XMLコンソーシアムWeek　Web 2.0 Day (1)
エンタープライズシステム革新の入口はSOA2.0か KM2.0か?

REST API + XSLT
エンタープライズ・マッシュアップの実際

Amazon, hon.jp 等 公開されているAPIの活用事例を中心に

2006-05-23
XMLコンソーシアム・エバンジェリスト
日本ユニシス(株)
小林 茂

本日の内容

- REST(GET)APIを利用する
  - Amazon.co.jp
  - hon.jp
- REST + XSLT (コード紹介)
  - hon.jpでのサンプル
- マッシュアップのサンプル
  - hon.jp + Aamazon.co.jp
  - Exif to RDF(kanzaki.com) + Google Map + hon.jp + ウェザーハックス + 東証XBRLデータ試験公開

XBRLとは

REST

- REST：REpresentational State Transfer
- HTTPプロトコル：GET，POST，PUT，・・・
- ここでは，HTTP(GET)を使う
- Webブラウザのアドレス欄にURLを入れて送信するとHTMLで返ってくる

RESTのパターン

- URLを送ると，何かが返ってくる。
- HTML，JPEG，ZIP，XML，・・・
- Webブラウザは，
  - HTML：ページの表示
  - JPEG：イメージの表示
  - ZIP：保存する?
  - XML：ソース表示
    - インデント，折り畳み表示

応答がXMLならば

- あるプログラムがHTTP (GET)で要求し，XMLで応答されれば，そのXMLデータを何に利用できるのでは！
- その手段は，
  - javascript (ECMAScript)
    - XMLHttpRequest オブジェクト，Ajax　W3Cでも検討中
  - XSLT (XMLを異なる形式に変換する)
    - document()関数　詳細は後で

AmazonのREST API

- まずはこの要求を　Amazon呼び出し
- この要求 URL を解剖すると

http://webservices.amazon.co.jp/onca/xml?
Service=*AWSECommerceService* (サービス名)
&AWSAccessKeyId=アクセスキーを指定　アクセスキーを予め取得しておくこと
&Version=バージョンの指定
&ResponseGroup=応答の内容を限定
&Operation=操作の種類
&'''=操作ごとに異なるパラメタ指定

パラメタ = その値

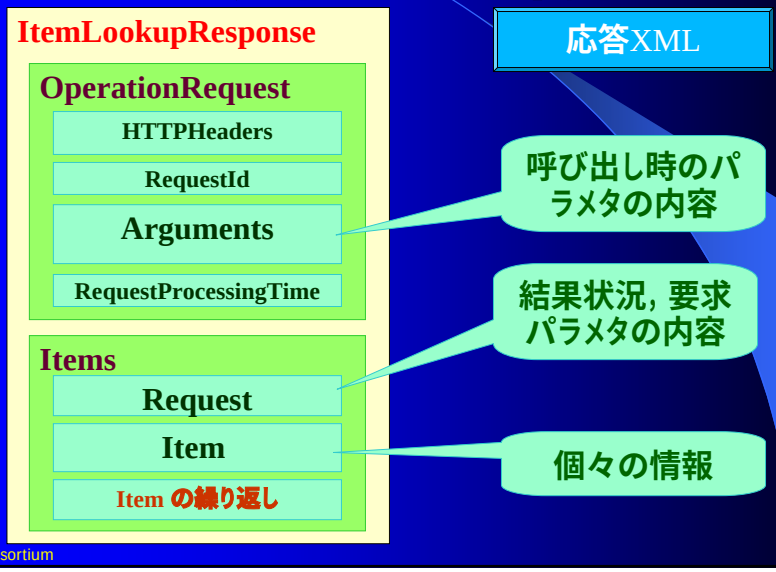
パラメタの内容(Amazon)

