ページ要素をデータベースに直接結び付けるフレームワークとWebサイト設計手法

新居雅行 - Masayuki Nii

情報処理学会
第153回データベースシステム研究会
B-3-4
2011/11/3 @エステック情報ビル

Agenda

Webフレームワークの現状

独自に開発した「INTER-Mediator」の特徴

クライアントサイドでの再帰的にノード複製を行うテンプレート処理の手法

処理の拡張作業

開発手法に関する考察

Webフレームワークの現状

MVC2モデルをベースにしたフレームワーク

- JavaベースのStruts等
- 大規模なシステム開発に利用される
- Ruby on Rails、PHPの各種フレームワークなども利用が進む

HTMLを拡張したフレームワーク

- 過去に存在したが、現在はあまり利用されていない
- 簡単に作成はできるものの、複雑なシステムではかえって大変

大規模開発に向けての仕組みは進展しているが、小規模なWebシステムを作る手法はむしろ後退している

小規模な開発に対する問題点

小規模開発では複雑さを排除したい

“言語の混合” が多くの場合に当たり前となっている

- <input type=”text” value=”<php echo $row[‘name’] ?>”>
- 複雑さを増大させることになる

単にテーブルのデータをそのまま使うことができない

- MVC2、レイヤー構造等により複雑なものに対応できるが、単純なアプリケーション開発でも各レイヤーの対処が必要
- デスクトップアプリケーションの「FileMaker Pro」「Microsoft Access」がさまざまな用途に使われている。そのテイストをWebアプリケーションでも取り入れたい

簡単な仕組みしか組み込めないと開発がより困難に

- 古い時代のフレームワーク問題点は解消しないといけない

開発したフレームワークの特徴

独自にフレームワーク「INTER-Mediator」を開発

独自言語や独自タグを使用しない

- HTMLのclass属性に指定を加える

HTML要素とテーブルのフィールドが連携する

- JavaScriptでテンプレート処理を進める、クライアントサイドのテンプレート処理
- →クライアントサイドでのバインディングはjQuery等で実装されているが、JavaScriptのプログラムが必要
- ノードの複製により「繰り返し」を処理する

連携する以上の機能の実現

- サーバ側での処理の追加が可能
- クライアント側での処理の追加が可能

ページ作成の手法

データベースを用意する

「定義ファイル」を用意する

- テーブル情報やリレーション情報などを記述する

HTMLを記述する

- class属性に「ターゲット指定」を追記する。ターゲット指定のあるノードを「リンクノード」と呼ぶ
- ターゲット指定 := コンテキスト名@フィールド名@ターゲット
- コンテキスト名の典型的な例は「テーブル名」
- リンク要素の子ノードに、フィールドのデータをテキスト要素として追加する。innerHTMLへの設定も可能
- ターゲットの記述により、要素の属性やスタイルへ、フィールドのデータを設定できる（追記も可能）

定義ファイルの例

```php
require_once ('../INTER-Mediator/INTER-Mediator.php');

IM_Entry(
    array(
        array(
            'records' => 1,
            'paging' => true,
            'name' => 'person',
            'key' => 'id',
            'query' => array( /* array( 'field'=>'id', 'value'=>'5', 'operator'=>'eq' ),*/),
            'sort' => array(array('field' => 'id', 'direction' => 'asc'),),
            'repeat-control' => 'insert delete',
        ),
        array('name' => 'contact',
            'key' => 'id',
            'relation' => array(
                array('foreign-key' => 'person_id', 'join-field' => 'id', 'operator' => '=')
            ),
            'repeat-control' => 'insert delete',
        ),
        array('name' => 'contact_way',
            'key' => 'id',
        ),
        array('name' => 'cor_way_kindname',
            'key' => 'id',
            'relation' => array(
                array('foreign-key' => 'way_id', 'join-field' => 'way', 'operator' => '=')
            ),
            'foreign-key' => 'way_id',
            'join-field' => 'way'
        ),
        array('name' => 'history',
            'key' => 'id',
            'relation' => array(
                array('foreign-key' => 'person_id', 'join-field' => 'id', 'operator' => '=')
```

