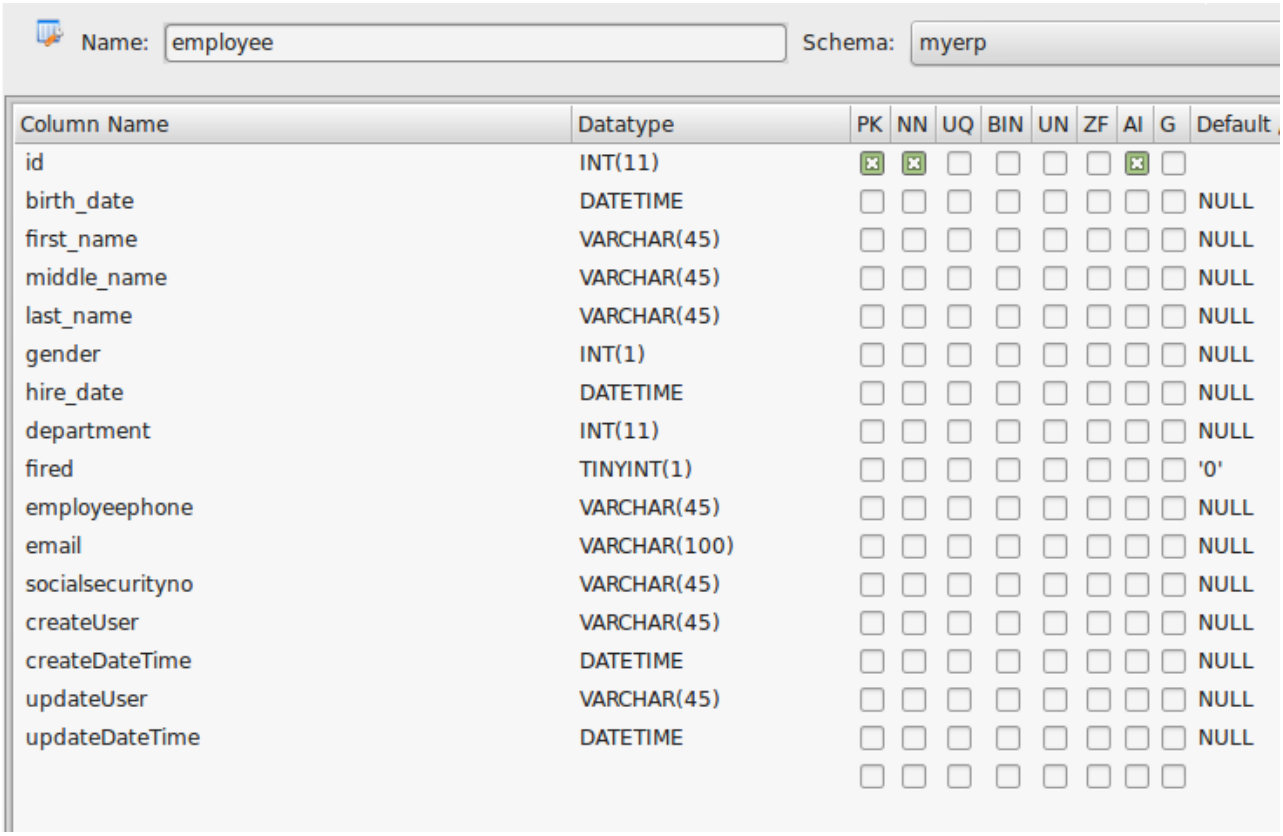
Класс Таблица (Table)

Используется для операций с данными (CRUD).

Разберем на примере таблицы Сотрудники (Employee).


Name: employee Schema: myerp

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default
id	INT(11)	☒	☒	☐	☐	☐	☐	☒	☐	
birth_date	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
first_name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
middle_name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
last_name	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
gender	INT(1)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
hire_date	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
department	INT(11)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
fired	TINYINT(1)	☐	☐	☐	☐	☐	☐	☐	☐	'0'
employeephone	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
email	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
socialsecurityno	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
createUser	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
createDateTime	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
updateUser	VARCHAR(45)	☐	☐	☐	☐	☐	☐	☐	☐	NULL
updateDateTime	DATETIME	☐	☐	☐	☐	☐	☐	☐	☐	NULL
		☐	☐	☐	☐	☐	☐	☐	☐	

Public functions

Созадим переменную объект типа Table ($employee) .

1. **void init() -** Метод инициализирует поля объекта значениями по-умолчанию установленными в базе данных.

   $employee->init();

2. **int insert(bool $commit <default true>)** - Метод вставки записи в физическую таблицу

   $employee→init();
   $employee→set_First_name('Иван');
   $employee→set_Last_name('Иванов');
   $employee→insert();

   Код выше вставляет запись с определенными значениями в базу данных. Возвращает id вставленной записи или false (0) если вставка не произведена.

   Если требуется вставить несколько значений в разные таблицы в рамках одной транзакции, то метод insert необходимо вызывать с параметром false. Вызов без параметра или true закроет текущую транзакцию.

