# MajorDoMo

## Style Guide

Документ описывает принципиальные требования по разработке модели базы данных и оформлению кода. Следование требованиям обязательно для всех участников проекта.

# MajorDoMo

Style Guide

## Соглашения по именованию объектов базы данных

### Общие требования

Имя любого объекта базы данных должно соответствовать её назначению и состоять не более чем из 30 символов. Некоторые имена объектов участвуют в составлении других имён объектов, поэтому для них существуют дополнительные ограничения. Длина имён таблиц должна быть не более 20 символов. Рекомендуемая длина индексных полей и полей первичного ключа – не более 10 символов.

Допускается использование только латинских букв. Запрещается использование символов @, #, $, %, ^, &, (), *,+,-.

При задании имён всех объектов базы данных должны использоваться символы в верхнем регистре.

### Разделение слов в именах

Слова, составляющие имя объекта, разделяются символом «_» (нижнее подчёркивание). В случае составного имени, из нескольких имён объектов, в именах которых присутствует символ разделения, имена объектов разделяются символами «__»(двойное подчёркивание). Использование других символов запрещается.

Пример: FK_TRACK_INFO__TRACK_ID

### Наименование таблиц

Имя таблицы должно быть задано на английском языке в единственном числе. Длина имён таблиц должна быть не более 20 символов.

Рекомендуется использовать названия, состоящие из одного слова минимальной длины. Имя таблицы должно в достаточной мере описывать сущность или часть сущности хранимой в таблице. Использование наименований с разделителем допускается только при отсутствии простых аналогов.

В системе допускается создавать логические группы таблиц на основе наименований. Для таблиц группы используется префикс, состоящий из уникального кода (не более 5 символов), за которым следует разделитель слов и имя таблицы.

Пример: APP_PRODUCT, APP_QUOTE и т.д.

| Группа таблиц | Префикс |
|---|---|
| Приложения | APP |
| Пользователи | USR |
| | |

## Наименование представлений данных

В названии представления данных (View) должен присутствовать префикс «V», за которым может следовать префикс, определяющий группу представления данных. Длина имен представлений должна быть не более 20 символов.

| Группа представлений данных | Префикс |
|---|---|
| Общесистемные | V_ |
| Информационные таблицы | V_XXXXX_ |

## Наименование полей таблиц и представлений данных

Длина имени поля должна быть не более 30 символов. Длина имени поля первичного или внешнего ключа должна быть не более 10 символов.

Наименование поля рекомендуется формировать из двух частей разделяемых символом «_» (нижнее подчёркивание):

1. Смысловое описание
2. Суффикс для обозначения наиболее часто встречающихся видов полей.

В качестве смыслового описания поля рекомендуется использовать единственное число имени атрибута, соответствующего этому полю или его аббревиатуру.

Полям, являющимися внешними ключами, рекомендуется давать имена, совпадающие с именами полей в главной таблице. Если в таблице имеется несколько ссылок на главную таблицу, то рекомендуется к имени поля в главной таблице добавить дополнительное содержательное слово.

Список суффиксов наиболее часто встречающихся видов атрибутов приведён в таблице:

| Тип атрибута | Суффикс |
|---|---|
| Автогенерируемый первичный ключ | ID |
| Код элемента справочника | CODE |
| Наименование элемента справочника | NAME |
| Краткое наименование элемента справочника | SNAME |
| Примечание | REM |
| Номер чего-либо | NUM |
| Количество чего-либо | VOL |

| | |
|---|---|
| Сумма чего-либо | SUM |
| Дата | DATE |

Кроме приведённых выше требований, в системе используются стандартные наименования полей, список которых приведён в следующей таблице.

| **Имя поля** | **Комментарий** |
|---|---|
| LM_DATE | Дата последней модификации |
| LM_USER | ID пользователя последней модификации |

## Наименования ограничений

### *Имена первичных ключей (Primary Key Constraint)*

Имя первичного ключа(PK) должно состоять из имени таблицы и префикса PK_.

### *Имена внешних ключей (Foreign Key Constraint)*

Имя вторичного ключа(FK) должно начинаться с префикса FK_ и имени таблицы, в которой создаётся внешний ключ. Затем, рекомендуется, через двойной разделитель задавать имя поля ключа. Если в результате общая длина имени более 30 символов, или ключ является составным, необходимо добавлять более короткое имя с соответствующей смысловой нагрузкой.

