REST

Д. Савченко

Все является ресурсом

- Приложение – набор ресурсов (сущностей)
- Каждый ресурс имеет свой URI
  - `http://music.site/users/max`
  - `http://music.site/albums/8`
- Коллекция ресурсов – также ресурс
  - `http://music.site/users`

Доступные операции

GET	POST	PUT	DELETE
Поиск	Создание	Изменение	Удаление
SELECT	INSERT	UPDATE	DELETE

Стандартные действия

- Неправильно
  - `POST /albums/create`
  - `GET /albums/show/2`
  - `POST /albums/update/2`
  - `POST /albums/destroy/2`

Стандартные действия

- Правильно
  - `POST /albums`
  - `GET /albums/2`
  - `PUT /albums/2`
  - `DELETE /albums/2`

Основные положения

- **Statelessness**
  - Состояние клиента – только на клиенте
  - Вся информация для обработки запроса – в запросе
- **Кешируемая архитектура**
  - Ответ сервера может быть кеширован и использован повторно без новых обращений
- **Клиент-серверное разделение**
  - Клиент не знает ничего лишнего о сервере

Ответы сервера

- Ответы сервера имеют HTTP-коды, показывающие статус операции
  - 200 – OK
  - 201 – Created
  - 202 – Accepted
  - 400 – Bad request
  - 403 – Forbidden
  - 404 – Not found
  - 500 – Server error
- WebDAV

GET

- Метод идемпотентен
- Получение сущности или списка сущностей
  - `GET /albums/`
  - `GET /albums/2`

POST

- Создание сущности
  - POST /albums

```
{
  artist: ‘Dimmu Borgir’
  , name: ’51k’
}
```

PUT

- Метод идемпотентен
- Изменение сущности
  - PUT /albums/2

```
{
  name: '52k'
}
```

DELETE

- Метод идемпотентен
- Удаление сущности
  - `DELETE /albums/2`

Twitter REST API v1.1

- Tweets
  - `GET statuses/retweets/:id`

  Возращает до 100 ретвитов твита id
  - `GET statuses/show/:id`

  Возвращает единственный твит id
  - `GET statuses/destroy/:id`

  Уничтожает твит
  - `GET statuses/update`

  Обновляет статус пользователя (создает новый твит)

