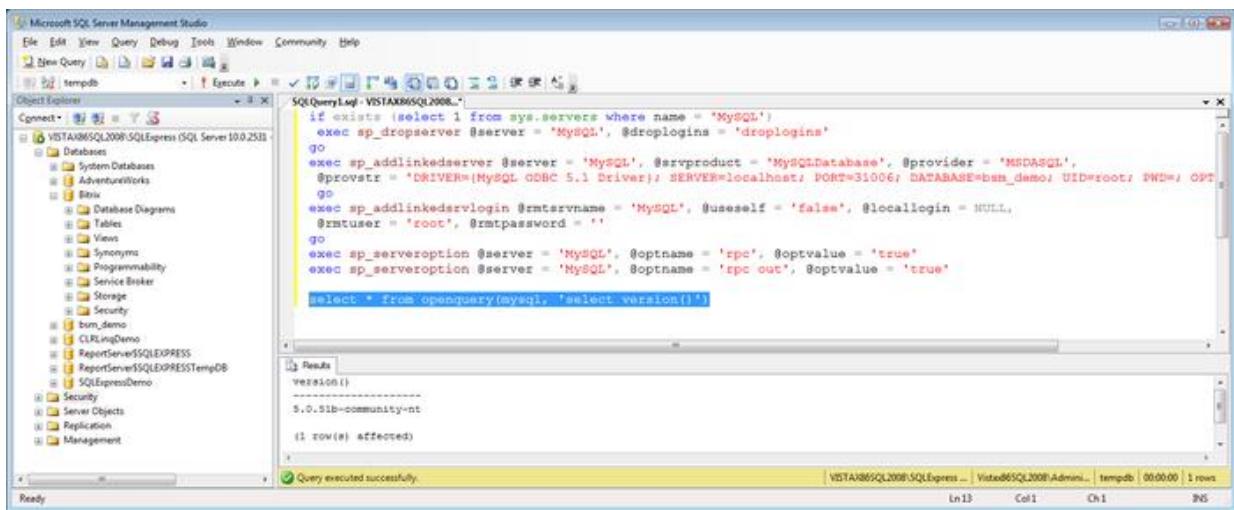
**⚠ Примечание:** Сервер Microsoft позволяет посыпать запросы не только к «родной» базе SQL Express, но и к базе MySQL. Для этого необходимо использовать специальный ODBC-драйвер.

**⚠ Создание сервера:**

- [Загрузите](#) ODBC-драйвер для MySQL 5.1.
- Установите драйвер в систему.
- Выполните команду Пуск > Все программы > SQL Server 2008 > SQL Server Management Studio.
- Создайте прилинкованный сервер без создания DSN:

```
if exists (select 1 from sys.servers where name = 'MySQL')
exec sp_dropserver @server = 'MySQL', @droplogins = 'droplogins'
go
exec sp_addlinkedserver @server = 'MySQL', @srvproduct = 'MySQLDatabase',
@provider = 'MSDASQL',
@provstr = 'DRIVER={MySQL ODBC 5.1 Driver}; SERVER=localhost; PORT=31006;
DATABASE=bsm_demo; UID=root; PWD=; OPTION=3'
go
exec sp_addlinkedsrvlogin @rmtsrvname = 'MySQL', @useself = 'false', @locallogin =
NULL,
@rmtuser = 'root', @rmtpassword = ''
go
exec sp_serveroption @server = 'MySQL', @optname = 'rpc', @optvalue = 'true'
exec sp_serveroption @server = 'MySQL', @optname = 'rpc out', @optvalue = 'true'
```

Теперь сервер позволяет выполнять на MySQL запросы, адресованные к SQL Server, и возвращать результат MySQL, как если бы это был результат SQL Server.



## Состав таблиц

В первую очередь необходимо выяснить, насколько отличаются базы по составу таблиц. Данный запрос выводит несовпадения:

```

use bitrix

with
tbl_rows_sqlsrv as (
select t.name, p.n from sys.tables t
join (select object_id, sum(row_count) n from sys.dm_db_partition_stats where index_id in (0, 1) group by object_id) p
on t.object_id = p.object_id
)
, tbl_rows_mysql as (
select * from openquery(mysql, 'select table_name name, table_rows n from information_schema.tables where table_schema = "bsm_demo"')
)
select sqlsrv.name, sqlsrv.n, mysql.name, mysql.n from tbl_rows_sqlsrv sqlsrv full outer join
tbl_rows_mysql mysql on sqlsrv.name = mysql.name
where sqlsrv.name is null or mysql.name is null or sqlsrv.n <> mysql.n
name n      name n
B_OPTION    112    b_option     113
B_STAT_SESSION_DATA   0      b_stat_session_data  1
B_FILE_ACTION    0      NULL    NULL
B_POSTING_LOCK   0      NULL    NULL
B_FAVORITE_LANG  0      NULL    NULL

```



**Выход.** Все 319 таблиц в базе при инсталляции MySQL имеют соответствия (т.е. таблицы с тем же именем) в инсталляции на SQL Server.

В инсталляции на SQL Server имеются 3 таблицы, не имеющих соответствия в инсталляции на MySQL:

- B\_FILE\_ACTION
- B\_POSTING\_LOCK
- B\_FAVORITE\_LANG

Это благоприятная ситуация для миграции, поскольку перенос данных планируется именно из MySQL в SQL Server. Хуже, если бы, наоборот, в MySQL имелись таблицы, которые бы было непонятно куда переносить в SQL Server.

Из 319 таблиц инсталляции MySQL две не совпадают по числу строк с соответствующими им таблицами в инсталляции на SQL Server:

SQL Server		MySQL	
B_OPTION	112	b_option	113
B_STAT_SESSION_DATA	0	b_stat_session_data	1

Этот факт имеет, роль комментария, поскольку в случае **Примера** инсталляции «1С-Битрикс: Управление сайтом» как в случае MySQL, так и SQL Server - "чистые", то есть без данных. В реальной жизни клиент уже будет работать продолжительный период с MySQL'ной базой, поэтому данных там будет больше, чем в только что установленной базе SQL Server.

Речь не идет о каком-то слиянии. Просто данные из MySQL требуется перенести в соответствующие таблицы SQL Server, перетерев все, что туда уже успел добавить процесс инсталляции CMS. Инсталляция «1С-Битрикс: Управление сайтом» в варианте SQL Server создает, по сути, готовые структуры для приема данных со стороны SQL Server и избавляет нас от необходимости рассматривать миграцию метаданных.

## Набор полей

Необходимо исследовать наборов полей в соответствующих таблицах: какие поля есть в таблице MySQL, которых нет в таблице SQL Server и наоборот.

### **Собственно набор полей**

```
use bitrix
```



```
with
col_sqlsrv as (
select t.name as tbl_name, c.name as col_name from sys.columns c join sys.tables t on
c.object_id = t.object_id
)
, col_mysql as (
select * from openquery(mysql, 'select table_name tbl_name, column_name col_name from
information_schema.columns where table_schema = "bsm_demo"')
)
select * from col_sqlsrv sqlsrv full outer join col_mysql mysql
on sqlsrv.tbl_name = mysql.tbl_name and sqlsrv.col_name = mysql.col_name
where sqlsrv.col_name is null or mysql.col_name is null
```

**Вывод.** Сравнение выявило абсолютное тождество наборов полей в соответствующих таблицах. Исключением являются три таблицы в SQL Server, которых нет в варианте установки MySQL.

