ICT MyMedia Project
2008-215006

Catalog Importer Quickstart

17 June 2009

Public Document

1 Contents

2 Target Audience 4

3 Introduction 4

4 Configuration 4

5 Creating a Catalog Importer 4

6 Importing Data 5

6.1 CatalogItem Class 6

6.1.1 Crid 6

6.1.2 isGroup, isOrdered and GroupType 6

6.1.3 Title, Description and Promotional Text 6

6.1.4 Genre 7

6.1.5 Keyword 7

6.1.6 Audio Language and Caption Language 7

6.1.7 Credits 8

6.1.8 Related Material 8

6.1.9 Other CatalogItem Properties 8

6.1.10 Locator 9

6.2 User Class 9

6.2.1 Other User Properties 9

6.3 RatingAction Class 10

6.3.1 User - Item Tags 10

6.3.2 Other User - Item Properties 10

7 References 10

Project acronym: MyMedia

Project full title: *Dynamic Personalization of Multimedia*

Work Package:	**2**

Document title:	**Catalog Importer Quickstart**
Version:	**N/A**
Official delivery date:	**N/A**
Actual publication date:	N/A
Type of document:	Open Source Software and Documentation
Nature:	Public

Authors:	Chris Newell, BBC
Approved by:	N/A

2 Target Audience

The intended target audience of this document are consumers of the MyMedia project software framework.

See the MyMedia softare license, included at the root of the distribution, for terms of use.

3 Introduction

A Catalog Importer is a software module designed to import and manage the catalog data held by a Recommender System. It provides an interface between the external metadata signaling format (e.g. TV-Anytime [1]) and the internal data format used by the Recommender Framework itself. The primary type of data supplied by Importers is content metadata. However, Importers may also be used to load user profile information and user - item feedback.

Each Importer is given a unique module identifier and this is recorded together with the time and date of each entry in the catalog. The source of each catalog entry can therefore be identified from its moduleId. This module allows changes and additions to the catalog to be tracked and managed.

4 Configuration

The MyMediaRecommenderQuickstart document contains information on configuring the database and other options using the MyMediaConfiguration options.

5 Creating a Catalog Importer

The following namespaces should be referenced by Importers:

```
MyMediaProject.RecommenderSystem.Core;

MyMediaProject.RecommenderSystem.Core.Databases;
```

Importers should sub-class the `CatalogItemFactory` class. After creating an instance of `CatalogItemFactory` they should call the `Initialize()` method.

The `CatalogItemFactory.DataContext` field provides access to the `CatalogItemStoreDataContext` instance which is used for all interaction with the catalog database. In this document this will be referred to as: `dataContext`.

The first task of an Importer is to test for the existence of the framework database and to create a new database if not found. This can be done with the methods:

```
dataContext.DatabaseExists();
```

```
    dataContext.CreateDatabase();
```

The Importer should then use the following method to create a `Module` that is used to identify itself in all import operations:

```
Module module = dataContext.CreateModule(...
```

6 Importing Data

This section describes the data model of the recommender system as it appears externally and shows how an Importer can enter data using the classes provided by the framework. The primary entities are CatalogItems, Users and RatingActions. Table 1 lists the fields of these classes most relevant to an Importer. Note that Importers do not have to support all of these fields.

CatalogItem	
Field	**Count**
Crid	1
isGroup	1
isOrdered	1
GroupType	1
Title	n
Description	n
PromotionalText	n
Genre	n
Keyword	n
AudioLanguage	n
CaptionLanguage	n
Credit	n
RelatedMaterial	n
Property	n
Locator	n

User	
Field	**Count**
UserName	1
FirstName	1
LastName	1
Email	1
IsDisabled	1
Guid	1
LastLogOn	1
Property	n

RatingAction	
Field	**Count**
ItemId	1
UserId	1
RatingValue	1
TagValue	1
OtherActionProperty	1

Table 1. Relevant fields of the CatalogItem, User and RatingAction classes

6.1 CatalogItem Class

The CatalogItem class represents a single item of content. Use the following procedure to create a new item:

```
CatalogItem item = dataContext.CreateItem();
item.Module = module;
dataContext.CatalogItems.InsertOnSubmit(item);
```

The remainder of this section will describe the individual fields of the Item class listed in Table 1 and illustrate how they are utilized. After the configuration of an item is complete the following method should be called to add the CatalogItem to the database:

```
dataContext.SubmitChanges();
```

6.1.1 Crid

Each item entered by an importer can be identified by a string called the Content Reference Identifier (CRID) which must be unique within the catalog. The format of this string is unspecified but may conform to the specification of a TV-Anytime CRID [1]:

crid://<authority>/<data>

where:

<authority>	is a registered Internet domain name
<data>	is a free format string that is URI compliant

The CRID field allows the external identity of each item in the catalog to be preserved.