HTMLファイルの例

```
<html>
<head>
    <meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
    <link rel="stylesheet" type="text/css" href="../sample.css"/>
    <title>INTER-Mediator - Sample - Form Style/MySQL</title>
    <script src="include_MySQL.php"></script>
    <script type="text/javascript"></script>
</head>
<body onload="if(INTERMediatorCheckBrowser()){INTERMediator.construct(true)}">
<div id="IM_NAVIGATOR">Navigation Controls by INTER-Mediator</div>
<table border="1">
    <tbody>
    <tr>
        <th>id</th>
        <td>
            <div class="IM[person@id]"></div>
        </td>
        <th>category</th>
        <td>
            <select class="IM[person@category]">
                <option value="101">Family</option>
                <option value="102">ClassMate</option>
                <option value="103">Collegue</option>
            </select>
        </td>
        <th>check</th>
        <td><input type="checkbox" title="person@checking" value="1"/></td>
    </tr>
    <tr>
        <th>name</th>
        <td colspan="5"><input type="text" class="IM[person@name]" value=""/></td>
    </tr>
    <tr>
        <th>mail</th>
        <td colspan="5"><input type="text" class="IM[person@mail]" value=""/></td>
```

テンプレート処理の基本

エンクロージャー／リピーター

- リンクノードとその上位ノードをたどり、リピータとエンクロージャを決定する

	エンクロージャー	リピーター
テーブル	tbody	tr
汎用	div [_im_enclosure]	div [_im_repeater]
汎用	span [_im_enclosure]	span [_im_repeater]
番号リスト	ol	li
箇条書き	ul	li
ポップアップ	select	option

1つのテーブルを表に展開する例

次以降のスライドで流れを説明

ポイント

- エンクロージャー／リピータの識別
- ノード複製により、複数のレコードを表に展開

リンクノード

エンクロージャー

リピーター

```
<table>
  <tbody>
  <tr>
     <td class="IM[postalcode@pcode]"></td>
     <td class="IM[postalcode@address]"></td>
  </tr>
  </tbody>
</table>
```

ターゲット指定

定義ファイル

```
array(
  name => postalcode )
```

table: postalcode

pcode	address
102-0094	千代田区紀尾井町
121-0813	足立区竹の塚
161-0035	新宿区中井

```
<table>
    <tbody>

    </tbody>
</table>
```

リピーターをいったん削除し保持

```
    <tr>
        <td class="IM[postalcode@pcode]"></td>
        <td class="IM[postalcode@address]"></td>
    </tr>
```

定義ファイル

```
array(
  name => postalcode )
```

table: postalcode

pcode	address
102-0094	千代田区紀尾井町
121-0813	足立区竹の塚
161-0035	新宿区中井

```
<table>
  <tbody>
   <tr>
      <td class="IM[postalcode@pcode]"> 102-0094 </td>
      <td class="IM[postalcode@address]"> 千代田区紀尾井町 </td>
   </tr>
  </tbody>
</table>
```

リピーターを複製し、エンクロージャーに追加

レコードを挿入

Repeater Nodes

```
<tr>
   <td class="IM[postalcode@pcode]"></td>
   <td class="IM[postalcode@address]"></td>
</tr>
```

定義ファイル

```
array(
  name => postalcode )
```

table: postalcode

pcode	address
102-0094	千代田区紀尾井町
121-0813	足立区竹の塚
161-0035	新宿区中井

リピーターを複製し、エンクロージャーに追加

レコードを挿入

```
<table>
   <tbody>
   <tr>
      <td class="IM[postalcode@pcode]"> 102-0094 </td>
      <td class="IM[postalcode@address]"> 千代田区紀尾井町 </td>
   </tr>
   <tr>
      <td class="IM[postalcode@pcode]"> 121-0813 </td>
      <td class="IM[postalcode@address]"> 足立区竹の塚 </td>
   </tr>
   </tbody>
</table>
```

Repeater Nodes

定義ファイル

```
array(
  name => postalcode )
```

table: postalcode

pcode	address
102-0094	千代田区紀尾井町
121-0813	足立区竹の塚
161-0035	新宿区中井

リレーションのあるテーブルを表に展開

次以降のスライドで流れを説明

ポイント

- リレーションによる1対多の展開

定義ファイル

```
array(
  name=>place
  key=>id ),
array(
  name=>postalcode
  relation=> array(
    foreign-key=>place_id,
    join-field=>id ),),
```

