**技術調査書**

Web レコードショップ

更新履歴

|  |  |  |
| --- | --- | --- |
| 日付 | 更新内容 | 作業者名 |
| 2010/11/19 | 第1版リリース | 古舘　昌伸 |

目次

[Web レコードショップ 1](#_Toc278380587)

[**RSS [1]** 3](#_Toc278380588)

[RSSとは 3](#_Toc278380589)

[RSS 2.0の標準フォーマット 3](#_Toc278380590)

[データの抽出 5](#_Toc278380591)

[**正規表現 [2][3]** 8](#_Toc278380592)

[正規表現とは 8](#_Toc278380593)

[正規表現の使用 8](#_Toc278380594)

[**メール送信 [2][4][5]** 8](#_Toc278380595)

[メールを送信する 8](#_Toc278380596)

[**ショッピングカート [6][7]** 10](#_Toc278380597)

[ショッピングカートとは 10](#_Toc278380598)

[PHPを用いたショッピングカートプログラム 10](#_Toc278380599)

[ショッピングカートプログラムの仕様 10](#_Toc278380600)

[プログラム例 10](#_Toc278380601)

[仕様 10](#_Toc278380602)

[仕様に基づいたプログラム 11](#_Toc278380603)

[カートの中身を表示 14](#_Toc278380604)

[**参考文献** 19](#_Toc278380605)

# **RSS [1]**

## RSSとは

Webサイトの見出しや要約などのメタデータを構造化して記述するXMLベースの文章フォーマット。主にウェブサイトの更新情報を公開するのに使われている。RSSで記述された文書には、Webサイトの各ページのタイトル、アドレス、見出し、要約、更新時刻などを記述することができる。RSS文書を用いることで、多数のWebサイトの更新情報を統一的な方法で効率的に把握することができる。

## RSS 2.0の標準フォーマット

RSS 標準では、フィードのコンテンツの定義と制約を設けている。フィード元はあらゆるデータ・ソースにすることが可能で、フィードがインターネット文書を定義し、本質的にはリンクとその説明のリストを構成する。

以下のフォーマットはNASAの「Liftoff News」フィードからのサンプル文書を使用している。

**リスト1. RSS2.0文書のサンプル**

|  |
| --- |
| <?xml version="1.0"?>  <rss version="2.0">  <channel>  <title>Liftoff News</title>  <link>http://liftoff.msfc.nasa.gov/</link>  <description>Liftoff to Space Exploration.</description>  <language>en-us</language>  <pubDate>Tue, 10 Jun 2003 04:00:00 GMT</pubDate>  <lastBuildDate>Tue, 10 Jun 2003 09:41:01 GMT</lastBuildDate>  <docs>http://blogs.law.harvard.edu/tech/rss</docs>  <generator>Weblog Editor 2.0</generator>  <managingEditor>editor@example.com</managingEditor>  <webMaster>webmaster@example.com</webMaster>  <item>  <title>Star City</title>  <link>http://liftoff.msfc.nasa.gov/news/2003/news-starcity.asp</link>  <description>How do Americans get ready to work with Russians aboard the  International Space Station? They take a crash course in culture, language  and protocol at Russia's Star City.</description>  <pubDate>Tue, 03 Jun 2003 09:39:21 GMT</pubDate>  <guid>http://liftoff.msfc.nasa.gov/2003/06/03.html#item573</guid>  </item>    <item>  <title>Space Exploration</title>  <link>http://liftoff.msfc.nasa.gov/</link>  <description>Sky watchers in Europe, Asia, and parts of Alaska and Canada  will experience a partial eclipse of the Sun on Saturday, May 31st.</description>  <pubDate>Fri, 30 May 2003 11:06:42 GMT</pubDate>  <guid>http://liftoff.msfc.nasa.gov/2003/05/30.html#item572</guid>  </item>    <item>  <title>The Engine That Does More</title>  <link>http://liftoff.msfc.nasa.gov/news/2003/news-VASIMR.asp</link>  <description>Before man travels to Mars, NASA hopes to design new engines  that will let us fly through the Solar System more quickly. The proposed  VASIMR engine would do that.</description>  <pubDate>Tue, 27 May 2003 08:37:32 GMT</pubDate>  <guid>http://liftoff.msfc.nasa.gov/2003/05/27.html#item571</guid>  </item>    <item>  <title>Astronauts' Dirty Laundry</title>  <link>http://liftoff.msfc.nasa.gov/news/2003/news-laundry.asp</link>  <description>Compared to earlier spacecraft, the International Space  Station has many luxuries, but laundry facilities are not one of them.  Instead, astronauts have other options.</description>  <pubDate>Tue, 20 May 2003 08:56:02 GMT</pubDate>  <guid>http://liftoff.msfc.nasa.gov/2003/05/20.html#item570</guid>  </item>  </channel>  </rss> |

