

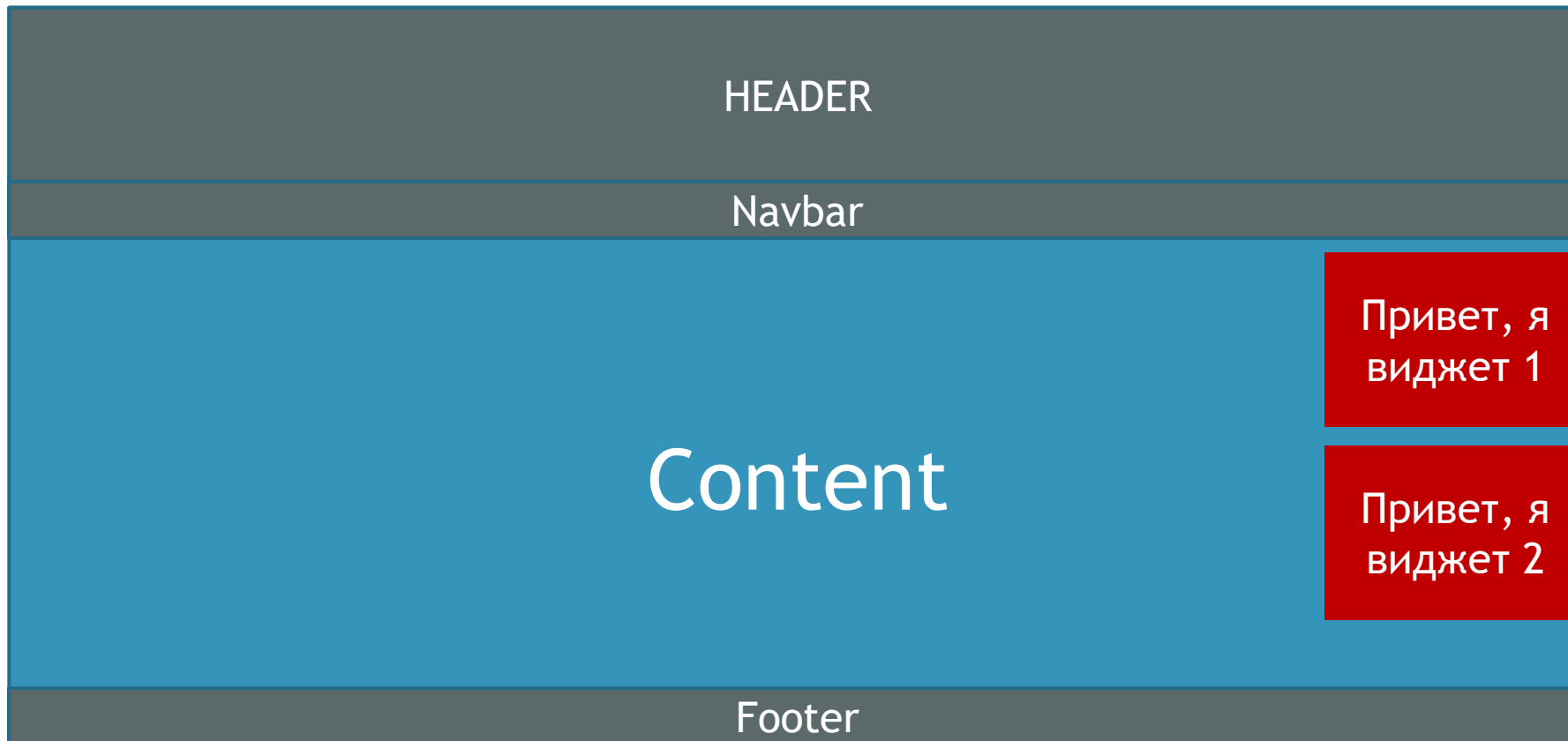
# Инструментальные средства ИС (2)

курс лекций

# Виджеты

# Виджеты

- ▶ Виджеты представляют собой многократно используемые строительные блоки, используемые в представлениях для создания сложных и настраиваемых элементов пользовательского интерфейса в рамках объектно-ориентированного подхода.

