WikiLink` package

Version 0.1, August 15, 2005, Scott Morrison
Available under MIT and GPL licenses.

Introduction

Provides a thin wrapper around a java class for manipulating Mediawiki connections.

Implementation

```
(*<pre>*)
```

```
BeginPackage["WikiLink`", {"JLink`"}];
```

```
AddJarPath::usage =
  "You'll need to call this, specifying the path to the jar files needed by this
    package. Hopefully, they're in the same place as the package itself.";
```

```
SetDefaultWikiURL::usage =
  "Specify a default Wiki URL to attempt to connect to. If a default URL has
    been specified, and a connection has not been made explicitly using
    CreateWikiConnection, the WikiLink` package will attempt to connect to the
    default URL, with no username or password, on the first attempted use.";
```

```
CreateWikiConnection::usage =
  "CreateWikiConnection[URL, username, password] initialises a
    connection to a mediawiki server. The URL should typically end
    in \"index.php\". The username and password are optional.\n";
WikiGetPageText::usage = "WikiGetPageText[pagename] returns
    the raw text of the specified page.";
WikiGetPageTexts::usage = "WikiGetPageTexts[{pagename1, pagename2,
    ...}] returns a list of pairs of the form {{pagename1, text1},
    {pagename2, text2}, ...}. If this failed for some pages, the
    expression $Failed will appear instead of the raw wiki text.";
WikiSetPageText::usage = "WikiSetPageText[pagename, text] overwrites
     the contents of the specificied page with the given text.\n" <>
   "WikiSetPageText[pagename, text, summary] overwrites the contents of the
     specificied page with the given text and notes summary in the change log.\n";
WikiSetPageTexts::usage = "WikiSetPageText[{{pagename1,
    text1},{pagename2,text2},...}] efficiently sets multiple pages,
    by first checking which texts are already up to date.\n";
WikiUploadFile::usage = "WikiUploadFile[name, description]
    uploads the specified file to the wiki.";
```

```
WikiUserName::usage =
  "WikiUserName[] returns either the name of the user you are logged
    in as, your IP address if you're not logged in, or
    $Failed if something more complicated has happened!";
```

```
WikiAllPages::usage =
  "WikiAllPages[url,initialtext,namespace,namespaceNumber] attempts to produce a list
    of all pages in the namespace with titles beginning with initialtext. The
    first argument, url, shoould be the url of the wiki, ending in index.php.";
```

```
WikiPageMatchQ;
WikiPageFreeQ;
WikiStringReplace;
WikiStringCases;
WikiPagesContaining;
```

```
Begin["`Private`"];
```

```
WikiLinkDirectory[] = ParentDirectory[DirectoryName[File /. Flatten[
      FileInformation[ToFileName[#, "WikiLink.m"]] & /@ ($Path /. "." → Directory[])]]];
```

```
mediawikiConnection; defaultURL = "";
```

```
SetDefaultWikiURL[URL_String] := (defaultURL = URL)
```

```
AddJarPath[path_String] :=
 Module[{PathSeparator, FileExists, jars},
  jars =
   ToFileName[path, #] & /@ {"commons-httpclient-3.0-rc2.jar", "commons-codec-1.3.jar",
     "commons-logging.jar", "commons-lang-2.1.jar", "jdom.jar", "wikilink.jar"};
  FileExists[file_String] := FileNames[file] =!= {};
  jars = Select[jars, FileExists];
  InstallJava[];
  AddToClassPath @@ jars;
 ]
```

```
AddJarPath[paths__String] := AddJarPath /@ {paths}
```

```
GuessJarPath[] :=
 AddJarPath[WikiLinkDirectory[], ToFileName[WikiLinkDirectory[], "jars"]]
```

```
CreateWikiConnection[baseURL_String] :=
 (InstallJava[];
  GuessJarPath[];
  mediawikiConnection = JavaNew["wikilink.MediawikiConnection", baseURL];)
```

```
CreateWikiConnection[baseURL_String, username_String, password_String] :=
 (InstallJava[];
  GuessJarPath[];
  mediawikiConnection = JavaNew["wikilink.MediawikiConnection",
    baseURL, username, password];)
```

```
WikiConnectionValidQ[] := If[JavaObjectQ[mediawikiConnection],
  True, If[defaultURL == "", False, CreateWikiConnection[defaultURL];
   defaultURL = "";
   WikiConnectionValidQ[]]]
```

```
WikiGetPageText::invalid =
  WikiSetPageText::invalid = "You must call CreateWikiConnection
     before using WikiGetPageText or WikiSetPageText";
```

```
WikifyName[name_String] := StringReplace[name, " " → "_"]
```

```
UnwikifyName[name_String] := StringReplace[name, "_" → " "]
```

```
WikiGetPageText[name_String] :=
 If[WikiConnectionValidQ[], mediawikiConnection@getPageText[WikifyName[name]],
  Message[WikiGetPageText::invalid]]
```

```
WikiGetPageTexts[names : {__String}] :=
 If[WikiConnectionValidQ[],
  Module[{results, getResult, wikinames = WikifyName /@ names},
   results = mediawikiConnection@getPageTexts[wikinames];
   getResult[name_] := Module[{c, r},
     c = Cases[results, {WikifyName[name], r_} :↦ r];
     If[Length[c] == 1, c[[1]], ""]
    ];
   {#, getResult[#]} & /@ names
  ]
 ]
```

General::spell : Possible spelling error: new symbol name "WikiGetPageTexts" is similar to existing symbols {WikiGetPageText, WikiSetPageTexts}. More...

