Dynamic forms, struts and the java.lang.IndexOutOfBoundsException and ArrayIndexOutOfBoundsException

Creating a input form with multiple lines where the number of lines depends on your business logic.

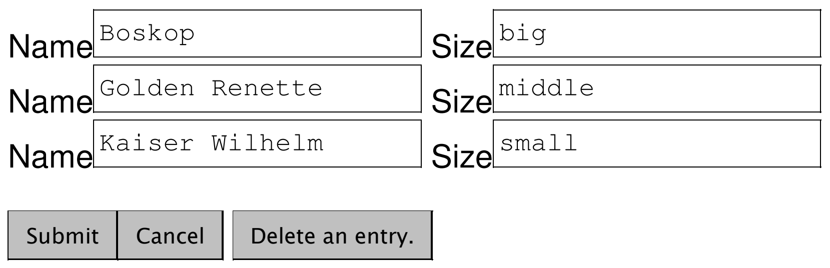
General

Author: Sebastian Hennebrüder
www.laliluna.de – Tutorials for Struts, EJB, xdoclet and eclipse.
Date: September, 13th 2004
Struts version 1.1 and 1.2
Source code for this tutorial can be downloaded here
http://www.laliluna.de/tutorial/struts-dynamic-forms/DynamicForms.zip
I did not put the struts and commons libs in the source code. Please add them to the directory. /WEB-INF/LIB/

I do imply knowledge of struts!
In case, you find too much nonsense in this tutorial, please contact me. Contact information can be found at http://www.laliluna.de.

Introduction

A dynamic form is a struts form where the number of entries is decided by the business logic. Here an example for multiple input lines:



Name	Boskop	Size	big
Name	Golden Renette	Size	middle
Name	Kaiser Wilhelm	Size	small

A typical example is a shopping form where you can add multiple items to a cart. Very likely you are running into the following problem:

```
java.lang.IndexOutOfBoundsException: Index: 2, Size: 2
        at java.util.ArrayList.RangeCheck(ArrayList.java:507)
        at java.util.ArrayList.get(ArrayList.java:324)
        at org.apache.commons.beanutils.PropertyUtils.getIndexedProperty(PropertyUtils.java:521)
        at org.apache.commons.beanutils.PropertyUtils.getIndexedProperty(PropertyUtils.java:428)
        at org.apache.commons.beanutils.PropertyUtils.getNestedProperty(PropertyUtils.java:770)
        at org.apache.commons.beanutils.PropertyUtils.getProperty(PropertyUtils.java:801)
        at org.apache.commons.beanutils.BeanUtils.setProperty(BeanUtils.java:881)
        at org.apache.commons.beanutils.BeanUtils.populate(BeanUtils.java:808)
```

You could also run into a ArrayIndexOutBoundsException
I will try to explain the reason for this problem.

Explanation

When you use a dynamic number of entries you will likely use an Array or ArrayList in a form bean like

```
private ArrayList apples;
```

Your struts jsp file will include something like the following.

```
<logic:iterate name="applesForm" property="apples" id="bingBong"
indexId="index">
Name<html:text name="bingBong" property="name" indexed="true"/>
Size<html:text name="bingBong" property="size" indexed="true"/>
</logic:iterate>
```

Using an action where the form bean is saved in the request, will lead to the following sequence when you press the submit button.

1. The form bean is reset.
2. BeanUtils are called to fill the form bean with the values.
3. PropertyUtils are called for each passed value from the form.
4. PropertyUtils need to find out what type the value is and calls for example: apples.get(0)
5. As the form bean has been reset. The arrayList is at least empty. So you will get the IndexOutOfBoundsException.

Workaround 1 (very bad)

Save the form bean in the session.
Excerpt from struts-config.xml:
No scope entry means session scope.

```
<action
   attribute="bugApplesForm"
   input="/form/bugApples.jsp"
   name="bugApplesForm"
   path="/bugApples"
   type="de.laliluna.dynamicForms.struts.action.BugAppleAction">
   <forward name="success" path="/form/bugApples.jsp" />
 </action>
```

This will lead to a working example because the form bean is no longer reset between the requests. When your ArrayList had three entries your three entries will be overridden by the new values.

