

Dököll Solutions, Inc.

Application Development

Project Name:	App Journals	Start Date:	2021.10.18.4.16.PM
Purpose:	Create Google Sheets v4 App Configuration Instructions	End Date:	2021.10.24.12.33.AM
Language:	Java, Script	Environment:	Google Cloud, Gradle NetBeans
Employee Name:	Dököll Solutions	Employee ID:	Dököll Solutions
Task(s):	Java App Configuration/Setup	Document:	Journal Entries

How-To Create a Google Sheets v4 Java Project

System Requirements

Software/Environment	Language/Technology	Protocol/Framework/Platform
Microsoft Windows 7, 8, 10	VBScript, Batch	Active Directory, WORKGROUP
Microsoft Edge	N/A	TCPIP, HTTP, Browser
Google Chrome	N/A	TCPIP, HTTP, Browser
Mozilla Firefox	N/A	TCPIP, HTTP, Browser
Google Drive Account	N/A	HTTP, Browser
Apache NetBeans 12.x.x	Java 1.8.xxx	HTTP, TCPIP, IDE
Apache Tomcat 8.5	Command, Batch	HTTP, TCPIP, Web Server
Gradle 7	Command, Batch	HTTP, TCPIP, Plugin

Disclaimer:

Information contained in the following is presented as is. This document assumes you have basic programming and software configuration knowledge. All Java samples are based on NetBeans and Gradle to connect to Google Sheets v4, using the Google Drive and Google Sheets APIs, see System Requirements for additional information. Should you need to familiarize yourself with Google Sheets v4, Google APIs, Apache NetBeans, and Gradle environments, prior to continuing, stop now and see our Journal Entries page on our website: www.dokollolutionsinc.com for additional support...

Foreword:

Samples included in this Journal Entries document are part of a series, be sure to consult other versions of the documentation to benefit in full.

Introduction:

Configure a Google Cloud Console environment to prepare for a Java application to read, write, and update Google Sheets v4 spreadsheets data located on Google Drive. This documentation assumes a Google account has already been setup and Google Drive is accessible. Developers should be ready to spend a good amount of time configuring their local coding environments. For example, an Integrated Development Environment (IDE) that runs Java should be installed on a computer or server box. Google API credentials are needed to authenticate to the Google Cloud infrastructure, they must be setup prior to creating an application on the Cloud Console, and prior to connecting the local developer IDE to the Cloud App. Once the IDE and access to the Cloud environment are properly configured, you are half way there; you are now ready to create a Google Sheets v4 project. Without further-a-do, let us begin... The specifics of the project and all areas of interest will be highlighted in this document as guide.

Journal Entry

2021.10.18.4.16.PM

The following steps about creating a Google Cloud Platform project was copied directly from Google to help you begin your journey. We will add to the instructions as we see fit so that you understand it the way we understand it.

Step 1: Create a Google Cloud Project

To use Google Workspace APIs, you need a Google Cloud Platform project. This project forms the basis for creating, enabling, and using all GCP services, including managing APIs, enabling billing, adding and removing collaborators, and managing permissions.

1. Open the [Google Cloud Console](#). (Google Account should be created prior)
2. Next to "Google Cloud Platform," click the Down arrow **arrow_drop_down**. A dialog listing current projects appears.
3. Click **New Project**. The New Project screen appears.
4. In the **Project Name** field, enter a descriptive name for your project. If you're executing a quickstart, use "Quickstart." (Example: Our existing project is called SocialMedia)
5. (Optional) To edit the **Project ID**, click **Edit**. The project ID can't be changed after the project is created, so choose an ID that meets your needs for the lifetime of the project.
6. Click **Organization** and select your organization.
7. In the **Location** field, click **Browse** to display potential locations for your project.
8. Click a location and click **Select**.

9. Click **Create**. The console navigates to the Dashboard page and your project is created within a few minutes.

Enable a Google Workspace API

1. Open the [Google Cloud Console](#).
2. Next to "Google Cloud Platform," click the Down arrow **arrow_drop_down** and select a project.
3. In the top-left corner, click Menu **menu** > **APIs & Services**
4. Click **Enable APIs and Services** The **Welcome to API Library** page appears.
5. In the search field, enter the name of the API you want to enable. For example, type "Google Drive API" to find the Google Drive API. If you are enabling an API for a quickstart, refer to the quickstart's Prerequisites section for the API to enable.
6. Click the API to enable. The API page appears.
7. Click **Enable**. The Overview page appears.
8. To enable an additional API, (i.e. Google Sheets v4), repeat steps 3 - 7.

Here is a list of requirements for this project. As we go through them, we will show you the ones that are necessary.

Manage APIs in the API Console

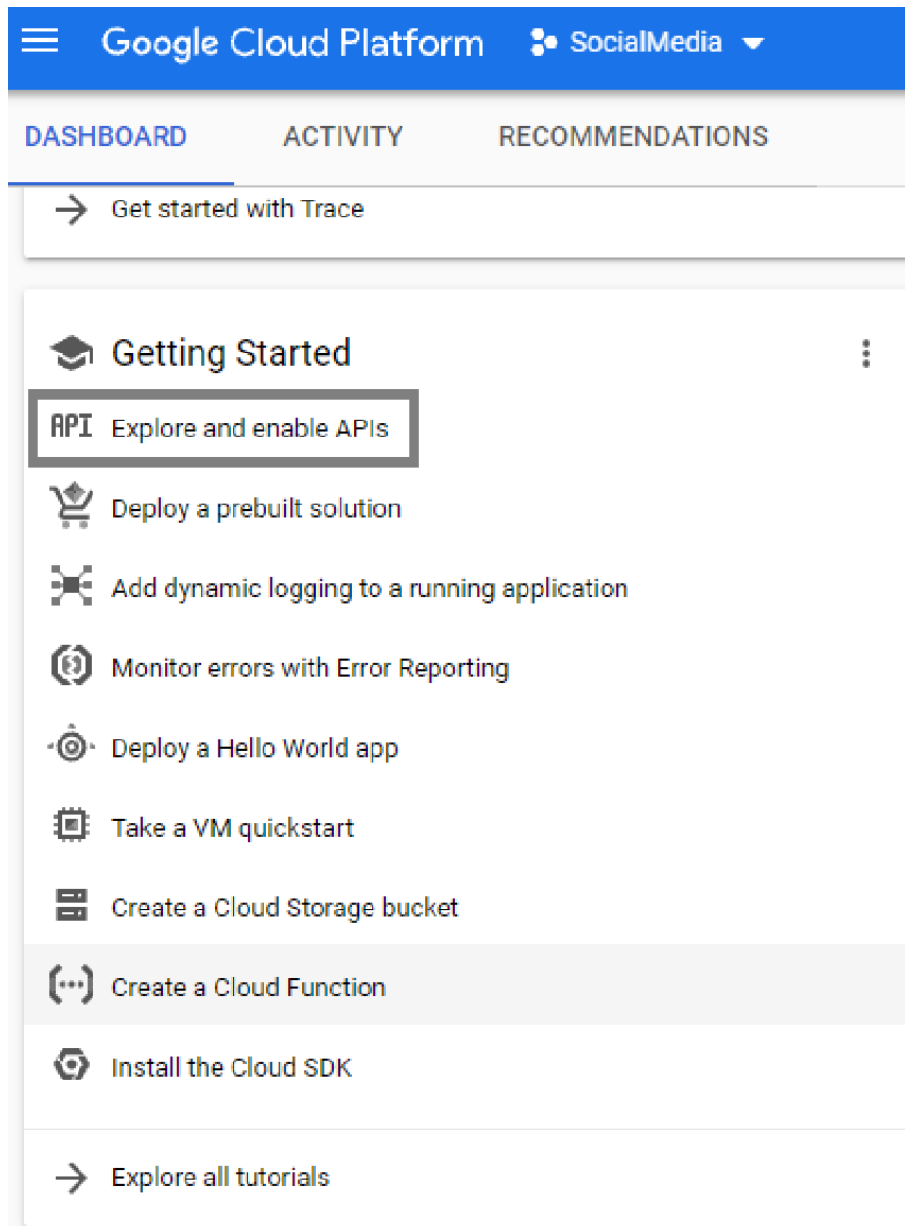
This is where you enable and disable APIs, manage and view traffic data, and set up authentication. The console is also where you manage billing for the Google APIs that you use.

- [Enable and disable APIs](#)
- [Credentials, access, security, and identity](#)
- [Setting up OAuth 2.0](#)
- [Setting up API keys](#)
- [Best practices for securely using API keys](#)
- [Monitoring APIs](#)
- [Capping API usage](#)
- [APIs and Billing](#)
- [Verifying domains for push notifications](#)

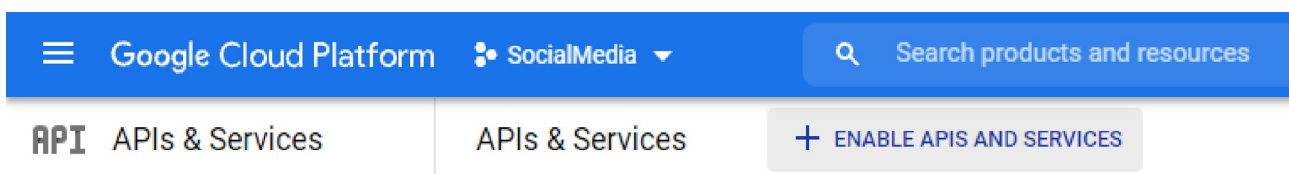
Hint: You cannot skip the following steps, you will not have access to the APIs to run your project... We shortened the above list for you. They are important but do not apply here since this is a developer sample meant to test out Google Sheets.

- [Enable and disable APIs](#)
- [Credentials, access, security, and identity](#)
- [Setting up OAuth 2.0](#)
- [Setting up API keys](#)
- [Verifying domains for push notifications](#)

Screenshots for Enabling APIs



Enable your APIs as seen below.... We have already done this, but we are going to show you which APIs you will need for your Java project-



You will need to enable both the Google Drive and Google Sheets APIs in this case....

Google Workspace

[VIEW ALL \(18\)](#)



Google Drive API

Google Enterprise API

The Google Drive API allows clients to access resources from Google Drive



Google Sheets API

Google Enterprise API

The Sheets API gives you full control over the content and appearance of your spreadsheet...

Step 2: Create Credentials for a Google Cloud Project

Once the right APIs are enabled, you can begin creating your Google Sheets Java Application Project... We will go through some of the steps with you. In **Step 1**, you created the project and you enabled both Google Drive and Google Sheets v4 APIs. Now, you will need to create an API key and a Client id in the credentials portion of the Google Cloud Console prior to writing code to read, write, and update spreadsheets data. Depending on your project or the type of authentication method you choose, you will get approval quickly from Google. Also, be aware that the scopes you choose also impact the wait time for you to get approved by Google to connect to their infrastructure. We will go through the types of scopes to choose later in this Journal Entries segment. Follow the steps below to create an API key for your project.

This information was copied directly from the Google Cloud Console page, and just as we have done in the first step above when creating our project, we will also modify the text to save you the effort of figuring out what certain sets of instructions mean. We will also show you what the setup should look like after the instructions.

Managing API keys

We recommend you use the Cloud Console to manage API keys. Navigate to the [APIs & services → Credentials](#) page in the Cloud Console. Your API keys are shown in the **API keys** section. On this page, you can create API keys, define API key restrictions, rotate API key strings, and take other actions.

Creating an API key

To create an API key in a project, the user must be granted the **Editor** basic role (**roles/editor**) on the project. See [basic roles](#) for more information.

To create an API key:

1. Navigate to the **APIs & Services**→**Credentials** panel in Cloud Console.
2. Select **Create credentials**, then select **API key** from the dropdown menu.

The **API key created** dialog box displays your newly created key.

An API key is a long string containing upper and lower case letters, numbers, and dashes, such as **a4db08b7-5729-4ba9-8c08-f2df493465a1**. In this case, Google will simply create the key for you. You can pretty much just accept all defaults.

You should copy your key and keep it secure. Unless you are using a testing key that you intend to delete later, add [application and API key restrictions](#).

Next step is a very crucial one, you will need an OAuth 2.0 Client ID for your project, this will allow you to download the credentials file (JSON format) to plug into your Java application to gain access to your Google Cloud Project from your IDE.

Create the OAuth Web Client ID

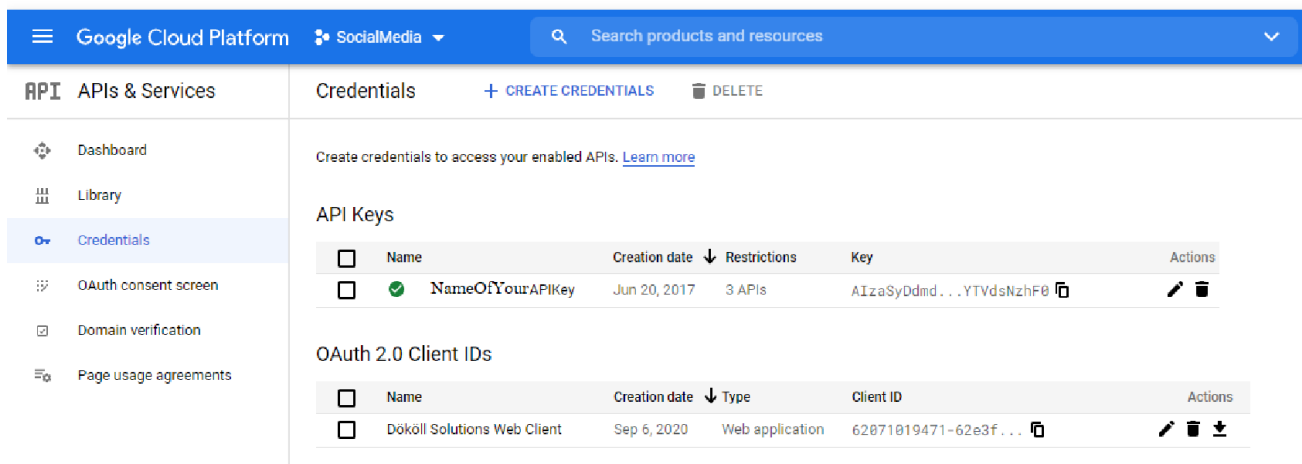
In the Google Cloud Console, create the OAuth web client ID for the Google Workspace Migrate platform.

1. In the Cloud Console, open the project you created earlier.
Note: If you manually created a service account, the project was created in Step 1: Use the Google Cloud Console to turn on APIs. If you created the project using a script, the project name is listed in the Cloud Shell Editor after the script runs.
2. Click **APIs & Services** > **OAuth consent screen**.
3. Set **User Type** to **Internal** and click **Create**.
4. In the **Application name** field, enter a name and click **Save**.
5. In the left menu, click **Credentials**.
6. Click **Create credentials** > **OAuth client ID**.
7. Select **Web application**.
8. In the **Name** field, enter a name for the OAuth web client.
9. In the **Authorized JavaScript origins** field, enter the URL that you'll use to access the Google Workspace Migrate platform (for example, <http://localhost:5131>).
10. Click **Create**.
11. Make a note of the client ID shown in the **Client ID** field. You'll need it when you set up the Google Workspace Migrate platform.
Tip: You can also access the client ID from **APIs & Service** > **Credentials**.
12. Click **OK**.