上記の XML 形式文書で最初にある子オブジェクトのチャネルは、フィード自体とその関連情報でしかない。チャネル・オブジェクトには title、link および description オブジェクトが必要。これらのオブジェクトがフィードに関する基本的な記述情報を定義する。その他にオプションのオブジェクトとして、language、copyright、managingEditor、webMasterなどがある。

チャネルに含めることのできる項目の数に制限はない。また、ITEM 要素のすべての要素はオプションだが、要素を有効にするためにはタイトルまたは説明が少なくとも 1 つ必要。要素には、title、link、description、authorなどがある。

## データの抽出

データベースからデータを抽出し、そのデータを RSS フィードのフォーマットにする。この RSS フィードは、呼び出されるたびにデータセットへの最新の追加内容を検索し、新しい RSS を要求側に返すように設定する。フィードは自分のサイト上にあるあらゆるデータ・ソースから送られてくるが、結局は、RSS フィードを受け取る側がそのデータを使用するのに十分な情報があることを確認する必要がある。そのためには少なくとも、URL の名前と説明が必要である。Web アプリケーションのデータベースに接続し、更新された情報を抽出して XML RSS 文書形式に設定するには、PHP を使用する。

使用するデータベースが決まったら、通常通りに接続を作成し、ユーザーが読み取れるようにレイアウトされた XML を表示するページを生成する。ページを要求側に送信するデータを独自のコードですべて確実にフォーマット設定したら、そのデータが適切に渡されることを確実にして、読者がURLをリーダーに入力したときに、期待するXML RSSフィードを取得できるようにする。

以下、実際のPHP文書である。

**リスト 2. RSS.php**

|  |
| --- |
| <?php  $mysql = mysql\_connect("localhost", "g031h140", "g031h140!g031h140") or die("接続できませんでした。");  mysql\_select\_db("g031h140", $mysql) or die("接続できませんでした。");  $query = "select link, headline, description from `headlines` limit 15";  $result = mysql\_query($query, $mysql);  while ($line = mysql\_fetch\_assoc($result))  {  $return[] = $line;  }  $now = date("D, d M Y H:i:s T");  $output = "<?xml version=\"1.0\"?>  <rss version=\"2.0\">  <channel>  <title>Our Demo RSS</title>  <link>http://www.tracypeterson.com/RSS/RSS.php</link>  <description>A Test RSS</description>  <language>en-us</language>  <pubDate>$now</pubDate>  <lastBuildDate>$now</lastBuildDate>  <docs>http://someurl.com</docs>  <managingEditor>you@youremail.com</managingEditor>  <webMaster>you@youremail.com</webMaster>  ";    foreach ($return as $line)  {  $output .= "<item><title>".htmlentities($line['headline'])."</title>  <link>".htmlentities($line['link'])."</link>    <description>".htmlentities(strip\_tags($line['description']))."</description>  </item>";  }  $output .= "</channel></rss>";  mysql\_close($mysql);  echo $output;  ?> |

このリストではまず、データベース接続オブジェクトを設定している。このデータベース内のテーブルには、見出し、リンク、説明フィールドが含まれるレコードがあり、後にこれらの情報を XML 応答に書き込むことになる。次にデータベース接続の結果を使って、MYSQL\_QUERY() により、そのテーブルに対して SQL クエリーを実行している。そのクエリーの実行結果オブジェクトは、WHILE を使って 1 つずつ書式を整え直され、そのデータはさらに新しい単純配列に再フォーマット設定される。新しい配列が準備できたら、$output 変数の中に XML ファイルの作成を開始し、返された応答ごとに 1 つずつ $line 配列を一巡して新しい要素を追加。SQL ステートメントで応答数を 15 に制限しているため、この処理には長い時間かからない。このスクリプトが実行されると、以下のような簡潔でわかりやすい RSS ファイルが出力される。

