Grouper WS Lite - REST input - output XHTML - XML - JSON

This document describes the strategy for parsing and generating XHTML, XML, JSON for Grouper WS Lite / REST

Summary

There is a custom automatic converter for the parsing and generation of XHTML. If we want to eventually support XML / JSON / whatever, we can plug that in. The parsing is very flexible, and will give warnings if things are not as expected (e.g. unexpected element / attribute). It is also possible to hose the parser (e.g. send the wrong type for a field). We can also add strict mode if we like so that invalid input (though preparing for WS upgrades) will be ok (e.g. unexpected element name or missing a required attribute).

For XML the same beans are translated with default XStream.

For JSON the same beans are translated with default XStream and Jettison.

Use

The server needs to know the request and response content type. If sending data in the request, if it is not HTTP params, then set the Content-type http header. Must be one of the content types specified in the javadoc for the enum WsLiteRequestContentType. Currently the valid values are: application /xhtml+xml, text/xml, text/x-json, and for http params (either dont set content type or set to application/x-www-form-urlencoded).

The response content type will be the same as the request content type. If the request content type is http params (null or form encoded), then the response will be whatever is specified in the grouper-ws.properties, and it defaults to xhtml. If the client wants to override this decision (to either get a different content type than was requested, or to specify a different than the one specified in the grouper-ws.properties), then it can be placed in the url.

e.g. here is a url where the content type is the same as request or what is configured in grouper-ws.properties: http://localhost/grouper-ws/servicesLite/v3_0_000/group/a:b

e.g. here is a url where the request is an http param request, and the configured response content type is xhtml, but the client wants xml: http://localhost/grouper-ws/servicesLite/*xml*/v3_0_000/group/a:b?includeGroupDetail=T

Guidelines

When working with the XHTML, please prepare for upgrades on the server. e.g. when parsing assume there might be different types of elements, new attributes, etc. If there is something not expected, then probably ignore it (unless something is badly malformed or something). Assume that things will be added to the service, and not changed or removed (unless there is a drastic upgrade). Also, the XML/XHTML/JSON is not indented, though this shouldnt matter. The XHTML should be valid based on the strict doctype (can validate from validator.w3.org). The XML and JSON should obviously be valid

Mapping of Java Object to XHTML

The objects used for the Grouper SOAP / XML-HTTP web service will be used for this web service also (though they dont have to be), it supports beans based on javabean properties (read / getters, write / setters). It supports only:

1. String fields
2. int fields
3. String arrays
4. int arrays
5. Bean fields
6. Bean arrays
7. Will not work with circular references

Will throw exception if something is not right... Does not support any other structures. Inheritance is not supported. Use the objects WsXhtmlOutputConverter, and WsXhtmlInputConverter once and throw away.

Each Object will be written to a div or li (if in list). Each scalar (currently only Strings or ints) will be written to p tags or li tags for lists. Each array will be a ul with li's for each item. Each div (except the root), p, and ul will have a "class" attribute of the fieldName. Each div and li (which is a bean and not scalar) will have a "title" attribute which is the simple name of the java class (non-fully qualified).

Null and the empty string are interchangeable and are represented as an empty element <p />