### Порядковый номер

Также существенным фактором является совпадение порядкового номера одноименной колонки, потому что если это так, процесс переноса данных будет проще. В противном случае придется явно перечислять поля в списке в нужном порядке или строить соответствие полей источника с полями назначения.

```
with
col_sqlsrv(tbl_name, col_name, col_pos) as (
select object_name(object_id), name, column_id from sys.columns
)
, col_mysql(tbl_name, col_name, col_pos) as (
select * from openquery(mysql, 'select table_name, column_name, ordinal_position from
information_schema.columns where table_schema = "bsm_demo"')
)
select * from col_sqlsrv sqlsrv join col_mysql mysql
on sqlsrv.tbl_name = mysql.tbl_name and sqlsrv.col_name = mysql.col_name
where sqlsrv.col_pos <> mysql.col_pos
```

**Вывод.** Порядковый номер колонки не совпадает в 80 случаях, что необходимо учитывать при переносе данных.

### Соответствие типов и длины полей



Необходимо выяснить соответствие типов и длины в одноименных колонках. Этим скриптом можно получить типы колонок, использующиеся «1С-Битрикс: Управление сайтом» в случае установки на MySQL и на SQL Server:

```
select * from openquery(mysql, 'select distinct data_type from information_schema.columns  
where table_schema = "bsm_demo"')  
  
select distinct type_name(c.user_type_id) from sys.columns c join sys.tables t on c.object_id =  
t.object_id where type = 'U'
```

Вывод. В *SQL Server* типов колонок - 11, в *MySQL* – 16. Например, в случае инсталляции *MySQL* используется тип **smallint**, а в варианте инсталляции *SQL Server* он не используется. В *MySQL* есть разновидности блобовских типов, которых действительно нет в *SQL Server*, например, **mediumtext**, **longtext**. В случае *SQL Server* им соответствует один тип **text**.

«1С-Битрикс: Управление сайтом» в *SQL Server* использует **text/image**, **datetime**. в *MySQL* – тип **date**. Необходимо вручную построить таблицу соответствия типов, т.е. какой тип *MySQL* в какой тип *SQL Server* можно без потерь переносить.

**⚠ Примечание:** Если существует экземпляр типа А, который не перенесется в тип Б без обрезания или дополнительных преобразований, такой перенос считается невозможным. Все типы и в *MySQL*, и в *SQL Server* можно разбить на числовые, строковые, бинарные и календарные. Перенос внутри каждой категории считается допустимым, при этом длина поля приемника должна быть не меньше, чем у источника, а, в случае численных полей с фиксированной точкой то же распространяется и на количество знаков после запятой.

Пример запроса, который проверяет нарушения этого правила:

```
with  
col_sqlsrv(tbl_name, col_name, col_type, col_len, col_prec, col_scal) as (  
select object_name(object_id), name, type_name(user_type_id), max_length, precision, scale  
from sys.columns  
)  
col_sqlsrv1(tbl_name, col_name, col_type, col_len, col_prec, col_scal) as (  
select tbl_name, col_name,  
case when col_type in ('bigint', 'int', 'tinyint', 'decimal', 'numeric') then 'N'  
when col_type in ('float') then 'F'  
when col_type in ('datetime') then 'D'  
when col_type in ('char', 'varchar', 'text') then 'C'  
when col_type in ('image') then 'B'  
end,  
case when col_type = 'text' then power(cast(2 as bigint), 31) - 1
```



```

    else col_len
  end,
  col_prec, col_scal from col_sqlsrv
)
, col_mysql(tbl_name, col_name, col_type, col_len, col_prec, col_scal) as (
select * from openquery(mysql, 'select table_name, column_name, data_type,
character_maximum_length, numeric_precision, numeric_scale
from information_schema.columns where table_schema = "bsm_demo"')
)
, col_mysql1(tbl_name, col_name, col_type, col_len, col_prec, col_scal) as (
select tbl_name, col_name,
case when col_type in ('bigint', 'int', 'tinyint', 'decimal', 'smallint') then 'N'
      when col_type in ('float') then 'F'
      when col_type in ('datetime', 'timestamp') then 'D'
      when col_type in ('char', 'varchar', 'text', 'mediumtext', 'longtext') then 'C'
      when col_type in ('longblob') then 'B' end,
  col_len, col_prec, col_scal from col_mysql
)
select sqlsrv.*, mysql.col_type, mysql.col_len, mysql.col_prec, mysql.col_scal from col_sqlsrv1
sqlsrv join col_mysql1 mysql
on sqlsrv.tbl_name = mysql.tbl_name and sqlsrv.col_name = mysql.col_name
where mysql.col_type <> sqlsrv.col_type
      or mysql.col_type = sqlsrv.col_type and mysql.col_len > sqlsrv.col_len
      or mysql.col_type = sqlsrv.col_type and (mysql.col_prec > sqlsrv.col_prec or mysql.col_scal >
sqlsrv.col_scal)
order by 1, 2

```

Первое условие в **where** не нарушается, что означает, что с преобразованием типов проблем нет – числовые типы переносятся в числовые, текстовые в текстовые и т.д. Однако выявлено 128 колонок, нарушающих второе и третье условие в **where**. Это означает, что при переносе данных возможно возникновение ошибки из-за недостаточного размера поля приемника.

Таблица	Колонка	SQL Express				MySQL			
		Категория типа	Размер поля	Численная длина	После запятой	Категория типа	Размер поля	Численная длина	После запятой
B_ADV_BANNER	COMMENTS	C	1000	0	0	C	65535	NULL	NUL



1С·БИТРИКС

Компания «1С-Битрикс» Системы управления веб-проектами

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

B_ADV_BANNER	KEYWORDS	C	1000	0	0	C	65535	NULL	NUL L
B_ADV_BANNER	STATUS_COMM ENTS	C	500	0	0	C	65535	NULL	NUL L
B_ADV_BANNER	URL	C	8000	0	0	C	65535	NULL	NUL L
B_ADV_CONTRAC T	ADMIN_COMM ENTS	C	500	0	0	C	65535	NULL	NUL L
B_ADV_CONTRAC T	DESCRIPTION	C	2000	0	0	C	65535	NULL	NUL L
B_ADV_CONTRAC T	KEYWORDS	C	1000	0	0	C	65535	NULL	NUL L
B_ADV_TYPE	DESCRIPTION	C	500	0	0	C	65535	NULL	NUL L
B_CATALOG_LOA D	VALUE	C	2000	0	0	C	65535	NULL	NUL L
B_EVENT	C_FIELDS	C	21474 83647	0	0	C	42949 67295	NULL	NUL L
B_EVENT_LOG	REQUEST_URI	C	2000	0	0	C	65535	NULL	NUL L
B_EVENT_LOG	USER_AGENT	C	2000	0	0	C	65535	NULL	NUL L
B_EVENT_TYPE	LID	C	2	0	0	C	201	NULL	NUL L
B_FAVORITE	COMMENTS	C	8000	0	0	C	65535	NULL	NUL L
B_FAVORITE	URL	C	8000	0	0	C	65535	NULL	NUL L
B_FORM	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL L
B_FORM	FILTER_RESUL T_TEMPLATE	C	8000	0	0	C	65535	NULL	NUL L
B_FORM	TABLE_RESULT _TEMPLATE	C	8000	0	0	C	65535	NULL	NUL L
B_FORM_2_GROU P	PERMISSION	N	1	3	0	N	NULL	10	0
B_FORM_ANSWER	FIELD_PARAM	C	8000	0	0	C	65535	NULL	NUL L
B_FORM_ANSWER	MESSAGE	C	8000	0	0	C	65535	NULL	NUL L
B_FORM_FIELD	COMMENTS	C	8000	0	0	C	65535	NULL	NUL L
B_FORM_FIFI D	FII TFR TITI F	C	8000	0	0	C	65535	NULL	NII II



