



Expedition Development Concept

Expedition is an extension for Windows Explorer that provides file system cataloging with file type itemization. A recursive snapshot of any file system can be saved as a data structure for browsing and searching. File system auditing is implemented to record the history of file backup copies. File system objects may be assigned to an item with a description and keywords. The item entity type controls the metadata profile contained in data table.

Items support hierarchal categorization and may have one or more associated files. Categories are defined by a static node tree and are linked with items. Item entity types provide the logic to import, update and render each item in the interface.

Expedition User Interface

Expedition consists of three main navigation items. A tree view is provided on the left for node navigation. A list view in a tab control is on the right to display the linked items. An item editor is dock able below to edit one or more selected items and their respective entity details.

The tree view contains special root nodes. Each root node type may be enabled or disabled in the configuration settings and saved as browsing styles. Base nodes dynamically load all node children on demand. Preload can be enabled to postpone the population of child nodes until the node is expanded.

Additional list views may be opened for multiple simultaneous item browsing. The selected tree node controls the items in the active list view. The list view displays the linked item list for the selected node and may also contain the associated child nodes at the beginning and an optional parent node link at the top. Filters are provided to limit the items displayed based on property values. The item view lists the items associated with the selected category node. Recursion is supported to display all offspring items from selected node. Multiple item views are supported based on the selected node entity type. Each item view provides small icon, large icon, detail and thumbnail display modes.

The item editor is dynamically configured based on the selected item(s) in the list view. Itemizations are renormalized whenever created or updated. Identical itemization records will be consolidated based on entity type to single item record with all associated file records. Selecting multiple items allows for batch edit of the common properties including categories.

Expedition Navigation Tree

Category Tree displays each category tree recursively using the node name and details columns. The entity type defaults to a category node but can be overridden to provide customized entity behavior. Tree nodes and list view items contain the category id in the name property and use the node name and details columns for data binding. The entity type id is assigned to the node tag property to override the style definition.

Dynamic Items displays a base node for each entity having a dynamic tree node. The base node entity type is used to determine the child item entity filter. The child node entity types are used to determine the available recursive queries. A node with no children will return a list of items. A node with a single child will return a list of items related to the current node. A node with multiple children will contain another base node for each child node entity type.

My Computer displays a live file system for read only browsing with each internal disk is enumerated as base nodes (Internal, External, Network, Optical). Child folders and files are dynamically cached when a node is selected. Each child node is loaded from the file system and synchronized to the offline cache to validate data integrity.

Itemized Media displays base nodes for each type of offline media (CD, DVD, External, Network, Remote) loaded from the root table records. Child folders and files are loaded from cache. Data integrity is validated if the file system is online. Tree nodes and list view items contain the entity id in the name property and use the entity record schema for data binding. Only Root, Path and File entity types are allowed as children.



Expedition Entity Behavior

A typed data set used as a static cache for all instances of Expedition. Data may be loaded or merged from any supported data store. All changes are cached until persistence is requested.

Each entity type defines the type of data itemized in the system. The entity object controls how to import, update and render a type of item in the system. Items may be related to a database table and share a common primary key. Additional entity types may be created to provide custom import, update and render operations.

The base entity class defines the required logic for common data navigation and itemization. This class is used to create the root tree nodes for the tree view. It is also responsible for creating the base tree nodes for any given root tree node. The category tree and item profile logic is also implemented here for simplicity.

Each custom entity class is inherited from the base entity class to preserve the object interface. A custom entity class is created for each type of node that is supported. Custom entities based on a typed data table require additional logic to populate and synchronize the metadata details.

A custom entity class exists to handle file system related objects. A single class contains the logic for browsing both offline and online file system data. Each drive node created will be checked against the root table for serial number and device name matches. Folders and files will be queried from the file system and matches will be made with the path and file tables. Matches will be associated with a node using a temporary item record.

Each node may have child nodes and items. A static node will render the list of child nodes associated by the tree column and a list of items associated by the link table. These nodes represent a fixed tree structure used for categorization and navigation. The node name and details columns are used to display the node properly while the entity type controls the node style. Dynamic nodes can provide automated foreign key filtering to create child nodes and items. Each child node represents related late bound child data tables. The child entity type is used to dynamically create the data relationship to return items.