### *Имена ограничений уникальности (Alternative Key)*

Имя вторичного ключа(Alternative Key) должно начинаться с префикса AK_ и имени таблицы, в которой создаётся внешний ключ и через двойной разделитель указывать имя поля ключа. Если в результате общая длина имени более 30 символов, или ключ является составным, необходимо добавлять более короткое имя с соответствующей смысловой нагрузкой.

### *Имена ограничений корректности значений (Check Constraint)*

Имя ограничения должно иметь префикс CC_ и нести смысловую нагрузку.

Имя ограничения корректности(CC) должно начитаться с префикса CC_ и имени таблицы, в которой создаётся внешний ключ, далее через двойной разделитель задавать имя поля. Если в результате общая длина имени более 30 символов, или ключ является составным, необходимо добавлять более короткое имя с соответствующей смысловой нагрузкой.

## Имена индексов

Имя индекса должно начинаться с префикса IX_ и имени таблицы, в которой он создаётся, далее через двойной разделитель указывается имя поля индекса. Если в результате общая длина имени более 30 символов, или индекс является составным, необходимо добавлять более короткое имя с соответствующей смысловой нагрузкой.

# Соглашения по оформлению кода

## Общие правила

1. Не экономьте на понятности и частоте кода ради скорости его набора.
2. Не используйте подчёркивание для отделения слов внутри идентификаторов. Это удлиняет идентификаторы и усложняет чтение. Вместо этого используйте верблюжий стиль(Camel).
3. Старайтесь не использовать сокращения лишний раз, помните о тех, кто будет читать код.
4. При создании нового общедоступного(public) класса, пространства имён(namespace) или интерфейса(interface), не используйте имена, потенциально или явно конфликтующие со стандартными идентификаторами.
5. Используйте имена, которые ясно и чётко описывают предназначение или смысл сущности.
6. Не используйте для разных сущностей имена, отличающиеся регистром или на один-два символа.

## Регистр букв

Паскаль - все первые буквы слов заглавные. Например: WindSpeed, AccessTime

Кэмел(Camel Case) – первая буква строчная, остальные первые буквы слов заглавные. Например: windSpeed, accessTime.

## Сокращения

1. Не используйте аббревиатуры, акронимы или неполные слова в идентификаторах, если они не являются общепринятыми. Например, пишите GetWind, а не GetWin.
2. Используйте широко распространённые акронимы для сокращения длинных фраз. Например UI вместо User Interface.
3. Если имеется идентификатор длиной менее трёх букв, являющийся сокращением, то его записывают заглавными буквами. Например: userID.

## Выбор слов

Не используйте имена , совпадающие с зарезервированными словами или совпадающие с названием других сущностей Например: Catch, Const, Date, volatile и т.д.

## Методы

1. Используйте глаголы или комбинацию глагола и существительных и прилагательных для имён методов.
2. Используйте стиль Паскаль для регистра букв.
3. Область видимости необходимо указывать для всех методов
4. Описание методов в стиле PhpDoc

Пример:

```
/**
 * Email validation
 * @param mixed $email Email address to check
* @return bool
```

```
*/
public function IsValidEmail($email)
{
  $emailPattern = "/^([_a-z0-9-]+)(\.[_a-z0-9-]+)*@([a-z0-9-]+)(\.[a-z0-9-]+)*(\.[a-z]{2,4})$/";
  $emailToCheck = strtolower($email);

  if (!preg_match($emailPattern, $emailToCheck))
    return false;

  return true;
}
```

# Стиль кода

## Отступы

1. Для отступа использовать пробелы вместо табуляции. В IDE настроить замену табуляции пробелами. Tab = 3 пробела.
2. Длина строки 120 символов. Если строка более чем 120 символов, то перенесите на другую строку.
3. Не размещайте несколько инструкций на одной строке. Каждая инструкция должна начинаться с новой строки.

## Пустые строки

1. Две пустые строки между логическими секциями в исходном коде.
2. Две пустые строки между объявлениями классов и интерфейсов.
3. Одна пустая строка между методами.
4. Одна пустая строка между логическими частями в методе.

## Пробелы

1. После запятой должен быть пробел. После точки с запятой, если она не последняя в строке(например в инструкции for), должен быть пробел. Перед запятой и точкой с запятой пробелы не ставятся.
2. Все операторы должны быть отделены от операндов пробелами с обеих сторон.