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
<table>
 <tbody>
<tr>
    <td class="IM[place@city]"></td>
    <td>
        <table>
          <tbody>
            <tr>
                <td class="IM[postalcode@pcode]"></td>
                <td class="IM[postalcode@town]"></td>
            </tr>
          </tbody>
        </table>
    </td>
</tr>
 </tbody>
</table>
```

エンクロージャー

リピーター

エンクロージャー

リピーター

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
<table>
    <tbody>

    </tbody>
</table>
```

リピーターをいったん削除し保持

```
        <tr>
            <td class="IM[place@city]"></td>
            <td>
                <table>
                    <tbody>
                        <tr>
                            <td class="IM[postalcode@pcode]"></td>
                            <td class="IM[postalcode@town]"></td>
                        </tr>
                    </tbody>
                </table>
            </td>
        </tr>
```

```
<table>
  <tbody>
    <tr>
      <td class="IM[place@city]">千代田区</td>
      <td>
        <table>
          <tbody>
            <tr>
              <td class="IM[postalcode@pcode]"></td>
              <td class="IM[postalcode@town]"></td>
            </tr>
          </tbody>
        </table>
      </td>
    </tr>
  </tbody>
</table>
```

リピーターを複製し、エンクロージャーに追加

レコードを挿入

Repeater Nodes

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
<table>
    <tbody>
    <tr>
        <td class="IM[place@city]">千代田区</td>

        <td>
            <table>
                <tbody>


                </tbody>
            </table>
        </td>
    </tr>
    </tbody>
</table>
```

Repeater Nodes 1

リピーターをいったん削除し保持

Repeater Nodes 2

```
<tr>
        <td class="IM[postalcode@pcode]"></td>
        <td class="IM[postalcode@town]"></td>
</tr>
```

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

postalcodeテーブル側の展開

```
<table>
  <tbody>
   <tr>
     <td class="IM[place@city]">千代田区</td>
     <td>
       <table>
         <tbody>
           展開中の
           リピーター
           id=901
         </tbody>
       </table>
     </td>
   </tr>
  </tbody>
</table>
```

Repeater Nodes 1

Repeater Nodes 2

```
<tr>
   <td class="IM[postalcode@pcode]"></td>
   <td class="IM[postalcode@town]"></td>
</tr>
```

定義ファイル

```
array(
  name=>postalcode
  relation=> array(
   join-field=>id,
   foreign-key=>place_id ),),
```

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
<table>
  <tbody>
  <tr>
    <td class="IM[place@city]">千代田区</t
    <td>
      <table>
        <tbody>
          <tr>
            <td class="IM[postalcode@pcode]"> 102-0094 </td>
            <td class="IM[postalcode@town]"> 紀尾井町 </td>
          </tr>
        </tbody>
      </table>
    </td>
  </tr>
  </tbody>
</table>
```

Repeater Nodes 1

Repeater Nodes 2

```
<tr>
  <td class="IM[postalcode@pcode]"></td>
  <td class="IM[postalcode@town]"></td>
</tr>
```

レコードを挿入

リピーターを複製し、エンクロージャーに追加

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
<table>
  <tbody>
  <tr>
    <td class="IM[place@city]">千代田区</td>
    <td>
      <table>
        <tbody>
          <tr>
            <td class="IM[postalcode@pcode]"> 102-0094 </td>
            <td class="IM[postalcode@town]"> 紀尾井町 </td>
          </tr>
          <tr>
            <td class="IM[postalcode@pcode]"> 100-0011 </td>
            <td class="IM[postalcode@town]"> 内幸町 </td>
          </tr>
        </tbody>
      </table>
    </td>
  </tr>
  </tbo
</table>
```

Repeater Nodes 1

Repeater Nodes 2

リピーターを複製し、エンクロージャーに追加

レコードを挿入

table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

```
                <td class="IM[postalcode@pcode]"> 100-0011 </td>
                <td class="IM[postalcode@town]"> 内幸町 </td>

            </tr>
          </tbody>
        </table>
    </td>
</tr>
<tr>
    <td class="IM[place@city]">足立区</td>

    <td>
        <table>
            <tbody>
                <tr>
                    <td class="IM[postalcode@pcode]"> 121-0813 </td>
                    <td class="IM[postalcode@town]"> 竹の塚 </td>

