nsHandlerService discussion

Alphan/Eden

Outline

- mimeType.rdf Replacement
- API Refinement
- Discussion

mimeType.rdf Replacement

- Add a new file (e.g. nsHandlerService-json.js) for JSON back-end implementation.
  - JSON store (HandlerStore.jsm) is also needed in this implementation
- Will have a new CID for JSON backend implementation.
- Use preference to decide the backend(RDF/JSON)
- For existing testcase,
  - uriloader/exthandler/tests/unit/test_handlerService.js
  - Need to test JSON backend as well

New testcase

- Add one more testcase for JSON backend.
  - Setup: Add a test rdf file in the test folder. Copy this file into profile folder before starting the test.
  - Procedure:
    - import data from rdf file, which means we will generate a corresponding JSON file.
    - Do “Datastore Utils” testing
      - add_task(function* testHasValue() {};
      - add_task(function* testGetValue() {};
      - add_task(function* testSetValue() {};
      - add_task(function* testRemoveValue() {};
    - The datastore utils may not the same as what we have in RDF backend.

API Refinement

- Synchronous to Asynchronous Study
- Directly Access mimeTypes.rdf
- Improve with nsIPermissionManager

nsIHandlerService APIs

nsIHandlerService.idl

nsIHandlerService is a service used for accessing the mimeTypes.rdf.

```
void store(in nsIHandlerInfo aInfo)
void remove(in nsIHandlerInfo aInfo)
nsISimpleEnumerator enumerate()
boolean exists(in nsIHandlerInfo aInfo)
ACString getTypeFromExtension(in ACString aFileExtension)
void fillHandlerInfo(in nsIHandlerInfo aInfo,
                     in ACString aOverrideType)
```

nsIHandlerInfo

Consumers of nsIHandlerService

Consumer	API
WebContentConverter.js	store()
PdfJs.jsm	store()
dialog.js	store()
application.js	store() enumerate()
nsExternalHelperAppService.cpp	exists() fillHandlerInfo() getTypeFromExtension()

Usage Study: store()

store() in WebContentConvertor.js

```
455            handlerInfo.alwaysAskBeforeHandling = true;
456
457            let hs = Cc["@mozilla.org/uriloader/handler-service;1"].
458                     getService(Ci.nsIHandlerService);
459            hs.store(handlerInfo);
460          }
461      };
```

Call nsIHandlerService::store() then do other thing

The result of store() is never checked

store() can be asynchronous

Usage Study: enumerate()

[enumerate() in applications.js]

```
1112     * Load the set of handlers defined by the application datastore.
1113     */
1114    _loadApplicationHandlers: function() {
1115      var wrappedHandlerInfos = this._handlerSvc.enumerate();
1116      while (wrappedHandlerInfos.hasMoreElements()) {
1117        let wrappedHandlerInfo =
1118          wrappedHandlerInfos.getNext().QueryInterface(Ci.nsIHandlerInfo);
1119        let type = wrappedHandlerInfo.type;
1120
1121        let handlerInfoWrapper;
1122        if (type in this._handledTypes)
1123          handlerInfoWrapper = this._handledTypes[type];
1124        else {
1125          handlerInfoWrapper = new HandlerInfoWrapper(type, wrappedHandlerInfo);
1126          this._handledTypes[type] = handlerInfoWrapper;
1127        }
1128
1129        handlerInfoWrapper.handledOnlyByPlugin = false;
1130      }
1131    },
```

_loadApplicationHandlers is called when opening application panel

Performance probably is not an issue

enumerate() can be asynchronous

Usage Study: exits(), fillHandlerInfo(), getTypeFromExtension()

nsExternalHelperAppService.cpp

These APIs are used in page loading flow.

Docshell needs to find the proper content handler for handling the content.

Searching mimeTypes.rdf is one step of finding algorithm and it can’t go to next step before checking the searching result.

These APIs can not be asynchronous.

Synchronous to Asynchronous Conclusion

API	Asynchronous
store()	O
remove()	O (No consumer for this API)
enumerate()	O
exits()	X
fillHandlerInfo()	X
getTypeFromExtension()	X

Directly Access mimeTypes.rdf Study

HelperApps.js

nsIHandlerSerivce::store() cannot save the file extension information into mimeTypes.rdf

file extension is an attribute of nsIMIMEInfo which is a child of nsIHandlerInfo. However, nsIHandlerService::store() only saves the attributes in nsIHandlerInfo.

Solutions:

1. Handle the attributes in API
2. Move the attributes into nsIHandlerInfo

Improve with nsIPermissionManager

bug1270416 is suggested to implement with nsIPermissionManager

1. Needn't implement in the complicated nsIHandlerInfo
2. Provides an unified UI for user to control the permission.

Issues:

1. Saving information in two different places
   mimeTypes.rdf (handler's information)
   nsIPermissionManager (handler's permission)

   What we move out from mimeTypes.rdf by using nsIPermissionManager?

Discussion

- Scope? JSON backend and ASYNC API call at the same time?
  a. RDF backend + SYNC API (/ ASYNC API ?)
  b. JSON backend + ASYNC API only?
- Conclusion:
  a. Keep the API as SYNC first. However, we do read synchronously and do write asynchronously in JSON store(similar to bug 853539).
  b. JSON store (HandlerStore.jsm) and a stub of the new nsIHandlerService implementation (nsHandlerService-json.js) can be developed in parallel.
  c. While development, we will removed useless API and revise the API if needed.
  d. The function of JSON store may not the same as data utils of RDF backend. Will find the efficient way to access JSON format and fit the need of new nsIHandlerService implementation.