- この要求 URL の内容は

&Version=*2006-03-08*	バージョン名がXML名前空間に設定される
&ResponseGroup=*Small*	Medium, Large, もっと限定したもの有り
&Operation=*ItemLookup*	特定商品の情報獲得 他に, ItemSearch, など有り
&IdType=*ASIN*	商品の型。この場合ASIN。アマゾン固有の型。
&ItemId=*4798010545*	商品のID。この場合ISBN。書籍の場合には, ASIN＝ISBN。

パラメタとその値

応答XMLの内容



hon.jp のREST API

- hon.jp：電子書籍ポータル
  - 株式会社hon.jp(インプレスグループ)
- REST APIの例

hon.jp

API呼び出し

http://hon.jp/srch/k/ebk_rest.php?
uid=ユーザ識別子
&keyword=キーワードの文字列
&max=表示最大件数
&page=ページ番号(複数ページにわたる場合)

アクセスキーを予め取得しておくこと

パラメタの内容 (hon.jp)

- この要求 URL の内容は

&keyword=*web*
キーワードの文字列。日本語を含む場合，URLエンコードが必要

&max=*5*
結果は，5件以内で表示

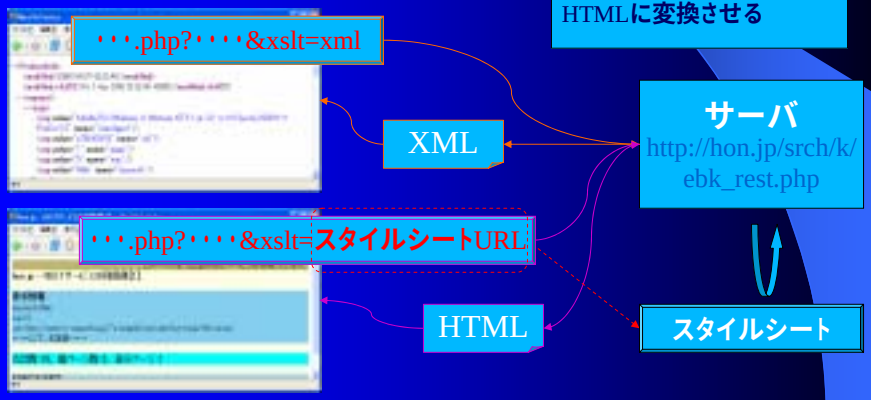
&page=*1*
最初のページを要求

&xslt=*xml*
スタイルシート(XSLT)を指定できる。"xml"を指定するかこのパラメタが無い場合，XMLで応答される

XSLTパラメタの指定

- &xslt=スタイルシートのURL

hon.jpサーバからスタイルシートを読み込んで，結果のXMLにそのXSLTでHTMLに変換させる



REST (HTTP GET)の呼び出し

- パラメタを付加したURLを生成し呼び出す
  - HTMLのformタグを使う
  - HTMLのaタグを使う
  - Javascript，ECMAScriptからXMLHttpRequestを使う
  - 日本語を含む場合，URLエンコードが必要
  - XSLTのdocument()関数を使う
    - XSLT 1.0ではURLエンコードにする機能が無い

XSLT document()関数の使用

Document()は, XSLTで規定する関数

hon.jpのAPIを呼び出すURLを変数として定義する

```
<xsl:variable name="honJpUrl">
  <xsl:text>http://hon.jp/srch/k/ebk_rest.php</xsl:text>
  <xsl:text>?</xsl:text>
  <xsl:text>&amp;</xsl:text>
  <xsl:text>uid=aTI9U00015</xsl:text>
  <xsl:text>&amp;</xsl:text>
  <xsl:text>keyword=</xsl:text>
  <xsl:value-of select="$keywords"/>
</xsl:variable>
```

$keywords = "web" とする

http://hon.jp/srch/k/ebk_rest.php?&uid=aTI9U00015&keyword=web

```
<xsl:variable   name="honJpResponse"
                select="document($honJpUrl)"/>
```