                </tr>
            </tbody>
        </table>
    </td>
</tr>
</tbody>
</table>
```

Repeater Nodes 1

Repeater Nodes 2

定義ファイル

```
array(
  name=>postalcode
  relation=> array(
    join-field=>id,
    foreign-key=>place_id ),),
```

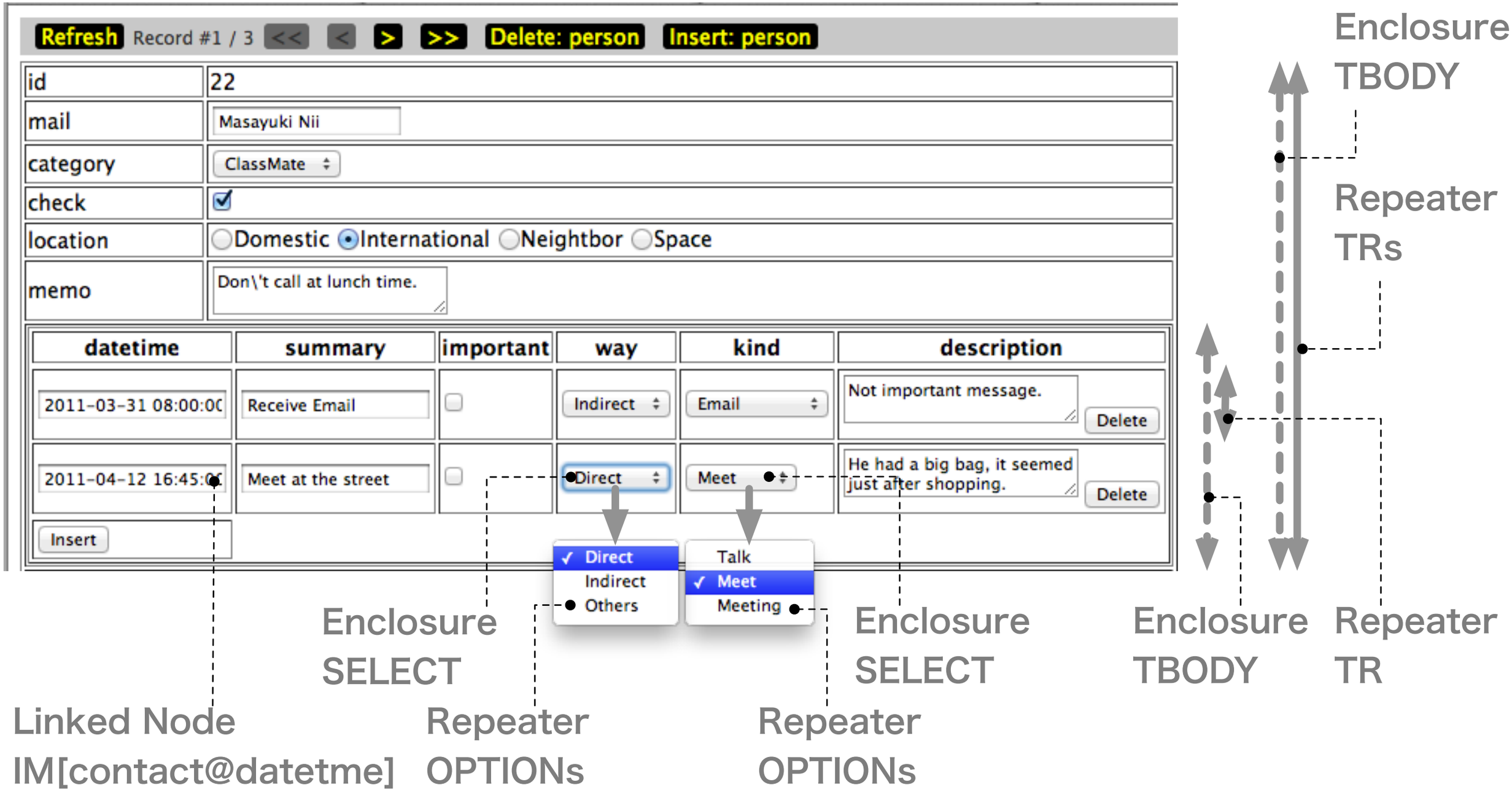
table: place

id	city
901	千代田区
902	足立区

table: postalcode

place_id	pcode	address
901	102-0094	紀尾井町
901	100-0011	内幸町
902	121-0813	竹の塚

再帰的に展開


Refresh
Record #1 / 3
<<
<
>
>>
Delete: person
Insert: person
id
22
mail
Masayuki Nii
category
ClassMate
check
location
Domestic
International
Neightbor
Space
memo
Don\'t call at lunch time.
datetime
summary
important
way
kind
description
2011-03-31 08:00:0(
Receive Email
Indirect
Email
Not important message.
Delete
2011-04-12 16:45:0(
Meet at the street
Direct
Meet
He had a big bag, it seemed just after shopping.
