Codelab: Entity Groups

Updated: April 28, 2015

This exercise leads you through an App Engine application that illustrates usage of Datastore Entity Groups.

Contents

- What You Will Learn
- What You Will Do
- Get Set Up
- Java Instructions
  - Eventual Consistency
  - Entity Groups and Ancestor Queries
- Python Instructions
  - Eventual Consistency
  - Entity Groups and Ancestor Queries
- Additional Resources
- Summary

What You Will Learn

You will learn:

- Eventual vs. Strong consistency in Datastore
- Entity Groups and usage of Ancestor Queries

What You Will Do

You will use an enhanced version of the App Engine application introduced in the Introduction to Datastore Hands on Exercise. You will uncomment certain sections of the code that illustrate datastore consistency, entity groups and ancestor queries.

Get Set Up

You will need to use existing project files that we have created for this codelab.

We have the Project ZIP files available for both Java and Python. Use any of them as given below:

- Java Datastore Entity Groups Project ZIP file
- Python Datastore Entity Groups Project ZIP file

You can also clone the projects directly from GitHub:

```
git clone https://github.com/GoogleCloudPlatformTraining/cp300-gae-entity-groups-java.git
```

OR

```
git clone https://github.com/GoogleCloudPlatformTraining/cp300-gae-entity-groups-python.git
```

In the Introduction to Datastore hands on exercise, we created the following: A Web application that allowed us to create and list Conference entities.

The application that we have provided you here extends it as follows:

1) In addition to creating the **Conference** entity, the code creates a number of Ticket entities. The number of Ticket entities created is equal to the **maxAttendees** defined for the **Conference** entity.
2) There is an additional HTML page, **showtickets.jsp (Java)** and **tickets.html (Python)** that displays the number of tickets for each Conference. The link to the above pages is available from the main page : **index.jsp (Java)** and **index.html (Python)**.

Java Instructions

We are going to look at the following 2 features:

- Eventual and Strong Consistency
- Entity Groups and Ancestor Queries

The next set of instructions will ask you to uncomment the lines of code that are marked TODO1 and TODO2 in specific files and you can see the concepts in action.

Eventual Consistency

Follow the steps:

1. In this exercise, examine the sections marked with TODO1 in **CreateConference.java .** Notice that we are creating **Ticket** entities for the number of attendees that the Conference can accomodate. The **Ticket** entity has the following attributes:
    a. ticketNumber : We are using a sequence number here.

   b. assignedTo : Assigned to a dummy user here.
   c. confTitle : This is the Conference title.
   d. confKeyString : This is the Entity Key of the Conference Entity for which the tickets are being created.
   e. available : This field indicates if the ticket is available or not. Currently set to 'available'
2. Observe that we have added an additional link in **index.jsp** that invokes **/showtickets.jsp**.
3. Review the TODO1 sections in **showtickets.jsp .** Here we simply filter on the confKeyString that was passed to it to retrieve the Ticket entities.
4. Run the application locally and visit the home page.
5. Visit the Admin console locally and delete any data that you have from previous runs.
6. Create a Conference with 20-50 maxAttendees.
7. Visit the Show Tickets link. You will observe that not all tickets are visible immediately. Keep Refreshing the Show Tickets page and eventually you will find that all the tickets i.e. count is equal to the **maxAttendees** for the Conference entity.

Entity Groups and Ancestor Queries

What you saw in the previous step was eventual consistency of the datastore. We will now introduce Entity Groups, where the Parent Entity will be the Conference and the Child Entity will be the Ticket. They belong to an Entity Group and while retrieving the Tickets , we will now observe strong consistency since the Datastore ensures that for an Entity Group, the indexes are updated and made consistent before returning the results.

Follow these steps:

1. For both **CreateConference.java** and **showtickets.jsp** , comment the sections for TODO1. Instead uncomment the sections for TODO2.
2. Inspect the TODO2 code in **CreateConference.java** file. While creating the **Ticket** entity, we are now specifying the ancestor i.e. Parent, which is the Conference Key. This creates an Entity Group.
3. Similarly, inspect the TODO2 code in **showtickets.jsp**. While querying the Ticket entities, we set the Ancestor Key i.e. Conference Key.
4. Run the Application locally and clear all data before you move ahead.
5. Create a Conference with 20-50 maxAttendees.
6. Visit the Show Tickets link. You will observe that **all tickets are visible immediately**. This demonstrates Strong Consistency.

Python Instructions

We are going to look at the following 2 features:

- Eventual and Strong Consistency
- Entity Groups and Ancestor Queries

The next set of instructions will ask you to uncomment the lines of code that are marked TODO1 and

TODO2 in specific files and you can see the concepts in action.

Eventual Consistency

Follow the steps as given below:

1. In this exercise, examine the sections marked with TODO1 in **datastore101.py .** Notice that we are creating **Ticket** entities for the number of attendees that the Conference can accomodate. The **Ticket** entity has the following attributes:
    a. ticket_number : We are using a sequence number here.
    b. assigned_to : Assigned to a dummy user here.
    c. conf_key_string : This is the Entity Key of the Conference Entity for which the tickets are being created.
    d. available : This field indicates if the ticket is available or not. Currently set to 'available'
2. Observe that we have added an additional link in **index.html** that invokes **/showtickets** which is mapped in **datastore101.py**.
3. Run the application locally and visit the home page.
4. Visit the Admin console locally and delete any data that you have from previous runs.
5. Create a Conference with 20-50 maxAttendees.
6. Visit the Show Tickets link. You will observe that not all tickets are visible immediately. Keep Refreshing the Show Tickets page and eventually you will find that all the tickets i.e. count is equal to the **maxAttendees** for the Conference entity.

Entity Groups and Ancestor Queries

What you saw in the previous step was eventual consistency of the datastore. We will now introduce Entity Groups, where the Parent Entity will be the Conference and the Child Entity will be the Ticket. They belong to an Entity Group and while retrieving the Tickets , we will now observe strong consistency since the Datastore ensures that for an Entity Group, the indexes are updated and made consistent before returning the results.

Follow the steps as given below:

1. Go to **datastore101.py** and comment the section for TODO1. Instead uncomment the section for TODO2.
2. Observe the TODO2 code. While creating the **Ticket** entity, we are now specifying the ancestor; i.e., Parent, which is the Conference Key. This creates an Entity Group.
3. Similarly, observe the **ShowTickets** handler in **datastore.py**. While querying the Ticket entities, we set the Ancestor Key i.e. Conference Key.
4. Run the Application locally and clear all data before you move ahead.
5. Create a Conference with 20-50 maxAttendees.
6. Visit the Show Tickets link. You will observe that **all tickets are visible immediately**. This demonstrates Strong Consistency.

Additional Resources

- Datastore Queries (Java) (Python)
- Datastore Indexes (Java) (Python)
- Configuring Indexes (Java) (Python)
- How Index Building Works
- Mastering the Datastore series of articles
- How entities are indexed and stored

Summary

In this exercise, you learned about eventual and strong consistency in the datastore using Entity Groups.