Hint: For Step 9 above, we have Apache Tomcat running, which loads in the browser using port 80,

thus our URL carries four digits 8080- this will make more sense when we attempt to connect...

Your credentials screen should look similarly to below. Obviously, for security purposes we cannot show you all of the data.



Food for Thought:

It is important to re-create your API key and Client ID to limit interaction with your Google Cloud credentials. You can do this right from the above Credentials page.

From here, you can click the copy or edit button at right in the API keys section to copy the full API key for your project. Please note, the API key name is NOT your Project name, and the API key name will not be used in the application. You will need to download the OAuth Client ID to your computer or server box. The file will be a JSON file containing some credentials data generated when you create the API key and Client ID. You will need to import that file into NetBeans later, so make sure that you know where the JSON is downloaded to, so you can retrieve it.

Google Cloud App Verification/Consent

Next step is get consent from Google to run code and reach their platform. This process helps verify your application can run when launched in your browser. Eventually, when you run your Java program locally, it will make an attempt to connect to Google. You will be prompted to sign in to your Google account to gain application/project access to your Google Sheets data.

Once you give your Google account Cloud App permission to run, it will send a one-time message to Google to verify and accept the connection from your local NetBeans App. There may be reasons why Google does not allow the connection to take place. This may be due to the way you filled out the consent form. This is where Project Scopes are important. We will show you how to get verification for your App now, then go into what scopes to choose for your App to run.

To submit for verification, follow the steps below:

1. Go to the Cloud Console OAuth consent screen page.
2. Click the **Project selector** drop-down at the top of the page.
3. On the **Select from** dialog that appears, select your project.

- If you can't find your project, and you know your project ID, you can construct a URL in your browser in the format `https://console.cloud.google.com/apis/credentials/consent?project=[PROJECT_ID]` where [PROJECT_ID] is the project ID you want to use.
4. Click the **Edit App** button.
 5. Enter the information required on the configuration page, and then click **Submit for verification**. If the submit for verification button does not appear at the end of the configuration pages, save what you have completed and repeat steps 1-4.
 6. Once you click Submit for Verification, a **Verification required** dialog box will appear, enter the appropriate justifications, and then click **Submit** to start the verification process.

Your Java application will have the scope hard-coded to run properly, but during your consent process, you will need to choose scope information that determines the security of the Cloud App- Below is a screenshot of the scopes to keep in mind when you are filling out your consent form.

Google Sheets API, v4

Scopes	
<code>https://www.googleapis.com/auth/drive</code>	See, edit, create, and delete all of your Google Drive files
<code>https://www.googleapis.com/auth/drive.file</code>	View and manage Google Drive files and folders that you have opened or created with this app
<code>https://www.googleapis.com/auth/drive.readonly</code>	See and download all your Google Drive files
<code>https://www.googleapis.com/auth/spreadsheets</code>	See, edit, create, and delete your spreadsheets in Google Drive
<code>https://www.googleapis.com/auth/spreadsheets.readonly</code>	View your Google Spreadsheets

Verification form sent, Now what?

At first glance, it may seem that verification is still needed for our application. But, our application is up and running and the message is still present. It is safe to say that if your App is not prompting for permission through your browser, you are good to go. You are likely not going to have an In Progress message in your trials. We wanted to post this screenshot anyway in case you see it, so you do not think you are out of luck or cannot continue. Truth is, you can... Keep reading to learn more...

The screenshot shows the Google Cloud Platform console interface. At the top, there is a blue navigation bar with the Google Cloud Platform logo, a 'SocialMedia' dropdown menu, and a search bar containing the text 'Search products and resources'. Below the navigation bar, the left sidebar is titled 'API APIs & Services' and contains a list of navigation items: Dashboard, Library, Credentials, OAuth consent screen (which is highlighted), Domain verification, and Page usage agreements. The main content area displays the configuration for the 'Dököll Solutions, Inc. Java Cloud' application. It shows the 'Verification Status' as 'Verification in progress' with an information icon. A message states: 'The Trust and Safety team has received your form. They will reach out to you via your contact email if needed. The review process can take up to 4-6 weeks. Expect the first email from our Trust and Safety team within 3-5 days. Your last approved consent screen is still in use. [Learn more](#)'. Below this, the 'Publishing status' is shown as 'In production' with an information icon.

Now, we are prepared to create our local application. All credentials are in place, we have downloaded the JSON file to reference in our project from the IDE. We are ready to take the next step to be able to use the Google Sheets v4 Quickstart sample.

Journal Entry

2021.10.18.6.51.PM

Step 3: Install NetBeans, Prepare for Google Sheets App

Let us first go through some instructions to install the Apache NetBeans IDE that we will be using to launch the Google Quickstart Java sample. For this document, we are using NetBeans, it comes with Gradle but you can download Gradle separately as a Standalone application to run the Google Sheets v4 sample. Once you've had a chance to go through this documentation, it will make more sense as to why we are choosing NetBeans as opposed to the Standalone Gradle software.

You should follow the instructions on this page very well. Google does a pretty good job providing the right information here to begin your Java project. It is quite obvious that you will need to do some of the leg work yourself. Since you will be starting a project in the NetBeans IDE as a Gradle project, there are certain configuration steps that must take place. In this Java Quickstart provided by Google, the version of Gradle being suggested is 2.3 or higher. You can use any Gradle version as long as it is at least 2.3. If you are comfortable with setting up your own IDE, just make sure that Gradle is either integrated or that you can download the plugin to set it up in your project. In our case, we are using **Apache NetBeans 12** because it is conveniently packaged with **Gradle 7**. We would recommend using NetBeans 12 since this is what we are using for this Journal Entries segment. Here is where you

can download that version:

<https://NetBeans.apache.org/download/nb120/nb120.html>

We are also providing a screenshot to show you the file we actually downloaded. Again, you can do whatever you want, as long as you have an IDE that runs Java and Gradle.

Downloading Apache NetBeans 12.0

Apache NetBeans 12.0 was released on June 4, 2020. See [Apache NetBeans 12.0 Features](#) for a full list of features.

Apache NetBeans 12.0 is available for download from your closest Apache mirror.

- Binaries: [netbeans-12.0-bin.zip](#) (SHA-512, PGP ASC)
- Source: [netbeans-12.0-source.zip](#) (SHA-512, PGP ASC)
- Installers:
 - [Apache-NetBeans-12.0-bin-windows-x64.exe](#) (SHA-512, PGP ASC)
 - [Apache-NetBeans-12.0-bin-linux-x64.sh](#) (SHA-512, PGP ASC)
 - [Apache-NetBeans-12.0-bin-macosx.dmg](#) (SHA-512, PGP ASC)
- Javadoc for this release is available at <https://bits.netbeans.org/12.0/javadoc>

The highlighted link corresponds to the following

<https://www.apache.org/dyn/closer.cgi/NetBeans/NetBeans/12.0/Apache-NetBeans-12.0-bin-windows-x64.exe>

which allows you to download/install from the following page

The screenshot shows the Apache NetBeans 12.0 download page. At the top left is the Apache Software Foundation logo. To the right is the slogan "COMMUNITY-LED DEVELOPMENT 'THE APACHE WAY'". Below this is a navigation bar with links for "Projects", "People", "Community", "License", and "Sponsors". The main content area starts with the text "We suggest the following mirror site for your download:" followed by a highlighted link: <https://dlcdn.apache.org/netbeans/netbeans/12.0/Apache-NetBeans-12.0-bin-windows-x64.exe>. Below this, it says "Other mirror sites are suggested below." and provides instructions on how to verify the integrity of the downloaded file using PGP signatures or hashes. At the bottom, under the heading "HTTP", the same highlighted link is repeated.

Journal Entry

2021.10.19.11.03.PM

At this point, we assume you installed NetBeans 12 from the URL posted above. You will be needing to copy the Quickstart Java sample from the following Google Sheets v4 webpage:
<https://developers.google.com/sheets/api/quickstart/java>

We want you to copy and paste the code into your environment without configuring the IDE to give you the ins and outs of working with the Google code; this will also help you get a better experience with this documentation. Right away, you will find that your code has a lot of errors. What we found out is even though we have all of the Google Sheets-specific imports in place, we still needed to do more configuration. Let us go through the steps now one by one...

Prepare to Use Google Sheets v4 code



As usual, we will post screenshots of the Google instructions and add our own, as we understand it. Again, you can use any IDE you wish to do this, as long as the Gradle plugin is available and you know how to configure your own environment. You can skip the Gradle portion in the screenshot below if you are using NetBeans. We did put some System Requirements atop this document, we assume you know which version of Gradle and Java being used in these Journal Entries...


Java Quickstart


Complete the steps described in the rest of this page to create a simple Java command-line application that makes requests to the Google Sheets API.

Prerequisites

To run this quickstart, you need the following prerequisites:

- Java 1.8 or greater.
- [Gradle 2.3 or greater](#) .
- A Google Cloud Platform project with the API enabled. To create a project and enable an API, refer to [Create a project and enable the API](#) .

 **Note:** For this quickstart, you are enabling the "Google Sheets API".

- Authorization credentials for a desktop application. To learn how to create credentials for a desktop application, refer to [Create credentials](#) .
- A Google account.

We have already taken care of creating a project, enabling APIs, and creating credentials in Steps 1 and 2 above. If you were following along, you are ahead of the game...

You can skip below as well since we are using NetBeans 12, which comes with Gradle. And we hope you did download the JSON file after creating your credentials. Be prepared to include the full file

name in your NetBeans/Gradle project.

Prepare the project

To prepare the project:

1. In your working directory, run the following commands to create a new project structure:

```
gradle init --type basic
mkdir -p src/main/java src/main/resources
```

2. Copy the `credentials.json` file you downloaded as a [prerequisite](#) into the newly-created `src/main/resources/` directory.
3. Open the default `build.gradle` file and replace its contents with the following code:

Technical Notes:

In the screenshot above, you do need to pay attention to the folder structure stated in **Step 2**. Since we are using NetBeans, and this may be your first time, it could get tricky. You actually do not need to create a src folder. In this case, the Source and Main structures will be created for you once the Gradle project is created. However, you will need to create the resources folder, you will get a pretty good idea of the importance of this folder, once the project is created.

Before you start your local project, we want to make one point. Our environment may look different than yours if you create your project as a Java Application as opposed to a Web Application. In our case, we created a Gradle + Web Application. We did this because we're thinking ahead, we will need to get database values into Google Sheets from a webpage which we will create in JSF (Java Server Faces). So we wanted to have our environment ready for that. If you are going this route, we suggest that you first download and install an Apache Tomcat Web Server to your computer or server box. We are using Tomcat 8.5 currently, downloaded to our system for other projects, you can get the latest version out there for your purpose.

You can either go online to fetch instructions on how to download and install Apache Tomcat or you can read helpful information in documents posted in our Apache Tomcat webpage; just scroll down to the downloadable content section of the page:

<https://www.dokollolutionsinc.com/JournalsApacheTomcatCoding.html>

While creating your web project (provided you selected that option), you will be prompted to choose a web server, you can go ahead and choose Apache Tomcat. We also have documentation on how to do this in NetBeans on the same page.

Journal Entry

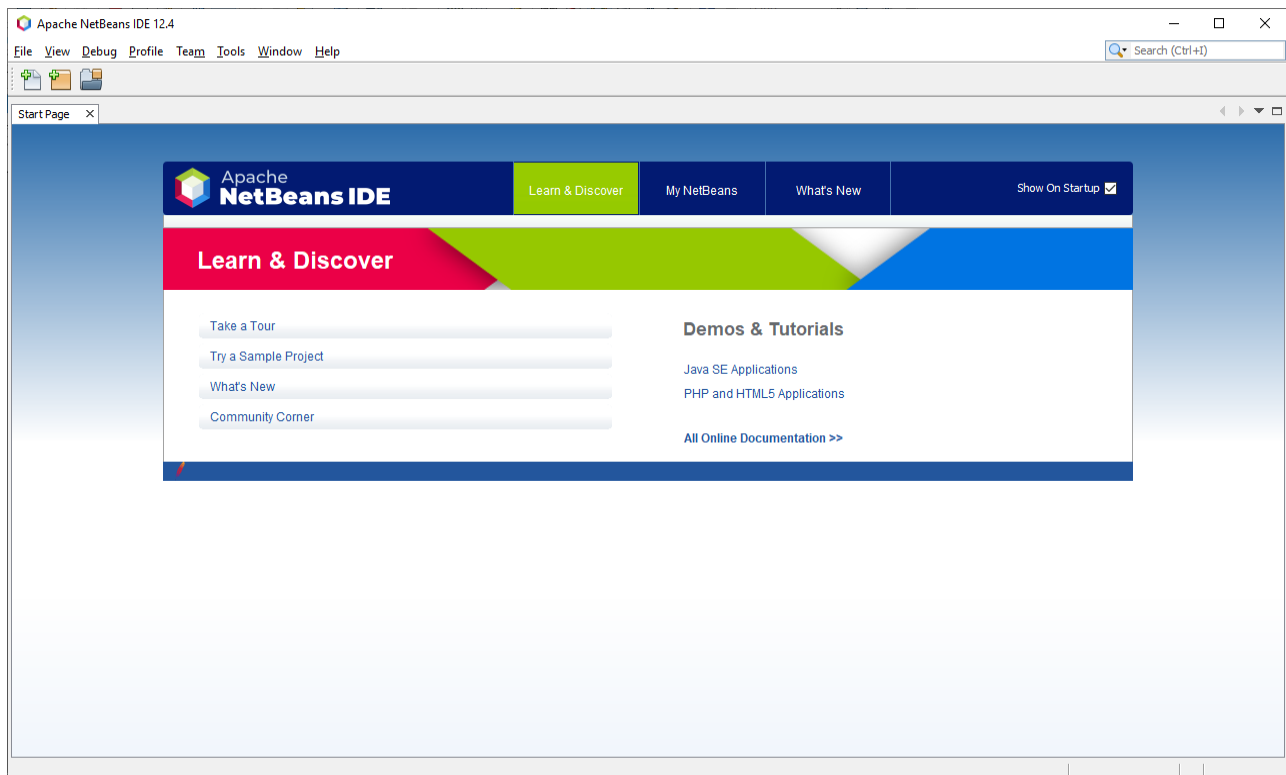
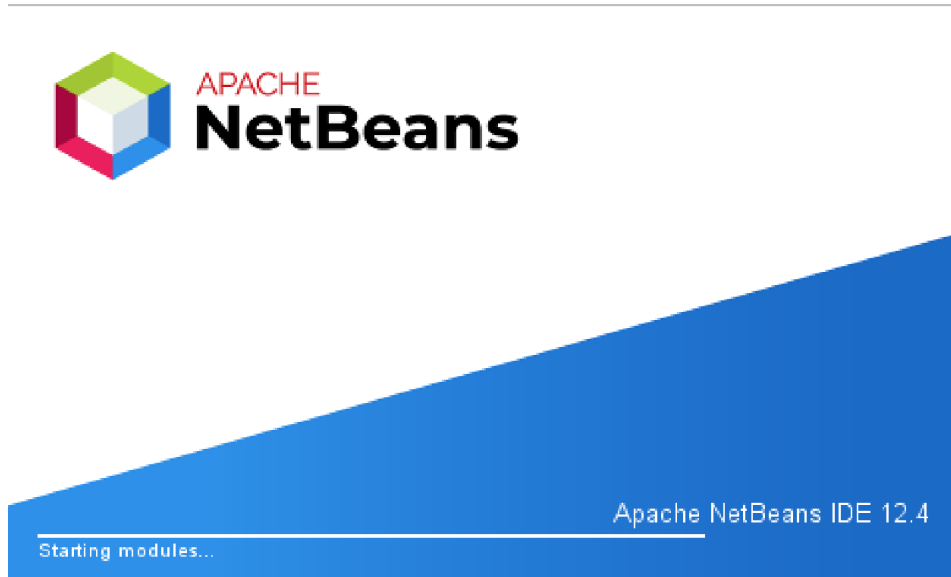
2021.10.20.6.13.AM