Delete
Insert
Direct
Indirect
Others
Talk
Meet
Meeting
Enclosure TBODY
Repeater TRs
Enclosure SELECT
Enclosure SELECT
Enclosure TBODY
Repeater TR
Linked Node IM[contact@datetme]
Repeater OPTIONs
Repeater OPTIONs

データベースへの書き戻し

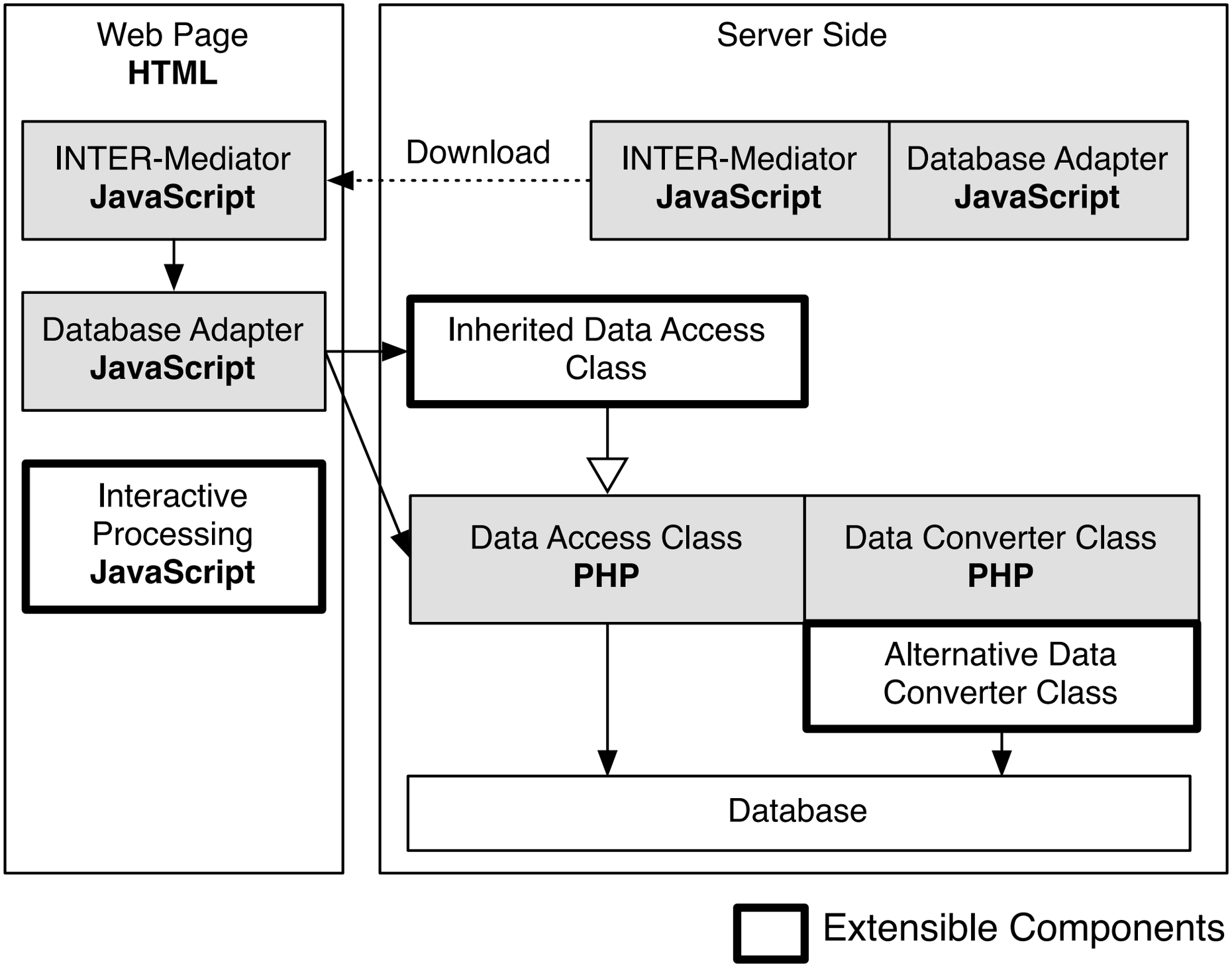
リンクノードへデータを設定時

- テーブル名、フィールド名、レコードのキー値、フィールドの値をJavaScriptのオブジェクトに記録
- changeイベントにより関数を呼び出すように設定

データ修正時

- 記録した情報からデータベースサーバに更新処理をかけるので、テキストフィールドの値を書き直すと結果がサーバに反映する
- 更新前に値をいちど取り出し違いがあるかどうかをチェックする（楽観的ロック）

複雑な処理への対応


Web Page
HTML
INTER-Mediator
JavaScript
Database Adapter
JavaScript
Interactive
Processing
JavaScript
Server Side
Download
INTER-Mediator
JavaScript
Database Adapter
JavaScript
Inherited Data Access
Class
Data Access Class
PHP
Data Converter Class
PHP
Alternative Data
Converter Class
Database
Extensible Components

データコンバータークラスの例