Example

For these three classes:

```
/*
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;


/**
 *
 */
public class BeanGrandparent {
```

```
/** field 1 */
private String field1;

/** field 2*/
private String field2;

/** string array */
private String[] stringArray;

/** field */
private BeanParent beanParent;

/** field */
private BeanParent[] beanParents;

/**
 * empty
 */
public BeanGrandparent() {
  //empty
}

/**
 * @param _field1
 * @param _field2
 * @param _stringArray
 * @param _beanParent
 * @param _beanParents
 */
public BeanGrandparent(String _field1, String _field2, String[] _stringArray,
    BeanParent _beanParent, BeanParent[] _beanParents) {
  super();
  this.field1 = _field1;
  this.field2 = _field2;
  this.stringArray = _stringArray;
  this.beanParent = _beanParent;
  this.beanParents = _beanParents;
}


/**
 * @return the field1
 */
public String getField1() {
  return this.field1;
}


/**
 * @param _field1 the field1 to set
 */
public void setField1(String _field1) {
  this.field1 = _field1;
}


/**
 * @return the field2
 */
public String getField2() {
  return this.field2;
}


/**
 * @param _field2 the field2 to set
 */
public void setField2(String _field2) {
  this.field2 = _field2;
}
```

```
  /**
   * @return the stringArray
   */
  public String[] getStringArray() {
    return this.stringArray;
  }


  /**
   * @param _stringArray the stringArray to set
   */
  public void setStringArray(String[] _stringArray) {
    this.stringArray = _stringArray;
  }


  /**
   * @return the beanParent
   */
  public BeanParent getBeanParent() {
    return this.beanParent;
  }


  /**
   * @param _beanParent the beanParent to set
   */
  public void setBeanParent(BeanParent _beanParent) {
    this.beanParent = _beanParent;
  }


  /**
   * @return the beanParents
   */
  public BeanParent[] getBeanParents() {
    return this.beanParents;
  }


  /**
   * @param _beanParents the beanParents to set
   */
  public void setBeanParents(BeanParent[] _beanParents) {
    this.beanParents = _beanParents;
  }
}
```

```
/*
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;


/**
 * parent bean
 */
public class BeanParent {
  /** field */
  private String parentField1;

  /** field */
  private String parentField2;

  /** field */
  private String[] parentStringArray;
```

```
/** field */
private int intField;

/** ean child */
private BeanChild beanChild;

/** child */
private BeanChild beanChild2;

/** child */
private BeanChild[] beanArray;

/** child */
private BeanChild[] beanArray2;

/**
 * @return the parentField1
 */
public String getParentField1() {
  return this.parentField1;
}


/**
 * @param _parentField1 the parentField1 to set
 */
public void setParentField1(String _parentField1) {
  this.parentField1 = _parentField1;
}


/**
 * @return the parentField2
 */
public String getParentField2() {
  return this.parentField2;
}


/**
 * @param _parentField2 the parentField2 to set
 */
public void setParentField2(String _parentField2) {
  this.parentField2 = _parentField2;
}


/**
 * @return the parentStringArray
 */
public String[] getParentStringArray() {
  return this.parentStringArray;
}


/**
 * @param _parentStringArray the parentStringArray to set
 */
public void setParentStringArray(String[] _parentStringArray) {
  this.parentStringArray = _parentStringArray;
}

/**
 * empty
 */
public BeanParent() {
  //empty
}


/**
```

```
 * @param _parentField1
 * @param _parentField2
 * @param _parentStringArray
 * @param _intField
 * @param _beanChild
 * @param _beanChild2
 * @param _beanArray
 * @param _beanArray2
 */
public BeanParent(String _parentField1, String _parentField2, String[] _parentStringArray,
    int _intField, BeanChild _beanChild, BeanChild _beanChild2, BeanChild[] _beanArray,
    BeanChild[] _beanArray2) {
  super();
  this.parentField1 = _parentField1;
  this.parentField2 = _parentField2;
  this.parentStringArray = _parentStringArray;
  this.intField = _intField;
  this.beanChild = _beanChild;
  this.beanChild2 = _beanChild2;
  this.beanArray = _beanArray;
  this.beanArray2 = _beanArray2;
}


/**
 * @return the intField
 */
public int getIntField() {
  return this.intField;
}



/**
 * @param _intField the intField to set
 */
public void setIntField(int _intField) {
  this.intField = _intField;
}



/**
 * @return the beanChild
 */
public BeanChild getBeanChild() {
  return this.beanChild;
}



/**
 * @param _beanChild the beanChild to set
 */
public void setBeanChild(BeanChild _beanChild) {
  this.beanChild = _beanChild;
}



/**
 * @return the beanChild2
 */
public BeanChild getBeanChild2() {
  return this.beanChild2;
}



/**
 * @param _beanChild2 the beanChild2 to set
```

```
   */
  public void setBeanChild2(BeanChild _beanChild2) {
    this.beanChild2 = _beanChild2;
  }



  /**
   * @return the beanArray
   */
  public BeanChild[] getBeanArray() {
    return this.beanArray;
  }



  /**
   * @param _beanArray the beanArray to set
   */
  public void setBeanArray(BeanChild[] _beanArray) {
    this.beanArray = _beanArray;
  }



  /**
   * @return the beanArray2
   */
  public BeanChild[] getBeanArray2() {
    return this.beanArray2;
  }



  /**
   * @param _beanArray2 the beanArray2 to set
   */
  public void setBeanArray2(BeanChild[] _beanArray2) {
    this.beanArray2 = _beanArray2;
  }



}
```

```
/*
 * @author mchyzer
 * $Id$
 */
package edu.internet2.middleware.grouper.ws.xml;


/**
 * child bean
 */
public class BeanChild {
  /** field */
  private String childField1;

  /** field */
  private String childField2;

  /** field */
  private String[] childStringArray;

  /** int array */
  private int[] childIntegerArray;

  /**
```

```
 * empty
 */
public BeanChild() {
  //empty
}

/**
 * @param _childField1
 * @param _childField2
 * @param _childStringArray
 * @param _childIntegerArray
 */
public BeanChild(String _childField1, String _childField2, String[] _childStringArray,
    int[] _childIntegerArray) {
  super();
  this.childField1 = _childField1;
  this.childField2 = _childField2;
  this.childStringArray = _childStringArray;
  this.childIntegerArray = _childIntegerArray;
}


/**
 * field
 * @return the childField1
 */
public String getChildField1() {
  return this.childField1;
}


/**
 * field
 * @param _childField1 the childField1 to set
 */
public void setChildField1(String _childField1) {
  this.childField1 = _childField1;
}


/**
 * field
 * @return the childField2
 */
public String getChildField2() {
  return this.childField2;
}


/**
 * field
 * @param _childField2 the childField2 to set
 */
public void setChildField2(String _childField2) {
  this.childField2 = _childField2;
}


/**
 * field
 * @return the childStringArray
 */
public String[] getChildStringArray() {
  return this.childStringArray;
}


/**
 * field
 * @param _childStringArray the childStringArray to set
 */
```

```
  public void setChildStringArray(String[] _childStringArray) {
    this.childStringArray = _childStringArray;
  }



  /**
   * field
   * @return the childIntegerArray
   */
  public int[] getChildIntegerArray() {
    return this.childIntegerArray;
  }



  /**
   * field
   * @param _childIntegerArray the childIntegerArray to set
   */
  public void setChildIntegerArray(int[] _childIntegerArray) {
    this.childIntegerArray = _childIntegerArray;
  }
}
```