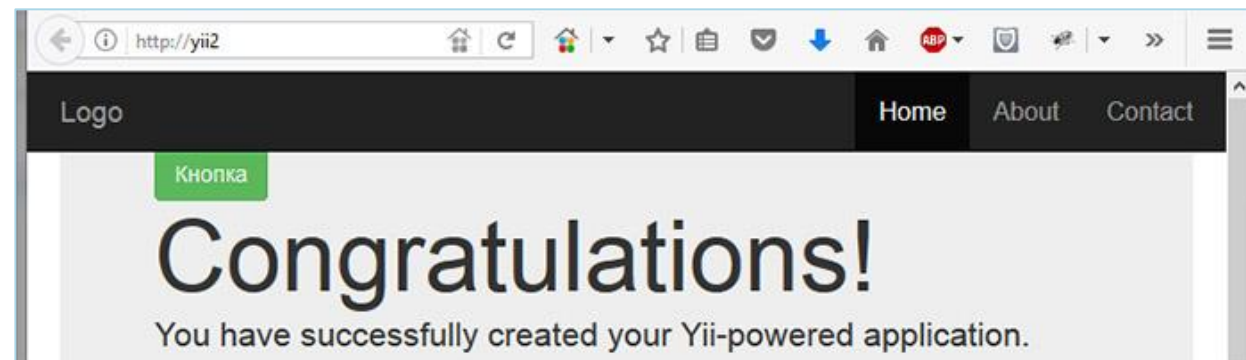


# Виджеты: виды

- ▶ Виджеты комплекта Yii: active form, menu, виджеты jQuery UI, виджеты Twitter Bootstrap, breadcrumbs, Navbar, виджеты для данных (ListView, GridView, DetailView);
- ▶ Собственные виджеты, реализующие элементы интерфейса, которые необходимо использовать многократно.
- ▶ b

# Виджет Navbar

```
<?php
    NavBar::begin([
        'options' => [
            'class' => 'navbar-inverse navbar-fixed-top',
        ],
        'brandLabel' => 'Logo',
        'brandUrl' => Yii::$app->homeUrl,
    ]);
    echo Nav::widget([
        'options' => ['class' => 'navbar-nav navbar-right'],
        'items' => [
            ['label' => 'Home', 'url' => ['/site/index']],
            ['label' => 'About', 'url' => ['/site/about']],
            ['label' => 'Contact', 'url' => ['/site/contact']],
        ],
    ]);
    NavBar::end();
?>
```



# Виджеты отображения данных

GridView









ListView

DetailView

# Виджеты отображения данных

GridView

Showing 1-4 of 4 items.

#	ID	Slug	Url	Name	Active	Crete datetime	Updated at	
		<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	
1	1		o-nas	О нас	Yes	Mar 30, 2016, 4:46:55 AM	2016-03-30 04:46:55	 
2	2		kontakty	Контакты	Yes	Mar 30, 2016, 4:47:08 AM	2016-03-30 04:47:08	 
3	3		portfolio	Портфолио	Yes	Mar 30, 2016, 4:47:38 AM	2016-03-30 04:47:38	 
4	4		akciya	Акция!	No	Mar 30, 2016, 4:47:49 AM	2016-03-30 04:47:49	 

DetailView

### View case

Update Delete

Ref No	#RERADRADAR
Case Description	SJNJSNSJNSJ
Raised On	2015-11-19 00:00:00
Status	inactive
Updated On	2015-11-17 00:00:00
Case Type	Robbery
Evidences	222222
Evidence Type	(not set)

# Виджеты отображения данных

- ▶ Виджет DetailView отображает детали по данным для одной model.
- ▶ Этот виджет лучше использовать для отображения данных модели в обычном формате(т.е. каждый атрибут модели будет представлен в виде строки в таблице). Модель может быть либо объектом класса yii\base\Model или его наследником, таких как active record, либо ассоциативным массивом.
- ▶ DetailView использует свойство \$attributes для определений, какие атрибуты модели должны быть показаны и в каком формате.

```
<?= DetailView::widget([  
    'model' => $model,  
    'attributes' => [  
        'id',  
        'category_id',  
        'title',  
        'excerpt',  
        'text:ntext',  
        'keywords',  
        'description',  
        'created',  
        'status',  
    ],  
) ?>
```



