pURL™

URL manager for the digerati

API Guide
version 1.0

110

DIGITAL OBJECTIVES®

Tools for the digerati™

Table of Contents

Introduction

This manual documents the Application Program Interface (API) supported by **pURL™**. The API enables third-party developers to treat pURL as a repositiory of URL information.

The API supports:

- Adding URLs to pURL
- Checking for the existence of an URL in pURL
- Retrieving URLs from pURL
- Triggering the launch of third-party applications from pURL

Using the pURL API

The pURL API is a set of methods in the base view. Access to any of the methods is achieved by sending the appropriate message to the base view via:

```
    GetRoot().|pURL:DigObj|:pURLmethod(parameters) ;
```

Each of the methods is discussed below.

Adding URLs

URLs are added to pURL via the AddpURL method.

:AddpURL(*urlList*)

Adds an URL or an array of URLs to the root folder.

urlList is either a frame or an array of frames. Each frame corresponds to an URL and has the following structure:

```
    {
        url:        urlString,   // include scheme if known
        name:       string,      // can be the webpage title e.g.
        comments:   rstring
    }
```

urlString is the full URL with scheme (if known).
rstring may be a rich string.

The return value from the method will be non-nil if the URL was successfully added. Should pURL encounter a soup related error, the method will issue a :Notify and return a value of nil.

The method is undo / redoable.

Example:

```
:AddpURL([{url:"http://members.aol.com/DigObj/pURL.html",
           name:"pURL’s Home Page",
           comments:"Where you get pURL"}]);
```

Checking the Existence of an URL

An URL can be checked to determine if it is already in the pURL repository by using the CheckpURL method.

:CheckpURL(*urlString*)

Checks for the existence of an URL that matches *urlString* . The match test is case insensitive.This method may be used before invoking AddpURL to prevent adding duplicate URLs.

urlString is the full URL with scheme.

If the URL is not found then the method will return the value nil. If fhe URL is found then the method will return a frame with the following structure:

```
{
        url:            urlStringSansScheme,    // does not include
                                                // scheme
        name:           string,
        comments:       rstring
}
```

Note that *urlStringSansScheme* will not contain the scheme.
rstring may be a rich string.

Example:

```
:CheckpURL("http://members.aol.com/DigObj/pURL.html");
```

Retrieving URLs

URLs can be retrieved from the pURL repository via the GetpURL method.

:GetpURL(*typeList*)

Returns an array of frames. Each frame contains information about a single URL and has the following structure:

```
{
        url:            urlString,      // includes scheme
        name:           string,
        comments:       rstring
}
```

rstring may be a rich string.

typeList is used to filter the URLs in the repository so that the only URLs returned are those that match the requested schemes. *TypeList* can be either the value nil, a symbol or an array of symbols.

If *typeList* is either nil or an empty array then all URLs in the repository are returned.

if *typeList* is a symbol or an array of symbols then the only URLs returned are those whose schemes correspond to the specified symbols.

Supported symbols are:

```
     'http, 'wais, 'ftp, 'mailto, 'news, 'gopher, 'unknown
```

This set will expand in future versions.

This method always returns an array. If any URLs are found then the array will contain frames of the structure discussed above. If no URLs are found then the array will be empty.

Example:

```
:GetpURL(['http,'news]);
```

Triggering the Launch of Third-party Applications

An URL can be used to trigger the opening of a third-party NIE based application via the LaunchFrompURL method.

:LaunchFrompURL(*urlString*)

The URL's scheme is associated with the target NIE based application via the pURL preferences. This method will open the target NIE based application and pass it the target URL.

urlString is the full URL with scheme.

Currently, pURL supports Newt's Cape™, Newt'sPaper™, and NetHopper® as target NIE based applications. This set will expand in future versions.

If pURL's preferences have not been installed then this method will issue a :Notify and return nil. If the target NIE based application is not found then pURL will issue a :Notify and return nil.

In all other cases, the method will propagate the return value from the launched application.

If the value returned from a Newt's Cape launch is nil then pURL will issue a :Notify.

Example:

```
:LaunchFrompURL("http://members.aol.com/DigObj/pURL.html");
```