```
class DataConverter_HTMLString
{

    function converterFromUserToDB($str)    {
        return $str;
    }

    function converterFromDBtoUser($str)    {
        return str_replace("\n", "<br/>",
            str_replace("\r", "<br/>",
                str_replace("\r\n", "<br/>",
                    str_replace(">", "&gt;",
                        str_replace("<", "&lt;",
                            str_replace("&", "&amp;", $str))))));
    }
}
```

データアクセスクラスの例

```
include_once('../INTER-Mediator/DB_FileMaker_FX.php');

class DB_WebSite_FMSFX extends DB_FileMaker_FX   {

    function getFromDB( $sourceName ){
        $returnValue = parent::getFromDB( $sourceName );
        if ( count( $returnValue ) > 1 )    {
            // Check for the language of documents
            $lang = array();
            foreach( $returnValue as $record )  {
                $lang[$record['Language']] += 1;
            }
                    :
        return $returnValue;
    }
    function setToDB($dataSourceName)  { … }
    function newToDB($dataSourceName) { … }
    function deleteFromDB($dataSourceName) { … }
}
```

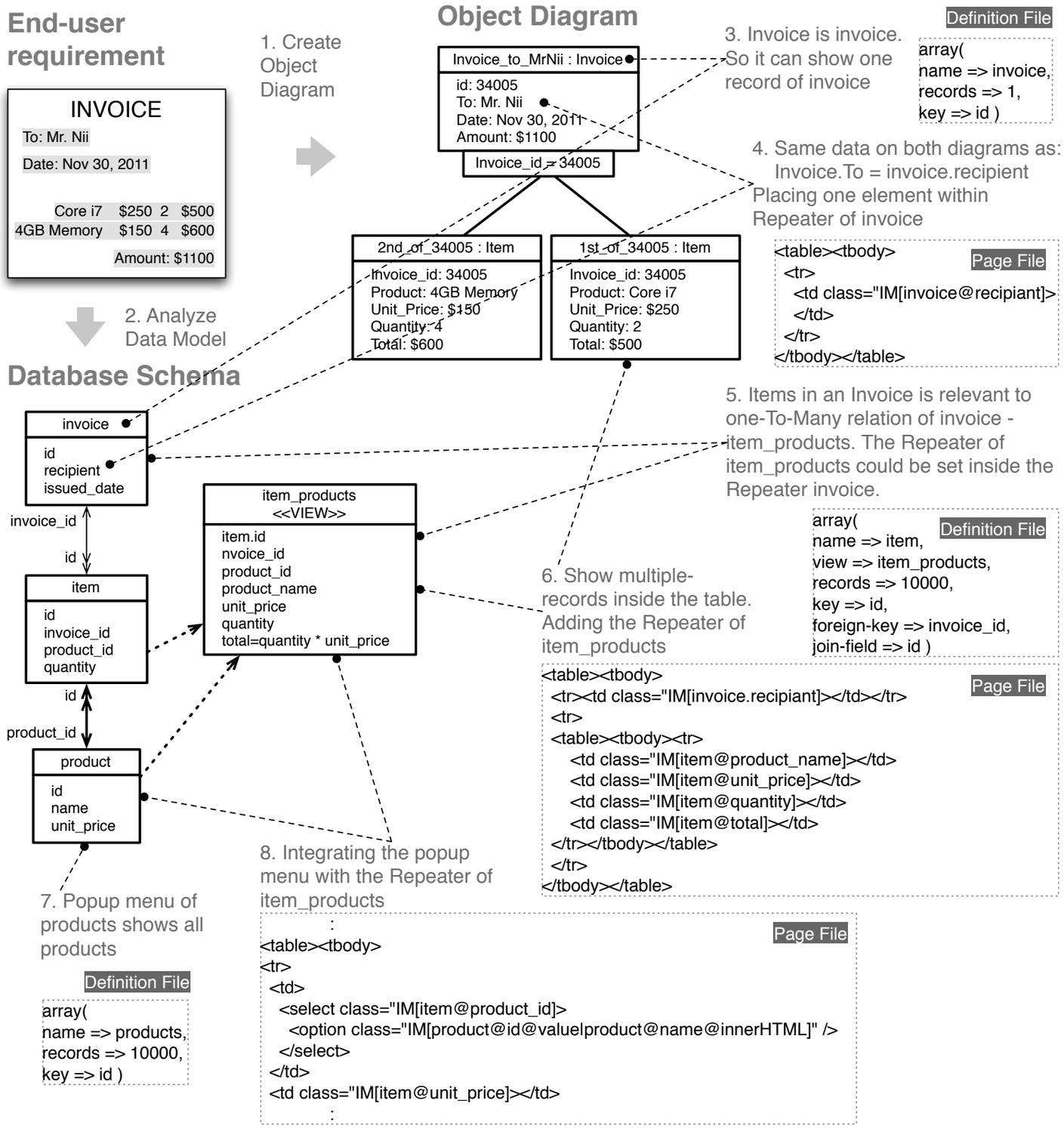
クライアント側のプログラム例

```
function modLine(target)  {
    var qtyId = INTERMediatorOnPage.getNodeIdFromIMDefinition("items@qty",target);
    var unitPriceId
