



# 1С-Битрикс: Управление сайтом

Быстрый старт разработчика

Компоненты





## Содержание

<b>Введение .....</b>	<b>3</b>
<b>Глава 1. Компоненты.....</b>	<b>4</b>
ПРОСТЫЕ И КОМПЛЕКСНЫЕ.....	5
Простые компоненты .....	5
Комплексные компоненты .....	5
РАЗМЕЩЕНИЕ КОМПОНЕНТОВ .....	6
Именованние .....	7
СТРУКТУРА КОМПОНЕНТА .....	8
ОПИСАНИЕ И ПАРАМЕТРЫ.....	9
ШАБЛОНЫ КОМПОНЕНТОВ .....	10
Шаблон простого компонента .....	11
Шаблон комплексного компонента .....	11
Редактирование шаблона .....	12
Пример шаблона .....	14
ФАЙЛЫ RESULT_MODIFIER И COMPONENT_EPILOG .....	15
КАСТОМИЗАЦИЯ КОМПОНЕНТОВ .....	17
СОЗДАНИЕ КОМПОНЕНТОВ.....	18
Порядок создания компонента.....	18
КЕШИРОВАНИЕ КОМПОНЕНТОВ.....	19



## Введение

---

Учебный курс для начинающего разработчика. Вы первый раз обращаетесь к популярной платформе Bitrix Framework и не знаете откуда начать? Начните с этого курса.

**⚠ Внимание!** Курс *Быстрый старт разработчика* - "выжимка" из огромного объема справочной информации по *Bitrix Framework*. Он создан с целью облегчить начинающему разработчику изучение системы и не является ни в коей мере заменой документации и базовых учебных курсов. В нем даются только основы с указанием более детальных источников информации.

При условии качественного изучения материалов курса, по его окончании специалист получит базовые понятия о платформе и ссылки на все, что может вам понадобиться при углубленном изучении *Bitrix Framework*.

Минимальный уровень знаний, необходимый при изучении:

- PHP
- HTML, CSS

**⚠ Примечание.** Для успешной разработки структуры информационных блоков сложных по структуре проектов, полезно будет получить хотя бы базовые знания по [реляционным базам данных](#).

**Список ссылок по теме:**

- [PHP](#)
- [HTML, CSS](#)



## Глава 1. Компоненты

---

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.

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.

Эволюция развития **Bitrix Framework** через компоненты обычного стандарта привела к действующим сейчас компонентам в стандарте 2.0. Обычный стандарт в данное время не поддерживается.

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.

#### **Список ссылок по теме:**

- Раздел [Компоненты 2.0](#) в **Документации для разработчиков**
- Учебный курс [Компоненты 2.0](#)
- Работа с компонентами на уровне [Контент-менеджера](#)

## **Простые и комплексные**

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, одностраничные) 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 одностраничных 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, посылает новый запрос комплексному компоненту.

**Список ссылок по теме:**

- [Комплексные компоненты](#) в **Документации для разработчика**

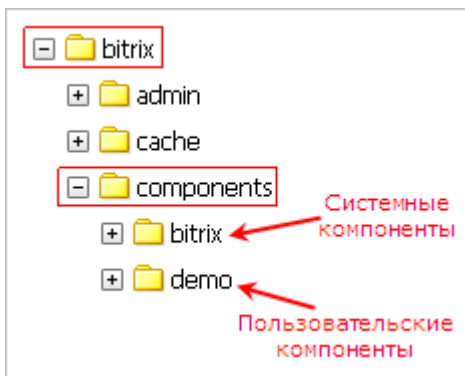
## **Размещение компонентов**

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/**.



## Именовании

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



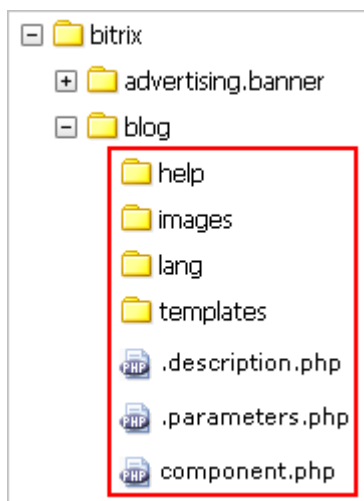
пространства имен. Полное имя имеет вид **пространство\_имен:имя\_компонента**. Например, **bitrix:catalog.list**. Если компонент лежит вне пространства имен, то пространство имен не указывается. Например, **catalog.section**.