   $employee→init();
   $employee→set_First_name('Василий');
   $employee→set_Last_name('Пупкин');
   $employee→insert(false);
   $employee→init();

```
$employee→set_First_name('Иван');
$employee→set_Last_name('Иванов');
$employee→insert(false);
// После всех вставок можно закрыть транзакцию
```

3. **void deleteById(int $id, bool $commit <default true>);**
   Удаление строки из таблицы с переданным id. В примере из БД будет удалена запись с id=1

   ```
   $employee->deleteByID(1);
   ```

4. **void getById(int$id);**
   Инициализарует поля объекта значениями из таблицы в БД. В примере поля будут инициализированы значениями из строки таблицы с id=1.

   ```
   $employee->getById(1);
   ```

5. **void update(bool $commit);**
   Сохранить новые значения в БД.

   ```
   // получаем строку с id=1
   $employee→getById(1);
   // устанавливаем новое значеие в поле first_name
   $employee→set_First_name('Вася');
   // обновляем значения в БД
   $employee→update();
   ```

6. **void setFilter(string column, string type, [mixed firstvalue, [mixed secondvalue]])**
   Метод определяет условия для выборки из БД
   В данном примере нам нужны все записи у которых
   first_name = Иван
   department любое из значений 1,2,3 или 4
   дата приема между значениями ***$date1*** и ***$date2***

   ```
   $employee→setFilter('first_name','=','Иван');
   $employee->setFilter('department','IN',[1,2,3,4]));
   $employee→setFilter('hire_date','BETWEEN', $date1,$date2);
   $employee->fetchAll();
   ```

   Второй параметр метода **type** может принимать следующие значения:
   1. '='- равно **firstvalue**
   2. '>'- больше **firstvalue**
   3. '>=' - больше либо равно **firstvalue**
   4. '<' - меньше **firstvalue**
   5. '<=' - меньше либо равно **firstvalue**
   6. '<>' - не равно **firstvalue**
   7. 'BETWEEN' значение между **firstvalue** и **secondvalue**
   8. 'NOT BETWEEN' значение не входит между **firstvalue** и **secondvalue**
   9. 'IN' значение входит в массив значений **firstvalue** (может быть массив или строка с разделителем запятая)
   10. 'NOT IN'значение **НЕ** входит в массив значений **firstvalue** (может быть массив или строка с разделителем запятая)
   11. 'LIKE' значение содержит **firstvalue**
   12. 'LIKE _%' начинается с **firstvalue**
   13. 'LIKE %_' заканчивается на **firstvalue**

7. **void reset()**

Сбрасывает все установленные фильтры, сортировки и ограничение выборки.

```
$employee→reset();
```

8. **void setOrder(string columns, string ordertypes <default ASC>)**

Устанавливает сортировку записей. ordertypes принимает только 2 значения ASC или DESC

```
$employee→setOrder('first_name','DESC');
```

Сортировка по нескольким полям

```
$employee→ setOrder('first_name,last_name', 'ASC,DESC');
```

9. **void setCursorColumns(array columns);**

Устанавливает колонки для выборки, если вам нужны не все колонки из таблиы БД

```
$employee->setCursorColumns(['first_name','last_name']));
print_r($employee→fetchAll());
```

Результат: ассоциативный массив с ключами id, first_name, last_name

*Внимание: результат всегда будет содержать ключ **id***

10. **void setLimit(arraylimit)**

Ограничение количества записей в выборке

```
$employee->setLimit([10,30]);
```

Результат: 30 записей начиная с 10-ой в выборке

11. **bool getFirstRow();**

Инициализирует значения полей первой записью из выборки

```
$employee→setFilter('first_name','=','Иван');
$employee→ setOrder('hire_date','DESC');
$employee→ getFirstRow();
```

$employee поля будут инициализированы значениями строки из таблицы где имя Иван и самая поздняя дата приема на работу.
Если значение в таблице не найдено, то метод вернет false

```
if ($employee→ getFirstRow()) { echo ‘Найдена запись’;}
```

12. int getCount();

Возвращает количество строк с определенным фильтром

```
$employee→setFilter('first_name','=','Иван');
 echo $employee→getCount();
```

13. String getGroupConcat(string column<default 'id' >)

Метод возвтращает строку с разделителем запятая из значений определенной колони в выборке. Пример данных:

Id	first_name	last_name
1	Иван	Иванов
2	Вася	Пупкин

```
echo $employee→getGroupConcat('first_name');
Результат метода:'Иван,Вася'

echo $employee→getGroupConcat();
Результат метода:'1,2'
```

14. array fetchAll();

Возвращает ассоциативный массив со значениями выборки.

```
$employee→setCursorColumns(array('first_name','last_name'));
$employee→setFilter('id','IN','1,2');
var_dump($employee→fetchAll());

Результат: [
        0=> ['id'=>'1','first_name'=>'Иван','last_name'=>'Иванов'],
        1=>['id'=>'2','first_name'=>'Вася','lastname'=>'Пупкин']
        ];
```

15. array fetchHashMap(string column<default id>)

Возвращает ассоциативный массив со значениями выборки и ключем column

```
$employee→setCursorColumns(array('first_name','last_name'));
 $employee→setFilter('id','IN','1,2');
 var_dump($employee→fetchHashMap());

 Результат: array(1=> ['id'=>'1','first_name'=>'Иван','last_name'=>'Иванов'],
          2=>['id'=>'2','first_name'=>'Вася','last_name'=>'Пупкин']];

 var_dump($employee→fetchHashMap('first_name'));

 result: array(
 'Иван'=>array('id'=>'1','first_name'=>'Иван','last_name'=>'Иванов'),
 'Вася'=>array('id'=>'2','first_name'=>'Вася','last_name'=>'Пупкин')
 );
```

Внимание: Если значения в колонке-ключе не уникальные, то значения будут инициализированы последней записью подходящей под это значение ключа.