Step 4: Create NetBeans Java Web Application

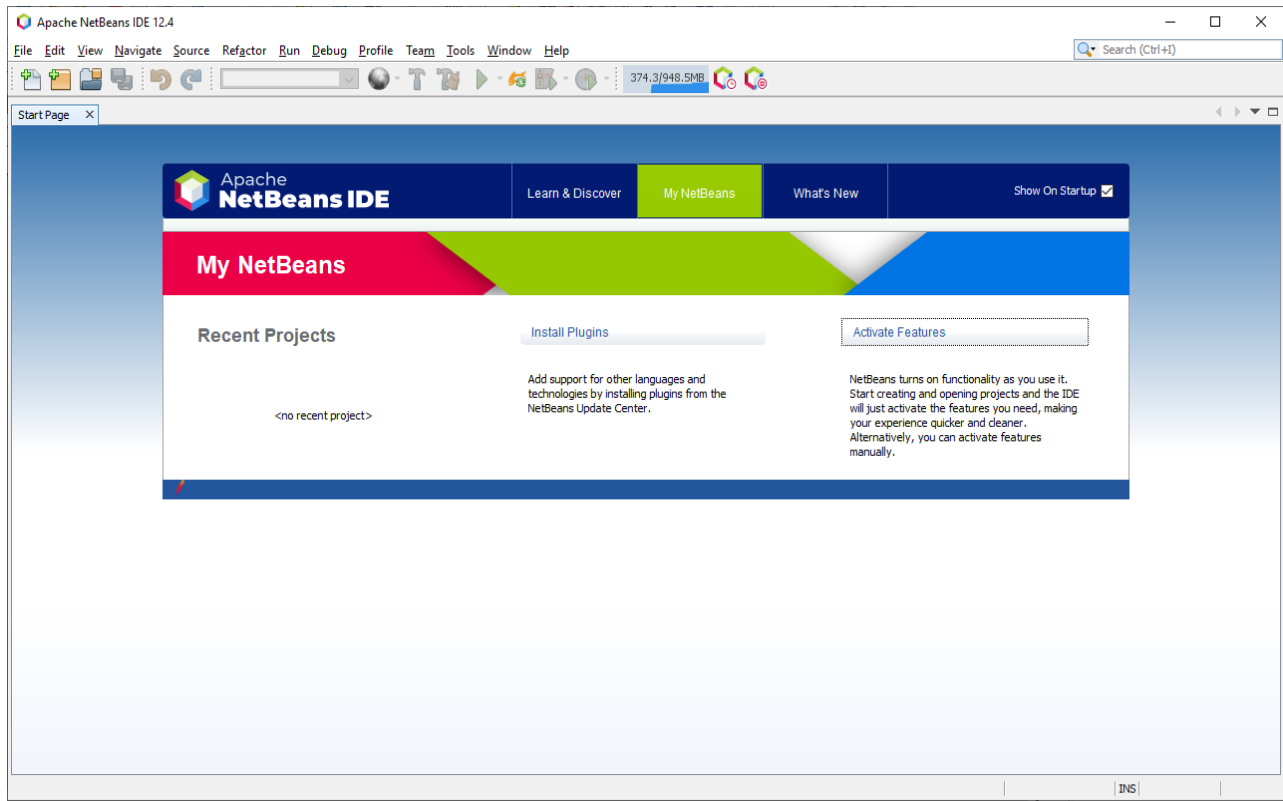
Go ahead and launch NetBeans 12 IDE

In this case, we will start things off for you. We will create a skeleton a project to make some comparisons between that and an existing working project, for educational purposes.

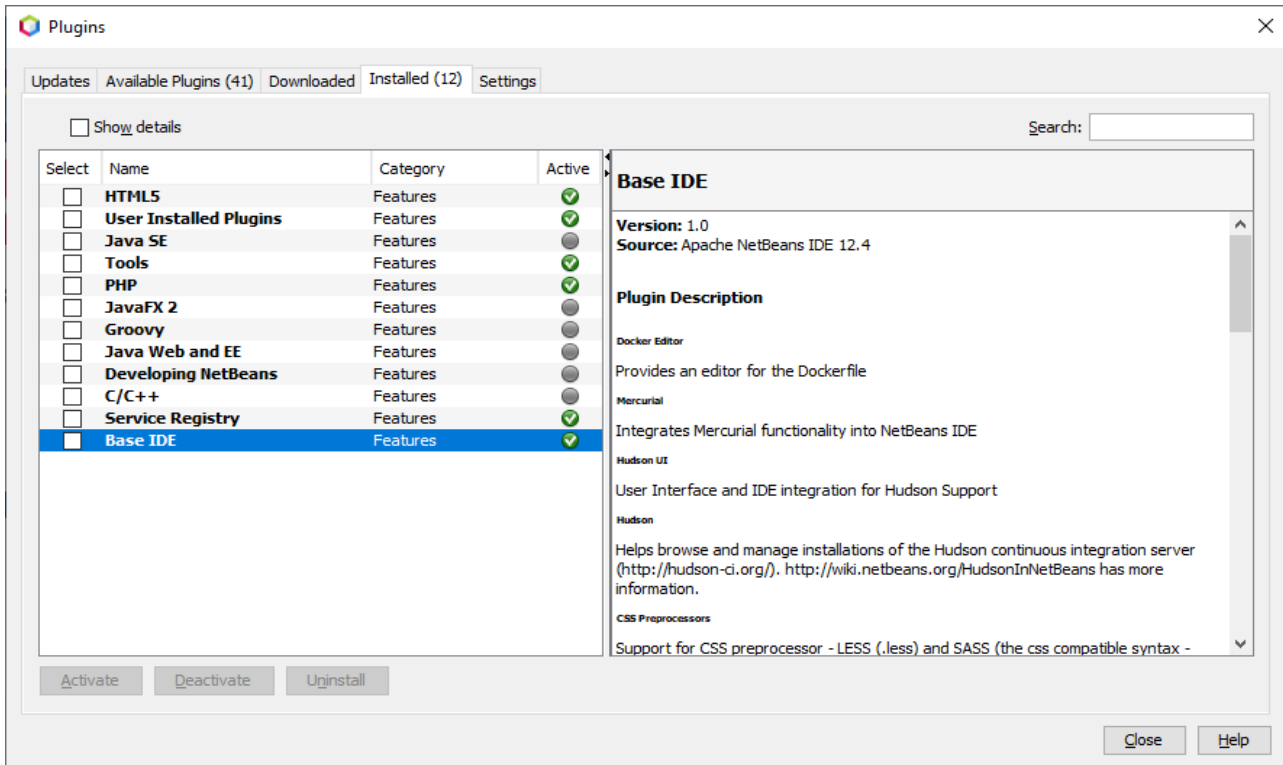
After launching NetBeans, you will be presented with the Splash screen, followed by the Start Page. Before using NetBeans 12 you will need to Activate plugins that it comes with. We are going to post all of the screenshots here to save you time reading.

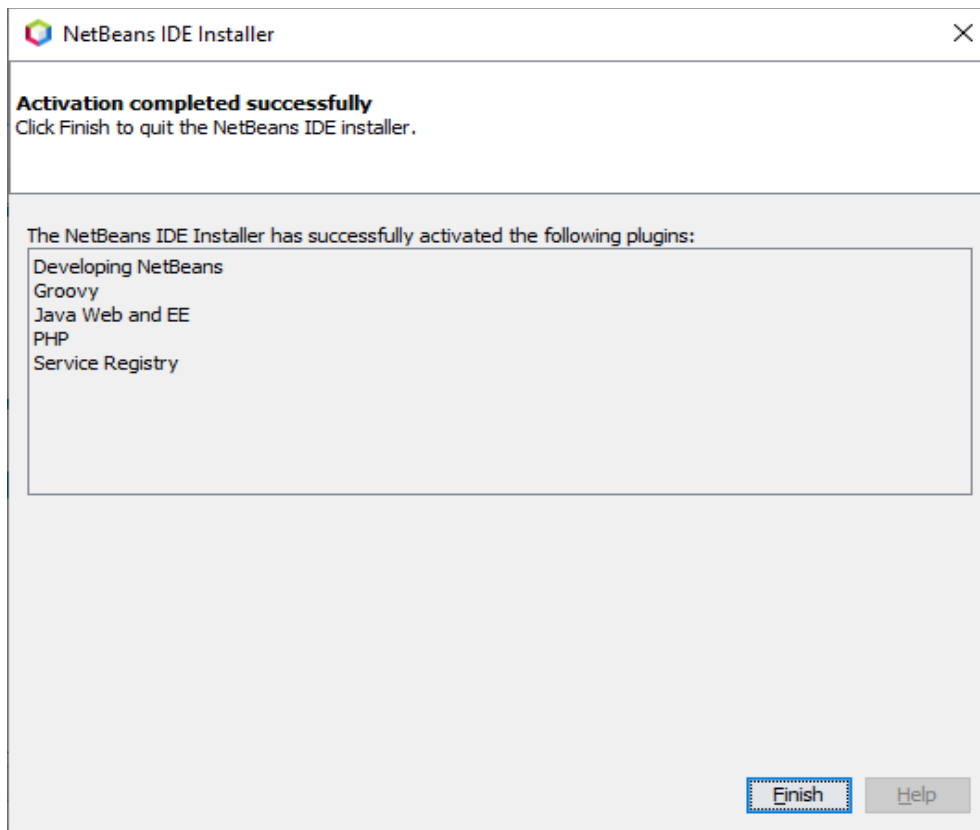


Be sure to activate environments- NetBeans will make available features as needed...



NetBeans has pre-installed some defaults for us

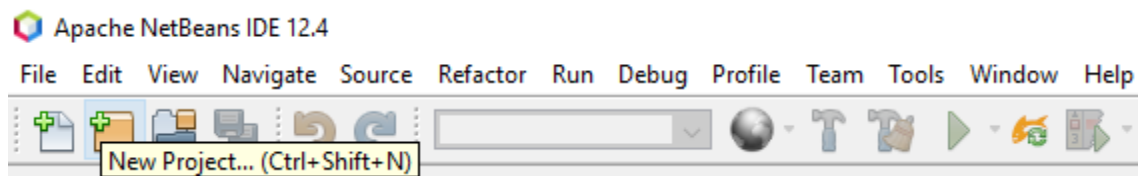


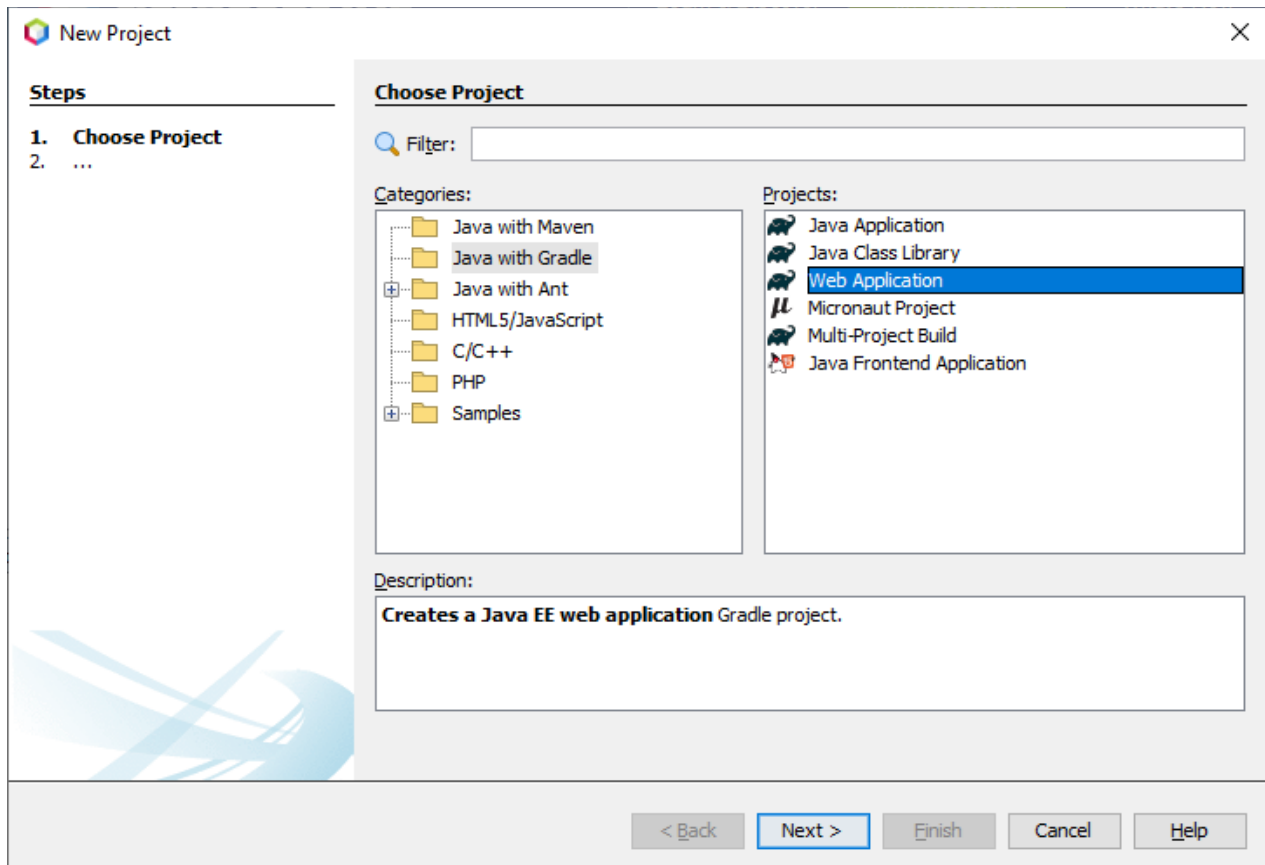


Now you are ready to actually do some configuring and coding...