General::spell1 : Possible spelling error: new symbol name "names" is similar to existing symbol "Names". More...

```
WikiSetPageText[name_String, contents_String] :=
 WikiSetPageText[WikifyName[name], contents, ""]
```

```
WikiSetPageText[name_String, contents_String, summary_String] :=
 If[WikiConnectionValidQ[],
  mediawikiConnection@setPageText[WikifyName[name], contents, summary],
  Message[WikiSetPageText::invalid]
 ]
```

```
WikiSetPageTexts[uploadPairs_List] := WikiSetPageTexts[uploadPairs, ""]
```

```
WikiSetPageTexts[uploadPairs_List, summary_String] :=
 If[WikiConnectionValidQ[],
  mediawikiConnection@
   setPageTexts[{WikifyName[#[[1]]], #[[2]]} & /@ uploadPairs, summary],
  Message[WikiSetPageText::invalid]
 ]
```

```
WikiUploadFile[name_String, description_String] := If[WikiConnectionValidQ[],
  mediawikiConnection@uploadFile[name, description], Message[WikiUploadFile::invalid]]
```

```
WikiUserName[] := Module[{text, cases},
  If[! WikiSetPageText["Who_Am_I", "-----"], Return[$Failed]];
  text = WikiGetPageText["Who_Am_I"];
  cases = StringCases[text, "--[[User:" ~~ x__ ~~ "|" ~~ x__ ~~ "]]" :> x];
  If[Length[cases] == 1, Return[cases[[1]]]];
  Return[$Failed];
 ]
```

```
WikiPageMatchQ[name_String, patt_] := StringMatchQ[WikiGetPageText[name], patt]
```

```
WikiPageMatchQ[names : {__String}, patt_] :=
 StringMatchQ[#[[2]], patt] & /@ WikiGetPageTexts[names]
```

```
WikiPagesContaining[names : {__String}, pattern_] :=
 Pick[names, Last /@ WikiGetPageTexts[names], _?(! StringFreeQ[#, pattern] &)]
```

```
WikiPageFreeQ[name_String, patt_] := StringFreeQ[WikiGetPageText[name], patt]
```

```
WikiPageFreeQ[names : {__String}, patt_] :=
 StringFreeQ[#[[2]], patt] & /@ WikiGetPageTexts[names]
```

```
WikiStringReplace[name_String, rules_] :=
 WikiSetPageText[name, StringReplace[WikiGetPageText[name], rules]]
```

```
WikiStringReplace[name_String, rules_, n_Integer] :=
 WikiSetPageText[name, StringReplace[WikiGetPageText[name], rules, n]]
```

```
WikiStringReplace[name_String, rules_, description_String] :=
 WikiSetPageText[name, StringReplace[WikiGetPageText[name], rules], description]
```

```
WikiStringReplace[name_String, rules_, n_Integer, description_String] :=
 WikiSetPageText[name, StringReplace[WikiGetPageText[name], rules, n], description]
```

```
WikiStringReplace[names : {__String}, rules_] :=
 WikiSetPageTexts[{#[[1]], StringReplace[#[[2]], rules]} & /@ WikiGetPageTexts[names]]
```

```
WikiStringReplace[names : {__String}, rules_, n_Integer] :=
 WikiSetPageTexts[{#[[1]], StringReplace[#[[2]], rules, n]} & /@ WikiGetPageTexts[names]]
```

```
WikiStringReplace[names : {__String}, rules_, summary_String] := WikiSetPageTexts[
   {#[[1]], StringReplace[#[[2]], rules]} & /@ WikiGetPageTexts[names], summary]
```

```
WikiStringReplace[names : {__String}, rules_, n_Integer, summary_String] :=
 WikiSetPageTexts[
   {#[[1]], StringReplace[#[[2]], rules, n]} & /@ WikiGetPageTexts[names], summary]
```

```
WikiStringCases[name_String, rules_] := StringCases[WikiGetPageText[name], rules]
```

```
WikiStringCases[names : {__String}, rules_] :=
 {#[[1]], StringCases[#[[2]], rules]} & /@ WikiGetPageTexts[names]
```

```
nextPageURL[hostname_, baseURL_, text_, initialText_ : ""] :=
 Module[{candidates}, candidates = StringCases[text, "<a href=\"" ~~
       nextURL : (baseURL ~~ "?title=Special:Allpages&amp;from=" ~~ initialText ~~
             ShortestMatch[__]) ~~ "\" title=\"Special:Allpages\">Next page" :>
      "http://" <> hostname <> StringReplace[nextURL, "&amp;" -> "&"], 1];
  If[Length[candidates] == 0, $Failed, candidates[[1]]]
 ]
```

```
WikiAllPages[hostname_String, baseURL_String : "/w/index.php", initialText_String : "",
  nameSpace_String : "", nameSpaceNumber_Integer : 0, maxPages_ : ∞] :=
 Module[{pages = {}, newPages, allPagesURL =
    "http://" <> hostname <> baseURL <> "?title=Special%3AAllpages&from=" <>
     initialText <> "&namespace=" <> ToString[nameSpaceNumber], allPagesText},
  While[allPagesURL =!= $Failed ∧ Length[pages] ≤ maxPages,
   Print["Looking at: " <> allPagesURL];
   allPagesText = Import[allPagesURL, "Text"];
   If[allPagesText == $Failed, Continue[]];
   allPagesURL = nextPageURL[hostname, baseURL, allPagesText, initialText];
   newPages =
    StringCases[allPagesText, "title=\"" ~~ pagename : (((nameSpace ~~ ":") | "") ~~
           initialText ~~ ShortestMatch[__]) ~~ "\"" :> pagename];
   Print["Found " <> ToString[Length[newPages]] <> " more pages."];
   pages = pages ~Join~ newPages;
  ];
  pages
 ]
```

```
End[];
```

```
EndPackage[];
```

```
(*</pre>[[Category:Source Code]]*)
```