L										
B_FORM_FIELD	RESULTS_TABLE_TITLE	C	8000	0	0	C	65535	NULL	NUL	L
B_FORM_FIELD	TITLE	C	8000	0	0	C	65535	NULL	NUL	L
B_FORM_RESULT_ANSWER	ANSWER_TEXT	C	8000	0	0	C	65535	NULL	NUL	L
B_FORM_RESULT_ANSWER	ANSWER_TEXT_SEARCH	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_FORM_RESULT_ANSWER	ANSWER_VALUE_SEARCH	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_FORM_RESULT_ANSWER	USER_TEXT	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_FORM_RESULT_ANSWER	USER_TEXT_SEARCH	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_FORM_STATUS	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL	L
B_FORUM	DESCRIPTION	C	1000	0	0	C	65535	NULL	NUL	L
B_FORUM_FILTER	USE_IT	C	1	0	0	C	50	NULL	NUL	L
B_FORUM_MESSAGE	ID	N	4	10	0	N	NULL	19	0	
B_FORUM_MESSAGE	TOPIC_ID	N	4	10	0	N	NULL	19	0	
B_FORUM_PRIVATE_MESSAGE	ID	N	4	10	0	N	NULL	19	0	
B_FORUM_PRIVATE_MESSAGE	IS_READ	C	1	0	0	C	50	NULL	NUL	L
B_FORUM_PRIVATE_MESSAGE	USE_SMILES	C	1	0	0	C	50	NULL	NUL	L
B_FORUM_STAT	ID	N	4	10	0	N	NULL	19	0	
B_FORUM_SUBSCRIBE	NEW_TOPIC_ONLY	C	1	0	0	C	50	NULL	NUL	L
B_FORUM_TOPIC	ID	N	4	10	0	N	NULL	19	0	
B_FORUM_TOPIC	LAST_MESSAGE_ID	N	4	10	0	N	NULL	19	0	
B_FORUM_TOPIC	TOPIC_ID	N	4	10	0	N	NULL	19	0	
B_FORUM_USER	ID	N	4	10	0	N	NULL	19	0	
B_FORUM_USER_TOPIC	ID	N	4	10	0	N	NULL	19	0	
BIRILOCKFIELD	DETAILTEXT	C	21474	0	0	C	42949	NIII	NII	



NT				83647				67295		L
B_IBLOCK_ELEMENT	PREVIEW_TEXT	C	2000	0	0	C	65535	NULL	NUL	L
B_IBLOCK_ELEMENT_PROPERTY	VALUE	C	2000	0	0	C	65535	NULL	NUL	L
B_IBLOCK_FIELDS	DEFAULT_VALUE	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_LDAP_SERVER	DESCRIPTION	C	5000	0	0	C	65535	NULL	NUL	L
B_LDAP_SERVER	FIELD_MAP	C	2000	0	0	C	65535	NULL	NUL	L
B_LEARN_CHAPTER	DETAIL_TEXT	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_LEARN_LESSON	DETAIL_TEXT	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_LIST_RUBRIC	DESCRIPTION	C	2000	0	0	C	65535	NULL	NUL	L
B_MAIL_FILTER	ACTION_VARS	C	5000	0	0	C	65535	NULL	NUL	L
B_MAIL_FILTER	DESCRIPTION	C	2000	0	0	C	65535	NULL	NUL	L
B_MAIL_FILTER_COND	STRINGS	C	5000	0	0	C	65535	NULL	NUL	L
B_MAIL_MAILBOX	DESCRIPTION	C	5000	0	0	C	65535	NULL	NUL	L
B_MAIL_MESSAGE	BODY	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_MAIL_MESSAGE	FULL_TEXT	C	21474 83647	0	0	C	42949 67295	NULL	NUL	L
B_MAIL_MSG_ATTACHMENT	FILE_DATA	B	16	0	0	B	42949 67295	NULL	NUL	L
B_PERF_ERROR	ERRFILE	C	2000	0	0	C	65535	NULL	NUL	L
B_PERF_ERROR	ERRSTR	C	2000	0	0	C	65535	NULL	NUL	L
B_PERF_HIT	REQUEST_URI	C	2000	0	0	C	65535	NULL	NUL	L
B_PERF_HIT	SCRIPT_NAME	C	2000	0	0	C	65535	NULL	NUL	L
B_PERF_SQL	COMPONENT_NAME	C	2000	0	0	C	65535	NULL	NUL	L
B_PERF_SQL	MODULE_NAME	C	2000	0	0	C	65535	NULL	NUL	L



b_search_content	PARAM1	C	1000	0	0	C	65535	NULL	NUL L
b_search_content	PARAM2	C	1000	0	0	C	65535	NULL	NUL L
b_search_content	SEARCHABLE_ CONTENT	C	21474 83647	0	0	C	42949 67295	NULL	NUL L
b_search_custom_r ank	PARAM1	C	2000	0	0	C	65535	NULL	NUL L
b_search_custom_r ank	PARAM2	C	2000	0	0	C	65535	NULL	NUL L
B_SEC_SESSION	SESSION_DATA	C	21474 83647	0	0	C	42949 67295	NULL	NUL L
B_STAT_ADV	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_EVENT	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_EVENT_LI ST	REDIRECT_URL	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_EVENT_LI ST	REFERER_URL	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_EVENT_LI ST	URL	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	FIRST_URL_FR OM	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	FIRST_URL_TO	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	LAST_CITY_INF O	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	LAST_COOKIE	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	LAST_URL_LAS T	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_GUEST	LAST_USER_A GENT	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_HIT	COOKIES	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_HIT	URL	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_HIT	URL_FROM	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_HIT	USER_AGENT	C	8000	0	0	C	65535	NULL	NUL L
B_STAT_PAGE	URI	C	2000	0	0	C	65535	NULL	NUL L



L										
B_STAT_PATH	PAGES	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_PATH_CACHE	PATH_PAGES	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_PHRASE_LIST	URL_FROM	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_PHRASE_LIST	URL_TO	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_REFERER_LIST	URL_FROM	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_REFERER_LIST	URL_TO	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SEARCHER	USER_AGENT	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SEARCHER_HIT	URL	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SEARCHER_HIT	USER_AGENT	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SESSION	URL_FROM	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SESSION	URL_LAST	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SESSION	URL_TO	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SESSION	USER_AGENT	C	8000	0	0	C	65535	NULL	NUL	L
B_STAT_SESSION_DATA	SESSION_DATA	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	COMMENTS	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	MESSAGE	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	URL_FROM	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	URL_REDIRECT	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	URL_TO	C	8000	0	0	C	65535	NULL	NUL	L
B_STOP_LIST	USER_AGENT	C	8000	0	0	C	65535	NULL	NUL	L
B_TICKET	AUTO_CLOSE_DAYS	N	1	3	0	N	NULL	10	0	L



