

App Dev - Storing Application Data in Cloud Datastore - Java

GSP167



Overview

[Google Cloud Datastore](#) is a NoSQL document database built for automatic scaling, high performance, and ease of application development. In this lab, you use Datastore to store application data for an online Quiz application. You also configure the application to retrieve from Datastore and display the data in the quiz.

The Quiz application skeleton has already been written. You clone the repository that contains the skeleton using Google Cloud Shell, review the code using the Cloud Shell editor, and view it using the Cloud Shell web preview feature. You then modify the code that stores data to use Cloud Datastore.

Objectives

In this lab, you perform the following tasks:

- Harness Cloud Shell as your development environment
- Preview the application
- Update the application code to integrate Cloud Datastore

Setup and Requirements

Qwiklabs setup

Before you click the Start Lab button

Read these instructions. Labs are timed and you cannot pause them. The timer, which starts when you click **Start Lab**, shows how long Google Cloud resources will be made available to you.

This Qwiklabs hands-on lab lets you do the lab activities yourself in a real cloud environment, not in a simulation or demo environment. It does so by giving you new, temporary credentials that you use to sign in and access Google Cloud for the duration of the lab.

What you need

To complete this lab, you need:

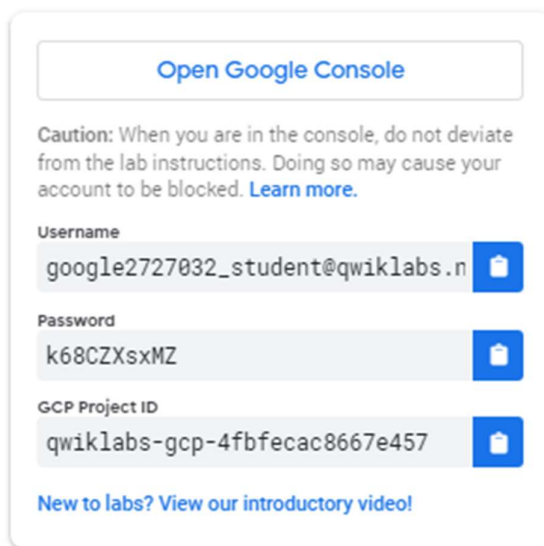
- Access to a standard internet browser (Chrome browser recommended).
- Time to complete the lab.

Note: If you already have your own personal Google Cloud account or project, do not use it for this lab.

Note: If you are using a Pixelbook, open an Incognito window to run this lab.

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is a panel populated with the temporary credentials that you must use for this lab.



[Open Google Console](#)

Caution: When you are in the console, do not deviate from the lab instructions. Doing so may cause your account to be blocked. [Learn more.](#)

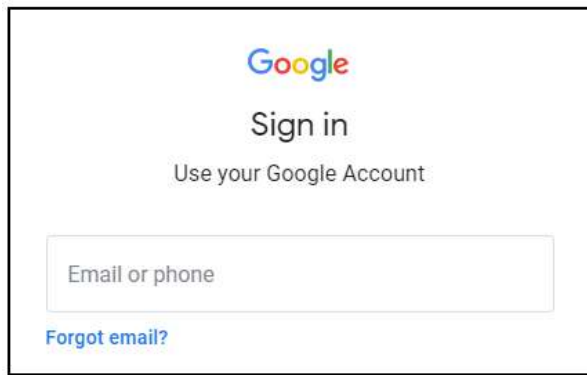
Username
google2727032_student@qwiklabs.n

Password
k68CZxsxMZ

GCP Project ID
qwiklabs-gcp-4fbfecac8667e457

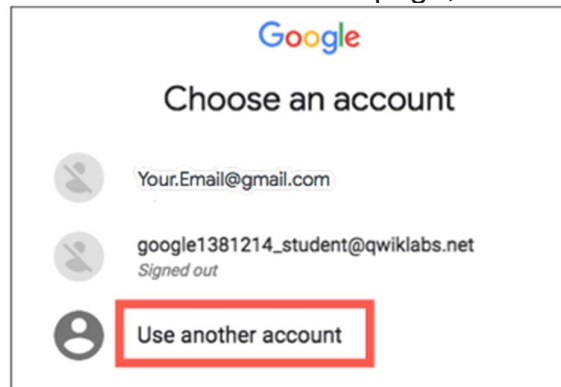
[New to labs? View our introductory video!](#)

2. Copy the username, and then click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.



Tip: Open the tabs in separate windows, side-by-side.

