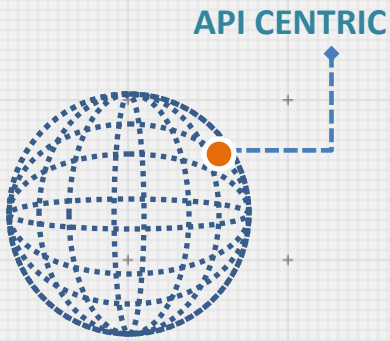
Разработка API-центричных веб-приложений с использованием технологий Javascript




```
{
"title": "Разработка API-центричных веб-приложений с использованием технологий Javascript ",
"keywords": [ "REST" , "Javascript", "RESTful" , "Framework" , "HTTP" , "JSON" , "XML" ],
"author": {
   "name": "Влад" ,
   "lastname": "Тертышный" }
}
```


Что мы рассмотрим?

Создание веб-приложений на основе API функционала

Что такое API?

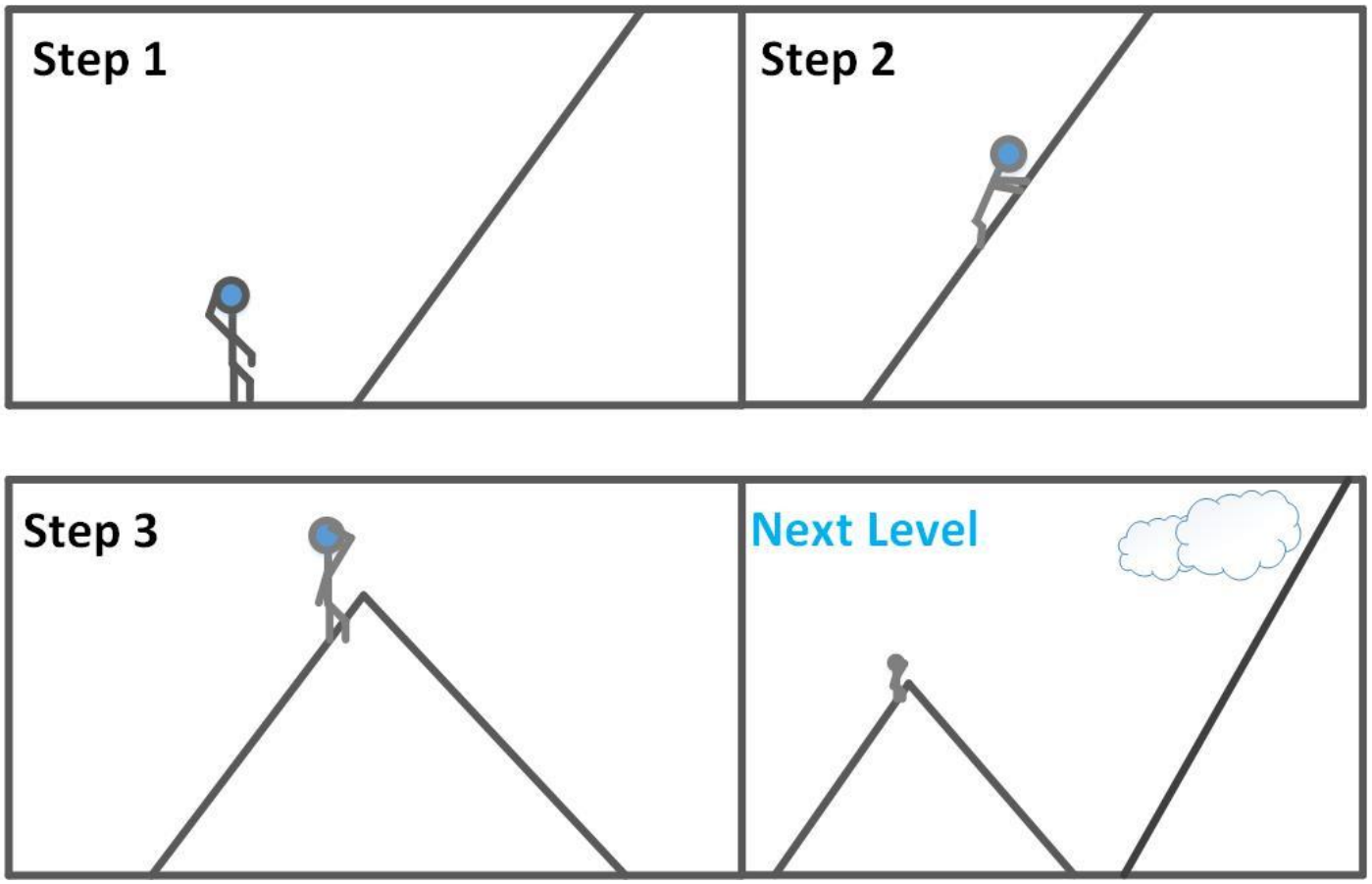
API *(application programming interface)* - набор готовых классов, процедур, функций, структур данных и констант, предоставляемых приложением (библиотекой, сервисом) для использования во **внешних программных** продуктах.