and for this code:

```
BeanChild beanChild = new BeanChild("va<l1", "val2", new String[]{"a"}, new int[]{1, 2});
    BeanParent beanParent = new BeanParent("qwe", "rtyu", new String[]{ "uio", "cv"}, 45,
        null, beanChild, null, new BeanChild[]{beanChild});

    BeanGrandparent beanGrandparent = new BeanGrandparent("xv", "",
        new String[]{null},beanParent, new BeanParent[]{beanParent, beanParent} );
```

It will generate this XHTML:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>the title</title>
  </head>
  <body>
    <div title="BeanGrandparent">
      <p class="field1">xv</p>
      <p class="field2" />
      <ul class="stringArray">
        <li />
      </ul>
      <div class="beanParent" title="BeanParent">
        <p class="parentField1">qwe</p>
        <p class="parentField2">rtyu</p>
        <ul class="parentStringArray">
          <li>uio</li>
          <li>cv</li>
        </ul>
        <p class="intField">45</p>
        <div class="beanChild" title="BeanChild" />
        <div class="beanChild2" title="BeanChild">
          <p class="childField1">va&lt;l1</p>
          <p class="childField2">val2</p>
          <ul class="childStringArray">
            <li>a</li>
          </ul>
          <ul class="childIntegerArray">
            <li>1</li>
            <li>2</li>
          </ul>
```

