

HTML タグ属性の追加による Web アプリケーションの Web サービス変換

惣宇利 卓寛† 紫合 治‡

本論文では、既存の Web アプリケーションに HTML タグの属性を付加することによって、求める Web サービスを簡単に構築する方式について提案する。この方式は、従来の Web ラッパー方式に比べて、Web アプリケーション自身に手を加えることが必要という問題はあるが、頻繁に行われる Web ページの改訂に対しても確実に誤りなくデータ抽出できるため、より実用性が高いといえる。また、複数の Web ページにまたがるデータ抽出も容易にできる。この方式による Web サービス変換システムを開発し、既存の Web サイトから Web サービスが容易に構築できることを確認した。

Converting Web Application to Web Service by HTML Tag Attributes

Takahiro Sori† and Osamu Shigo‡

This paper proposes a new method and a system to obtain required Web services by converting existing Web applications into Web services using new HTML tag attributes. This method extracts requested data from Web pages correctly and precisely when the original Web application is modified, while existing Web page wrapping method frequently requires the rule changes. Also the proposed method can treat multi-paged data extraction effectively. We have developed the Web service conversion system based on the proposed method, and recognized that the method successfully converts existing Web applications into required Web services.

1. はじめに

近年、SOA の発展に沿って Web サービスの研究、実用化が盛んであり、Google Web APIs[1]や Amazon Web Services[2]等が公開され、広く利用されている。また、従来の Web アプリケーションの機能強化をする場合、より柔軟な SOA を利用した構造に置き換えることも多い。このため、Web サービスの効率的な開発が要請されるが、Web サービスの開発には WSDL や UDDI 等の必要となる知識も多く、その導入には多くのコストがかかることが問題になる。

Web サービスを簡単に作る方法として、既存の Web アプリケーションを利用し、Web アプリケーションから

Web サービスに変換する方式が提案されている[3][4]。変換のための技術として、HTML から必要なデータを抽出する Web ページラッピングがある[5][6][7]。これらは、HTML の構造とそこから抽出するデータを示すルールを定め、それをもとに HTML を分析して必要なデータを取り出す。これは、Web アプリケーション自身に何等手を加えることなくデータを取り出せる方法であるが、HTML が修正されるたびに、ルールもそれに応じて修正する必要が生じる。有用な Web アプリケーションほど、頻繁に Web ページが変更されたり、データが加えられたりすることを考えると、その都度ルールの修正を検討する必要があることは問題である。

この問題に対処するために、我々は HTML のタグにデータ抽出のための属性を追加し、Web サービス変換では、この属性を見ながらデータ抽出処理を行う方式を提案する。新たな属性は、通常のブラウザは無視するため、この修正はもとの Web アプリケーション自身には何等影響しない。この属性は、データ抽出だけで

†東京電機大学大学院 情報環境学研究科 情報環境工学専攻

Information Environment Engineering, Graduate school of Tokyo Denki University

‡東京電機大学 情報環境学部

School of information Environment, Tokyo Denki University

なく、HTML のフォーム指定によるパラメータ付のページ遷移にも利用することで、Web ページのネストや複数ページに渡るデータの抽出も可能になる。

属性方式は、Web ラッパー方式では必要のなかった Web アプリケーション側の修正が必要になる。しかし、修正を HTML のタグ属性追加に限定しているため、Web アプリケーションの構成(アーキテクチャ、プログラミング言語、フレームワーク、データベース等)に関わりなく、汎用的に適用することができる。Web アプリケーションの提供者が、Web サービスを追加提供する場合にはこの方式が便利に使える。

本論文では、このような属性追加による Web サービス変換の方式の提案と、それに基づいた Web サービス変換システムについて述べる。2章では、Web アプリケーションの Web サービス変換に必要なプロセスと従来方式を説明し、我々の提案方式と従来方式との違いを述べる。3章では、提案方式で導入される Web サービス定義と追加する属性値について述べ、4章では提案方式に基づき開発した Web サービス変換システムについて述べる。5章で我々の Web サイトでの実例による評価、6章でまとめと今後の課題について議論する。

