

Xdepo - GUIDA ALL'IMPLEMENTAZIONE

Xdepo è un software libero sotto licenza LGPL <http://xdepo.org>

Sommario

Introduzione: un ambiente web per gestire documenti XML	p. 3
1. Uso delle liste di valori	p. 5
2. Uso di CForms con Xdepo	p. 6
2.1 Localizzazione dei file	p. 6
2.2 Il file forms.xml	p. 7
2.3 Il file default.xml	p. 7
2.4 Il file form.xml	p. 8
2.5 Il file display.xml	p. 8
2.6 Il file binding.xml	p. 19
2.7 Gli identificatori i18n	p. 19

*Il presente manuale, redatto in francese da **Martin Sévigny** e **Myriam Delperier** (AJLSM, Francia) e terminato il 14.03.2005, è stato tradotto in italiano da Valeria Andreoli. La traduzione è stata rivista da Giuliana De Francesco (MiBAC) in collaborazione con Alessandra Stella. La correttezza della terminologia tecnica è stata verificata da Massimiliano Filacchioni (CASPUR).*

Introduzione: un ambiente Web per gestire documenti XML

Martin Sévigny

Questo documento contiene la **guida all'implementazione** della piattaforma Xdepo. Se ne può trovare una versione per la consultazione online sul sito Web Xdepo.

Xdepo è un software libero che vi permette, in ambiente Web, di creare, importare, modificare, esportare, consultare e ricercare documenti XML che vengono gestiti, organizzati e immagazzinati mediante un sistema di database XML. La homepage di Xdepo è: <http://xdepo.org>.

Per scaricare o recuperare i più recenti codici sorgente di Xdepo occorre passare attraverso il sito di sviluppo ospitato dalla piattaforma SourceForge : <http://sourceforge.net/projects/xdepo/>. Le istruzioni per l'installazione sono disponibili anche nella presente documentazione.

Per qualsiasi domanda o commento su Xdepo, si può utilizzare la lista di discussione per posta elettronica `xdepo-users` (informazioni disponibili all'indirizzo <http://lists.sourceforge.net/lists/listinfo/xdepo-users>).

Xdepo è uno strumento che può essere installato e utilizzato così com'è. Tuttavia, la sua utilità sarà sicuramente maggiore se si prova a parametrizzarlo e a configurarlo, o anche a sviluppare un'applicazione specifica. In quest'ultimo caso, Xdepo va considerato come un set di strumenti (*toolbox*) piuttosto che come un'applicazione chiavi in mano.

La *Guida all'implementazione* è destinata agli *integratori Xdepo*, ovvero a coloro che si occuperanno di configurarlo, parametrizzarlo o di sviluppare con esso un'applicazione più specifica. La parametrizzazione, la configurazione o lo sviluppo specifico richiedono competenze e conoscenze distinte, ed è per questo che invitiamo tutti i lettori a consultare bene questa documentazione prima di ritenere che un compito sia troppo difficile da svolgere. Talvolta si tratta solo di modificare dei file di configurazione.

Questa Guida è tratta diversi argomenti che di seguito riassumiamo per orientare i lettori sui temi che desiderano approfondire.

Parametri di applicazione

In questa parte si apprende come modificare determinati parametri generali di un'applicazione Xdepo, come la homepage predefinita, le funzioni di identificazione dei titoli, i database disponibili, ecc.

Modificare l'interfaccia

In questa parte sono presentate le istruzioni per modificare diversi elementi dell'interfaccia, come l'intestazione e il fondo pagina, i colori, i font, ecc.

Testi dell'interfaccia

Questa parte contiene le istruzioni specifiche relative ai testi dell'interfaccia, che permettono in particolare di tradurli per presentare Xdepo in una lingua diversa.

Inizializzazione di una base dati

E' possibile predefinire il contenuto di una base di dati Xdepo affinché il programma la crei e vi immetta del contenuto al primo avvio. Questa parte vi permetterà di sapere come fare questa inizializzazione.

Uso delle liste di valori

Xdepo propone un'interfaccia semplificata per creare e gestire liste di termini multilingue. Queste liste possono in seguito essere usate in maschere di impostazione.