```
  </div>
  <ul class="beanArray2">
    <li title="BeanChild">
      <p class="childField1">va&lt;l1</p>
      <p class="childField2">val2</p>
      <ul class="childStringArray">
        <li>a</li>
      </ul>
      <ul class="childIntegerArray">
        <li>1</li>
        <li>2</li>
      </ul>
    </li>
  </ul>
</div>
<ul class="beanParents">
  <li title="BeanParent">
    <p class="parentField1">qwe</p>
    <p class="parentField2">rtyu</p>
    <ul class="parentStringArray">
      <li>uio</li>
      <li>cv</li>
    </ul>
    <p class="intField">45</p>
    <div class="beanChild" title="BeanChild" />
    <div class="beanChild2" title="BeanChild">
      <p class="childField1">va&lt;l1</p>
      <p class="childField2">val2</p>
      <ul class="childStringArray">
        <li>a</li>
      </ul>
      <ul class="childIntegerArray">
        <li>1</li>
        <li>2</li>
      </ul>
    </div>
    <ul class="beanArray2">
      <li title="BeanChild">
        <p class="childField1">va&lt;l1</p>
        <p class="childField2">val2</p>
        <ul class="childStringArray">
          <li>a</li>
        </ul>
        <ul class="childIntegerArray">
          <li>1</li>
          <li>2</li>
        </ul>
      </li>
    </ul>
  </li>
  <li title="BeanParent">
    <p class="parentField1">qwe</p>
    <p class="parentField2">rtyu</p>
    <ul class="parentStringArray">
      <li>uio</li>
      <li>cv</li>
    </ul>
    <p class="intField">45</p>
    <div class="beanChild" title="BeanChild" />
    <div class="beanChild2" title="BeanChild">
      <p class="childField1">va&lt;l1</p>
      <p class="childField2">val2</p>
      <ul class="childStringArray">
        <li>a</li>
      </ul>
      <ul class="childIntegerArray">
        <li>1</li>
        <li>2</li>
      </ul>
    </div>
    <ul class="beanArray2">
```

```
            <li title="BeanChild">
              <p class="childField1">va&lt;l1</p>
              <p class="childField2">val2</p>
              <ul class="childStringArray">
                <li>a</li>
              </ul>
              <ul class="childIntegerArray">
                <li>1</li>
                <li>2</li>
              </ul>
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </body>
</html>
```

It generates this XML:

```
<BeanGrandparent>
  <field1>xv</field1>
  <stringArray>
    <null />
  </stringArray>
  <beanParent>
    <parentField1>qwe</parentField1>
    <parentField2>rtyu</parentField2>
    <parentStringArray>
      <string>uio</string>
      <string>cv</string>
    </parentStringArray>
    <intField>45</intField>
    <beanChild2>
      <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
      <childField2>val2</childField2>
      <childStringArray>
        <string>a</string>
      </childStringArray>
      <childIntegerArray>
        <int>1</int>
        <int>2</int>
      </childIntegerArray>
    </beanChild2>
    <beanArray2>
      <BeanChild>
        <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
        <childField2>val2</childField2>
        <childStringArray>
          <string>a</string>
        </childStringArray>
        <childIntegerArray>
          <int>1</int>
          <int>2</int>
        </childIntegerArray>
      </BeanChild>
    </beanArray2>
  </beanParent>
  <beanParents>
    <BeanParent>
      <parentField1>qwe</parentField1>
      <parentField2>rtyu</parentField2>
      <parentStringArray>
        <string>uio</string>
        <string>cv</string>
      </parentStringArray>
      <intField>45</intField>
      <beanChild2>
        <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
        <childField2>val2</childField2>
```