6.1.2 isGroup, isOrdered and GroupType

The isGroup field is a boolean flag which identifies whether the CatalogItem represents a single item of content or a group of content items (e.g. a series of programs). The default value is false.

If the isGroup field is set to true the isOrdered and GroupType fields are relevant. The isOrdered field can be set to indicate that the order of the items within the group is important. The GroupType field can be used to distinguish different types of group (e.g. series, season, brand). The coding of the GroupType is determined by the Importer. The default values of the isOrdered and GroupType fields are false and null respectively.

6.1.3 Title, Description and Promotional Text

Each CatalogItem may have one of more Titles, Descriptions and PromotionalTexts. Descriptions are generic descriptions of an item whereas PromotionalTexts are descriptions specific to a particular showing or scheduling e.g. a premiere showing.

In each case there may be multiple instances in different languages or with different lengths. The language must be specified in each case.

Use the following procedure to add a title to an item. A similar procedure is used to add a description:

```
ItemTitle itemTitle = dataContext.CreateTitle(...
itemTitle.Module = module;
item.AddTitle(context, itemTitle);
```

Use the following procedure to add a PromotionalText:

```
itemPromotionalInfo = new ItemPromotionalInfo();
itemPromotionalInfo.Created = DateTime.Now;
itemPromotionalInfo.Modified = DateTime.Now;
itemPromotionalInfo.Module = module;
itemPromotionalInfo.Language = dataContext.CreateLanguage(...
itemPromotionalInfo.PromotionalText = promotionalText;
item.AddPromotionalInfo(context, itemPromotionalInfo);
```

6.1.4 Genre

Each CatalogItem may have one of more genres. Each genre has a human-readable name and should be identified by a unique URI. The genre classification scheme is decided by the importer (e.g. using the TV-Anytime genre classification schemes [2]). Use the following procedure to add a genre to a CatalogItem:

```
Genre genre = dataContext.CreateGenre(genre_name, genre_uri);
genre.Module = module;
item.AddGenre(dataContext, genre, module);
```

6.1.5 Keyword

Each CatalogItem may have one or more keywords associated with it. The language of each keyword must be specified. Use the following procedure to add a keyword to a CatalogItem:

```
Keyword keyword = dataContext.CreateKeyword(...
keyword.Module = module;
item.AddKeyword(dataContext, keyword, type, module);
```

6.1.6 Audio Language and Caption Language

The AudioLanguage and CaptionLanguage fields are used to list the languages of the available audio and caption streams.

It is possible to signal whether audio or caption streams provide additional information for the visually or hearing impaired using the supplemental flag. The default value of the supplemental flag is false. It is also possible to indicate whether caption streams are "closed" i.e. delivered separately and not embedded in the video signal. The default value of the closed flag is false.

Use the following method to add an AudioLanguage to a CatalogItem:

```
item.AddAudioLanguage(...
```

Use the following procedure to add a CaptionLanguage to a CatalogItem:

```
CaptionLanguage captionLanguage = new CaptionLanguage();
captionLanguage.Language = dataContext.CreateLanguage(...
captionLanguage.IsClosedCaption = true;
captionLanguage.IsSupplemental = true;
captionLanguage.Module = module;
item.AddCaptionLanguage(dataContext, captionLanguage);
```

6.1.7 Credits

Each CatalogItem may have one or more associated credits. Credits represent a person and/or a fictional character and the role they fulfill (e.g. actor, director etc). Each role has a human-readable name and should be identified by a unique URI. The role classification scheme is determined by the importer (e.g. using the TV-Anytime "TVARoleCS" classification scheme [2]).

