CocoonFormsScratchpad

Plans for discussion, revision, and implementation

Please note that even though these are just proposals, the descriptions are written as though the features already existed, to make it easier to convert the descriptions into official documentation when and if consensus is reached and a suitable implementation is completed.

Macros and Macro Repositories – define/expand/inherit and import

A macro is a definition (or blueprint) consisting of a list of one or more definitions for widgets, bindings, or templates. Expanding a macro does not create a macro container, but instead creates instances of the definitions it contains and inlines them into the surrounding environment.

Note that while groups are used for clarity in the examples below, they are not required. A macro may be expanded anywhere, such as within the root form object or inside a repeater row.

For example, typing this macro definition, group definition, and macro expansion request:

```
<fd:macro define="mymacro">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:macro>

<fd:group id="mygroup">
  <fd:macro expand="mymacro"/>
</fd:group>
```

is equivalent to typing this:

```
<fd:group id="mygroup">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:group>
```

...except that you now also have defined a reusable macro named "mymacro", which you could expand in several places in your form instead of retyping the list of definitions in each place where those definitions are required.

You can also define a general macro, and create variants based upon it. Think of this as creating a blueprint which contains most of your details, and then making photocopies of this blueprint to fill in different finishing touches to customize each copy. In terms of forms, this would consist of adding or deleting the widget, binding, or template definitions in the copies of the main macro.

For example:

```
<fd:macro define="mymacro">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:macro>

<fd:macro define="yourmacro" inherit="mymacro">
  <fd:add>
    <fd:field id="anotherfield">
      ...
    </fd:field>
  </fd:add>
</fd:macro>

<fd:group id="mygroup">
  <fd:macro expand="mymacro"/>
</fd:group>

<fd:group id="yourgroup">
  <fd:macro expand="yourmacro"/>
</fd:group>
```

is equivalent to:

```
<fd:group id="mygroup">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:group>

<fd:group id="yourgroup">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
  <fd:field id="anotherfield">
    ...
  </fd:field>
</fd:group>
```

If your customizations are only used in one place, then you may specify the changes within the expansion request.

For example:

```
<fd:macro define="mymacro">
  <fd:field id="myfield">
    ...
  </fd:field>
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:macro>

<fd:group id="yourgroup">
  <fd:macro expand="yourmacro">
    <fd:del>
      <fd:field id="myfield"/>
    </fd:del>
  </fd:macro>
</fd:group>
```

is equivalent to:

```
<fd:group id="yourgroup">
  <fd:field id="yourfield">
    ...
  </fd:field>
</fd:group>
```

You can also modify definitions while expanding or inheriting from a macro.

For example:

```
<fd:macro define="mymacro">
  <fd:field id="myfield">
    ...
  </fd:field>
</fd:macro>

<fd:macro expand="yourmacro">
  <fd:rename from="myfield" to="yourfield"/>
</fd:macro>
```

is equivalent to:

```
<fd:field id="yourfield">
  ...
</fd:field>
```

This is useful for tasks such as assigning non-conflicting id's, adding validation rules, and setting datatypes.

The specific syntax for these various types of modifications is still to be determined.

Collections of macros may be stored anywhere accessible via a uri (including such uri's as cocoon://somepath), to be imported as shared macro repositories. Macros are brought into scope via an import statement and referenced via a local prefix.

An important point to note is that macros and macro repositories are cached and reused, so using them saves both memory and processing time.

Given this file at "cocoon:/mysharedmacros.xml":

```
<fd:macro-set>
  <fd:macro define="yourmacro">
    <fd:booleanfield id="mybool">
      ...
    </fd:booleanfield>
  </fd:macro>
  <fd:macro define="someothermacro">
    ...
  </fd:macro>
  ...
</fd:macro-set>
```

this sample:

```
<fd:import prefix="mymacros" uri="cocoon:/mysharedmacros.xml"/>
<fd:macro expand="mymacros:yourmacro"/>
```

is equivalent to:

```
<fd:booleanfield id="mybool">
  ...
</fd:booleanfield>
```

A source containing a macro-set is also permitted to itself import macros from other sources via this same mechanism. However, these nested imports are not automatically made available to files which import the macro-set.

You can individually re-export the macros manually:

```
<fd:macro-set>
  <fd:import prefix="yourmacros" uri="someothermacros.xml"/>
  <fd:macro define="somemacro" inherit="yourmacros:somemacro"/>
</fd:macro-set>
```