```
        <childStringArray>
          <string>a</string>
        </childStringArray>
        <childIntegerArray>
          <int>1</int>
          <int>2</int>
        </childIntegerArray>
      </beanChild2>
      <beanArray2>
        <BeanChild>
          <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
          <childField2>val2</childField2>
          <childStringArray>
            <string>a</string>
          </childStringArray>
          <childIntegerArray>
            <int>1</int>
            <int>2</int>
          </childIntegerArray>
        </BeanChild>
      </beanArray2>
    </BeanParent>
    <BeanParent>
      <parentField1>qwe</parentField1>
      <parentField2>rtyu</parentField2>
      <parentStringArray>
        <string>uio</string>
        <string>cv</string>
      </parentStringArray>
      <intField>45</intField>
      <beanChild2>
        <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
        <childField2>val2</childField2>
        <childStringArray>
          <string>a</string>
        </childStringArray>
        <childIntegerArray>
          <int>1</int>
          <int>2</int>
        </childIntegerArray>
      </beanChild2>
      <beanArray2>
        <BeanChild>
          <childField1>v&amp;quot;a&amp;lt;l{1}</childField1>
          <childField2>val2</childField2>
          <childStringArray>
            <string>a</string>
          </childStringArray>
          <childIntegerArray>
            <int>1</int>
            <int>2</int>
          </childIntegerArray>
        </BeanChild>
      </beanArray2>
    </BeanParent>
  </beanParents>
</BeanGrandparent>
```

And it generates this JSON:

```
{
   "BeanGrandparent":{
      "field1":"xv",
      "stringArray":{
         "null":""
      },
      "beanParent":{
         "parentField1":"qwe",
         "parentField2":"rtyu",
         "parentStringArray":{
```

```
      "string":[
         "uio",
         "cv"
      ]
   },
   "intField":45,
   "beanChild2":{
      "childField1":"v\"a<l{1}",
      "childField2":"val2",
      "childStringArray":{
         "string":"a"
      },
      "childIntegerArray":{
         "int":[
            1,
            2
         ]
      }
   },
   "beanArray2":{
      "BeanChild":{
         "childField1":"v\"a<l{1}",
         "childField2":"val2",
         "childStringArray":{
            "string":"a"
         },
         "childIntegerArray":{
            "int":[
               1,
               2
            ]
         }
      }
   }
},
"beanParents":{
   "BeanParent":[
      {
         "parentField1":"qwe",
         "parentField2":"rtyu",
         "parentStringArray":{
            "string":[
               "uio",
               "cv"
            ]
         },
         "intField":45,
         "beanChild2":{
            "childField1":"v\"a<l{1}",
            "childField2":"val2",
            "childStringArray":{
               "string":"a"
            },
            "childIntegerArray":{
               "int":[
                  1,
                  2
               ]
            }
         },
         "beanArray2":{
            "BeanChild":{
               "childField1":"v\"a<l{1}",
               "childField2":"val2",
               "childStringArray":{
                  "string":"a"
               },
               "childIntegerArray":{
                  "int":[
                     1,
                     2
```

```
                        ]
                     }
                  }
               }
            },
            {
               "parentField1":"qwe",
               "parentField2":"rtyu",
               "parentStringArray":{
                  "string":[
                     "uio",
                     "cv"
                  ]
               },
               "intField":45,
               "beanChild2":{
                  "childField1":"v\"a<l{1}",
                  "childField2":"val2",
                  "childStringArray":{
                     "string":"a"
                  },
                  "childIntegerArray":{
                     "int":[
                        1,
                        2
                     ]
                  }
               },
               "beanArray2":{
                  "BeanChild":{
                     "childField1":"v\"a<l{1}",
                     "childField2":"val2",
                     "childStringArray":{
                        "string":"a"
                     },
                     "childIntegerArray":{
                        "int":[
                           1,
                           2
                        ]
                     }
                  }
               }
            }
         ]
      }
   }
}
```

Warnings example

Here is INVALID MARKUP, (extra attribute, and extra element) but it is successfully parsed and here is the warning that the web service will produce in the response back to the client (though the request will still succeed if everything else is ok)

```
<div title=" BeanChild " id=" something">
  <span>something</span>
  <p class="childField1">va&lt;l1</p>
  <p class="childField2">val2</p>
  <ul class="childStringArray">
    <li>a</li>
  </ul>
  <ul class="childIntegerArray">
    <li>1</li>
    <li>2</li>
  </ul>
</div>
```

Warnings:

```
Element 'div' is not expecting attribute: 'id'
Element 'div' is not expecting child element: 'span'
```