Backlog 移行ツール

当ツールを用いて、課題やWikiを別のスペースへ移行できます。同一スペースでプロジェクトを分けたいときにもご利用できます。

必須要件

1. Java 8 実行環境
2. Java の実行可能 jar ファイルを実行できる
3. コマンドラインで簡単なコマンドを実行できる
4. 移行元のスペースの管理者である、もしくは移行元プロジェクトのプロジェクト管理者である
   [プロジェクト管理者の設定]
   http://www.backlog.jp/help/adminsguide/project-setting/userguide2449.html

5. 移行先のスペースの管理者である

任意要件

1. Backlog API の[課題一覧の取得]のクエリパラメータを作ることができる（移行する課題を絞り込みたいときのみ必要）

   [課題一覧の取得]
   https://developer.nulab-inc.com/ja/docs/backlog/api/2/get-issues

ダウンロード

以下から最新版の jar ファイルを、お手元の環境に ダウンロードしてください。
http://www.backlog.jp/backlog-migration/releases.html

以下のコマンドでヘルプが表示されるかお試しください。
java -jar backlog-migration-<latest version>.jar

前準備

1. 移行先と移行元 両方のプロジェクトで、API キーを生成します。

   [APIの設定]
   http://www.backlog.jp/help/usersguide/personal-settings/userguide2378.html

2. 作業用のディレクトリ(ここではworkとします)を作成し、jar ファイルを移動します。

Mac

```
$ mkdir work
$ cd work
$ mv ダウンロードディレクトリ/backlog-migration-[latest version].jar .
```

Windows

```
c:¥> md work
c:¥> cd work
c:¥work> move ダウンロードフォルダ¥backlog-migration-[latest version].jar .
```

移行

1. 移行先プロジェクトの作成

新しいプロジェクトに移行される場合は、あらかじめプロジェクトを作成してください。

[プロジェクトの追加]
http://www.backlog.jp/help/adminsguide/project-setting/userguide353.html

2. 初期化

init コマンドを実行し、マッピングファイルを準備します。マッピングファイルは移行先と移行元のユーザを紐づけるのに必要です。

```
java -jar backlog-migration-[latest version].jar \
init \
--src.key [移行元のAPIキー] \
--src.url [https://移行元のスペースID.backlog.jp] \
--dst.key [移行先の API キー] \
--dst.url [https://移行先のスペースID.backlog.jp] \
--projectKey [移行元のプロジェクトキー:移行先のプロジェクトキー]
```

以下のオプションは必須です。

--src.key	前準備で作成した移行元のAPIキー
--src.url	移行元のBacklog URL
--dst.key	前準備で作成した移行先のAPIキー
--dst.url	移行先のBacklog URL

--projectKey	移行元のプロジェクトキー、移行先のプロジェクトキーの順番でコロンを挟んで指定してください 例）--projectKey SRC_PRJCT:DST_PRJCT 移行元と移行先のプロジェクトキーが一致する場合以下のように省略可能です。 例）--projectKey PRJCT

init コマンドを実行すると、以下のようなディレクトリとファイルが生成されます。

```
実行ディレクトリ/
├ mapping/
│   └ users.json
└ log/
   ├ backlog-migration-warn.log
   └ backlog-migration.log
```

mapping/user.json	ユーザのマッピングファイルです。下記「**マッピングファイルの修正**」で使用します。
log/backlog-migration-warn.log	ログファイルです。init や execute 処理で致命的ではない警告ログが出力されます。
log/backlog-migration.log	ログファイルです。init や execute 処理の内容が追記出力されます。

3. マッピングファイルの修正

user.json はjson形式で出力されるテキストファイルです。 項目descriptionを参考に、**項目dstの値を埋めてください**。

サンプル

```
{
  "description": "移行先Backlogに設定可能なユーザーは[admin,tanaka,yamada]です。",
  "mappings": [{
    "info": {
      "name": "山田",
      "mail": "yamada@yyyy.com"
    },
    "mappingType": "UserId",
    "src": "yamada",
    "dst": "yamada"
  }, {
    "info": {
      "name": "高橋",
      "mail": "takahashi@xxxx.co.jp"
    },
    "mappingType": "UserId",
    "src": "takahashi",
    "dst": ""
  }, {
    "info": {
      "name": "佐々木（高橋の旧姓）",
      "mail": ""
    },
    "mappingType": "Name",
    "src": "佐々木（高橋の旧姓）",
    "dst": ""
  }]
}
```

description	移行先のスペースに所属するユーザのユーザIDがカンマ区切りで出力されます。変更しないでください。

mappings:[{ info:{name} }]	移行元ユーザーの名前。移行先ユーザーを入力する際の参考情報になります。
mappings:[{ info:{mail} }]	移行元ユーザーのメールアドレス。移行先ユーザーを入力する際の参考情報になります。
Mappings:[{mappingType}]	mappingTypeが"UserId"の場合は、移行元のプロジェクトメンバーであることを示します。 mappingTypeが"Name"の場合は、「名前が変更された」・「アカウントが削除された」等により、移行元に存在が確認できないユーザを示します。課題の更新履歴の移行に必要になります。name項目が一緒に出力されます。
mappings:[{src}]	移行元のプロジェクトに所属するユーザを表します。 mappingTypeが”UserId”の場合は、userId、”Name”の場合は名前が設定されます。src以下は変更しないでください。
mappings:[{dst}]	移行先のスペースに属するユーザのユーザIDを指定します。
	移行元と同じユーザID を持つユーザが移行先スペースにも存在した場合は、ツールが自動でそのユーザIDを出力します。手動で書き換えても構いません。 自動割り当てができなかった場合は空で出力されます。 descriptionを参考に**必ず埋めてください**。 ここで指定したユーザ が、移行先プロジェクトのメンバーではない場合、後の execute 時に、ツールが自動でメンバーとして追加します。

