XMLスキーマを利用した Webユーザインターフェイス自動生成手法の提案

榎本 俊文　　鈴木 源吾　　小林 伸幸　　星野 隆

NTT ソフトウェアイノベーションセンタ

{*enomoto.toshifumi,suzuki.gengo,kobayasi.nobuyuki,hoshino.takashi*}*@lab.ntt.co.jp*

概要　Web アプリケーションの広がりにより, システム開発における Web ユーザインターフェイス (Web UI) の開発コストの低減が課題となっている. 本論文では, XML スキーマ等の XML 技術を適用した Web UI の自動生成手法を提案する. 本手法は, UI として表示されるデータは XML であることを前提としており, XML 構造に, レイアウトパターンを対応させることにより, そのレイアウトに従った HTML に自動的に変換する. その対応付けには XML スキーマを利用しており, その型定義を有効に UI に利用できる. 単純型 (数値・文字・時刻等)・複合型に従った表示や制約チェックができ, 複数の型を組み合わせたレイアウトを容易に実現することができる. 本手法は定型的な UI を持った, 企業内の OA システム等への適用が有望であり, 実際に, XML 利用システムの開発工程におけるデータ作成・確認などのテストツールとして適用された実績がある.

キーワード　システム開発, XML, XML スキーマ, Web ユーザインターフェイス, Web データベースシステム

1　はじめに

Web データベースシステムの開発において, Web ユーザインターフェイス (Web UI) 作成にかかわるコストが一定量占めている. これは純粋に作成するコストだけではなく, システム開発中の仕様変更による修正のコストが大きい. Web UI の作成とは, 動的に HTML を生成するプログラムの作成であるが, 動的に変更する個々の値と各種ロジック (レイアウト出力や値の検証など) が個別に関係してしまうことが, 修正コストを大きくしている要因である.

そこで本論文では, 動的な値をまとめて XML 形式で構造化して取り扱い, XML の構造に合わせて HTML に自動的にレイアウトすることで Web UI を作成する手法について提案する. 同時に, データ入力において XML スキーマを活用する手法を提案する. これにより, Web データベースシステムの開発における Web UI 作成コストの削減を行う.

2　研究の背景

2.1　従来のシステム開発

Web UI には静的ページと動的ページがあるが, Web データベースシステムでは動的ページが必須となる. 動的ページの作成とは, HTML を生成するプログラムの作成であり, 表示内容に合わせてレイアウトを記述した雛形 HTML に対して, データベース (DB) から取得した値など, 動的に変更させたい各種の値を, それぞれ HTML の適切な箇所に埋め込んでいく, というロジックが実装される. また, データ入力ページの場合は, 入力フォームにユーザが入力したそれぞれの値が適切かどうか検証を行うロジックも必要となり, 作成コストも大きい.

したがって, DB で管理される動的な値と, 各種ロジック (レイアウト記述, HTML への値の埋め込み, 入力値の検証など) の間は, 非常に緊密な関係となる. このため, 設計コストはもちろん, 動的な値が変更 (追加や型変更など) されたり, ロジックが変更されたりすると, お互いに影響を受け, 修正コストが大きくなってしまう.

2.2　XML データベースの利用

XML データベース (XMLDB) は, XML データを蓄積し高速に検索することができる DB である. 我々も PostgreSQL に対し機能拡張を行うことで XMLDB 機能を実現する技術 [1][2] を提案している.

XMLDB は構造の異なる XML データを混在して管理できるといった柔軟性が特徴であるため, システムに適用することで, 仕様変更に対する修正コストの低減が期待できる. 我々はその一手法として, システム更改時の修正開発コストを低減するシステム設計方法を提案している. [3] ただし, これは Web データベースシステムのサーバ側のアプリケーション開発に対する手法であり, Web UI の作成まで含まれていない.

2.3　XSLT スタイルシートによる XML の表示

XML データを Web ブラウザでレイアウト表示する方法として, XSLT スタイルシート [4] の利用がある. 対象の XML データの構造に合わせた HTML への変換ルールを記したスタイルシートを作成し, XML データ内にそのスタイルシートの URL を記すことで利用できる.