2. Web サービス変換方式

2.1. 変換プロセス

Web アプリケーションから Web サービスへ変換するプロセスは、図 1 に示すように、Web サービスリクエストを Web アプリケーションリクエストに変換する部分と、結果として得られる Web アプリケーションレスポンスを Web サービスレスポンスに変換する部分からなる。

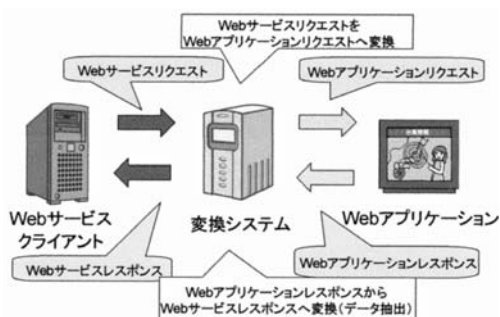


図 1. Web サービスへの変換プロセス

前者は Web サービスリクエストを SOAP プロトコルで

受取り、その内容を示す XML をもとに Web アプリケーションリクエストである HTTP クエリを出す。Web サービスの入力となる XML のデータ(サービスのパラメータ)をもとに HTTP クエリを出す処理が必要になるが、これは予め定められた HTTP クエリパターンのパラメータ部分を、Web サービスの入力データで置き換えることにより得られる。

後者の Web アプリケーションレスポンスから Web サービスレスポンスへの変換は、前者のクエリによって得られた Web ページの HTML を分析して、Web サービスの出力になるデータを抽出する処理が必要になる。通常、Web アプリケーションから出力された HTML には、要求に関わるデータ以外に、Web ページのデザインに係わる記述や不要なデータが混在しているので、そこから必要なデータを見極める技法が求められる。

2.2. Web ラッパーによるデータ抽出

Web アプリケーションの出力する HTML から求めるデータを抽出する方法として Web ページラッピングがある[5][6][7]。例えば、HTML の構造を正規表現で記述し、そこでの非終端記号名によって抽出データを規定する等の「データ抽出ルール」をラッパーに登録することによって処理される。このデータ抽出ルールは、HTML の種類ごと、Web サービスごとに記述することが必要になる。

Web ページラッピングは、データ抽出ルールはラッパーが持つので、もとの HTML の変更は必要なく、Web アプリケーションには全く手を加えなくてもよいといった利点がある。しかし、HTML の構造が変わると正しく抽出することが出来なくなることや、構造が変更される度に抽出ルールを作り直さなければならない。通常、有用な Web アプリケーションほど頻繁に改訂されることが多く、その度にルールの修正を検討しなければならないことは問題になる。特に、データ自身は変更がなくても、Web ページのデザインやデータ部分の位置といった HTML の構造の変更があると正しく抽出出来なくなり、変換ルールの作り直しが必要となる。また、Web サイトによって Web ページの構造が違う場合もあり、それぞれのページごとに抽出ルールを作る必要もある。

Web ラッパーは1つの Web ページからのデータ抽出が主であり、ネストした Web ページからなる複数の Web ページにまたがったデータの抽出は、そのままで難しい。高橋等[3]は、Web ラッパーと Web サービス合成によって、複数の Web ページからのデータ抽出を

可能にする方式を提案している。しかし、Web サービス合成は、コストがかかる問題がある。

2.3. 属性追加方式(提案方式)

属性付加方式は、データ抽出ルールをラッパーに持たせる代りに、それを HTML に属性として付加することで、属性を見ながら HTML からデータ抽出を行う。また、Web サービスの定義として、サービス名、入出力データの型等を定めておく。Web サービスのリクエストから、入力データを取り出し、最初の Web ページを取り出す。そして、Web ページの HTML を見ながら、付加属性を探し、その属性情報に沿ってデータを抽出していく。このとき、定義された Web サービスの出力となるデータのみを抽出する。また、属性の指定によっては、HTML のフォームタグを調べ、必要なパラメータを Web サービスの入力データに置換えて、フォームの処理を行って HTTP クエリを出して別の Web ページを取得し、そのページの HTML の処理を続ける。