B_TICKET	LAST_MESSAGE_SID	C	8000	0	0	C	65535	NULL	NUL
B_TICKET	OWNER_SID	C	8000	0	0	C	65535	NULL	NUL
B_TICKET	SUPPORT_COMMENTS	C	8000	0	0	C	65535	NULL	NUL
B_TICKET	TITLE	C	2000	0	0	C	65535	NULL	NUL
B_TICKET_DICTIO NARY	DESCR	C	8000	0	0	C	65535	NULL	NUL
B_TICKET_MESSA GE	EXTERNAL_FIE LD_1	C	8000	0	0	C	65535	NULL	NUL
B_TICKET_MESSA GE	MESSAGE	C	21474 83647	0	0	C	42949 67295	NULL	NUL
B_TICKET_MESSA GE	MESSAGE_SEA RCH	C	21474 83647	0	0	C	42949 67295	NULL	NUL
B_TICKET_MESSA GE	OWNER_SID	C	8000	0	0	C	65535	NULL	NUL
B_TICKET_SLA	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL
B_VOTE	DESCRIPTION	C	5000	0	0	C	65535	NULL	NUL
B_VOTE_ANSWER	FIELD_TYPE	N	1	3	0	N	NULL	10	0
B_VOTE_ANSWER	MESSAGE	C	5000	0	0	C	65535	NULL	NUL
B_VOTE_EVENT_A NSWER	MESSAGE	C	8000	0	0	C	65535	NULL	NUL
B_VOTE_QUESTION	QUESTION	C	5000	0	0	C	65535	NULL	NUL
B_WORKFLOW_D OCUMENT	COMMENTS	C	8000	0	0	C	65535	NULL	NUL
B_WORKFLOW_LOG	COMMENTS	C	8000	0	0	C	65535	NULL	NUL
B_WORKFLOW_ST ATUS	DESCRIPTION	C	8000	0	0	C	65535	NULL	NUL

**⚠ Примечание:** Стого говоря, следует исследовать тождественность признаков NULL у соответствующих полей и других ограничений. Например, если некоторое поле допускает NULLы в MySQL, но является NOT NULL в SQL Server, перенос данных может завершиться с ошибкой. Если сразу перейти к процессу переноса данных и такие несоответствия имеются, то они будут выявлены в ходе переноса.



## Перенос данных

Теперь можно приступать к собственно переносу данных. Однако использовать MySQL ODBC Connector 5.1 для переноса данных - не лучший вариант, поскольку есть ситуации, в которых его работоспособность нарушается. В числе таких ситуаций: отсутствие поддержки полей типа **longtext**.

```
select * from openquery(mysql, 'select DETAIL_TEXT from b_learn_lesson')
```

-----  
*Msg 0, Level 11, State 0, Line 0*

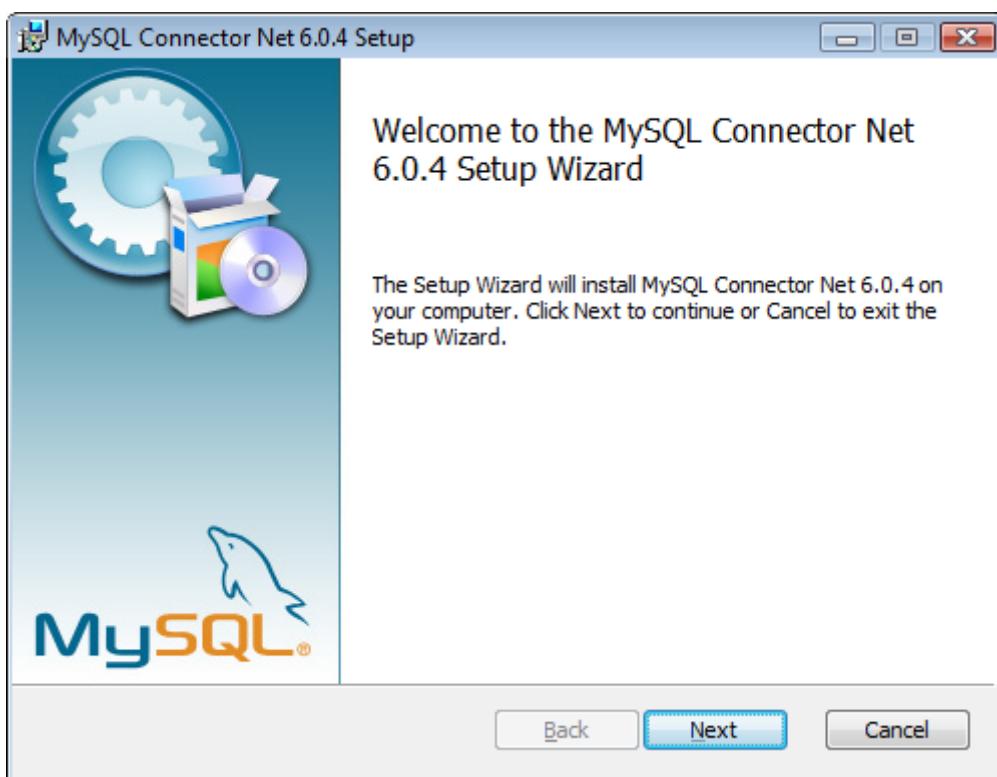
*A severe error occurred on the current command. The results, if any, should be discarded.*

Можно обеспечить перенос подстроками по 4000 символов, но такой способ нельзя признать оптимальным.

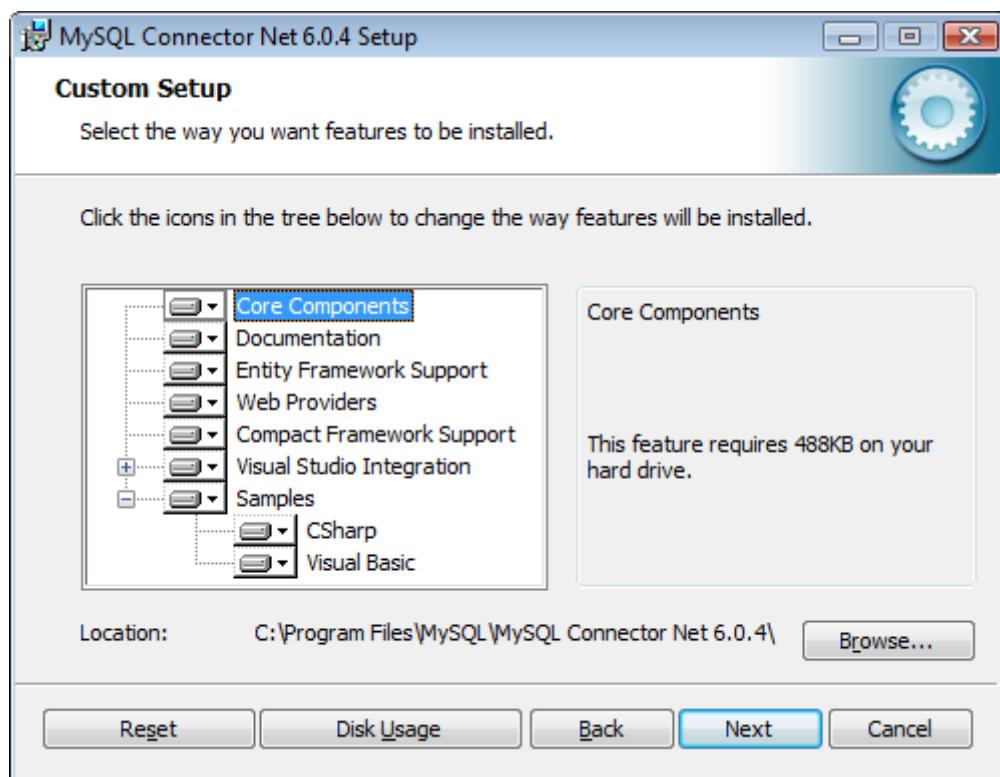
## Установка MySQL Connector/Net 6.0

Рекомендуется использовать для переноса данных MySQL Connector/Net 6.0. ([Скачать](#))

