Batch apex salesforce developer guide

Continue

There are many declarative automation tools and features introduced by Salesforce in the recent past, which has greatly reduced the need for Apex code. But developers still use the humble Batch Apex.Batch Apex to process a large number of records. Because the code works asynchronously, records are processed within the platform. After calling the Apex package, records are processed in batches using the provided execution logic. Each batch is considered a separate Apex transaction. Batch Apex can handle large amounts of data, allows you to stay within the governor's limits, and allows you to roll back only for a failed batch of records. The Apex package is also used to recalculate the Apex managed exchange. Normal sharing changes in an organization are automatically recalculated. However, Apex exchange changes should be updated in bulk for all entries, which is best done with the Apex.A quick look at the overall syntax Batch Apexglobal class name implements Database.Batchable'lt;sObject'gt; Iterable) start (Database.BatchableContext bc) / // a collection of batches of records or objects that will be transferred to perform List of records) // The process of each batch of entries - the global void of finishing - what records will be considered throughout the process of retremation of any of the two types of objects - Database.QueryLocator or Iterized copter is the most commonly used object for extracting records without first processing, returns up to 50 million records. Iterable is useful when a lot of initial logic has to be applied to extracting a dataset before actually processing it. However, this is governed by restrictions on the total number of entries that are returned by SO'L requests. Processes recording packages from the Start Method (sObject records or sObject record list), up to 200 entries at a timeste 'Database.BatchableContext' - useful for information about the Apex Package The second parameter contains an area that contains the area 'Start'Ha doesn't control how to handle batchesPost processing - actions that need to be done when all batches are processed in a real-world scenario to understand that Batch Apex BetterCityworks processing is a parcel delivery company that has delivery stations in multiple states and delivers about 1,000 packages each day. The company uses the Salesforce authority to manage data about delivery agents and their delivery destinations. </sObject></sObject></sObject>agents are stored in a custom facility called Agent Service. A children's facility called Delivery is used to store supplies assigned to each agent. A secondary Salesforce organization is used to manage each agent's delivery assignments. It provides a REST-based service that returns new delivery data. At the end of each day, the status of all new deliveries must be updated in the primary organization of Salesforce. To meet this requirement, high-level steps will be: Connecting the external system (secondary Salesforce org) to the primary salesforce org. Write a Batch Apex class that calls the external system to get the data you need and update the primary organization. Batch Apex class schedule to run every night. This case of use requires the use of the Apex package with some special features and interfaces, namely, Database.Stateful interface, Database.AllowCallouts interface, and Schedulable.Connecting external system for primary Salesforce orgGetting data from the external system (secondary org)Since the secondary org provides a REST-based service that returns delivery data, it can be used by the primary org to get the necessary data at the end of each day. Note: For convenience and simplicity, the external system in the script is only Salesforce. The next steps show an easy way to connect to Salesforce. Setting up an external system (secondary org)Connected App: The first step is to allow the primary organization to connect to a secondary organization. You can do this by creating a connected app in a secondary organization by switching to customization of the App Manager's New Connected App. Create a new connected app, an automatically generated consumer key and a secret in a connected app, you can use in your primary organization to connect to a secondary organization. Setting up the primary orgAn 'Authentication Provider' and 'Named Credentials' can be configured in the primary organization to connect to a secondary organization. The authentication provider must use a consumer key and a secret derived from a connected application. Create a named A Batch Apex class, as the following can be created to make a call on a secondary org, get data, and update records in the primary org.global class BatchApexCalloutFOF implements Database.Batchable'lt;sObject'gt;, Database.PermitsCallouts, Database.Stateful-Global List'lt;ExternalIDs'lt;String'gt;(new list); Global'lt;String,String,String,Gt; UpdateDeliveryStatus map - new map qlt,String,String'gt;;); Line a; String b; String Request;/ConstructGloor BatchbalApexCall ();/Установить тайм-аут до 1 минуты, чтобы избежать ошибки времени чтения (только если он появляется)request.setTimeout (60000);request.setEndpoint ('callout:Apex_Rest_Services_Test/services/apexrest/retrieveDeliveries');request.setMethod('GET'); Ответ httpResponse - http.send (запрос);в то время как</String,String> </String,String> </String> </sObject> </sObject> No 302) request.setEndpoint (answer.getHeader);response new Http ().send (request); If the request is successful, clarify JSON's response. System.debug (response.getBody();/JSON to get only external ID on the list, map to the corresponding delivery statusJSONParser parser - JSON.createParser (response.getBody); while (parser.nextToken() JSONToken.FIELD_NAME! ExternalIDs.add(a); still if (parser.getText)) - 'Status__c')) parser.nextToken ();b - parser.getText (); DeliveryStatus.put (a,b); The question is if the data comes in during runtimeSystem.debug ('JSON External IDs list: '- ExternalIds); System.debug ('Map:' - UpdateDeliveryStatus); Global Launch Database.querylocator (Database.BatchableContext BC) .system.debug ('Inside the Launch Statement');/Creating a query with a list of external identifiers to limit the scope of coverage: 'Choose id, name, External_ID__c, Service_Agent__c, Status__c from Delivery__c where External_ID__c in : ExternalIds'; Global Void Performance (Database.BatchableContext BC, Delivery__c):Delivery__c Delivery__c. System.debug ('Map:' - UpdateDeliveryStatus);/Loop via entries in the area, package-wisefor (Delivery__c s: scope) System.debug ('Current entry in the