このように、HTML タグ属性付加方式はもとの HTML を修正する必要があるため、Web アプリケーションの運用者や開発者向けの方式といえる。ただ、Web ページラッパーでのデータ抽出ルールよりも簡単で正確にデータ抽出ができ、Web ページのレイアウト変更や別の情報の追加等の変更には影響されないという利点がある。さらに、変換を HTML のタグ属性付加だけに限っているため、ブラウザによる Web アプリケーションには何の影響も与えない。また、Web アプリケーションが単なる HTML ファイルの場合から、Perl や Servlet 利用の場合、PHP による場合、JSF や EJB を利用している場合等、どんな Web アプリケーションに対しても適用することができる。

3. HTML タグ属性付加による Web サービス変換方式

3.1. 方式構成概要

HTML タグ属性付加による Web サービス変換方式は、求める Web サービスの仕様を定義した「Web サービス定義」と、もとの Web アプリケーションの HTML にルールを埋め込むための HTML タグ属性の規定からなる。Web サービス定義は、WSDL に記述すべき情報、最初の入口となる Web ページの URL、HTML 中のフォーム情報でのパラメータの指定、Web サービス仕様ドキュメント情報等からなる。

HTML タグの付加属性としては、Web サービス名、処理方式(リンクやフォームによる Web ページ取得処理、データ抽出処理等)、抽出データの型やデータ名等を指定するものがある。

Web サービス変換システムは Web サービス定義と属性が付加された HTML を見ながら、必要なデータを取り出し、結果を Web サービスリクエストに返す。

3.2. Web サービス定義

Web サービスの製作者は、Web アプリケーションからどのような Web サービスに変換するのかを定める「Web サービス定義」を作成する。このサービス定義には以下の情報が含まれる。

1. サービス名 (name)
2. サービス開始 URL(location)
3. メソッド定義(operation)
 - A) 入力パラメータ(argument)
 - ① フォーム名(form)
 - ② パラメータ名(param)
 - ③ パラメータの型(type)
 1. 固定(hidden)
 2. 可変(string, int, double, boolean)
 - B) 出力パラメータ(return)
4. データ構造の定義(dataStruct)
5. サービスの説明(document)

サービス名は Web サービスリクエストから見ればメソッド名となり、また HTML 付加属性の処理では、処理すべきタグの識別に使われる。サービス開始 URL は、Web サービス実行時、最初に取得する Web ページの URL を示す。メソッド定義は、Web サービスリクエストがサービスを呼出す時のインタフェースメソッドを示すもので、入力パラメータと出力パラメータ(戻り値)の型を定義する。ここで、入力パラメータは、フォーム名とパラメータ名とパラメータの型からなる。フォーム名は HTML の FORM タグに設定した名前に対応し、パラメータ名は HTML の INPUT や SELECT, TEXTAREA タグに現れる名前に対応し、その値を入力パラメータの値で置換える処理がなされる。入力パラメータの値は、固定の場合はサービス定義に記述した値を使用し、可変の場合は Web サービスリクエストのパラメータに置き換える。可変の場合に Web サービスリクエストとして使用できるデータ型は string, int, double, boolean の4種類とする。サービス定義で記述がされない入力パラメー

タに関してはHTMLに記述されているパラメータをそのまま使用する。出力パラメータにはメソッドの戻り値のデータ型を規定する。ここで使われるデータ型の詳細は、データ構造の定義で規定される。データ構造の定義に現れるパラメータ名は、Java クラスの場合のフィールド名を規定するとともに、HTML 中の付加属性で規定された対応するデータ名を指定する。サービスの説明には Web サービスのクライアントに対し、この Web サービス定義によって提供される Web サービスの概要について記述する。

図 2 に図書検索システムの Web アプリケーションによる本の検索機能を Web サービスに変換した場合の Web サービス定義例を示す。これは、近年出版された書籍の中からタイトルに含まれる文字列と期間から、対応する本のデータ(タイトル、内容、著者、価格)のリストを取り出す。