16. mixed getSum(string column), getAvg(string column), getMax(string column), getMin(string column)

Агрегатные функции.

```
$employee→ setFilter('first_name','=','Иван');
echo $employee→ getMax('hire_date');
```

Результат: Последняя дата приема сотрудника с именем Иван.

17. void bulkInsert(array records, bool commit<defaulttrue>)

Вставка нескольких строк из массива. Ключи должны совпадать с именами полей.

```
$records = array(0=> array('first_name'=>'Иван','last_name'=>'Иванов'),
                 1=>array('first_name'=>'Вася','last_name'=>'Пупкин'));

$employee→ bulkInsert($records);
```

Внимание: Ключи не совпадающие со значениями полей будут проигнорированы.

18. void setSelectionView(string viewName)

Установить View для выборки записей. Вы можете создать View в базе данных и присоединить к полям таблицы значения из других таблиц, тогда выборки будут произведены из этого объекта БД.

```
CREATE
VIEW `employee_view` AS
    SELECT
        `employee`.`id` AS `id`,
        `employee`.`birth_date` AS `birth_date`,
        `employee`.`first_name` AS `first_name`,
        `employee`.`middle_name` AS `middle_name`,
        `employee`.`last_name` AS `last_name`,
        `employee`.`gender` AS `gender`,
        `employee`.`hire_date` AS `hire_date`,
        `employee`.`department` AS `department`,
        `departments`.`dept_name` AS `department_name`,
        `employee`.`fired` AS `fired`,
        `employee`.`employeephone` AS `employeephone`,
        `employee`.`email` AS `email`,
        `employee`.`socialsecurityno` AS `socialsecurityno`,
        `employee`.`createUser` AS `createUser`,
        `employee`.`createDateTime` AS `createDateTime`,
        `employee`.`updateUser` AS `updateUser`,
        `employee`.`updateDateTime` AS
    `updateDateTime`FROM
(`employee`
LEFT JOIN `departments` ON ((`employee`.`department` = `departments`.`id`)))
```

id	birth_date	first_name	middle_name	last_name	gender	hire_date	department	department_name	fired	employeephone
1	1930-09-05 00:00:00	Neil	Alden	Armstrong	1	1955-09-05 00:00:00	1	Accounting	1	(895)564-97-56
2	1982-01-02 00:00:00	Andrew	S	Fuller	1	2016-11-04 00:00:00	2	Administration	1	(786)546-56-46
4	2016-07-04 00:00:00	John		Smith	1	2016-11-03 00:00:00	1	Accounting	1	(879)787-87-98
5	2016-11-23 00:00:00	Nancy		Nodier	2	2016-11-08 00:00:00	4	IT	1	(786)546-56-46
7	2016-11-02 00:00:00	Mike		Lemberg	1	2016-11-09 00:00:00	3	Finance	1	(785)489-55-49
8	1980-02-12 00:00:00	Alice		York	2	1986-11-01 00:00:00	9	Warehouse	1	(899)456-46-86

id	dept_name
1	Accounting
2	Administration
3	Finance
4	IT
5	Security
6	Transport
7	Logistics

```
$employee->setSelectionView('employee_view');
$employee->setFilter('department_name','=','Accounting');
var_dump($employee->fetchAll());
```

Результат : Ассоциативный массив со всеми полями из employee_view

Полезно, если вам необходимо агрегировать, фильтровать, сортировать по значениям из других таблиц.

```
$employee→setSelectionView(null); // Установить выборку данных только из таблицы
```

*Внимание: View должен включать поле **id (первичный ключ)**.*
Функции insert, update, delete будут отрабатывать только на данных таблицы employee не затрагивая полей из других таблиц.

Вы не сможете установить значения для поля department_name,
*но сможете его получить (**$employee->get_Department_name()**).*

Setters and Getters

1. Setter'ы и getter'ы создаются автоматически на основе данных таблицы.

Setter:**set_<Field name>(<Value>);**
Getter:**get _<Field name>();**
Внимание: После «set_» or «get_» идет название поля с большой буквы для визуального определения (чтобы не путать с другими функциями), что это функция устанавливает или получает данные из таблицы
get_First_name — правильно
get_first_name - н е п р а в и л ь н о

```
$employee→ getById(1);
echo $employee→get_First_name();
echo $employee→ get_Last_name();
```

result: Иван
Иванов

$employee→ getById(1);
$employee→ set_First_name('Ваня');
$employee→ update();

Результат : Изменили first_name со значения Иван на Ваня в

записи id == 1

переменная(объект $employee) также сохраняет предыдущее значения.

$employee→ getById(1);
$employee→ set_First_name('Ваня');
$employee→ update();
echo $employee→ getOld_First_name();

result:Иван (Несмотря на то, что мы изменили значение в БД, переменная продолжает

хранить предыдущее значение.)

