# ЛАБОРАТОРНАЯ РАБОТА №9 АВТОРИЗАЦИЯ ПО ПРОТОКОЛУ OAUTH (SPRING BOOT)

/\*

Описать бизнес-процесс регистрации приложения и реализации протокола OAuth от поставщиков: Google, Github, Facebook

\*/

**ОГЛАВЛЕНИЕ**

[ЛАБОРАТОРНАЯ РАБОТА №9 АВТОРИЗАЦИЯ ПО ПРОТОКОЛУ OAUTH (SPRING BOOT) 1](#_Toc124399021)

[Общие сведения 2](#_Toc124399022)

[Google 3](#_Toc124399023)

[GitHub 8](#_Toc124399024)

[Facebook 9](#_Toc124399025)

## Общие сведения

*OAuth 2* представляет собой фреймворк для авторизации, позволяющий приложениям осуществлять ограниченный доступ к пользовательским аккаунтам на HTTP сервисах, например, на Facebook, GitHub и DigitalOcean. Он работает по принципу делегирования аутентификации пользователя сервису, на котором находится аккаунт пользователя, позволяя стороннему приложению получать доступ к аккаунту пользователя. OAuth 2 работает в вебе, на десктопных и мобильных приложениях.

OAuth определяет *следующие роли*:

*- Владелец ресурса:* Пользователь

Владельцем ресурса является пользователь, который авторизует приложение для доступа к своему аккаунту. Доступ приложения к пользовательскому аккаунту ограничен «областью видимости» (scope) предоставленных прав авторизации (например, доступ на чтение или запись).

*- Сервер ресурсов и сервер авторизации:* API

Сервер ресурсов непосредственно хранит защищённые данные аккаунтов пользователей, а авторизационный сервер проверяет подлинность информации, предоставленной пользователем, а затем создаёт авторизационные токены для приложения, с помощью которых приложение будет осуществлять доступ к пользовательским данным. С точки зрения разработчика приложения API сервиса одновременно выполняет и роль сервера ресурсов и роль сервера авторизации.

*- Клиент:* Приложение

Клиентом является приложение, которое хочет осуществить доступ к аккаунту пользователя. Перед осуществлением доступа приложение должно быть авторизовано пользователем, а авторизация должна быть одобрена со стороны API.

На [рисунке 1](#_Hlk124392401) представлено абстрактное описание протокола, в котором указаны следующие шаги:

1. Приложение запрашивает у пользователя авторизацию на доступ к серверу ресурсов.

2. Если пользователь авторизует запрос, приложение получает разрешение на авторизацию (authorization grant).

3. Приложение запрашивает авторизационный токен у сервера авторизации (API) путём предоставления информации о самом себе и разрешении на авторизацию от пользователя.

4. Если подлинность приложения подтверждена и разрешение на авторизацию действительно, сервер авторизации (API) создаёт токен доступа для приложения. Процесс авторизации завершён.

5. Приложение запрашивает ресурс у сервера ресурсов (API), предоставляя при этом токен доступа для аутентификации.

6. Если токен действителен, сервер ресурсов (API) предоставляет запрашиваемый ресурс приложению.



Рисунок 1

## Google

Чтобы включить аутентификацию Google используя Spring Boot, нужен идентификатор клиента и общий секрет (client identifier и shared secret), который необходимо передать в качестве параметра конфигурации в приложение. Ниже представлена пошаговая инструкция.

1) Войдите в консоль API Google, используя свою учетную запись Google.

2) Включите службы Gmail, Analytics и Google+ API в облачной консоли Google ([рисунок 2](#_Hlk124392449)).

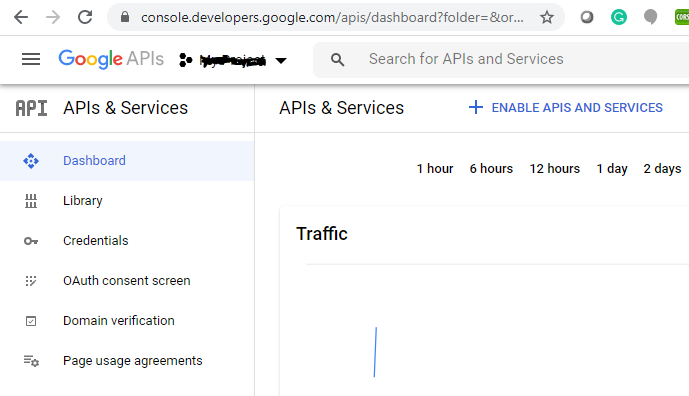


Рисунок 2

3) После включения вышеуказанных служб перейдите в раздел учетных данных (credentials section) в левом меню и выберите параметр идентификатора клиента OAuth (OAuth client ID) ([рисунок 3](#_Hlk124392701)).