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinition name="SearchRecentlyBooks"
location="http://127.0.0.1:8080/convertapp/">
  <operation>
    <arguments>
      <argument type="string" param="title" form="sform1"
comment="検索するタイトル"></argument>
      <argument type="hidden" param="from" form="sform1"
comment="期間始まり">2003</argument>
      <argument type="int" param="to" form="sform1"
comment="期間終わり"></argument>
      <argument type="hidden" param="way" form="sform1"
value="title" />
    </arguments>
    <return type="Book[]" />
  </operation>
  <dataStruct>
    <member name="Book">
      <data type="string" param="title"/>
      <data type="string" param="content"/>
      <data type="int" param="price"/>
      <data type="Author" param="author"/>
    </member>
    <member name="Author">
      <data type="string" param="name"/>
      <data type="int" param="age"/>
    </member>
  </dataStruct>
  <document>このサービスは・・・</document>
</serviceDefinition>
```

図 2. サービス定義例

ここで、期間の始まりについては固定パラメータとして"2003"と指定されているので、Web サービスリクエストが指定できる入力パラメータは検索文字列と期間の終了の2つとなる。そして、戻り値の型である Book は構造化されたデータとなっており、その詳細が dataStruct

部で定義されている。さらに、Book には Author が使われており、その定義もなされる。

3.3. 付加属性値

属性値は、HTML タグに既に記述されている属性か、新たに属性を追加したものに付加していく。この方法は、Web ページの意味表現として XML 文章を付加するセマンティック Web とは違い、HTML に情報を埋め込むので、近年注目されている microformats[8]を応用した。ただし、本システムで使用する HTML 属性のキーは class とする。属性値としては Web サービス定義で定めたサービス名、メソッド定義でのパラメータ名に対応するデータ名、ページ遷移の指定がある。

これらの属性値の形式を表 1 に示す。属性値の斜字の部分は入力値に置き換わる。

表 1. 付加する属性値

名称	属性値	入力値
サービス名	ws.name. <i>NAME</i>	文字列
処理方法	ws.type. <i>TYPE</i>	link form data
フィールド名	ws.field. <i>FIELD</i>	文字列

それぞれの意味は以下の通りである。

1. サービス名: Web サービスの名前。
2. 処理方法: 抽出システムの振る舞いを指定
 - link: ハイパーリンクとしてページを遷移
 - form: Web アプリケーションへの入力
 - data: Web アプリケーションのデータ抽出
3. フィールド名: データ、フォームの名前を示す

サービス名属性が付加された要素の開始タグから終了タグまでが、そのサービスの適用範囲となる。この属性には複数のサービス名を記述することが出来るので、異なるサービスが同じ HTML の部分(開始タグから終了タグまで)を利用することができる。

処理方法は、link であれば A タグに記述され、href 属性の値である URL に遷移し、そこから属性の処理を再開する。form であれば action 属性の値や、INPUT タグ等を抽出し、HTTP クエリのデータを生成する。その際、クエリの入力パラメータの値を Web サービス定義に従って Web サービスリクエストの入力パラメータで置き換え(可変入力)、Web サービス定義での固定パラメータ値で置換える(固定入力)。link と form によるページ遷移で、遷移先の Web ページの処理が終了したあとは、もとの A タグや FORM タグの直後から処理を再開する。つまり、ページ遷移は丁度プログラムのメソッド呼

び出しと同様の制御がなされる。また data であれば、フィールド名 (*FIELD*) が Web サービス定義の出力のデータ構造の名前に現れていればその内容を出力する (データ抽出)。

最後にフィールド名 (*ws.field.FIELD*) は、処理方法で data が指定されたときに、抽出したデータを Web サービス定義の出力データにマッピングする際に用いられる。つまり、ここでの名前が、Web サービス定義のデータ構造中の名前、つまり Java クラスの場合のフィールド名に対応することになる。

3.4. 属性付加例

3.3 で提案した属性の付加例を、図書検索システムの HTML を用いて解説する。ここではサービス名を "SearchBooks" とし、図 3 に示す検索フォームの処理方法は "form" となる。実際にこれらの属性を付加した HTML の内容を図 4 に示す。FORM タグの属性にはサービス名として "SearchBooks"、処理方法として "form"、フォームの名前として "sform1" が記述されている。FORM タグにサービス名属性が付加されているので、図 4 のように括弧の中に入っている子要素である INPUT タグも同じサービスの適用範囲内となる。



図 3. Web ページでのフォーム表示