この変数に応答XMLのドキュメントノードが設定される

hon.jp RESTサービスの呼び出し

この画面の呼び出し

http://www1.u-netsurf.ne.jp/~s-koba84/xml/honJp/dispHonStyle.xml

要求・応答のパターン

dispHonStyle.xml

XML + XSLT
→XHTML
formにてhon.jpを
アクセスする

スタイル
付け要求
なし

REST
要求

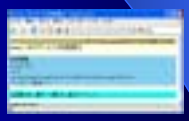
スタイル
付け要求
を付加

hon.jp
の
サーバ

応答

応答は
XML

応答



応答は
HTML

応答

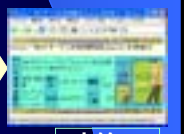
応答HTML
その中に
formあり

REST
要求

スタイル
付け要求
を付加

Amazon
の
サーバ

応答



応答は
HTML

hon.jp 応答XMLの確認

```
<ProductInfo>
  <request>
    <args>
      <arg  value="1"   name="page"/>
      <arg  value="5"   name="max"/>
      <arg  value="Web"   name="keyword"/>
    </args>
  </request>
  <totalResults>28</totalResults>
  <totalPages>6</totalPages>
  <results>
    ・・・・・ 以下，結果情報 ・・・・・
```

名前空間の指定は無い

keywordの値を得るXPath

XPath表現

/ProductInfo/request/args/arg[@name=' keyword']/@value

結果の総数を得るXPath

/ProductInfo/totalResults

XPath表現

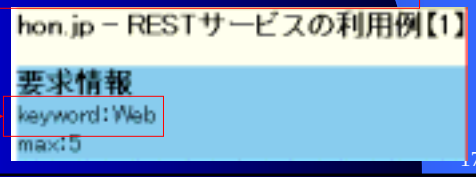
一部省略

スタイルシート (1)

```
<xsl:template match="ProductInfo">
  ・・・・・中略・・・・・
   <xsl:call-template name="request"/>
  ・・・・・中略・・・・・
</xsl:template>

<xsl:template   name="request">
   <div   class="request">
      <div   class="title">要求情報</div>
         <div>
            <xsl:text>keyword：</xsl:text>
            <xsl:apply-templates
               select="request/args/arg[@name='keyword']/@value"/>
         </div>
    ・・・・・中略・・・・・
</xsl:template>
```

応答XMLからキーワードの値を取り出し，見出しに「keyword:」と付加して表示する部分



スタイルシート (2)

```
<xsl:template name="dispMsgStyle">
   <div class="xmlReq">
      <a>
         <xsl:attribute name="href">
            <xsl:call-template name="genRestUrl4RequestPageNo">
               <xsl:with-param name="pageNo" select="$thisPageNo"/>
               <xsl:with-param name="xslt">xml</xsl:with-param>
            </xsl:call-template>
         </xsl:attribute>
         <xsl:text>応答メッセージ(XML)</xsl:text>
      </a>
      <xsl:if test="$xsltUrl">
         <a href="{$xsltUrl}">スタイルシート(XSLT)</a>
      </xsl:if>
   </div>
</xsl:template>
```

要求されたときのパラメタを参照して，XML要求を呼び出すURLを生成する。現在のページとスタイルをwith-paramで渡す

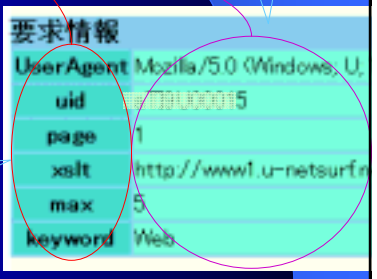
XMLとスタイルシートにリンクを張る

スタイルシート (3)

```
<xsl:template match="request">
  <table>
    <xsl:for-each select="args/arg">
      <tr>
        <th>
          <xsl:value-of select="@name" />
        </th>
        <td>
          <xsl:value-of select="@value" />
        </td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

要求したパラメタ値を表示

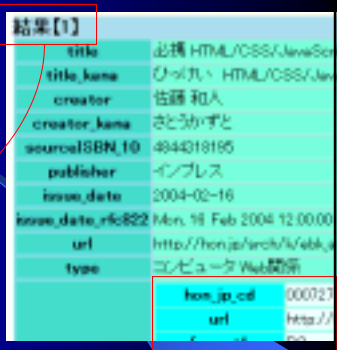
要素名をそのまま表示



スタイルシート (4)