Tipi di dati e maschere

Questa parte contiene tutte le informazioni necessarie per aiutarvi a definire i vostri tipi di dati e le relative maschere. Si tratta del più importante lavoro di configurazione di un'applicazione Xdepo.

|

|

1. Uso delle liste di valori

Martin Sévigny

Nelle maschere di impostazione capita spesso che delle liste di valori vengano utilizzate per semplificare l'impostazione o per predeterminare il contenuto di un elemento del documento. Xdepo propone un'interfaccia per l'impostazione di queste liste con dei valori in più lingue.

Le liste di Xdepo sono dei documenti XML che rispettano una struttura particolare. Xdepo offre poi degli strumenti che permettono di utilizzare queste liste in contesti diversi, purché rispettino tale formato. Per creare una lista, vi invitiamo a consultare il documento sulla [creazione di documenti](#) della *Guida per l'utente*. In questa sezione vi indicheremo come utilizzare queste liste.

2. Uso di CForms con Xdepo

Martin Sévigny e Myriam Delperier

Le maschere che permettono di modificare ed impostare documenti XML in Xdepo si basano sulla tecnologia *CForms* del progetto Apache Cocoon. Per agevolare lo sviluppo di determinate maschere e per permetterne la perfetta integrazione nell'ambiente Xdepo, è tuttavia necessaria qualche nota di uso. Questo documento presenta queste note e i principali elementi CForms.

Invitiamo vivamente i lettori che desiderano sviluppare maschere di immissione proprie a consultare la documentazione [CForms](#) di Cocoon, oltre agli esempi forniti da questa piattaforma.

2.1 Localizzazione dei file

Ciascun documento modificabile tramite una maschera in Xdepo è associato a un *tipo di dati*. Per ciascun tipo di dati si possono avere più maschere di immissione. Xdepo riconosce i tipi di dati e le maschere messe a disposizione da un'installazione esplorando le cartelle ed i file disponibili a un determinato percorso.

Prendiamo come esempio un tipo di dati `dc` per la base di dati il cui codice è `test`. Se ci sono delle maschere (*form*) associate a questo tipo di dati, la struttura dei file sarà la seguente:

```
01 xdepo-local/datatypes
02     test
03         dc
04             forms
05                 forms.xml
06                     form1
07                         binding.xml
08                         default.xml
09                         display.xml
10                         form.xml
11                 form2
12                     binding1.xml
13                     binding2.xml
14                     default.xml
15                     display1.xml
16                     display2.xml
17                     form1.xml
18                     form2.xml
```

Alla riga 5, il file `forms.xml` fornisce delle informazioni sulle maschere disponibili, permettendo di dare loro un nome in diverse lingue. Questo nome apparirà nella lista scorrevole del menu di creazione di un documento.

Di seguito, le cartelle `form1` e `form2` contengono tutti e due i file di specificazione di una maschera. Ci sono sempre quattro tipi di file per ciascun maschera: `binding`, `default`, `display` e `form`. Il ruolo di questi file viene spiegato più avanti nel corso di questo documento.

Se una maschera viene presentata su più pagine, bisogna ripetere i file `binding`, `display` e `form` aggiungendo come suffisso una cifra (1, 2, ...) che indica l'ordine delle pagine. La maschera `form2` nell'esempio riportato qui sopra, contiene file di specifiche per una maschera di due pagine.

2.2 Il file `forms.xml`

In questo file indicherete la lista delle maschere disponibili per un particolare tipo di dati.

Questa lista rispetta il formato delle liste Cforms/Xdepo, come illustrato nell'esempio seguente:

```
<fd:selection-list
  xmlns:fd="http://apache.org/cocoon/forms/1.0#definition">
  <fd:item value="form1">
    <fd:label>Mon premier formulaire</fd:label>
  </fd:item>
  <fd:item value="form2">
    <fd:label>Mon second formulaire</fd:label>
  </fd:item>
</fd:selection-list>
```

Si noti che per le etichette, inserite nell'elemento `<fd:label>`, è preferibile utilizzare degli identificatori i18n per poterle visualizzare in diverse lingue.

2.3 Il file `default.xml`

Questo file contiene il modello predefinito dei documenti creati tramite questa maschera. Un modello è un documento XML che contiene dei valori già impostati o perlomeno la struttura di base.

Ecco alcune considerazioni a proposito di questo file e più in generale dei modelli di documenti:

- se un elemento di `default.xml` non è trattato dalla maschera, verrà ricopiato così com'è nel documento;
- se viene trattato, verrà sostituito dal o dai valori forniti dal maschera;
- Il maschera può trattare degli elementi non presenti nel modello;
- il valore di un elemento del modello varrà da valore predefinito al momento della presentazione dell'elemento nella maschera.