If you see the **Choose an account** page, click **Use Another**



Account.

3. In the **Sign in** page, paste the username that you copied from the Connection Details panel. Then copy and paste the password.

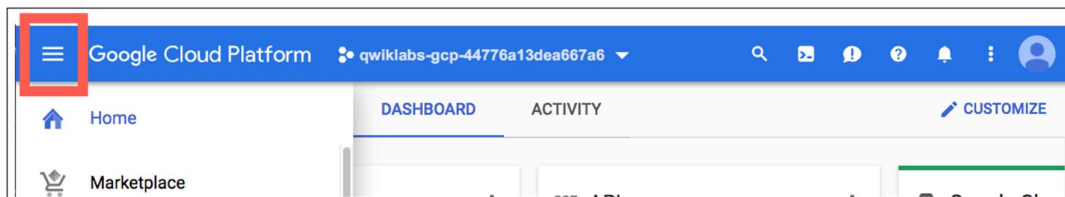
Important: You must use the credentials from the Connection Details panel. Do not use your Qwiklabs credentials. If you have your own Google Cloud account, do not use it for this lab (avoids incurring charges).

4. Click through the subsequent pages:

- Accept the terms and conditions.
- Do not add recovery options or two-factor authentication (because this is a temporary account).
- Do not sign up for free trials.

After a few moments, the Cloud Console opens in this tab.

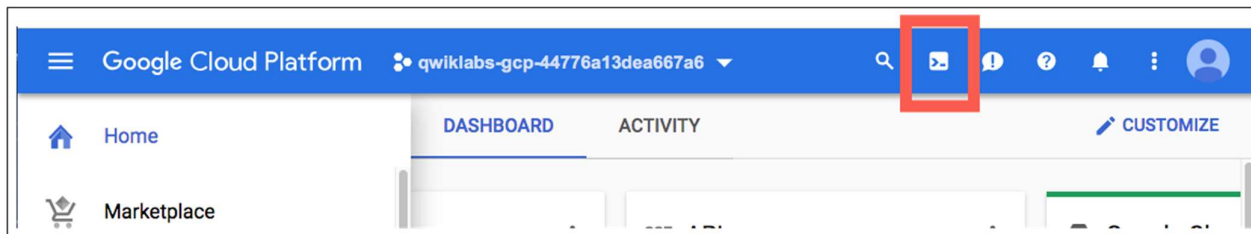
Note: You can view the menu with a list of Google Cloud Products and Services by clicking the **Navigation menu** at the top-left.



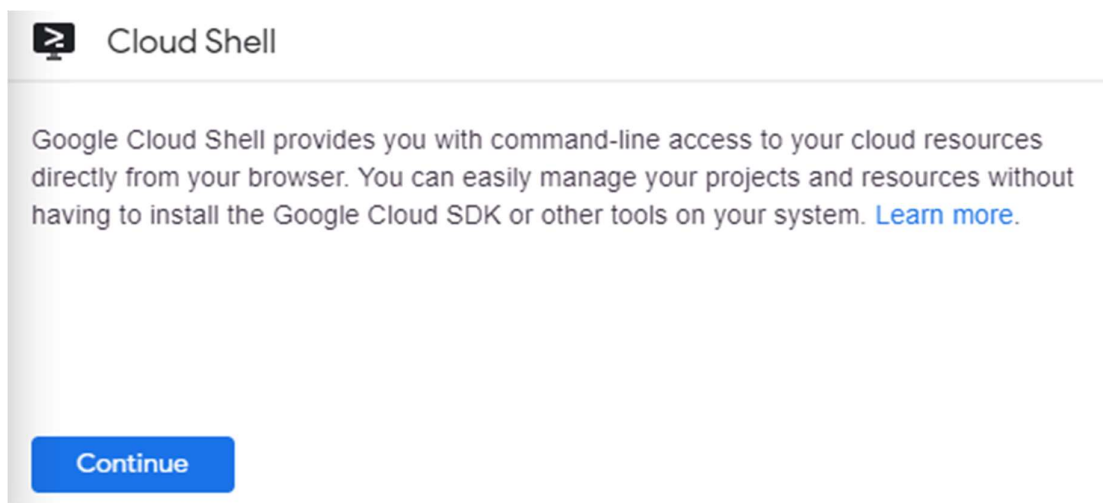
Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