# Виджеты отображения данных

- ▶ Виджет ListView использует для отображения информации провайдера данных. Каждая модель отображается, используя определённый вид. Поскольку провайдер включает в себя разбивку на страницы, сортировку и фильтрацию, то его использование удобно для отображения информации конечному пользователю и создания интерфейса управления данными.
- ▶ Позволяет работать с моделью и дополнительно настраивать переменные:
  - ▶ \$key: mixed, значение ключа в соответствии с данными.
  - ▶ \$index: integer, индекс элемента данных в массиве элементов, возвращенных поставщику данных, который начинается с 0.
  - ▶ \$widget: ListView, это экземпляр виджета.

# Виджеты отображения данных

- ▶ Таблица данных или **GridView** - это один из сверхмощных Yii виджетов. Он может быть полезен, если необходимо быстро создать административный раздел системы. GridView использует данные, как провайдер данных и отображает каждую строку используя columns для предоставления данных в таблице.
- ▶ Каждая строка из таблицы представлена данными из одиночной записи и колонка, как правило, представляет собой атрибут записи (некоторые столбцы могут соответствовать сложным выражениям атрибутов или статическому тексту).

# Виджеты отображения данных

```
use yii\grid\GridView;  
use yii\data\ActiveDataProvider;  
  
$dataProvider = new ActiveDataProvider([  
    'query' => Post::find(),  
    'pagination' => [  
        'pageSize' => 20,  
    ],  
]);  
echo GridView::widget([  
    'dataProvider' => $dataProvider,  
]);
```

Создание провайдера данных

Отображение атрибутов для каждого элемента из провайдера данных с поддержкой сортировки и пагинации

# Виджеты отображения данных

- ▶ Колонки таблицы настраиваются с помощью определённых `yii\grid\Column` классов, которые настраиваются в свойстве `columns` виджета `GridView`.
- ▶ В зависимости от типа колонки и их настроек, данные отображаются по разному. По умолчанию это класс `yii\grid\DataColumn`, который представляет атрибут модели с возможностью сортировки и фильтрации по нему.
- ▶ Если `columns` не сконфигурирована, то `Yii` попытается отобразить все возможные колонки из провайдера данных.

```
echo GridView::widget([
    'dataProvider' => $dataProvider,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],
        // Обычные поля определенные данными содержащимися в $dataProvider.
        // Будут использованы данные из полей модели.
        'id',
        'username',
        // Более сложный пример.
        [
            'class' => 'yii\grid\DataColumn', // может быть опущено
            'value' => function ($data) {
                return $data->name; // $data['name'] для массивов
            },
        ],
    ],
]);
```

# Виджеты отображения данных

- ▶ В дополнение к классам колонок от Yii, вы можете самостоятельно создать свой собственный класс.
- ▶ Каждый класс колонки наследуется от `yii\grid\Column`, так что есть некоторые общие параметры, которые можно установить при настройке колонок.
  - ▶ `header` позволяет установить содержание для строки заголовка.
  - ▶ `footer` позволяет установить содержание для "подвала".
  - ▶ `visible` определяет, должен ли столбец быть видимым.
  - ▶ `content` позволяет передавать действительный обратный вызов, который будет возвращать данные для строки

# Виджеты отображения данных

- Data column используется для отображения и сортировки данных. По умолчанию этот тип используется для всех колонок.

```
'columns' => [  
  [  
    'attribute' => 'name',  
    'format' => 'text'  
  ],  
  [  
    'attribute' => 'birthday',  
    'format' => ['date', 'php:Y-m-d']  
  ],  
]
```

# Виджеты отображения данных

- ▶ ActionColumn отображает кнопки действия, такие как изменение или удаление для каждой строки. Доступные свойства для конфигурации:
  - ▶ **controller** это идентификатор контроллера, который должен обрабатывать действия.
  - ▶ **template** определяет шаблон для каждой ячейки в колонке действия. Маркеры заключённые в фигурные скобки являются ID действием контроллера
  - ▶ **buttons** массив из функций для отображения кнопок.
  - ▶ **visibleButtons** это массив условий видимости каждой из кнопок.

```
echo GridView::widget([
    'dataProvider' => $dataProvider,
    'columns' => [
        [
            'class' => 'yii\grid\ActionColumn',
            // вы можете настроить дополнительные свойства здесь.
        ],
    ],
]);
```

# Виджеты отображения данных

- ▶ Serial column выводит в строках номера начиная с 1 и увеличивая их по мере вывода строк.