Можно использовать для проверки изменилось ли текущее значение или нет

getOld_<Field_name>

2. Вместо setter'ов можно использовать функцию copyFieldValuesFrom():
(когда вам неизвестно сколько и каких полей необходимо обновить)

$empl = ['first_name'=>'Иван', 'last_name'='Иванов'];
// ассоциативный массив (ключи совпадают с названиями полей) или другой объект типа таблица Table

Очень удобно если значения переносятся из одной таблицы в другую. Можно просто добавить поля с одинаковыми именами и не надо будет искать/добавлять/исправлять в коде.

$employee→ init();
$employee→ copyFieldValuesFrom($empl);
$employee→ insert();

Внимание:
1.copyFieldValuesFrom не копирует ***id (т.к. по умолчанию там автоинкремент). Если требуется перенести вместе с id, то используйте функцию*** copyAllFieldValuesFrom()
2. keys in variable**$empl,**which don`t match field names in**$employee,**will beignored.

3. Getting tableobjects.

Если у таблицы в БД есть внешние ключи на другую таблицу, то можно получить объект Table на которую ссылается внешний ключ.

obj_<Field_name>(uppercase first letter).

В нашем примере таблица employee имеет внешний ключ на таблицу departments

Мы можем получить объект типа department

$department = $employee->obj_Department();
Внимание: Исходный объект (employee) должен быть инициализирован значениями, иначе данный метод вернет NULL

```
$employee->getById(1);
```

Табличные триггеры.

Класс Table имеет функции, которые вызываются при изменении, вставке, удалении строк в БД.

Функции: **onInsert(), onUpdate(), onDelete(), onAfterInsert(), onAfterUpdate(), onAfterDelete().**

Изначально эти функции не выполняют никаких действий. Если требуется выполнять какие-то операции в случае вставки/изменения/удаления строк, то их можно переписать в классе наследнике, либо, если нужны функции логирования, то можно переписать в исходном классе.

Также класс Table имеет полезные функции для отслеживания изменений.

Bool isChanged_<Field_name>() (Field_name uppercase first letter);

Пример использования:

```
protected function onUpdate(){

    if($this→isChanged_First_name()){

            какие-то действия при изменении поля first_name}

  }
```

Также внутри можно обращаться к другим объектам типа Table используя функцию

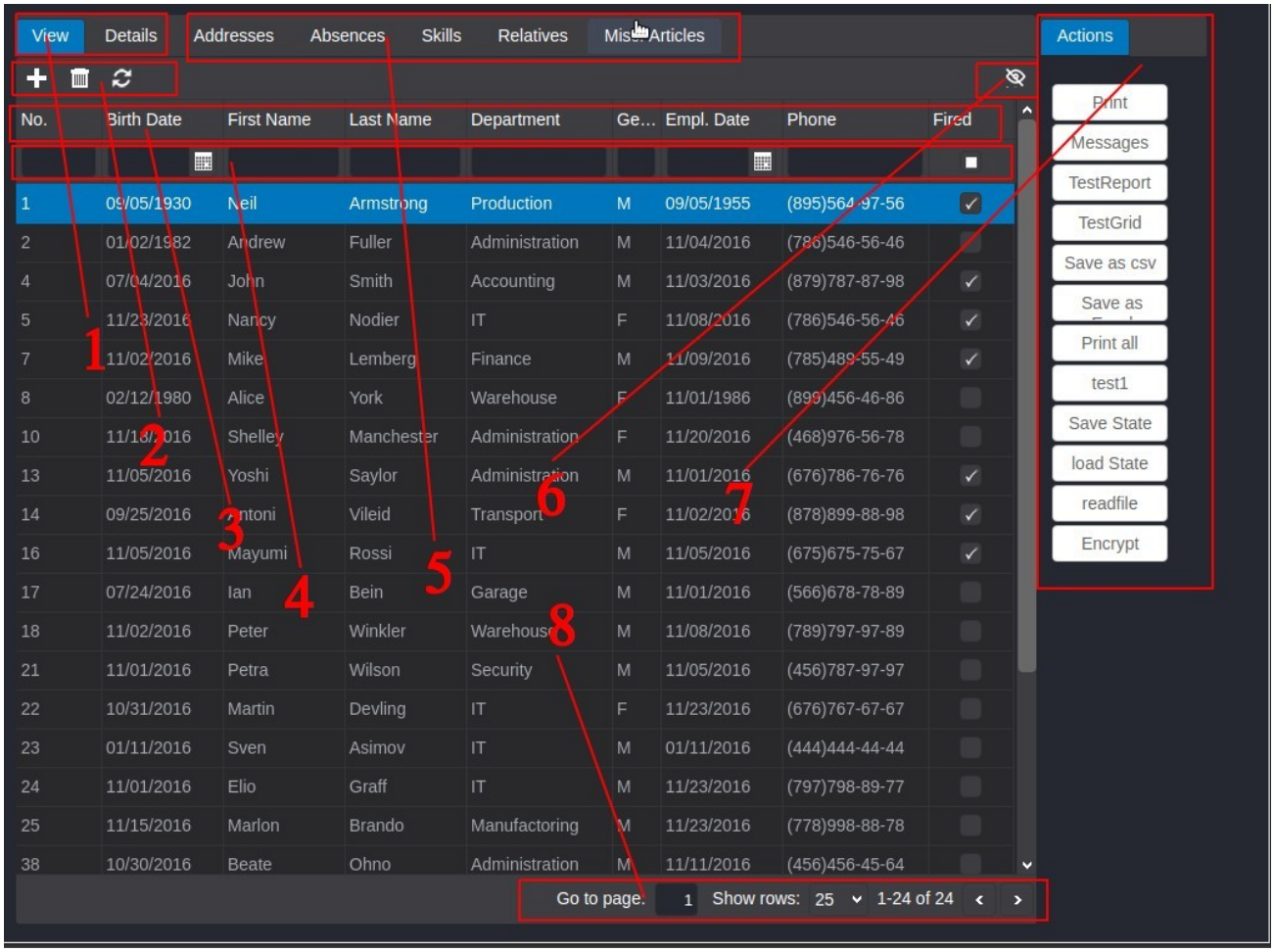
Table getTable(string tablename)

