Zend_Serializer - Marc Bennewitz

<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_Serializer Component Proposal

Proposed Component Name	Zend_Serializer
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Serializer
Proposers	Marc Bennewitz
Zend Liaison	TBD
Revision	1.0 - 20. Oct 2009: Initial Draft. 1.1 - 07. Dec 2009: Milestone 3 & 4 finished (wiki revision: 12)

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

Zend_Serializer is a simple component to un-/serialize values by different adapters.

2. References

- serialize/unserialize
- var_export/eval
- wddx
- Zend_Json
- igbinary
- pickle

3. Component Requirements, Constraints, and Acceptance Criteria

- The component factory **will** use a configurable plugin loader to instantiate adapters.
- The component **will** throw an exception if un-/serialization failed (instead of returning false).

- The component adapters **doesn't** implement the Serializable interface of spl.

4. Dependencies on Other Framework Components

- Zend_Exception
- Zend_Json (only json adapter)
- Zend_Amf (only amf[0|3] adapter)
- Zend_Loader_PluginLoader (to load adapter class)

5. Theory of Operation

The Component instantiate a adapter on factory which implements un-/serialize methods, these methods will be used generates a string representation of a value and vice versa.
The static implementation within Zend_Serializer::[un]serialize generates an instance of the selected adapter in given options or an configurable default adapter will be used.

6. Milestones / Tasks

- Milestone 1: [DONE]: Finish proposal
- Milestone 2: [DONE]: Working prototype
- Milestone 3: [DONE]: Prototype checked into the incubator
- Milestone 4: [DONE]: Unit tests exist finished and component is working
- Milestone 5: Initial documentation exists
- Milestone 6: Changed related components
- Milestone 7: Moved to core.

7. Class Index

- Zend_Serializer
- Zend_Serializer_Exception
- Zend_Serializer_Adapter_AdapterInterface
- Zend_Serializer_Adapter_AbstractAdapter
- Zend_Serializer_Adapter_PhpSerialize (using [un]serialize)
- Zend_Serializer_Adapter_PhpCode (using var_export + eval)
- Zend_Serializer_Adapter_Wddx
- Zend_Serializer_Adapter_Json
- Zend_Serializer_Adapter_Igbinary
- Zend_Serializer_Adapter_PythonPickle
- Zend_Serializer_Adapter_Amf0
- Zend_Serializer_Adapter_Amf3

8. Use Cases

UC-01

```
$serializer = Zend_Serializer::factory('PhpSerialize');
try {
    $serializedData   = $serializer->serialize($data, $options);
    $unserializedData = $serializer->unserialize($serializedData, $options);
} catch (Zend_Serializer_Exception $e) {
    echo $e;
}
```

UC-02

```
Zend_Serializer::setDefaultAdapter('Wddx');
try {
    $serializedData   = Zend_Serializer::serialize($data, $options);
    $unserializedData = Zend_Serializer::unserialize($serializedData, $options);
} catch (Zend_Serializer_Exception $e) {
    echo $e;
}
```

9. Class Skeletons

```
class Zend_Serializer
{
    /**
     * Plugin loader to load adapter class.
     *
     * @var null|Zend_Loader_PluginLoader
     */
    private static $_adapterLoader =  null;

    /**
     * The default adapter.
     *
     * @var string|Zend_Serializer_AdapterInterface
     */
    protected static $_defaultAdapter = 'PhpSerialize';

    public static function factory($adapterName);

    public static function getAdapterLoader();

    public static function setAdapterLoader(Zend_Loader_PluginLoader $pluginLoader);

    public static function setDefaultAdapter($adapter);

    public static function getDefaultAdapter();

    public static function serialize($value, $options);
```

```
    public static function unserialize($serialized, $options);
}

class Zend_Serializer_Exception extends Zend_Exception
{}

interface Zend_Serializer_Adapter_AdapterInterface
{

    /**
     * Generates a storable representation of a value.
     *
     * @param mixed $value Data to serialize
     * @param array $options Serialize options
     * @return string
     * @throws Zend_Serializer_Exception
     */
    public function serialize($value, array $options = array());

    /**
     * Creates a PHP value from a stored representation.
     *
     * @param string $serialized Serialized string
     * @param array $options Unserialize options
     * @return mixed
     * @throws Zend_Serializer_Exception
     */
    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_PhpSerialize implements
Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_PhpCode implements
Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_Wddx implements Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());
```

```
}

class Zend_Serializer_Adapter_Json implements Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_Igbinary implements
Zend_Serializer_Adapter_AdapterInterface
{

    public function __constructor() {
        if (!extension_loaded('igbinary')) {
            require_once 'Zend/Serializer/Exception.php';
            throw new Zend_Serializer_Exception('PHP extension "igbinary" is required
for this adapter');
        }
    }

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_PythonPickle implements
Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_Amf0 implements Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());

}

class Zend_Serializer_Adapter_Amf3 implements Zend_Serializer_Adapter_AdapterInterface
{

    public function serialize($value, array $options = array());

    public function unserialize($serialized, array $options = array());
```

```
}
```

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