Какая цель?

Исследование методов разработки веб-приложений.
Осваивание базовых навыков веб разработки.

return report.slide(3);

Путь развития



return report.slide(4);

На чем создать API Centric Application?

REST

Что такое REST?

REST (сокр. англ. Representational State Transfer, «передача состояния представления» или «передача репрезентативного состояния») — стиль построения архитектуры распределенного приложения. Был описан и популяризован в 2000 году Роем Филдингом (Roy Fielding), одним из создателей протокола HTTP.

Три кита REST

JSON XML HTML

Базовые навыки и знания

http-протокол

Методы:

- post
- get
- put
- delete

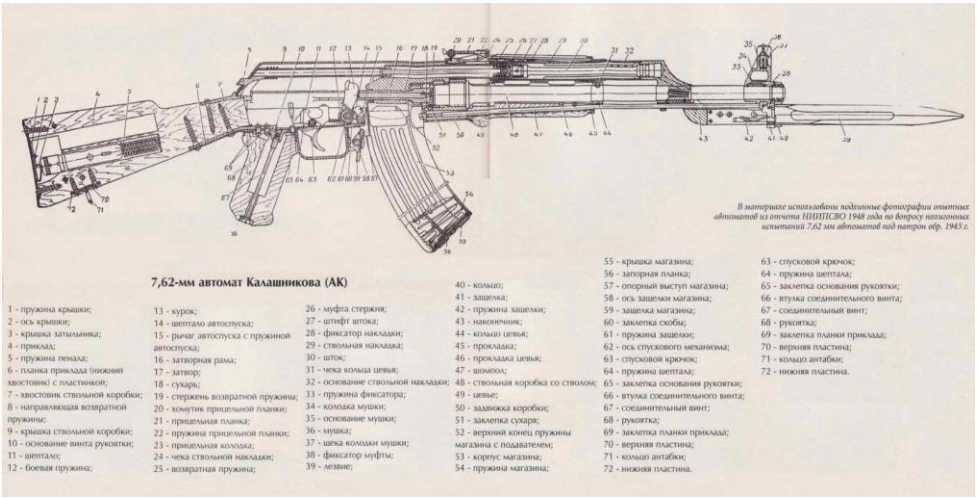
Коды состояния сервера

- 201
- 202
- 423
- 502
- и т. д.