Amazon S3

- Операции с объектами
  - `GET BucketName.s3.amazonaws.com/ObjectName`

  Получить объект как файл
  - `DELETE BucketName.s3.amazonaws.com/ObjectName`

  Удалить объект
  - `PUT BucketName.s3.amazonaws.com/ObjectName`

  Изменить объект
  - `POST BucketName.s3.amazonaws.com/ObjectName`

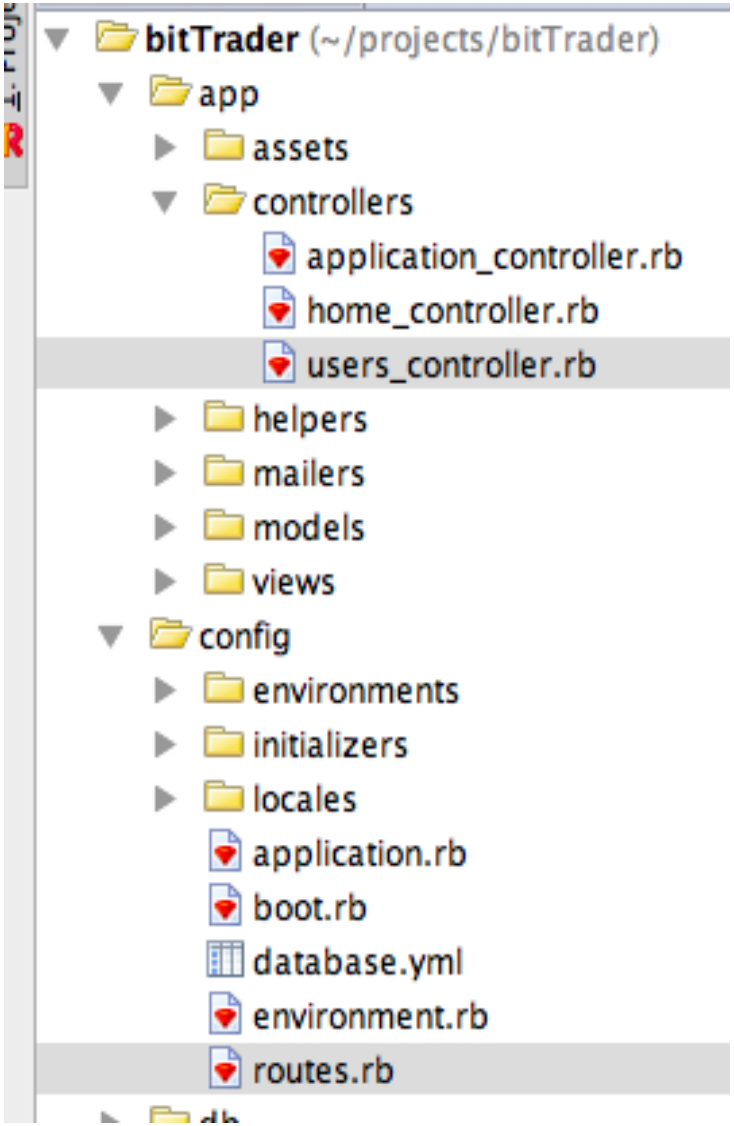
  Добавить объект

Работка собственного RESTful-сервиса

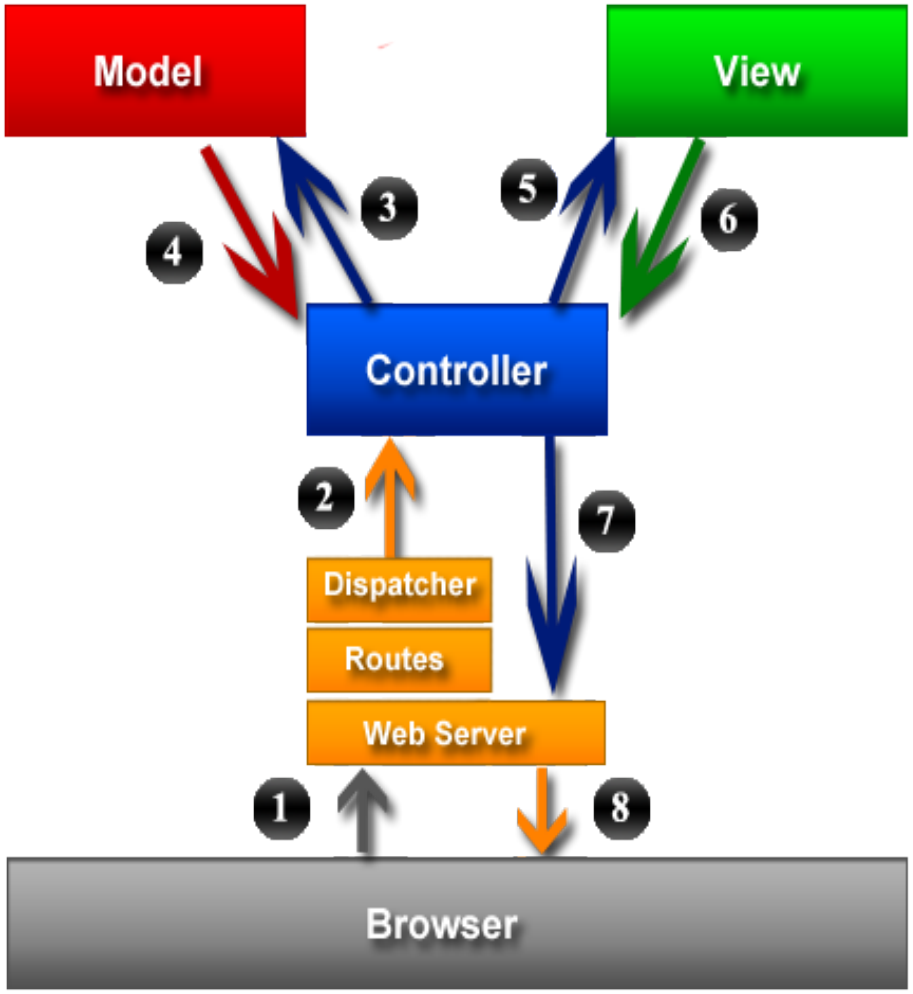
Сервер

- Основная технология – Ruby on Rails
  - Имеет эталонную реализацию ресурсной модели
  - Прост в освоении и понимании
  - Имеет очень большое комьюнити в США и Канаде