## 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. В этой папке находятся подпапки с именами, равными кодам языков, где находятся файлы языковых сообщений (переводов) компонента. Рекомендуется называть файл языковых сообщений для какого-либо файла компонента так же, как называется этот файл. И располагать 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.



- 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. 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.
- 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**.

## Описание и параметры

### Описание

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).

### Параметры

В файле **.parameters.php** содержится описание входных параметров 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 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 **\$arParamsComponentParameters**, 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.

### Список ссылок по теме:

- [Описание компонента](#) в Документации для разработчиков



- [Параметры компонента](#) в Документации для разработчиков

## Шаблоны компонентов

**!** *Шаблон компонента - программный код, преобразующий данные, подготовленные компонентом, непосредственно в HTML-код.*

Шаблоны компонента делятся на **системные** и **пользовательские**:

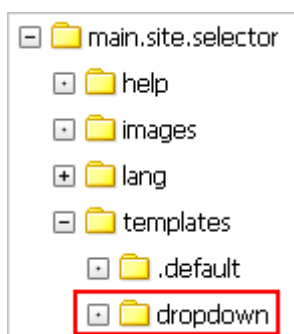
- Системные шаблоны поставляются вместе с компонентом и лежат в подпапке **templates** папки компонента.
- Пользовательские шаблоны компонента - шаблоны, которые изменены под нужды конкретного сайта. Они должны лежать в папках шаблонов сайтов (т.е. в **/bitrix/templates/шаблон\_сайта/**). При копировании шаблона компонента средствами системы, они будут расположены по следующему пути: **/bitrix/templates/шаблон\_сайта/components/namespace/название\_компонента/название\_шаблона**.

Шаблоны компонента определяются по именам. Шаблон по умолчанию имеет имя **.default**. Если в настройках параметра компонента не указывается имя шаблона, вызывается шаблон по умолчанию. Остальные шаблоны могут называться произвольно.

Шаблоны компонента могут быть папками или файлами. Если шаблону не требуется перевод на другие языки, собственные стили и прочие ресурсы, такой шаблон можно расположить в файле. В противном случае шаблон следует располагать в папке.

Каждый шаблон компонента является неделимым целым. Если требуется изменить системный шаблон компонента под конкретный сайт, то этот шаблон компонента нужно целиком скопировать в папку шаблона сайта.

Например, у компонента **bitrix:catalog.site.selector** есть системный шаблон **dropdown**, который лежит в подпапке **templates** папки компонента:



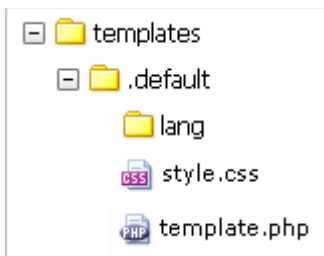
Если этот шаблон компонента требуется изменить под конкретный сайт, то папку шаблона **dropdown** следует скопировать в папку **/bitrix/templates/шаблон\_сайта/components/bitrix/catalog.site.selector/** и изменить по своему усмотрению.

При включении компонента на страницу сайта администратор настраивает, какой именно шаблон вывода будет использоваться в данном конкретном случае.



## Шаблон простого компонента

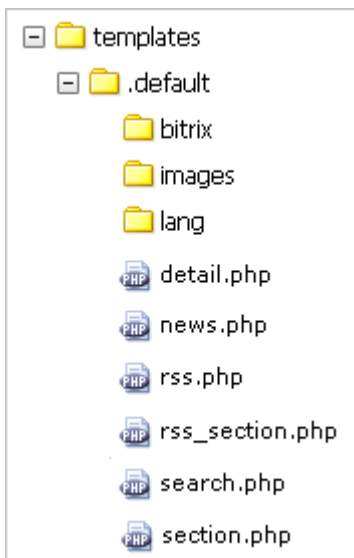
Папка шаблона простого компонента может содержать следующие подпапки и файлы:



- подпапку **/lang**, в которой расположены файлы языковых сообщений (переводов) шаблона компонента;
- файл **result\_modifier.php**, который подключается непосредственно перед подключением шаблона компонента. Этот файл получает на вход массив результатов работы компонента `$arResult` и массив параметров вызова компонента `$arParams`. Таким образом, можно, например, изменить массив результатов работы компонента под конкретный шаблон.
- файл **style.css**, который определяет стили, необходимые данному шаблону.
- файл **script.js**, который определяет и подключает ява-скрипты, необходимые данному шаблону. Этот файл может отсутствовать.
- файл **.description.php**, который содержит название и описание шаблона для визуального редактора.
- файл **.parameters.php**, который содержит описание дополнительных входных параметров шаблона для визуального редактора.
- файл **template.ext**, который и является собственно шаблоном. Расширение **ext** зависит от того, какой движок шаблонизации нужно подключать. По умолчанию расширение равно **php**. Этот файл должен обязательно присутствовать.
- любые другие папки и файлы с ресурсами, необходимыми шаблону компонента. Например, папка **image**, содержащая изображения, необходимые шаблону.

## Шаблон комплексного компонента

Шаблон комплексного компонента содержит все те же папки, что и шаблон простого компонента, и дополнительно:



- шаблоны простых компонентов, которые входят в состав комплексного. Эти шаблоны располагаются в папках вида **/пространство\_имен/название\_простого\_компонента/** относительно папки шаблона комплексного компонента.
- простые компоненты, входящие в состав комплексного, подключаются на шаблонах страниц комплексного компонента.

### Подключение шаблона

В коде подключения указывается только `<namespace>`, имя компонента, имя шаблона (и параметры самого компонента). При обработке кода ядро сначала проверяет наличие шаблона компонента в шаблоне текущего сайта: `/bitrix/templates/<имя шаблона сайта>/components//<имя компонента>/<имя шаблона>/template.php`.

Если - **bitrix**, то это папка для шаблонов стандартных компонентов. Если - выбранное вами `<имя>` для своих, лежащих в `/bitrix/components/<имя>`, то это папка для шаблонов ваших компонентов.

Если файла шаблона нет, проверяется шаблон сайта по умолчанию: `/bitrix/templates/.default/components//<имя компонента>/<имя шаблона>/template.php`.

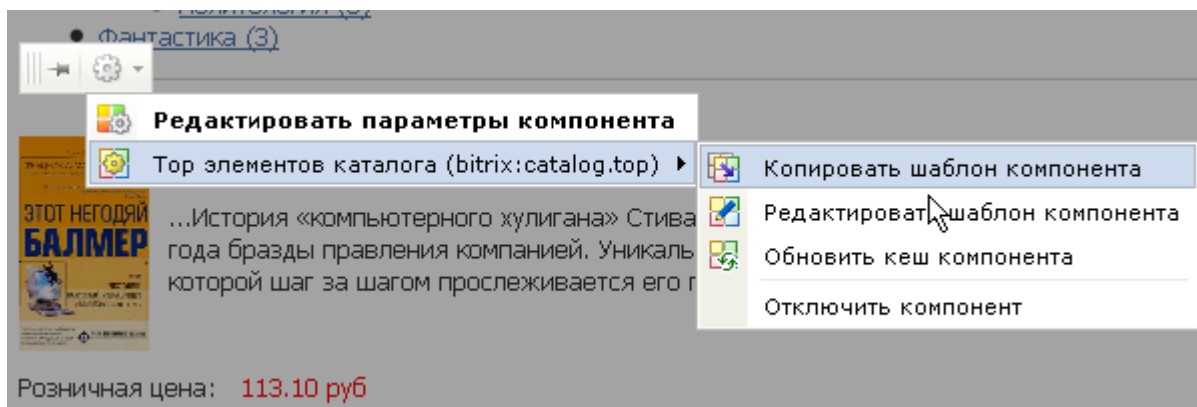
И только после этого происходит подключение шаблона компонента из папки компонента.