418 I'm a teapot

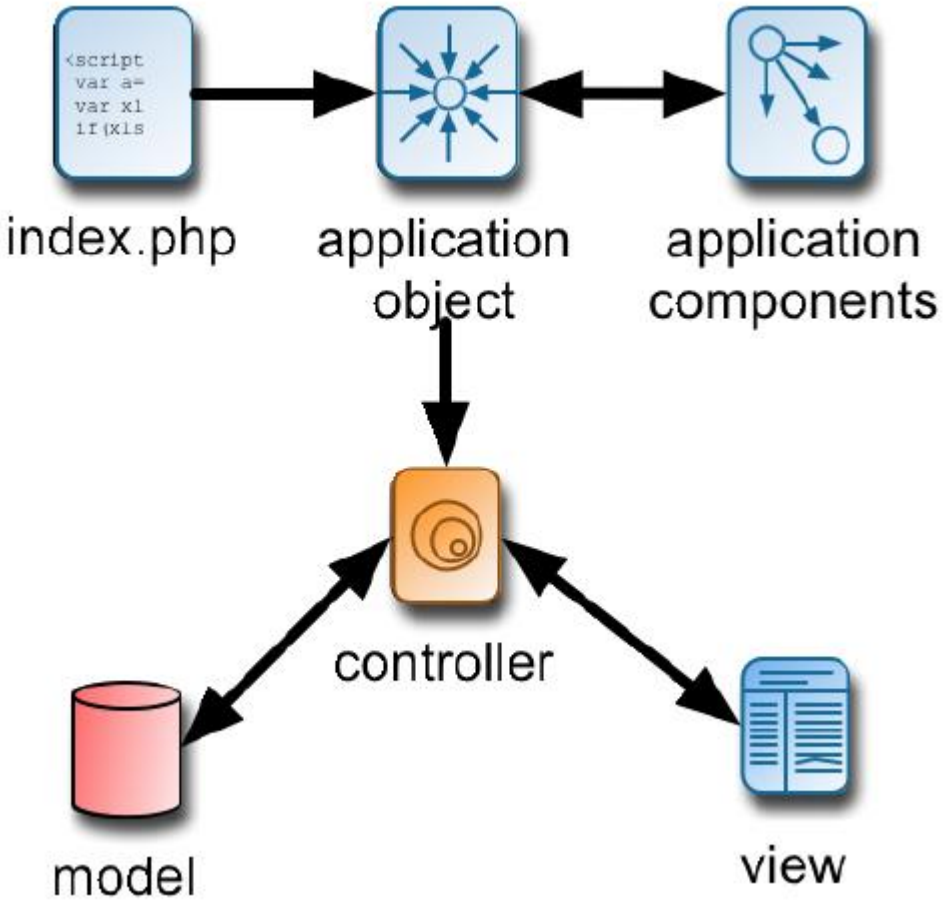
Почему REST API?

- Все является ресурсами с уникальным идентификатором (URL)
- Все операции клиента с сервером stateless, т.е. сервер не должен хранить вообще никакой информации о клиенте – никакой сессии
- Все запросы можно поделить на 4 типа в соответствии с CRUD, причем каждому типу сопоставляется HTTP метод – Post, Get, Put и Delete
- Вся логика крутится вокруг ресурсов, а не операций



return report.slide(7);

MVC



"The Yii Book" by Larry Ullman, 2013

return report.slide(8);

Создание RESTful сервера

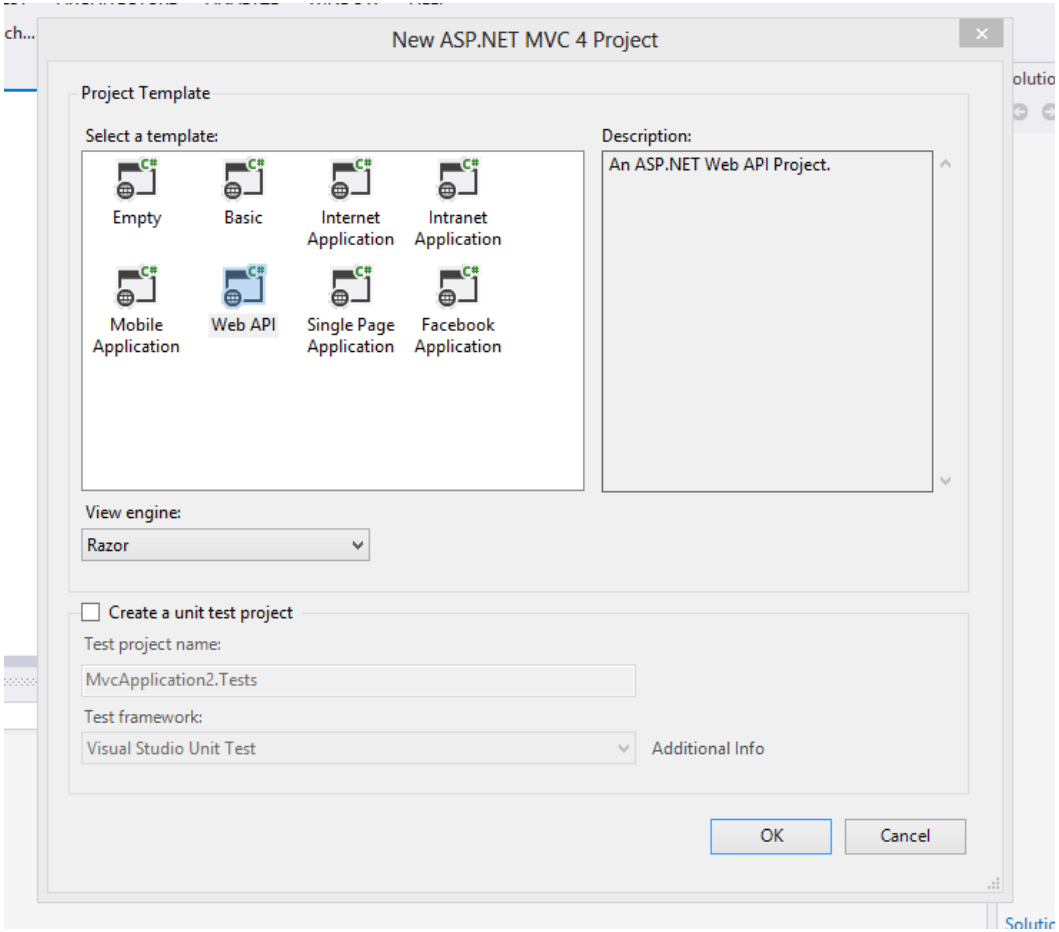
API сервер (back end)

