ArrayTool

Deprecated. Please use ListTool instead.

JUnit tests can be found here.

```
/*
 * Copyright 2003-2004 The Apache Software Foundation.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package org.apache.velocity.tools.generic;

import java.lang.reflect.Array;
import java.util.ArrayList;
import java.util.List;

/**
 * Tool for working with arrays in Velocity templates.
 * It provides a method to get and set specified elements,
 * retrieve the length, and create clones of an array object.
 * Also provides a method to convert arrays into java.util.List's.
 *
 * <p><pre>
 * Example uses:
 *  $primes                   -> new int[] {2, 3, 5, 7}
 *  $array.length($primes)    -> 4
 *  $array.get($primes, 2)    -> 5
 *  $array.clone($primes)     -> int[] {2, 3, 5, 7}, != $primes
 *  $array.set($primes, 2, 1) -> (primes[2] becomes 1)
 *  $array.get($primes, 2)    -> 1
 *  $array.get($clone, 2)     -> 5
 *
 * Example toolbox.xml config (if you want to use this with VelocityView):
 * &lt;tool&gt;
 *   &lt;key&gt;array&lt;/key&gt;
 *   &lt;scope&gt;application&lt;/scope&gt;
 *   &lt;class&gt;org.apache.velocity.tools.generic.ArrayTool&lt;/class&gt;
 * &lt;/tool&gt;
 * </pre></p>
 *
 * <p>This tool is entirely threadsafe, and has no instance members.
 * It may be used in any scope (request, session, or application).
 * </p>
 *
 * @author <a href="mailto:shinobu@ieee.org">Shinobu Kawai</a>
 * @version $Id: $
 */
public class ArrayTool
{

    /**
     * Default constructor.
     */
    public ArrayTool()
    {
    }

    /**
```

```
     * Converts an array object into a java.util.List.
     * <ul>
     *   <li>
     *     If the object is an array of an Object,
     *     it will return a java.util.List of the elements.
     *   </li>
     *   <li>
     *     If the object is an array of a primitive type,
     *     it will return a java.util.List of the elements wrapped in their wrapper class.
     *   </li>
     *   <li>
     *     If the object is none of the above, it will return null.
     *   </li>
     * </ul>
     * @param array an array object.
     * @return the converted java.util.List.
     */
    public List list(Object array)
    {
        if (!this.isArray(array))
        {
            return null;
        }

        // Thanks to Eric Fixler for this refactor.
        int length = Array.getLength(array);
        List asList = new ArrayList(length);
        for (int index = 0; index < length; ++index)
        {
            asList.add(Array.get(array, index));
        }
        return asList;
    }

    /**
     * Gets the specified element of an array.
     * It will return null under the following conditions:
     * <ul>
     *   <li><code>array</code> is null.</li>
     *   <li><code>array</code> is not an array.</li>
     *   <li><code>array</code> doesn't have an <code>index</code>th value.</li>
     * </ul>
     * @param array the array object.
     * @param index the index of the array to get.
     * @return the specified element of the array.
     */
    public Object get(Object array, int index)
    {
        if (!this.isArray(array))
        {
            return null;
        }

        try
        {
            return Array.get(array, index);
        }
        catch (IndexOutOfBoundsException e)
        {
            // The index was wrong.
            return null;
        }
    }

    /**
     * Sets the specified element of an array.
     * It will return null under the following conditions:
     * <ul>
     *   <li><code>array</code> is null.</li>
     *   <li><code>array</code> is not an array.</li>
     *   <li><code>array</code> doesn't have an <code>index</code>th value.</li>
```

```
     * </ul>
     * @param array the array object.
     * @param index the index of the array to set.
     * @param value the element to set.
     */
    public Object set(Object array, int index, Object value)
    {
        if (!this.isArray(array))
        {
            return null;
        }

        try
        {
            Array.set(array, index, value);
            return "";
        }
        catch (IndexOutOfBoundsException e)
        {
            // The index was wrong.
            return null;
        }
    }

    /**
     * Gets the length of an array.
     * It will return null under the following conditions:
     * <ul>
     *   <li><code>array</code> is null.</li>
     *   <li><code>array</code> is not an array.</li>
     * </ul>
     * @param array the array object.
     * @return the length of the array.
     */
    public Integer length(Object array)
    {
        if (!this.isArray(array))
        {
            return null;
        }

        // Thanks to Eric Fixler for this refactor.
        return new Integer(Array.getLength(array));
    }

    /**
     * Gets the clone of an array.
     * It will return null under the following conditions:
     * <ul>
     *   <li><code>array</code> is null.</li>
     *   <li><code>array</code> is not an array.</li>
     * </ul>
     * @param array the array object.
     * @return the clone of the array.
     */
    public Object clone(Object array)
    {
        if (!this.isArray(array))
        {
            return null;
        }

        Class type = array.getClass().getComponentType();
        int length = Array.getLength(array);
        Object clone = Array.newInstance(type, length);
        System.arraycopy(array, 0, clone, 0, length);
        return clone;
    }

    /**
     * Checks if an object is an array.
```

```
     * @param object the object to check.
     * @return <code>true</code> if the object is an array.
     */
    public boolean isArray(Object object)
    {
        if (object == null)
        {
            return false;
        }
        return object.getClass().isArray();
    }

}
```