```
<form action="/servlet/SearchBooks"
method="POST" class="ws.name.SearchBooks
ws.type.form ws.field.sform1">
  タイトル: <input type="text" name="title" value=""
size="30">
  出版年: <input type="text" name="from" value=""
size="5">
  ~<input type="text" name="to" value="" size="5">
  <input type="submit" name="submit" value="検索">
</form>
```

図 4. フォームへの属性付加例



図 5. Web ページでの検索結果表示

```
...
<table border="1" width="90%" title="検索結果"
class="ws.name.SearchBooks ws.type.data
ws.field.Books">
  <tr>
    <th width="30%">タイトル</th>
    <th width="50%">内容</th>
    <th width="10%">出版年</th>
  </tr>
  <tr class="ws.type.data ws.field.Book">
    <th class="ws.type.data ws.field.Title">
      ソフトウェア・テスト PRESS Vol.1</th>
    <td class="ws.type.data ws.field.Content">
      テストエンジニアを兼任する一般のプログラマやSEを対象にソフトウェアテストの実践的なノウハウをわかりやすくコンパクトに紹介。</td>
    <td class="ws.type.data ws.field.Year">2005</td>
  </tr>
  <tr class="ws.type.data ws.field.Book">
    <th class="ws.type.data ws.field.Title">独習 Java
    第3版</th>
  </tr>
  ...
```

図 6. 検索結果への属性付加例

検索の結果、Web ブラウザでは図 5 のように表形式で表示された。属性は、図 6 に示すように Web ページのサービス名は同様に "SearchBooks" とし、処理方法はデータ抽出なので "data"、フィールド名は "Title" と "Content" として付加した。ここで付加したのはデータ抽出に係わる属性なので、任意のタグに付加することが出来る。

このように各データを表すタグの属性にフィールド名を書くことで、サービス定義とマッピングし、データ構造を構築することが出来るようになる。

4. Web サービス変換システム

4.1. システム構成

HTML タグ属性付加方式による Web サービス変換システムは、Web サービスリクエストと Web アプリケーションの間に位置し、Web サービスリクエストに対しては Web サービスプロバイダとなり、さらに Web アプリケーションに対してはクライアントの役目を果たす。ただし、現在、JavaScript には対応していない。さらに、システムは Web サービス定義機能として、サービス定義から WSDL の生成、Web サービス仕様ドキュメントの生成、Web サービス定義の出力データ構造が対応する Web アプリケーションの HTML タグ属性と正しくマッピングできるかどうかのチェック等をもつ。図 7 にシステム構成を示す。このシステムの動作環境として、JDK5.0、Apache Tomcat5.5、Axis1.3、HTML Parser1.6[9]を利用した。

