

Zend_Config Improvements - Chris Gutierrez

<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_Config Improvements Component Proposal

Proposed Component Name	Zend_Config Improvements
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Config Improvements
Proposers	My E-mail Address
Revision	1.1 - 1 August 2006: Updated from community comments. (wiki revision: 20)

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

This is essentially a group of improvements that I would like to add to the Zend_Config component to increase usability.

2. References

3. Component Requirements, Constraints, and Acceptance Criteria

- The proposed improvements will leave the Zend_Config component completely backward compatible and pass all current unit tests.
- The improvements will make it easier to retrieve configuration data on the fly.
- the configuration file will be parsed once and configuration sections will be retrieved on a per need basis
- a method will be added called Zend_Config_*::getSection() which will facilitate the retrieval of needed sections
- any key/value pairs not within a section will be appended to the requested section upon component instantiation .

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Config
- Zend_Config_Exception

5. Theory of Operation

as a bit of back story, here is how these "improvements" were inspired. There have been several instances where, in the bootstrap file of my application, I would request a section from my config file, primarily using Zend_Config_Ini. There have been several times throughout the rest of my application where I have wanted to retrieve other more specific sections of my configuration. With the existing component, there are two ways I could go about this:

- I could have just requested all of the sections I needed within the bootstrap. I didn't really like doing that though because why store all that processed configuration data for areas that you may not need it in.
- just instantiate a new config object with the currently needed section, I didn't want to do this because of the regular opening and reparsing of the config file.

With these improvements, the configuration file gets parsed once during instantiation and is stored in its parsed state to allow for retrieval of any section at any time.

6. Milestones / Tasks

- Milestone 1: Complete proposal
- Milestone 2: Build working prototype and submit to incubator supporting use cases
- Milestone 3: flush out to work with existing unit tests and provide unit tests for new methods
- Milestone 4: flush out documentation

7. Class Index

- Zend_Exception
- Zend_Config_Exception
- Zend_Config_Ini
- Zend_Config_Xml
- Zend_Config

8. Use Cases

UC-01

Here is an example of how these updates can be used.

here is a sample ini configuration

```
misckey = a value
misckey2 = another value

[production]
somevariable = somevalue
db.host = host
db.user = user
db.password = password

[development]
somevariable = somevalue
db.host = host
db.user = user
db.password = password

[modulespecific]
foo = bar
```

now lets say in your boot strap you need to get an "environment" configuration. you would do something like this

```
...
$config = new Zend_Config_Ini('configfile.ini', 'production', $config);
Zend_Registry::set('config', $config);
/*
  when doing a var_dump or print_r on the config object, the following data will be
  available

  somevariable = somevalue
  db.host = host
  db.user = user
  db.password = password
  misckey = a value
  misckey2 = another value

  any non section oriented keys get appended to the requested data upon instantiation.

  */
...
```

you would now have a configuration object containing the data of the 'production' section, and in this case, I am adding it to the registry.

now lets suppose that later, in another controller or module, you have more specific data that you need. you would now be able to get that data like so

```

..
// get the current config object from the registry
$config = Zend_Registry::get('config');

// get the desired section
$modulespecific = $config->getSection('modulespecific');

echo $modulespecific->foo;
// prints out 'bar'
// or...
$array = $modulespecific->toArray();
echo $array['foo'];

..

```

the complete ini data is stored along with the config object. it is kept in its parsed state.
the getSection method also has 'native' options regarding how to return the configuration data

```

// return the not extended version of the config data
$modulespecific = $config->getSection('modulespecific', false);

// return the data as an array
$modulespecific = $config->getSection('modulespecific', true, false);

```

9. Class Skeletons

```

class Zend_Config_Ini extends Zend_Config {

    public function getSection($section, $processextends = true, $returnobject = true)
    {

    }

}

class Zend_Config_Xml extends Zend_Config {

    public function getSection($section, $processextends = true, $returnobject = true)
    {

    }

}

```

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