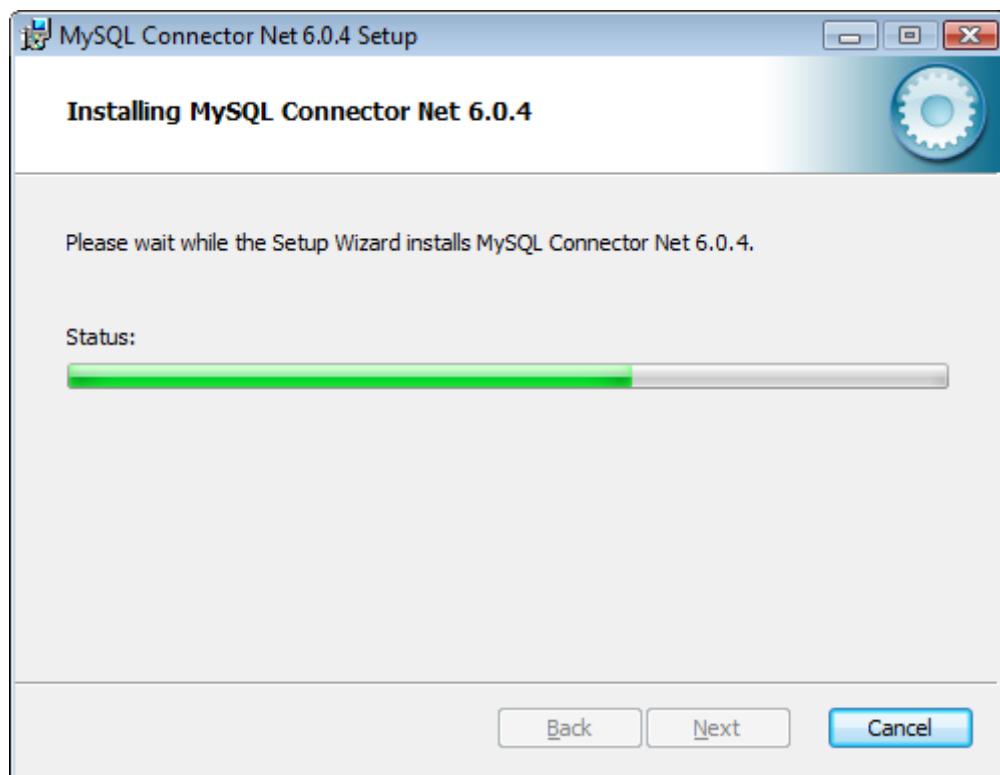
Первый шаг установки коннектора:



На втором шаге выберите нужные компоненты для установки. Если не уверены в выборе, то оставьте параметры по умолчанию.



На третьем шаге произойдет собственно установка коннектора.



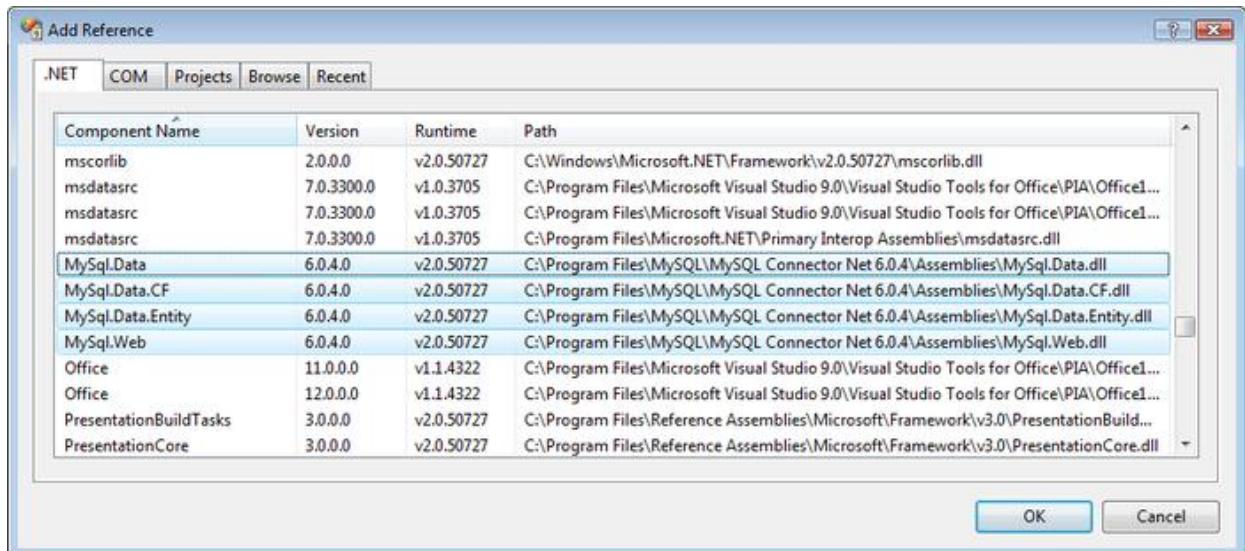
После завершения установки MySQL Connector/Net 6.0 автоматически добавит новые пространства имен **MySql.Data** в .NET.



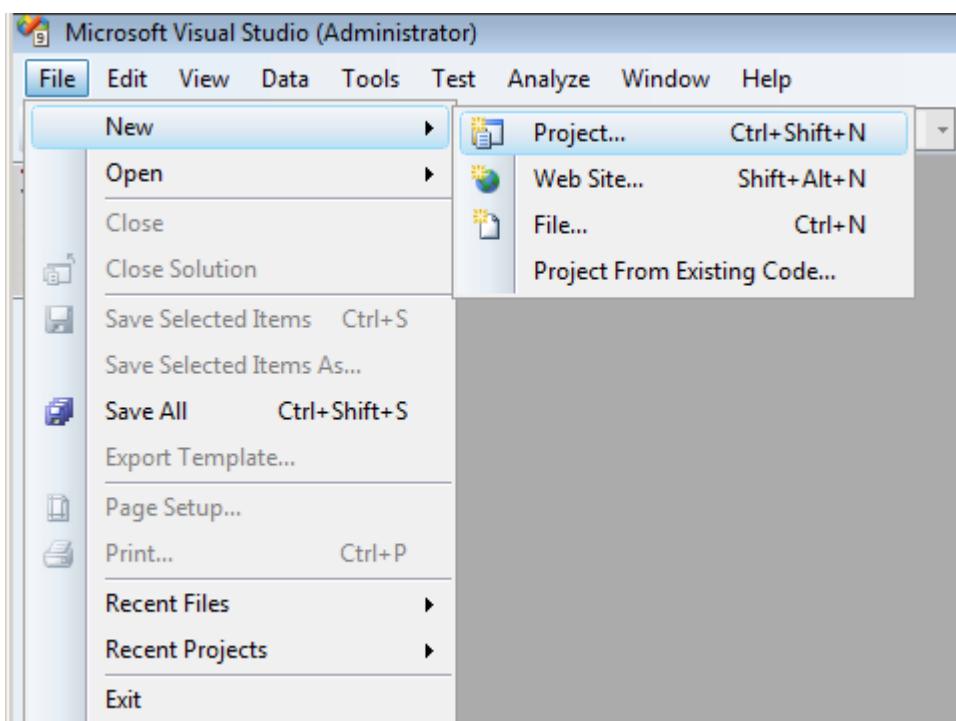
## Перенос с помощью Visual Studio

В качестве рабочего инструмента для миграции можно использовать *Visual Studio*, в частности, ее [бесплатную редакцию](#).