E' obbligatorio aggiungere l'attributo `xdepo:datatype` all'elemento radice di ogni modello, incluso il file `default.xml`. Il valore dell'attributo `xdepo:datatype` deve essere un percorso del tipo `datatypes/[codice base di dati]/forms/form[numero della maschera]`. L'elemento radice può anche includere l'attributo `xdepo:savetype`, per il quale sono possibili 3 valori:

- 1 `auto` : l'identificatore Xdepo del documento sarà generato automaticamente;
- 2 `set` : l'identificatore Xdepo del documento viene inserito manualmente al momento della sua creazione;
- 3 `append` : l'identificatore Xdepo è il frutto della concatenazione di un identificatore automatico e di uno impostato manualmente.

Non bisogna dimenticare di dichiarare lo spazio dei nomi Xdepo in questo documento:

```

<test xmlns:xdepo="http://xdepo.org/xdepo/1.0"
  xdepo:datatype="datatypes/test/forms/form1">
  <identifier>Identifiant par défaut</identifier>
  <title>Titre par défaut</title>
</test>

```

2.4 Il file `form.xml`

In questo file si dichiarano i diversi controlli (*widgets*) presenti nella maschera. Si può quindi considerare che questo file contiene la definizione di base degli elementi della maschera. Nella terminologia Cocoon, si tratta del *form definition file*.

Per informazioni complete sulla struttura di questo file e dei controlli che può contenere si rinvia alla [documentazione di Cocoon](#) su questo tema, e in particolare alla parte sui *widgets*.

2.5 Il file `display.xml`

Questo file servirà per l'impaginazione della maschera, dal momento che si basa sulla definizione dei controlli nel file `form.xml`. Nella terminologia Cocoon, si tratta del *form template*.

In questo documento si può inserire il contenuto che si desidera, in quanto esso definisce solo l'impaginazione delle maschere. Tuttavia, se volete restare nell'ambito della logica delle maschere Xdepo e della loro integrazione nell'applicazione, ecco qualche utile indicazione.

L'elemento radice è `document`. L'elemento `header`, invece, consente di specificare il titolo della pagina e i link ai fogli di stile CSS.

```

<header>
  <title>
    <i18n:text>Test Form</i18n:text>
  </title>
  <style href="lib/xdepo/css/viewer.css"/>
</header>

```

Successivamente, si può inserire nel corpo del documento un'intestazione che consentirà di visualizzare l'identificatore del documento in corso di creazione (se esiste):

```

<body>
  <div class="main">
    <div class="infocontext">
      <p><i18n:text>Test Form</i18n:text> : ${cocoon.request.doc}</p>
    </div>
    ...
  </div>
</body>

```

Poi si può inserire la maschera vera e propria. Nell'ambiente Xdepo è necessario fornire le seguenti informazioni:

- azione : `#{$continuation/id}.continue?uri=${cocoon.request.uri}&doc=$`

```
{cocoon.request.doc}
```

- metodo : POST
- in caso di caricamento di documenti (*file upload*), l'attributo `enctype="multipart/form-data"`

Ecco ciò che ne può risultare:

```
<ft:form-template
  method="POST"
  action="#{$continuation/id}.continue?uri=${cocoon.request.uri}&doc=${
cocoon.request.doc}"
  enctype="multipart/form-data">
```

Si può di seguito inserire la formattazione degli errori di validazione, solo per averli all'inizio della pagina, come riportato qui di seguito :

```
<fi:validation-errors>
  <header>
    <i18n:text key="validation.errors.header"/>
  </header>
  <footer>
    <i18n:text key="validation.errors.footer"/>
  </footer>
</fi:validation-errors>
```

Il risultato sullo schermo si presenterà così:

Collection Form : test-collection.xml

Veuillez corriger les erreurs suivantes:

- category : general.field-required

Soumettez ensuite le formulaire.