1. Go to File + New Project
2. Select Gradle, then Web Application (as seen below)

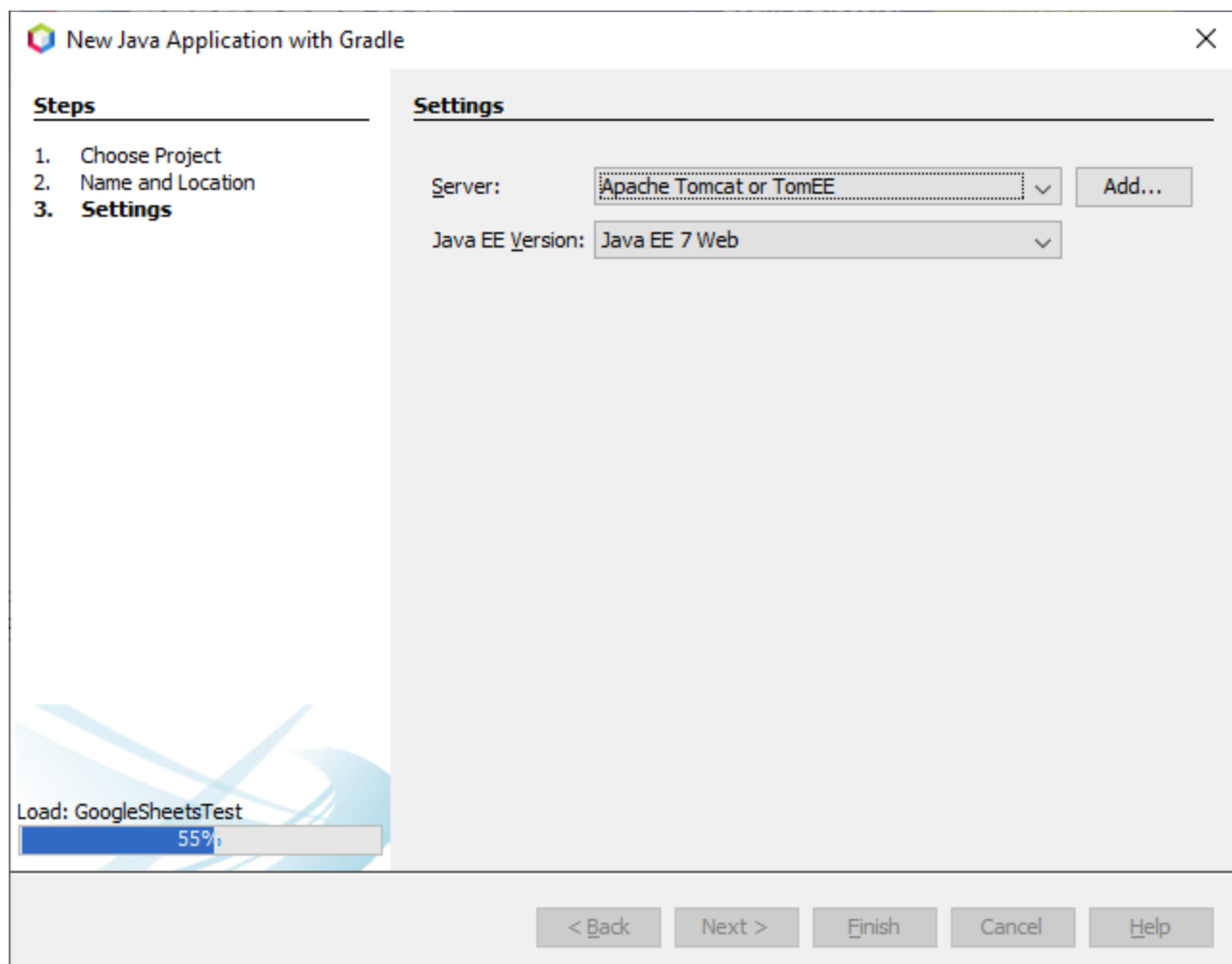
Hint: You can also start a new project without going through the file menu, by clicking on the navigation bar/ribbon button, as seen below





We're going to create a new project to walk you through the basic steps, then jump ahead to showing what our file structure looks like.

Provided you had already setup a server (Apache Tomcat) for your application or project, you would be presented with the following screen in the wizard. Otherwise, you would be setting up a server here by clicking the Add button. You might recall the URL we posted earlier, if not, take a look to download and setup an Apache Tomcat Web Server. You can go ahead and do that, if you are creating a Java Web Application.



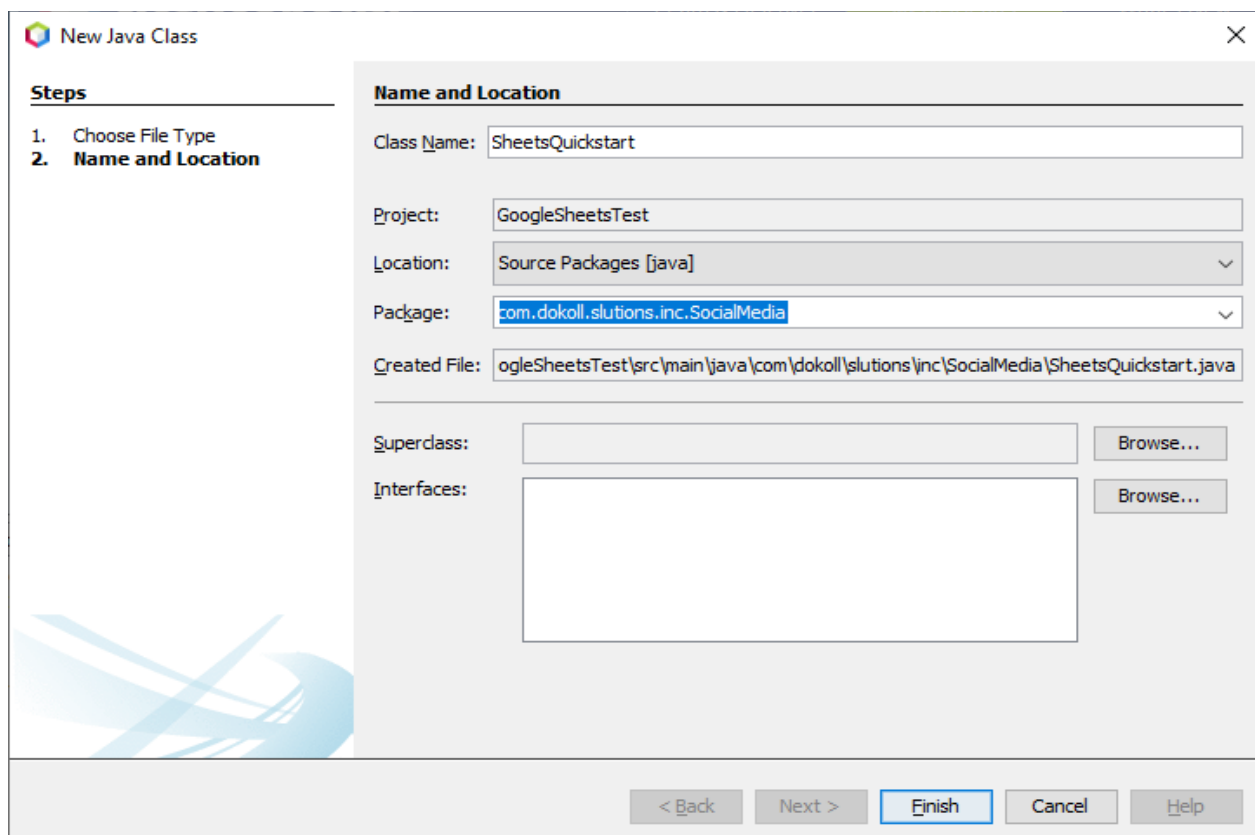
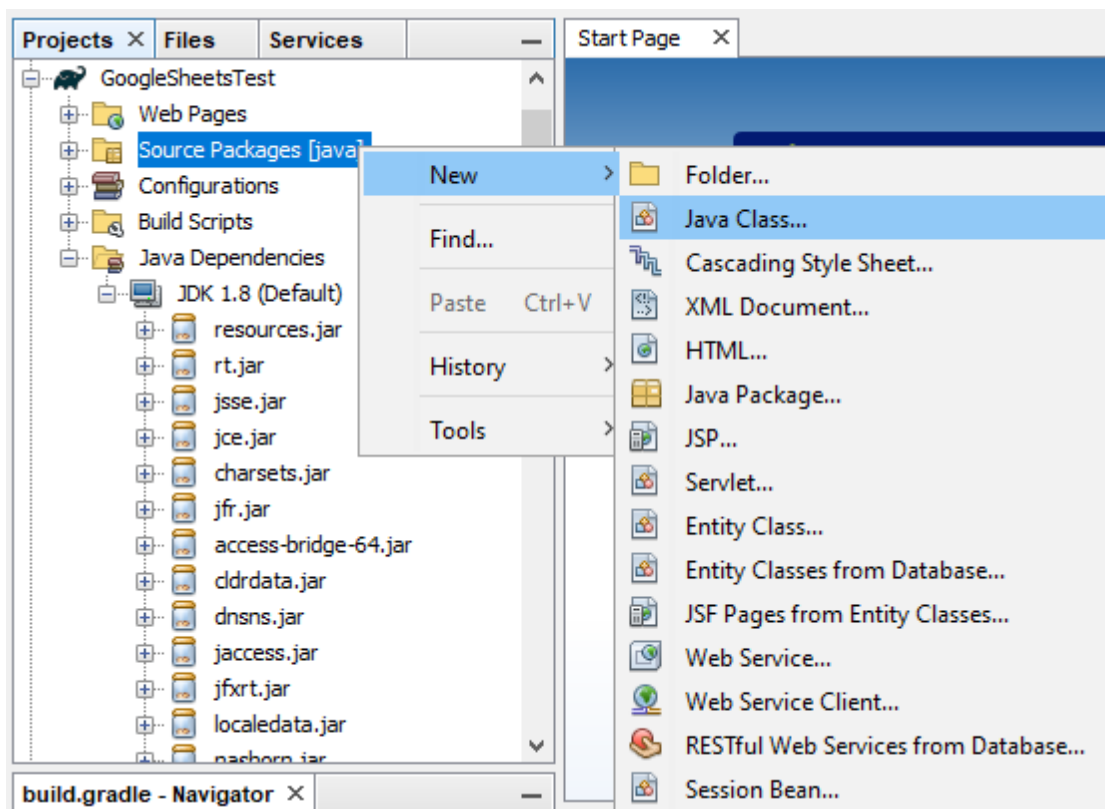
In our case, after clicking finish, this is what our environment looked like. We will delete the test application afterwards, but we want to show you the steps. We are going to create a Java class right away though so the GoogleSheetsTest project structure closely matches our existing working model. You are going to see a lot of errors in the code, just like we discussed earlier in this document.

Step 5: Create Java Class

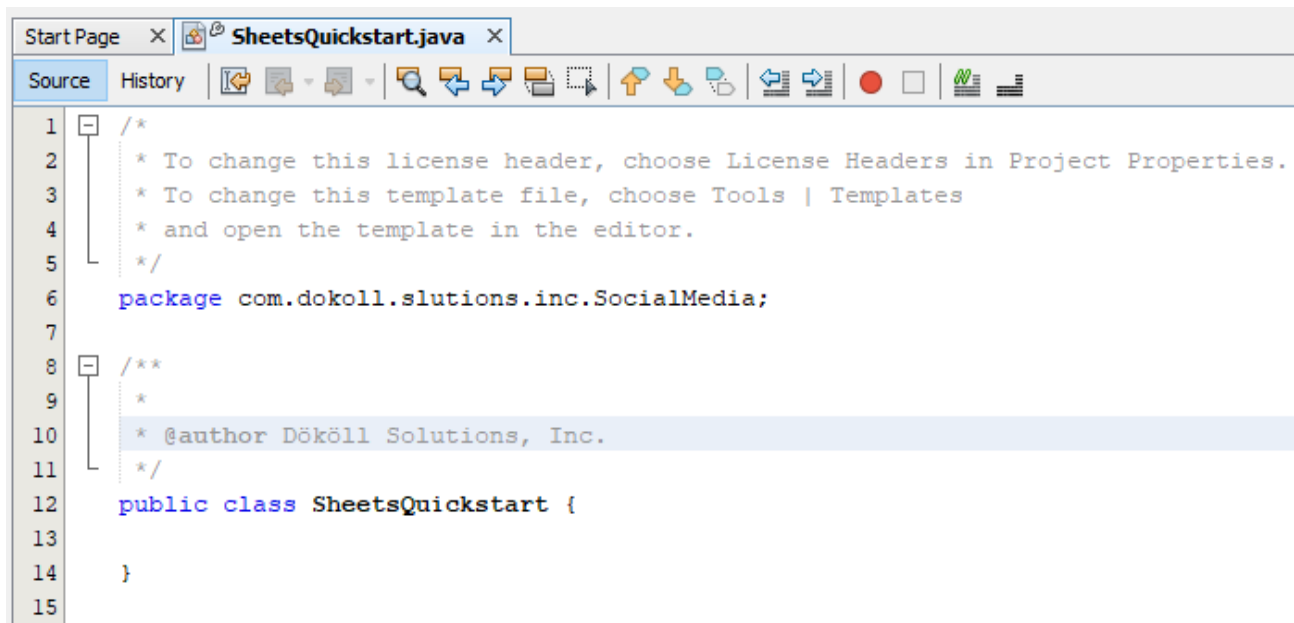
Here are the steps to creating your first Java class in NetBeans:

1. Right-click on the project's Source Packages section
2. Choose New + Java Class
3. Name your new class SheetsQuickstart
4. Replace the generated class code with the full code from Google

Hint: It is a good idea to create a package for all of your Java classes to keep your code and the file structure tidy. In our case, we called our package `com.dokoll.solutions.inc.SocialMedia`



Generated NetBeans Code sample



```
1  /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6  package com.dokoll.slutions.inc.SocialMedia;
7
8  /**
9  *
10 * @author Dököll Solutions, Inc.
11 */
12 public class SheetsQuickstart {
13
14 }
15
```

We went ahead and grabbed below Google Sheets sample for you, copy and paste it to your IDE:

<https://developers.google.com/sheets/api/quickstart/java>

```
import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.sheets.v4.Sheets;
import com.google.api.services.sheets.v4.SheetsScopes;
import com.google.api.services.sheets.v4.model.ValueRange;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.security.GeneralSecurityException;
import java.util.Collections;
import java.util.List;
public class SheetsQuickstart {
    private static final String APPLICATION_NAME = "Google Sheets API Java Quickstart";
    private static final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();
    private static final String TOKENS_DIRECTORY_PATH = "tokens";
    /**
     * Global instance of the scopes required by this quickstart.
     * If modifying these scopes, delete your previously saved tokens/ folder.
     */
    private static final List<String> SCOPES =
Collections.singletonList(SheetsScopes.SPREADSHEETS_READONLY);
    private static final String CREDENTIALS_FILE_PATH = "/credentials.json";
    /**
```

```

    * Creates an authorized Credential object.
    * @param HTTP_TRANSPORT The network HTTP Transport.
    * @return An authorized Credential object.
    * @throws IOException If the credentials.json file cannot be found.
    */
    private static Credential getCredentials(final NetHttpTransport HTTP_TRANSPORT) throws
IOException {
        // Load client secrets.
        InputStream in = SheetsQuickstart.class.getResourceAsStream(CREDENTIALS_FILE_PATH);
        if (in == null) {
            throw new FileNotFoundException("Resource not found: " + CREDENTIALS_FILE_PATH);
        }
        GoogleClientSecrets clientSecrets = GoogleClientSecrets.load(JSON_FACTORY, new
InputStreamReader(in));
        // Build flow and trigger user authorization request.
        GoogleAuthorizationCodeFlow flow = new GoogleAuthorizationCodeFlow.Builder(
            HTTP_TRANSPORT, JSON_FACTORY, clientSecrets, SCOPES)
            .setDataStoreFactory(new FileDataStoreFactory(new
java.io.File(TOKENS_DIRECTORY_PATH)))
            .setAccessType("offline")
            .build();
        LocalServerReceiver receiver = new
LocalServerReceiver.Builder().setPort(8888).build();
        return new AuthorizationCodeInstalledApp(flow, receiver).authorize("user");
    }
    /**
     * Prints the names and majors of students in a sample spreadsheet:
     *
     * https://docs.google.com/spreadsheets/d/1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs74OgvE2upms/edit
     */
    public static void main(String... args) throws IOException, GeneralSecurityException {
        // Build a new authorized API client service.
        final NetHttpTransport HTTP_TRANSPORT =
GoogleNetHttpTransport.newTrustedTransport();
        final String spreadsheetId = "1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs74OgvE2upms";
        final String range = "Class Data!A2:E";
        Sheets service = new Sheets.Builder(HTTP_TRANSPORT, JSON_FACTORY,
getCredentials(HTTP_TRANSPORT))
            .setApplicationName(APPLICATION_NAME)
            .build();
        ValueRange response = service.spreadsheets().values()
            .get(spreadsheetId, range)
            .execute();
        List<List<Object>> values = response.getValues();
        if (values == null || values.isEmpty()) {
            System.out.println("No data found.");
        } else {
            System.out.println("Name, Major");
            for (List row : values) {
                // Print columns A and E, which correspond to indices 0 and 4.
                System.out.printf("%s, %s\n", row.get(0), row.get(4));
            }
        }
    }
}