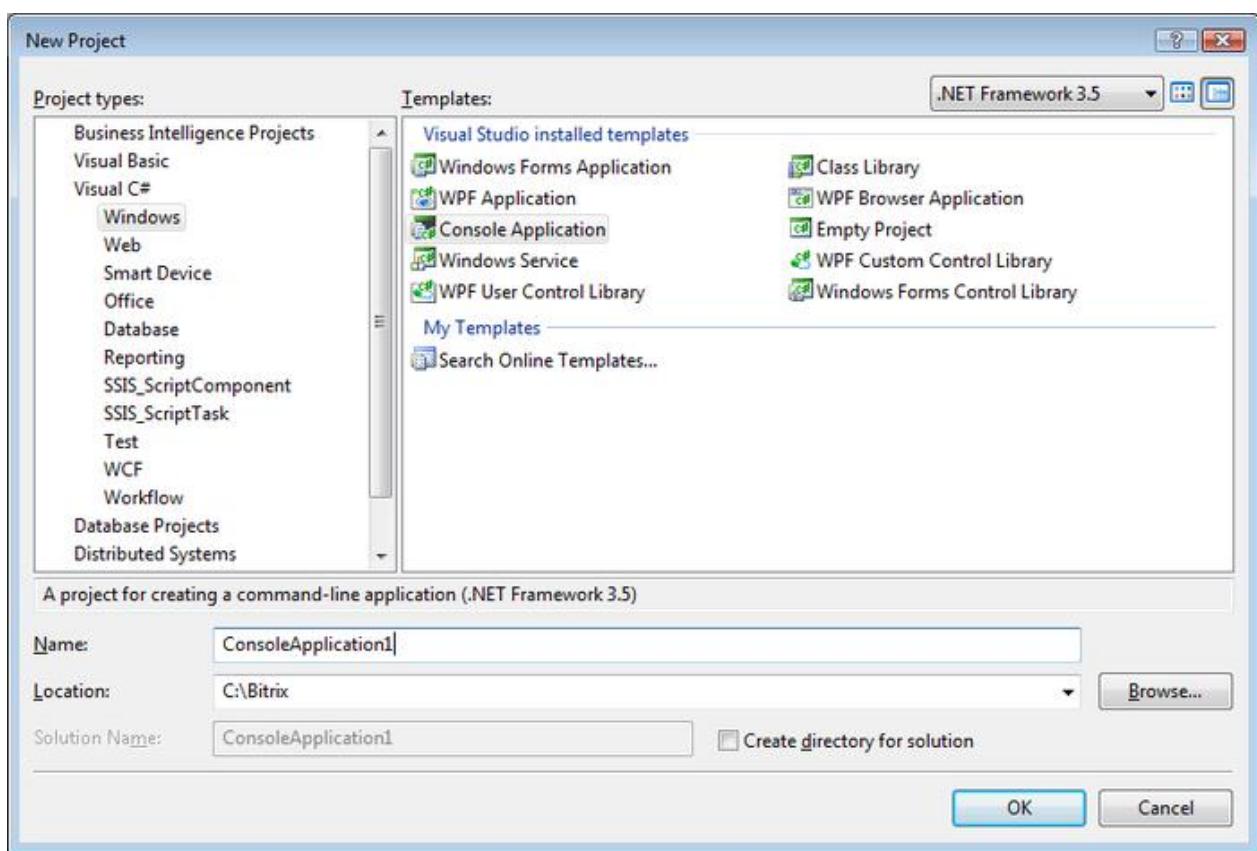
- Установите *Visual Studio* как это описано в документации Microsoft.
- Откройте *Visual Studio* и добавьте ссылку на добавленные пространства имен:



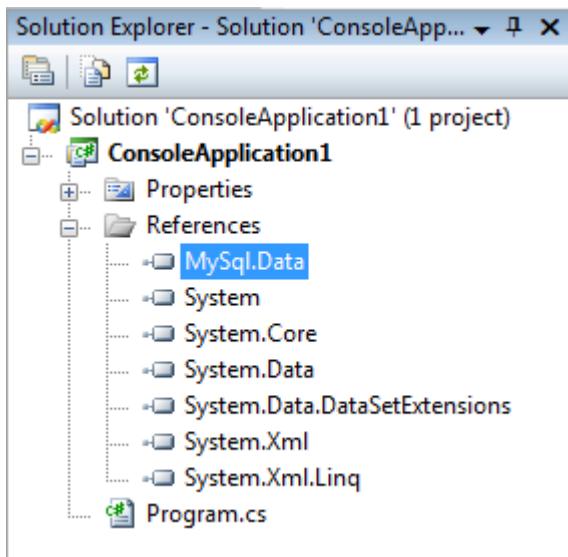
- Выберите в меню *File > New > Project*:



- Выберите в качестве шаблона проекта создание нового консольного приложения на C#:



- Добавьте к ссылкам (References) проекта пространство имен MySql.Data, которое добавляет MySql Connector/Net 6.0:



- Выполните следующий код:

```
using System;  
using System.Text;  
  
using MySql.Data.MySqlClient;
```

```
using System.Data;
using System.Diagnostics;
using System.Data.SqlClient;

class Program
{
    static MySqlConnection mySqlCnn;
    static SqlConnection sqlSrvCnn;

    static void Main(string[] args)
    {
        sqlSrvCnn = new SqlConnection(@"server=(local)\SQLExpress;database=bitrix;trusted_connection=true;MultipleActiveResultSets=true");
        sqlSrvCnn.Open();

        mySqlCnn = new MySqlConnection("server=127.0.0.1;port=31006;uid=root;pwd=;database=bsm_demo;Pooling=False");
        mySqlCnn.Open();

        DisEnableFKConstraints(true);
        DataTable tblList = GetSourceTablesFromMySQLDB();
        CleanDestTablesInSQLSrvDB(tblList);
        TransferData(tblList);
        DisEnableFKConstraints(false);

        mySqlCnn.Close();
        sqlSrvCnn.Close();
    }

    ///
    /// Копирует данные из таблицы в MySQL в одноименную таблицу в SQL Server
    /// Предполагается, что множества имен полей в таблицах совпадают. Порядок
    /// может отличаться.
    ///
    /// Имя таблицы
    static void CopyDataFromMySQLTblToCorrespondingSQLSrvTbl(string tblName)
    {
```

```
//Читаем по порядку поля в таблице-назначения
SqlCommand sqlSrvCmd = sqlSrvCnn.CreateCommand();
sqlSrvCmd.CommandText = "select name from sys.columns where object_id =
object_id(@tblName) order by column_id";
sqlSrvCmd.Parameters.AddWithValue("@tblName", tblName);
SqlDataReader sqlSrvDr = sqlSrvCmd.ExecuteReader(CommandBehavior.SingleResult);
//Составляем строку запроса для источника, перечисляя туда поля в том порядке,
как они следуют в назначении
StringBuilder mySqlCmdText = new StringBuilder("select ");
//Имя поля заключаем в аналог квадратных скобок - на случай, если оно будет
совпадать с одним из зарезервированных слов MySQL.
while (sqlSrvDr.Read()) mySqlCmdText.Append("`" + sqlSrvDr.GetSqlString(0).Value + `",");
sqlSrvDr.Close();
mySqlCmdText.Remove(mySqlCmdText.Length - 1, 1);
mySqlCmdText.Append(" from " + tblName);

MySqlCommand mySqlCmd = new MySqlCommand(mySqlCmdText.ToString(), mySqlCnn);
MySqlDataReader mySqlDr = mySqlCmd.ExecuteReader();

SqlBulkCopy bcp = new SqlBulkCopy(sqlSrvCnn, SqlBulkCopyOptions.KeepIdentity, null);
//KeepIdentity означает set identity_insert on/off
//Поскольку в mySqlDr поля идут в том же порядке, что и в назначении,
SqlBulkCopy.ColumnMappings не требуется.
bcp.DestinationTableName = tblName;
// Заправляем шланг ридера объекту SqlBulkCopy, чтобы он качал из него содержимое в
bcp.DestinationTableName
bcp.WriteToServer(mySqlDr);

mySqlDr.Close();
}

////
/// Получает список таблиц из MySQLной базы
///
/// Список таблиц
static DataTable GetSourceTablesFromMySQLDB()
{
    DataTable tbl = new DataTable();

