Altair

HyperWorks

Altair HyperWorks Collaboration Tools

HWCT-1500: Profiles

altairhyperworks.com

HWCT-1500: Profiles

HyperWorks Collaboration Tools profiles are a collection of content types, parsers and customizations using HyperWorks Collaboration Tools APIs and tcl/tk scripting. In this tutorial, you will learn how to create a Profile for managing Microsoft Office documents within a HyperWorks Collaboration Tools library.

Background Information

Content definitions

Content definitions encapsulate information about the types of data and information managed by HyperWorks Collaboration Tools and viewable within the **Organize** browser. This information includes:

- Metadata associated with the content
- Dependencies between the metadata
- Relationships that can be created between the content type and others

Any type of information or file can be managed by HyperWorks Collaboration Tools, provided it can be described using the appropriate content definition format.

Content definition xml schema

Content definitions are created using simple `.xml` files, conforming to the HyperWorks XML Schema. Note that HyperWorks Collaboration Tools does not have a schema validating XML parser; hence, it is the responsibility of the publisher to make sure the content type XML description is validated externally.

- At the root of the document is the element ContentType, with name as a mandatory attribute. It contains a list of metadata.
- Each metadata has name and type as required attributes. There are many other attributes, listed below, and best seen by example.
- If the "mandatory" attribute is true, then this attribute must have a value when the content is published to the library. In this case, it is good practice to specify a "defaultvalue".
- Metadata may have a set of valid values.

Metadata attributes

- name (mandatory - string): name of metadata.
- type (mandatory - string): metadata value type. Should be one of string, int, float, boolean.
- dname (boolean): if it is true then the metadata is used as name for the related content.

> **Note:** dname can only be true for one metadata per content definition.

- editable (boolean): if it is true then the metadata is editable.
- displayable (boolean): if it is true then the metadata is displayed.
- mandatory (boolean): if it is true then the metadata is required to be filled by you when creating content.

- displayName (string): metadata display name.
- defaultvalue (string): metadata default value.
- category (string): metadata category.

Parser

A parser automatically extracts information from files when they are added to a library. The extracted information is stored as content metadata, or as indexed metadata, and can be used when searching for contents in the Organize browser. Searches can be metadata-based or full-text based.

Step 1: Update the librarypreferences.xml file

A typical profile consists of the following:

- Install.tcl → To install the profile
- libraryprofiles.xml → Contains path to Content definitions, parsers and custom integration `.tcl` files.
- Main folder → Contains custom scripts, content definition `.xml` files and parser `.tcl` files.

1. Copy this start package zip file (`OfficeProfile_Raw.zip`) from the Altair HyperWorks Enterprise `hwe.zip` directory and extract it to a convenient location, for example `C: /OfficeProfiles`/. The start package contains a generic install `.tcl` file that does not need any editing. Now you will take a look at the `libraryprofiles.xml` file. The image below shows a typical `libraryprofile.xml` file.

```
<?xml version="1.0" encoding="UTF-8"?>
<LibraryProfiles schemaVersion="1">
  <!-- Profile name should be given lower case.
  Profile names will be compared by using case insensitive string comparisons. -->
  <LibraryProfile name="mat" displayname="Material Profile">
    <ContentDefinitions>
      <ContentDefinition path="@INST@/Material_Lib/Definitions/MaterialFile.xml" />
    </ContentDefinitions>
    <!-- Integration Module should implement profile_load and profile_unload procs -->
    <Module name="Mat Integration" lang="tcl" module="MatIntegration"
      namespace="::custom::integration::mat" path="@INST@/Material_Lib/MatIntegration" />
    <Parsers path="@INST@/Material_Lib/parsers.xml" />
    <Preferences path="@INST@/Material_Lib/preferences.xml" />
  </LibraryProfile>
</LibraryProfiles>
```

Edit the following information:

2. To change the profile name edit the line, as shown below:

```
<LibraryProfile name="office"  displayname="MSOffice Profile">
```

3. Edit the path to content definition, as shown below:

```
<ContentDefinitions>
    <ContentDefinition path="@INST@/Profiles_Main/ContentDefinitions/OfficeDocument.xml" />
</ContentDefinitions>
```