### Редактирование шаблона

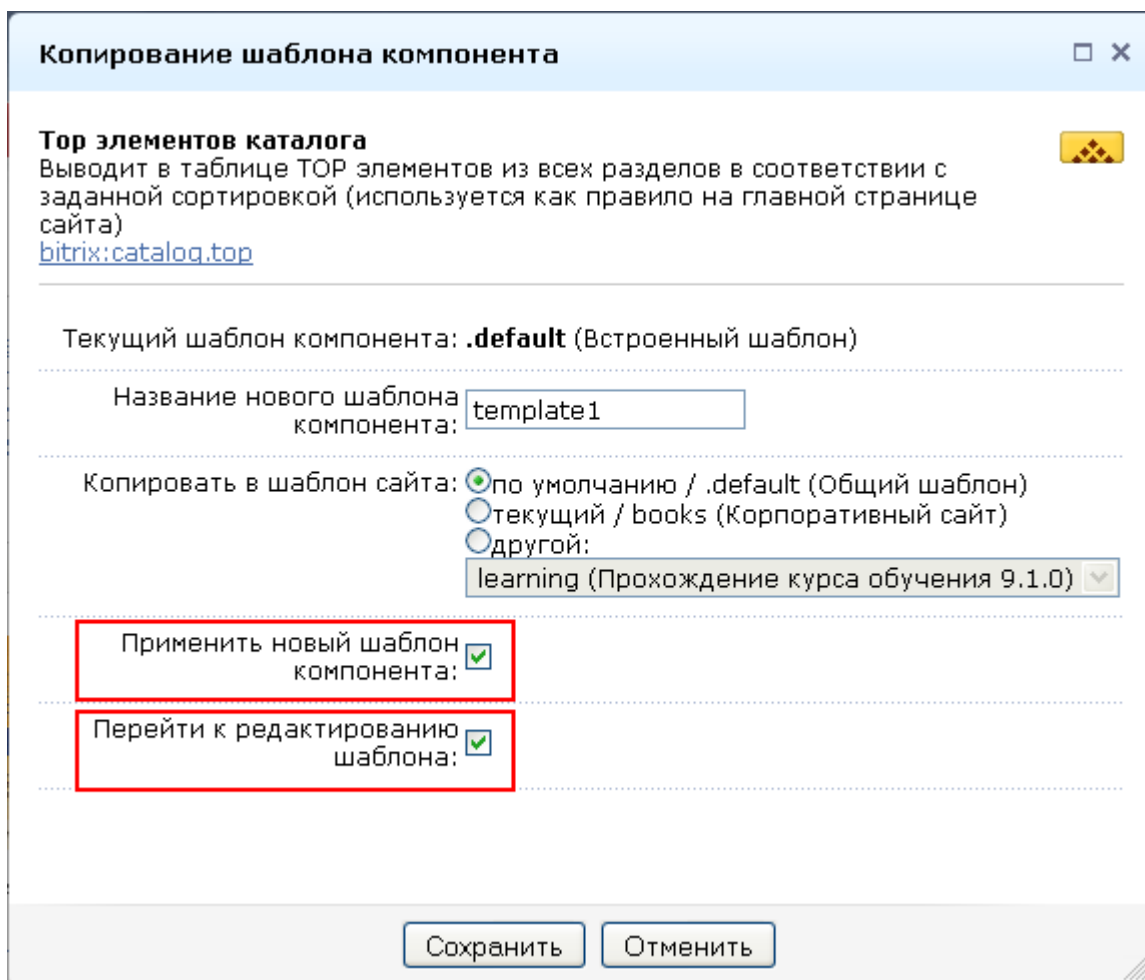
Редактирование шаблона - один из способов получения нужного результата работы компонента под требования сайта. Редактировать системный шаблон не рекомендуется в силу того, что при первом же обновлении системный шаблон восстановит свое прежнее состояние. Редактировать и применять можно только пользовательский шаблон.



Копирование шаблона выполняется по команде **Копировать шаблон компонента** при включенном режиме **Правка**:



При копировании шаблона можно сразу задать его применение к компоненту и открыть форму для редактирования шаблона:



При отмеченной опции **Перейти к редактированию шаблона** сразу будет осуществлен переход на страницу редактирования шаблона компонента. В противном случае переход к редактированию осуществляется с помощью менеджера файлов либо при помощи команды **Редактировать шаблон компонента**.



