Google Cloud Java-Client

Wichtige Punkte

- Es scheint wahrscheinlich, dass zur Verwendung von `com.google.cloud` in Java eine Java-Entwicklungsumgebung eingerichtet, Abhängigkeiten der Google Cloud Client-Bibliotheken hinzugefügt, die Authentifizierung konfiguriert und die Bibliotheken verwendet werden müssen, um mit Google Cloud-Diensten zu interagieren.
- Die Forschung legt nahe, dass die Authentifizierung automatisch auf Google Cloud-Plattformen oder manuell für die lokale Entwicklung mit Service Account-Anmeldeinformationen erfolgen kann.
- Die Beweise sprechen dafür, Maven oder Gradle zur Verwaltung von Abhängigkeiten zu verwenden, wobei Beispiele für Cloud Storage als häufigen Anwendungsfall bereitgestellt werden.

Einrichtung Ihrer Entwicklungsumgebung

Um loszulegen, stellen Sie sicher, dass Sie eine Java Development Kit (JDK) Version 8 oder höher installiert haben, zusammen mit einem Build-Tool wie Maven oder Gradle. Diese Tools helfen Ihnen, Ihre Projektabhängigkeiten und Build-Prozesse zu verwalten.

Hinzufügen von Abhängigkeiten

Fügen Sie die Abhängigkeiten der Google Cloud Client-Bibliotheken zu Ihrem Projekt hinzu. Für Maven fügen Sie die Bill of Materials (BOM) und spezifische Dienstbibliotheken in Ihre `pom.xml`-Datei ein. Zum Beispiel, um Cloud Storage zu verwenden:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.google.cloud</groupId>
            <artifactId>libraries-bom</artifactId>
            <version>latest_version</version>
            <type>pom</type>
            <scope.import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>com.google.cloud</groupId>
        <artifactId>google-cloud-storage</artifactId>
```

```
    </dependency>
</dependencies>
```

Ersetzen Sie “latest_version”durch die tatsächliche Version aus dem Google Cloud Java Client Libraries GitHub Repository.

Konfiguration der Authentifizierung

Die Authentifizierung wird oft automatisch gehandhabt, wenn Ihre Anwendung auf Google Cloud-Plattformen wie Compute Engine oder App Engine läuft. Für die lokale Entwicklung legen Sie die Umgebungsvariable `GOOGLE_APPLICATION_CREDENTIALS` fest, um auf eine JSON-Schlüsseldatei eines Service Accounts zu verweisen, oder konfigurieren Sie sie programmgesteuert.

Verwendung der Bibliotheken

Sobald alles eingerichtet ist, importieren Sie die erforderlichen Klassen, erstellen Sie ein Dienstobjekt und rufen Sie API-Aufrufe auf. Zum Beispiel, um Eimer in Cloud Storage aufzulisten:

```
import com.google.cloud.storage.*;
Storage storage = StorageOptions.getDefaultInstance().getService();
Page<Bucket> buckets = storage.list();
for (Bucket bucket : buckets.iterateAll()) {
    System.out.println(bucket.getName());
}
```

Ein unerwartetes Detail ist, dass die Bibliotheken verschiedene Google Cloud-Dienste unterstützen, jeder mit seinem eigenen Unterpaket unter `com.google.cloud`, wie `com.google.cloud.bigquery` für BigQuery, das umfangreiche Funktionalitäten über die Speicherung hinaus bietet.

Umfragehinweis: Umfassender Leitfaden zur Verwendung von `com.google.cloud` in Java

Dieser Hinweis bietet eine detaillierte Untersuchung der Verwendung der Google Cloud Java Client-Bibliotheken, die sich speziell auf das `com.google.cloud`-Paket konzentrieren, um mit Google Cloud-Diensten zu interagieren. Er erweitert die direkte Antwort, indem alle relevanten Details aus der Forschung aufgenommen werden, die für Klarheit und Tiefe organisiert sind, geeignet für Entwickler, die ein umfassendes Verständnis suchen.

Einführung in die Google Cloud Java Client-Bibliotheken Die Google Cloud Java Client-Bibliotheken, die unter dem `com.google.cloud`-Paket zugänglich sind, bieten idiomatische und intuitive Schnittstellen zur Interaktion mit Google Cloud-Diensten wie Cloud Storage, BigQuery und Compute Engine. Diese Bibliotheken sind so gestaltet, dass sie Boilerplate-Code reduzieren, niedrigschichtige Kommunikationsdetails handhaben und nahtlos in Java-Entwicklungspraktiken integriert werden. Sie sind besonders nützlich für die Erstellung cloud-nativer Anwendungen, die Tools wie Spring, Maven und Kubernetes nutzen, wie in der offiziellen Dokumentation hervorgehoben.

Einrichtung der Entwicklungsumgebung Um zu beginnen, wird eine Java Development Kit (JDK) Version 8 oder höher benötigt, um die Kompatibilität mit den Bibliotheken sicherzustellen. Die empfohlene Distribution ist Eclipse Temurin, eine Open-Source-Option, die Java SE TCK-zertifiziert ist, wie in den Setup-Anleitungen angegeben. Zusätzlich ist ein Build-Automatisierungstool wie Maven oder Gradle für die Verwaltung von Abhängigkeiten unerlässlich. Das Google Cloud CLI (`gcloud`) kann auch installiert werden, um mit Ressourcen von der Befehlszeile aus zu interagieren, was das Bereitstellen und Überwachen von Aufgaben erleichtert.

Verwaltung von Abhängigkeiten Die Verwaltung von Abhängigkeiten wird durch die Bill of Materials (BOM) von Google Cloud vereinfacht, die hilft, Versionen über mehrere Bibliotheken hinweg zu verwalten. Für Maven fügen Sie Folgendes zu Ihrer `pom.xml` hinzu:

```
<dependencyManagement>
    <dependencies>
        <dependency>
            <groupId>com.google.cloud</groupId>
            <artifactId>libraries-bom</artifactId>
            <version>latest_version</version>
            <type>pom</type>
            <scope.import</scope>
        </dependency>
    </dependencies>
</dependencyManagement>
<dependencies>
    <dependency>
        <groupId>com.google.cloud</groupId>
        <artifactId>google-cloud-storage</artifactId>
    </dependency>
</dependencies>
```

Für Gradle gelten ähnliche Konfigurationen, die die Versionskonsistenz sicherstellen. Die Versionsnummer sollte gegen das Google Cloud Java Client Libraries GitHub Repository überprüft werden, um die neuesten

Updates zu erhalten. Dieses Repository enthält auch unterstützte Plattformen wie x86_64, Mac OS X, Windows und Linux, weist jedoch auf Einschränkungen bei Android und Raspberry Pi hin.

Authentifizierungsmechanismen Die Authentifizierung ist ein kritischer Schritt, wobei die Optionen je nach Umgebung variieren. Auf Google Cloud-Plattformen wie Compute Engine, Kubernetes Engine oder App Engine werden Anmeldeinformationen automatisch abgeleitet, was den Prozess vereinfacht. Für andere Umgebungen, wie die lokale Entwicklung, stehen folgende Methoden zur Verfügung:

- **Service Account (Empfohlen):** Generieren Sie eine JSON-Schlüsseldatei aus der Google Cloud Console und legen Sie die Umgebungsvariable `GOOGLE_APPLICATION_CREDENTIALS` auf ihren Pfad fest. Alternativ können Sie sie programmgesteuert laden:

```
import com.google.auth.oauth2.GoogleCredentials;
import com.google.cloud.storage.*;
GoogleCredentials credentials = GoogleCredentials.fromStream(new FileInputStream("path/to/key.json"));
Storage storage = StorageOptions.newBuilder().setCredentials(credentials).build().getService();
```

- **Lokale Entwicklung/Testen:** Verwenden Sie das Google Cloud SDK mit `gcloud auth application-default login` für temporäre Anmeldeinformationen.
- **Bestehendes OAuth2-Token:** Verwenden Sie `GoogleCredentials.create(new AccessToken(accessToken, expirationTime))` für spezifische Anwendungsfälle.

Die Reihenfolge der Präferenz für die Angabe der Projekt-ID umfasst Dienstoptionen, die Umgebungsvariable `GOOGLE_CLOUD_PROJECT`, App Engine/Compute Engine, JSON-Anmeldeinformationen und Google Cloud SDK, wobei `ServiceOptions.getDefaultProjectId()` hilft, die Projekt-ID abzuleiten.

Verwendung der Client-Bibliotheken Sobald Abhängigkeiten und Authentifizierung eingerichtet sind, können Entwickler die Bibliotheken verwenden, um mit Google Cloud-Diensten zu interagieren. Jeder Dienst hat sein eigenes Unterpaket unter `com.google.cloud`, wie `com.google.cloud.storage` für Cloud Storage oder `com.google.cloud.bigquery` für BigQuery. Hier ist ein detailliertes Beispiel für Cloud Storage:

```
import com.google.cloud.storage.*;
Storage storage = StorageOptions.getDefaultInstance().getService();
Page<Bucket> buckets = storage.list();
for (Bucket bucket : buckets.iterateAll()) {
    System.out.println(bucket.getName());
}
```

Dieses Beispiel listet alle Eimer auf, aber die Bibliothek unterstützt Operationen wie das Hochladen von Objekten, das Herunterladen von Dateien und das Verwalten von Eimerrichtlinien. Für andere Dienste gelten ähnliche Muster, wobei detaillierte Methoden in den jeweiligen Javadocs verfügbar sind, wie denen für BigQuery unter Google Cloud Java Referenzdokumentation.

Fortgeschrittene Funktionen und Überlegungen Die Bibliotheken unterstützen fortgeschrittene Funktionen wie langlaufende Operationen (LROs) mit `OperationFuture`, mit konfigurierbaren Zeitüberschreitungen und Wiederholungsrichtlinien. Zum Beispiel enthält AI Platform (v3.24.0) Standardwerte wie eine anfängliche Wiederholungsverzögerung von 5000 ms, einen Multiplikator von 1,5, eine maximale Wiederholungsverzögerung von 45000 ms und eine Gesamtzeitüberschreitung von 300000 ms. Auch die Proxy-Konfiguration wird unterstützt, mit `https.proxyHost` und `https.proxyPort` für HTTPS/gRPC, und benutzerdefinierte Optionen für gRPC über `ProxyDetector`.

API-Schlüssel-Authentifizierung ist für einige APIs verfügbar und wird manuell über Header für gRPC oder REST festgelegt, wie in Beispielen für den Language-Dienst gezeigt. Das Testen wird mit bereitgestellten Tools erleichtert, die im TESTING.md des Repositorys detailliert beschrieben sind, und IDE-Plugins für IntelliJ und Eclipse verbessern die Entwicklung mit der Bibliothek.

Unterstützte Plattformen und Einschränkungen Die Bibliotheken sind mit verschiedenen Plattformen kompatibel, wobei HTTP-Clients überall funktionieren und gRPC-Clients auf x86_64, Mac OS X, Windows und Linux unterstützt werden. Sie werden jedoch nicht auf Android, Raspberry Pi oder App Engine Standard Java 7 unterstützt, außer für Datastore, Storage und BigQuery. Unterstützte Umgebungen umfassen Windows x86_64, Mac OS X x86_64, Linux x86_64, GCE, GKE, GAE Std J8, GAE Flex und Alpine Linux (Java 11+).

Ressourcen und weiterführende Literatur Für zusätzliche Anleitung bietet das Google Cloud Java Client Libraries GitHub Repository Codebeispiele, Beitragsrichtlinien und Fehlerbehebungsressourcen. Tutorials wie die auf Baeldung bieten praktische Beispiele, wie die Verwendung von Cloud Storage, während die offizielle Dokumentation unter Google Cloud for Developers breitere App-Entwicklungskonzepte abdeckt.

Tabelle: Wichtige Konfigurationsdetails

Aspekt	Details
Java-Version	Erfordert Java 8 oder höher.
Abhängigkeitsverwaltung	Verwenden Sie BOM für Versionskonsistenz, z. B. Maven `libraries-bom`.
Authentifizierungsmethoden	Automatisch auf GCE/GKE/GAE, manuell über `GOOGLE_APPLICATION_CREDENTIALS` oder programmgesteuert.
Unterstützte Plattformen	HTTP-Clients überall, gRPC auf x86_64, Mac OS X, Windows, Linux; nicht auf Android, Raspberry Pi.
Beispiel-Dienst	Cloud Storage: `com.google.cloud.storage`, Methoden zur Eimerauflistung, Objektverwaltung.

Dieser umfassende Leitfaden stellt sicher, dass Entwickler alle notwendigen Informationen haben, um `com.google.cloud` in Java effektiv zu verwenden, wobei Aufbau, Authentifizierung, Nutzung und fortgeschrittene Funktionen behandelt werden, mit Verweisen auf autoritative Quellen für weitere Erkundungen.

Wichtige Zitate

- Google Cloud Java Client Libraries GitHub Repository detaillierter Code und Setup
- Google Cloud-Dokumentation für Java-Entwicklung Überblick
- Baeldung-Tutorial zu Google Cloud Storage mit Java praktische Beispiele