Error 'Apex heap size too large'

Knowledge Article Number 000321537

Description

Salesforce enforces an Apex Heap Size Limit of 6MB for synchronous transactions and 12MB for asynchronous transactions.

The "Apex heap size too large" error occurs when too much data is being stored in memory during processing. The limit depends on the type of execution (E.g. synchronous vs asynchronous calls) and current value can be found in the **Apex Developer's Guide**.

It is important to review your code and follow best practices to ensure that the heap limit does not exceed the maximum.

Here is an example of Apex code that will exceed the heap size limit.

```
@isTest
private class heapCheckIssue{
    static testMethod void myTest(){
        String tStr = 'aaaaa bbbbb ccccc ddddd eeeeee fffff ggggg 11111
        List<String> baseList = tStr.split(' ');
        List<String> bigList = baseList;
        Map<integer, List<String>> SampleMap = new Map<integer, List<St
        SampleMap.put(1, bigList);

        for (integer i=0; i<50; i++) {
            List<String> tempList = new List<String>();
            tempList = SampleMap.get(1);
            bigList.addAll(tempList);
        }
        system.debug('FINAL LIST SIZE IS '+bigList.size());
    }
}
```

This example shows an incorrect use of collections.
The List baseList, the value of SampleMap and the List tempList are pointing to the same memory address. As a result, the heap size doubles with each iteration of the loop.

Resolution

This example shows a similar algorithm that will not exceed the heap size:

```
@isTest

private class heapCheckSuccess{
  static testMethod void myTest(){
    String tStr = 'aaaaa bbbbb ccccc ddddd eeeeee fffff ggggg 11111 222
    List<String> baseList = tStr.split(' ');
    Map<integer, List<String>> Sample = new Map<integer, List<String>>(
    List<String> bigList = baseList;

    Sample.put(1, bigList);
    List<string> myList = new list<string>(); //Declare a new list

    for (integer i=0; i<50; i++) {
      List<String> tempList = new List<String>();
      tempList = Sample.get(1);
      system.debug('templist: ' + tempList.size());
      system.debug(' bigList: ' + bigList.size());

      myList.addall(tempList); //original code is  bigList.addall(tempL
    }

    system.debug('FINAL LIST SIZE OF bigList IS '+ bigList.size());
    system.debug('myList IS '+mylist.size());
  }
}
```

In the above example, the heap size does not get too large because we are using a new List to store the value of tempList in each iteration and then appending the list, instead of using bigList variable.

Best practices for running within the Apex heap size

- Don't use class level variables to store a large amounts of data.
- Utilize SOQL For Loops to iterate and process data from large queries.
- Construct methods and loops that allow variables to go out of scope as soon as they are no longer needed.

In the below sample, while we could be processing 50000 items, the largest heap would contain a search term, a query, a list of 200 accounts, and a list of the results. This is relatively small in the scope of everything that's being processed.

```
public with sharing class myClass{

private String myString;

    public myClass(String searchString) {
        myString = searchString;
    }

    private string getQueryString() {
        return 'SELECT Id, Name FROM Account LIMIT 50000';
    }

    public List<Account> getSearchResults() {
        List<Account> searchResults = new List<Account>();
        for(List<Account> accts : Database.Query(getQueryString())) {
            // Each loop processes 200 items
            for(Account a : accts) {
                if (a.Name != null && a.Name.contains(myString)) {
                    searchResults.add(a);
                }
            }
        }
        return searchResults;
    }
}
```