Expedition Pseudo Code

1. Load root nodes in expedition tree
 - a. My Computer Enumerate local drive letters
 - b. Category Tree Load categories in category tree
 - c. Itemized Media Enumerate grouped root records by device
 - d. Dynamic Items Create base nodes for each dynamic root node
2. Select expedition tree node
 - a. Check itemization of node stored in node name
 - b. Use node entity type to refresh node style
 - c. Update tree view status for selected node
 - i. Set entity item specific context menus
 - ii. Update status bar with entity item details
3. Expand expedition tree node
 - a. Check itemization of node stored in node name
 - b. Use node entity type to refresh node style
 - c. Check loaded status of selected node
 - d. Preload children if node is not loaded
 - e. Use child node entity types to refresh styles
4. Refresh expedition list view
 - a. Reset list view style control and clear items
 - i. Preserve a cached history of recent item lists and styles
 - ii. Reset image lists based on view style and options
 - b. Add special node items to list view
 - i. Parent node selector
 - ii. Child node selectors
 - c. Populate list view with linked items
 - d. Use item entity type to refresh node style
5. Select expedition view item
 - a. Save previous item changes
 - b. Reset item edit control form
 - i. Find and match item profile values
 - ii. Load entity forms for selected items
6. Active expedition view item
 - a. Active tree change event if entity nodes
 - b. Launch associated application for entity items



Expedition Class Interfaces

Expedition Entity

Create(entity, data)	creates an typed entity record
Itemize(item, name, details)	creates item profile for entity
GetData(item)	returns the entity data table row
SetData(item, data)	updates late bound entity data
Delete(item)	removes all related item records
GetKeywords(item)	returns list of associated keywords
SetKeywords(item, string[])	sets list of assigned keywords
GetCategories(item)	returns category nodes collection
GetCategories(item, depth)	returns recursive category nodes
SetCategories(item, node[])	sets list of assigned item categories
GetNodes(item)	returns list of child nodes
GetNodes(item, type)	
GetItems(entity, item)	returns list of linked items
GetItemStyle(item, style)	applies the specified node style

Expedition Tree

CreateItemizedMedia()	creates itemized media root node
CreateDynamicItems()	create dynamic items root node
CreateCategoryTree()	creates category tree root node
CreateMyComputer()	creates my computer root node
NodeLoaded(node)	applies default entity item style
NodeSelected(node)	auto loads selected node children
NodeExpanded(node)	populates children of selected node
NodeCollapsed(node)	refreshes item entity style for node
OnItemSelected()	event handler for selected node changes

Expedition View

ItemLoaded(item)	applies default entity item style
ItemSelected(item)	auto loads selected item details
OnItemSelected()	event handler for selected item changes
OnItemExecuted()	event handler for item event execution



Expedition Item System Schema

Entity Table – provides data integrity for late data binding

EntityName	text description of entity type
EntityTable	name of late bound data table
File:	Root, Path, File, Copy
Audio:	Artist, Album, Track
Media:	Group, Shoot, Photo, Video

Item Table – provides searching profile for itemized files

EntityID	foreign key to entity types
RatingSum	total value from multiple ratings
RatingCount	count of unique rating events
StyleName	item visualization style
ItemName	default item name or format string
Description	user defined text description
Keywords	user defined keyword list

Link Table – links items as static node children

ItemID	foreign key to item table
NodeID	foreign key to node table

Node Table – hierarchal arrangement of entity typed nodes

EntityID	foreign key to entity types
TreeID	foreign key to parent node
NodeType	node behavior identifier
NodeName	display name for node
NodeDetails	display details for node
Category	Photoshoots
Category	Buildings
Category	Modern
Category	Ancient
Dynamic	Group
Dynamic	Shoot
Dynamic	Photo
Dynamic	Video
Dynamic	Artist
Dynamic	Album
Dynamic	Track



Expedition File System Schema

Root Table – describes the root file system objects

Name	user defined name for root object
Serial	source specific media identifier
Label	physical media drive label
Device	logical system device id
Machine	windows machine name
Description	user defined text details

Type	Path	Name	Serial	Label	Device
1	C:	System	HD01	Latitude	HD01
2	?:	My Photos	DVD01	Photos05	DVD
3	?:*	Directory	EXT01	My Photos	EXT001

Path Table – unique file paths from root objects

RootID	foreign key to root table
RelativePath	relative path from root to files

File Table – unique file items and queries

PathID	foreign key to path table
ItemID	optical foreign key to item table
FileType	file existing group lookup key 0: Static – uses file extension (*.ext) 1: Query – file name is query (*.ext)
FileSize	size of file object in bytes
FileName	name relative to parent
FileCRC	CRC32 checksum value
FileMD5	MD5 hash validation

Copy Table – maintains a link between two duplicate files

CopyID	foreign key to source file entity
FileID	foreign key to duplicate file entity



Expedition Photo System Schema

Group Table – root table for photo system hierarchy

GroupName	user defined name for group
GroupDetails	user defined text details fro group

Shoot Table – provides location and time based photo sets

GroupID	foreign key to group table
ShootTitle	user defined name for shoot
Location	geographical location of shoot
StartDate	date time when shoot began
StopDate	date time when shoot finished

Photo Table – describes individual photos with details

ShootID	foreign key to shoot table
Height	height of image in pixels
Width	width of image in pixels
Title	user defined text title
Data	image metadata text
Taken	date time of photo

Video Table – describes individual videos with details

ShootID	foreign key to shoot table
Height	height of video in pixels
Width	width of video in pixels
Duration	length of video in seconds
Title	user defined text title
Data	video metadata text
Taken	date time of photo

