Tapestry5DisplayableSelectionModel

Displayable Selection Model

This code is free to use and modify with no limitations.

This is a twist on the Tapestry5GenericSelectionModel that used the Bean Utils library. Instead of using Bean Utils, this implementation only requires that the objects you want to display implement a lightweight interface that provides a String output method.
The main goals are the same as with Tapestry5GenericSelectionModel :

- To get you up an running in Tapestry with less hassle
- To allow you to use more robust objects in your selection model
- To let the object decide how it is displayed

Another new feature I worked in helps solve two problems:

1. Many times you want to inform the user they need to select something, without presenting error messages to them. This is especially true if they've never visited the page before. You don't want your users to think they've already selected something, unless you are setting it for them. 2. If you only have one item in the list, you can't select it if you use the "onchange" event to submit the enclosing form. The value will be displayed already, so "onchange" won't fire! This is pertinent if you don't plan to have a submit button for your select box.

For problem 1, HLS suggested using Groups. Groups are nice in that they allow like options to be grouped together under a select component. They also provide indention and look slick.
However, the label for the group is not selectable, so it doesn't help with the second issue. (This implementation still lets you supply GroupModels if you want!)

For problem 2, you could preselect the single item for them, but then you'd have to go about triggering the form events somehow (there may be multiple events!), and also making sure validation gets triggered.

The implementations below allow you to have an instruction on the select component, which has a null value. Your list of objects doesn't actually contain this null item though, so it doesn't get in the way. If you don't supply the label for the informative step, you don't get it in your select. It's that simple.

Let's take a look.

IDisplayable.java

Here is the simple interface.

```
public interface IDisplayable {
        public String toDisplay();
}
```

DisplayableSelectionModel.java

Here is the selection model. Note the two different constructors, one with and one without the instruction String.

```
import IDisplayable;

import org.apache.tapestry.OptionGroupModel;
import org.apache.tapestry.OptionModel;
import org.apache.tapestry.internal.OptionGroupModelImpl;
import org.apache.tapestry.internal.OptionModelImpl;
import org.apache.tapestry.util.AbstractSelectModel;

/**
 * @author jued
 *
 * @param <T>
 */
public class DisplayableSelectionModel<T extends IDisplayable> extends
                AbstractSelectModel {

        private List<T> list;

        private String groupLabel = null;
```

```
	private String instructions = null;

	public DisplayableSelectionModel(List<T> list) {
		this.list = list;
	}

	public DisplayableSelectionModel(List<T> list, String groupLabelIn) {
		this.list = list;
		this.groupLabel = groupLabelIn;
	}

	public DisplayableSelectionModel(List<T> list, String groupLabelIn,
			String instructionsIn) {
		this.list = list;
		this.groupLabel = groupLabelIn;
		this.instructions = instructionsIn;
	}

	private List<OptionModel> generateOptionModelList() {
		List<OptionModel> optionModelList = new ArrayList<OptionModel>();
		if (list != null) {
			for (T obj : list) {
				optionModelList.add(new OptionModelImpl(obj.toDisplay(), false,
						obj, new String[0]));
			}
		}
		return optionModelList;
	}

	private List<OptionModel> generateEmptyOptionModelList() {
		List<OptionModel> l = new ArrayList<OptionModel>();
		l.add(new OptionModelImpl(instructions, false, null, new String[0]));
		return l;
	}

	public List<OptionGroupModel> getOptionGroups() {
		if (groupLabel == null) {
			return null;
		} else {
			List<OptionGroupModel> optionModelList = new ArrayList<OptionGroupModel>();
			if (this.instructions != null) {
				// append a group for the sole purpose of having the instruction
				// item appear before other groups, and not inside one
				// particular group.
				optionModelList.add(new OptionGroupModelImpl("Instructions",
						false, generateEmptyOptionModelList()));
			}
			optionModelList.add(new OptionGroupModelImpl(this.groupLabel,
					false, generateOptionModelList()));

			return optionModelList;
		}
	}

	public List<OptionModel> getOptions() {
		if (groupLabel == null) {
			List<OptionModel> l = generateOptionModelList();
			if (this.instructions != null) {
				// append the instructions as a null item.
				l.add(0, new OptionModelImpl(instructions, false, null,
						new String[0]));
			}
			return l;
		} else {
			return null;
		}
	}
}
```

DisplayableValueEncoder.java

```
import java.util.List;

import IDisplayable;

import org.apache.tapestry.ValueEncoder;

public class DisplayableValueEncoder<T extends IDisplayable> implements
                ValueEncoder<T> {
        private List<T> list;

        public DisplayableValueEncoder(List<T> list) {
                this.list = list;
        }

        public String toClient(T obj) {
                if (obj == null) {
                        return null;
                } else {
                        return obj.toDisplay();
                }
        }

        public T toValue(String string) {
                if (list != null) {
                        for (T obj : list) {
                                if (obj.toDisplay().equals(string)) {
                                        return obj;
                                }
                        }
                }
                return null;
        }
}
```

Usage

Here is an example of the usage, using Month objects. A Month object might have many details inside it, but all we care about is that it's an object we want to display – so it implements IDisplayable.

```
import java.util.List;

import myapp.model.Model;
import myapp.model.domain.Month;

import org.apache.tapestry.annotations.Component;
import org.apache.tapestry.annotations.Persist;
import org.apache.tapestry.corelib.components.Select;

import DisplayableSelectionModel;
import DisplayableValueEncoder;

public class SelectMonth{
        @Persist
        private List<Month> list;

        @Persist
        private Month selected;

        @Component(id = "monthSelect", parameters = { "value = selected",
                        "model = model", "encoder = encoder",
                        "onchange = 'this.form.submit();'" })
        private Select monthSelect;

        public DisplayableValueEncoder<Month> getEncoder() {
                return new DisplayableValueEncoder<Month>(getList());
        }

        /**
         * @return the list
         */
        public List<Month> getList() {
                if (list == null) {
                        list = Model.getMonths();
                }
                return list;
        }

        public DisplayableSelectionModel<Month> getModel() {
                return new DisplayableSelectionModel<Month>(getList(), null,
                                "Please Select");
        }

        /**
         * @return the selected month
         */
        public Month getSelected() {
                return selected;
        }

        /**
         * @param list
         *            the list to set
         */
        public void setList(List<Month> list) {
                this.list = list;
        }

        /**
         * @param month
         *            the month to set
         */
        public void setSelected(Month month) {
                selected = month;
        }
}
```