ASP.NET
ASP.NET MVC
PHP
Node.js
Java EE, Jersey

Клиентская часть (front end)

Ajax
jQuery
Angularjs
Backbone.js
Bootstrap

ASP.NET MVC 4



return report.slide(10);

ASP.NET MVC 4

```
...
[Route("/users/{userid}", "POST")]
    public class AddRate
    {
        public long UserId { get; set; }
        public int Amount { get; set; }
    }

    public class UsersResponse
    {
        public IEnumerable<User> Users { get; set; }
    }

    [Route("/users", "GET")]
    public class Users
    {
    }
...
```

yiiframework

```
...
'urlManager'=>array(
    'urlFormat'=>'path',
    'rules'=>array(
        'post/<id:\d+>/<title:.*?>'=>'post/view',
        'posts/<tag:.*?>'=>'post/index',
        // REST patterns
        array('api/list', 'pattern'=>'api/<model:\w+>', 'verb'=>'GET'),
        array('api/view', 'pattern'=>'api/<model:\w+>/<id:\d+>', 'verb'=>'GET'),
        array('api/update', 'pattern'=>'api/<model:\w+>/<id:\d+>', 'verb'=>'PUT'),
        array('api/delete', 'pattern'=>'api/<model:\w+>/<id:\d+>', 'verb'=>'DELETE'),
        array('api/create', 'pattern'=>'api/<model:\w+>', 'verb'=>'POST'),
        // Other controllers
        '<controller:\w+>/<action:\w+>'=>'<controller>/<action>',
    ),
),
...
```

return report.slide(12);

Rating competition

Method	Pattern	Action
GET	/	Главная
GET	api/customers	Список всех игроков
GET	api/customers/12	Рейтинг игрока
GET	api/customers/chart/12	График рейтинга игрока
POST	api/customers	Добавление игрока
POST	api/customers/12	Добавление данных в рейтинг игрока
PUT	api/customers/12	Обновление данных рейтинга игрока
DELETE	api/customers/12	Удаление игрока

Реализация. Выбор инструментов.

phalcon

Bootstrap

jqPlot
pure javascript plotting

ANGULARJS
by Google

jQuery
write less, do more.

return report.slide(14);

```
//Переход на главну страницу
        $app->get('/', function() {
            echo 'Главная страница';
        });
// Получение списка всех пользователей
        $app->get('/api/users', function() {
            echo 'Список пользователей';
        });

// Поиск пользователей с $name в названии
        $app->get('/api/users/search/{name}', function() {
          echo $name;
        });

// Добавление нового пользователя
        $app->post('/api/users', function() {
            echo 'Post!';
        });
```

```
// Получение пользователя по указанному ключу
        $app->get('/api/users/{id:[0-9]+}', function($id) {
            echo "Передача id с помощью get: " . $id;
        });

// Обновление пользователя по ключу
        $app->put('/api/users/{id:[0-9]+}', function($id) {
            echo "Передача id с помощью PUT: " . $id;
        });


// Удаление пользователя по ключу
    $app->delete('/api/users/{id:[0-9]+}', function($id) {
        echo "Передача id с помощью <strong>DELETE</strong>: ". $id;
    });

    $app->handle();
```

return report.slide(16);