```
<xsl:template match="results">
  <div class="title">
    結果【<xsl:value-of select="position()"/>】
  </div>
  <table class="hon">
    <xsl:apply-templates select="*[local-name()!='details']"/>
    <xsl:for-each select="details">
      <tr>
        <th>詳細【<xsl:value-of select="position()"/>】</th>
        <td>
          <div>
            <table>
              <xsl:apply-templates/>
            </table>
          </div>
        </td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>
```

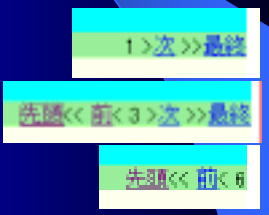
```
<xsl:template match="results/* | details/*">
  <tr>
    <th>
      <xsl:value-of select="local-name()"/>
    </th>
    <td>
      <xsl:value-of select="."/>
    </td>
  </tr>
</xsl:template>
```



スタイルシート (5)

- HTTP (GET)は，ステートレス。前後のページへリンクするには，前，後ページ用のURLを生成する必要がある
- 応答XML中にある要求パラメタ情報が役立つ。現在のページ，最後のページの情報も。

```
<request>
  <args>
  <arg  value="・・・"  name="UserAgent"/>
  <arg  value="・・・"  name="uid"/>
  <arg  value="1"  name="page"/>
  <arg  value="xml"  name="xslt"/>
  <arg  value="5"  name="max"/>
  <arg  value="Web"  name="keyword"/>
  </args>
</request>
<totalResults>28</totalResults>
<totalPages>6</totalPages>
```

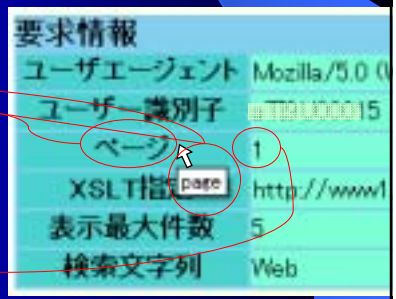


genRestUrl4RequestPageNo
が役立つ

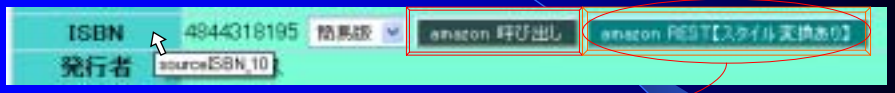
スタイルシート (6)