Пример:

```
protected function onUpdate(){
        if($this->isChanged_First_name()){
            $department = $this->getTable('departments');
            do something...
        }
}
```

Формы

Все формы имеют стандартный вид.



1. Закладка «Список» содержит список записей из таблицы. Здесь возможна навигация, выбор(отметки через клавишу Ctrl), фильтрация, сортировка, добавление, удаление, изменение записей.

Закладка «Обзор» содержит поля одной выбранной на закладке «Список» записи.

2. Панель с кнопками (Добавить, Удалить, Обновить)Toolbarwith columnnames
3. Заголовки столбцов таблицы
4. Панель с фильтрами
5. Панель с закладками (количество неограничено)
6. Скрыть/Показать столбцы на закладке Список
7. Панель с функиональными кнопками. Добавляются кнопки выполняющие различные базнес операции с данными.
8. Кнопки постраничной навигации.

Создание форм

Создадим форму для таблицы Сотрудники (Employee)

Откройте Меню → Администрирование → Разработка → Формы

Добавьте новую запись.
В поле **Название** запишем название для новой формы *employee*
На панели кнопок нажимаем кнопку Редактирование.

Для кода форм используется язык программирования Javascript

Требования:

Необходим объект основных настроек. Либо готовый объект, либо функция возвращающая объект определенной структуры. В нашем случае назовем объект
Maindata :

Он содержит обязательно следующие свойства именно с такими именами

{
dataSourceTableName: `<Имя физической таблицы в БД>`,
fields:<массив с описанием полей таблицы>
}

fields должен содержать следующие свойства
name : имя поля в таблице обязательно
type: может принимать одно из значений `number`, `string`, `date`, `bool` обязательно
label: отображение на форме обязательно
width: ширина поля в px обязательно
editable:**true**,**false** (по умолчанию **false**)
fieldformat:обязательно для полей типа type:'date'
gridColumnProperties:отдельная кастомизация полей (можно посмотреть в mfxGridColums.js)
input:optional (кастомизация полей на вкладке Обзор).
Свойства можно мосмотреть в файлах
type: number —mfxNumberInput.js
type: string -mfxInput.js
type: date -mfxDateTimeInput.js
type: bool -mfxCheckBox.js

Пример минимального кода для запуска формы:

```
var dataSourceTableName= 'employee'; // the name of table in database
// array of fields, describe every field as object
var fields= [
      {name:'id',type:'number',label:'#',width:70},
      {name:'birth_date',type:'date',label:'Дата рождения', width:100,fieldformat:'MM/dd/yyyy',editable:true,
      gridColumnProperties:{filtertype:'range'}},
      {name:'first_name',type:'string',label:'Имя',width:100,editable:true},
      {name:'middle_name',type:'string',label:'Отчество',width:100,editable:true,hidden:true},
      {name:'last_name',type:'string',label:'Фамилия',width:100,editable:true},
      {name: 'fired',type:'bool',label:'Уволен(а)',width:70,editable:true,gridColumnProperties:
{columntype:'checkbox'}}];
      var maindata= {dataSourseTableName:dataSourceTableName,fields:fields};
runForm(maindata);
```

showlabel: true, false надо ли показывать label на вкладке Обзор (по умолчанию **true**)

display: надо ли показывать поле ввода на вкладке Обзор (по умолчанию true)

описание объекта для выбора значений из другой таблицы.
dropdowngrid не очень удобен в использовании и ограничен в функционале. Лучше использовать popupgrid
Добавим поля на форму для выбора пола сотрудника

```
{name:'gender', type:'number', label:'Пол', width:50, editable:true, gridColumnProperties:{hidden:true},
showlabel:false, display:false
},

{name:'gender__shortname',type:'string',label:'Пол',width:40,
```

в свойство колонки добавим открытие другой таблицы для выбора значения

```
dropdowngrid:{datatable:'gender',

                needfield:'id',  // поле, которое выбираем из таблицы gender

                setfield:'gender', // поле, которое установим в таблице employee при клике на значение
                    fields:[     //набор полей в открывающейся форме таблицы gender
                                {name:'id',type:'number',label:'#',width:70,gridColumnProperties:{hidden:true}},
                                {name:'name',type:'string',label:'Пол',width:150}
                            ],
                    gridProperties:{showfilterrow:false,pageable:false,showheader:false}
                    }
}
```

gridProperties: можно посмотреть в файле mfxGrid.js

Если у таблицы есть внешний ключ на другую таблицу (в нашем примере таблица *employee содержит внешний ключ на* таблицу *gender*), то для отображения значений полей из внешней таблицы достаточно использовать в названии поля двойное подчеркивание.