Структура проекта



Как происходит запрос



routes.rb

```
resources :users do # Пользователи..
# /users
  resources :notes do # Их записи..
  # /users/1/notes
    resources :comments # И комменты
    # /users/1/notes/2/comments
  end
end
```

Структура контроллера RoR

```
class UsersController < ActionController::Base
  # GET /users
  def index
    @users = User.all
    render :json => @users
  end
  # POST /users
  def create
    @user = User.new(params[:user])
    if @user.save
      render :status => :created, :json => @user
    else
      render :status => :unprocessable_entity
                        , :json => @user.errors
    end
  end
```

Пример контроллера RoR

```
  # GET /users/1
  def show
    @user = User.find(params[:id])
    render :json => @user
  end
  # PUT /users/1
  def update
    @user = User.find(params[:id])
    if @user.update_attributes(params[:user])
      head :no_content
    else
      render :json => @user, :status
=> :unprocessable_entity
    end
  end
```

Пример контроллера RoR

```
  # DELETE /users/1
  def destroy
    @user = User.find(params[:id])
    @user.destroy
  end
end
```

Пример сервиса на Java

```
@Path("/stores")
public class StoreService {

@GET
@Produces("application/xml")
public JAXBElement <Stores> getStoresAsXML()     {
   Stores stores = Stores.getStores();
   return new JAXBElement <Stores>
      ( new Qname("Stores"), Stores.class, stores);
}

@Path("/{id}")
@GET
@Produces("application/xml")
public Store getStoreAsXML(@ PathParam("id") String id) {
   // implementation here
}
```

Daigneau, Robert. Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Pearson Education. Kindle Edition.

Пример сервиса на Java

```
@POST
@Consumes("application/xml")
@Produces("application/xml")
public Store createStore(JAXBElement <Store> store)
{
    // implementation here
}

@Path("/{id}")
@PUT
@Produces("application/xml")
public Store updateStore(@PathParam("id") String id)
{
    // implementation here
}

}
```

Daigneau, Robert. Service Design Patterns: Fundamental Design Solutions for SOAP/WSDL and RESTful Web Services. Pearson Education. Kindle Edition.

Клиент

- Методы сервера можно вызывать при помощи jQuery:

```
$.ajax({
   url: '/users/1'
   , method: 'PUT'
   , data: {
     name: 'sk_'
   }
});
```

Backbone.JS

- Использует RESTful API для создания клиентского MVC и абстрагирует программиста от прямых вызовов HTTP
- Имеет два вида моделей:
  - Модель – единичная сущность ресурса
  - Коллекция – список сущностей

Модель

```
var User = (Backbone.Model.extend({
    defaults: {
      id: 0
      , name: ''
    }
    , urlRoot: '/users'
}));
```

Коллекция

```
var Users =
 new(Backbone.Collection.extend({
    model: User
    , url: ‘/users’
}));
```