Per la formattazione della maschera, utilizzare una tabella così definita:

```
<table align="center" class="general">
  <tbody>
    l'intestazione
    gli elementi della maschera
    la navigazione
  </tbody>
</table>
```

L'intestazione della maschera avrà la seguente struttura HTML:

```
<tr>
  <td colspan="2" align="center" class="formHeader">Description</td>
```

```
</tr>
```

A schermo si vedrà una formattazione simile alle seguente :



Per navigare tra le diverse pagine di una maschera multipagina, si possono utilizzare tre tipi di pulsanti così definiti:

1 precedente: `<input type="submit" name="prev" value="Prev" i18n:attr="value"/>`

2 successivo: `<input type="submit" name="next" value="Next" i18n:attr="value"/>`

3 salvare: `<input type="submit" name="save" value="Save" i18n:attr="value"/>`

La struttura HTML sarà del tipo:

```
<tr>
  <td colspan="1" align="left">
    <input type="submit" name="prev" value="Prev" i18n:attr="value"/>
  </td>
  <td colspan="1" align="right">
    <input type="submit" name="save" value="Save" i18n:attr="value"/>
  </td>
</tr>
```

Sullo schermo ritroverete una formattazione di questo tipo:



I singoli controlli non ripetibili in una maschera verranno trattati secondo il seguente modello:

```
<tr class="groupe">
  <td class="label">
    etichetta del controllo
```

```

    </td>
    <td class="value">
        controllo
    </td>
</tr>

```

Il risultato a schermo sarà simile a:

L'elemento `<ft:widget-label>` contiene l'etichetta associata al singolo controllo.

L'elemento `<ft:widget>` permette di visualizzare il controllo vero e proprio. Ad esso si può aggiungere un sotto elemento `<fi:styling>` per modificare la formattazione predefinita del controllo.

L'elemento `<fd:field>`, senza lista, definisce un controllo di tipo area di testo semplice. Con un sotto elemento `fi:styling` si può alterare l'aspetto, classificare, dare una misura a questo controllo.

La struttura da utilizzare sarà del tipo:

```

<ft:widget id="id1">
    <fi:styling size="40" style="background-color:purple; color:white;"/>
</ft:widget>
<ft:widget id="id2">
    <fi:styling class="forms-field-required"/>
</ft:widget>

```

A schermo otterrete una formattazione di questo tipo:

L'elemento `fi:styling type="hidden"` permette di definire un campo nascosto.

L'elemento `fi:styling type="textarea"` permette di visualizzare un'area di testo con più righe. Per esempio:

```

<ft:widget id="id3">
    <fi:styling type="textarea" style="background-color:yellow;"/>
</ft:widget>

```

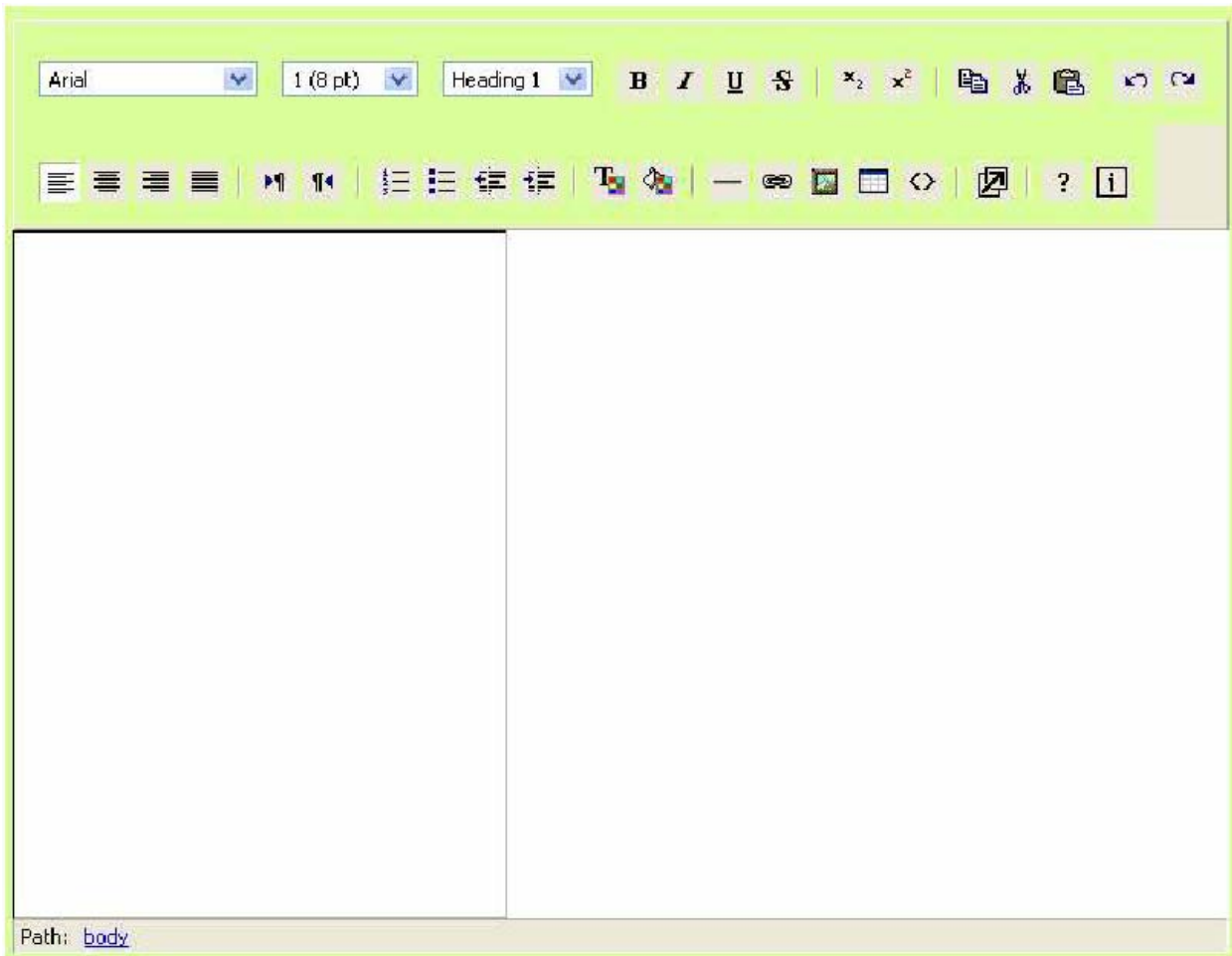
Il formato di visualizzazione sarà come il seguente:



L'elemento `fi:styling type="htmlarea"` permette di visualizzare un'area di testo ricca. Per esempio:

```
<ft:widget id="id4">
  <fi:styling type="htmlarea" />
</ft:widget>
```

Sullo schermo otterrete un controllo complesso come questo:



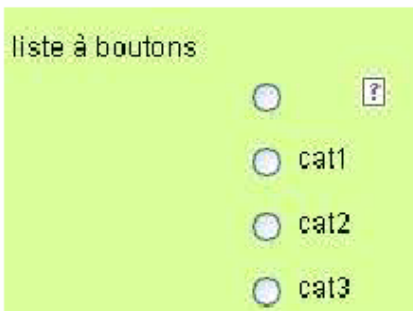
Vi sono varie possibilità per la visualizzazione di lista. La visualizzazione predefinita è la seguente:



Per un output con dei pulsanti radio, utilizzare la formattazione `fi:styling list-type="radio"` come segue:

```
<ft:widget id="category6">
  <fi:styling list-type="radio"/>
</ft:widget>
```

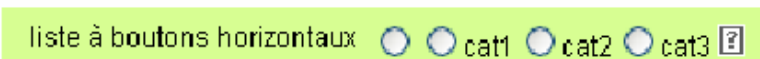
Otterrete sullo schermo qualcosa del genere:



Potete disporre gli stessi pulsanti radio in orizzontale modificando leggermente le istruzioni di formattazione con `fi:styling list-type="radio" list-orientation="horizontal"`, come in questo esempio:

```
<ft:widget id="category6">
  <fi:styling list-type="radio" list-orientation="horizontal"/>
</ft:widget>
```

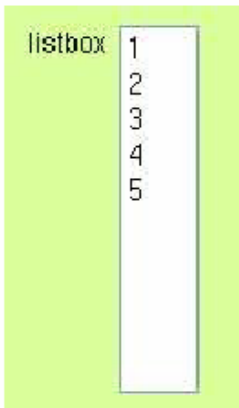
La formattazione risulterà così:



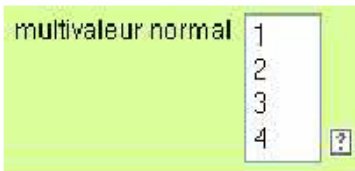
Per ottenere una lista non scorrevole, ma ad altezza fissa, utilizzare `fi:styling listtype="listbox" listbox-size=""`, come segue:

```
<ft:widget id="category2">
  <fi:styling list-type="listbox" listbox-size="10"/>
</ft:widget>
```