В строке таблицы только одно поле gender со значением 1, но на форме отображается М (значение поля shortname из таблицы *gender*). Мы всего лишь определили имя колонки как *gender__shortname*(<поле из employee><двойное подчеркивание><поле из таблицы, на которую ссылается внешний ключ>)

#	id	birth_date	first_name	middle_na	last_name	gender	hire_date	department	fired	employeephone	email
1	1	2018-08-15 00:00:00	Neil	Alden	Armstrong	1	2018-08-09 00:00:00	11	1	(895)564-97-56	example

filterBefore: массив с фильтрами. Например в выпадающей форме нам необходимо отобразить только записи с *id* > 1.

filterBefore:[[`id`,`>`,`1`]] ([[filter1],[filter2],…,[filterN]])

popupgrid: все аналогично dropdowngrid, только откроется в
модальном окне.

Основное отличие popupgrid он может быть описан в виде функции с параметром текущая строка. Если вам необходимо открыть разные окна в зависимости от значений в текущей строке. Может быть полезно, например в формах заказов, если необходим выбор из разных таблиц или с разными фильтрами в зависимости от значений в текущей строке.

Пример со статичным объектом:

```
{name:'department',type:'string',label:'Department',width:40,

      popupgrid:{datatable:'department',needfield:'id',
                        fields:[    {name:'id',type:'number',label:'No.',width:70},
                                    {name:'dept_name',type:'string',label:'Department',width:150}
                              ],
                        gridProperties:{showfilterrow:false,pageable:true,showheader:true}
                        }
}
```

Пример с функцией:
в таблице ресурсы (*resources) заполняем поле* employeeid только если тип ресурса сотрудник допустим определили этот тип значением 1 (type=1), ресурсом может быть время работы оборудования или сырье для производства и т.д.

```
{name:'type',type: 'number',label:'Тип ресурса',width: 70,
{name:'employeeid',type: 'number',label:'Сотрудник',width:
                  70,popupgrid:function(currentrow){
                              if(currentrow.type!=1)returnundefined;//no popup window if type!

                              return{datatable:'employee',needfield:'id',
                                          fields:[
                                                      {name:'id',type:'number',label:'№',width:70},
                                                      {name:'first_name',type:'string',label:'Имя',width:150},
                                                      {name:'last_name',type:'string',label:'Фамилия',width:150}
                                                ]};
                                    }
}
```

Субформы (отображают связанные данные содержащиеся в других таблицах).

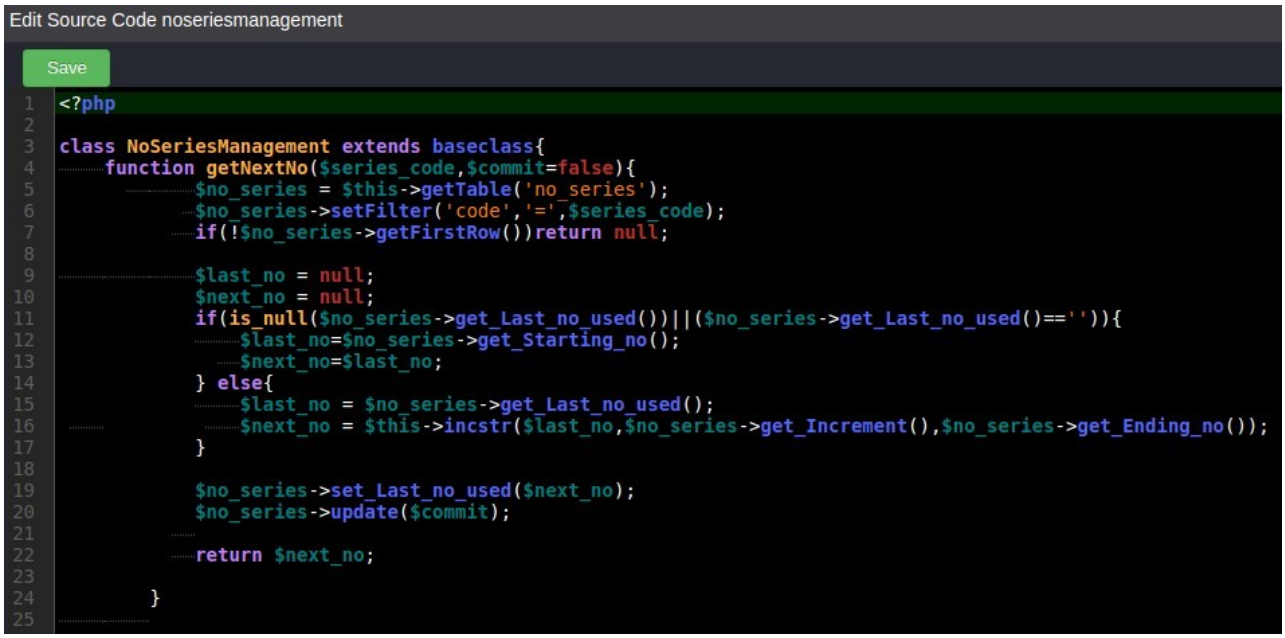
```
//
var subdata= [sub1, sub2… subN];
```

Классы

Администрирование → Разработка → Классы

Здесь должен содержаться прикладной бизнес функционал, оперирующий данными в БД.

Чтобы добавить новый класс, добавьте новую строку и заполните **название**. Далее нажмите кнопку Редактирование, чтобы внести прикладной код класса.
Используемый язык программирования PHP.