Использование

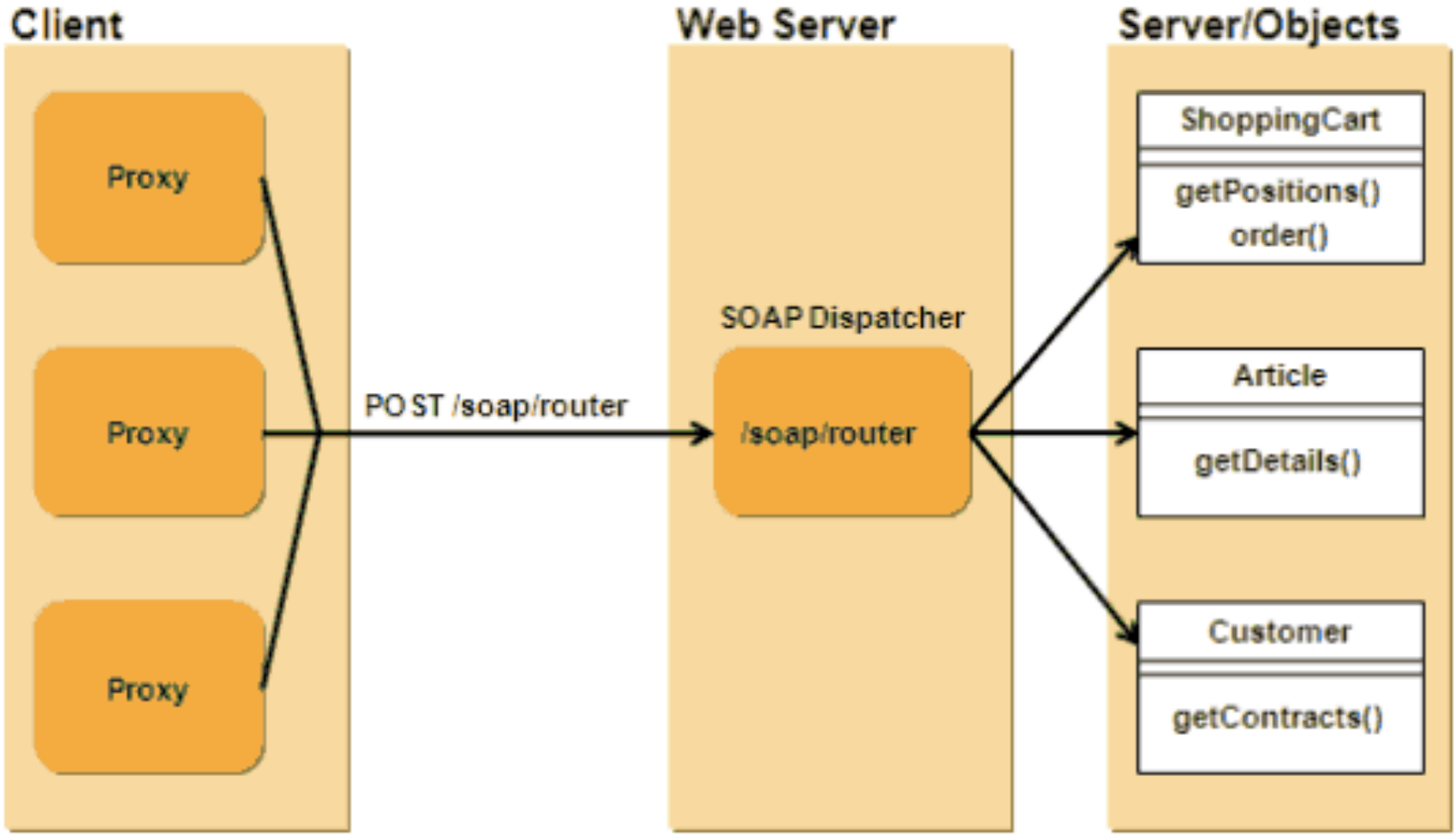
```
var model = new User();
model.set({ name: 'sk_' });
model.save();

var users = new Users();
users.bind("reset", renderUsers)
users.fetch();
```