cycle' s.External_ID__c); s.Status__c - UpdateDeliveryStatus.get (s.External_ID__c);d elivs.add(s); Update delivs; Global Void Finish (Database.BatchableContext BC) admin@fof Messages. SingleEmailMessage Mail new Messaging.SingleEmailMessage ();mail.setToAddresses (new line)lyTo ('admin@fof.com'); mail.setSenderDisplayName ('Batch Process';'mail.setSubject;'Delivery statuses updated successfully'); mail.setPlainTextBody ('Batch Process.'; Messaging.sendEmail (new messenger. SingleEmailMessage Note: This code does not handle exceptions. The general idea behind using the Apex package code here is to call REST into an external system so that 'GET' the data that is needed in the form of a JSON response, disassemble the answer, store external documents and delivery statuses in variables, get records to be updated, and then update those records in batches. Here's an example of an external response: attributes: type: Delivery__c,url: /services/data/v48.0/sobjects/Delivery__c/a032v0006DbH3oAAF. The question:Id: a032v0006DbH3oAAF,Name: D-0000002,Service_Agent__c: a022v000001yJFjzAAG,Status__c Cancelled - attributes: type: Delivery__c,url: /services/data/v48.0/sobjects/Delivery__c/a032v0006DbH3aAAF.,Id a032v0006DbH3jAAF.Name: D-00000001,Service_Agent__c: a022v0001yJFjzAG, Status__c: qlt;/Delivery__c/gt Delivery__c; Delivery__cType: Delivery__c,url: /services/data/v48.0/sobjects/Delivery__c/a032v00006DbH3tA AAF.Name D-0000003,Service_Agent__c: a022v00001yJFk4AAG,Status__c: The purpose of the code here is to update only records for which there is an update from the external system. Because the system will already have a large set of records, you only need to process new shipments, i.e. about 1,000 records per day. Some important considerations are: Because the code is calling, the 'Database.PermitsCallouts' interface needs to be implemented. Before the Start method defines the process area, the call is made in the constructor's method. Because the query is to obtain data from an external system, the http 'GET' method is used here. The answer is in JSON format. After the answer is received, external identifiers and relevant delivery statuses are collected using JSON Parser methods. This collection of external identifiers is used in the Start method to only return records that need to be updated. The list and map identified in the designer are by default returned outside the designer because the package processing has no state. To maintain the status (value) of the list and maps of various transactions, the class implements the Database.Stateful interface. The query is built in a start-up method using the ExternalIds list and returned as a queryLocator. The execution method handles entries received from the request in batches. The map makes it easy to update the relevant record, thereby reducing the number of lines of code. The finishing method is used to provide an update to the administrator (or any other relevant user) via email after updating all batches of records. Referring to the Apex package class for the script, the Apex package class may be scheduled to run at a certain time, especially when no more updates are expected. The apex package job code is placed in another Apex class that implements a 'schedulable' interface. This class can be scheduled to run with Apex Scheduler at Setup.Code to schedule the Apex package: ScheduleDeliveryUpdateFOF Global Class Implements Schedulable'global void execution (SchedulableContext sc) - BatchApexCalloutFOF quickupdate - new BatchApexCalloutFOF(); Database.executeBatch (fast, 100); The Batch Apex class schedule, however, has other ways of calling the Apex Class:1 package. The Apex package class can be called using the 'Database.executeBatch' method in the Apex Anonymous window in the developer's console. BatchApexCallOutFOF quickupdate - new BatchApexCallOutFOF ();D atabase.executeBatch (fast,100); where 100 limits the number of entries per batch (area parameter) Sample log from developer console 2. Apex package can triggered by the Apex trigger. But the trigger doesn't have to add more batch tasks than the limit. How can I test the Apex package? The Apex package class can be tested simply by inserting some sample records into the test class and processing them using a packet class. In a test class, only one perform method is possible. Because Apex testing methods don't support a call to an external system, you can create a call layout to dent the actual call. How can I control Apex package jobs?1. The Apex Jobs page in the setup can be used to monitor the apex package jobs. 2. A link to a new package job page can be used. You can click on the More Information section to see more information. More information about Batch Jobs 3. Batch Apex jobs can also be controlled by software. A copy of 'Database.BatchableContext' can be used in a combination of 'getJobID' to obtain the current batch job ID in the packet class. This can be used in conjunction with 'AsyncApexJob' for more information about batch work. Database.executeBatch returns the current job ID when calling apex.global invalid finish (Database.BatchableContext bc) AsyncApexJob a SELECT Id, Status, NumberOfErrors, JobItemsProcessed, TotalJobItems, CreatedBy.Email FROM AsyncApexJob WHERE E.B.B.B.ID :BC.get-a-fact AsyncApexJob apexj - SELECT Id, Status, JobItems Processed, TotalJobItems, NumberOfErrors from AsyncApexJob WHERE ID: batchprocessid; Batch Apex jobs can be chained to each other; one job can start another job. This is possible by calling Database.executeBatch or System.scheduleBatch in the method of finishing the current apex batch class. The new task of the package will begin after the end of the existing party job. This is a good way to work with large amounts of data. Best practices to follow when writing a classUse package of effective SO'L requests to minimize delays and stay within the governor's limits. Complete Batch Apex tasks as quickly as possible, minimizing call times. Use the Apex package only when you need to process more than 200 entries at a time (consider the need to process packages against regular Apex code) Consider the number of batches that will queue up if any automation is involved in calling the package class, such as trigger/process/flow process - there can only be five jobs in the queue or active batches at a time. Identify all methods in the class as global or public. Don't call or announce future methods in the Apex class. For more information on Batch Apex What certification are you studying Focus on power currently provides practice exams and study guides for nine certification certifications