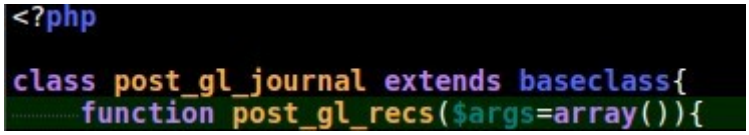


Если вашему классу потребуется взаимодействие с БД, то наследуйте его от **baseclass.**
Базовый класс содержит следующие функции:
getTable(<tablename>)- создает объект типа Table
getClass(<classname>)- создает объект другого класса, чтобы можно было использовать его функционал внутри вашего класса
beginTransaction()- открывает транзакцию
commitTransaction() - подтверждает транзакцию
rollbackTransaction()- откатывает изменения в БД

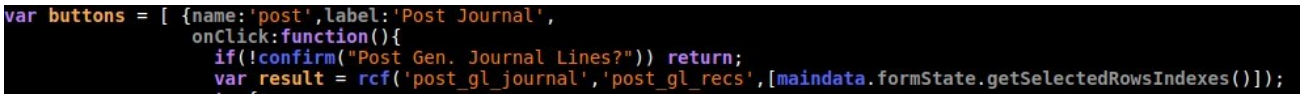
пример кода:

```
try{
      $this->beginTransaction();
      … do something withdata...
      $this->commitTransaction();
}catch(Exception $e){
… do something ifwrong….
$this->rollbackTransaction();
}
```

Если функция будет напрямую вызываться из кода JS на формах



пример вызова из Javascript функции onClick на кнопке формы:



(*Run Class Function*) function **rcf(<class name>,<function name>, array of arguments)**

Отчет

Администрирование → Разработка → Отчеты

Добавляем новую запись. Заполняем название отчета

Отчет должен возвращать HTML код, который будет выведен на печатной форме. Для верстки шаблона отчета можно использовать любой текстовый редактор. Например LibreOffice Writer, Microsoft Word и т.п. потом сохранить как HTML файл.

Шаблон отчета необходимо разбить на секции при помощи специальных тэгов в двойных квадратных скобках.

Например секция заголовок отчета. Дадим ей имя report_header. В этом случае нам следует использовать следующую разметку

[[report_header_begin]]
… здесь тело секции (некий html код)…
[[report_header_end]]

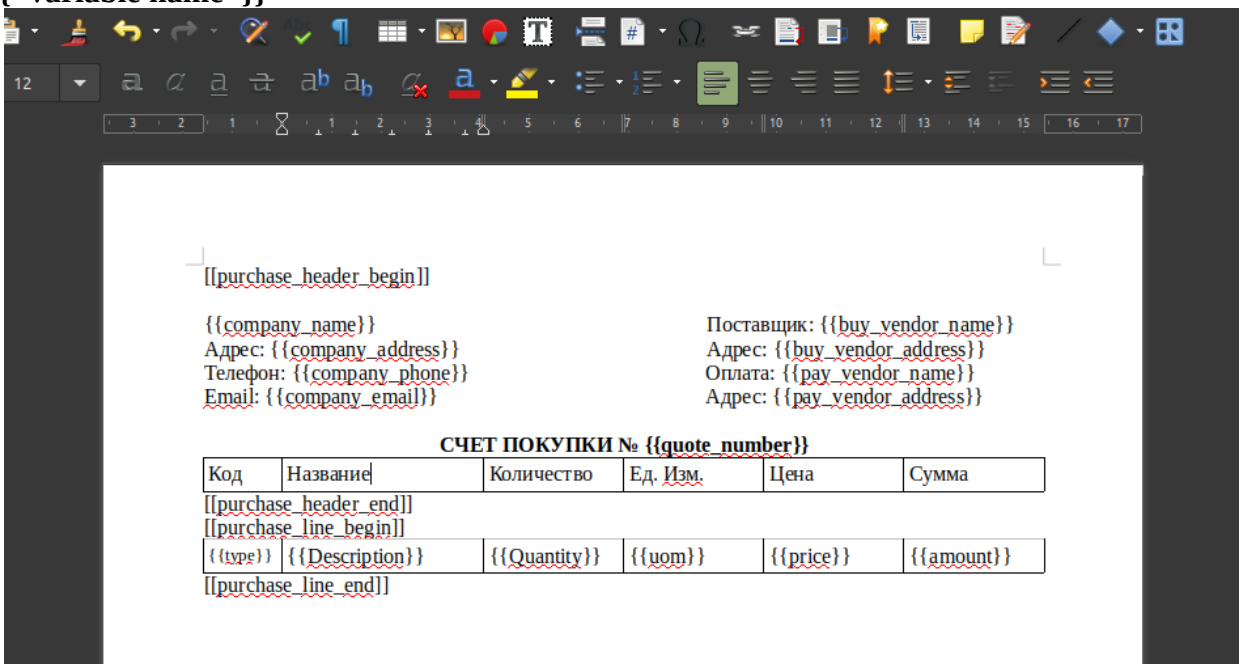
По этим тегам будет считан код секции автоматически.