```
<xsl:for-each select="args/arg">
  <xsl:variable name="requestItemLabel">
    <xsl:choose>
      <xsl:when  test="@name='page'">ページ</xsl:when>
      <xsl:when  test="@name='keyword'">検索文字列</xsl:when>
      ・・・・・・・
      <xsl:otherwise>
        <xsl:value-of select="@name"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:variable>
  <tr>
    <th  title="{@name}">
      <xsl:value-of  select="$requestItemLabel"/>
    </th>
    <td>
      <xsl:value-of  select="@value"/>
    </td>
  </tr>
</xsl:for-each>
```

変数の定義
(日本語を値とする)

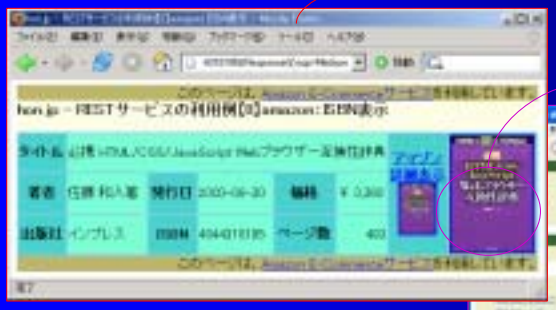


Amazonとマッシュアップ



REST呼出し

サンプルメイン
呼び出し(007)

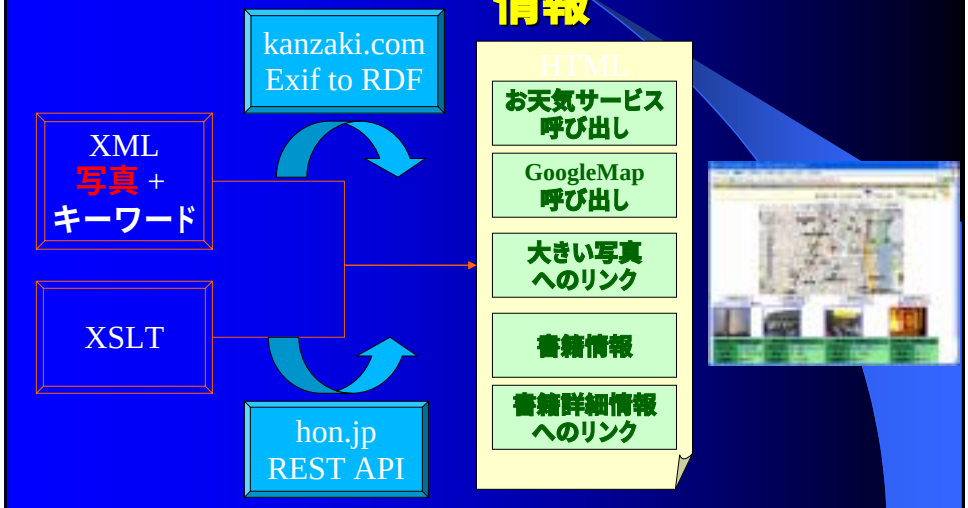


REST呼出し

スタイル変更

サンプルメイン
呼び出し(008)

さらにマッシュアップの例
写真+RDF+地図+書籍情報+天気情報



関連付けXMLデータ

```
<photoMap xmlns="urn:myPhoto">
  ・・・・中略・・・・
 <photo  href="http://・・・/expo2005/SN310580.JPG">
  <title>ビュッフェながくて</title>
  <description>ビュッフェながくて</description>
   <keywords>
    <keyword>長久手</keyword>
   </keywords>
  </photo>
</photoMap>
```

Exif to RDF

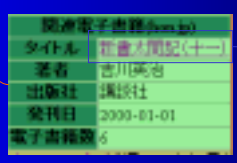
RDF

写真のタイトル

地図上に表示

位置情報
を参照

hon.jpで検索



Pivot Metadata

ライブドアのお天気サービス
ウェザーハックス

```
<xsl:variable name="todayWeatherUrl">
  <xsl:call-template name="genWeatherUrl">
    <xsl:with-param name="city">63</xsl:with-param>
    <xsl:with-param name="day">today</xsl:with-param>
  </xsl:call-template>
</xsl:variable>
```

東京は 63,
大島は 64 等

Today
Tommorow
dayaftertomorrow

```
<xsl:template  name="genWeatherUrl">
  <xsl:param  name="city" />
  <xsl:param  name="day" />
  <xsl:text>お天気サービスのURL</xsl:text>
  <xsl:text>?city=</xsl:text>
  <xsl:value-of  select="$city"/>
  <xsl:text>&amp;amp;</xsl:text>
  <xsl:text>day=</xsl:text>
  <xsl:value-of  select="$day"/>
</xsl:template>
```

URLの組立て

```
<xsl:variable  name="todayWeather"
        select="document($todayWeatherUrl)" />
<xsl:value-of  select="$todayWeather/lwws/telop" />
<img     src="{$todayWeather/lwws/image/url}" />
```

写真のメタデータ

