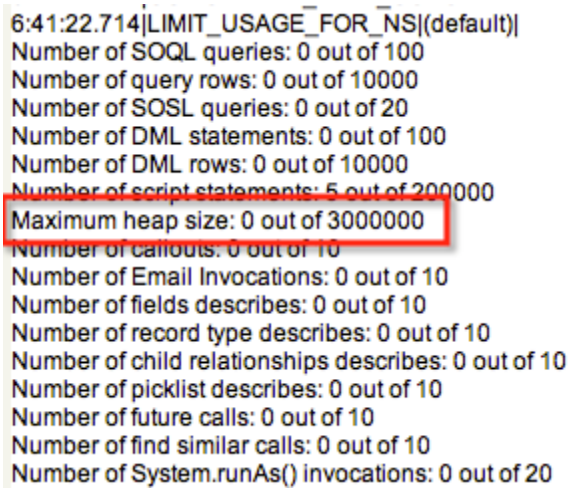
Watch the Heap
When your scripts run you can view the heap size in the debug logs. If you notice your heap approaching the limit, you will need to investigate why and try to refactor your code accordingly.



Use the Transient Keyword
Try using the "Transient" keyword with variables in your controllers and extensions. The transient keyword is used to declare instance variables that cannot be saved, and shouldn't be transmitted as part of the view state for a Visualforce page.

Use Limit Methods
Use heap limits methods in your Apex code to monitor/manage the heap during execution.

- **Limits.getHeapSize()** - Returns the approximate amount of memory (in bytes) that has been used for the heap in the current context.
- **Limits.getLimitHeapSize()** - Returns the total amount of memory (in bytes) that can be used for the heap in the current context.

```
// check the heap size at runtime
if (Limits.getHeapSize > 275000) {
    // implement logic to reduce
}
```

One strategy to reduce heap size during runtime is to remove items from the collection as you iterate over it.

Put Your Objects on a Diet
If the objects in your collection contain related objects (i.e., Account objects with a number of related Contacts for each) make sure the related objects only contain the fields that are actually needed by your script.

Use SOQL For Loops
SOQL "for" loops differ from standard SOQL statements because of the method they use to retrieve sObjects. To avoid heap size limits, developers should always use a SOQL "for" loop to process query results that return many records. SOQL "for" loops retrieve all sObjects in a query and process multiple batches of records through the use of internal calls to query and queryMore.

```
for (List<Contact> contacts : [select id, firstname, lastname
  from contact where billingState = 'NY']) {


  // implement your code to modify records in this batch


  // commit this batch of 200 records
  update contacts;
}
```

Note: There used to be a strategy for dealing with heap intensive scripts by moving them asynchronous, @Future methods. However, since the heap limits were increased in Summer '10 this is no longer a reason to simply use @Future method as the limits are the same.

Always use WHERE condition and Query less fields to reduce the HEAP Size