WhiteBoardCocoonForms

Description

This is an area for experimental Cocoon Forms features. It is intended to allow for discussion of actual code instead of just abstract ideas, without incurring the overhead of supporting the features before their designs have been finalized.

This is an experimental development area, so changes may be made without notice, and no assurances are made that everything will be in a working state at any given moment.

That being said, the area is for collaboration and so should usually be in a mostly working state to allow for evaluation and incremental improvement.

Current experimental features

The documentation which follows describes what is currently implemented by the code. If you modify the code, please change this documentation as well.

All aspects of these features (syntax, implementation, etc.) are subject to discussion and change.

Macro library and field extension support

*Define - Creates a reusable macro definition.
*Expand - Creates an instance of a macro definition.
*Library - Defines a reusable macro library.
*Import - Brings a macro library into scope.

The syntax and examples below are written assuming use of macros in a form definition. Identical macro support is also present for the form binding and the form template, using the respective name spaces.

Define a macro

Creates a reusable macro definition.

```
Syntax:

  <fd:macro define="...">
    <!-- Widget definitions -->
  </fd:macro>

Example:

  <fd:macro define="product">
    <fd:field id="name">
      <fd:label>Name</fd:label>
      <fd:datatype base="string"/>
    </fd:field>
    <fd:field id="price">
      <fd:label>Price</fd:label>
      <fd:datatype base="decimal"/>
    </fd:field>
  </fd:macro>
```

Expand

Creates an instance of a macro definition.

```
Syntax:

  <fd:macro expand="..."/>

Example:

  <fd:macro expand="product"/>

Effective result:

  <fd:field id="name">
    <fd:label>Name</fd:label>
    <fd:datatype base="string"/>
  </fd:field>
  <fd:field id="price">
    <fd:label>Price</fd:label>
    <fd:datatype base="decimal"/>
  </fd:field>
```

Macros

Defines a reusable library.

```
Syntax:

  <fd:library>
    <!-- Macro definitions -->
    <!-- Reusable field definitions -->
  </fd:library>

Note:

  The "fd:library" element must be the root of the document and contains macros and fields.

Example:

  <fd:library>
    <fd:macro define="product">
      <fd:field id="name">
        <fd:label>Name</fd:label>
        <fd:datatype base="string"/>
      </fd:field>
      <fd:field id="price">
        <fd:label>Price</fd:label>
        <fd:datatype base="decimal"/>
      </fd:field>
    </fd:macro>

    <fd:field id="email">
      <fd:label>email</fd:label>
      <fd:datatype base="string"/>
      <fd:validation>
         <fd:email/>
      </fd:validation>
    </fd:field>

    <fd:field id="customer-number">
      <fd:label>customer number</fd:label>
      <fd:datatype base="string"/>
      <fd:validation>
        <fd:length exact="16"/>
      </fd:validation>
    </fd:field>


  </fd:library>
```

Import (using the library's elements)

Bring library elements into scope.

```
Syntax:

  <fd:import prefix="..." uri="..."/>

Example:

  <fd:import prefix="my-library" uri="cocoon:/custom-fd-library.xml"/>
  <fd:macro expand="my-library:product"/>
  <fd:field id="customer-email" extends="my-library:email">
    <fd:label>customer's email</fd:label>
    <fd:validation>
      <fd:length exact="30"/>
    </fd:validation>
  </fd:field>
  <fd:field id="cust" extends="my-library:customer-number"/>

Effective result:

  <fd:field id="name">
    <fd:label>Name</fd:label>
    <fd:datatype base="string"/>
  </fd:field>
  <fd:field id="price">
    <fd:label>Price</fd:label>
    <fd:datatype base="decimal"/>
  </fd:field>

  <fd:field id="customer-email">
    <fd:label>customer's email</fd:label>
    <fd:validation>
      <fd:length exact="30"/>
      <fd:email/>
    </fd:validation>
  </fd:field>

  <fd:field id="cust">
    <fd:label>customer number</fd:label>
    <fd:datatype base="string"/>
    <fd:validation>
       <fd:length exact="16"/>
    </fd:validation>
  </fd:field>
```

The "prefix" is a user-defined string used when expanding or extending macros to specify from which import to get the base macros.

The "uri" identifies the source for the macro library. Any protocol understood by Cocoon may be used.

Note that single fields don't need to be defined within an `fd:macro` element. This (global) definitions can be reused in the custom form definition using the `extends` attribut.

Additional Notes

Imports are permitted inside macro libraries. This allows use of the imported macros within the library, but does not automatically expose the imported macros to users of the library.

Import forms a flat namespace in the form model, but a tree shaped namespace in the binding. This is on purpose, so that both designs can be evaluated. Which would be the preferred design to settle on, and why?

What should we call the root element of a macro library (currently "macros", but perhaps there is a better name)?

There is preliminary support for parameterized macros in the form template code.