```

As expected, you should have received a bunch of errors...

```
6 package com.dokoll.slutions.inc.SocialMedia;
7
8 /**
9 package com.google.api.client.auth.oauth2 does not exist
10 ----
11 (Alt-Enter shows hints)
12
13 import com.google.api.client.auth.oauth2.Credential;
14 import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
15 import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
16 import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
17 import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
18 import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
19 import com.google.api.client.http.javanet.NetHttpTransport;
20 import com.google.api.client.json.JsonFactory;
21 import com.google.api.client.json.jackson2.JacksonFactory;
22 import com.google.api.client.util.store.FileDataStoreFactory;
23 import com.google.api.services.sheets.v4.Sheets;
24 import com.google.api.services.sheets.v4.SheetsScopes;
25 import com.google.api.services.sheets.v4.model.ValueRange;
```

Hint: The above tool-tip was achieved by adding our mouse cursor on one of the error bubbles on the number line, in this case, for the Credential class import.

Journal Entry

2021.10.20.12.15.PM

Okay, now that we have these errors, let us see what we can do to remove them so the code is clean and ready to roll... We are going to take you through the steps to getting rid of the errors

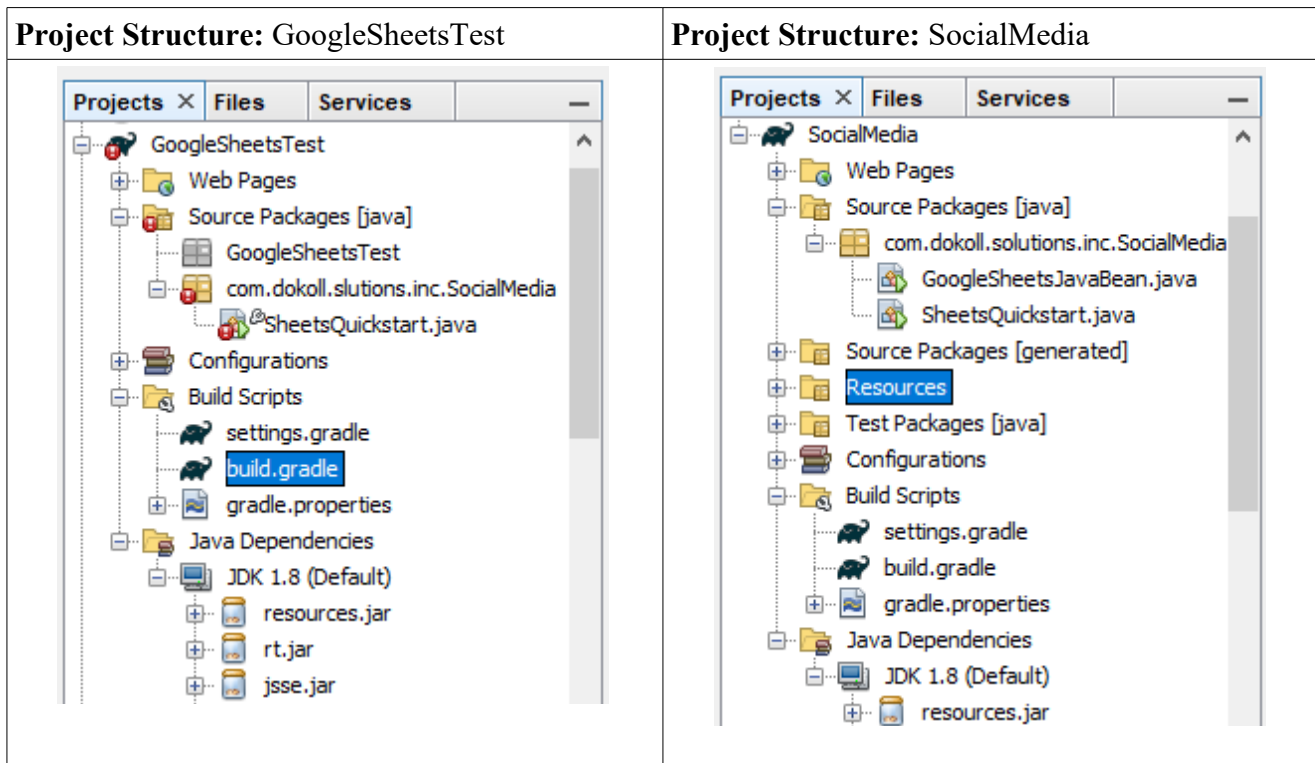
Step 6: Create and Configure Project Dependencies

First, we will select the default build.gradle file that NetBeans created for us to show you the contents. We're making it easier for you to see what we've added to our working project to connect to Google Cloud. In the Test App below, the build file is missing a lot of important configuration items...

```
1 apply plugin: 'java'
2 apply plugin: 'jacoco'
3 apply plugin: 'war'
4
5 repositories {
6     mavenCentral()
7 }
8
9 dependencies {
10     providedCompile 'javax:javaee-web-api:7.0'
11
12     testImplementation 'junit:junit:4.13'
13 }
```

The dependencies portion is your friend. That is where all of the work will be done to import all of the libraries to run Google Sheets v4 code in NetBeans. If you recall, we mentioned that NetBeans does a lot of the work for us, as far as the file and folder structures for code and resources (i.e. src/main). If you look above the *Build Scripts* section where the build.gradle file is located, you can see a *Source Packages* section. That portion of the structure corresponds to **src/main**.

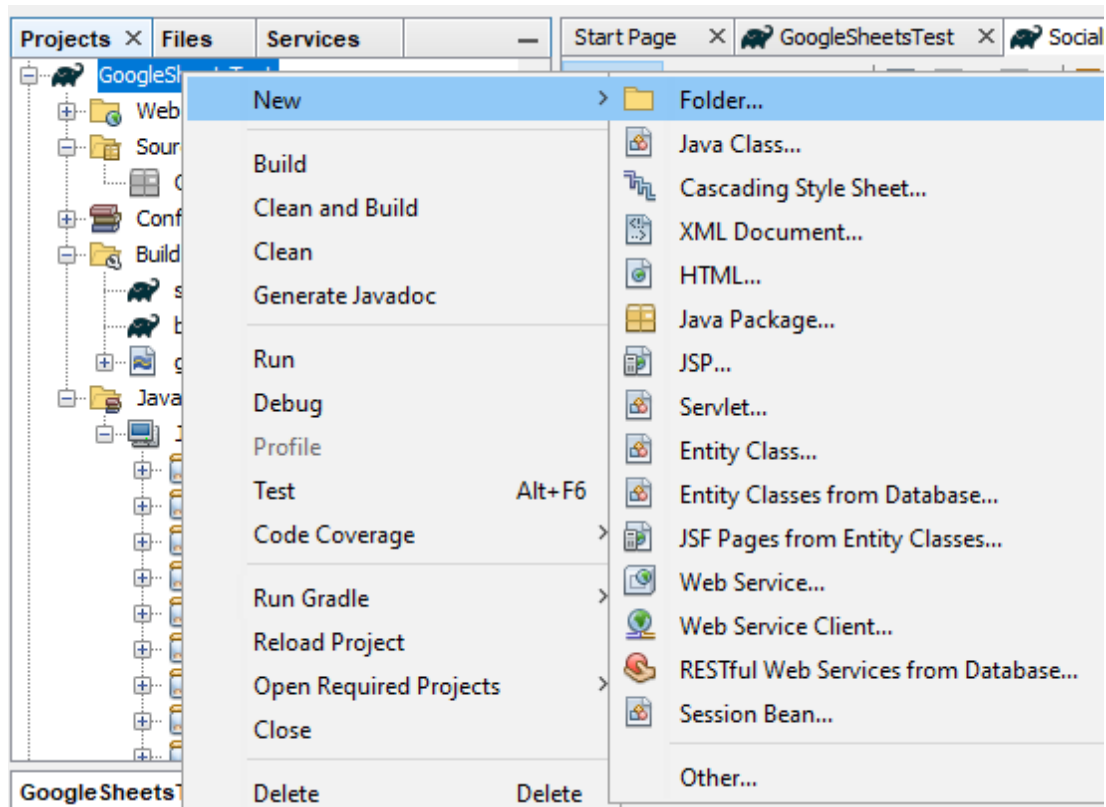
Now, we have the sample code, we know we need to configure the build.gradle file with dependency information. How do we get started? Well, if you look below, the screenshot on the left is our test project, the one on the right is the completed project. We have created a Resources folder to hold the very important JSON file.



We are going to come back to Gradle in a moment and skip ahead to showing you how to create a placeholder (Resources folder) for the JSON file needed for the project.

Go ahead and create your own Resources folder in NetBeans to call the JSON file, here are the steps:

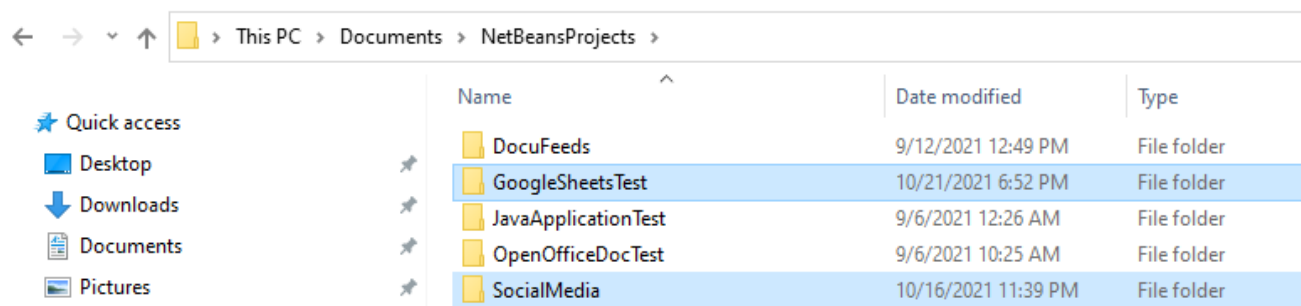
1. Right-click on the project package name (example: GoogleSheetsTest)
2. Choose New + Folder + call the folder Resources



Technical Notes:

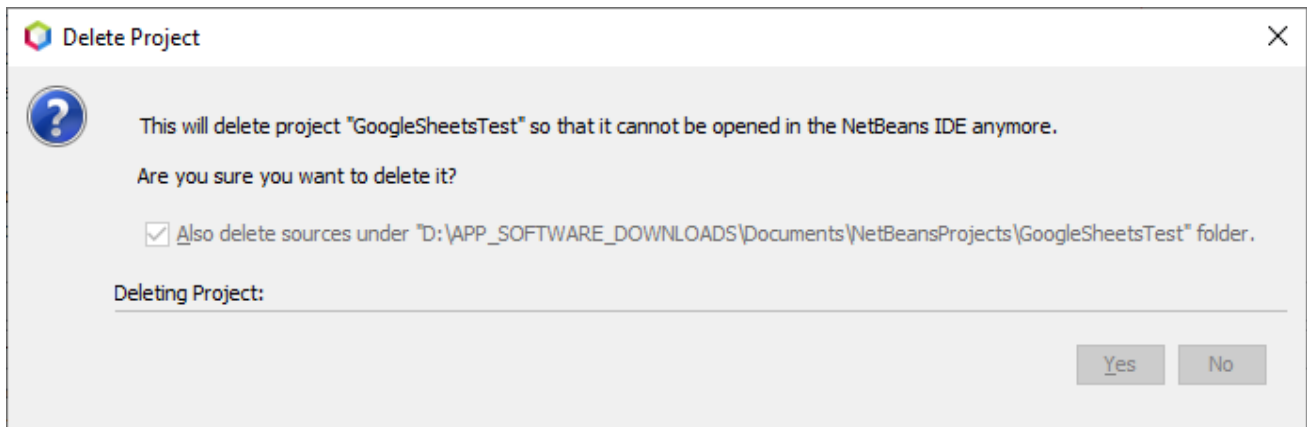
It is a good idea to know where NetBeans puts your project files in your environment (i.e. computer or server box). You will need to visit that Resources folder location to put the project's JSON file into. If you recall, you downloaded a JSON file with credentials data from your Google Cloud Console environment.

We installed NetBeans in the following directory. Let us go there to have a look:



Okay, that looks familiar, we can see both projects in the directory. However, we think you have seen enough of the basics, so we are going to get rid of the test application. We are going to go ahead and continue our documentation with the actual working project. Delete the test 'GoogleSheetsTest' project, you can simply right-click and delete from the directory. NetBeans does a good job removing all files if it is done from the delete task within the IDE. You just have to right-click the project in the

explorer window and choose Delete. You should be presented with the following:



*If the project is still showing in the directory, you can just hit the refresh button in the Windows Explorer path- or you can also right-click anywhere that is empty/blank in Windows Explorer, choose Refresh in the menu...

