Tapestry5HowToAddMapBindingPrefix

This page describes how to add a map binding prefix to Tapestry 5.

The map binding prefix allows you to specify a map like this:

```
${map:key1=value1,key2=value2} or <t:component t:parameter="map:key1=value1,key2=value2"/>
```

This will create a Map<String, Object> with key1 and key2 as keys and the values of the bindings value1 and value2 as values.

First create the MapBinding:

```
public class MapBinding extends AbstractBinding {

  private final Map<String, Binding> delegates;
  private final boolean invariant;

  public MapBinding(Map<String, Binding> delegates, boolean invariant) {
    this.delegates = delegates;
    this.invariant = invariant;
  }

  public Object get() {
    Map<String, Object> values = new HashMap<String, Object>(delegates.size());

    for (Map.Entry<String, Binding> entry : delegates.entrySet()) {
      values.put(entry.getKey(), entry.getValue().get());
    }

    return values;
  }

  public boolean isInvariant() {
    return invariant;
  }

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

Then create the MapBindingFactory:

```
/**
 * Factory for map bindings. Map bindings parse comma-delimited lists of key=value pairs into {@link Map maps}.
Keys are
 * always string literals while the values can be any binding expression, for which the default prefix is prop.
 */
public class MapBindingFactory implements BindingFactory {

  private final BindingSource bindingSource;

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

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

    for (String entry : entries) {
      String[] parts = entry.split("=");

      if (parts.length != 2) throw new RuntimeException(String.format("Entry '%s' is malformed", entry));

      String name = parts[0];
      String value = parts[1];

      Binding binding = bindingSource.newBinding(description, container, component,
                                                 BindingConstants.PROP, value, location);
      invariant = invariant && binding.isInvariant();
      delegates.put(name, binding);
    }

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

And then contribute it to the BindingSource configuration:

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

Again, the keys are always string literals and the values themselves can be any binding expression, for which the default binding prefix is prop.

That's all there is to it.