```
<rdf:RDF
      xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
      xmlns:dc="http://purl.org/dc/elements/1.1/"
      xmlns:foaf="http://xmlns.com/foaf/0.1/"
>
  <foaf:Image  rdf:about="http://・・・/expo2005/SN310580.JPG">
    <dc:date>2005-05-15T11:52:22</dc:date>
    <foaf:topic  rdf:parseType="Resource">
      <geo:lat>35.1762194444</geo:lat>
      <geo:long>137.085819444</geo:long>
    </foaf:topic>
  </foaf:Image>
  ・・・・・以下省略・・・・・
```

Exif形式で
メタデータを
含む写真

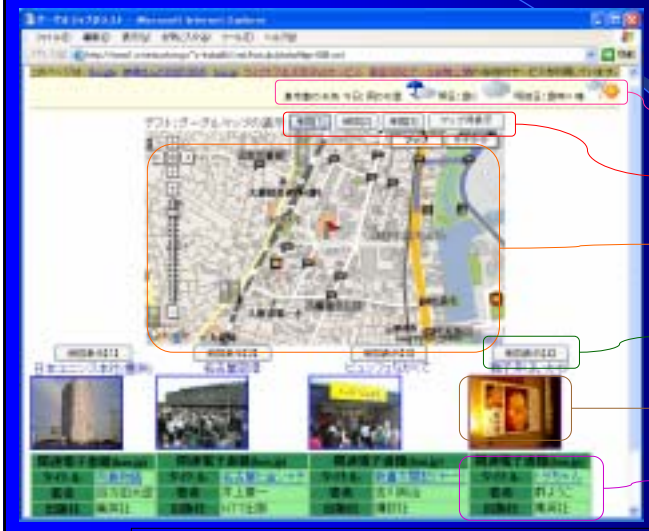
作成日付

緯度

緯度

写真のメタデータを
Exif to RDFにより取り
出し，RDFとして生成

マッシュアップページ



サンプル呼び出し

お天気情報

地図の切り替え

GoogleMapの表示

地図上の場所移動

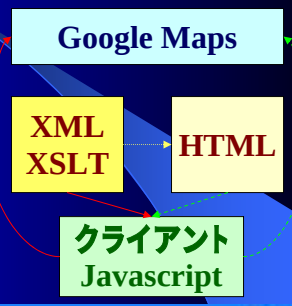
写真の表示

関連書籍の表示

http://www1.u-netsurf.ne.jp/~s-koba84/xml/honJp/photoMap-008.xml

異なるドメインへの呼び出し

- Javascriptでは，異なるドメインへの呼び出しは禁じられる
  - IE 6 では，セキュリティオプションのドメイン間でのデータソースのアクセスを有効にするとアクセス可能となる。
  - Firefox 1.5では，アクセスできない。
- 事前にXHTMLにしたものは，OK



東証：XBRLデータ試験公開

33社を対象に試験的に，本年9月30日まで公開予定

連結決算短信，直近3期分

閲覧ツールも提供
- IEプラグイン
- 複数比較用ビューア

http://www.tse.or.jp/listing/xbrl/japanese/1_xbrl_demonstration_program/11_program_summary.html

XBRLとは (eXtensible Business Report Language)

- ビジネスレポートを公開，交換する際の記述言語
  - XBRL Internationalで規定。日本では，XBRL Japan 普及・推進
  - タクソノミ
    - 財務データで用いる語彙(勘定科目)を定義
    - XML Schema を利用してタクソノミを定義
    - 構造，計算方法，ラベル名等を定義
    - XLinkの拡張リンクを利用して規定する
  - インスタンス
    - 各科目に対する値を指定。期別に値を指定できる
- 技術的特長(XML技術としてみた場合の)
  - XMLスキーマの活用(代替グループ機能)
  - XLinkで規定する拡張リンクを駆使
  - 比較的簡素なXBRLインスタンス

タクソノミの拡張が可能

リンクベース

XBRLファイルの関係

XML Schemaに従った記述

XBRLインスタンスのスキーマ
XBRLインスタンスの構造と語彙，データ型定義を規定
XBRLとして規定

XBRLタクソノミの定義
XBRLインスタンスに記述する勘定科目を定義
XBRL応用で定義

XLinkに従った記述

XBRLリンクベースの定義
データモデルの定義，表示名称の定義，表示順序の定義等
XBRL応用で定義

構造定義
計算定義
資産
＋
流動資産
＋
固定資産
名称定義

Assets | 金額型 | item型 | ID
名前の定義
データ型指定
item/tupple の指定
ID 指定

金額型 | xbrl
文字型 | item
日付型 | context
item型
要素定義
データ型の定義

XBRLインスタンス
上記のスキーマを利用したXML文書。
勘定科目，その値，コンテキストIDの指定
コンテキストのID，内容を定義

c1h | 上期

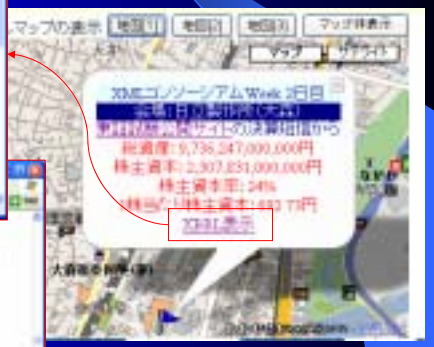
Assets | c1h | 5億
Assets | c2h | 7億
要素名
コンテキストID
値

	上期	下期
流動資産	3億	4億
固定資産	2億	3億

アプリケーションからの出力

XBRLもマッシュアップ 地図上に財務情報を表示

東証のXBRLデータ試験公開サイトからXBRLを得て財務情報を表示



吹き出しデータの設定

format-number(tse-ed-pt:TotalAssets[@contextRef='ThisYearInstant'],'#,##0')

