Zend_Ldap - Extended support - Stefan Gehrig

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

<ac:macro ac:name="unmigrated-inline-wiki-markup"><ac:plain-text-body><![CDATA[

Zend Framework: Zend_Ldap - Extended support(Zend_Ldap_Ext) Component Proposal

Proposed Component Name	Zend_Ldap - Extended support(Zend_Ldap_Ext)
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Ldap - Extended support(Zend_Ldap_Ext)
Proposers	Stefan Gehrig Liasion: Ralph Schindler
Revision	0.1 - 29 March 2008: Opened proposal. 0.1.1 - 3 April 2008: Changed layout to use decks. 0.2 - 13 April 2008: Added Zend_Ldap_Ext::count() method 0.3 - 17 April 2008: Completely reworked the class skeletons to match the current version, I'm working on. 0.4 - 5 July 2008: Changed respository location 0.5 - 10 August 2008: Reworked the proposal as it was fairly outdated 0.6 - 27 November 2008: Updated proposal to match Zend requirements (merging of Zend_Ldap_Ext into Zend_Ldap) 0.7 - 4 December 2008: Updated proposal (includes new features like LDIF support, schema browsing, etc.) (wiki revision: 33)

Table of Contents

1. Overview
2. References
3. Component Requirements, Constraints, and Acceptance Criteria
4. Dependencies on Other Framework Components
5. Theory of Operation
6. Milestones / Tasks
7. Class Index
8. Use Cases
9. Class Skeletons

1. Overview

The existing Zend_Ldap component currently just responds to authentication use cases in all their varieties. There is no posibility to query a LDAP directory service in a unified and consistent way. The current component also lacks core CRUD (Create, Retrieve, Update and Delete) functionality - operations that are crucial to for example database abstraction layers.
This proposals tries to resolve these deficiencies in that it provides a simple two-ply object oriented model to connect to, query and perfom CRUD operations on an LDAP server. The first layer is a wrapper around the ext/ldap functions, spiced up with extended functionality such as copying and moving (renaming in a LDAP context) nodes and subtrees.
The second layer (Zend_Ldap_Node) provides an active-record-like interface to LDAP entries and stresses the tree-structure of LDAP data in providing (recursive) tree traversal methods.
To simplify the usage of the unfamiliar LDAP filter syntax this components proposes an object oriented approach to LDAP filter string generation, which can loosely be compared to Zend_Db_Select.
Usefull helper classes for creating and modifying LDAP DNs and converting attribute values complete this component.
It is important to note, that this proposal is a complete replacement for the current Zend_Ldap component and **does not** break backwards-compatibility.

2. References

- PHP Manual LDAP Functions
- Zend Framework Programmer's Reference Guide - Chapter 21. Zend_Ldap
- RFC2253 - Lightweight Directory Access Protocol (v3): UTF-8 Str
- RFC2254 - The String Representation of LDAP Search Filters

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** replace Zend_Ldap.
- This component **will not** break backwards-compatibility with the existing Zend_Ldap.
- This component **will** provide generic CRUD functions (getEntry(), add(), update() and delete()).
- This component **will** provide generic search functions (search()).
- This component **will** abstracts a LDAP search result resource into a collection class to abstract the cumbersome use of ext/ldap to retrieve attributes in a common format.
- This component **will** provide an object oriented API to LDAP entries that factors in the tree-structure of LDAP.
- This component **will** provide an active-record-like interface to single LDAP entries.
- This component **will** provide an API to manipulate DN strings.
- This component **will** allow for the creation of LDAP filters in an object oriented way.
- This component **will** assist the user in populating and reading LDAP entry arrays.
- This component **will** detect LDAP boolean values ('TRUE' and 'FALSE') and convert them to PHP booleans.
- This component **will** provide a method to populate an LDAP entry array regardless of the PHP variable type.
- This component **will** handle file resources transparently so that it's possible to add the contents of a file to an LDAP attribute.
- This component **will** provide a method to set a LDAP userPassword attribute with SHA1 or MD5 hashed passwords.

4. Dependencies on Other Framework Components

- PHP ext/ldap

5. Theory of Operation

This replacement Zend_Ldap component builds on the foundations layed out by the current Zend_Ldap component and extends its capabilities with methods to query an LDAP directory service and to perform creation, updating, retrieval and deletion operations on the LDAP server. It therefore wraps ext/ldap function calls in an object-oriented interface and unifies result handling. On this layer LDAP data is represented in an array format to allow for round-trips of the data. Helper functions will allow developers to handle LDAP data in a common way; this includes conversion of LDAP date/time attributes, LDAP boolean attributes and the creation of LDAP password attributes. These methods will also help the developer to build consistent LDAP data arrays for use with the different data modification methods of Zend_Ldap and will allow the creation of LDAP DN strings (a lot of these functions are inspired by PEAR::Package::Net_LDAP2).
Query results are encapsulated in Zend_Ldap_Collection which acts as an interface to the LDAP resultset. Zend_Ldap_Collection implemets the common PHP SPL interfaces Iterator and Countable and includes a LDAP entry cache to speed up multiple iterations.
On top of these core functionality Zend_Ldap_Node provides an active-record-like interface to single LDAP entries. Through the use of the SPL RecursiveIterator interface developers can traverse complete LDAP trees recursively with a single foreach()-loop. Zend_Ldap_Node can be extended by deleopers and form a basis for a LDAP data model.
Building LDAP filter strings is unfamiliar to SQL-accustomed developers and not always very intuitive with all its parentheses. Zend_Ldap_Filter proposes an object oriented approach to filter creation which is also inspired by PEAR::Package::Net_LDAP2. With automatic value escaping LDAP filter string creation could be a no-brainer this way.

6. Milestones / Tasks

- Milestone 1: [DONE] Write initial proposal
- Milestone 2: [CURRENT] Review by community

- Milestone 3: [DONE] Checked in at http://zend-ldap.googlecode.com/svn/trunk/
- Milestone 4: [DONE] Review by Zend
- Milestone 5: Component incubated
- Milestone 6: [DONE] Write unit tests
- Milestone 7: Write documentation
- Milestone 8: Component cored

7. Class Index

- Zend_Ldap
- Zend_Ldap_Attribute
- Zend_Ldap_Collection
- Zend_Ldap_Converter
- Zend_Ldap_Dn
- Zend_Ldap_Exception
- Zend_Ldap_Filter
- Zend_Ldap_Filter_Abstract
- Zend_Ldap_Filter_And
- Zend_Ldap_Filter_Exception
- Zend_Ldap_Filter_Logical
- Zend_Ldap_Filter_Mask
- Zend_Ldap_Filter_Not
- Zend_Ldap_Filter_Or
- Zend_Ldap_Filter_String
- Zend_Ldap_Node
- Zend_Ldap_Node_ChildrenIterator
- Zend_Ldap_Node_Collection
- Zend_Ldap_QueryResult

8. Use Cases

Core classes

Filter subpackage

Node subpackage

9. Class Skeletons

Core classes

Filter subpackage

Node subpackage

LDIF subpackage

]]></ac:plain-text-body></ac:macro>
]]></ac:plain-text-body></ac:macro>