Примеры:

```
$subscriptionStatus = GetSubscriptionStatus($userEmail, $subscriptionTypeID);
$subscriptionStatus=GetSubscriptionStatus($userEmail,$subscriptionTypeID);
```

```
$userEmail='test@test.ru';      // неверно
$userEmail = 'test@test.ru';    // верно
```

```
for ($i=0;$i<10; $i++)          // неверно
{
```

```
}
for ($i = 0; $i < 10; $i++)       // верно
{
}
```

## Одинарные и двойные кавычки

1. Если необходимо вывести текст как есть, то всегда используем одинарные кавычки.
2. Если необходимо добавить к тесту значение какой-нибудь переменной, то используем подстановку(конкатенацию) строк.
3. Если строка содержит апострофы, управляющие последовательности или какие-нибудь специальные символы, то разрешается использование двойных кавычек. Особенно актуально для SQL-запросов.

Пример:

```
$query = "SELECT *
            FROM project_modules
           WHERE NAME = '" . $moduleName . "'";
```

```
$userEmail = 'test@test.ru';
echo 'Email is: ' . $userEmail;
```

### Различия

Строку, заключённую в одинарные кавычки, интерпретатор php выводит как есть, а заключённую в двойные кавычки проверяет на наличие переменных и, найдя их, подставляет значения.

### Одинарные кавычки

Для того чтобы использовать одинарную кавычку внутри строки, заключенной в одинарные кавычки, передней нужно экранировать.

Лучше подходят для:

- Строк, которые не нужно анализировать на наличие переменных
- Имён переменных, которые нужно написать как текст.
- Строк, в которых не нужно обрабатывать специальные символы и управляющие последовательности.

### Двойные кавычки

Лучше подходят для:

- Escaped strings
- Строк с использованием переменных

## Переменные

1. Локальные переменные - стиль Кэмел.
2. Глобальные переменные – стиль Паскаль
3. Счётчики в циклах традиционно называют i, j, k, m, n.

4. Комментарии к переменной должны быть написаны так, чтобы было понятно назначение и способ использования переменной.

## Комментарии

1. Комментируя код, старайтесь объяснять, что он делает, а не какая операция производится.

## Константы

Имена констант указываются в верхнем регистре. Например: `const VERSION = '1.0';`

## Файлы

1. Во всех PHP файлах использовать окончания строк LFи кодировку utf-8.
2. Все PHP файлы необходимо заканчивать пустой строкой.
3. Закрывающий тэг «?>» необходимо удалять из файлов содержащих только PHP
4. Вместо "\n" используйте константу PHP_EOL

## Инструкции

1. Каждая инструкция должна располагаться на новой строке.
2. Одиночные инструкции оформляются без фигурных скобок на отдельной строке, сдвинутой на стандартный отступ.
3. Составные инструкции оформляются открывающей фигурной скобкой на отдельной строке, списком инструкция и закрывающей фигурной скобкой также на новой строке.
4. После ключевого слова (if, while) перед открывающей круглой скобкой должен быть пробел.

### Оформление if, if-else

#### If

// Верно:

```
if (isset($userEmail))
{
  GetSubscriptionStatus($userEmail, $subscriptionTypeID);
  ...
}

if (!isset($userEmail))
  return false;
```

// Не верно:

```
if(isset($userEmail)){
  GetSubscriptionStatus($userEmail, $subscriptionTypeID);
  ...
}
```

```
if (!isset($userEmail)) return false;
```

```
if (!isset($userEmail))
  {
  return false;
  }
```

### *If с последующим else*

// Верно:

```
if (isset($userEmail))
{
  $subscriptionStatus = GetSubscriptionStatus($userEmail, $subscriptionTypeID);
  ...
}
else
{
  DoSomethingElse();
  ...
}
```

```
if (!isset($userEmail))
  DoSomething();
else
  DoSomethingElse();
```

// Не верно:

```
if (isset($userEmail)){
  $subscriptionStatus = GetSubscriptionStatus($userEmail, $subscriptionTypeID);
  ...
} else {
  DoSomethingElse();
  ...
}
```

```
if (!isset($userEmail)) DoSomething(); else DoSomethingElse();
```