Otterrete così sullo schermo:



Se optate per una lista con possibilità di scelta multipla, utilizzare il codice `fd:multivaluefield` e l'output predefinito sarà del tipo:



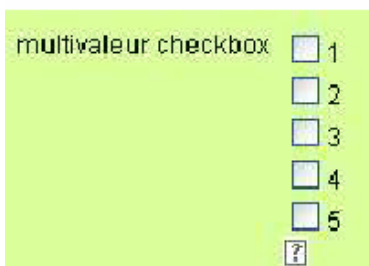
Il codice `fi:styling list-type="listbox" listbox-size=""` permette di specificare la dimensione della lista multipla. Attenzione, in questo caso specifico non ci saranno pulsanti di scorrimento, quindi si deve specificare una dimensione non inferiore al numero dei valori della lista.

```
<ft:widget id="toto">
  <fi:styling list-type="listbox" listbox-size="5"/>
</ft:widget>
```

Per presentare i valori della lista sotto forma di caselle da spuntare (checkbox), utilizzare `fi:styling list-type="checkbox"`, come nel seguente esempio:

```
<ft:widget id="toto">
  <fi:styling list-type="checkbox"/>
</ft:widget>
```

La formattazione risulterà così:



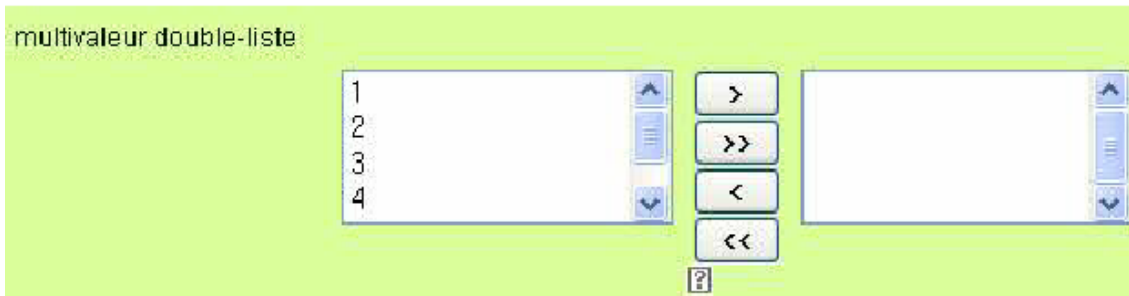
Per ottenere una lista doppia, vale a dire una lista che permette di selezionare degli item e un'altra che permette di vedere gli item già selezionati, utilizzare `fi:styling list-type="double-listbox" listbox-size=""`, come in questo esempio:

```

<ft:widget id="toto2">
  <fi:styling list-type="double-listbox" listbox-size="4"/>
</ft:widget>

```

La formattazione risulterà come segue:



Spesso è interessante raggruppare dei comandi per indicare all'utente che questi hanno delle caratteristiche comuni o che si riferiscono allo stesso tipo di informazione. Si può utilizzare a tale scopo una struttura HTML come questa:

```

<tr class="groupe">
  <td colspan="2">
    <fi:group>
      <fi:styling type="tabs"/>
      <fi:state>
        <ft:widget id="tab-state"/>
      </fi:state>
      <fi:items>
        <fi:group>
          <fi:label>Per colonne</fi:label>
          <fi:styling layout="columns"/>
          <fi:items>
            <ft:widget id="identifiant"/>
            <ft:widget id="titre"/>
            <ft:widget id="access-control"/>
          </fi:items>
        </fi:group>
        <fi:group>
          <fi:label>Per righe</fi:label>
          <fi:styling layout="rows"/>
          <fi:items>
            <ft:widget id="size"/>
            <ft:widget id="accrual"/>
          </fi:items>
        </fi:group>
        <fi:group>
          <fi:label>Per colonna</fi:label>
          <fi:styling layout="column"/>
          <fi:items>
            <ft:widget id="standards"/>
            <ft:widget id="legal-status"/>
          </fi:items>
        </fi:group>
        <fi:group>
          <fi:label>Per riga</fi:label>
          <fi:styling layout="row"/>
          <fi:items>

```

```

        <ft:widget id="title2"/>
        <ft:widget id="title3"/>
    </fi:items>
</fi:group>
</fi:items>
</fi:group>
</td>
</tr>

```

Con il codice `fi:styling type="tabs"` si otterrà un raggruppamento per schede come nella seguente illustrazione:

Avec des onglets

Par colonnes Par lignes Par colonne Par ligne

identifïer ?

title ?

access-control ?