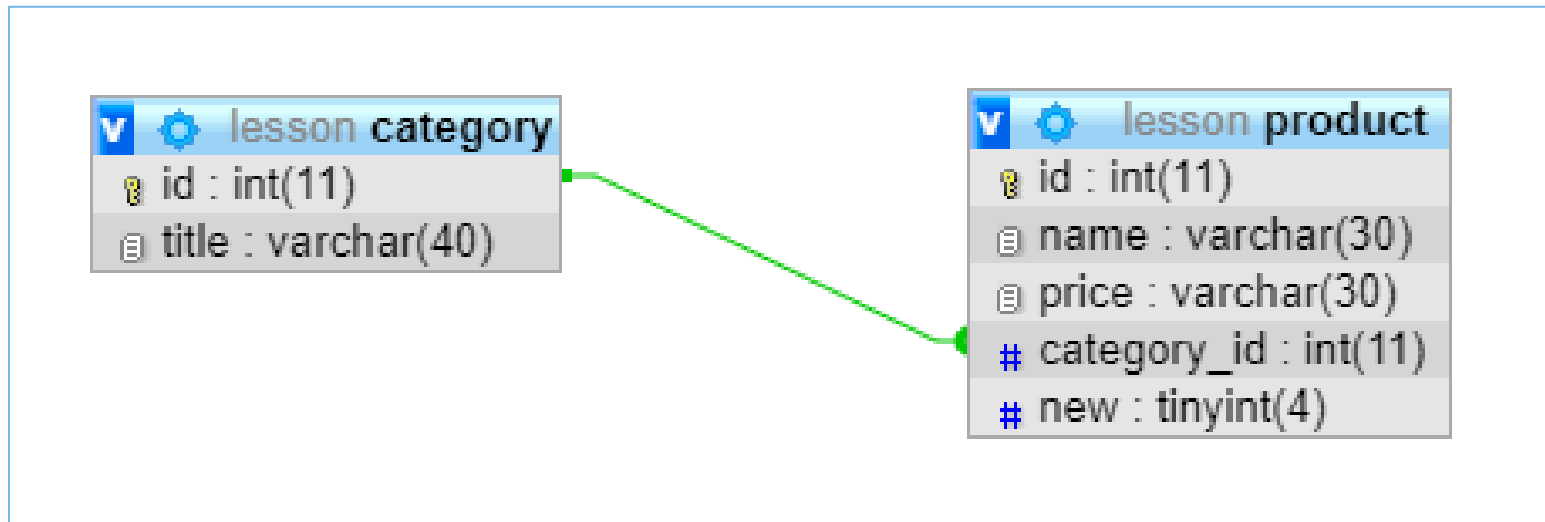
```
echo GridView::widget([  
    'dataProvider' => $dataProvider,  
    'columns' => [  
        ['class' => 'yii\grid\SerialColumn'], // <-- тут  
        // ...  
    ],  
]);
```



# Виджеты отображения данных

- ▶ Для фильтрации данных в GridView необходима модель, которая описывает форму для фильтрации, внося условия в запрос поиска для провайдера данных.
- ▶ Общепринятой практикой считается использование active records и создание для неё класса модели для поиска, которая содержит необходимую функциональность(может быть сгенерирована через Gii).
- ▶ Класс модели для поиска должен описывать правила валидации и реализовать метод search(), который будет возвращать провайдер данных.

# Пример настройки виджетов



# Пример настройки виджетов

- ▶ Создаем модели Product, Category
- ▶ Создаем CRUD

Create Product

Showing 1-12 of 12 items.