	すでに移行元のスペースから削除された等のユーザアカウントである場合は、ダミーユーザを移行先に登録し、移行作業が終わった後、そのアカウントを削除してください。

4. 移行

execute コマンドを実行すると、課題とWikiがインポートされます。移行元の課題やWikiが削除されることはありません。

```
$ java -jar backlog-migration-<latest version>.jar \
execute \
--src.key [移行元のAPIキー] \
--src.url [https://移行元のスペースID.backlog.jp] \
--dst.key [移行先の API キー] \
--dst.url [https://移行先のスペースID.backlog.jp] \
--projectKey [移行元のプロジェクトキー:移行先のプロジェクトキー]
```

以下のオプションは必須です。

--src.key	前準備で作成した移行元のAPIキー
--src.url	移行元のBacklog URL
--dst.key	前準備で作成した移行先のAPIキー
--dst.url	移行先のBacklog URL
--projectKey	移行元のプロジェクトキー、移行先のプロジェクトキーの順番でコロンを挟んで指定してください 例）--projectKey SRC_PRJCT:DST_PRJCT 移行元と移行先のプロジェクトキーが一致する場合以下のように省略可能です。 例）--projectKey PRJCT

また、以下のオプションでカスタマイズできます。

--fitIssueKey	可能な限り課題番号を一致させます。課題詳細やコメント、Wiki に課題番号を記載している場合はこのオプションを指定することで、リンク切れを回避できます。 プロジェクトの統合等により、先に課題が存在するなど物理的に不可能な場合は一致させることはできません。
--exclude	課題やWikiを移行しないよう制御できます。値は issue もしくは wiki です。 --exclude wiki --exclude issue

応用1：移行する課題のフィルタリング

移行元のプロジェクトの課題をフィルタリングして、移行することができます。
フィルタリングの指定方法は、Backlog API v2の課題一覧の取得APIのクエリとほぼ同じです。
[課題一覧の取得]
https://developer.nulab-inc.com/ja/docs/backlog/api/2/get-issues

execute コマンドでオプション--filterを指定してください。(initコマンドでは使用できませんのでご注意ください。) ~~init, execute コマンド両方でオプション--filterを指定してください。~~
　例として、比較的よく使われそうなフィルタリングオプションを以下の表に示します。詳細は上記の「課題一覧の取得」ページをご確認ください。

オプション指定例	型	内容
--filter “issueTypeId[]=1&issueTypeId[]=2”	数値	種別のIDでフィルタリングします。ID がわからない場合は、Backlog をブラウザで開き、プロジェクト設定＞種別の一覧画面で、種別名にカーソルを合わせるとEditIssueType.action?issueType.id=xとリンクが表示されます。issueType.idの値を指定してください。 複数指定したい場合は、オプション指定例のように、複数のissueTypeId[] を&で連結してください。
--filter	数値	カテゴリーのIDでフィルタリングします。ID がわ

“categoryId[]=99&categoryId[]=98”		からない場合は、Backlog をブラウザで開き、プロジェクト設定＞カテゴリーの一覧画面で、カテゴリー名にカーソルを合わせるとEditComponent.action?component.id=xとリンクが表示されますcomponent.idの値を指定してください。
--filter “id[]=123&id[]=456”	数値	課題のIDでフィルタリングします。 移行に失敗した課題が明確な場合、その課題の移行のみをリトライしたい時に指定すると便利です。
--filter “offset=1000”	数値	課題のオフセットです。 移行する課題が大量にある場合、時間を要するため、このオプションで分割して移行されることをお勧めします。 --filter countと組み合わせてご利用ください。 例）1件目から100件移行したい場合 --filter “offset=0&count=100” 例）1001件目から100件移行したい場合 --filter “offset=1000&count=100”
--filter “count=100”	数値	課題の移行最大数です。 --filter offsetと組み合わせてご利用ください。

再移行時の重複チェック

「課題を登録したユーザID」、「課題の登録時間」、「課題の件名」が全て一致する場合は、すでに移行された課題として、重複して移行されることはありません。何らかのエラーで移行が途中で終了し、再度 execute が必要なとき等に有益な重複チェックです。

「Wikiのページ名」が一致する場合は、すでに移行されたWikiとして、重複して移行されることはありません。再度 execute する際に有益な重複チェックです。

制限事項

1.親課題を変更後にプロジェクトキーを変えた場合、移行元では親課題の変更を反映できません。