```
<xsl:call-template name="genHtmlDivString">
  <xsl:with-param name="css"
  >color:red;</xsl:with-param>
  <xsl:with-param name="text">
    <xsl:text>総資産：</xsl:text>
    <xsl:value-of   select="  総資産の値  "/>
    <xsl:text>円</xsl:text>
  </xsl:with-param>
  <xsl:with-param name="title">
    <xsl:value-of
   select="name(tse-ed-pt:TotalAssets)"/>
  </xsl:with-param>
</xsl:call-template>
```

divタグ生成

```
<xsl:template name="genHtmlDivString">
  <xsl:param name="css"/>
  <xsl:param name="text"/>
  <xsl:param name="title"/>
    <xsl:text>&lt;div  style="</xsl:text>
    <xsl:value-of  select="$css"/>
    <xsl:text>" title="</xsl:text>
    <xsl:value-of  select="$title"/>
    <xsl:text>"&gt;</xsl:text>
    <xsl:value-of  select="$text"/>
    <xsl:text>&lt;/div&gt;</xsl:text>
</xsl:template>
```

'<', '>'の文字をエスケープする

留意事項

- クロスドメインの問題
  - クライアント処理からサーバ処理へ
- クロスブラウザの問題
  - ブラウザを選ぶ。既存のノウハウを習得し利用
- 利用するサービス継続的存続の問題
  - 魅力的なビジネスとしての契約が必要
- 異常ケース時の問題解決とその対策の問題
  - いくつかのサーバを跨って，RESTを呼び出す場合，そのうち1つでも異常な状態であると，全体的にうまく行かない
- 権利，品質，信頼の問題
  - 著作権，安定性，正しいか，信頼できるか，・・・

マッシュアップの魅力

- 公開されたサービスを容易に利用できる
- 何かをトリガーに複数のサービスを組み合わせる
- 新たな姿で，新たな価値を創出する
- それをヒントに新たなサービスを派生する
- 一人で考えるのではなく，多くの人の知恵が活かせる