#	ID	Name	Price	Category ID	New	
1	17	Ряженка	56	1	0	  
2	18	Творог	34	1	0	  
3	19	Молоко	90	1	0	  
4	20	Сливки 33%	65	1	0	  
5	21	Глазированный сырок	18	1	0	  
6	22	Йогурт	31	1	0	  
7	23	Dove	120	2	0	  
8	24	Lindt	182	2	0	  
9	25	Alpen Gold	75	2	0 <sup>19</sup>	  

# Пример настройки виджетов

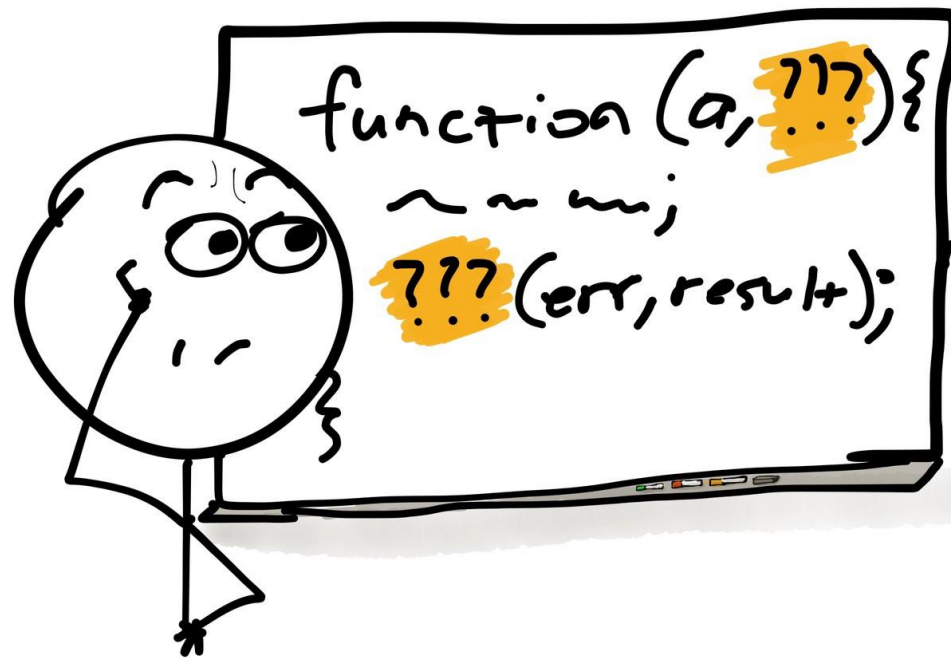
```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],

        'id',
        'name',
        'price',
        'category_id',
        'new',

        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
```

# Пример настройки виджетов

- ▶ Функция обратного вызова (callback) - это передача исполняемого кода в качестве одного из параметров другого кода. То есть обратный вызов функций позволяет в определенном методе исполнять код, который задается в ее аргументах. Функции обратного вызова, это - как правило обычные функции, которые напрямую никогда не вызываются. Функции обратного вызова обычно передаются как параметр для другой функции.



# Пример настройки виджетов

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,
    'columns' => [
        ['class' => 'yii\grid\SerialColumn'],

        'id',
        'name',
        'price',
        [
            'attribute' => 'category_id',
            'value' => function($data) {
                return $data->category->title;
            }
        ],
        [
            'attribute' => 'new',
            'value' => function($data) {
                return $data->new ? "<p>Нет</p>" : "<p>Да</p>";
            },
            'format' => 'html',
        ],
        ['class' => 'yii\grid\ActionColumn'],
    ],
]); ?>
```

Вывод названия категории из  
getCategory



















HTML-формат вывода данных

'status:boolean'

# Пример настройки виджетов

Create Product

Showing 1-12 of 12 items.

#	ID	Name	Price	Category ID	New	
1	17	Ряженка	56	Молочные продукты	Да	  
2	18	Творог	34	Молочные продукты	Да	  
3	19	Молоко	90	Молочные продукты	Да	  
4	20	Сливки 33%	65	Молочные продукты	Да	  
5	21	Глазированный сырок	18	Молочные продукты	Да	  
6	22	Йогурт	31	Молочные продукты	Да	  
7	23	Dove	120	Шоколад	Да	  
8	24	Lindt	182	Шоколад	Да	  
9	25	Alpen Gold	75	Шоколад	Да	  

# Пример настройки виджетов