しかし, XSLT は任意の XML データに対して記述するのが困難で, 機能も限定される. 例えば, 任意の名前空

間に対応することは非常に困難である. したがって, 各 XML データの構造に合わせて個別にルールを作成することになるが, そうすると構造の変更に合わせてルールも個別修正することになり, コスト削減につながらない.

3 解決へのアプローチ

本論文では, 以下のアプローチで Web UI 作成コストの削減を実現する.

(1) レイアウトなど表示に必要なロジックと, データ処理のロジックを分離し, ロジック間では動的な値群をまとめて XML 形式で受渡しを行う.

(2) 表示においては, レイアウトをパターン化し, XML 構造から自動的に部分 HTML に変換する. この際, 木構造に沿った汎用的なルールにより実現する.

(3) データ入力ページに対し, XML スキーマを利用することで, 入力値の検証などの既存の XML 技術を活用する. XML スキーマは, 関係データベース (RDB) のスキーマと異なり, DB 以外でも利用できるという特徴を利用する.

(1) により並行開発による効率化と修正コストの低減が期待でき, (3) により入力値の検証ロジックの作成コストを削減できる. (2) については, コストと機能のトレードオフであるが, 以下の前提によりメリットが大きいと考えている.

XML 構造は多様な表現がなされる可能性があり, レイアウトもパターン化したとしても複数あるため, 変換ロジックが複雑化したり, 組合せ数の増加による個別化により, 結局その実装に多大なコストがかかってしまう, といった懸念ある. しかし,

- 通常, XML データの構造は意味的に理解し易い設計とするため, 同じ意味を表す構造は類似する.
- レイアウトも人間が理解し易くするためのものであるため, 適用したい XML データも特定の意味を持つものに限定される. 意味が同じであれば, 構造も限定されるため, 変換の組合せパターンは限られる.

といった理由から, 実用的には問題にならないと考えている. また, レイアウトをパターン化すると, 細かな調整はできなくなる. しかし, ユーザに強く印象付けることを目的とした複雑なデザインが必要なシステムなどの一部を除き, 多くのシステムでは定型的なレイアウトの組合せで構成されているため, 適用範囲は十分に広いと考えている. 典型的には, 企業内の OA システムなどへの適用には有用である.

4 提案方式と実装

本論文で提案する方式のシステム構成を図 1 に示す.

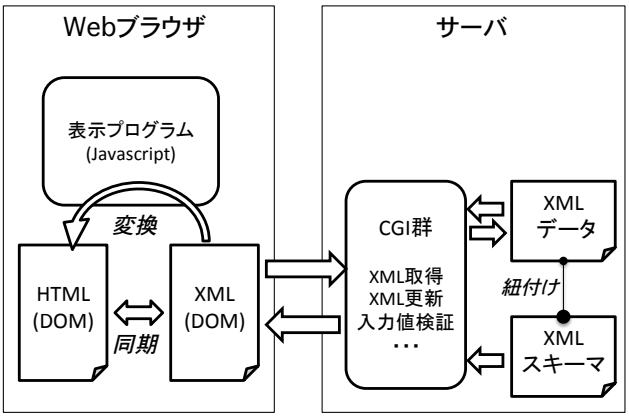

図 1 システム構成

表示ロジックは Web ブラウザ側の Javascript プログラムで実装する. 後述する複数のレイアウトパターンへの変換機能と, サーバとの XML データの送受信の機能を持つ. また, 表示している HTML DOM だけでなく, XML データの DOM を保持し, 2 つのデータを同期させる. こうすることで, データ入力ページなどデータ更新・入力値検証が必要な場合に, 保持している XML データをそのままサーバに送信でき, レイアウトパターンごとに HTML から XML への逆変換ロジックを実装することを不要にしている. さらに, 今後の機能拡張やカスタム処理を追加していく場合にレイアウト処理とデータ操作処理を分離して実装できること, といった長所もある.

Web ブラウザとサーバ間のデータ授受はすべて XML データで行うため, サーバ側では HTML 操作を意識する必要はなく, XML データの処理に集中できる. XML スキーマと紐付けた XML データの取得・更新・妥当性検証といった基本的な操作 CGI 群を用意する. ここで, XML スキーマはいくつか規格があるが, 本論文では W3C XML Schema[5] を利用する.