The screenshot shows a Windows Explorer window with the following table of contents:

Name	Date modified	Type
DocuFeeds	9/12/2021 12:49 PM	File folder
JavaApplicationTest	9/6/2021 12:26 AM	File folder
OpenOfficeDocTest	9/6/2021 10:25 AM	File folder
SocialMedia	10/16/2021 11:39 PM	File folder
WebFeeds	9/7/2021 12:51 PM	File folder
WebTest	9/21/2021 7:32 PM	File folder

Journal Entry

2021.10.20.4.55.PM

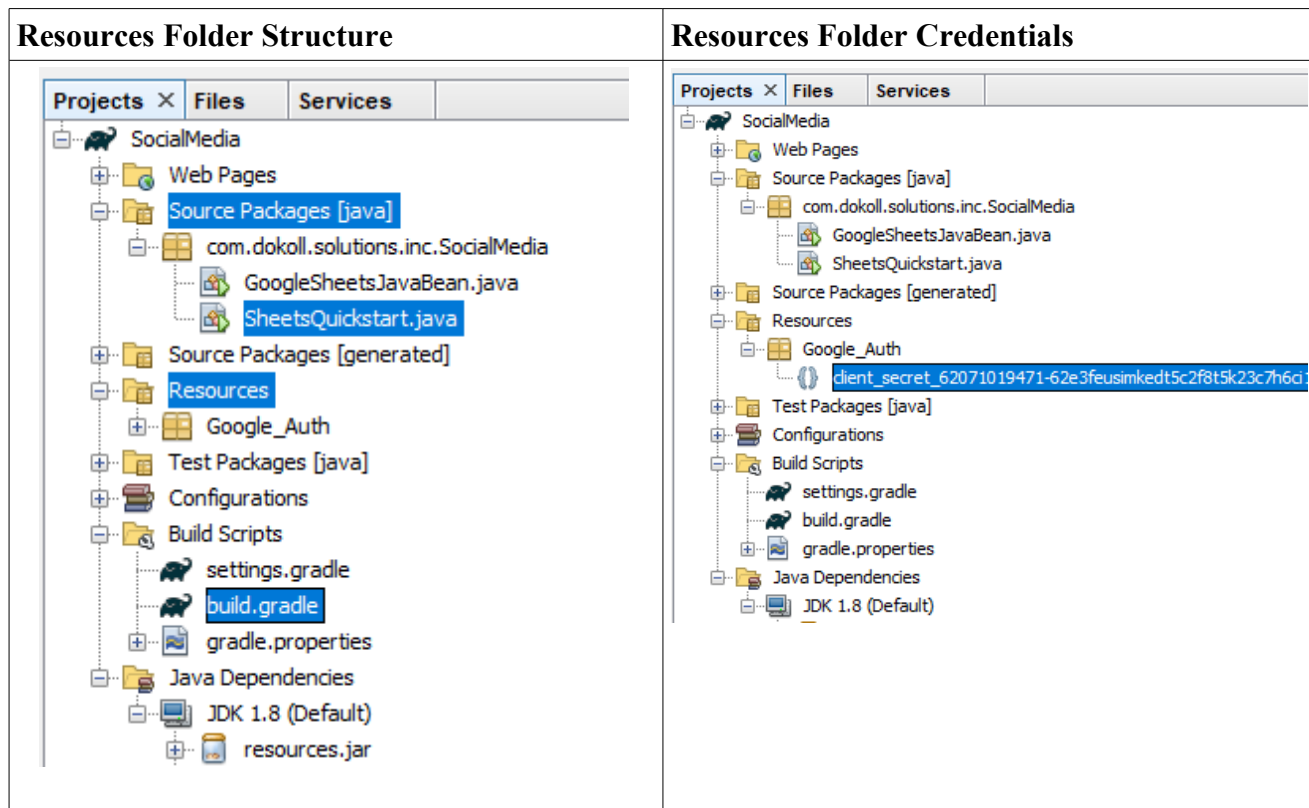
Let us get inside our SocialMedia project to see the file structure from a local directory stand-point. Your Resources folder will be empty.

We made a special point to create a Sub-folder within the Resources folder called Google_Auth to put our authentication-specific JSON file for the project. We will go through that important file in a moment.

We also wanted you look at our Resources folder from the project file structure standpoint... Let us have a look next...

Step 7: Create and Configure Project Dependencies

Here is what our Resources folder looks like from within the project



As mentioned previously, the highlighted portions are must-haves for your project to work. The other piece of the puzzle is to configure the build.gradle file to pull in the library packages from the Google APIs into the project. Once you have implemented all packages in the build.gradle file, all errors showing in your code will vanish like magic.

We are going to show you what is included in our build.gradle file so you can see our configuration. You should also take a good look at the Gradle configuration sample Google made available on their website to compare. It is important to take a look at that because there are some important distinctions.

Below is Google's configuration of the Gradle project to run Google Sheets v4 code. If you are not familiar with Gradle, there are tons of tutorials online about downloading Gradle and using it through command-line code, the latest version can be found here: <https://gradle.org/install/>.

We have made it easier for you by having you run a Gradle project within a UI instead. We understand that to some beginners, command-line programming can be daunting, that is where NetBeans comes in... Now, take look at the configuration below, based on Gradle 2.3. The file makes use of a variable *mainClassName* to allow your project to call and run the Quickstart (SheetsQuickstart.java) sample as the default class.

Obviously, you can switch to any other Java class as needed. Go ahead and compare the two screenshots below, then we will dive into why they are important in the configuration...

Google Sheets v4 Gradle Configuration

```
sheets/quickstart/build.gradle View on GitHub
```

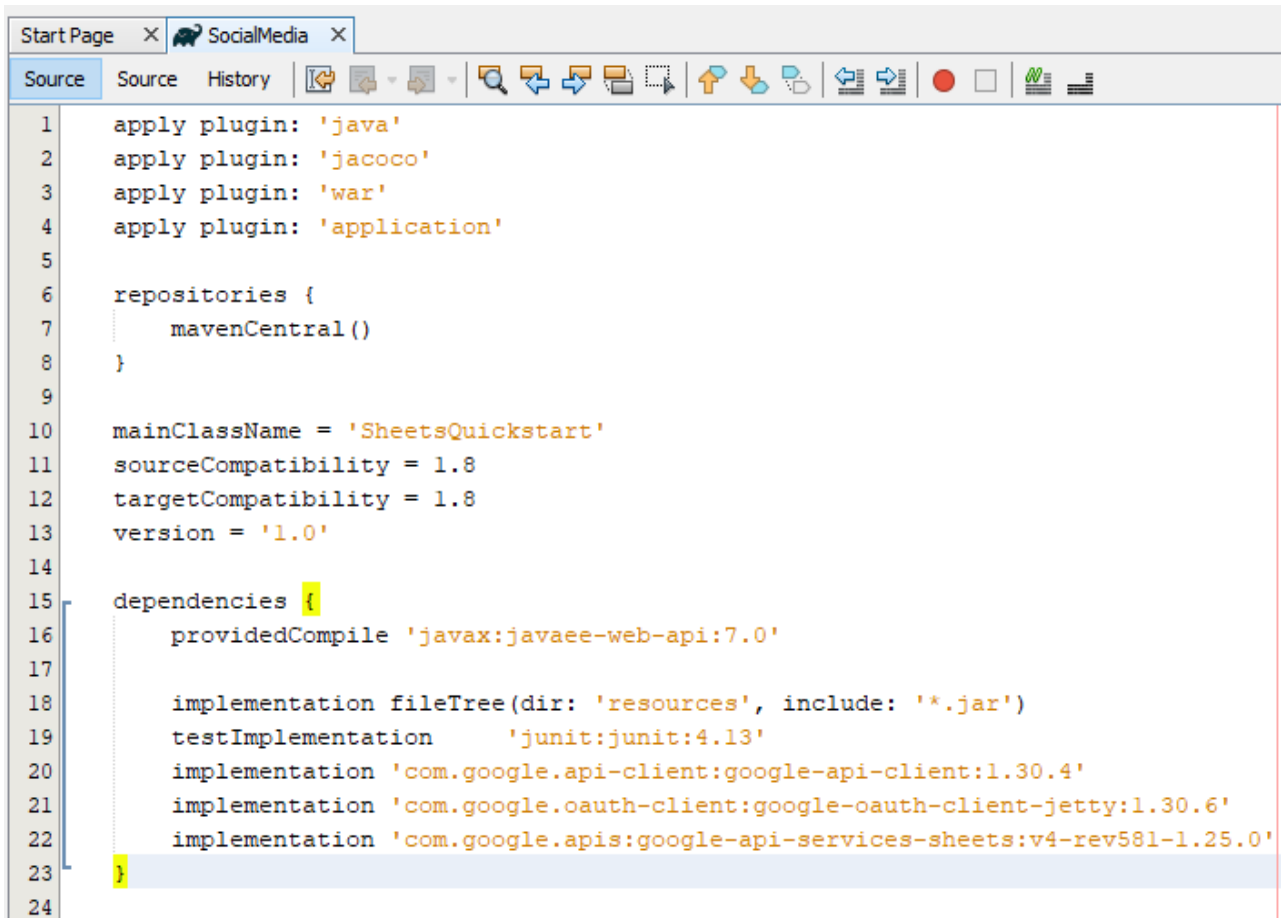
```
apply plugin: 'java'
apply plugin: 'application'

mainClassName = 'SheetsQuickstart'
sourceCompatibility = 1.8
targetCompatibility = 1.8
version = '1.0'

repositories {
    mavenCentral()
}

dependencies {
    compile 'com.google.api-client:google-api-client:1.30.4'
    compile 'com.google.oauth-client:google-oauth-client-jetty:1.30.6'
    compile 'com.google.apis:google-api-services-sheets:v4-rev581-1.25.0'
}
```

Google Sheets v4 Gradle Configuration in NetBeans



The screenshot shows the NetBeans IDE interface with a source code editor. The editor displays the following Gradle build file configuration:

```
1  apply plugin: 'java'
2  apply plugin: 'jacoco'
3  apply plugin: 'war'
4  apply plugin: 'application'
5
6  repositories {
7      mavenCentral()
8  }
9
10 mainClassName = 'SheetsQuickstart'
11 sourceCompatibility = 1.8
12 targetCompatibility = 1.8
13 version = '1.0'
14
15 dependencies {
16     providedCompile 'javax:javaee-web-api:7.0'
17
18     implementation fileTree(dir: 'resources', include: '*.jar')
19     testImplementation 'junit:junit:4.13'
20     implementation 'com.google.api-client:google-api-client:1.30.4'
21     implementation 'com.google.oauth-client:google-oauth-client-jetty:1.30.6'
22     implementation 'com.google.apis:google-api-services-sheets:v4-rev581-1.25.0'
23 }
24
```

In addition to the `mainClassName` variable, you have to tell Gradle which JDK (Java Development Kid) is currently installed on the current machine so that your program uses that version, identified by the `sourceCompatibility` variable. Furthermore, the configuration pulls in Google API, Oauth, and Sheets packages/classes using the dependencies block. This is a major difference in the way NetBeans uses the power of Gradle to call Google classes into the project to run Google Sheets v4 code versus a standard Standalone Gradle application.

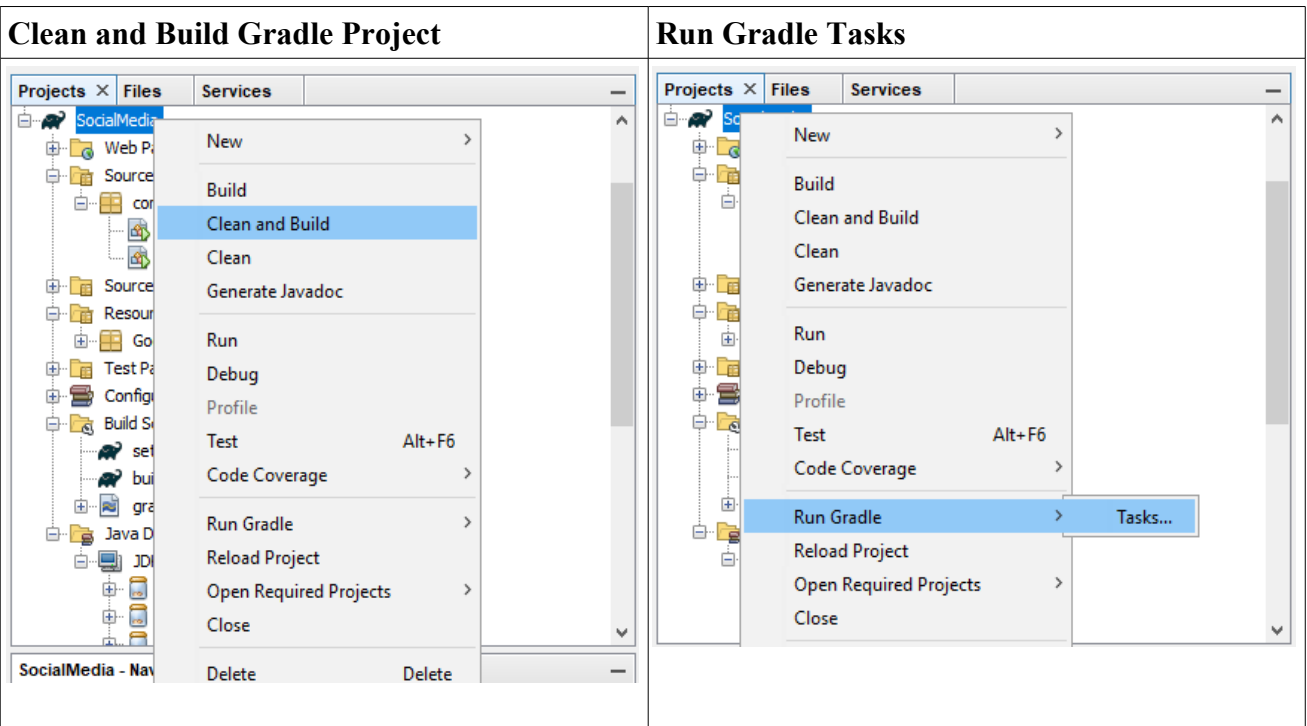
Standalone Gradle compiles packages from Google APIs

```
dependencies {
    compile 'com.google.api-client:google-api-client:1.30.4'
    compile 'com.google.oauth-client:google-oauth-client-jetty:1.30.6'
    compile 'com.google.apis:google-api-services-sheets:v4-rev581-1.25.0'
}
```

Gradle in NetBeans implements packages from Google APIs

```
20 implementation 'com.google.api-client:google-api-client:1.30.4'
21 implementation 'com.google.oauth-client:google-oauth-client-jetty:1.30.6'
22 implementation 'com.google.apis:google-api-services-sheets:v4-rev581-1.25.0'
```

Provided your Gradle project is ready to be run, go ahead and Clean/Build your project to ensure all errors in your code go away... You should also Run Gradle to make certain that the implementation is solid.



