说的准确点，断点续传指的就是在下载或上传时，将下载或上传任务（一个文件）人为的划分为几个部分，每一个部分采用一个线程进行上传或下载，如果碰到网络故障或人为因素导致程序终止后，下次还可以从已经上传或下载的部分开始继续上传下载未上传下载的部分，而没有必要重头开始上传下载。关键可以方便用户，节省时间，提高效率。

二、软件功能介绍

本软件使用Java Swing界面编程，多线程，序列化与反序列化等J2SE基础知识，其中断点续传的核心技术是Java对象的序列化和反序列化。

本软件的主要功能如下：

<!--[if !supportLists]-->1. <!--[endif]-->新建任务

新建下载任务，填写url,线程数，保存路径点“确定”后主窗口中添加一个下载任务。

<!--[if !supportLists]-->2. <!--[endif]-->任务开始

在等待下载的记录中选择一条记录点“任务开始”后被选择的记录开始下载

<!--[if !supportLists]-->3. <!--[endif]-->任务暂停

在正在下载的记录中选择一天记录执行此功能，暂停记录的下载

<!--[if !supportLists]-->4. <!--[endif]-->任务续传

一条记录暂停后如果需要重新下载可以选择“任务续传”功能，继续下载本条记录。

<!--[if !supportLists]-->5. <!--[endif]-->任务取消

从下载任务中删除不符合条件的纪录。

<!--[if !supportLists]-->6. <!--[endif]-->视图

在视图功能中可以显示和隐藏工具栏、状态栏

<!--[if !supportLists]-->7. <!--[endif]-->帮助

帮助中显示软件的作者，联系方式等相关信息。

三、软件核心代码：

断点续传内核的实现主要用了8个类：

FileInfo.java 描述下载文件的详细信息，下载文件的网络地址 URL

FileTask.java 任务主线程，同时还描述任务的详细信息

MainFrame.java 下载界面

MyControl.java 工具类，放一些方法

MyListener.java 主要是一些事件操作

NewTask.java 新建任务界面

SplitFileTask.java 分任务线程

TestMain.java 最后还有一个测试类

1. FileInfo.java，描述下载文件的详细信息

Java代码

查看源代码

打印

001.package download;

002.

003.

004.

005.import java.io.File;

006.

007.import java.io.FileInputStream;

008.

009.import java.io.FileNotFoundException;

010.

011.import java.io.FileOutputStream;

012.

013.import java.io.IOException;

014.

015.import java.io.ObjectInputStream;

016.

017.import java.io.ObjectOutputStream;

018.

019.import java.io.Serializable;

020.

021.import java.net.HttpURLConnection;

022.

023.import java.net.MalformedURLException;

024.

025.import java.net.URL;

026.

027.import java.util.Calendar;

028.

029.import java.util.concurrent.ExecutorService;

030.

031.

032.

033.import javax.swing.JOptionPane;

034.

035.import javax.swing.JTable;

036.

037.

038.

039.@SuppressWarnings("serial")

040.

041.public class FileTask implements Serializable, Runnable {

042.

043. private FileInfo fileInfo = null;// 任务文件信息

044.

045. private SplitFileTask[] splitTask = null;// 分任务数组

046.

047. private long fileLength;// 获得文件的长度

048.

049. private long[] startPos;// 分任务开始位置

050.

051. private long[] endPos;// 分任务结束位置

052.

053. private int countThread;// 分任务的数目

054.

055. private boolean isFirst = true; // 是否第一次取文件

056.

057. private boolean isStop = true; // 停止标志

058.

059. private File tempFile; // 文件下载的临时信息

060.

061. private ExecutorService executor = null;// 线程池

062.

063. private JTable viewTable = null;

064.

065. private int row = 0;// 任务编号

066.

067. private String speed = null;// 任务下载的速度

068.

069.

070.

071. public FileTask(FileInfo fileInfo, ExecutorService executor) {

072.

073. this.fileInfo = fileInfo;// 文件信息

074.

075. this.executor = executor;// 线程池

076.

077. }

078.

079.

080.

081. public void run() {

082.

083. if (fileInfo.isOver())// 入文件已下载完毕，则结束任务返回

084.

085. return;

086.

087. else

088.

089. isStop = false;// 否则，将isStop 置为false

090.

091.

092.

093. tempFile = new File(fileInfo.getFilePath() + File.separator

094.

095. + fileInfo.getFileName() + ".info");

096.

097. countThread = fileInfo.getConutThread();// 分任务的数目

098.

099.

100.

101. if (tempFile.exists()) {// 文件已存在，是续传文件

102.

103. isFirst = false;

104.

105. System.out.println("文件已存在，是续传文件" + tempFile);

106.

107. readFilePos();// 读取保存的下载信息（文件指针位置）

108.

109. } else {

110.

111. startPos = new long[countThread];

112.

113. endPos = new long[countThread];

114.

115. splitTask = new SplitFileTask[countThread];

116.

117. fileLength = getFileLong();// 获取文件的长度

118.

119. if (fileLength <= 0) {

120.

121. JOptionPane jp = new JOptionPane("网络资源不正确！");

122.

123. jp.createDialog("提示信息");

124.

125. return;

126.

127. }

128.

129. fileInfo.setStartDate(Calendar.getInstance());// 设置任务开始时间

130.

131. fileInfo.setFileLength(fileLength);// 设置文件的长度

132.

133. }

134.

135. long tem = fileLength / countThread;// 每个分任务的长度

136.

137. if (isFirst) {// 若是第一次

138.

139. for (int i = 0; i < countThread; i++) {

140.

141. startPos[i] = i \* tem; // 分任务开始位置

142.

143. }

144.

145. for (int i = 0; i < countThread - 1; i++) {

146.

147. endPos[i] = startPos[i + 1]; // 分任务结束位置

148.

149. }

150.

151. endPos[countThread - 1] = fileLength;

152.

153. for (int i = 0; i < countThread; i++) {// 创建分任务

154.

155. /\*

156.

157. \* String path = fileInfo.getFilePath() + "//" +

158.

159. \* fileInfo.getFileName() + ".temp";

160.

161. \*/

162.

163. String path = fileInfo.getFilePath() + "//"

164.

165. + fileInfo.getFileName();

166.

167. splitTask[i] = new SplitFileTask(fileInfo.getSiteURL(), path,

168.

169. i, startPos[i], endPos[i]);

170.

171. }

172.

173. }

174.

175.

176.

177. // 执行分任务

178.

179. for (int i = 0; i < countThread; i++) {

180.

181. // if(!splitTask[i].isOver())

182.

183. executor.execute(splitTask[i]);

184.

185. }

186.

187. long tempLength = testFileLength();// 测量文件已下载的长度

188.

189.

190.

191. boolean flag = true;

192.

193. // 等待子线程结束while循环

194.

195. while (!isStop) {

196.

197. speed = MyControl.speed(tempLength, testFileLength(), 0.5f);// 测量文件下载的速度

198.

199. tempLength = testFileLength();

200.

201. showViewTable();// ////刷新任务栏

202.

203. writeFilePos();

204.

205. if (flag) {

206.

207. fileInfo.setSpendTime(fileInfo.getSpendTime() + 1);

208.

209. }

210.

211. flag = !flag;

212.

213. MyControl.sleep(500);// 睡眠500毫秒

214.

215. int i = 0;

216.

217. for (; i < countThread; i++) {

218.

219. if (!splitTask[i].isStop()) {// 分任务没有结束

220.

221. break;