### *Оформление for, foreach*

### *for*

```
for ($i = 0; $i < 10; $++)
{
  ...
}
```

### foreach

```
foreach($items as $k => $v)
{
  DoSomething($v);
}
```

### *Оформление while, do-while*

### while

```
$i = 0;
while ($i <= 10)
{
  DoSomething($i);
  ...
}
```

### *do-while*

```
$i = 0;
do
{
  DoSomething($i);
  ...
}
while ($i > 0);
```

### *Оформление switch*

```
switch($paramType)
{
  case 0:
    $paramValue = "test";
    break;
  case 1:
    $paramValue = "test1";
    break;
  default:
    $paramValue = "test2";
}
```

## SQL запросы

1. Не смешивать php-код и запросы к базе данных. Выносить SQL-запросы к базе данных из кода в отдельный метод или класс.
2. Названия методов должно состоять из ключевого слова определяющего действие или возвращаемого результата, и названия отражающего то, что данный метод делает и/или

название поля по которому будет осуществляться какое-либо действие. Например, по названию метода GetModuleNameByID можно понять, что:

a. Возвращаемый результат будет в ед. числе, так как указано ключевое слово Get.
b. Вернется название модуля – ModuleName
c. Поиск названия модуля будет осуществляться по его ID.

3. Ключевые слова писать с новой строки с выравниванием по правой границе первого ключевого слова.
4. Каждая из таблиц, перечисленных в FROM должна быть записана на одной строке, если количество символов превышает длину строки(см. Отступы), то необходимо перенести название таблицы на новую строку и выровнять по левому краю первой таблицы.
5. Операторы следует выравнивать друг под другом.

Таблица с ключевыми словами:

| | |
|---|---|
| Get | Результат метода вернёт значение в ед. числе |
| Select | Результат метода вернет набор данных, массив и т.д. |
| Set или Add | Вставка данных |
| Delete | Удаление данных |

Пример: Для примера использовался файл /lib/modules/module.class.php

1. В данном файле конструкция вида: *$rec=SQLSelectOne("SELECT * FROM project_modules WHERE NAME='".$this->name."'");* встречается 5 раз. Т.е. если нужно будет что-то изменить, то придётся 5(пять) раз изменять код по всему файлу. Вместо этого, достаточно один раз создать метод SelectProjectModulesByName и обращаться к нему.
2. Тоже самое относится к удалению модуля.
3. Читаемость «неправильного» метода намного ниже чем «правильного».

Неправильно:

```
/**
* UnInstalling current module
*
* Removing information about module in project registry and
*
* @access private
*/
 function uninstall() {
  $rec=SQLSelectOne("SELECT * FROM project_modules WHERE NAME='".$this->name."'");
  if (IsSet($rec["ID"])) {
   SQLExec("DELETE FROM project_modules WHERE ID='".$rec["ID"]."'");
  }
  if (file_exists(DIR_MODULES.$this->name."/installed")) {
   unlink(DIR_MODULES.$this->name."/installed");
  }
 }
```

Правильно:

```
/**
 * Select project module data by name
 * @param mixed $moduleName Module name
 * @return array
 */
private function SelectProjectModulesByName($moduleName)
{
  $record = SQLSelectOne("SELECT *
                            FROM project_modules
                           WHERE NAME = '" . $moduleName . "'");

  return $record;
}

/**
 * Delete module by ID
 * @param mixed $moduleID Module id
 */
private function DeleteProjectModuleByID($moduleID)
{
  SQLExec("DELETE
             FROM project_modules
            WHERE ID = '" . $moduleID . "'");
}

/**
 * Removing information about module in project registry
 */
private function uninstall()
{
  $rec = SelectProjectModulesByName($this->name);

  if (isset($rec["ID"]))
    DeleteProjectModuleByID($rec["ID"]);

  if (file_exists(DIR_MODULES.$this->name."/installed"))
     unlink(DIR_MODULES.$this->name."/installed");
}
```

## Файловая структура проекта

- majordomo – корневая директория
  - backups - бэкапы системы

- documents - документация к проекту
- logs - логи
- sources – исходный код проекта.
  - resources – локально хранимые ресурсы проекта
    - css – css файлы
    - js - javascript файлы и библиотеки
    - images - картинки
    - data - остальные бинарные файлы
  - plugins - приложения из магазина
  - ….
  - vendors - файлы и библиотеки сторонних разработчиков