**リスト 3. RSS.php の出力**

|  |
| --- |
| <?xml version="1.0"?>  <rss version="0.97">  <channel>  <title>Our Demo RSS</title>  <link>http://www.tracypeterson.com/RSS/RSS.php</link>  <description>A Test RSS</description>  <language>en-us</language>  <pubDate>Mon, 13 Nov 2006 22:46:06 PST</pubDate>  <lastBuildDate>Mon, 13 Nov 2006 22:46:06 PST</lastBuildDate>  <docs>http://someurl.com</docs>  <managingEditor>you@youremail.com</managingEditor>  <webMaster>you@youremail.com</webMaster>  <item rdf:about="http://www.tracypeterson.com/">  <title>This is Tracy's Web Page!</title>  <link>http://www.tracypeterson.com/</link>  <description>This is a demonstration of how to get PHP to work for  your RSS feed.</description>  </item><item rdf:about="http://www.tracypeterson.com">  <title>This is Tracy's site again!</title>  <link>http://www.tracypeterson.com</link>  <description>Again, this is a demonstration of the power of PHP  coupled with RSS.</description>  </item></channel></rss> |

# **正規表現 [2][3]**

## 正規表現とは

端的に言えば、「いくつかの文字列を一つの形式で表現するための表現方法」。正規表現は「数字のパターン」や「半角英字のパターン」などを、「メタキャラクタ」という記号を組み合わせて表現出来る。例えばWebアプリケーションなどで、入力値として数字のみを期待しているところにそれ以外の文字が入力されないようにチェックする場合、正規表現の「数字のパターン」を使ってチェックする。

## 正規表現の使用

正規表現をPHPで使用する場合、preg\_match()関数を用いる。

以下は、Webアプリケーションにてメールアドレスを記入する際に、POSTで受け取ってきた値について「xxx@xxx.xx」（xは半角英数字と「.」「-」の記号）の形以外の場合はエラー文を表示するPHPプログラム（一部分のみ）である。

**リスト 4. 正規表現の使用**

If(!preg\_match("/^[a-zA-Z0-9\_\.\-]+@[A-Za-z0-9]+\.[A-Za-z0-9\.\-]+$/", $\_POST["sub\_mail"]))

　　print "メールアドレスが不適切です。再入力してください。<br/>";

# **メール送信 [2][4][5]**

## メールを送信する

PHPにはメール送信関数としてmail関数があるが、日本語を用いると文字化けしてしまうため、通常はマルチバイト対応のmb\_send\_mail関数を利用する。実行するときには引数として宛先、件名、本文、追加ヘッダーを指定する。この関数は送信が成功するとTRUE、失敗するとFALSEを返す。

以下に実際にPHPプログラム中でメールを送信する際のソースコードと実行結果を示す。

**リスト 5. メール送信**

<?php

　mb\_language("Ja") ;

　mb\_internal\_encoding("UTF-8") ;

　$mailto="aaa@aaa.com";

　$subject="タイトル";

　$content="内容を入れます\nあああ\nいいい";

　$mailfrom="From:" .mb\_encode\_mimeheader("テスト") ."<bbb@bbb.com>";

　if (!mb\_send\_mail($mailto, $subject, $content, $mailfrom)){

print "メールの送信に失敗しました。";

exit;

　}else

print "メールを送信しました。"; }

?>

**リスト 6. 実行結果**

From: "テスト" <bbb@bbb.com>

To: <aaa@aaa.com>

Sent: Monday, November 22, 2010 9:11 PM

Subject: タイトル

内容を入れます

あああ

いいい

各行の解説は以下の通りである。

mb\_language("Ja"): 言語の設定を行う。日本語なのでJaとする。

mb\_internal\_encoding("UTF-8") : 内部文字エンコードを指定。

$mailto="aaa@aaa.com": メールの宛先アドレス。

$subject="タイトル": メールの件名。

$content="内容を入れます\nあああ\nいいい": メール本文。\nで改行。

$mailfrom="From:" .mb\_encode\_mimeheader("テスト") ."<bbb@bbb.com>": ここは追加ヘッダーの部分で、ここにメールの差出人を入れる。名前の部分が日本語の場合、mb\_encode\_mimeheaderでエンコードする必要がある。

mb\_send\_mail($mailto,$subject,$content,$mailfrom): 送信。

# **ショッピングカート [6][7]**

## ショッピングカートとは

オンラインショップが備える機能の一つで、購入するつもりの商品を一時的に保存しておく機能。あるユーザ専用の購入予定商品の一覧ページを用意し、商品を選択すると一覧に追加されていき、最後にまとめて購入手続きを行なう機能のこと。