```
<?= DetailView::widget([
    'model' => $model,
    'attributes' => [
        'id',
        'name',
        'price',
        [
            'attribute' => 'category_id',
            'value' => function($model) {
                return $model->category->title;
            }
        ],
        [
            'attribute' => 'new',
            'value' => function($data) {
                return $data->new ? "<p style='color:red'>Нет</p>" :
                    "<p style='color:green'>Да</p>";
            },
            'format'=>'html',
        ],
    ],
]) ?>
```

## Ряженка

[Update](#)[Delete](#)

ID	17
Name	Ряженка
Price	56
Category ID	Молочные продукты
New	Да



# Пример настройки виджетов

```
<?= $form->field($model, 'category_id')->dropDownList(  
    ArrayHelper::map(Category::find()->all(), 'id', 'title')  
    )  
?>
```



## Update Product: Ряженка

Name

Price

Category ID

Молочные продукты ▼

Молочные продукты

Шоколад

Чай

# Пример настройки виджетов

Получение массива объектов

```
public function actionIndex()
{
    $modelCategory = Category::find()->orderBy('title ASC')->all();
    foreach ($modelCategory as $value) {
        $arrCategory[$value->id] = $value->title;
    }

    $dataProvider = new ActiveDataProvider([
        'query' => Product::find(),
    ]);

    return $this->render('index', [
        'dataProvider' => $dataProvider,
        'arrCategory' => $arrCategory,
    ]);
}
```

Передача в представление

# Пример настройки виджетов

//1 способ

```
[
    'attribute' => 'category_id',
    'value' => function($data) {
        return $data->category->title;
    }
],
```

//2 способ

```
[
    'attribute' => 'category_id',
    'value' => 'category.title',
],
```

## Products

Create Product

Showing 1-12 of 12 items.

#	ID	Name	Price	Category ID	Category ID	New	
1	17	Ряженка	56	Молочные продукты	Молочные продукты	Да	👁️✎️🗑️
2	18	Творог	34	Молочные продукты	Молочные продукты	Да	👁️✎️🗑️
3	19	Молоко	90	Молочные продукты	Молочные продукты	Нет	👁️✎️🗑️
4	20	Сливки 33%	65	Молочные продукты	Молочные продукты	Да	👁️✎️🗑️
5	21	Глазированный сырок	18	Молочные продукты	Молочные продукты	Да	👁️✎️🗑️
6	22	Йогурт	31	Молочные продукты	Молочные продукты	Да	👁️✎️🗑️
7	23	Dove	120	Шоколад	Шоколад	Да	👁️✎️🗑️

# Пример настройки виджетов

```
public function createAction()
{
    $model = new Product();

    $modelCategory = Category::find()->orderBy('title ASC')->all();
    foreach ($modelCategory as $value) {
        $arrCategory[$value->id] = $value->title;
    }

    if ($model->load(Yii::$app->request->post()) && $model->save()) {
        return $this->redirect(['view', 'id' => $model->id]);
    }

    return $this->render('create', [
        'model' => $model,
        'arrCategory' => $arrCategory,
    ]);
}
```



```
<?= $this->render('_form', [
    'model' => $model,
    'arrCategory' => $arrCategory,
]) ?>
```

# Пример настройки виджетов

```
<?= $form->field($model, 'category_id')->dropDownList(  
    ArrayHelper::map(Category::find()->all(), 'id', 'title')  
)  
?>  
  
<?= $form->field($model, 'category_id')->dropDownList($arrCategory) ?>
```



## Create Product

Name

Price




Category ID

Category ID

# Пример настройки виджетов

```
['class' => 'yii\grid\ActionColumn',  
 'template' => '{delete}',  
],
```



#	ID	Name	Price	Category ID	Category ID	New	
1	17	Ряженка	56	Молочные продукты	Молочные продукты	Да	
2	18	Творог	34	Молочные продукты	Молочные продукты	Да	
3	19	Молоко	90	Молочные продукты	Молочные продукты	Нет	

# Пример настройки виджетов