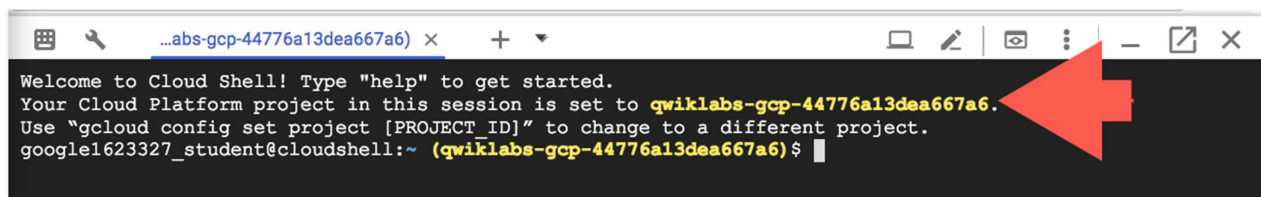
In the Cloud Console, in the top right toolbar, click the **Activate Cloud Shell** button.



Click **Continue**.



It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



`gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

You can list the active account name with this command:

```
gcloud auth list
```

(Output)

```
Credentialed accounts:  
- <myaccount>@<mydomain>.com (active)
```

(Example output)

```
Credentialed accounts:  
- google1623327 student@qwiklabs.net
```

You can list the project ID with this command:

```
gcloud config list project
```

(Output)

```
[core]  
project = <project ID>
```

(Example output)

```
[core]  
project = qwiklabs-gcp-44776a13dea667a6
```

For full documentation of `gcloud` see the [gcloud command-line tool overview](#).

Prepare the Quiz Application

In this section, you access Cloud Shell, clone the git repository containing the Quiz application, and run the application.

Clone source code in Cloud Shell

In Cloud Shell command line, enter the following command to clone the repository for the class:

```
git clone https://github.com/GoogleCloudPlatform/training-data-analyst
```

Configure and run the Quiz application

1. To navigate to the working directory, enter the following command:

```
2. cd ~/training-data-analyst/courses/developingapps/java/datastore/start
```

3. Create the environment variable `G_CLOUD_PROJECT` that references the Project ID:

```
4. export G_CLOUD_PROJECT=$DEVSHIELD_PROJECT_ID
```

5. Install the application dependencies:

```
6. mvn clean install
```

The installation may take a couple of minutes. It's complete when you see output similar to the following:

Example output:

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 38.805 s  
[INFO] Finished at: 2018-05-27T22:40:49-04:00  
[INFO] Final Memory: 35M/84M  
[INFO] -----
```

7. Run the application:

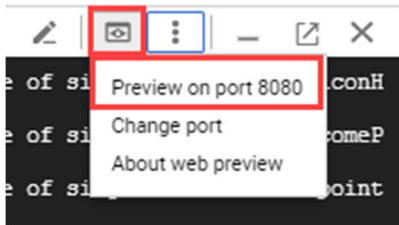
```
mvn spring-boot:run
```

The application is running When you see output similar to the following.

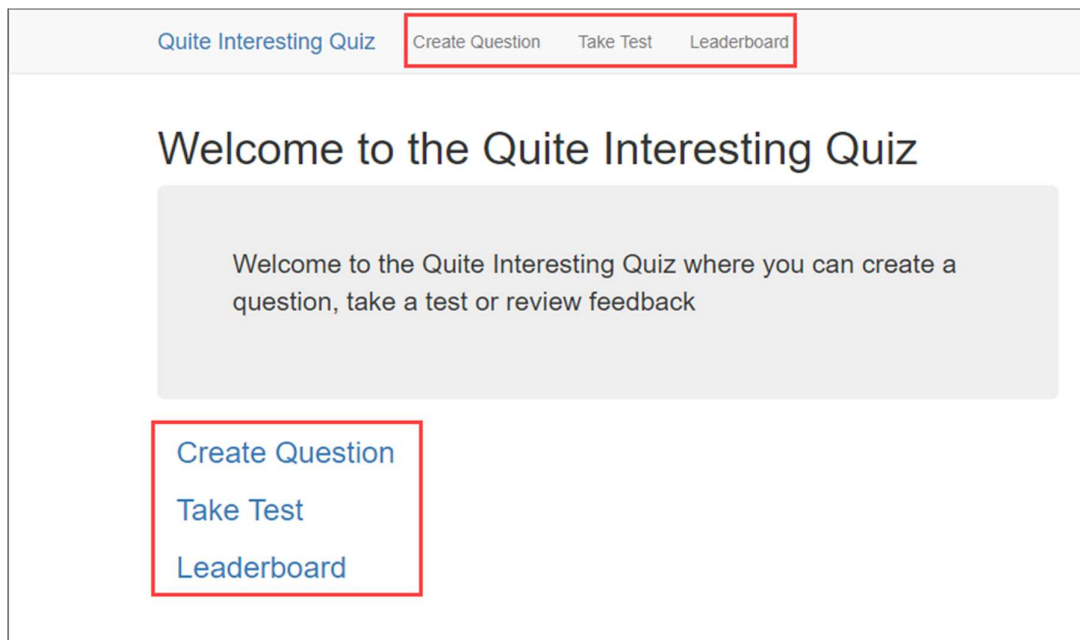
```
01:39:26.404 [restartedMain] INFO c.g.training.appdev.QuizApplication - Started QuizApplication in 12.628 seconds (JVM running for 14.157)
```

Review the Quiz application

1. In Cloud Shell, click **Web preview > Preview on port 8080** to preview the quiz application.



You should see the user interface for the web application. The three main parts to the application are **Create Question**, **Take Test**, and **Leaderboard**. Links to each are shown in the top navigation bar and on the page.



2. Click **Create Question**.

You should see a simple form that contains textboxes for the question and answers with radio buttons to select the correct answer. Quiz authors can add questions in this part of the application.

This part of the application is written as a server-side web application using the popular Java web application framework Spring Boot.

3. Click **Take Test**.

You are taken to a client application. Since you haven't made any questions, the Quite Interesting Quiz is blank.

4. In the top navigation bar, click **GCP**.

You should see a Dummy Title and Dummy Answers. The Dummy Title represents a sample question.

Quiz takers will answer questions in this part of the application.

This part of the application is written as a client-side web application using the popular JavaScript framework AngularJS. It receives JSON data from the server and uses JavaScript in the browser to display questions and collect answers.

To return to the server-side application, click on the **Quite Interesting Quiz** link in the navigation bar.

Examine the Quiz Application Code

In this section and throughout the lab you'll review the Quiz application code in a code editor. You can use the shell editors that are installed on Cloud Shell, such as nano or vim, or use the Cloud Shell code editor. This lab uses the Cloud Shell code editor.

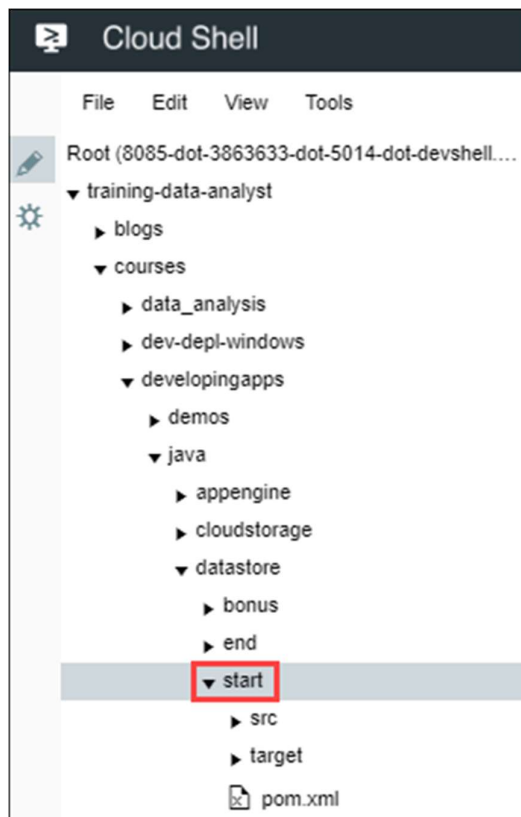
Launch the Cloud Shell text editor

From Cloud Shell, click on the **Open Editor** icon.

Review the code

The application is a standard Java application written using the popular Spring Boot application framework.

1. Navigate to the `/training-data-analyst/courses/developingapps/java/datastore/start` folder using the file browser panel on the left side of the editor.



This is the root folder for the application.

In the `datastore` folder, notice the `end` folder. The `end` folder contains the same files as the `start` folder, but each file in the `end` folder contains the complete code required to perform this lab.