        = INTERMediatorOnPage.getNodeIdFromIMDefinition("items@unitprice",target);
    var productPriceId
        = INTERMediatorOnPage.getNodeIdFromIMDefinition("product@unitprice",target);
    var amountId
        = INTERMediatorOnPage.getNodeIdFromIMDefinition("items@amount",target);
    if ( qtyId != null && unitPriceId != null && productPriceId != null && amountId != null) {
        var unitPrice = new Number( document.getElementById(unitPriceId).value > 0 ?
                                    document.getElementById(unitPriceId).value :
                                    document.getElementById(productPriceId).innerHTML );
        var amountPrice = unitPrice * document.getElementById(qtyId).value;
        document.getElementById(amountId).innerHTML
                    = INTERMediatorLib.numberFormat(amountPrice);
    }
}
```

現実的な開発作業

オブジェクト図とクラス図の突き合わせ

- ページ上に欲しい結果をオブジェクト図で記述
- データベースのスキーマをクラス図で記述
- 定義ファイルを作成し、リンクノードの指定を行う



今後の課題

認証とアクセス権をまかなう仕組みの追加

画像やファイルなどのメディアを扱う仕組みの追加

スマートな動作をするキャッシュ

HTML5のローカルデータベース対応

クラウドのデータベース対応

まとめ

クライアントサイドのテンプレート処理

- データベースとHTMLページ要素の連携を実現
- HTMLページに独自の言語要素等を追加しない
- データベースの更新にも対応
- ノードを複製による複数レコードに対応した繰り返し

開発モデルに関して

- プレゼンテーションとロジックのファイルレベルでの分離
- 基本的な処理はプログラムを書かなくても対処できる

フレームワークをオープンソースとして公開

- クライアントはHTML5対応ブラウザ
- サーバサイドはPDOないしはFileMaker Serverに対応