```



```
tbl.Load(new MySqlCommand("show tables", mySqlCnn).ExecuteReader());  
return tbl;  
}  
  
///  
/// Удаляет в каждой таблице из списка все ее записи  
///  
/// Список таблиц  
static void CleanDestTablesInSQLSrvDB(DataTable tblList)  
{  
    Debug.WriteLine("Очистка таблиц назначения...");  
    foreach (DataRow r in tblList.Rows)  
    {  
        new SqlCommand("delete " + r[0].ToString(), sqlSrvCnn).ExecuteNonQuery();  
        Debug.WriteLine("Очищена таблица " + r[0].ToString());  
    }  
    Debug.WriteLine("Очистка закончена.");  
}  
  
static void TransferData(DataTable tblList)  
{  
    Debug.WriteLine("Загрузка данных...");  
    foreach (DataRow r in tblList.Rows)  
    {  
        CopyDataFromMySQLTblToCorrespondingSQLSrvTbl(r[0].ToString());  
        Debug.WriteLine("Перенесена таблица " + r[0].ToString());  
    }  
    Debug.WriteLine("Загрузка завершена.");  
}  
  
///  
/// Процедура отключает/включает все ограничения внешнего ключа над таблицами  
в БД SQL Server  
///  
/// Если да, то отключить, нет - включить  
static void DisEnableFKConstraints(bool switchOff)  
{
```



```
string prefix = switchOff ? "Om" : "B";
Debug.WriteLine(prefix + "ключение FK-ограничений...");
SqlDataReader sdr = new SqlCommand("select name, object_name(parent_object_id) from sys.foreign_keys", sqlSrvCnn).ExecuteReader();
while (sdr.Read())
{
    string fkName = sdr.GetString(0), tblName = sdr.GetString(1);
    new SqlCommand(String.Format("alter table {0} {1}check constraint {2}", tblName,
switchOff ? "no" : "", fkName), sqlSrvCnn).ExecuteNonQuery();
    Debug.WriteLine(String.Format("{0}ключено ограничение {1} в таблице {2}", prefix,
fkName, tblName));
}
sdr.Close();
Debug.WriteLine(prefix + "ключение FK-ограничений завершено.");
}
```

## Комментарии к коду

Необходимо сделать некоторые комментарии к коду.

Как показывает

```
select * from sys.objects where type = 'F'
```

(или sys.foreign\_keys/ sys.foreign\_key\_columns) в базе имеются ограничения внешнего ключа. Следовательно, первоначально следует вставлять данные в **referenced\_object (PK)**, а затем в **parent\_object (FK)**, чтобы избежать нарушений ограничений внешнего ключа.

Возможны ситуации, когда **referenced\_object** сам, в свою очередь, имеет **referenced\_object**. Следовательно, требуется упорядочить таблицы, выбрав сначала те **referenced\_objects**, которые не имеют FK-ограничений, вставить данные в них, затем в те таблицы, для которых они являются PK-таблицами и т.д.

Чтобы не усложнять скрипт миграции, было принято решение на время переноса данных отключить все FK-ограничения, вставить данные, а затем снова включить. Отключение FK-ограничений выполняется при помощи команды:

```
ALTER TABLE <имя FK таблицы> NOCHECK CONSTRAINT <имя ограничения>
```

А включение, соответственно, - CHECK.



Отключение/включение ограничений внешнего ключа делает процедура **DisEnableFKConstraints (bool switchOff)**. В том, что FK-ограничения отключены, можно убедиться по запросу

```
select * from sys.foreign_keys
```

в результатах которого колонка `is_disabled` стала 1 для всех записей.

Перед загрузкой данных содержимое таблиц SQL Express следует очистить. Несмотря на отключенные ограничения чистить таблицы при помощи команды `TRUNCATE TABLE` не получится. Приходится использовать команду `DELETE <имя таблицы>` для удаления из каждой таблицы всех ее записей.

По команде

```
select * from sys.columns where is_identity = 1
```

или

```
select * from sys.identity_columns
```

можно увидеть, что на некоторых таблицах имеются колонки с автоинкрементом. Однако специально отключать автоинкремент перед вставкой `SET IDENTITY_INSERT <имя таблицы> ON | OFF` не требуется, т.к. это "за сценой" делает объект **SqlBulkCopy** при помощи параметра **KeepIdentity**.

Общая последовательность действий по переносу данных выглядит следующим образом:

- Открываются соединения с БД MySQL и SQL Express.
- Включается **MARS** в SQL Server'ном соединении для выполнения процедуры **DisEnableFKConstraints**, где держится на соединении открытый DataReader со списком FK. По этому списку на каждой записи выполняется команда `ExecuteNonQuery()` на том же соединении.
- Отключаются все ограничения внешнего ключа в БД SQL Express, чтобы не заботиться о последовательности очистки и заливки.
- Получается список таблиц из БД MySQL. Он сохраняется в **DataTable tblList**.
- Очищаются таблицы из этого списка.
- Последовательно по этому списку переносятся данные из каждой таблицы MySQL в одноименную таблицу SQL Express.

Перед выполнением загрузки из MySQL рекомендуется выполнить резервное копирование базы данных на SQL Express несмотря на то, что она пуста, т.е. содержит только " заводские" установки и весь наработанный контент хранится в базе MySQL. Резервное копирование базы данных SQL Server можно выполнить при помощи команды:



```
backup database Bitrix to disk = 'c:\Bitrix\bitrix.bak' with noformat, init, name = N'Bitrix-Full Database Backup', skip, stats = 10
```

Восстановление (при необходимости) выполняется командой:

```
alter database Bitrix set single_user with rollback immediate
use master
restore database Bitrix from disk = 'c:\Bitrix\Bitrix.bak' with recovery, stats = 20
```