2. From the `start` folder, navigate to the `src/main/java/com/google/training/appdev` folder using the file browser panel on the left side of the editor.

The paths for Java source code files are relative to the `appdev` folder.

Review the Spring Boot Web application

1. Select the `.../QuizApplication.java` file.

In this file, the class contains the entrypoint for the Spring Boot application.

2. Select the `.../services/gcp/domain/Question.java` file.

In this file, the domain class represents question data submitted in the question form and questions displayed when taking a quiz.

3. Select the `.../web/QuestionsController.java` file.

This file contains the handlers that display the form and collect form data posted by quiz authors in the web application.

In the `QuestionsController.java` file, find the handler that responds to HTTP POST requests for the `/questions/add` route.

Notice that the controller delegates the implementation of the handler to a service, `questionService`.

4. Navigate to the `/training-data-analyst/courses/developingapps/java/datastore/start/src/main/resources` folder using the file browser panel on the left side of the editor.

This folder contains templates for the web application user interface and static content displayed in the client-side web application.

5. Select the `templates` folder.

This folder contains the template for the web application user interface using the Thyme templating engine.

6. Select the `.../templates/new_question.html` file.

This file contains the template for the Create Question form. Notice how there is a select list to pick a quiz, textboxes where an author can enter the question and answers, and radio buttons to select the correct answer.

Return to the folder containing Java source code using the file browser panel on the left side of the editor. (Do you remember?

It's `start/src/main/java/com/google/training/appdev.`)

7. Select the `.../api/QuizEndpoint.java` file.

This file contains the handler that sends JSON data to students taking a test. Notice that the handlers also make use of the `questionService` object.

8. Select the `.../services/gcp/datastore/QuestionService.java` file.

This is the file where you write Datastore code to save and load quiz questions to and from Cloud Datastore. The web application and API use this class.

Add Entities to Cloud Datastore

In this section, you write code to save form data in Cloud Datastore.

Important: Update code within the sections marked as follows: `// TODO` `// END TODO` To maximize your learning, review the code, inline comments, and related API documentation. For more information see [Google Cloud Datastore Documentation](#)

Create an App Engine application to provision Cloud Datastore

From Cloud Shell, click on the **Open Terminal** icon.

Stop the application by pressing **Ctrl+C**.

To create an App Engine application in your project, execute the following command in Cloud Shell:

```
gcloud app create --region "us-central"
```

Note: Although You aren't yet using App Engine for your web application, Cloud Datastore requires you to create an App Engine application in your project.

Click *Check my progress* to verify the objective.

Import and use the Java Datastore package

From Cloud Shell, click on the **Open Editor** icon.

1. Open the `.../services/gcp/datastore/QuestionService.java` file in the Cloud Shell editor.
2. Write a star import for the `com.google.cloud.datastore.*` package.

```
// TODO: Import the com.google.cloud.datastore.* package
import com.google.cloud.datastore.*;

// END TODO
```

3. Declare a `Datastore` client object named `datastore` and initialize it.

```
// TODO: Create a Datastore client object, datastore
// The DatastoreOptions class has a getDefaultInstance()
// static method.
// Use the getService() method of the DatastoreOptions
// object to get the Datastore client
```

```

private Datastore datastore =
    DatastoreOptions.getDefaultInstance().getService();

// END TODO

```

After the updates, the first part of `QuestionService.java` is as follows:

```

package com.google.training.appdev.services.gcp.datastore;

// TODO: Import the com.google.cloud.datastore.* package
import com.google.cloud.datastore.*;

// END TODO

import com.google.training.appdev.services.gcp.domain.Question;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.springframework.stereotype.Service;

@Service
public class QuestionService {

    // TODO: Create a Datastore client object, datastore
    // The DatastoreOptions class has a getDefaultInstance()
    // static method.
    // Use the getService() method of the DatastoreOptions
    // object to get the Datastore client

    private Datastore datastore =
        DatastoreOptions.getDefaultInstance().getService();

// END TODO

```

Write code to create a Cloud Datastore entity

1. Declare a static final string named `ENTITY_KIND`, initialized with the value `"Question"`.

```

// TODO: Declare a static final String named kind
//The Datastore key is the equivalent of a primary key in a // relational
database.
// There are two main ways of writing a key:
// 1. Specify the kind, and let Datastore generate a unique //      numeric id
// 2. Specify the kind and a unique string id

private static final String ENTITY_KIND = "Question";

// END TODO

```

2. Create a `KeyFactory` for Question entities.

For more information on entities, see [Entities, Properties, and Keys](#).

```
// TODO: Create a KeyFactory for Question entities

private final KeyFactory keyFactory =
    datastore.newKeyFactory().setKind(ENTITY_KIND);

// END TODO
```

3. In the `createQuestion(Question question)` method, modify the method's return type to `Key`.

```
// TODO: Modify return type to Key

    public Key createQuestion(Question question) {

// END TODO
```

4. Declare a key with an allocated ID for the Question entity using the datastore client and Key Factory.

```
// TODO: Declare the entity key,
// with a Datastore allocated id

    Key key = datastore.allocateId(keyFactory.newKey());

// END TODO
```

For more information see [Class KeyFactory](#).

5. Declare an entity named `questionEntity`, and initialize it using an entity builder.

```
// TODO: Declare the entity object, with the key and data
// The entity's members are set using the Entity.Builder.
// This has a set method for property names and values
// Values are retrieved from the Domain object
Entity questionEntity = Entity.newBuilder(key)
    .set(Question.QUIZ, question.getQuiz())
    .set(Question.AUTHOR, question.getAuthor())
    .set(Question.TITLE, question.getTitle())
    .set(Question.ANSWER_ONE, question.getAnswerOne())
    .set(Question.ANSWER_TWO, question.getAnswerTwo())
    .set(Question.ANSWER_THREE, question.getAnswerThree())
    .set(Question.ANSWER_FOUR, question.getAnswerFour())
    .set(Question.CORRECT_ANSWER,
        question.getCorrectAnswer())
    .build();

// END TODO
```

6. Use the Datastore client object (`datastore`) to save the entity by calling its `put(questionEntity)` method.

```
// TODO: Save the entity

    datastore.put(questionEntity);

// END TODO
```

7. Modify the return statement to return the key for the entity.

```
// TODO: Return the key

    return key;
```

```
// END TODO
```

8. Save the file.

The following is the `QuestionService.java` content with all updates to this point.