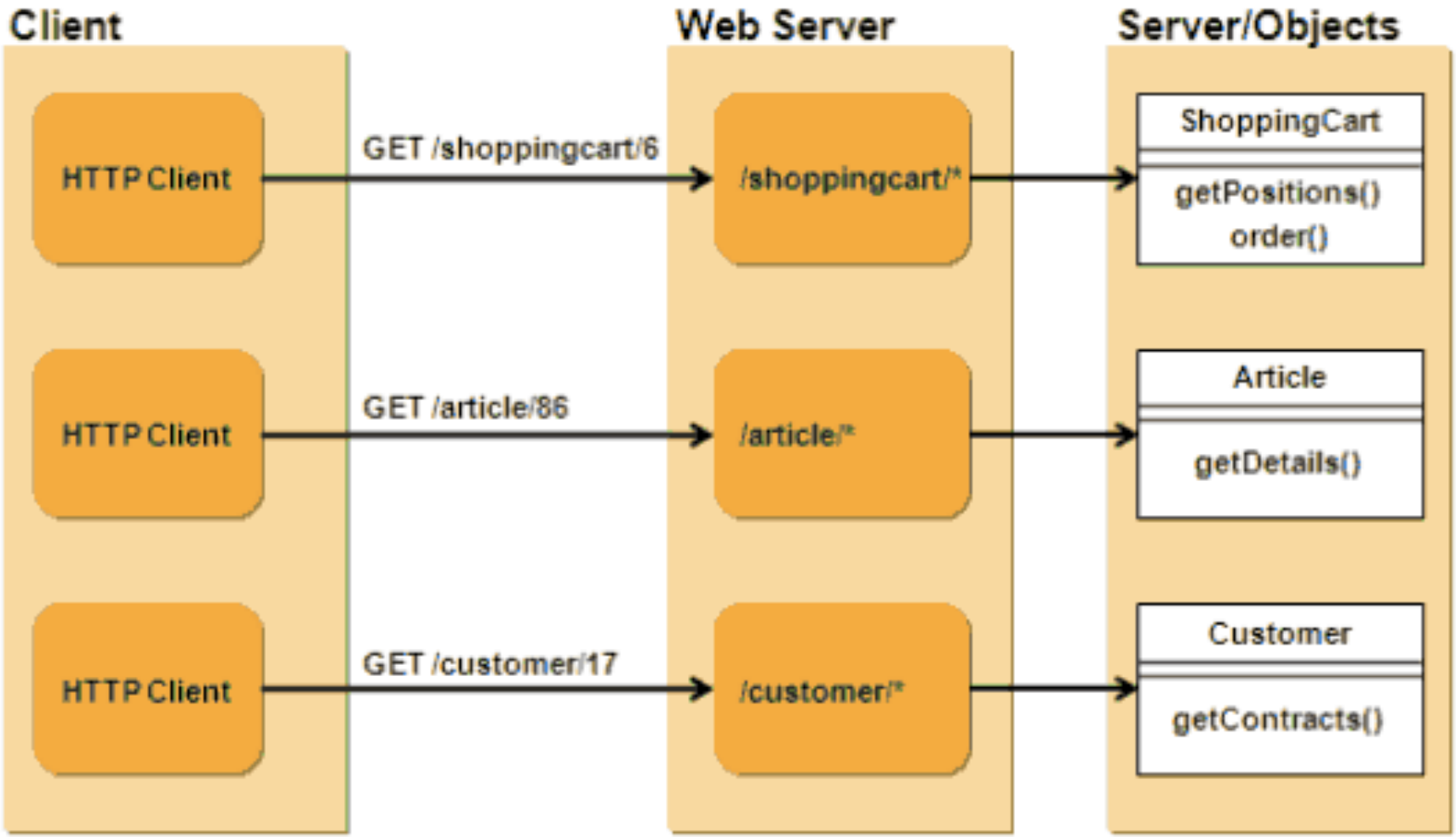
REST vs SOAP

- SOAP-сервисы имеют описание (WSDL), позволяющее сгенерировать клиента
- SOAP не допускает кеширования запросов
- SOAP работает только с POST-запросами
- Применение
  - SOAP – бизнес-приложения, инфраструктура распределенной системы
  - REST – внешний интерфейс системы

