Contents

1 FORMS: User Applications

FORMS application provides set of CSS stylesheets for compact forms definitions and also it provides database model for storing metadata information about documents, fields and validations.

1.1 Overview

The basic idea that stands behind form models is that N2O forms are able to be generated from its metamodel which is also a root for other generated persisted Erlang records for KVS storage. N2O book is the best for the taxonomy of N2O forms and KVS interface. This kind of metainterpretation and unification of containers is usual for enterprise and common object oriented systems.

1.2 Metainformation

Metainformation declares the documents (#document) and its fields (#field) which forms a document level entity that can be stored in database. Usually somewhere in ACT or in DBS applications you can find its document definition in Erlang records which is entered with forms.

1.3 Document

The #document object is an application form definition. It consists of sections (#sec) that include fields with its descriptions and validations.

```
-record(document,   { ?ITERATOR(feed),
                      name,
                      base,
                      sections,
                      fields,
                      buttons,
                      access }).
```

Each section #sec of forms are entitled with heading font.

```
-record(sec,        { id, name, desc="" }).
```

Forms age given with its control buttons (#but). The information from field postback in button is directly translated to N2O element postback during forms:new/2.

```
-record(but,        { id, postback, name, title,
                      sources=[], class }).
```

1.4 Validation

Since document consists of validations and fields, here is their record definitions in FORMS model:

```
-record(validation, { name, type, msg,
                      extract = fun(X) -> X end, options=[],
                      function, field={record,pos} }).

-record(opt,        { id, name, title, postback, checked=false }).
-record(sel,        { id, name, title, postback }).
-record(field,      { id, sec=1, name, pos, title, layout,
                      visible=true, disabled=false,
                      format="~w", curr="", options=[],
                      postfun=[], desc, wide=normal,
                      type=binary, etc, access, tooltips=[] }).
```

1.5 Data Model

KVS Data Model is being generated from Metainformation. KVS layer along with FEEDS server provide persistence facilities.

1.6 Application

JavaScript Web Application is generated using Metainformation and Data Model. N2O is used as DSL language for forms generation. JavaScript/OTP is also used for forms generation. Forms average render speed is 50 FPS (forms per second).

1.7 Documents

```
document(Name,Phone) ->
               #document { name = Name,

   sections =[ #sec      { name= [<<"Input OTP sent to ">>,
                           wf:to_list(Phone#phone.number)] } ],

   buttons  =[ #but      { name='decline', title = <<"Cancel">>,
                           class=cancel,
                           postback={'CloseOpenedForm', Name} },

               #but      { name='next', title = <<"Continue">>,
                           class=[button,green],
                           postback={'Spinner', {'OpenForm',Name}},
                           sources=[otp] } ],

   fields   =[ #field    { name='otp', type=otp,
                           title= <<"Password">>,
                           labelClass=label,
                           fieldClass=column3} ] }.
```

1.8 Fields

1.9 Validation Rules

Validation rules should be applied by developer. Erlang and JavaScript/OTP is used to define validation rules applied to documents during workflow.

1.10 Form Autogeneration

1.11 XForms and XMPP Data Forms

The other well known standard is XForms that could be easily converted to both directions by FORMS application. XForms W3C standard strives to be supported by browsers. The other XML forms standard is XEP-0004 Data Forms which is supported by most XMPP clients:

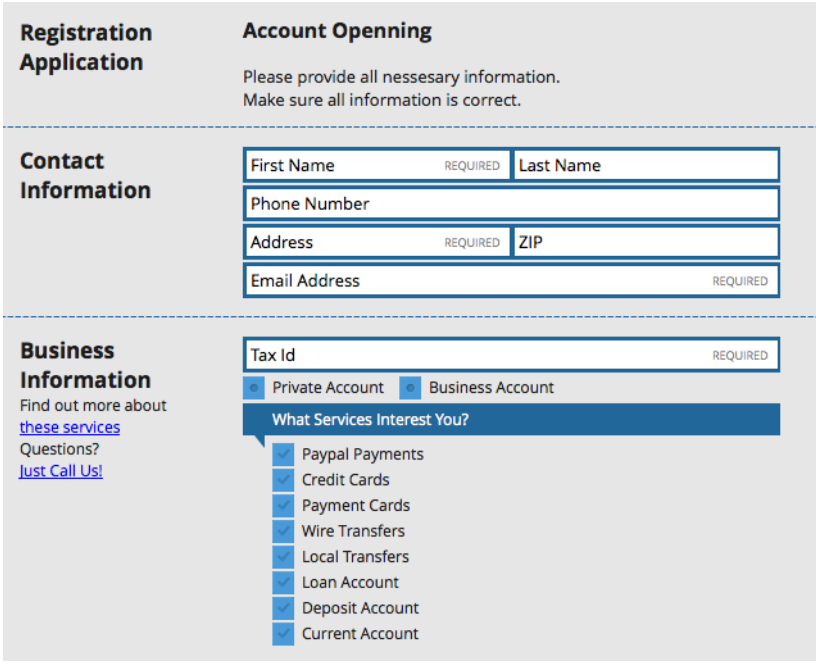

Figure 1: Form Sample

```
<x xmlns='jabber:x:data' type='{form-type}'>
  <title/><instructions/><desc/>
  <field var='OS' type='int' label='description'>
    <value>3</value>
    <option label='Windows'><value>3</value></option>
    <option label='Mac'><value>2</value></option>
    <option label='Linux'><value>1</value></option>
  </field>
</x>
```