Java 並行処理を行う例

[Java]

並行処理コード抜粋

```
package info.typea.checker;

import info.typea.checker.SourceCodeAnalyzer;
import info.typea.checker.SourceCodeParser;
import info.typea.checker.JavaProgram;
import info.typea.checker.checkcase.CheckCase;
import info.typea.checker.checkcase.FormatCheckCase;
import info.typea.checker.checkcase.InvalidSyntaxCheckCase;

import java.io.File;
import java.io.FileOutputStream;
import java.io.FilenameFilter;
import java.io.OutputStream;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

/**
 * 並行処理で Java ソースコードをチェックする
 */
public class SourceCodeChecker {

    private static final int THREAD_SIZE = 8; // スレッドプールのサイズ
    private ExecutorService threadPool;       // スレッドプール

    /**
     * @param args
     * @throws Exception
     */
    public static void main(String[] args) throws Exception {

        if (args.length < 1) {
            System.out.println(usage());
            System.exit(-1);
        }
        String dirName  = args[0];
        SourceCodeChecker me = new SourceCodeChecker();
        me.check(new File(dirName));
        System.out.println("終了しました");
    }

    /**
     * @param dir 対象ディレクトリ
     * @throws Exception
     */
    public void check(File dir) throws Exception {
        if (!dir.exists() || !dir.isDirectory()) {
            throw new IllegalArgumentException("対象ディレクトリが不正です");
        }

        // 出力先ディレクトリ作成
        SimpleDateFormat sdf = (SimpleDateFormat) DateFormat.getInstance();
        sdf.applyPattern("yyyyMMdd_hhmmss");
           String outdirName = dir.getAbsolutePath() + File.separator + "out_" + sdf.format(new
Date());
        File outdir = new File(outdirName);

        if (!outdir.exists()) {
            if (!outdir.mkdirs()) {
                throw new IllegalStateException("出力先ディレクトリが作成できません " + outdirName);

            }
        }

        // ソースコードに対してチェックするタスクを追加する
        List<CheckCase> cases = new ArrayList<CheckCase>();
```

```
            cases.add(new FormatCheckCase());
            cases.add(new InvalidSyntaxCheckCase());

            // 対象ファイルを拡張子で判断
            File[] files = dir.listFiles(new FilenameFilter(){
                public boolean accept(File dir, String name) {
                    return name.toLowerCase().endsWith(".py") || name.toLowerCase().endsWith(".java");
                }});

            // ************* ここから並行処理 *******************
            threadPool = Executors.newFixedThreadPool(THREAD_SIZE);
            List<CheckTask> tasks = new ArrayList<CheckTask>();

            for (File file : files) {
                tasks.add(new CheckTask(file, outdir, cases));
            }

            // すべてのタスクの完了を待つ
            List<Future<Long>> result = threadPool.invokeAll(tasks);

            for (Future<Long> f : result) {
                System.out.printf("終了しました ... %d\n" , f.get());
            }
            // ************* ここまで並行処理 *******************
        }

        public static class CheckTask implements Callable<Long> {
            private File file;
            private File outdir;
            List<CheckCase> cases;

            /**
             *
             */
            public CheckTask(File file,  File outdir, List<CheckCase> cases) {
                this.file = file;
                this.outdir = outdir;
                this.cases = cases;
            }
            /* (non-Javadoc)
             * @see java.util.concurrent.Callable#call()
             */
            public Long call() throws Exception {
                System.out.printf("処理開始 ... %s\n", file.getName());

                  File outfile = new File(outdir.getAbsolutePath() + File.separator + file.getName() +
"_report.txt");
                OutputStream out = new FileOutputStream(outfile);

                SourceCodeParser parser = new SourceCodeParser();
                JavaProgram pgm = parser.parse(file);

                SourceCodeAnalyzer analyzer = new SourceCodeAnalyzer(pgm);
                long ret = analyzer.analyze(cases);

                pgm.printSource(out);

                out.close();

                return new Long(ret);
            }
        }
        public static String usage() {
            return "java SourceCodeChecker target directory";
        }
    }
```