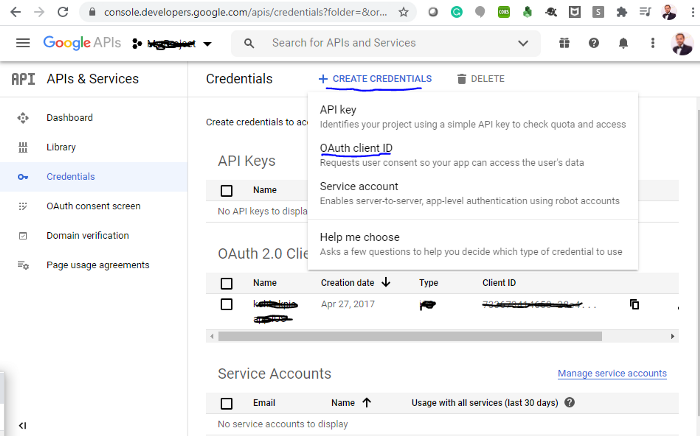


Рисунок 3

4) Если вы впервые обращаетесь к этому сервису, вам необходимо создать проект.

5) После выбора идентификатора клиента OAuth затем выберите тип приложения, в примере – «Веб-приложение», которое предоставит авторизованные источники javascript (javascript origins) и форму авторизованных redirect URI (это URL, на который будет переадресован браузер пользователя после разрешения им прав доступа при получении ключа доступа) ([рисунок 4](#_Hlk124393337)).

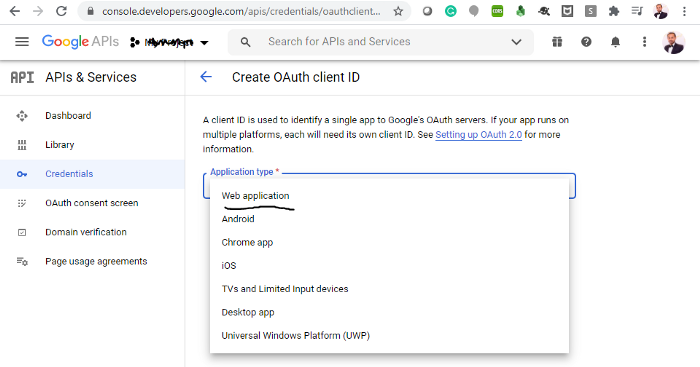


Рисунок 4

6) Продолжая пример с веб-приложением, после выбора типа приложения давайте укажем имя приложения, авторизованные источники javascript (javascript origins), авторизованные redirect URI и выберем вариант сохранить ([рисунок 5](#_Hlk124393547)).

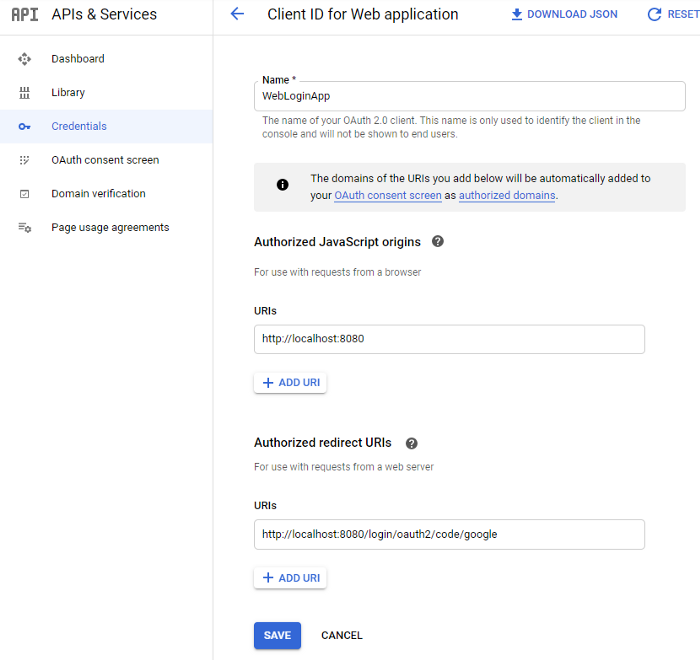


Рисунок 5

7) Наконец, вы найдете идентификатор клиента и секрет клиента в разделе, созданном клиентом OAuth. Скопируйте идентификатор клиента и секрет клиента для последующего использования при настройке веб-приложения ([рисунок 6](#_Hlk124393772)).

