解析 Java 命令行選項

JOptSimple 是一個設計用來簡化應用程式中命令行選項解析的 Java 庫。它允許你以靈活和直觀的方式定義和處理選項，例如標誌（例如 `-v` 用於詳細模式）和帶有參數的選項（例如 `--file input.txt`）。以下是如何有效使用 JOptSimple 的分步指南。

步驟 1：將 JOptSimple 添加到您的項目

要使用 JOptSimple，首先需要在 Java 專案中包含它。如果您使用 Maven，請將以下依賴項添加到 `pom.xml` 文件中：

```
<dependency>
    <groupId>net.sf.jopt-simple</groupId>
    <artifactId>jopt-simple</artifactId>
    <version>5.0.4</version>
</dependency>
```

請確保在 Maven Central 上檢查最新版本，因為 `5.0.4` 可能不是最新的。對於其他構建工具（例如 Gradle），您可以相應地適應依賴項（例如 `implementation 'net.sf.jopt-simple:jopt-simple:5.0.4'`）。

步驟 2：創建一個 OptionParser

JOptSimple 的核心是 `OptionParser` 類，您可以使用它來定義和解析命令行選項。首先在 `main` 方法中創建它的實例：

```
import joptsimple.OptionParser;
import joptsimple.OptionSet;

public class MyApp {
    public static void main(String[] args) {
        OptionParser parser = new OptionParser();
        // 在這裡定義選項（請參閱步驟 3）
    }
}
```

步驟 3：定義命令行選項

您可以使用 `accepts` 或 `acceptsAll` 方法來定義選項。選項可以是標誌（無參數）或需要參數的選項（例如文件名或數字）。以下是如何設置它們的方法：

- **標誌**：使用 `accepts` 為單個選項名稱或 `acceptsAll` 來指定別名（例如 `-v` 和 `--verbose`）：

  ```
  parser.acceptsAll(Arrays.asList("v", "verbose"), " 啟用詳細模式");
  ```

- **帶有參數的選項**：使用 `withRequiredArg()` 表示選項需要一個值，並可選地使用 `ofType()` 來指定其類型：

  ```
  parser.acceptsAll(Arrays.asList("f", "file"), " 指定輸入文件").withRequiredArg();
  parser.acceptsAll(Arrays.asList("c", "count"), " 指定計數").withRequiredArg().ofType(Integer.class).defau
  ```

  - `defaultsTo(0)` 如果未提供選項則設置默認值（例如 `0`）。
  - `ofType(Integer.class)` 確保參數被解析為整數。

- **幫助選項**：添加幫助標誌（例如 `-h` 或 `--help`）以顯示使用信息：

  ```
  parser.acceptsAll(Arrays.asList("h", "help"), " 顯示此幫助訊息");
  ```

步驟 4：解析命令行參數

將 `args` 陣列從您的 `main` 方法傳遞給解析器以處理命令行輸入。這將返回一個包含解析選項的 `OptionSet` 對象：

```
OptionSet options = parser.parse(args);
```

將其包裹在 `try-catch` 塊中以處理解析錯誤（例如無效選項或缺少參數）：

```
try {
    OptionSet options = parser.parse(args);
    // 處理選項（請參閱步驟 5）
} catch (Exception e) {
    System.err.println(" 錯誤: " + e.getMessage());
    try {
        parser.printHelpOn(System.err);
    } catch (IOException ex) {
        ex.printStackTrace();
    }
    System.exit(1);
}
```

步驟 5：訪問解析選項

使用 `OptionSet` 來檢查標誌、檢索選項值並獲取非選項參數：

- **檢查標誌**：使用 `has()` 來查看標誌是否存在：

```
boolean verbose = options.has("v");
if (verbose) {
    System.out.println(" 啟用詳細模式");
}
```

- **獲取選項值**：使用 `valueOf()` 來檢索選項的參數，如果需要則將其轉換為適當的類型：

```
String fileName = (String) options.valueOf("f"); // 如果未指定則返回 null
int count = (Integer) options.valueOf("c");      // 由於 defaultsTo(0) 返回 0
```

  如果您指定了 `ofType()` 和 `defaultsTo()`，`valueOf()` 返回類型化值或默認值。

- **非選項參數**：使用 `nonOptionArguments()` 獲取未與選項關聯的參數（例如文件列表）：

```
List<String> files = options.nonOptionArguments();
System.out.println(" 文件: " + files);
```

- **處理幫助**：如果存在幫助選項，則打印使用信息：

```
if (options.has("h")) {
    parser.printHelpOn(System.out);
    System.exit(0);
}
```

範例：將所有內容結合在一起

以下是一個接受詳細標誌、計數選項和文件列表的完整程序範例：

```
import joptsimple.OptionParser;
import joptsimple.OptionSet;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;

public class MyApp {
    public static void main(String[] args) {
        OptionParser parser = new OptionParser();
```

```
        parser.acceptsAll(Arrays.asList("v", "verbose"), " 啟用詳細模式");
        parser.acceptsAll(Arrays.asList("c", "count"), " 指定計數")
              .withRequiredArg()
              .ofType(Integer.class)
              .defaultsTo(0);
        parser.acceptsAll(Arrays.asList("h", "help"), " 顯示此幫助訊息");

        try {
            OptionSet options = parser.parse(args);

            if (options.has("h")) {
                parser.printHelpOn(System.out);
                System.exit(0);
            }

            boolean verbose = options.has("v");
            int count = (Integer) options.valueOf("c");
            List<String> files = options.nonOptionArguments();

            if (verbose) {
                System.out.println(" 啟用詳細模式");
            }
            System.out.println(" 計數: " + count);
            System.out.println(" 文件: " + files);

        } catch (Exception e) {
            System.err.println(" 錯誤: " + e.getMessage());
            try {
                parser.printHelpOn(System.err);
            } catch (IOException ex) {
                ex.printStackTrace();
            }
            System.exit(1);
        }
    }
}
```

使用 `java MyApp -v -c 5 file1 file2` 運行此程序，它輸出：

```
啟用詳細模式
計數: 5
```

```
文件: [file1, file2]
```

使用 `-h` 運行，它顯示包含所有定義選項的使用訊息。

其他功能

- **多個值**：允許選項出現多次（例如 `-I dir1 -I dir2`）：

```
parser.accepts("I").withRequiredArg().ofType(String.class);
List<String> includes = options.valuesOf("I"); // 返回一個列表
```

- **可選參數**：使用 `withOptionalArg()` 而不是 `withRequiredArg()` 如果參數不是必需的：

```
parser.accepts("o").withOptionalArg().ofType(String.class);
```

- **錯誤處理**：JOptSimple 為解析錯誤拋出 `OptionException`，您可以捕捉它以提供用戶友好的反饋。

摘要

JOptSimple 使得在 Java 中處理命令行選項變得簡單。您：1. 將庫添加到您的項目中。2. 創建一個 `OptionParser` 並定義選項（標誌、參數、默認值）。3. 將 `args` 陣列解析為 `OptionSet`。4. 使用 `OptionSet` 來訪問標誌、值和非選項參數。5. 根據需要處理錯誤並提供幫助。

有了這個基礎，您可以構建針對應用程式需求量身定制的強大命令行介面！