Zend_Soap_ComplexTypeValuesValidator - Jeannie Boffel

<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_Soap_ComplexTypeValuesValidator Component Proposal

Proposed Component Name	Zend_Soap_ComplexTypeValuesValidator
Developer Notes	http://framework.zend.com/wiki/display/ZFDEV/Zend_Soap_ComplexTypeValuesValidator
Proposers	Jeannie BOFFEL
Zend Liaison	TBD
Revision	1.0 - 07 December 2010: Initial Draft. (wiki revision: 8)

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

I'd like to introduce to Zend Framework a new layer to check/validates values returned in a SOAP response against what user declared in the schema through auto discovery system.

2. References

- Zend Framework AutoDiscovery
- ZF-10789
- Zend_Soap_SimpleTypeValuesValidator

3. Component Requirements, Constraints, and Acceptance Criteria

- This component **will** add support for checking validity of values against XSD declaration from auto discovery support.

- This component **will not** make sure values are meaningful.
- This component **will not** validate the XSD generated. Meanning we don't prevent user from doing wrong XSD by adding unexisting attribute to an element for example.
- This component **will not** be exhaustive and **will not** support all XSD features immediately.

4. Dependencies on Other Framework Components

- PHP >= 5.3.0
- Zend_Soap
- Zend_Soap_AutoDiscovery
- Zend_Soap_Zend_Soap_Wsdl_Strategy_DefaultComplexType (modified by proposal)
- Zend_Soap_SimpleValuesValidator (in other proposal)

5. Theory of Operation

The component is instantiated through extending of class.

6. Milestones / Tasks

- Feature Request 1: Support more XSD base type than just PHP variable base set type.

- Milestone #: [DONE] Design
- Milestone #: [DONE] Proof of concept by implement new complex type and simple type value checking
- Milestone #: [DONE] Start support of all base type compatible with PHP variable base set type
- Milestone #: [DONE] Start support of user defined type through SimpleType in XSD (other proposal)
- Milestone #: [DONE] Start support of user defined type through ComplexType in XSD
- Milestone 1: Working prototype checked into the incubator supporting use cases #1.
- Milestone 2: Unit tests exist, work, and are checked into SVN.
- Milestone 3: Initial documentation exists.

7. Class Index

- Zend_Soap_ComplexTypeValuesValidator

8. Use Cases

Use case in future.

UC-01

Exemple of definition of complex type which will check and assign values.

```
/**
 * @xsd sequence
 * @xsd complexType
 */
class brotherDetail extends Zend_Soap_ComplexTypeValuesValidator
{
    /**
     * @var string
     */
    public $firstName;
    /**
     * @var RankType
     */
    public $rank;
    /**
     * @var AgeType
     */
    public $age;

    /**
     * @param   string  $firstName
     * @param   RankType    $rank
     * @param   AgeType $age
     * @return  brotherDetail
     */
    public function __construct($firstName, $rank, $age) {
        $this->Check_firstName = $firstName;
        $this->Check_rank = $rank;
        $this->Check_age = $age;

        return $this;
    }

}
If everything is ok, no output, if one value is wrong, exception.
```

UC-02

Exemple of definition of complex type which will only check assigned values then manually assign values.
__set magic method has syntax limitation, it will not be called even if property doesn't exist when used with auto index array for example.
We do not bother memory for nothing here, the $this->CheckO_Anything will never really exist in class properties, it's just a trick to call __set of parent class.

```
/**
 * @xsd sequence
 * @xsd complexType
 */
class detailsBrotherList extends Zend_Soap_ComplexTypeValuesValidator
{
    /**
     * @var brotherDetail   array('minOccurs'=>0, 'maxOccurs'=>"unbounded")
     */
    public $brotherDetail;

    /**
     * @return  detailsBrotherList
     */
    public function __construct() {
        $this->brotherDetail = array();
        return $this;
    }

    /**
     * @param   brotherDetail   $brotherDetail
     */
    public function addStb($brotherDetail) {
        $this->brotherDetail[] = $this->CheckO_brotherDetail = $brotherDetail;
    }

}
```

If everything is ok, no output, if one value is wrong, exception.

UC-03

Here the same example but with handling of exception for debug help at programming moment and/or to send custome SoapFault to customer instead of invalid values or even try to fix the wrong value.

```
/**
 * @xsd sequence
 * @xsd complexType
 */
class detailsBrotherList extends Zend_Soap_ComplexTypeValuesValidator
{
    /**
     * @var brotherDetail   array('minOccurs'=>0, 'maxOccurs'=>"unbounded")
     */
    public $brotherDetail;

    /**
     * @return  detailsBrotherList
     */
    public function __construct() {
        $this->brotherDetail = array();
        return $this;
    }

    /**
     * @param   brotherDetail   $brotherDetail
     */
    public function addStb($brotherDetail) {
        try {
            $this->brotherDetail[] = $this->CheckO_brotherDetail = $brotherDetail;
        } catch (Exception $e) {
            throw new SoapFault('Server', $e->getMessage());
        }
    }

}
```

If everything is ok, no output, if one value is wrong, exception.

9. Class Skeletons

```
class Zend_Soap_ComplexTypeValuesValidator {

 /**
  * Reflection result on each checked class
  *
  * @var object
  */
 private $_classReflection = null;

 /**
  * Magic function to overload properties of children class.
  *
  * @param string $name
  * @param mixed $value
  */
 public function __set($name, $value) {

  $isCheckOnly = false;

  if (strstr($name, "CheckO_") !== false)
   $isCheckOnly = true;

  $name = substr(strchr($name, '_'), 1);
  $className = get_called_class();

  if ($this->_classReflection === null)
   $this->_classReflection = new ReflectionClass($className);

  if ($this->_classReflection->getProperty($name)->isPublic()
                    && preg_match_all('/@var\s+([^\s]+)/m',
$this->_classReflection->getProperty($name)->getDocComment(), $matches))
   $type = $matches[1][0];
  else
   throw new Exception('Undefined type for variable: ' . $name . ', in class ' . $className, 500);

  if (!$this->check($type, $value))
   throw new Exception('Value: ' . print_r($value, true) . ' does not match XSD declaration', 500);

  if ($isCheckOnly === false)
   $this->$name = $value;
 }

 /**
  * Function to check if value match type.
  *
  * @param string $type
  * @param mixed $value
  */
 public static function check($type, $value) {
  if (is_callable($type . '::getInstance')) {
   return $type::getInstance()->test($value);
  } else if (is_callable($type . '::check')) {
   return $value instanceof $type;
  } else {
   return SimpleTypeValuesValidator::checkBaseType($type, $value);
  }
 }
}
```

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