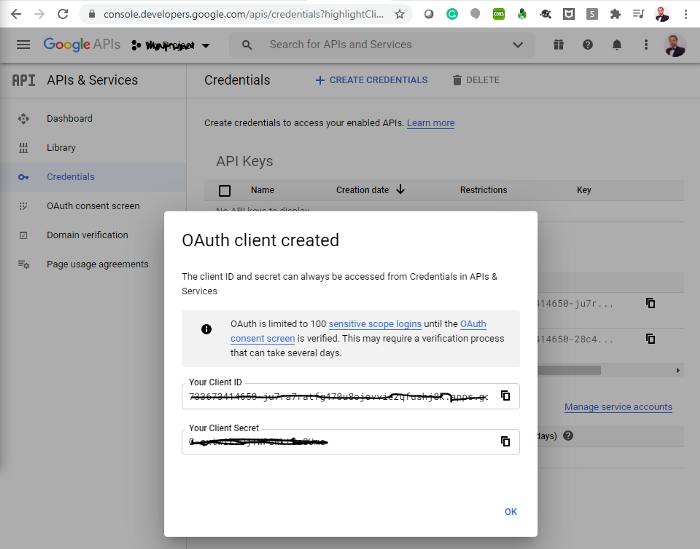


Рисунок 6

8) Теперь мы успешно создали клиент авторизации в облачной консоли Google ([рисунок 7](#_Hlk124393895)).

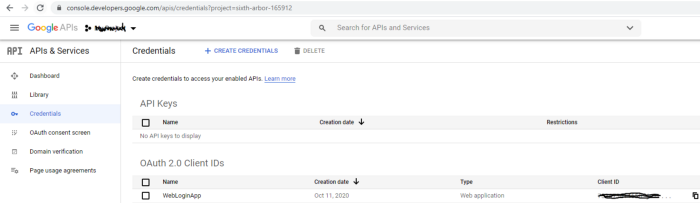


Рисунок 7

В веб-приложении для примера будут использоваться зависимости, представленные [рисунке 8](#_Hlk124394129).

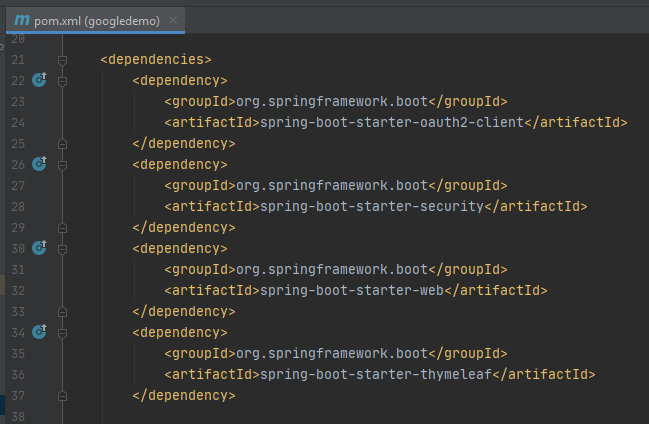


Рисунок 8

Ниже кратко представлено содержание приложения-примера.

[Рисунок 9](#_Hlk124394431) – домашняя страница index.html с использованием Thymeleaf (**src/main/resources/templates/index.html).**

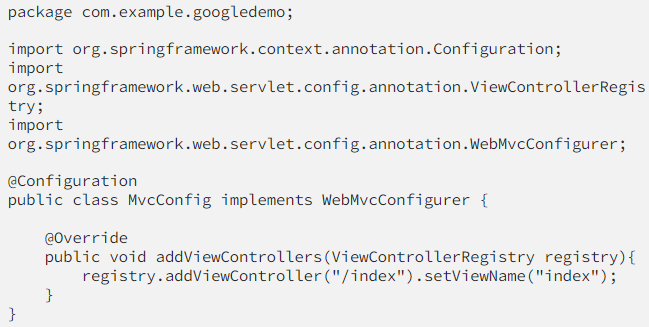


Рисунок 9

[Рисуок 10](#_Hlk124394734) – Конфигурация Spring MVC (src/main/java/com/example/ googledemo/MvcConfig.java).

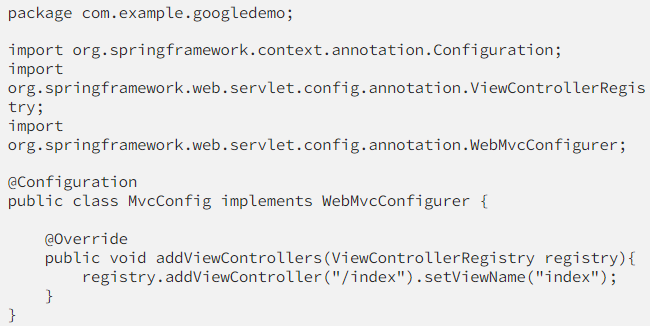


Рисунок 10

Если вы хотите защитить и ограничить индексную страницу от неавторизованных пользователей, доступ к индексной странице необходимо защитить, заставив пользователя перейти на страницу входа в Google – для этого можно использовать Spring Security (зависимость уже была внедрена). Создадим класс SecurityConfig.java и аннотируем его @EnableWebSecurity, чтобы включить поддержку веб-безопасности Spring для Spring MVC (если сейчас запустить приложение, то в браузере появится форма для регистрации по умолчанию). Теперь нужно переопределить методы безопасности spring по умолчанию, чтобы применить логику аутентификации приложения. Нужно расширить класс **WebSecurityConfigurerAdapter** и переопределить несколько его методов, чтобы установить некоторые пользовательские конфигурации доступа к безопасности. Метод configure(HttpSecurity)необходимо переопределить, чтобы определить, какие URL-адреса должны быть защищены, а какие нет.