But imagine the following:
When you show a list with two entries instead of three, click browser back and submit a form where three entries are listed.
Then you will get what you don't like.

```
java.lang.IndexOutOfBoundsException: Index: 2, Size: 2
	at java.util.ArrayList.RangeCheck(ArrayList.java:507)
	at java.util.ArrayList.get(ArrayList.java:324)
	at org.apache.commons.beanutils.PropertyUtils.getIndexedProperty(PropertyUtils.java:521)
	at org.apache.commons.beanutils.PropertyUtils.getIndexedProperty(PropertyUtils.java:428)
	at org.apache.commons.beanutils.PropertyUtils.getNestedProperty(PropertyUtils.java:770)
	at org.apache.commons.beanutils.PropertyUtils.getProperty(PropertyUtils.java:801)
	at org.apache.commons.beanutils.BeanUtils.setProperty(BeanUtils.java:881)
	at org.apache.commons.beanutils.BeanUtils.populate(BeanUtils.java:808)
```

Your application will throw very ugly messages.

Test the bugApple example. Delete an entry, click browser-back and submit the three apples.

Workaround 2 (better)

In your form bean add a getBingBong function. You can use any name for bingBong. This function will add a new entry to the ArrayList when the ArrayList is not big enough.

```
private ArrayList apples;
public Apple getBingBong(int index){
        while(apples.size() <= index){
          apples.add(new Apple());
        }
        return (Apple) apples.get(index);
   }
```

What is very important is:
In your jsp you have to use bingBong as the id for the logic:iterate. So your source code will look like:

```
<html:form action="/editApples">
<logic:iterate name="applesForm" property="apples" id="bingBong" indexId="index">
Name<html:text name="bingBong" property="name" indexed="true"/>
Size<html:text name="bingBong" property="size" indexed="true"/>
<br>
</logic:iterate>
<br>
<html:submit/><html:cancel/>
</html:form>
```

Try the editAppleAction from the sources.

Workaround 3 (best) – state of the art

Create a Decorator around your list that overrides the get function. The decorator knows a factory to create a new instance and creates it when the list is to short.

This was not my idea but from the gurus of struts.

Try the editNestedApple action from the sources.

You did not understand what this means. Here an easier explanation.

Easy explanation of the background.

Each List has a get(int index) functions delivering the Object at this position.

We will create a List which will grow larger automatically when this function is called with an index greater than the actual list.

In addition to that, the List will fill the called position with an initialized object.

Instead of reinventing the List we will put a class just in front of it. This class will replace the get function. A class providing a new feature for an existing is a design pattern called „decorator".

Our decorator has to know a class which create a new instance of our object (Apple). Such a class is called factory.

Know each time the get(int index) function is called, it will create a new object with the factory.

```
public Object get(int index) {
 while (index >= list.size())
  list.add(objectFactory.create());
  return list.get(index);
}
```

An easy example can be found in the package

package de.laliluna.dynamicForms.decoratorExample;

in the tutorial sources.

Continue to see the professional way.

Get the Commons Collections in the version 3.1

http://jakarta.apache.org/site/binindex.cgi

When you are using an older struts version and don't want to replace your commons collections you will need the following source files

org.apache.commons.collections.list.LazyList

org.apache.commons.collections.list.AbstractListDecorator

org.apache.commons.collections.list.AbstractSerializableListDecorator

org.apache.commons.collections.collection.AbstractCollectionDecorator

org.apache.commons.collections.Factory

Download them from http://jakarta.apache.org/site/sourceindex.cgi or take them from the example sources.

Create the class to be included in your list

For example an apple. You can use the full example from the tutorial source code.

```
public class Apple {
   private String name;
   public Apple(){
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
}
```

Create the action form.

We will create a form bean including a List of apples.

```
public class NestedApplesForm extends ActionForm {

    /** our List of apples*/
    private List apples;
```

Provide a getter for the list

```
    public List getApples() {
        return apples;
    }
```

and now the new important thing. Initialize your apples with the decorated list:

```
public void reset(ActionMapping arg0, HttpServletRequest arg1) {
 Factory factory = new Factory() {
  public Object create() {
   return new Apple();
  }
 };
 apples = LazyList.decorate(new ArrayList(), factory);
}
```

Create the action class.

Make the struts-config entries.

Have a look at the NestedApple action in the sources.

Finish

That's it.

Possible Solution – needed changes in struts

I am not very experienced in Struts. So the following text has to be understood as simple ideas which might be wiped away by a struts guru.

Struts has to find out what type of data your ArrayList contains to properly fill the form bean.
So the first idea is thast Struts or better BeanUtils could force the existance of a getPropertyTyp function in the bean, which returns the type of an ArrayList. Then it creates a new instance of this type and adds it to the arrayList.

Second idea is to wait for the next java version with „typed“ arrays.