Первое приложение на AngularJS

Последнее обновление: 1.11.2015

В предыдущей теме мы вкратце познакомились с AngularJS, теперь создадим первое приложение, которое позволит глубже ощутить фреймворк.

Для написания приложений нам потребуется обычный текстовый редактор, хотя можно использовать специальные среды программирования, как Visual Studio, Netbeans, PhpStorm и другие.

Кроме того, ряд задач AngularJS может включать работу с веб-сервером (ajax-запросы и т.д.), поэтому рекомендую настроить веб-сервер и помещать все файлы приложения на AngularJS на веб-сервере. Хотя в большинстве задач можно обойтись и без веб-сервера, просто открыв нужный html-файл в браузере. В качестве сервера опять же можно использовать множество различных серверов - Apache, IIS, NodeJS и т.д.

Ключевой особенностью AngularJS является использование паттерна MVC, предполагающее разделение приложения на три функциональные части: контроллер, представление и модель. В процессе содания первого приложения мы рассмотрим их взаимодействие.

Итак, создадим простую html-страницу со следующим кодом:

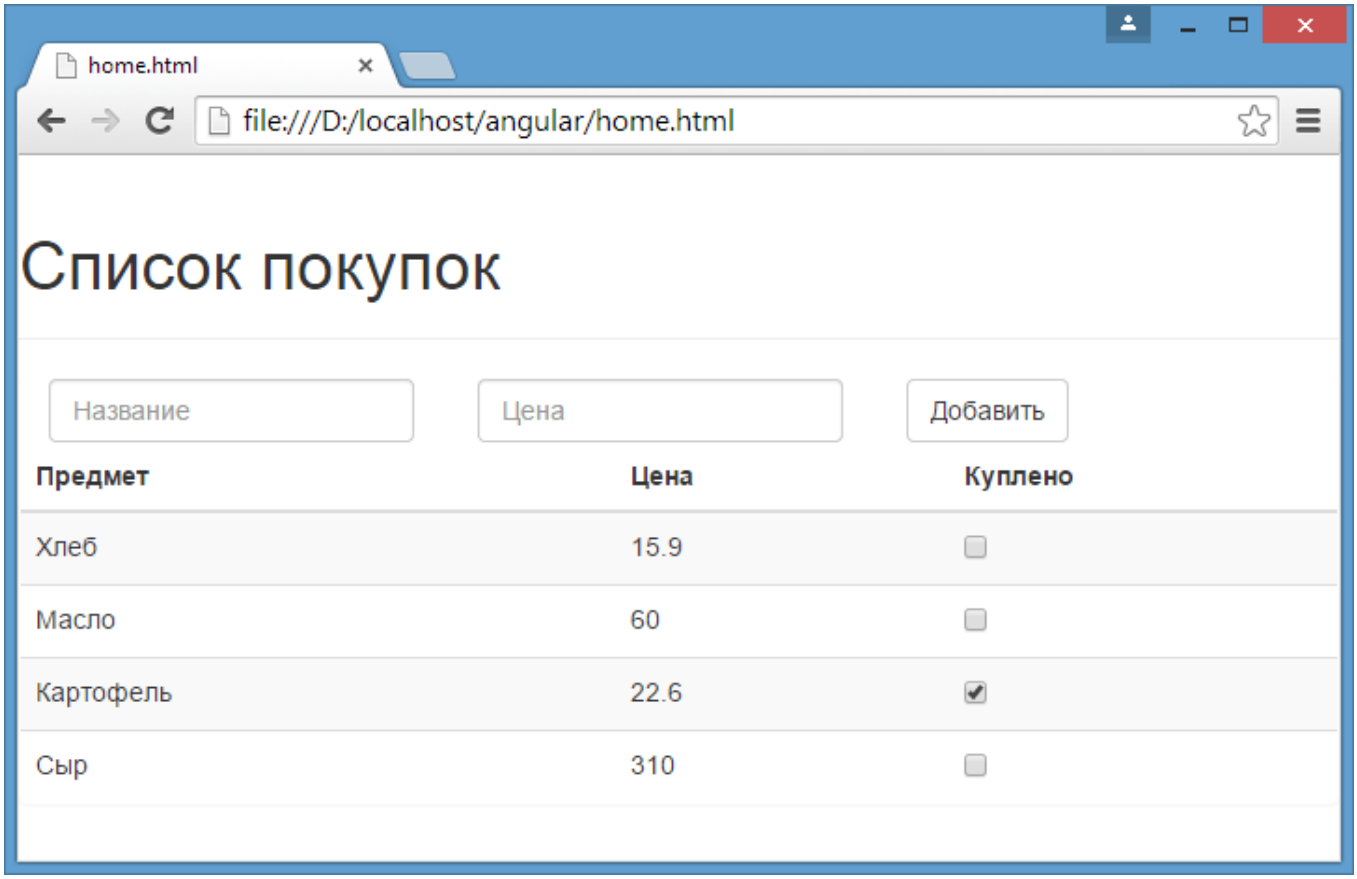
```
<!doctype html>
<html ng-app="purchaseApp">
<head>
<meta charset="utf-8">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.2
/css/bootstrap.min.css">
</head>
<body ng-controller="purchaseController">
    <div class="page-header">
        <h1> Список покупок </h1>
```

```
    </div>
    <div class="panel">
        <div class="form-inline">
            <div class="form-group">
                <div class="col-md-8">
                    <input class="form-control" ng-model="text" placeholder = "Название" />
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-6">
                    <input type="number" class="form-control" ng-model="price"
placeholder="Цена" />
                </div>
            </div>
            <div class="form-group">
                <div class="col-md-offset-2 col-md-8">
                    <button class="btn btn-default" ng-click="addItem(text,
price)">Добавить</button>
                </div>
            </div>
        </div>
        <table class="table table-striped">
            <thead>
                <tr>
                    <th>Предмет</th>
                    <th>Цена</th>
                    <th>Куплено</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="item in list.items">
                    <td>{{item.purchase}}</td>
                    <td>{{item.price}}</td>
                    <td><input type="checkbox" ng-model="item.done" /></td>
                </tr>
            </tbody>
        </table>
    </div>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.3.11/angular.min.js"></script>
<script>
var model = {
    items: [
        { purchase: "Хлеб", done: false, price: 15.9 },
        { purchase: "Масло", done: false, price: 60 },
        { purchase: "Картофель", done: true, price: 22.6 },
        { purchase: "Сыр", done: false, price:310 }
    ]
};
var purchaseApp = angular.module("purchaseApp", []);
    purchaseApp.controller("purchaseController", function ($scope) {
    $scope.list = model;
    $scope.addItem = function (text, price) {
        price = parseFloat(price); // преобразуем введенное значение к числу
        if(text != "" && !isNaN(price)) // если текст установлен и введено число, то
добавляем
        {
            $scope.list.items.push({ purchase: text, price: price, done: false });
        }
    }
```

```
});
</script>
</body>
</html>
```