入力値の妥当性検証は本来 Web ブラウザ側で行うことが望ましいのだが, Web ブラウザに XML スキーマを処理する機能がないため, すべてサーバ側で行っている. ただし, Web ブラウザで入力値検証ができたとしても, サーバ側での検証も Web システムの場合は必須であるため, 不要にはならない.

4.1 XML 構造とレイアウトの対応

実装を行ったレイアウトパターンの中から代表的なものを, その対象となる XML 構造も踏まえて, それぞれ説明する.

階層レイアウト

基本的なレイアウトパターンで, XML 構造の要素の深さにしたがってインデントしながら 1 行ずつ表示する. 要素名をタイトルとして左に, テキストノード値をその右側に表示する.

例えば, 図 2 に示す XML データを, 図 3 のようにレイアウトする.

```
<クラシック演奏会>
  <楽団>NTT 研究所交響楽団</楽団>
  <名称>第 1 回演奏会</名称>
  <日時>
    <開始日>2013-04-08</開始日>
    <終了日>2013-04-08</終了日>
    <開始時間>午前 11 時</開始時間>
  </日時>
  <場所>
    <会場>○○○市民ホール</会場>
    <住所>神奈川県○○○市○○区○○</住所>
    <アクセス>○○駅徒歩 5 分</アクセス>
  </場所>
</クラシック演奏会>
```

図 2　XML 例

図 3 階層レイアウト

階層レイアウトは, 混合内容を除く任意の XML 構造に適用可能である.

表レイアウト

リーフ要素 (子要素を持たない要素) の要素名とテキストノード値をペアにして, 表形式で表示するレイアウトパターンである. オプションとして, 行数および列数を指定することができる.

例えば, 図 2 に示す XML データを, 図 4 のようにレイアウトする.

表レイアウトも混合内容を除く任意の XML 構造に適用可能である.

楽団	NTT研究所交響楽団	名称	第1回演奏会
開始日	2013-04-08	終了日	2013-04-08
開始時間	午前11時	会場	○○○市民ホール
住所	神奈川県○○○市○○区○○	アクセス	○○駅徒歩5分

図 4 表レイアウト

繰り返し表レイアウト

繰り返し要素を持つ XML データを, タイトル行を持つ表形式で表示するレイアウトパターンである.

例えば, 図 5 に示す XML データを, 図 6 のようにレイアウトする. 「曲名」「作曲者」という 2 つの項目を持つ「演奏曲」が繰り返される構造に対し, 各項目を列にまとめてレイアウトされる.

```
<演奏内容>
  <演奏曲>
    <曲名>トッカータとフーガ</曲名>
    <作曲者>バッハ</作曲者>
  </演奏曲>
  <演奏曲>
   <曲名>ピアノソナタ第 24 番 テレーゼ</曲名>
   <作曲者>ベートーヴェン</作曲者>
  </演奏曲>
  <演奏曲>
    <曲名>クラリネット・ソナタ</曲名>
    <作曲者>サン＝サーンス</作曲者>
  </演奏曲>
</演奏内容>
```

図 5 繰り返し要素のある XML 例

演奏曲	
曲名	**作曲者**
トッカータとフーガ	バッハ
ピアノソナタ第24番 テレーゼ	ベートーヴェン
クラリネット・ソナタ	サン＝サーンス

図 6 繰り返し表レイアウト

円グラフレイアウト

数値データを持つ XML データを, その階層に合わせて円グラフ表示するレイアウトパターンである.

```
<コンサート集計>
  <年>2012</年>
  <集計>
    <クラシック>
      <オーケストラ>15</オーケストラ>
      <器楽_室内楽>9</器楽_室内楽>
      <オペラ>6</オペラ>
      <声楽_合唱>7</声楽_合唱>
      <吹奏楽>10</吹奏楽>
      <その他>5</その他>
    </クラシック>
    <邦楽>
      <ポピュラー>24</ポピュラー>
      <ロック>8</ロック>
      <その他>6</その他>
    </邦楽>
    <洋楽>
      <ポピュラー>12</ポピュラー>
      <ロック>7</ロック>
      <その他>3</その他>
    </洋楽>
  </集計>
</コンサート集計>
```

図 7 数値データを持つ XML 例

例えば, 図 7 に示す XML データを, 図 8 のようにレイアウトする.