Se si impiega il codice `fi:styling type="choice"`, il raggruppamento è fatto con una selezione tramite lista scorrevole:

Avec liste déroulante

Fais ton choix ▼

title ? title ?

Il codice `fi:styling type="fieldset"` produce il raggruppamento in un *fieldset* HTML come questo:

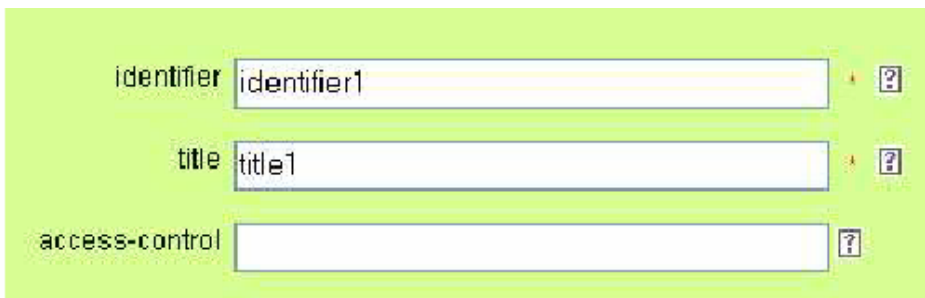
Avec fieldset

titre du fieldset

identifïer ?

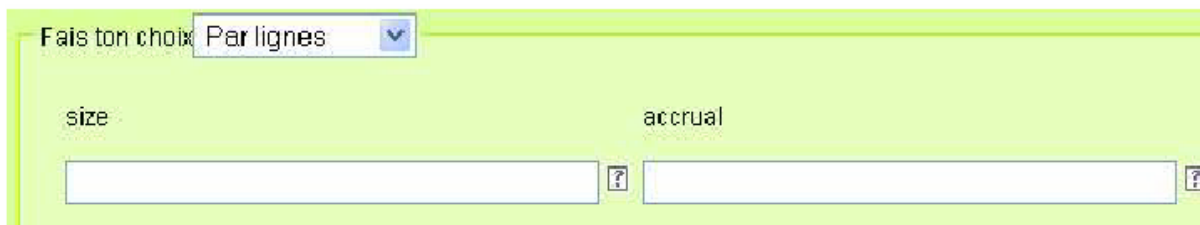
title ?

Con un codice `fi:styling layout="columns"` otterrete una colonna per le etichette e una colonna per i campi:



A screenshot of a form with a light green background. It features three rows of controls. The first row has the label 'identifier' on the left and a text input field containing 'identifier1' on the right, with a small question mark icon to the right of the field. The second row has the label 'title' on the left and a text input field containing 'title1' on the right, also with a question mark icon. The third row has the label 'access-control' on the left and an empty text input field on the right, with a question mark icon.

Il codice `fi:styling layout="rows"` permette di ottenere una riga per le etichette e una per i campi:



A screenshot of a form with a light green background. At the top, there is a label 'Fais ton choix' followed by a dropdown menu showing 'Par lignes'. Below this, there are two rows of controls. The first row has the label 'size' on the left and an empty text input field on the right, with a question mark icon. The second row has the label 'accrual' on the left and an empty text input field on the right, with a question mark icon.

Il codice `fi:styling layout="column"` permette di ordinare tutto sulla stessa colonna:



A screenshot of a form with a light green background. It features three rows of controls. The first row has the label 'standards' on the left and an empty text input field on the right, with a question mark icon. The second row has the label 'legal-status' on the left and a text input field containing 'status1' on the right, with a question mark icon. The third row has the label 'title' on the left and an empty text input field on the right, with a question mark icon.

Il codice `fi:styling layout="row"` permette di ottenere tutto sulla stessa riga:



A screenshot of a form with a light green background. It features a single row of controls. On the left, there is a label 'title' followed by an empty text input field with a question mark icon. To the right of this field, there is another label 'title' followed by another empty text input field with a question mark icon.