Редактирование шаблона допускает добавление в него логики действий, но такую модификацию лучше выносить в файлы **result\_modifier.php** и **component\_epilog.php** для более сложного изменения результата работы.

## Пример шаблона

Шаблон компонента Меню	
Проверка включения	<code>&lt;?if (!defined("B_PROLOG_INCLUDED")    B_PROLOG_INCLUDED!==true)die();?&gt;</code>
Старт скрипта	<code>&lt;?if (!empty(\$arResult)):?&gt;</code>
Открытие тега <code>&lt;ul&gt;</code> — нумерованный список	<code>&lt;ul class="..."&gt;</code>
Старт цикла поиска	<code>&lt;?foreach (\$arResult as \$arResultItem):?&gt;</code>
Вывод ссылки	<code>&lt;?if(\$arResultItem["SELECTED"]):?&gt;</code>
Ссылка активная	<code>&lt;li&gt;&lt;a href="&lt;?=\$arResultItem["LINK"]?&gt;" class="selected"&gt;&lt;?=\$arResultItem["TEXT"]?&gt;&lt;/a&gt;&lt;/li&gt;</code>
Проверка на продолжение цикла	<code>&lt;?else:?&gt;</code>
Ссылка неактивная	<code>&lt;li&gt;&lt;a href="&lt;?=\$arResultItem["LINK"]?&gt;"&gt;&lt;?=\$arResultItem["TEXT"]?&gt;&lt;/a&gt;&lt;/li&gt;</code>
Завершение вывода ссылки	<code>&lt;?endif?&gt;</code>
Завершение цикла поиска	<code>&lt;?endforeach?&gt;</code>
Закрытие тега <code>&lt;ul&gt;</code> — нумерованный список	<code>&lt;/ul&gt;</code>
Завершение скрипта	<code>&lt;?endif?&gt;</code>

### Список ссылок по теме:

- [Шаблоны компонента](#) в курсе **Компоненты 2.0**
- [Шаблоны вывода](#) в **Документации для разработчиков**
- [Структура шаблона простого компонента](#) в **Документации для разработчиков**



- [Структура шаблона комплексного компонента](#) в **Документации для разработчиков**
- [Поиск шаблона компонента](#) в **Документации для разработчиков**

## Файлы `result_modifier` и `component_epilog`

**!** Файл `result_modifier.php` - инструмент для модификации данных работы компонента произвольным образом. Создается разработчиком самостоятельно.

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.

Если в папке шаблона есть файл `result_modifier.php`, то он вызывается перед подключением шаблона.

**!** Файл `component_epilog.php` - инструмент для модификации данных работы компонента с включенным кэшированием. Создается разработчиком самостоятельно.

Это файл подключается после исполнения шаблона. В нем доступны те же переменные, что и в шаблоне: `$arParams`, `$arResult`. 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 кэширования. Данные `$arParams`, `$arResult` при этом берутся из кеша. Точно так же при вызове дочерних компонентов в шаблоне нужно передавать значение `$component`.

**!** **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 закэшированных 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.

}

?>
```

### 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"], $arResultOptions);  
  
}  
  
?>
```

В этом случае вы не получите желаемого результата, выведутся данные компонента, а не `component_epilog.php`.

#### Список ссылок по теме:

- [Пример использования файла `result\_modifier.php`](#)

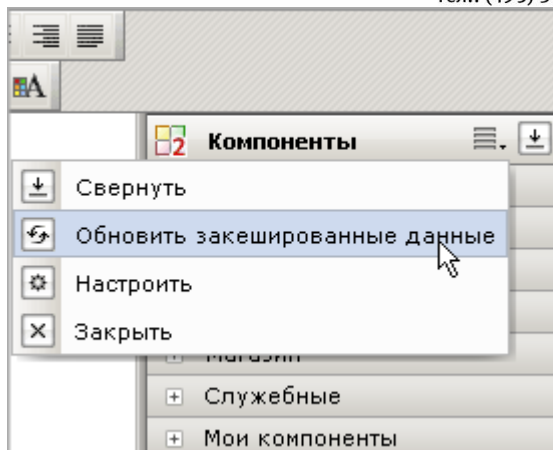
## Кастомизация компонентов

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**:



## Создание компонентов

Стандартный набор компонентов *Bitrix Framework* не всегда является достаточным. И приходит время, когда разработчик должен научиться создавать свои компоненты.