Use the following procedures to create Credits and Roles:

```
Credit person = new Credit();
person.GivenName = givenName;
person.FamilyName = familyName;
person.Created = DateTime.Now;
person.Modified = person.Created;
person.Module = module;

Credit character = new Credit();
character.GivenName = givenName;
character.FamilyName = familyName;
character.Created = DateTime.Now;
character.Modified = character.Created;
character.Module = module;

Role role = dataContext.CreateRole(role_name, role_uri);
role.Module = module;
```

Use either of the following methods to add Credits to a CatalogItem:

```
item.AddCredit(dataContext, person, character, role, module);

item.AddCredit(dataContext, person, role, module);
```

6.1.8 Related Material

Each CatalogItem may be associated with one or more items of related material (e.g. a still image, web page, or program transcript). Each item of related material is identified by a URL. The nature of the relationship is signaled by a human-readable name and should be identified by a unique URI. The relationship classification scheme is decided by the importer (e.g. using the TV-Anytime "HowRelated" classification scheme [2]). Each relationship type has a human-readable name and is identified by a unique URI.

```
Relationship relationship = dataContext.CreateRelationship(name, uri);
relationship.Module = module;
item.AddRelationship(dataContext, relationship, relatedItemUrl, module);
```

6.1.9 Other CatalogItem Properties

Any CatalogItem properties that are not specifically supported by the data model can be signaled using a generic Property mechanism which specifies a property name / property value pair. Use the following procedure to add a Property to a CatalogItem:

```
Property property = dataContext.CreateProperty(propertyName, propertyValue);
property.Module = module;
item.AddProperty(dataContext, property);
```

6.1.10 Locator

Each CatalogItem may have one or more locators. A locator can either be an on demand locator or a scheduled locator (which identifies the location of an item of content in a broadcast stream). Each locator may identify a Service to which the item belongs and specifies the time window over which the item is available. A locator may also specify the number of audio channels (e.g. mono, stereo, surround) and the video aspect ratio. The coding scheme for the aspect ratio field is determined by the importer.

Use the following procedure to create a Service if required:

```
Service service = dataContext.CreateService(name, serviceUrl, owner);
service.ModuleId = moduleId;
service.Genre = genre;
```

Use the following procedure to add a Locator to a CatalogItem:

```
Locator locator = dataContext.CreateLocator(url);
locator.IsScheduled = false;
locator.Service = service;
locator.AvailabilityStart = startDateTime;
locator.AvailabilityEnd = endDateTime;
locator.DurationInSeconds = 1800;
locator.AspectRatio = 1;
locator.AudioChannels = 2;
locator.Module = module;
item.AddLocator(dataContext, locator);
```

6.2 User Class

The User class represents a user of the recommender system. Table 1 lists the fields most relevant to an Importer. However, none of these fields are mandatory. The UserName is intended to identify the user externally to the MyMedia system and should be a unique string. The Guid is allocated automatically by the Framework.

Use the following procedure to create a new user:

```
User user = new User();
user.ModuleId = moduleId;
user.UserName = username;
(set other properties as required...)
context.Users.InsertOnSubmit(user);
```

After the configuration of a User is complete the following method should be called to add the User to the database:

```
dataContext.SubmitChanges();
```

6.2.1 Other User Properties

Additional User properties can be signaled using a generic Property mechanism which specifies a property name / property value pair. Use the following procedure to add a Property to a User:

```
Property property = dataContext.CreateProperty(propertyName, propertyValue);
property.Module = module;
user.AddProperty(dataContext, property);
```

6.3 RatingAction Class

The RatingAction class is used to represent user - item feedback. Table 1 lists the fields most relevant to an Importer which may wish to import bulk feedback data for training purposes.

The class can be used to represent an actual rating by a specific user for a specific CatalogItem. Use the following procedure to create a RatingAction:

```
RatingAction feedback = new RatingAction();
feedback.UserId = user.UserId;
feedback.ItemId = item.ItemId;
feedback.RatingValue = rating;
dataContext.UserActions.InsertOnSubmit(feedback);
```

The RatingValue should be a floating point value between -1 and 1. It is possible but not necessary to specify a rating value if the RatingAction is used to represent a Tag and/or OtherActionProperty as described in the following sections.

After the configuration of a RatingAction is complete the following method should be called to add the RatingAction to the database:

```
dataContext.SubmitChanges();
```

6.3.1 User - Item Tags

The RatingAction class can also be used to add user – item tags to the database. Use the following procedure to add a tag to a RatingAction:

```
feedback.TagValue = new TagValue("Tag text");
```

6.3.2 Other User - Item Properties

Additional User - Item properties can be signaled using a generic mechanism which specifies a property name / property value pair. Use the following procedure to add an OtherActionProperty to a RatingAction:

```
feedback.OtherActionProperty = new OtherActionProperty(propertyName, value);
```

7 References

[1] ETSI TS 102 822-4 (V1.3.1): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 4: Content referencing".

[2] ETSI TS 102 822-3-1 (V1.4.1): "Broadcast and On-line Services: Search, select, and rightful use of content on personal storage systems ("TV-Anytime"); Part 3: Metadata; Sub-part 1: Phase 1 - Metadata schemas".