4. Edit the custom integration information, as shown below. This information will be used by the core to point to appropriate tcl procedures which are used to perform custom operations in the Organize browser.

```
<Module name="Office Integration" lang="tcl" module="officeIntegration"
namespace="::custom::integration::office" path="@INST@/Profiles_Main/OfficeIntegration1.0" />
```

5. Edit the path to the `parser.xml` file, which points to parser tcl files. You will look into `parser.xml` in the following steps.

```
<Parsers path="@INST@/Profiles_Main/parsers.xml" />
```

6. Edit the path to the preferences file as shown below:

```
<Preferences path="@INST@/Profiles_Main/preferences.xml" />
```

7. Save your updated `librarypreferences.xml` file.

Step 2: Create content definition for office documents

Now you will create content definition for office documents. The folder \Profile_Main\ContentDefinitions contains all the content definition files of a profile. Since you are creating only one content definition for this profile the start package has only one content `.xml` file. A profile can have multiple content definitions.

1. Rename the `.xml` file under Profile_Main\ContentDefintitions to `OfficeDocument.xml`.

2. Open the `OfficeDocument.xml` file in a text editor, change the file types and save. This informs the core that this content type has three types of files, Word, Excel and PowerPoint.

```
<tns:ValidVal>Excel</tns:ValidVal>
<tns:ValidVal>Word</tns:ValidVal>
<tns:ValidVal>PowerPoint</tns:ValidVal>
```

Now take a look at the `parser.xml` which points to parser code in tcl.

3. Open the `\Profiles_Main\parser.xml` in a text editor and edit the information, as shown below. The module informs the core which tcl package to use and the namespace is to inform the core which tcl namespace to use to parse the data.

```
<Parser name="Office Parser" lang="tcl" module="officeParsers"
namespace="::custom::integration::office::officeFileParser"/>
```

Step 3: Parser TCL code

In the previous step you edited the `parser.xml` file to point to the appropriate tcl code. Now you will take a look at the parser tcl code. For simplicity the start package code does not need any editing.

1. Open `\Profiles_Main\OfficeParser1.0\OfficeFileParser.tcl` in a text editor.

   `Procedure ::custom::integration::office::officeFileParser::impl_canParse` informs the core whether a file can be parsed. You can add your own logic to decide if the file can be parsed.

   A return value of 0 informs the core that the file cannot be parsed. Under such a situation the file will be added as a general file in the library.

   A return value of 1 informs the core that this file can be parsed. In the code below you will check for the file extensions and if they are `.docx`, `.pptx` or `.xlsx` then set the return as 1. Otherwise it will return 0.

   `Procedure ::custom::integration::office::officeFileParser::impl_Parse` parses the information and sets the parsed information as metadata on the content being added to the library. In the code below the display name is set as the file name by using the API `ParsedDocument_addMetadata`. For more information, refer to the HyperWorks Collaboration Tools API documentation.

```
#Package Name
package provide officeParsers 1.0

# Namespace eval
namespace eval ::custom {}
namespace eval ::custom::office {}
namespace eval ::custom::office::integration {}

namespace eval ::custom::integration::office::officeFileParser {}

#Procedure which informs the core that this file can be parsed.
#Return 0 --> Fail
#Retunr 1--> Pass
proc ::custom::integration::office::officeFileParser::impl_canParse {filen} {

set filepath [file tail "$filen"];
set fileextension [lindex [split $filepath "."] end]

if {$fileextension == "docx" || $fileextension == "pptx" || $fileextension == "xlsx"} {

return 1;
}
 return 0;
}


#Procedure which passes the parsed information.
#Return 0 --> Fail
#Retunr 1--> Pass
proc ::custom::integration::office::officeFileParser::impl_parse {filen documentp} {

  ParsedDocument_setType $documentp "office"

# To set the displayname as file name.
  set metadataName "displayName"
  set value [file tail "$filen"]

  set vv [hwVariant];
  $vv setString $value;
  set dmd [new_DocumentMetadata  $metadataName 1 $vv];
  ParsedDocument_addMetadata $documentp $dmd;

  return 0;

}
```

Step 4: Customize the Organize browser

Now you will work on the customization of the Organize browser. Via profiles you can add your own custom context menus, toolbar icons and also have custom procedures which will be run before or after a lifecycle operation. For example, if you want to display a message saying "Check in successful" after a content is checked in you could add this message as part of the post check-in call. Refer to the HyperWorks Collaboration Tools Profiles documentation for more details.