```
// The createQuestion(Question question) method
// is passed a Question object using data from the form
// Extract the form data and add it to Datastore
// TODO: Modify return type to Key

    public Key createQuestion(Question question) {

// END TODO

// TODO: Declare the entity key,
// with a Datastore allocated id

    Key key = datastore.allocateId(keyFactory.newKey());

// END TODO

// TODO: Declare the entity object, with the key and data
// The entity's members are set using the Entity.Builder.
// This has a set method for property names and values
// Values are retrieved from the Domain object
    Entity questionEntity = Entity.newBuilder(key)
        .set(Question.QUIZ, question.getQuiz())
        .set(Question.AUTHOR, question.getAuthor())
        .set(Question.TITLE, question.getTitle())
        .set(Question.ANSWER_ONE, question.getAnswerOne())
        .set(Question.ANSWER_TWO, question.getAnswerTwo())
        .set(Question.ANSWER_THREE, question.getAnswerThree())
        .set(Question.ANSWER_FOUR, question.getAnswerFour())
        .set(Question.CORRECT_ANSWER,
            question.getCorrectAnswer())
        .build();

// END TODO

// TODO: Save the entity

    datastore.put(questionEntity);

// END TODO

// TODO: Return the key

    return key;

// END TODO
```

From Cloud Shell, click on the **Open Terminal** icon.

Run the application and create a Cloud Datastore entity

1. In Cloud Shell, run the application:

```
mvn spring-boot:run
```

When the application is running you'll see output similar to the following:

```
08:11:19.014 [restartedMain] INFO c.g.training.appdev.QuizApplication - Started QuizApplication in 10.401 seconds (JVM running for 11.28)
```

2. In Cloud Shell, click **Web preview > Preview on port 8080** to preview the quiz application.
3. Click **Create Question**, complete the form with the following values, and then click **Save**.

Form Field	Value
Author	Your Name
Quiz	Google Cloud
Title	Which company owns Google Cloud?
Answer 1	Amazon
Answer 2	Google (select the Answer 2 radio button!)
Answer 3	IBM
Answer 4	Microsoft

You are returned to the application home page.

The question you just made is now in DataReturn. In the Console, click **Navigation menu > Datastore > Entities** to see your new question!

Click *Check my progress* to verify the objective.

Run the application and create a Cloud Datastore entity

Check my progress

Query Cloud Datastore

In this section, you write code to retrieve entity data from Cloud Datastore.

Write code to retrieve Cloud Datastore entities

From Cloud Shell, click on the **Open Editor** icon.

1. Move to the `getAllQuestions(String quiz)` method in the `.../services/gcp/datastore/QuestionService.java` file, and remove the code for the existing Dummy questions.

```
// TODO: Remove this code

List<Question> questions = new ArrayList<>();
Question dummy = new Question.Builder()
    .withQuiz("gcp")
    .withAuthor("Dummy Author")
    .withTitle("Dummy Title")
    .withAnswerOne("Dummy Answer One")
    .withAnswerTwo("Dummy Answer Two")
    .withAnswerThree("Dummy Answer Three")
    .withAnswerFour("Dummy Answer Four")
    .withCorrectAnswer(1)
    .withId(-1)
    .build();
questions.add(dummy);

return questions;

// END TODO
```

2. Return the transformed results, using `buildQuestions(entities)` method to convert Datastore entities to domain objects.

```
// TODO: Return the transformed results
// Use the buildQuestions(entities) method to convert
// from Datastore entities to domain objects

return null;

// END TODO
```

3. In the `getAllQuestions(String quiz)` method, create a query object and initialize it with a query that retrieves Question entities for a specific quiz from Cloud Datastore.

```
public List<Question> getAllQuestions(String quiz){

// TODO: Create the query
// The Query class has a static newEntityQueryBuilder()
// method that allows you to specify the kind(s) of
// entities to be retrieved.
// The query can be customized to filter the Question
// entities for one quiz.
```

```

Query<Entity> query = Query.newEntityQueryBuilder()
    .setKind(ENTITY_KIND)
    .setFilter(StructuredQuery.PropertyFilter.eq(
        Question.QUIZ, quiz))
    .build();

// END TODO

```

For more information, see: [Datastore Queries](#).

4. Call the Datastore client object's `datastore.run(query)` method, and assign the result to entity iterator named `entities`.

```

// TODO: Execute the query
// The datastore.run(query) method returns an iterator
// for entities

Iterator<Entity> entities = datastore.run(query);

// END TODO

```

5. Uncomment the `buildQuestions(...)` and `entityToQuestion(...)` helper methods provided in the `QuestionService` class and use them to map the iterator to a list of question domain objects.