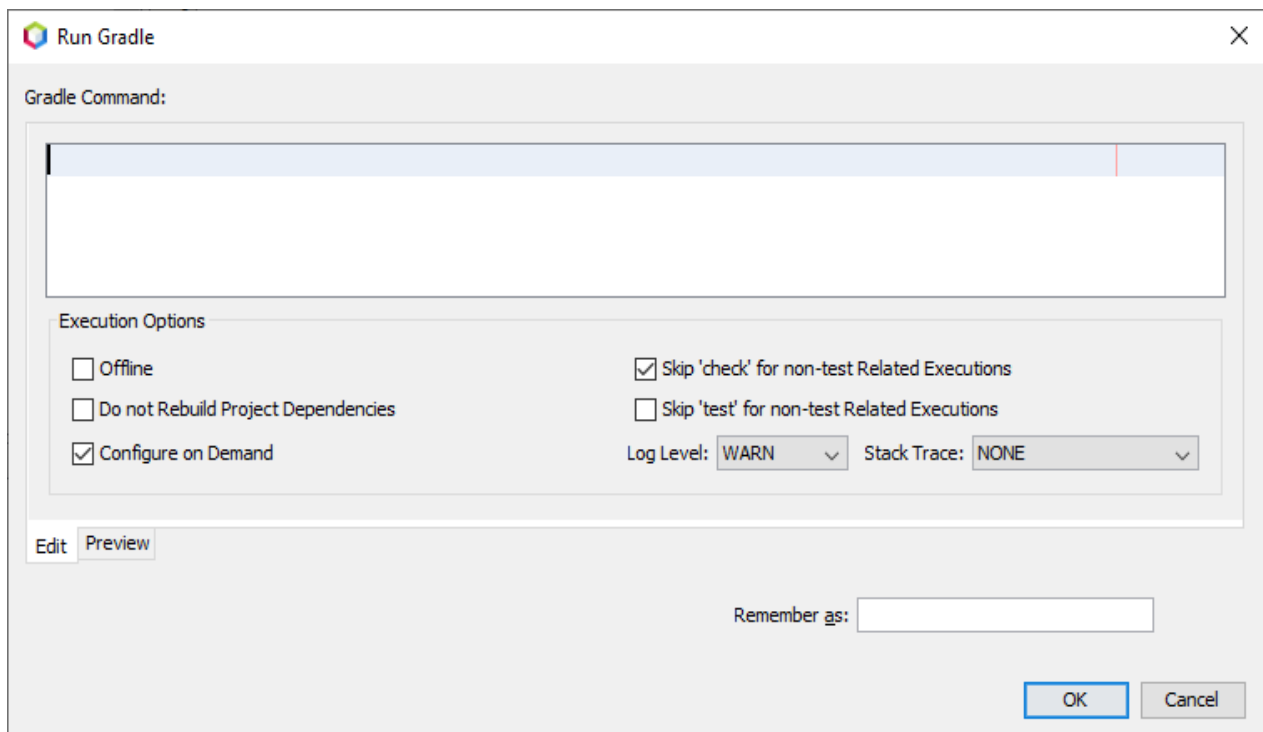
In our case, after Cleaning and Building our project we got the following results; we're making available an excerpt of the readout-



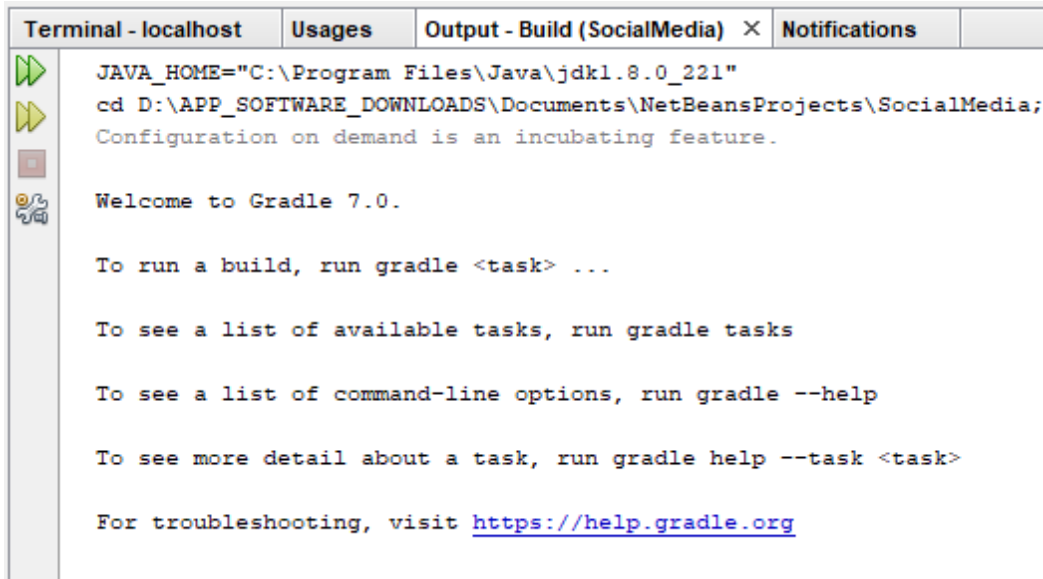
```
Terminal - localhost  Usages  Output - Build (SocialMedia) x  Notifications
JAVA_HOME="C:\Program Files\Java\jdk1.8.0_221"
cd D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia; .\gradlew.bat
Configuration on demand is an incubating feature.
> Task :clean
> Task :compileJava
> Task :processResources
> Task :classes
> Task :jar
> Task :startScripts
> Task :distTar
> Task :distZip
> Task :explodedWar
> Task :war
> Task :assemble
> Task :build

BUILD SUCCESSFUL in 8s
9 actionable tasks: 9 executed
```

the implementation of Google API packages need to be checked too, so the way to do this is to run the tasks in Gradle. As in the above screenshot at right, once you select Tasks, you are presented with the following-



From here, you can run a set of command-line code samples to ensure all is well. However, we will skip that step since we're mostly concerned with ensuring that our implementation of Google packages are good to go- You can just hit okay.



```
Terminal - localhost  Usages  Output - Build (SocialMedia) x  Notifications

JAVA_HOME="C:\Program Files\Java\jdk1.8.0_221"
cd D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia;
Configuration on demand is an incubating feature.

Welcome to Gradle 7.0.

To run a build, run gradle <task> ...

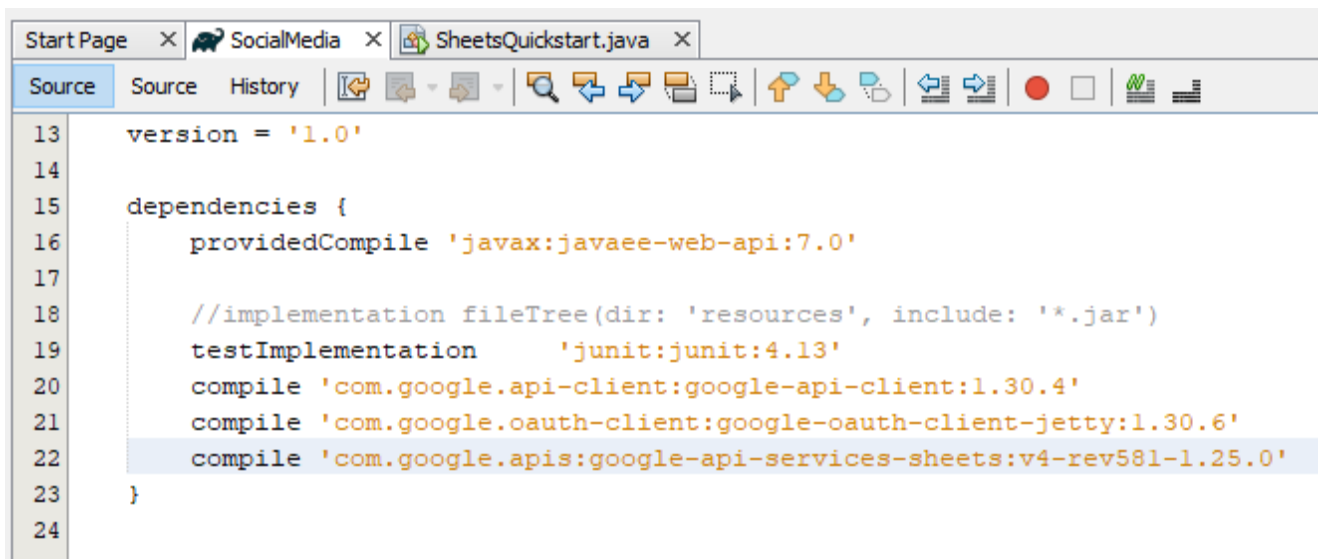
To see a list of available tasks, run gradle tasks

To see a list of command-line options, run gradle --help

To see more detail about a task, run gradle help --task <task>

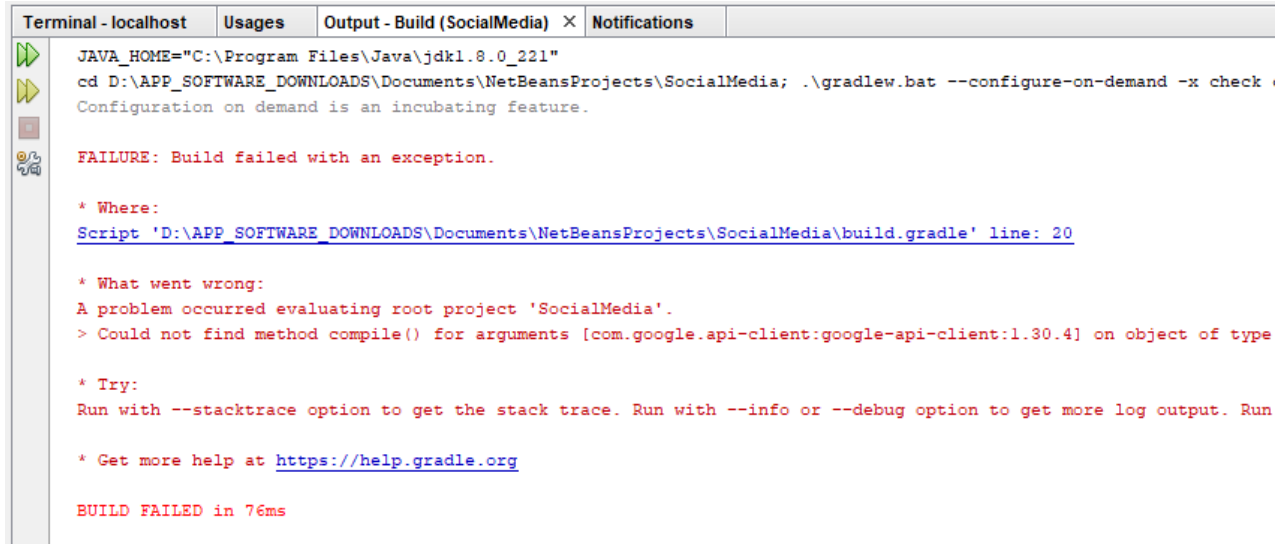
For troubleshooting, visit https://help.gradle.org
```

At this point, if your Gradle configuration was not properly setup, after hitting okay you would have received some build script errors- We are going to actually use the build.gradle code as is from Google to show you what we mean.



```
Start Page x  SocialMedia x  SheetsQuickstart.java x
Source  Source  History  [Icons]
13  version = '1.0'
14
15  dependencies {
16      providedCompile 'javax:javaee-web-api:7.0'
17
18      //implementation fileTree(dir: 'resources', include: '*.jar')
19      testImplementation 'junit:junit:4.13'
20      compile 'com.google.api-client:google-api-client:1.30.4'
21      compile 'com.google.oauth-client:google-oauth-client-jetty:1.30.6'
22      compile 'com.google.apis:google-api-services-sheets:v4-rev581-1.25.0'
23  }
24
```

Okay, so now, we're going to Clean and Build the project in NetBeans... you should see that Gradle could not locate Google API packages...



```
Terminal - localhost | Usages | Output - Build (SocialMedia) x | Notifications
JAVA_HOME="C:\Program Files\Java\jdk1.8.0_221"
cd D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia; .\gradlew.bat --configure-on-demand -x check
Configuration on demand is an incubating feature.

FAILURE: Build failed with an exception.

* Where:
Script 'D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia\build.gradle' line: 20

* What went wrong:
A problem occurred evaluating root project 'SocialMedia'.
> Could not find method compile() for arguments [com.google.api-client:google-api-client:1.30.4] on object of type

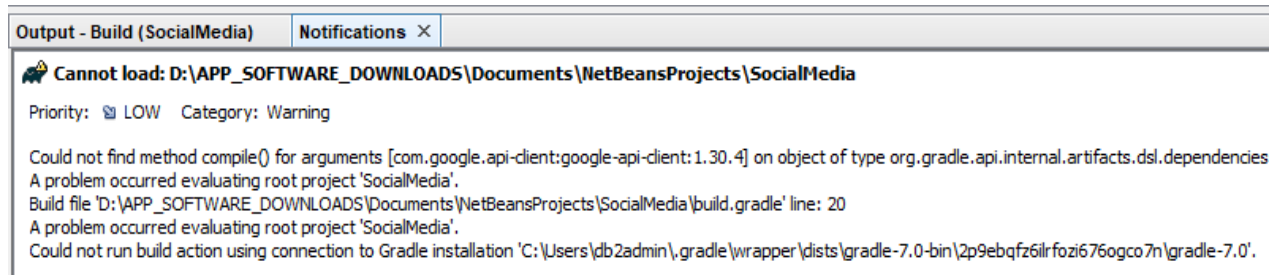
* Try:
Run with --stacktrace option to get the stack trace. Run with --info or --debug option to get more log output. Run

* Get more help at https://help.gradle.org

BUILD FAILED in 76ms
```

that is just an excerpt of the full error message but you can plainly see the method called “compile” is not recognized...

We are going to also run Gradle Tasks so you can observe what the Gradle project in NetBeans generates this time around...



```
Output - Build (SocialMedia) | Notifications x
Cannot load: D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia
Priority: LOW Category: Warning
Could not find method compile() for arguments [com.google.api-client:google-api-client:1.30.4] on object of type org.gradle.api.internal.artifacts.dsl.dependencies
A problem occurred evaluating root project 'SocialMedia'.
Build file 'D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia\build.gradle' line: 20
A problem occurred evaluating root project 'SocialMedia'.
Could not run build action using connection to Gradle installation 'C:\Users\db2admin\gradle\wrapper\dists\gradle-7.0-bin\2p9ebqfz6ilrfozi676ogco7n\gradle-7.0'.
```

Again, that is just a portion of the full error log... this proves our point before, that the compile method will only work through the Standalone Gradle project but not within NetBeans. We suspect it may have the same behaviour in other IDEs but that is yet to be tested....

We're putting the build.gradle file back to normal and Cleaning/Building the project again...

Okay, so, you Cleaned and Rebuilt your Gradle project and all look good, now what? Well, let us connect the local NetBeans/Gradle Java application to Google to see what we can pull up-

Journal Entry
2021.10.21.7.01.PM

The Google Sheets Quickstart code is merely a sample, which means that you cannot just copy and paste the code into your IDE and expect it to work, even though you had gone through your NetBeans configurations and Gradle throws no errors. There are some items you will need to get sorted out first. As one example, if you remember, you downloaded a JSON file, your File Name for that will most likely be different than what is hard-coded in the Google Sheets Quickstart sample. Let us go through the code chunk by chunk and tell you what you will need to change so you are prepared to connect your NetBeans application to your Google Cloud project.

Step 8: Running SheetsQuickstart Java Application

First order of business is, you need to change the Application Name in the code. You have to add your actual Google Cloud project information. If you need to remember what you called your project, take look at **Step 1** at the beginning of these Journal Entries.

```
import com.google.api.client.auth.oauth2.Credential;
import com.google.api.client.extensions.java6.auth.oauth2.AuthorizationCodeInstalledApp;
import com.google.api.client.extensions.jetty.auth.oauth2.LocalServerReceiver;
import com.google.api.client.googleapis.auth.oauth2.GoogleAuthorizationCodeFlow;
import com.google.api.client.googleapis.auth.oauth2.GoogleClientSecrets;
import com.google.api.client.googleapis.javanet.GoogleNetHttpTransport;
import com.google.api.client.http.javanet.NetHttpTransport;
import com.google.api.client.json.JsonFactory;
import com.google.api.client.json.jackson2.JacksonFactory;
import com.google.api.client.util.store.FileDataStoreFactory;
import com.google.api.services.sheets.v4.Sheets;
import com.google.api.services.sheets.v4.SheetsScopes;
import com.google.api.services.sheets.v4.model.ValueRange;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.security.GeneralSecurityException;
import java.util.Collections;
import java.util.List;

public class SheetsQuickstart {
    private static final String APPLICATION_NAME = "Google Sheets API Java Quickstart";
    private static final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();
    private static final String TOKENS_DIRECTORY_PATH = "tokens";
```

We are highlighting the most important parts of the code so you can complete your own. Be sure make the modification in the full sample posted about mid-portion of the Journal Entries (**Step 5**)... We want to go through what we did to get it to work beyond the configuration so you are successful.