Demo

https://bitbucket.org/vtertyshnyj/microrest/

Atlassian Bitbucket

return report.slide(17);

Ajax + Micro App + REST API = **SPA**

```
function say_hello($name) {
    echo "<h1>Hello! $name</h1>";
}

$app->get('/say/hello/{name}', "say_hello");

$app->get('/say/hello/{name}', "SomeClass::someSayMethod");

$myController = new MyController();
$app->get('/say/hello/{name}', array($myController, "someAction"));

$app->get('/say/hello/{name}', function ($name) {
    echo "<h1>Hello! $name</h1>";
});
```

[-] Request

Method PUT URL http://localhost/microapp/api/users/2 SEND

Body

Request Body

Main Users

Передача id с помощью PUT: 2

[-] Request

Method DELETE URL http://localhost/microapp/api/users/2 SEND

Body

Main Users

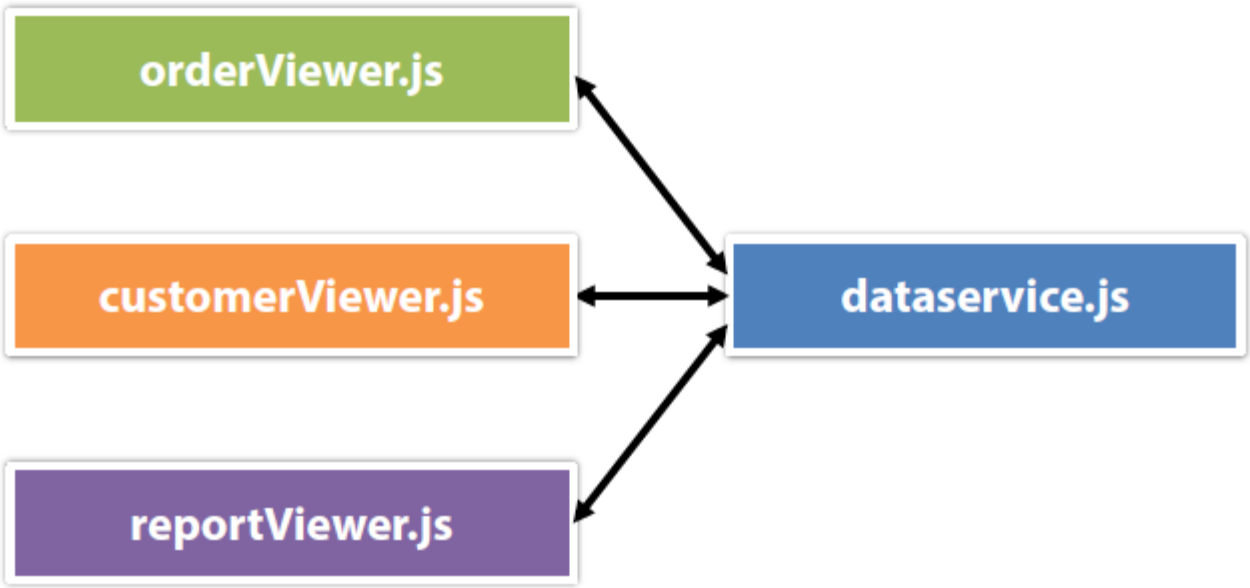
Передача id с помощью DELETE: 2

return report.slide(20);

jQuery

```
$("#CustomerButton").click(function() {
        $.getJSON("api/Customers",
        function(data) {
                varcust= data[0];
                $("#ID").text(cust.ID);
                $("#FirstName").val(cust.FirstName);
                $("#LastName").val(cust.LastName);
        });
});
```

return report.slide(21);



return report.slide(22);

```
var dataService = new function () {
    var serviceBase = '/api/dataService/',
    getCustomers = function() {
        return $.getJSON(serviceBase + 'customers');
    };

    return {
        getCustomers : getCustomers
    };
}();
```

Return promise

return report.slide(23);