Extending Jobs to speed up Eclipse

Thirumala Reddy Mutchukota, Google Inc.

Overview

- Motivation
- Background
- JobGroup API
- File Search Speedup
- Job.join() with timeout

Motivation

Many single threaded operations

- File Search
- Java project classpath resolution
- Populating JavaModel caches
- Other search operations (e.g., find all references)
- Workspace refresh
- ???

How can we fix that?

- Break a large task into small pieces and run them in parallel.

- Need a simple way to group a set of Eclipse Jobs that are responsible for pieces of the same large task.

Background

Jobs & Job Families

- Job - to execute a task asynchronously.
- Job Family - to group set of jobs.
- Job.belongsTo(Object family).
- Any Object can act as a family object.
- A job can belongs to any number of families at the same time.
- No way to limit the number of jobs executed concurrently.

Job Groups

An efficient way to group a set of Jobs with …

- Throttling
- Join
- Cancel
- Combined progress reporting
- Combined error reporting

Job Groups (contd...)

Throttling

- Controls the number of jobs to be executed in parallel.
- No throttling when maxThreads is set to Zero.

```
public JobGroup(String name, int maxThreads, int seedJobsCount);
```

Job Groups (contd...)

Join

- Waits until all jobs belonging to the job group have finished.
- or the given timeout has expired.
- or the given monitor is cancelled.
- also used to get the combined progress reporting.

Cancel

- Cancels all jobs belonging to the job group.

```
public boolean join(long timeoutMillis, IProgressMonitor monitor);
public void cancel();
```

Job Groups (contd...)

Custom cancellation strategy

- Job group is cancelled when any of the jobs belongs to the group is failed.
- That behaviour can changed by the users.

```
protected boolean shouldCancel(IStatus lastCompletedJobResult,
    int numberOfFailedJobs, int numberOfCancelledJobs);
```

Job Groups (contd...)

Custom error reporting

- Individual job failure notifications are suppressed.
- A single multistatus with all the job failure statuses is reported at the end of job group completion.
- That behaviour can be changed by the users.

```
protected MultiStatus computeGroupResult(List<IStatus> jobResults);
public MultiStatus getResult();
```

Job Groups (contd...)

States

- NONE
- ACTIVE
- CANCELLING

```
public JobGroup(String name, int maxThreads, int seedJobsCount);
public int getState();
public List<Job> getActiveJobs();
```

Job Groups (contd...)

Deadlock Avoidance

- Joining another job from the same job group is not allowed.

For more information on Job groups - http://bugs.eclipse.org/432049

API Changes

A newly added *JobGroup* class with …

```
public JobGroup(String name, int maxThreads, int seedJobsCount);
public MultiStatus getResult();
public int getState();
public List<Job> getActiveJobs();
public void cancel();
public boolean join(long timeoutMillis, IProgressMonitor monitor);
protected boolean shouldCancel(IStatus lastCompletedJobResult,
      int numberOfFailedJobs, int numberOfCancelledJobs);
protected MultiStatus computeGroupResult(List<IStatus> jobResults);
```

API Changes (contd…)

Addition to the *Job* class

```
public void setJobGroup(JobGroup jobGroup);
public JobGroup getJobGroup();
```

Example

```
JobGroup group = new JobGroup("Directory Digger", 10, 1);
List<File> filesCollector = new ArrayList<>();
DirectoryDiggerJob job = new DirectoryDiggerJob(path, filesCollector);
job.setJobGroup(group);
job.setSystem(true);
job.schedule();

boolean success = group.join(10000, monitor);
if (!success) { // Join timed out, cancel the remaining jobs.
    group.cancel();
}
```

```
File dir = dirPath.toFile();
if (!dir.isDirectory())
     return new Status(Status.ERROR, "Test", dirPath + " is not a directory");

for (String child : dir.list()) {
     if (monitor.isCanceled())
          return Status.CANCEL_STATUS;
     IPath childPath = dirPath.append(child);
     File childFile = childPath.toFile();
     if (childFile.isDirectory()) {
          DirectoryDigger newJob = new DirectoryDigger(childPath, filesCollector);
          newJob.setJobGroup(getJobGroup());
          newJob.setSystem(true);
          newJob.schedule();
     } else {
          synchronized(filesCollector) {
          filesCollector.add(childFile);
     }
}
return Status.OK_STATUS;
```

File Search Speedup

- Job Groups are used to parallelize the file search operation
- 3x to 4x performance improvement

For more information - http://bugs.eclipse.org/441016

Job.join() with timeout

```
public boolean join(long timeoutMillis, IProgressMonitor monitor);
```

- Waits only for the given timeout duration.
- Can be cancelled using the passed in monitor.

Questions ???

Google

eclipsecon 2015

Evaluate the sessions

Sign in: www.eclipsecon.org