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.

**⚠ Внимание!** Названия создаваемых компонентов не должны пересекаться со стандартными.

## Порядок создания компонента

- 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](#)).

**Список ссылок по теме:**

- [Программирование компонентов](#) в курсе **Компоненты 2.0**

**Кеширование компонентов**

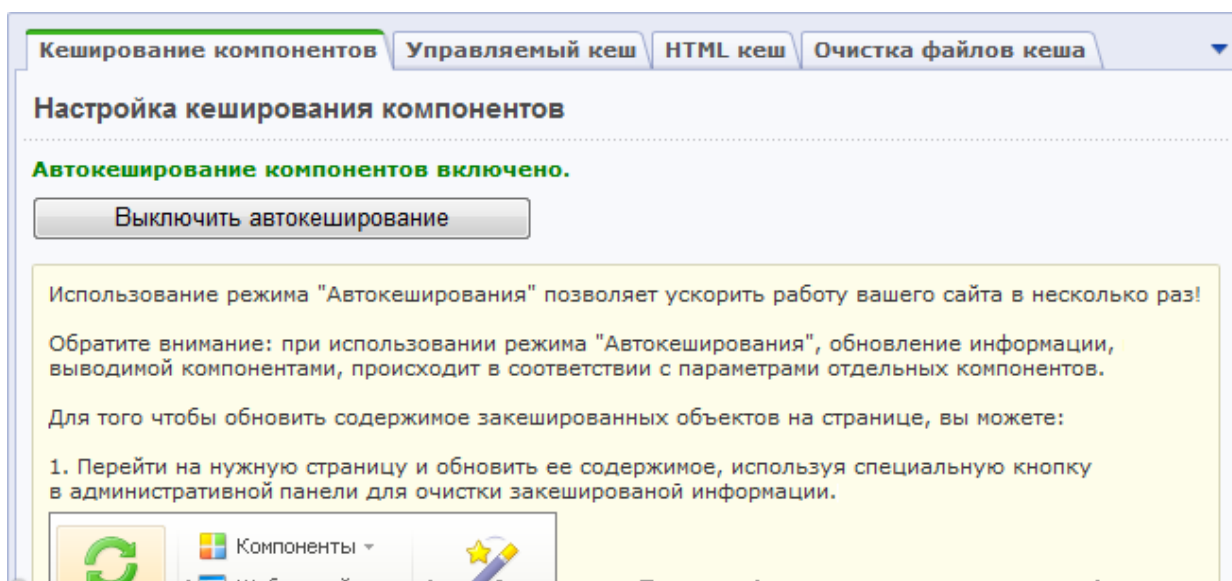
Одним из видов кеширования в *Bitrix Framework* является кеширование компонентов.

Для ускорения обработки запроса клиента и уменьшения нагрузки на сервер компоненты должны использовать кеширование. Кешировать, 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 кеш 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**:



**⚠ 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:

1. Перейти на нужную страницу и обновить ее содержимое, используя кнопку **Сбросить кеш** на панели инструментов.
2. В режиме **Правки сайта** использовать кнопки для очистки кеша в панели отдельных компонентов.
3. Использовать автоматический сброс кеша по истечении времени кеширования, для чего в настройках компонента выбрать режим кеширования **Кешировать** или **Авто + Управляемое**.
4. Использовать автоматический сброс кеша при изменении данных, для чего в настройках компонента выбрать режим кеширования **Авто + Управляемое**.
5. Перейти к настройкам выбранных компонентов и перевести их в режим работы без кеширования.

**Список ссылок по теме:**

- [Настройки кеширования](#) в курсе **Администратор. Базовый**.
- [Кеширование компонентов](#) в **Документации для разработчиков**
- [Тегированный кеш](#)