Hint: In our experience, what actually worked for us was the Application ID associated with the Project Name, as opposed to the name itself, thus SocialMedia did not work but the Id linked to Project

Name SocialMedia in our Google Cloud account worked...

*If you are running into issues, remember to switch to the Project Id...

The second thing that was needed was to pull in the JSON file to grab our Cloud project credentials. Without your JSON, you will have no chance getting a connection to the Cloud Application...

We showed you how to create your credentials in **Step 2** in this document. Go take a look there to get a sense where you might have downloaded it. You will also get a pretty good idea how to reference the file in your project if you look at **Step 6** of the documentation, when we discussed our Resources folder with you...

```
private static final String APPLICATION_NAME = "Google Sheets API Java Quickstart";
private static final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();
private static final String TOKENS_DIRECTORY_PATH = "tokens";

/**
 * Global instance of the scopes required by this quickstart.
 * If modifying these scopes, delete your previously saved tokens/ folder.
 */
private static final List<String> SCOPES = Collections.singletonList(SheetsScopes.SPREADSHEETS
private static final String CREDENTIALS_FILE_PATH = "/credentials.json";
```

Here is what it looks like in our version of the code

```
public class SheetsQuickstart {
    private static final String APPLICATION_NAME = "YourAppID";
    private static final JsonFactory JSON_FACTORY = JacksonFactory.getDefaultInstance();
    private static final String TOKENS_DIRECTORY_PATH = "tokens";

    /**
     * Global instance of the scopes required by this Quick start.
     * If modifying these scopes, delete your previously saved tokens/ folder.
     */
    private static final List<String> SCOPES = Collections.singletonList(SheetsScopes.SPREADSHEETS_READONLY);
    private static final String CREDENTIALS_FILE_PATH = "/Google_Auth/client_secret_62071019471-62e3feusimkedt
```

Hint: You can rename the downloaded JSON so it reads credentials.json, but in this case we kept ours as is. You are obviously looking at a portion of the file name for security reasons. The goal is you want to make sure it is place in your Resources folder (see **Step 6**), example Resources/Google_Auth in your project.

The next portion in the code that demands attention is the method that reads the actual Spreadsheet. If you did not create one yet, this is a good time to do so. You are going to need the ID of that Spreadsheet, which you can collect from the URL after you create the web file.

Take a look at this example from Google:


```

/**
 * Prints the names and majors of students in a sample spreadsheet:
 * https://docs.google.com/spreadsheets/d/1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs740gvE2upms/edit
 */
public static void main(String... args) throws IOException, GeneralSecurityException {
    // Build a new authorized API client service.
    final NetHttpTransport HTTP_TRANSPORT = GoogleNetHttpTransport.newTrustedTransport();
    final String spreadsheetId = "1BxiMVs0XRA5nFMdKvBdBZjgmUUqptlbs740gvE2upms";
    final String range = "Class Data!A2:E";
}

```

Journal Entry

2021.10.22.6.55.AM

Immediately after creating your first Sheet, you can either rename the SheetName in the tab below, which will likely be called Sheet1, or you can change to something more meaningful.

If our case, we left it as is...

The screenshot shows a Google Sheet interface. The title bar reads 'CompanyWorksheetsAddRow' with a star icon and a share icon. Below the title bar is a menu with 'File', 'Edit', 'View', 'Insert', 'Format', 'Data', and 'Tools'. The toolbar includes icons for undo, redo, print, and insert, along with a zoom level of 100%, currency symbols (\$, %), and decimal formatting options (.0, .00) and a number 123. The active cell is A1, containing the text 'Developer'. The formula bar shows 'fx' and 'Developer'. The spreadsheet grid has columns A, B, and C, and rows 1 through 15. The data in the grid is as follows:

	A	B	C
1	Developer	Code/Design	
2	Genevieve	Java	
3	Martin	CSharp	
4	Kyle	PHP	
5	Blanche	JavaScript	
6	Thierry	HTML5	
7	Max	CSS	
8	Vorbe	JSON	
9	Bhea	Ajax	
10	Benoit	jQuery	
11	Zoe	ASP	
12			
13			
14			
15			

At the bottom of the sheet, there is a tab labeled 'Sheet1' and a sheet name 'MyNewTestSheet'.

So, in our version of the code, we are reading Sheet1 in this fashion

```
// Build a new authorized API client service.
final NetHttpTransport HTTP_TRANSPORT = GoogleNetHttpTransport.newTrustedTransport();
final String spreadsheetId = "1fljLSW1LXT1NBWVpaTF0hYuRvvMt8hm2WnDlCyP3FNI";
final String range = "Sheet1!A2:B";
Sheets service = new Sheets.Builder(HTTP_TRANSPORT, JSON_FACTORY, getCredentials(HTTP_TRANSPORT))
    .setApplicationName(APPLICATION_NAME)
    .build();
ValueRange response = service.spreadsheets().values()
    .get(spreadsheetId, range)
    .execute();
List<List<Object>> values = response.getValues();
if (values == null || values.isEmpty()) {
    System.out.println("No data found.");
} else {
    System.out.println("Developer, Code/Design");
    for (List row : values) {
        // Print columns A and B, which correspond to indices 0 and 1.
        System.out.printf("%s, %s\n", row.get(0), row.get(1));
    }
}
```

Before you can actually do this however, you need to first be able to run your code to connect to your Google Cloud project to gain permission to reach the Spreadsheet in your Google Drive Account. The first time you run the code from NetBeans, your default browser will make an attempt to connect to your Google (existing) account.

Hint: Your Google account should be one that is, first of all valid, and it must be one that was used to connect to the your Google Drive in the first place, and the same account that was used to create the Google Cloud Console project. When your browser launches your application and your are prompted to sign, you must use that and only that account...

If your Cloud App has not yet been verified with Google, you will need to go through the verification process. An email similar to the following will be deposited to your inbox. After you have filled out the necessary information.

! Your Google Cloud Project **socialmedia-123456**

Action Needed

Hi,

Thank you for reaching out.

Currently, the project: **socialmedia-123456** is submitted for [brand](#) verification.

If you decide to skip the verification process, as stated in our previous response, this won't impact the existing functionality of your app, as long as the scopes used by your project don't change.

- Access to unverified [restricted](#) and/or [sensitive](#) scopes will trigger the [unverified app screen](#) and eventually the **unverified app user cap** will be enforced.

Verification has to occur prior for your project to being able to give Google account access to run the local Java Application.

Hint: If you recall, in Step 2, after creating your Credentials, you filled out some information through the OAuth Consent screen (excerpt below) which generated the above email to your Google account.

OAuth consent screen

Dököll Solutions, Inc. Java Cloud [EDIT APP](#)

Verification Status

! Verification in progress

The Trust and Safety team has received your form. They will reach out to you via your contact email if needed. The review process can take up to 4-6 weeks. Expect the first email from our Trust and Safety team within 3-5 days. Your last approved consent screen is still in use. [Learn more](#)

In the email, Google is telling you that you can move ahead and give your Cloud project permission to use your existing Google account to continue. That will then allow you to run your application based on the Non-restrictive scopes you setup in your Cloud project. When you make your first attempt to run your local App through NetBeans, your browser will pull up your Google sign in form to choose an account to use to run the application. Once you select the account, Google will ask permission to grant access to that account to continue... After your project grants permissions to your Google account, you will see the following in your browser.



Google hasn't verified this app

The app is requesting access to sensitive info in your Google Account. Until the developer (MyGoogleAccounts@gmail.com) verifies this app with Google, you shouldn't use it.

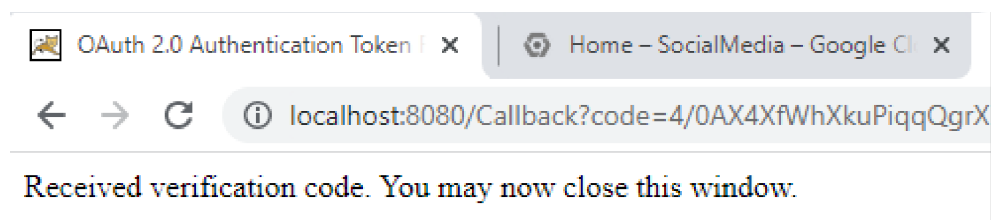
If you're the developer, submit a verification request to remove this screen. [Learn more](#)

[Advanced](#)

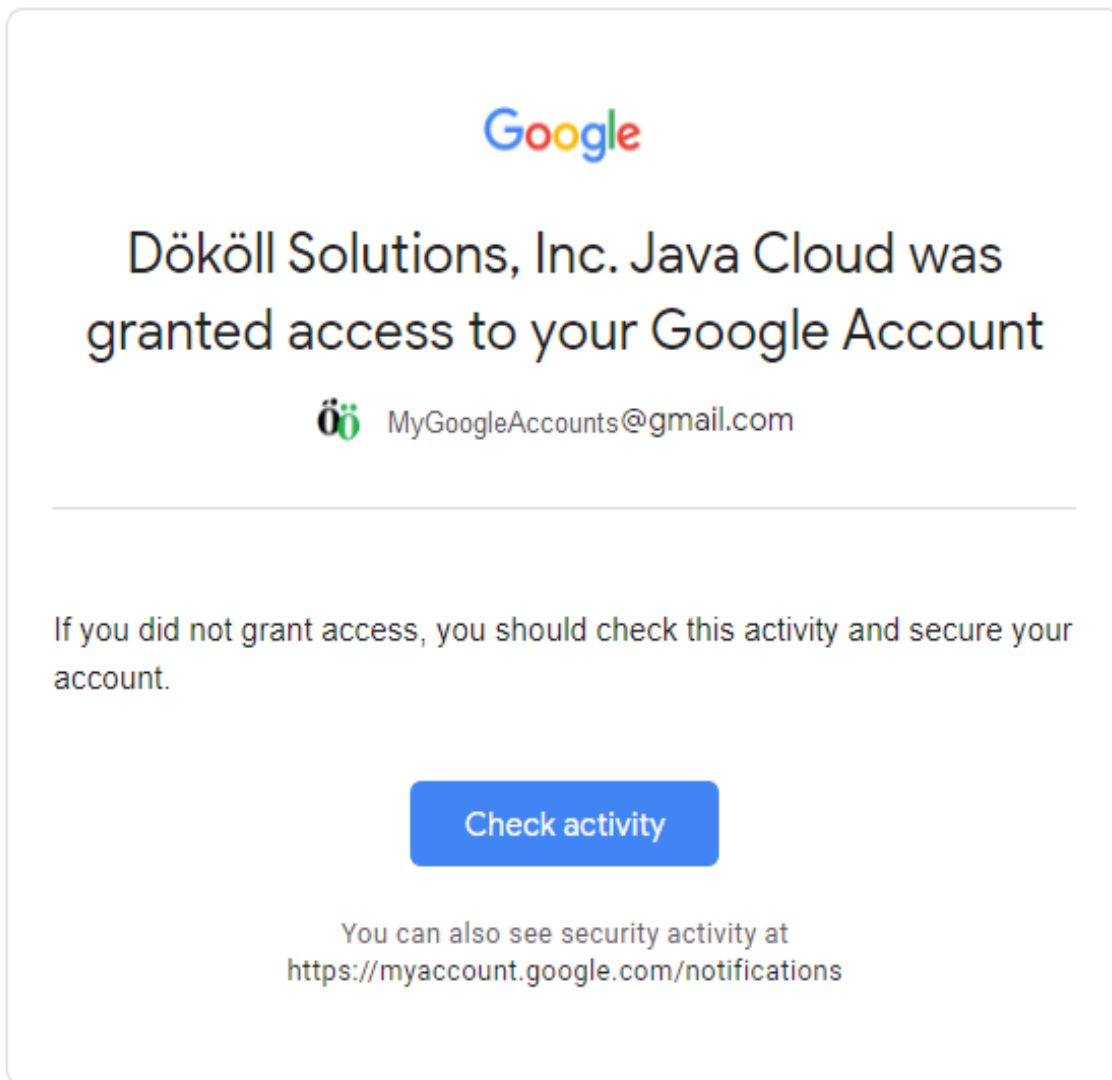
[BACK TO SAFETY](#)

The above looks worse than it is, Google is just making sure that you are aware a Local application from a localhost server is trying to connect to a Cloud project...

All you have to do here is go to Advanced, and continue running the application. Once you are able to continue, you will see below message in your browser. Google will receive the verification of your App through the browser as consent that the App can run from now on using that Google Account. This is a one-time action, you will never see the following message again.



If you check your email, you should see a message from Google stating the following



You received this email to let you know about important changes to your Google Account and services.

© 2021 Google LLC, 1600 Amphitheatre Parkway, Mountain View, CA 94043, USA

Now, you should go back to your Gradle application to see the results of your first launch, which should generate a list of data based on any range you set forth in your code. In our case, we wanted to return two column Developer and Code/Design, stated in the spreadsheet's **A2 : B** range.

```

final String range = "Sheet1!A2:B";
Sheets service = new Sheets.Builder(HTTP_TRANSPORT, JSON_FACTORY, getCredentials(HTTP_TRANSPORT))
    .setApplicationName(APPLICATION_NAME)
    .build();
ValueRange response = service.spreadsheets().values()
    .get(spreadsheetId, range)
    .execute();

```

And there you have it....

The screenshot shows a terminal window with the following content:

```

com.dokoll.solutions.inc.SocialMedia.SheetsQuickstart > main > range >
Terminal - localhost | Usages | Output - Run (SheetsQuickstart) x | Notifications
> JAVA_HOME="C:\Program Files\Java\jdk1.8.0_221"
> cd D:\APP_SOFTWARE_DOWNLOADS\Documents\NetBeansProjects\SocialMedia; .\gradlew.bat
Configuration on demand is an incubating feature.
> Task :compileJava
> Task :processResources UP-TO-DATE
> Task :classes

> Task :runSingle

Developer, Code/Design
Genevieve, Java
Martin, CSharp
Kyle, PHP
Blanche, JavaScript
Thierry, HTML5
Max, CSS
Vorbe, JSON
Bhea, Ajax
Benoit, jQuery
Zoe, ASP

BUILD SUCCESSFUL in 9s
3 actionable tasks: 2 executed, 1 up-to-date

```

Conclusion:

It took 38 pages to do this for you, but we hope it was worth it... your very own Google Sheets v4 Gradle Java application in NetBeans 12 from scratch. Our next trial will deal with a local IBM DB2, Oracle 12, and/or MySQL database values into Google Sheets... Do stay tuned!

Courtesy: Dököll Solutions, Inc.

Version: 2021.10.24.12.33.AM