```

// TODO: Return the transformed results
// Use the buildQuestions(entities) method to convert
// from Datastore entities to domain objects

return buildQuestions(entities);

// END TODO

```

The following is the updated `QuestionService.java`:

```

public List<Question> getAllQuestions(String quiz){

    // TODO: Create the query
    // The Query class has a static newEntityQueryBuilder()
    // method that allows you to specify the kind(s) of
    // entities to be retrieved.
    // The query can be customized to filter the Question
    // entities for one quiz.

    Query<Entity> query = Query.newEntityQueryBuilder()
        .setKind(ENTITY_KIND)
        .setFilter(StructuredQuery.PropertyFilter.eq(
            Question.QUIZ, quiz))
        .build();

    // END TODO

    // TODO: Execute the query
    // The datastore.run(query) method returns an iterator
    // for entities

    Iterator<Entity> entities = datastore.run(query);

    // END TODO

    // TODO: Return the transformed results
    // Use the buildQuestions(entities) method to convert
    // from Datastore entities to domain objects

```

```

return buildQuestions(entities);

// END TODO
}

private List<Question> buildQuestions(Iterator<Entity> entities){
    List<Question> questions = new ArrayList<>();
    entities.forEachRemaining(entity-> questions.add(entityToQuestion(entity)));
    return questions;
}

private Question entityToQuestion(Entity entity){
    return new Question.Builder()
        .withQuiz(entity.getString(Question.QUIZ))
        .withAuthor(entity.getString(Question.AUTHOR))
        .withTitle(entity.getString(Question.TITLE))
        .withAnswerOne(entity.getString(Question.ANSWER_ONE))
        .withAnswerTwo(entity.getString(Question.ANSWER_TWO))
        .withAnswerThree(entity.getString(Question.ANSWER_THREE))
        .withAnswerFour(entity.getString(Question.ANSWER_FOUR))
        .withCorrectAnswer(entity.getLong(Question.CORRECT_ANSWER))
        .withId(entity.getKey().getId())
        .build();
}
}

```

Run the application and test the Cloud Datastore query

1. Save the `.../services/gcp/datastore/QuestionService.java` file, and then return to the Cloud Shell command.
2. From Cloud Shell, click on the **Open Terminal** icon. Stop the application by pressing **Ctrl+C**.
3. Start the application.
4. In Cloud Shell, click **Web preview > Preview on port 8080** to preview the quiz application.
5. Replace the query string at the end of the application's URL with `/api/quizzes/gcp`.

The URL is in the form: `https://8080-dot-####-dot-devshell.appspot.com/api/quizzes/gcp` You should see that JSON data has been returned to the client corresponding to the question you added in the web application!

6. Return to the application home page, and click **Take Test** and then click **GCP**.

You should see that the quiz question has been formatted inside the client-side web application!

Congratulations!

You learned how to provision a Compute Engine virtual machine and install software libraries for Java software development on Google Cloud.

Finish your Quest



This lab is part of the [Application Development - Java](#) and [Cloud Development](#) Quests. A Quest is a series of related labs that form a learning path. Completing this Quest earns you the badge above, to recognize your achievement. You can make your badge (or badges) public and link to them in your online resume or social media account. Enroll in this Quest and get immediate completion credit if you've taken this lab. See other available [Qwiklabs Quests](#).

Next steps / learn more

- Learn more about [Java on the Google Cloud](#).
- Get up to speed with Datastore, take [Datastore: Qwik Start](#), or check out [Google Cloud Datastore Documentation](#).

Google Cloud Training & Certification

...helps you make the most of Google Cloud technologies. [Our classes](#) include technical skills and best practices to help you get up to speed quickly and continue your learning journey. We offer fundamental to advanced level training, with on-demand, live, and virtual options to suit your busy schedule. [Certifications](#) help you validate and prove your skill and expertise in Google Cloud technologies.

Manual Last Updated August 13, 2020

Lab Last Tested July 22, 2020

Copyright 2021 Google LLC All rights reserved. Google and the Google logo are trademarks of Google LLC. All other company and product names may be trademarks of the respective companies with which they are associated.