В итоге у нас получится следующее приложение:



Это небольшое приложение, представляющее список покупок. Сам список покупок представляет собой модель, которая определена в javascript:

```
var model = {
    items: [
        { purchase: "Хлеб", done: false, price: 15.9 },
        { purchase: "Масло", done: false, price: 60 },
        { purchase: "Картофель", done: true, price: 22.6 },
        { purchase: "Сыр", done: false, price:310 }
    ]
}
```

Модель представляет обычный javascript-объект, хранящий определенные данные.

Приложение состоит из одного или несколько модулей, поэтому вначале объявляется модуль `purchaseApp`:

```
var purchaseApp = angular.module("purchaseApp", []);
```

Ключевым звеном приложения является **контроллер**, который управляет бизнес-логикой:

```
purchaseApp.controller("purchaseController", function ($scope) {
```

```
    $scope.list = model;
    $scope.addItem = function (text, price) {
        price = parseFloat(price); // преобразуем введенное значение к числу
        if(text != "" && !isNaN(price)) // если текст установлен и введено число, то
добавляем
        {
            $scope.list.items.push({ purchase: text, price: price, done: false });
        }
    }
})
```

С помощью специальной функции `controller` у ранее определенного модуля создается контроллер `purchaseController`. Вторым параметром в эту функцию передается функция, которая устанавливает объект **$scope**. $scope - это специальный объект, через который контроллер передает данные в представление.

В данном случае в объекте $scope устанавливается объект `list`, хранящий все элементы модели, и метод `addItem`, который будет использоваться для добавления новых элементов в модель.

И третья часть - создание представление, которое является кодом html, содержащий различные элементы. Для стилизации представления в данном случае использует css-фреймворк bootstrap.

В представлении устанавливается связь с контроллером с помощью директивы `ng-controller`:

```
<body ng-controller="purchaseController">
```

Вначале представления идет форма из двух элементов ввода и кнопки. Каждый элемент ввода имеет привязку к определенному объекту с помощью директивы **ng-model**:

```
<input class="form-control" ng-model="text" placeholder = "Название" />
```

Для обработки нажатия кнопки определена директива `ng-click="addItem(text, price)"`. В качестве обработчика здесь используется та функция, которая определена в `$scope.addItem`. А благодаря привязке элементов управления к моделям (ng-model="text") AngularJS будет знать, что в качестве параметров text и price надо подставлять в функцию значения соответствующих элементов управления.

При создании таблицы происходит перебор всех элементов в списке `$scope.list.items` с помощью директивы **ng-repeat="item in list.items"**, которая во многом аналогична действию циклов в стиле "foreach".

Назад Содержание Вперед

Владислав Грабовский • 9 месяцев назад

это лучшее объяснение работы енгуляра, евер

1 • Ответить • Поделиться ›

Илья • 12 дней назад

Есть ли метод обратный push()?

• Ответить • Поделиться ›

Metanit Модератор Илья • 12 дней назад

да метод pop - подробнее про работу с массивами - http://metanit.com/web/javascr...

• Ответить • Поделиться ›

Alexander • 8 месяцев назад

Приветствую всех, коллеги!
Хотел бы предложить несколько подправленный свой вариант контроллера 'purchaseController', потому что если пользователь кликает на "Добавить" и при этом не ввел никакую информацию в поля text и price, то происходит автоматическое добавление пустых строчек с checkbox, поэтому пользователю необходимо перекрыть такую возможность...

```
purchaseApp.controller("purchaseController", function ($scope) {
$scope.list = model;

$scope.text = ""; // добавлено
$scope.price = ""; // добавлено

$scope.addItem = function (text, price) {
if($scope.text != "" && $scope.price != "") // добавлено
{ // добавлено
$scope.list.items.push({ purchase: text, price: price, done: false });
$scope.text = ""; //очистка полей после ввода
$scope.price = ""; //очистка полей после ввода
} // добавлено

}
});
```

• Ответить • Поделиться ›

Metanit Модератор Alexander • 8 месяцев назад

поправил, но ради справедливости надо сказать, что надо гораздо больше поверок, чтобы валидировать введенные значения. Для этого лучше использовать регулярные выражения

r | Google+ | Канал сайта на youtube | Помощь сайту

ı: metanit22@mail.ru