Переменные в тэгах заключаются в двойные фигурные скобки

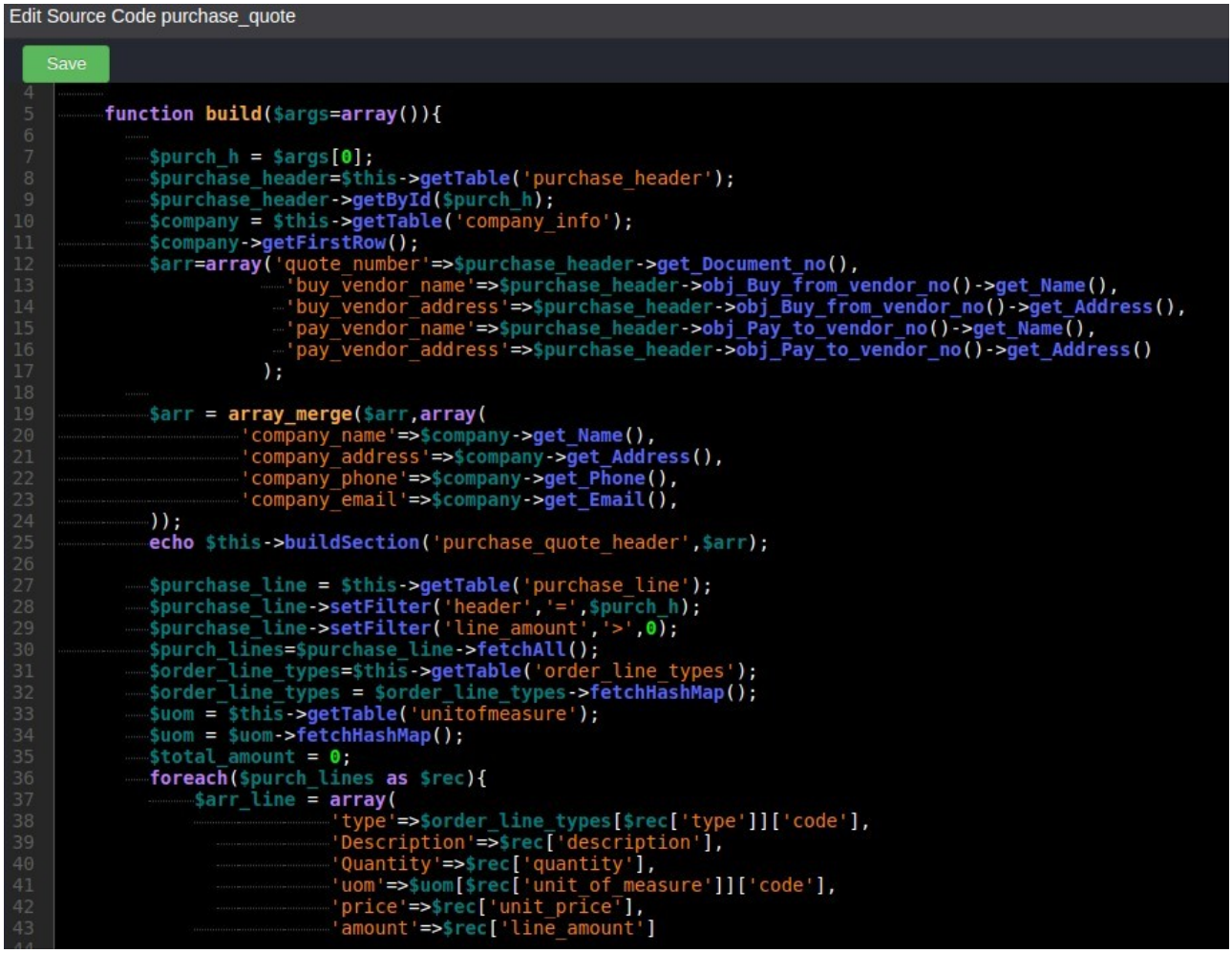
{{variable}}
Их значения передаются в составе массива и будут заменены значениями из этих переменных.

[[<section name>_begin]]… текст секции ...**[[<section name>_end]]**

{{<variable name>}}



Файл Шаблона (*.html) необходимо загрузить на сервер. Администрирование – Разработка – Отчеты . Кнопка **Загрузить шаблон**



На картинке не весь текст

```
<?php
    class Purchase_quote extends Report {
        function build($args=[]){
            ……................
        }
}
?>
```

Язык программирования PHP

Наследуйте ваш класс от Report. Переопределите функцию **build**

Заполните массив значениями Ключ (переменная в фигурных скобках) = ее значение.

```
$arr = [`quote_number`=>...,
  `buy_vendor_name`=>…,
  `buy_vendor_address`=>…,
  ….
];
```

Вызовите функцию **buildSection**

Можно собрать результат выполнения в переменную, можно сразу отправить клиенту через echo.

echo $this→buildSection*(<section name>, <array with values>);*
<section name> как вы назвали секцию в шаблоне. В примере 2 секции purchase_header выводится один раз, purchase_line выводится несколько раз в цикле.

На форме добавьте кнопку, вызовите функцию отчета и выведите на печать:

```
{name:'print', label:'Print', onClick:function(){
        var result = rr('purchase_quote',[maindata.formState.rowdata.id]);
        if(result!=''){
            print(result);

        }

}},
```

(*Run Report*) function **rr(<report name>,<arguments>)**
<arguments> будут переданы в переменную $args в функции **build** вашего отчета.
На картинке передается значение id из выделенной строки на форме.