Zend_Tree - Andries Seutens

<ac:macro ac:name="note"><ac:parameter ac:name="title">Under Construction</ac:parameter><ac:rich-text-body>
This proposal is under construction and is not ready for review.

When Zend_Tree got proposed (April 2006) the Zend team was very much focused on making enough progress on what they considered the absolutely must-have components.

For that reason, Zend_Tree had to be put aside because even if the Zend team doesn't do the proposal writing and coding, each such component requires a significant amount of energy on their side on reviewing, helping with, and polishing up the component and making it ready for release.

I have received an email from Andi Gutmans, where he tells me that we are at the point now where most of what we labeled "must-have" is starting to stabilize. He asked me if I was still interested in "re-proposing" Zend_Tree, so it could be reconsidered.

This proposal is a revised proposal, based on the feedback that I have received throughout the months, from various contributors.

I have decided to put this proposal in the standard proposal template, as fast as possible, so we can gather feedback from the very start. </ac:rich-text-body></ac:macro>

<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_Tree Component Proposal

Proposed Component Name	Zend_Tree
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Tree
Proposers	Andries Seutens
Revision	0.1 - 28 January 2007: Added first draft for proposal (wiki revision: 13)

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

About tree's
A tree is a data structure that emulates a tree structure with a set of linked nodes. Each node has zero or more child nodes, which are below it in the tree. A node that has a child is called the child's parent node. A node has at most one parent.

The root node
The topmost node in a tree is called the root node. Being the topmost node, the root node will not have parents. It is the node at which all operations on the tree begin. All other nodes can be reached from it by following edges or links. Every node in a tree can be seen as the root node of the subtree rooted at that node.

A subtree

A subtree is a portion of a tree data structure that can be viewed as a complete tree in itself. Any node in a tree T, together with all the nodes below it, comprise a subtree of T. The subtree corresponding to the root node is the entire tree; the subtree corresponding to any other node is called a proper subtree.

Leaf nodes
Nodes at the bottom most level of the tree are called leaf nodes. Since they are at the bottom most level, they will not have any children. An internal node or inner node is any node of a tree data structure that has child nodes and is thus not a leaf node.

Zend_Tree's role
Zend_Tree will be a data structure which would probably have three most interesting adapters. Db, PHP array, and XML:

- DB because it will help with catalogues and other Db related data.
- PHP array's for sessions and other PHP centric functionality.
- XML for data exchange.

2. References

Other known components

- PEAR's Tree
- Horde Framework's Tree

Popular tree data structures:

- AA tree
- AVL tree
- Red-black tree
- Splay tree
- Scapegoat tree

Other tree data structures:

- B-tree (2-3 tree, B+ tree, B*-tree)
- DSW algorithm
- Dancing tree
- R-tree
- Radix tree
- Skip list

3. Component Requirements, Constraints, and Acceptance Criteria

This component will:

- Allow subclassing;
- Support various import/export formats;
- Support the enumerating all the items;
- Support searching for an item;
- Support adding a new item at a certain position on the tree;
- Support deleting an item;
- Support removing a whole section of a tree;
- Support adding a whole section to a tree;
- Support finding the root for any node;
- Support manipulating hierarchical data;
- Make information easy to search;
- Support exporting to a variety of different formats;
- Know how to serialize itself (could be useful in storing it in Zend_Session);

This component will not:

- Create tight connection to Zend_Acl, Zend_Session, or other modules;

4. Dependencies on Other Framework Components

- Zend_Exception

5. Theory of Operation

todo

6. Milestones / Tasks

- Milestone 1: write proposal / use cases / class skeletons and try to gather feedback from community
- Milestone 2: Class development
- Milestone 3: Unit tests and debugging
- Milestone 4: Documentation

7. Class Index

- Zend_Tree
- Zend_Tree_Exception
- Zend_Tree_Adapter_Abstract
- Zend_Tree_Adapter_Db
- Zend_Tree_Adapter_Xml
- Zend_Tree_Adapter_Array

8. Use Cases

Working with XML (comparible to simple xml)

```
// zend.xml
<root>
  <users>
    <user>
      <fname>Darby</fname>
      <name>Felton</name>
   </user>
    <user>
      <fname>Gavin</fname>
      <name>Vess</name>
   </user>
  </users
</root>
```

9. Class Skeletons

todo

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