Tapestry5HowToAddBindingPrefix

In this example we will create a list binding prefix to support array/list values directly in templates without needing to write getter methods.

This is tested on Tapestry 5.0.14, it will most likely be a part of the framework in some way, but currently it isn't.

Example: You have a product page, product/view, and you want to be able to compare the current product to a list of related products. So you write this in your template:

```
<t:loop t:source="relatedproducts" t:value="relatedproduct">
  <t:pagelink page="product/compare" context="productcomparecontext">Compare with ${relatedproduct.name}</t:
pagelink>
</t:loop>
```

And this in your Java class:

```
public Object[] getProductCompareContext(){
    return new Object[]{ product.getId(), relatedProduct.getId() };
}
```

The list binding prefix simplifies this to:

```
<t:loop t:source="relatedproducts" t:value="relatedproduct">
  <t:pagelink page="product/compare" context="list:product.id,relatedproduct.id">Compare with ${relatedproduct.
name}</t:pagelink>
</t:loop>
```

Let's implement the list binding prefix.

Add this to your AppModule.java:

```
    public static void contributeBindingSource(
            MappedConfiguration<String, BindingFactory> configuration,
            BindingSource bindingSource)
    {
        configuration.add("list", new ListBindingFactory(bindingSource));
    }
```

The ListBindingFactory:

```
/**
 * Factory for list bindings. List bindings parse comma-delimited lists of binding expressions into {@link List
lists}
 * of values. The default binding prefix for each binding expression is prop.
 */
public class ListBindingFactory implements BindingFactory {

  private final BindingSource bindingSource;

  public ListBindingFactory(BindingSource source) {
    this.bindingSource = source;
  }

  public Binding newBinding(String description, ComponentResources container, ComponentResources component,
                            String expression, Location location) {
    List<Binding> delegates = new ArrayList<Binding>();
    String[] items = expression.split(",");
    boolean invariant = true;

    for (String item : items) {
      Binding binding = bindingSource.newBinding(description, container, component,
                                                 BindingConstants.PROP, item, location);
      invariant = invariant && binding.isInvariant();
      delegates.add(binding);
    }

    return new ListBinding(delegates, invariant);
  }
}
```

The ListBinding :

```
public class ListBinding extends AbstractBinding {

  private final List<Binding> delegates;
  private final boolean invariant;

  public ListBinding(List<Binding> delegates, boolean invariant) {
    this.delegates = delegates;
    this.invariant = invariant;
  }

  public Object get() {
    List<Object> values = new ArrayList<Object>(delegates.size());

    for (Binding binding : delegates) {
      values.add(binding.get());
    }

    return values;
  }

  public boolean isInvariant() {
    return invariant;
  }

  public Class<List> getBindingType() {
    return List.class;
  }
}
```

Notice: You must override isInvariant and return false to avoid the value being cached. In this example we return true if all binding expressions are invariant.

And that's it. You now have a list binding prefix. It splits the expression on "," and each subexpression will be evaluated as it was a single binding expression.

For example a chart link - similar to the chart example (Tapestry5HowToCreateASimpleGraphComponent) on this wiki - can be used like this with the list binding prefix:

```
<t:chart width="200" height="150" context="list:'aa',22,'bb',5" popup="popupSize"/>
<t:chart width="200" height="150" context="list:'aa',29,'bb',30,'cc',10" popup="popupSize"/>
```