Certi controlli possono essere ripetibili. Per visualizzarli correttamente, includendo i pulsanti che permettono di aggiungere o di eliminare le singole istanze dei controlli, utilizzare nel file `display.xml` il seguente codice HTML:

```

<tr class="groupe">
  <td colspan="2">
    <fi:group>
      <fi:styling type="fieldset"/>
      <fi:label>
        l'etichetta del repeater
        il pulsante aggiungi
      </fi:label>
      <fi:items>
        <ft:repeater-widget id="descriptions">
          <fi:group>
            *****quando ci sono dei sotto elementi*****
            <fi:styling type="fieldset"/>
            <fi:label>
              il pulsante rimuovi
              il pulsante su
              il pulsante giù
            </fi:label>
            <fi:items>
              <table align="center">
                <tbody>
                  i sotto elementi
                </tbody>
              </table>
            </fi:items>
            *****senza sotto elementi*****
            <fi:label/>
            <fi:items>
              il pulsante rimuovi
              il pulsante su
              il pulsante giù
              il valore dell'elemento
            </fi:items>
          </fi:group>
        </ft:repeater-widget>
      </fi:items>
    </fi:group>
  </td>
</tr>

```

L'elemento `ft:repeater-widget-label id="" widget-id=""` permette di visualizzare l'etichetta del controllo ripetibile.

Il pulsante per l'aggiunta dell'istanza di un controllo si può includere così:

```

<ft:widget id="add{id}">
  <xdepo:button type="img" role="add" code="{id}"/>
</ft:widget>

```

Il pulsante per eliminare l'istanza di un controllo può essere incluso così:

```

<ft:widget id="remove{id}">
  <xdepo:button type="img" role="remove" code="{id}"/>
</ft:widget>

```

Il pulsante che permette di anteporre l'istanza di un controllo a quella che lo precede si può inserire così:

```

<ft:widget id="up{id}">

```

```

        <xdepo:button type="img" role="up" code="{id}"/>
    </ft:widget>

```

Il pulsante che permette di spostare l'istanza di un controllo dopo quella che la segue può essere inserito così:

```

<ft:widget id="down{id}">
    <xdepo:button type="img" role="down" code="{id}"/>
</ft:widget>

```

Questi vari pulsanti si presenteranno così:



2.6 Il file `binding.xml`

Questo file contiene le istruzioni che permettono di collegare gli elementi della maschera di immissione con la struttura XML del documento da modificare o da creare. Così, se volete che il controllo del nome *id* della maschera diventi l'elemento *identifier* del documento XML, lo dovrete specificare nel file `binding.xml`. Nella terminologia Cocoon, si tratta del *binding framework*.

2.7 Gli identificatori i18n

E' possibile generare il contenuto testuale delle pagine delle maschere mediante istruzioni i18n di Cocoon, come segue:

```

<i18n:text i18n:catalogue="forms" key="test.identifier">Identificativo
predefinito</i18n:text>

```

Ciò significa che se disponete di un catalogo `michael-forms` e se questo possiede una voce del tipo `<message key="test.identifier">Il mio identificativo</message>`, allora il vostro identificatore i18n sarà sostituito con il testo fornito dal catalogo, qui *Il mio identificatore*. In caso contrario il valore sarà il contenuto dell'identificatore i18n, vale a dire *Identificatore predefinito*.

I cataloghi sono definiti nel file `xdepo-system/sitemap/sitemap1.xmap` in questo modo:

```

<catalogue id="michael-forms" name="michael-forms" location="xdepo-
local/translations"/>

```

Il valore dell'attributo `id` corrisponde al valore dell'attributo `i18n:catalogue` dell'elemento `i18n:text`. Il valore dell'attributo `name` identifica l'inizio del nome del file XML corrispondente al catalogo; il nome completo di tale file e' dato dal valore dell'attributo `name` seguito da `_{codice_paese}.xml`.

L'attributo `location` indica la localizzazione del catalogo. Per una localizzazione franco-italiana, un catalogo corrisponderà ai file:

- xdepo-local/translations/michael-forms_fr.xml
- xdepo-local/translations/michael-forms_it.xml

La forma del catalogo sarà:

```
<catalogue xml:lang="{la locale ex: fr}">
  <message key="test.identifier">identifier</message>
  <messages key="...">...</message>
  ...
</catalogue>
```

In un identificatore è possibile specificare un attributo `i18n:attr={un_nome_d'attributo_dell'identificatore un_altro_nome ...}` (ciascun valore separato da uno spazio) affinché questi attributi siano trattati secondo il meccanismo `i18n`.

La documentazione di Cocoon fornisce maggiori informazioni sui meccanismi `i18n`.