Tapestry5HowToAddMessageFormatBindingPrefix

This page describes how to add a format prefix to Tapestry 5.

The format prefix allows you to specify values to use in a message inline like this:

```
${format:key=value1,value2} or <t:component t:parameter="format:key=value1,value2"/>
```

Where the component's message catalog contains:

```
key = This is the first value: %s, it's a string. This is the second value: %d, it's a number.
```

First create the FormatBinding:

```
public class FormatBinding extends AbstractBinding {

  private final Messages messages;
  private final boolean invariant;
  private final List<Binding> keyBindings;
  private final List<Binding> valueBindings;

  public FormatBinding(Location location, Messages messages, boolean invariant,
                       ArrayList<Binding> keyBindings, List<Binding> valueBindings) {
    super(location);

    this.messages = messages;
    this.invariant = invariant;
    this.keyBindings = keyBindings;
    this.valueBindings = valueBindings;
  }

  public FormatBinding(Location location, Messages messages, boolean invariant, ArrayList<Binding> keyBindings)
{
    super(location);

    this.messages = messages;
    this.invariant = invariant;
    this.keyBindings = keyBindings;
    this.valueBindings = null;
  }

  public Object get() {
    String key = "";
    for (Binding keyBinding : keyBindings) {
      key += keyBinding.get();
    }

    if (null == valueBindings) return messages.get(key);

    List<Object> values = new ArrayList<Object>(valueBindings.size());
    for (Binding valueBinding : valueBindings) {
      values.add(valueBinding.get());
    }

    return messages.format(key, values.toArray());
  }

  public boolean isInvariant() {
    return this.invariant;
  }

  @SuppressWarnings("unchecked")
  public Class getBindingType() {
    return String.class;
  }
}
```

Then create the class FormatBindingFactory:

```
public class FormatBindingFactory
    implements BindingFactory {

  private static final String SEPARATOR = "=";
  private static final String DELIMITER = ",";

  private static final String KEY_PREFIX = TapestryConstants.LITERAL_BINDING_PREFIX;
  private static final String VALUE_PREFIX = TapestryConstants.PROP_BINDING_PREFIX;

  private final BindingSource bindingSource;

  public FormatBindingFactory(BindingSource bindingSource) {
    this.bindingSource = bindingSource;
  }

  public Binding newBinding(String description, ComponentResources container, ComponentResources component,
                            String expression, Location location) {
    int separatorIndex = expression.indexOf(SEPARATOR);

    if (-1 == separatorIndex) {
      List<String> keys = Arrays.asList(expression.split(DELIMITER));

      ArrayList<Binding> keyBindings = createBindings(description, container, component, KEY_PREFIX, keys,
location);

      boolean invariant = isInvariant(keyBindings);
      return new FormatBinding(location, container.getMessages(), invariant, keyBindings);
    }

    List<String> keys = Arrays.asList(expression.substring(0, separatorIndex).split(DELIMITER));
    ArrayList<Binding> keyBindings = createBindings(description, container, component, KEY_PREFIX, keys,
location);

    List<String> values = Arrays.asList(expression.substring(separatorIndex + 1).split(DELIMITER));
    ArrayList<Binding> valueBindings = createBindings(description, container, component, VALUE_PREFIX, values,
                                                      location);

    boolean invariant = isInvariant(keyBindings) && isInvariant(valueBindings);
    return new FormatBinding(location, container.getMessages(), invariant, keyBindings, valueBindings);
  }

  private ArrayList<Binding> createBindings(String description, ComponentResources container,
                                            ComponentResources component, String defaultPrefix,
                                            List<String> expressions, Location location) {
    ArrayList<Binding> bindings = new ArrayList<Binding>(expressions.size());

    for (String expression : expressions) {
      bindings.add(bindingSource.newBinding(description, container, component, defaultPrefix, expression,
location));
    }

    return bindings;
  }

  private boolean isInvariant(ArrayList<Binding> bindings) {
    for (Binding binding : bindings) {
      if (!binding.isInvariant()) return false;
    }

    return true;
  }
}
```

And then contribute it to the BindingSource configuration:

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

Notes about the key part of the binding:

1. The key can contain multiple parts separated by commas, and 2. each part of the key can use an optional binding prefix (defaults to literal).

This means you can do stuff like ${format:gender-,prop:user.gender} which will look up the messages gender-male and gender-female depending on the value of user.gender.

Notes about the value part of the binding:

1. Literal values should be enclosed in single quotes, and 2. it is possible to use other prefixes for the values.

That's all there is to it.