Теперь нам нужно настроить сгенерированные учетные данные клиента в файле application.properties в папке ресурсов (resources folder). Префиксы Spring, связанные с безопасностью, начинаются с «spring.security.oauth2.client. registration» и следуют за пользовательскими свойствами, связанными с клиентом. Нужны:

- spring.security.oauth2.client.registration.google.client-id;

- spring.security.oauth2.client.registration.google.client-secret.

После добавления вышеуказанных свойств приложения для клиента Google это позволит включить все необходимые компоненты для входа в систему OAuth 2.

На [рисунке 11](#_Hlk124395957) представлен конечный код класса SecurityConfig.java.



Рисунок 11

Внесение общедоступной домашней страницы в белый список: мы можем внести в белый список любые представления из аутентификации, предоставив разрешение PermitAll(): «.antMatchers("/home").permitAll()».

Остается только собрать и запустить приложение.

*//Следующие примеры – без создания кода приложения.*

## GitHub

Чтобы использовать систему аутентификации GitHub OAuth 2.0, мы должны сначала создать новое приложение GitHub.

Для этого необходимо зайти в свою учетную запись и перейти по следующему пути: Settings->Developer Settings->OAuth Apps. В итоговом окне нажать кнопку «Register a new application» ([рисунок 12](#_Hlk124397212)).

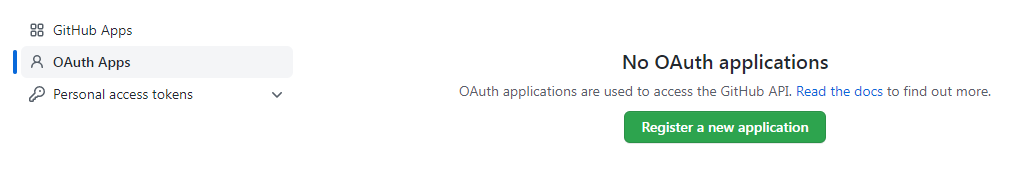


Рисунок 12

На [рисунке 13](#_Hlk124397529) представлен пример заполнения формы для создания приложения OAuth в GitHub.

Redirect URI (reply URL) — это URI, на который сервер авторизации (например, GitHub) отправляет код авторизации, который впоследствии обменивается на токен доступа (access token), который используется для **аутентификации** последующих вызовов API.

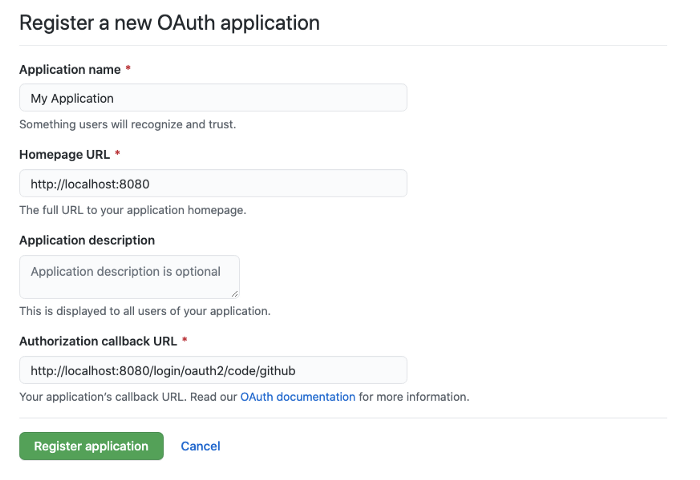


Рисунок 13

## Facebook

Необходимо произвести аналогичные действия:

- Перейти на https://developers.facebook.com/;

- Войти и создать новое приложение OAuth;

- Добавьте такие сведения, как имя приложения, URL-адрес и прочие необходимые данные;

- После успешного создания приложения в появившемся окне можно перейти к «**настройки -> основные»** и скопировать идентификатор клиента и секрет клиента ([рисунок 14](#_Hlk124398651)).

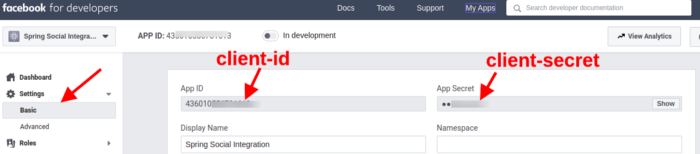


Рисунок 14