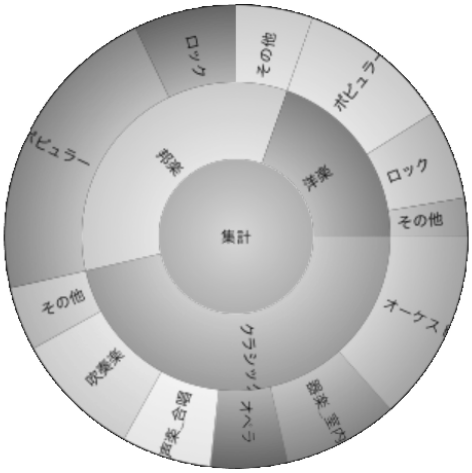

図 8 円グラフレイアウト

数値データはグラフ化して表示したいケースは多く, 他のグラフレイアウトも有用であろう.

レイアウトの複合

複数のレイアウトパターンを混ぜることも可能である.

```
<クラシック演奏会>
  <楽団>NTT 研究所交響楽団</楽団>
  <名称>第 1 回演奏会</名称>
  <日時>
    <開始日>2013-04-08</開始日>
    <終了日>2013-04-08</終了日>
    <開始時間>午前 11 時</開始時間>
  </日時>
  <場所>
    <会場>○○○市民ホール</会場>
    <住所>神奈川県○○○市○○区○○</住所>
    <アクセス>○○駅徒歩 5 分</アクセス>
  </場所>
  <指揮者>星野隆</指揮者>
  <ピアノ>小林伸幸</ピアノ>
  <クラリネット>鈴木源吾</クラリネット>
  <演奏内容>
    <演奏曲>
      <曲名>トッカータとフーガ</曲名>
      <作曲者>バッハ</作曲者>
    </演奏曲>
    <演奏曲>
     <曲名>ピアノソナタ第 24 番 テレーゼ</曲名>
     <作曲者>ベートーヴェン</作曲者>
    </演奏曲>
    <演奏曲>
      <曲名>クラリネット・ソナタ</曲名>
      <作曲者>サン＝サーンス</作曲者>
    </演奏曲>
  </演奏内容>
</クラシック演奏会>
```

図 9 レイアウト複合向け XML 例

例えば, 図 9 の XML に対し, 階層レイアウトをベースに表レイアウトと繰り返し表レイアウトを部分的に組合せた例を, 図 10 に示す.

図 10 レイアウトの複合

図 9 の「日時」「場所」を部分的に表レイアウトし,「入場料」「演奏曲」を部分的に繰り返し表レイアウトにしている.

4.2 データ入力値の検証と入力支援

データ入力ページにおいても, 前述のレイアウトを活用する. 加えて, テキストノード値の表示部分をマウスクリックすることで, 入力フォームに動的に変更する機能を実現する.

さらに, XML スキーマの定義を活用することで, 入力支援 UI の提示と入力値の検証を実現する.

XML スキーマによる入力支援

XML スキーマに定義されている入力値のデータ型を利用することで, 適切な入力支援 UI を自動的に表示することができる.

例えば, 図 11 のように数値型 (“xs:integer” [1]) で定義されている場合は, 図 12 のように数値入力用の UI を提示する.

```
<xs:element name="料金" type="xs:integer"/>
```

図 11 XML Schema でのデータ型定義例

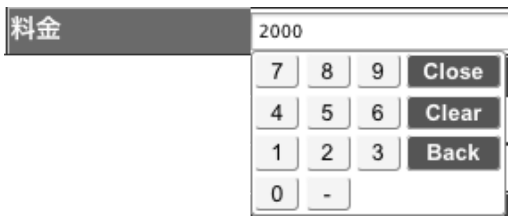

図 12 数値入力

[1] “xs:” はスキーマの名前空間を示している.

同様に, 日付型の場合は図 13 のような UI を表示する.