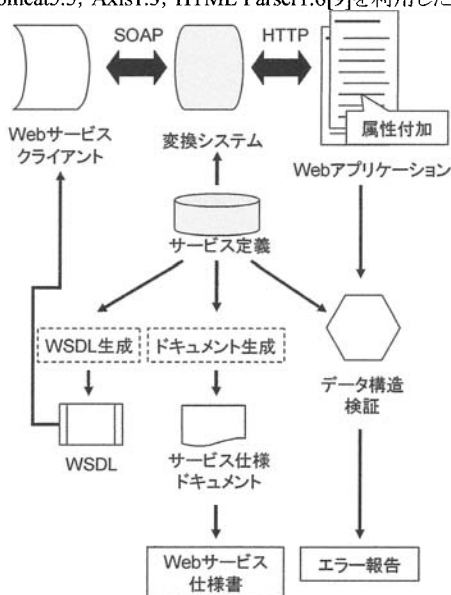


図 7. システム構成図

4.2. 変換処理部

変換処理部は、Web サービスプロバイダとして動作する。まず、Web サービスリクエストからの Web サービスリクエストを受け取ると、対応する Web サービス定義をもとにそのリクエストの内容を解析し、Web サービス定義のロケーションで指定された URL の Web ページを取得する。その Web ページを HTML Parser で解析し、もとのサービス名が指定されているタグの範囲のみを取出し、その中の属性を調べる。そして、3.3 で述べたように、属性毎にページ遷移処理やデータ出力処理を

進めていく。

Web アプリケーションから出力される HTML をシステムが処理をしていく方法は、図 8 のように、まず HTML の最初から読み取っていき、サービス名と対応する属性が見つかったら、その範囲を処理し、リンク処理があればそのリンクを辿って行く。リンク先の抽出処理が終わったら、もとのタグの次の所に戻って処理を再開する。このようにして最初の Web ページの終わりまで処理を続けた後、抽出したデータを Web サービスレスポンス形式に変換し Web サービスリクエストへ返す。

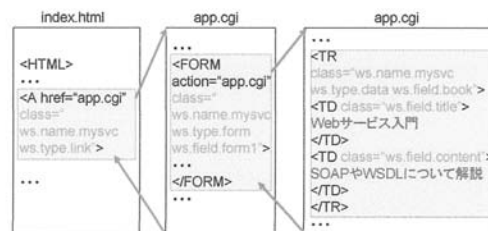


図 8. 変換システムの処理フロー

4.3. Web サービス定義機能

Web サービス定義から、WSDL と仕様ドキュメントを生成し、HTML との整合性を検証する機能をもつ。

1) WSDL 生成

サービス定義の入力パラメータや出力パラメータ、サービス名、サービス実行 URL 等から WSDL を自動生成する。入力パラメータは、システムとして必要な引数 (URL) が入るので、WSDL に記述される引数はサービス定義で記述した引数の数より多くなる。

2) Web サービス仕様ドキュメント

サービス仕様ドキュメントは、サービス定義に記述されたメソッドやメソッドのパラメータの説明から生成される。これはサービスを利用するクライアントの利便性を考慮したもので、通常のブラウザから利用できる。

3) データ構造検証

データ構造検証では、サービス定義と実際の Web アプリケーションの出す HTML の付加属性とが正しく対応しているかをチェックする。この検証は、サービス定義をもとに、対応する Web アプリケーションから HTML を順に取り出し、抽出できるデータが過不足なくサービス定義のデータ構造に整合していることを確認する。

5. 評価実験

ここでは、Perl で書かれたウィキクローンである FreeStyleWiki に本方式を適用し、評価を行った。これは Wiki の利用の仕方や編集の仕方、プラグインをユーザに説明する為の Web ページで、キーワードによって検索された項目の説明がなされるページをリストアップする検索機能等が含まれている。この実験ではキーワード検索によって該当するページをリストアップする機能を Web サービスへ変換した。

手順は、まず、Perl で書かれているこのアプリケーションの検索モジュールのソースコードを修正する。修正は以下の手順で行った。

1. ソースコードの検索フォームを出力する部分に、以下の属性値を追加する。修正したソースコードを図 9 に示す。

ws.name.SearchPages ws.type.form