In this exercise you will edit an existing context menu procedure to show a message when it is clicked from the Organize browser.

1. Open the file `Profiles_Main\OfficeIntegration1.0\officeIntegration.tcl` in a text editor.
2. Search for `procedure ::custom::integration::office::CtxMenuAddCustomItems`. The core will look at this procedure to add the custom context menus.
3. Add command `-command "::custom::integration::office::ShowMessage $obj` as shown below:

```
proc ::custom::integration::office::CtxMenuAddCustomItems {objBrowser menu obj} {
  set lsSelectedObjs [::hwe::browser::Browser $objBrowser cget -selectedObj];

  $menu add command -label "Name : [Object_GetName $obj]" \
    -compound left -image [::hwt::CacheImage "toolkit-16.png"] \
    -command "::custom::integration::office::ShowMessage $obj" ;
}
```

Note that you are calling another procedure `::custom::integration::office::ShowMessage` as an action after the custom context menu is clicked. Copy the code below in the `officeIntegration.tcl` file. It does not matter where the lines below are pasted in the `.tcl` file.

```
proc ::custom::integration::office::ShowMessage {obj} {

tk_messageBox -message "File Name is [Object_GetName $obj]"

}
```

Step 5: Install a profile

1. To install a profile run `C:\OfficeProfile\install.tcl` in HyperWorks Desktop session. Once the installation is complete, an Installation Complete message is displayed in the tcl command window. Restart the Altair HyperWorks session after the Profile installation is complete.

Step 6: Create a new library using the office profile

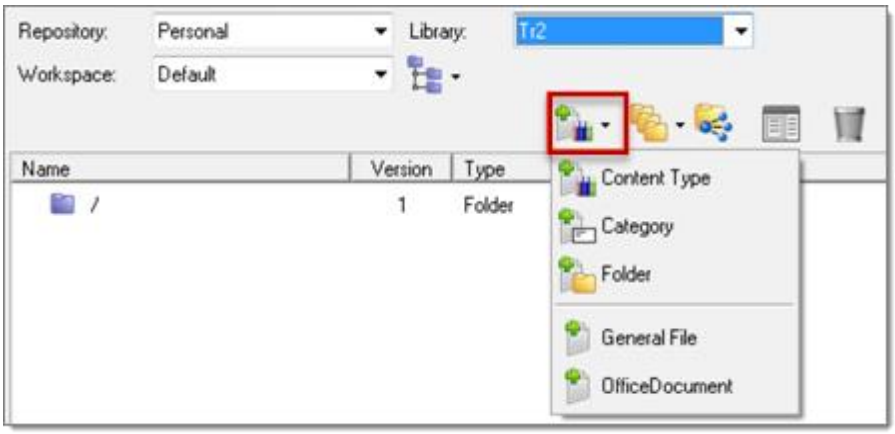
1. From the Organize browser select the **Repository** type ***Personal***.

2. Select the combo box from the **Library** and select ***New Library***. A **New Library** dialog displays. Enter the **Name** and select the path where you want to create the library.
3. Click the ***Profile*** field to see the list installed and select ***Office Profile***. Leave the **Workspace** as the default.
4. Click ***OK*** to create a new library.

Step 7: Add a new content to the office library

In the previous step you created a new library using the Office profile. Now you will add a PowerPoint document to this library.