図 13 カレンダー入力

また, 図 14 に示すよ列挙型 (enumeration 制約) の場合は, 選択候補を XML スキーマ定義から取得し, 図 15 に示すようなプルダウンメニューを表示する.

```
<xs:element name="店舗">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="品川店" />
      <xs:enumeration value="大阪店" />
      <xs:enumeration value="福岡店" />
      <xs:enumeration value="札幌店" />
      <xs:enumeration value="名古屋店"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

図 14 列挙型の定義例

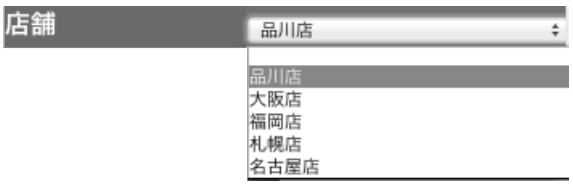

図 15　選択入力

XML スキーマによる妥当性検証

XML データが XML スキーマに対して妥当かどうか検証するモジュールは, Java 言語に含まれているものを代表に, 複数公開されている. したがって, 入力値群を全体的に検証することはそれらモジュールを利用することで実現できる.

それに加えて本論文では, 入力値ごとに個別に検証する機能を実装した. これにより, 図 11 の整数型と定義された項目「料金」に対し, 適切でない値を入力した場合には, 図 16 に示すように誤り警告を入力完了後すぐに表示することを実現できる. この例では, 背景色を反転させて警告を発している.

図 16 妥当でない値を入力した場合の警告表示

4.3　ページ作成方法

ページ作成には, HTML 内に独自の表記で記述する方式を採用した. 基本的なページの場合, 以下の情報を定められた表記にしたがって記すだけでよい.

- XML データの取得元 (URL)
- 対応する XML スキーマの ID
- レイアウトパターンとそのオプション (必要な場合)

図 17 に記述例を示す.

```
1  <html>
2  <head>
3    <script type="text/javascript"
4            src="xui.js"/></script>
5    <link rel="stylesheet" type="text/css"
6          href="xui.css">
7  </head>
8  <body>
9    <xui>
10     <xui-load id="xml1" url="get.cgi"
11      urlParam="docid=101&amp;schema=s1;"/>
12     <xui-layout dto="xml1" layout="hierarchy"/>
13   </xui>
14 </body>
15 </html>
```

図 17 ページ記述例

3〜6 行目は定型的な記述で, 表示プログラムの指定と, デフォルト定義されたスタイルシートの指定である. 指定された表示プログラムは, HTML ロード完了後に, 9〜13 行目に記された独自拡張の記述 (“xui” タグ) を発見・パースして, 適切な処理を実行する.

10〜11 行目の “xui-load” タグが XML データの取得 CGI と, 対応するスキーマ ID を “url” および “urlParam” 属性で記している. また, “id” 属性で, 取得した XML データの内部 ID を指定する.

12 行目の “xui-layout” タグがレイアウトパターンの指定であり, 実際に変換された部分 HTML が展開される場所でもある. “id” 属性で, 変換を行う XML データを指定している. この例では階層レイアウト (hierarchy) を指定しているが, 必要に応じて他のレイアウトパターンを指定する.

データ入力ページの場合は, 12 行目を

```
<xui-layout dto="xml1" layout="hierarchy"
            mode="edit"/>
<xui-option name="update" url="update.cgi"/>
```

のように, “xui-layout” タグ内に編集可能モード指定と, “xui-option” タグにより更新 CGI をオプション指定する.

また, 複合レイアウトの場合は, 12 行目を