```
my $buf = "<form method='GET'
action='\". $wiki->config('script_name')." \"'
class='\"ws.name.SearchPages ws.type.form
ws.field.form1\"'>\". \"キーワード
<input type='\"TEXT\"' name='\"word\"' size='\"20\"'>\";
```

図 9.修正したソースコード(フォーム)

2. 検索が実行され、結果が表示される部分には、AND 検索と OR 検索の2つがあるので、それぞれに対して以下の属性を追加する。修正したソースコードを図 10 に示す。

ws.name.SearchPages ws.type.data ws.field.Result

```
$buf .= "<li><a href='\". $wiki->config('script_name')."
\"?page='\". Util::uri_encode($name)." \"'
class='\"ws.type.data ws.field.Result\"'>\". $cgi-
>escapeHTML($name)."</a> -
\". Util::escapeHTML(&get_match_content($wiki,$name,
$page,$index)).\"</li>\"";
```

図.10 修正したソースコード(出力)

3. Web ブラウザで表示し、アプリケーションの動作と属性の付加を確認する。

次に、このサービスに対応する Web サービス定義を作る。サービス定義は図 11 のようになり、サービス名は"SearchPages", Web サービスリクエストからの入力パラメータは検索キーワードの文字列となり、ラジオボタンは OR に、チェックボックスはチェック状態に固定した。

出力パラメータは文字列の配列となる。

実際に変換システムを通して Web サービスを実行した。入力パラメータは"plugin attach"とした。この機能を Web ブラウザから行った場合は図 12 のように、キーワードを"plugin attach", ラジオボタンを OR, チェックボックスをチェックした状態になる。

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceDefinition name="SearchPages"
location="http://www.se.sie.dendai.ac.jp/wiki/">
<operation>
<arguments>
<argument type="hidden" param="l"
form="form1" comment="AND, OR検索">OR</argument>
<argument type="hidden" param="c"
form="form1" comment="内容を含めるか">true</argument>
<argument type="string" param="word"
form="form1" comment="検索文字列"></argument>
</arguments>
<return type="string[]" />
</operation>
<dataStruct></dataStruct>
<document>このサービスは、FreeStyleWikiのページリストから、指定した文字列をタイトルに含むページリストを返すサービスです。</document>
</serviceDefinition>
```

図 11.Web サービス定義

この実験の結果、Web サービスレスポンスとして図 13 のように2件の検索結果が得られた。図 14 に示す Web ブラウザで実行した画面にも2件の検索結果が表示され、どちらも同じタイトルであることから変換が正しく出来たことが確かめられた。なお、Perl のソースコードの修正からサービス定義ファイルの作成には1時間程度で出来た。



図 12.検索フォーム表示

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<soap:Body><SearchPagesResponse
xmlns="http://www.se.sie.dendai.ac.jp/page/">
<SearchPagesResult>
<Result>Footer</Result><Result>PluginHelp</Result>
</SearchPagesResult></SearchPagesResponse>
</soap:Body></soap:Envelope>
```

図 13.Web サービスレスポンス



図 14.Web ブラウザで表示した検索結果

6. おわりに

本論文では、既存の Web アプリケーションに HTML タグの属性を付加することによって、求める Web サービスを簡単に構築する方式について述べた。これによって、Web ページの頻繁な改装や修正に影響されずに、正しくデータ抽出ができた。この方式は Web アプリケーションに手を加える必要があるが、Web アプリケーションの運用者向けという制約はあるが、HTML タグの属性を追加するだけなので、Web アプリケーションのアーキテクチャは何であっても適用可能である。

今後の課題として、実際に使われている Web アプリケーションに本方式を適用して、より実用的な Web サービスの構築や、フリーで公開されている様々なプログラム言語による Web アプリケーションに本方式を適用して便利な Web サービスを構築することによって、本方式の評価を行い、システムの機能強化を進めていく所存である。

参考文献

- [1] Google Web APIs, <http://code.google.com/>
- [2] Amazon.com: Homepage: Amazon Web Services <http://aws.amazon.com/>
- [3] 高橋 健一, 紫合 治: Web アプリケーションの Web サービス変換, ソフトウェアエンジニアリング最前線 2006
- [4] Markus L. Noga, Max Volkel: From Web Pages to Web Service with wal, NCWS2003
- [5] Sahuguet, A., and azavant, F. Building intelligent web applications using lightweight wrappers. Data and Knowledge Engineering 36, 3 (2001)
- [6] Crescenzi, V., Mecca, G., and Merialdo, P. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In Proceedings of the 26th International Conference on Very Large Database Systems (Rome, Italy, 2001), pp.109-118
- [7] Liu, L., Pu, C., and Han, W. XWRAP: An XML-enable Wrapper Construction System for Web Information Sources. In Proceedings of 16th IEEE International Conference on Data Engineering (San Diego, California, 2000)
- [8] microformats, <http://microformats.org/>
- [9] HTML Parser – HTML Parser <http://htmlparser.sourceforge.net/>