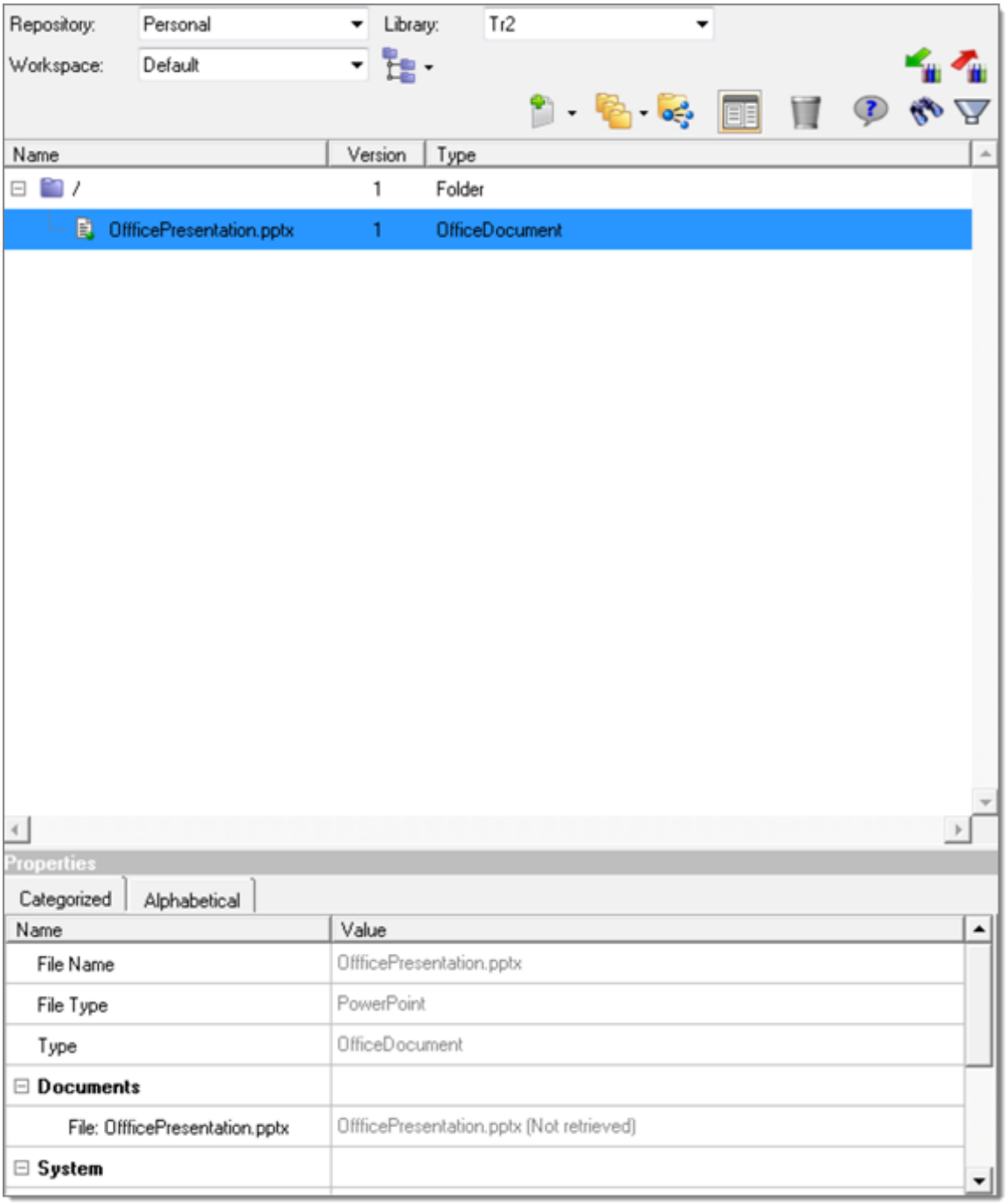
1. Select ***OfficeDocument*** from the **New Content** toolbar. A **New Office Document** dialog displays.



2. Click in the **Value** area of the **Documents** row, click the ***File Browser*** and browse to and select an Office file.
3. Click ***Open***. Note that the **File Name** is automatically populated.
4. From the **File Type** row select the appropriate file type.

Note: As an enhancement to the previously created parser, you could update the parser to set this value automatically.

5. Click ***Save*** to add the PowerPoint file to the library.



Step 8: View the custom context menu

Now you will review the action of adding a new custom context menu.

1. Select the ***PowerPointFile*** content you added to the library earlier.
2. Right-click the above selection to view the context menus. Note that the custom menu appears on the top of a list of available context menus.
3. Select the custom context menu to display the filename. Click ***OK*** to return to the Organize browser.

Summary

In this tutorial you:

- Created a new profile
- Added a new content definition
- Added a new parser
- Customized the context menu to display the file name

HyperWorks Collaboration Tools profiles can be used to do so much more. Refer to the HyperWorks Collaboration Tools Profiles and API documentation for more information.