Запросы в SOAP



Запросы в REST



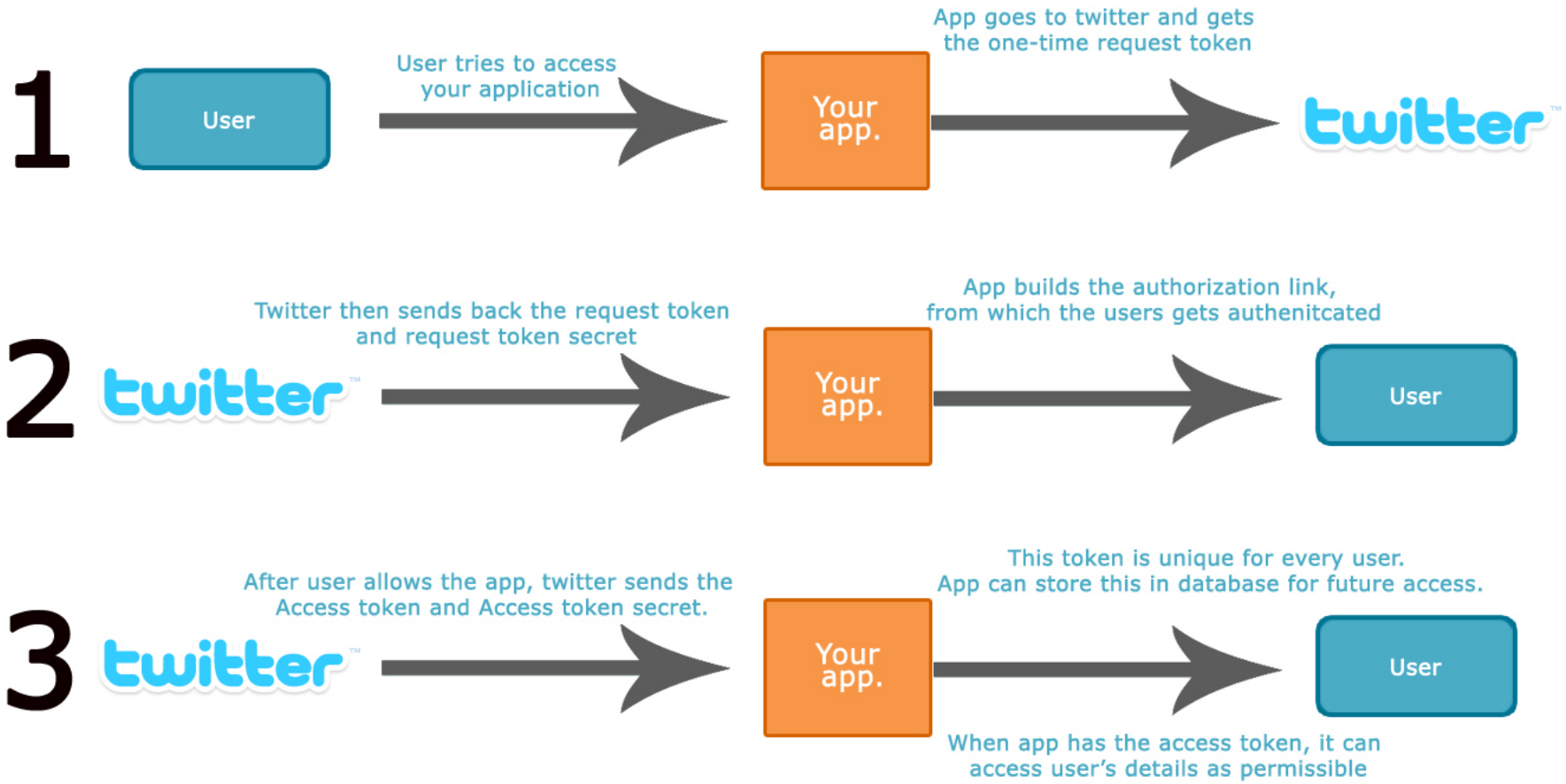
Безопасность в REST

- REST-сервис обычно публично доступен
  - Защита просто необходима!
  - Statelessness облегчает балансировку
  - Для аутентификации используется уникальный токен пользователя
  - HTTPS
  - …
  - PROFIT!!!

Токен

- Клиент получает все свои данные (и токен) во время логина
- Токен однозначно идентифицирует пользователя
- Токен прикладывается к каждому авторизованному обращению (в качестве параметра или в HTTP-заголовке Authorization)

OAuth


1
User
User tries to access your application
Your app.
App goes to twitter and gets the one-time request token
twitter
2
twitter
Twitter then sends back the request token and request token secret
Your app.
App builds the authorization link, from which the users gets authenitcated
User
3
twitter
After user allows the app, twitter sends the Access token and Access token secret.
Your app.
This token is unique for every user. App can store this in database for future access.
User
When app has the access token, it can access user's details as permissible

Что стоит почитать

- http://habrahabr.ru/post/144011/ - хороший, годный гайдлайн
- http://guides.rubyonrails.org/getting_started.html - введение в RoR
- http://techart.ru/files/university/doc-34-1245056300.pdf - глубокий обзор REST
- http://adrianmejia.com/blog/2012/09/11/backbone-dot-js-for-absolute-beginners-getting-started/ - введение в Backbone.JS