```
<xui-layout dto="xml1" layout="hierarchy">
  <xui-innerLayout layout="table"
                   path="./日時"/>
</xui-layout>
```

のように, “xui-innerLayout” タグを内部に加え, 適用範囲を示す XPath(ルート要素からの相対パス) と, 部分的に変更するレイアウト指定を記述する.

4.4 カスタマイズ

以下のようなカスタマイズも可能である.

通常の HTML 記述との複合

図 17 の 9〜13 行目の “xui” タグ以外の箇所は, 通常の HTML 記述を行うことができる.

したがって, 静的な表示内容は従来通りに HTML・CSS 記述し, 動的表示したい任意の場所に “xui” タグを埋め込むことで, 全体のレイアウト・デザインは自由にカスタマイズできる.

複数の XML データの埋め込み

HTML 内に “xui” タグを複数記述することで, 複数の XML データを同一ページの異なる場所に表示することができる.

例えば, オンラインショッピングのページでは, 選択した商品の情報を中心に表示すると同時に, ユーザの情報を端に表示されている場合があるが, そのようなページを作成することができる.

XML 構造に対するスタイル指定

CSS はレイアウトに対するスタイル指定であるが, XML 構造に対してスタイルを指定できる.

例えば, 図 10 の表示で「会場」を強調表示したい場合, レイアウトされた HTML の要素を指定するのではなく, 図 9 の XML 構造を指定することができる. そうした場合, レイアウトを変更しても CSS を変更することなく, 強調表示が維持できる.

5 サンプルシステム作成による評価

提案手法を用いて, 簡単なサンプルシステムの構築を行った. コンサート情報の XML データ (図 2) の検索システムであり, 以下の Web UI を作成した.

- サービス機能
  - 検索条件入力 (静的ページ)
  - 検索結果一覧
  - 結果詳細表示
- 管理機能
  - データ登録
  - 検索条件入力 (静的ページ)
  - データ更新

静的なページを除くと, 4 ページに適用したが, 全体的にはわずかな機能追加のみで作成できた.

「結果詳細表示」「データ登録」「データ更新」は問題なく自動作成でき, 編集可能モードの指定の違いを除くと同じ記述で実現できた. 一方, 「検索結果一覧」は少し機能追加が必要となった. 図 18 に示すように, 表レイアウトの適用することで, 基本的には問題はなかったが, 「結果詳細表示」への遷移するハイパーリンク (右側ボタン) と, 検索結果数が多くなった場合のページャ(下側ボタン) をアプリケーション機能として追加した.

検索結果

楽団	NTT研究所交響楽団	名称	第1回演奏会	
開始日	2013-04-08	終了日	2013-04-08	
開始時間	午前11時	会場	○○○市民ホール	詳細
楽団	△△△フィルハーモニー	名称	△△△市民まつり	
開始日	2013-04-22	終了日	2013-04-22	
開始時間	午後17時30分	会場	△△△アリーナ	詳細
楽団	○○管弦楽団	名称	第13回定期演奏会	
開始日	2013-04-10	終了日	2013-04-10	
開始時間	午後6時30分	会場	○○市民ホール	詳細

前へ 次へ

図 18 検索結果一覧ページ

6 おわりに

本論文では, Web UI 開発コストの低減を目的とした, XML 技術を利用した Web UI 自動生成手法を提案した. 本手法は, XML スキーマの型をレイアウトパターンと対応づけることにより, 型に対応した表示や制約チェックが再利用される特徴を持ち, 簡単なサンプルシステムへの適用により, その実用性を確認することができた. また, 実際に XML データをやりとりするシステムの開発工程におけるデータ作成・確認などのテストツールとして適用された実績もある.

今後の課題としては, レイアウトパターンの充実, 画面遷移への対応, XML スキーマの利用を容易にするための支援ツール等があり, それらの課題を解決し, 機能拡張を行いつつ, 実際の適用を想定した評価を行い, 適用領域を拡大したいと考えている.

参考文献

[1] 兵藤正樹, 江田毅晴, 榎本俊文, 山室雅司: pgBoscage : PostgreSQL を用いた XMLDB の実装, 電子情報通信学会総合大会 (2008)

[2] 榎本俊文, 鈴木源吾, 小林伸幸, 山室雅司: PostgreSQL 向け高速 XML データ型の提案, 情報処理学会 研究報告 マルチメディア通信と分散処理 (DPS), 2012-DPS-150, Vol.23, 2012

[3] 榎本俊文, 鈴木源吾, 小林伸幸, 山室雅司: DB スキーマ変更を吸収する XMLDB 向け DAO 設計パターンの提案, 電子情報通信学会技術研究報告, ソフトウェアサイエンス 111(268), pp.13-18, 2011

[4] http://www.w3.org/TR/xslt20/

[5] http://www.w3.org/XML/Schema