# TaskStatus

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Describes the current status of a task. This is

\* not intended to be a comprehensive piece of data.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

枚举：

**enum** Phase{*STARTING*, *MAP*, *SHUFFLE*, *SORT*, *REDUCE*, *CLEANUP*}

**enum** State{*RUNNING*, *SUCCEEDED*, *FAILED*, *UNASSIGNED*, *KILLED*, *COMMIT\_PENDING*, *FAILED\_UNCLEAN*, *KILLED\_UNCLEAN*}

属性：

1. **private** **final** TaskAttemptID taskid;
2. **private** **float** progress;
3. **private** **volatile** State runState;
4. **private** String stateString;
5. **private** **int** numSlots;
6. **private** **long** outputSize = -1L;
7. **private** **volatile** Phase phase = Phase.*STARTING*;
8. **private** Counters counters;
9. **private** **boolean** includeCounters;
10. **private** SortedRanges.Range nextRecordRange = **new** SortedRanges.Range();

方法：

1. **void** setPhase(Phase phase)
   1. 如果phase和原先的oldPhase不同，则要检查：
      1. 如果phase == TaskStatus.Phase.*SORT*，则表示*SHUFFLE*结束了，要设置setShuffleFinishTime(System.*currentTimeMillis*());
      2. 如果phase == TaskStatus.Phase.*REDUCE*，则表示*SORT*结束了，设置setSortFinishTime(System.*currentTimeMillis*());
   2. 最后设置**this**.phase = phase;
2. **boolean** inTaskCleanupPhase()

如果阶段为*CLEANUP*，且状态为*FAILED\_UNCLEAN*或*KILLED\_UNCLEAN*则返回true

## MapTaskStatus

继承自TaskStatus，getIsMap方法返回true，没有其他区别

## ReduceTaskStatus

继承自TaskStatus，

属性：

1. **private** **long** shuffleFinishTime;
2. **private** **long** sortFinishTime;
3. **private** List<TaskAttemptID> failedFetchTasks = **new** ArrayList<TaskAttemptID>(1);

注意，shuffleFinishTime，sortFinishTime和父类的finishTime三个属性存在先后关系，如果要设置finishTime，但是shuffleFinishTime和sortFinishTime都还没设置，则直接设置shuffleFinishTime，sortFinishTime为finishTime。同样shuffleFinishTime和sortFinishTime也存在先后关系