### Перенос данных с помощью PowerShell

Если у клиента нет *Visual Studio* или он по каким-либо причинам не может установить ее Express-редакцию, ниже приводится вариация скрипта, мигрирующего базу «1С-Битрикс: Управление сайтом» на языке сценариев PowerShell:

```
cls

function DisEnableFKConstraints([bool] $switchOff)
{
    [string] $prefix; if ($switchOff) { $prefix = "Om" } else { $prefix = "B" };
    Write-Host ($prefix + "ключение FK-ограничений...")

    [System.Data.SqlClient.SqlDataReader] $sdr = (New-Object
System.Data.SqlClient.SqlCommand("select name, object_name(parent_object_id) from sys.foreign_keys", $sqlSrvCnn)).ExecuteReader()
    while ($sdr.Read())
    {
        [string] $fkName = $sdr.GetString(0); [string] $tblName = $sdr.GetString(1)
        [string] $prefix1 = ""; if ($switchOff) {$prefix1 = "no"}
        [string] $cmdText = "alter table {0} {1}check constraint {2}" -f $tblName,
$prefix1, $fkName
        (New-Object
System.Data.SqlClient.SqlCommand($cmdText,
$sqlSrvCnn)).ExecuteNonQuery()

        Write-Host ("'{0}ключено ограничение {1} в таблице {2}" -f $prefix, $fkName,
$tblName)
    }
    $sdr.Close();
    Write-Host ($prefix + "ключение FK-ограничений завершено.")
}
```



```
function CleanDestTablesInSQLSrvDB([System.Data.DataTable] $tblList)
{
    Write-Host "Очистка таблиц назначения..."
    foreach ($r in $tblList.Rows)
    {
        [string] $cmdText = "delete " + $r[0]
        (New-Object System.Data.SqlClient.SqlCommand($cmdText,
$SqlSrvCnn)).ExecuteNonQuery()
        Write-Host ("Очищена таблица " + $r[0])
    }
    Write-Host "Очистка закончена."
}

function TransferData([System.Data.DataTable] $tblList)
{
    Write-Host "Загрузка данных..."
    foreach ($r in $tblList.Rows)
    {
        CopyDataFromMySQLTblToCorrespondingSQLSrvTbl($r[0])
        Write-Host ("Перенесена таблица " + $r[0])
    }
    Write-Host "Загрузка завершена."
}

function CopyDataFromMySQLTblToCorrespondingSQLSrvTbl([string] $tblName)
{
    [System.Data.SqlClient.SqlCommand] $sqlSrvCmd = $sqlSrvCnn.CreateCommand()
    $sqlSrvCmd.CommandText = "select name from sys.columns where object_id = object_id(@tblName) order by column_id"
    $sqlSrvCmd.Parameters.AddWithValue("@tblName", $tblName)
    [System.Data.SqlClient.SqlDataReader] $sqlSrvRdr = $sqlSrvCmd.ExecuteReader()
    [System.Text.StringBuilder] $mySqlCmdText = New-Object System.Text.StringBuilder("select ")
    while ($sqlSrvRdr.Read()) { $mySqlCmdText.Append(`"` + $sqlSrvRdr.GetSqlString(0) + `","`)}
    $sqlSrvRdr.Close()
    $mySqlCmdText.Remove($mySqlCmdText.Length - 1, 1)
    $mySqlCmdText.Append(" from " + $tblName)
```



```
[ MySql.Data.MySqlClient.MySqlCommand ]      $mySqlCmd      =      New-Object
MySql.Data.MySqlClient.MySqlCommand($mySqlCmdText.ToString(), $mySqlCnn);

[ MySql.Data.MySqlClient.MySqlDataReader ] $mySqlRdr = $mySqlCmd.ExecuteReader()

[System.Data.SqlClient.SqlBulkCopy]           $bcp          =      New-Object
System.Data.SqlClient.SqlBulkCopy($sqlSrvCnn,
[ System.Data.SqlClient.SqlBulkCopyOptions ]::KeepIdentity, $null)

$bcp.DestinationTableName = $tblName
$bcp.WriteToServer($mySqlRdr)
$bcp.Close()

$mySqlRdr.Close()
}

#####
##### MAIN #####
#####

[System.Data.SqlClient.SqlConnection]      $sqlSrvCnn      =      New-Object
System.Data.SqlClient.SqlConnection("server=(local)\SQLEXPRESS;database=bitrix;trusted_connection=true;MultipleActiveResultSets=true")
$sqlSrvCnn.Open()

[void][system.reflection.Assembly]::LoadWithPartialName(" MySql.Data ")

[ MySql.Data.MySqlClient.MySqlConnection ]      $mySqlCnn      =      New-Object
MySql.Data.MySqlClient.MySqlConnection("server=127.0.0.1;port=31006;uid=root;pwd=;database=bsm_demo;Pooling=False")
$mySqlCnn.Open()

$mySqlRdr = (New-Object MySql.Data.MySqlClient.MySqlCommand("show tables;", $mySqlCnn)).ExecuteReader()

[System.Data.DataTable] $tblList = New-Object System.Data.DataTable
$tblList.Load($mySqlRdr)
$mySqlRdr.Close()

$DisEnableFKConstraints $true | Out-Null
$CleanDestTablesInSQLSrvDB $tblList
$TransferData $tblList | Out-Null
$DisEnableFKConstraints $false | Out-Null
```

```
$sqlSrvCnn.Close()  
$mySqlCnn.Close()
```

## Общая последовательность действий

Общая последовательность действий по переносу данных на локальной установке:

- Установить *SQL Express*.
  - Переименовать каталог `/www` в папке `/Bitrix/Environment` и установить «*1C-Битрикс: Управление сайтом*» на *SQL Express*.
  - Установить *.NET Connector* к *MySQL*.
  - Закрыть все вкладки в браузере с локальной версией сайта, остановить процесс **Bitrix Environment**.
  - Проверить, что *MySQL* по-прежнему запущен (`mysqld-opt.exe` значится в числе работающих процессов в **Диспетчере задач**). Если нет, то запустить процесс командой:

"C:\Program Files\Bitrix Environment\mysql\bin\mysqld-opt.exe" --port=31006

- Провести перенос данных с помощью *Visual Studio* или *PowerShell*:

The screenshot shows the PowerGUI Script Editor interface. The main window displays a PowerShell script named 'Migration.ps1'. The script uses .NET's System.Data.SqlClient and MySQL.Data.MySqlClient namespaces to copy data from a MySQL database to a SQL Server database. It includes functions for selecting columns from MySQL tables and performing a bulk insert into a SQL Server table. The script also handles connection strings for both databases.

```
function CopyDataFromMySQLToCorrespondingSQLServerTbl([string] $tblName)
{
    [System.Data.SqlClient.SqlCommand] $sqlSrvCmd = $sqlSrvCnn.CreateCommand()
    $sqlSrvCmd.CommandText = "select name from sys.columns where object_id = object_id($tblName) order by column_id"
    $sqlSrvCmd.Parameters.AddWithValue("tblName", $tblName)
    [System.Data.SqlClient.SqlDataReader] $sqlSrvRdr = $sqlSrvCmd.ExecuteReader()
    [System.Text.StringBuilder] $mySqlCmdText = New-Object System.Text.StringBuilder("select ")
    while ($sqlSrvRdr.Read()) { $mySqlCmdText.Append("`" + $sqlSrvRdr.GetSqlString(0) + "`", ",") }
    $sqlSrvRdr.Close()
    $mySqlCmdText.Remove($mySqlCmdText.Length - 1, 1)
    $mySqlCmdText.Append(" from ` - $tblName")
    [ MySql.Data.MySqlClient.MySqlCommand ] $mySqlCmd = New-Object MySql.Data.MySqlClient.MySqlCommand($mySqlCmdText.ToString(), $mySqlCnn);
    [ MySql.Data.MySqlClient.MySqlDataReader ] $mySqlRdr = $mySqlCmd.ExecuteReader()

    [System.Data.SqlClient.SqlBulkCopy] $bcp = New-Object System.Data.SqlClient.SqlBulkCopy($sqlSrvCnn, [System.Data.SqlClient.SqlBulkCopyOptions]::KeepIdentity)
    $bcp.DestinationTableName = $tblName
    $bcp.WriteToServer($mySqlRdr)
    $bcp.Close()

    $mySqlRdr.Close()
}

#####
MAIN #####
####

[System.Data.SqlClient.SqlConnection] $sqlSrvCnn = New-Object System.Data.SqlClient.SqlConnection("server=(local)\SQLExpress;database=bitrix;trusted_connection=True")
$sqlSrvCnn.Open()
[void][System.reflection.Assembly]::LoadWithPartialName("MySql.Data")
[ MySql.Data.MySqlClient.MySqlConnection ] $mySqlCnn = New-Object MySql.Data.MySqlClient.MySqlConnection("server=127.0.0.1:port=31006;uid=root;pwd=bash;database=bitrix")

$Variables
```

- Вновь запустить **Bitrix Environment** и открыть сайт.



1С·БИТРИКС

Компания «1С-Битрикс» Системы управления веб-проектами

Тел.: (495) 363-37-53; (4012) 51-05-64; e-mail: info@1c-bitrix.ru, http://www.1c-bitrix.ru

**⚠ Примечание:** При возникновении непредвиденной ситуации вернуться на исходную позицию можно, вернув переименованной папке с установкой «1С-Битрикс: Управление сайтом» на MySQL имя /www.