```
[
  'label' => "VK",
  'format' => 'raw',
  'value' => function($data) {
    return Html::a('VK', $data->url);
  }
]
```



## Products

Create Product

Showing 1-12 of 12 items.

#	ID	Name	Price	Category ID	Category ID	New		VK
1	17	Ряженка	56	Молочные продукты	Молочные продукты	Да		VK
2	18	Творог	34	Молочные продукты	Молочные продукты	Да		VK
3	19	Молоко	90	Молочные продукты	Молочные продукты	Нет		VK

# Пример настройки виджетов

```
<?= GridView::widget([
    'dataProvider' => $dataProvider,

    'tableOptions' => [
        'class'=> 'table table-hover'
    ],





    'rowOptions' => [
        'class' => 'bg-danger'
    ],
])
```



## Products

Create Product

Showing 1-12 of 12 items.

#	ID	Name	Price	Category ID	Category ID	New	VK
1	17	Ряженка	56	Молочные продукты	Молочные продукты	Да	 VK
2	18	Творог	34	Молочные продукты	Молочные продукты	Да	 VK
3	19	Молоко	90	Молочные продукты	Молочные продукты	Нет	 VK
4	20	Сливки 33%	65	Молочные продукты	Молочные продукты	Да	 VK



# Создание виджетов

# Создание виджета

- ▶ Создать папку `components` в корне приложения
- ▶ Создать новый файл Виджета `ИмяWidget`. Проверить использование внутри класса пространства имен `app\components`
- ▶ Создать класс виджета - наследник базового класса `yii\base\Widget`
- ▶ Переопределить внутри класса методы:
  - ▶ `init` (нормализация свойства виджета - установка его значений)
  - ▶ `run` (*возвращение результата рендеринга*)
- ▶ Вернуть результат выполнения через `return` или вывести на экран `echo`
- ▶ В случае формирования виджетом большого фрагмента html-кода, создать рядом с классом виджета папку `view` и поместить в нее представление для виджета
- ▶ Вызвать виджет статичным методом `widget()` внутри представления

# Создание виджета-1

```
<div style="width:200px; background-color: red">
    hello
</div>
```



```
<?php

namespace app\components;

class HelloWidget extends \yii\base\Widget {

    public function run() {
        return $this->render('hello');
    }
}
```

# Создание виджета-1

```
<?= app\components\HelloWidget::widget() ?>
```



## About

hello

This is the About page. You may modify the following file to customize its content:

D:\openserver\OSPanel\domains\localhost\basic\views\site\about.php

# Создание виджета-2

```
namespace app\components;

class HelloWorldWidget extends \yii\base\Widget {

    public $name;

    public function init() {
        if ($this->name===null) {
            $this->name='???';
        }
    }

    public function run() {
        return $this->render('hello', ['name'=>$this->name]);
    }
}
```

Инициализация параметра-свойства


Представление виджета

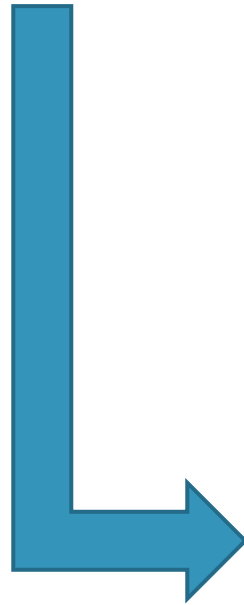
```
<div style="background-color:red">
    hello, <?= $name ?>
</div>
```

Вызов

```
<?= app\components\HelloWidget::widget(['name'=>"Mary"]) ?>
```

# Создание виджета-3

#	Имя	Тип	Сравнение
1	id 	int(11)	
2	name	varchar(30)	utf8mb4_unicode_ci
3	price	varchar(30)	utf8mb4_unicode_ci
4	description	text	utf8mb4_unicode_ci



```
<?php
namespace app\models;
use yii\db\ActiveRecord;

class Product extends ActiveRecord {

    public static function tableName() {
        return 'product';
    }

}
```

# Создание виджета-3

```
<?php
namespace app\components;
use yii\base\Widget;
use app\models\Product;

class ProductWidget extends Widget{

    public function run()
    {
        $pr = Product::find()->count();
        return $this->render('product', compact('pr'));
    }
}
```

Подсчет числа записей

Передача в представление виджета

# Расширения-виджеты <https://github.com/kartik-v/yii2-widgets>