## PHPを用いたショッピングカートプログラム

ショッピングカートプログラムではCookieやSessionを使用する。ネットショップなどを利用するユーザーは、一旦商品をカートに入れておいてから、再度買い物を続けたり、例えば送料についての説明ページを見たりと、他のページをいろいろ見てから、最後にカートへ進んで清算をする。通常このような場合、全てのページ間でユーザーの情報をパラメータとして渡す必要がある。つまり全てのページをphp形式にして、全てのリンクに?に続いてパラメータを渡すことになる。しかしこの場合次のような問題が出てくる。

・全てのページを.phpにする必要があるので、管理面でやや手間がかかる

・SEO的にはパラメータ付きのURLは、有利ではない

・パラメータの設置を間違えたり、忘れたりする箇所があると、不具合が起きる

そこでこのような問題を解決するために、CookieとSessionを使う。この2つを併用することで、ユーザーが拡張子htmlのファイルを見たり、またリンクにパラメータを設置しなくても、ユーザーが一度カートに入れた値を保持しておくことができる。

## ショッピングカートプログラムの仕様

・表示内容は、購入した商品名・商品画像・価格・数量・小計金額・合計金額、購入画面へ進むボタン。

・金額は3桁ごとのカンマ区切り表示

・カート表示画面で、数量変更・削除が可能

・購入画面へ進むボタンを押すと、送付先情報入力画面へ移動

## プログラム例

### 仕様

1.商品詳細ページから、数量を選択してカートに商品を入れる

2.一度カートに入れた商品と数量は、ブラウザを閉じるまで記憶

3.同じ商品をカートに入れた場合は、後から入れた時の数量に変更

4.カートの中身は、商品リスト・商品詳細ページで常に表示

### 仕様に基づいたプログラム

・カートに入れるボタンの設置

1の仕様に基づき、商品詳細ページにカートに入れるボタンを設置する。このボタンの<form>は以下のようになる。

**リスト 7. ボタンの<form>**

<form action="ファイル名" method="POST">

<table>

<tr>

<th>品名</th>

<td>#itemName#</td>

</tr>

<tr>

<th>価格</th>

<td>#price#円</td>

</tr>

<tr>

<th>数量</th>

<td>

<input type="hidden" name="mode" value="itemDetail" />　//①

<input type="hidden" name="mode2" value="cartIn" />　//②

<input type="hidden" name="itemID" value="#itemID#" />　//③

<input type="hidden" name="itemName" value="#itemName#" />　//④

<select id="num" name="orderNum">

<option value="1">1個</option>

<option value="2">2個</option>

<option value="3">3個</option>

<option value="4">4個</option>

<option value="5">5個</option>

</select>

</td>

</tr>

<tr>

<td colspan="2"><input type="submit" name="cartIn" value="カートに入れる" /></td>

</tr>

</table>

</form>

まずリスト中③、④でカートに入れる商品を判別している。また、①でボタンが押されてカートに入れるための一連の処理を行った後は、再度商品詳細ページを表示するようにしている。だが、これだけでは商品詳細ページのみを表示するのか、カートに入れてから商品詳細ページを表示するのかを判別できない。そのため、2つの処理の判別させるために、②がある。

カートに入れるボタンがおされた場合、mode=itemDetailの値が送信されることにより、商品詳細ページを表示するitemDetail関数が実行されるようになる。itemDetail関数内では商品詳細を表示する処理の他に、カートに入れるボタンが押された場合に、カートに商品を入れる処理を記述する。カートに商品を入れる処理は、長くなるため、cartInという関数にしてこれを呼び出すようにしている。このとき、商品ID・商品名・購入数をcartIn関数に渡している。

**リスト 8. itemDetail関数内でcartIn関数を呼び出す処理**

if($FORM["mode2"] == "cartIn"){

cartIn($FORM["itemID"], $FORM["itemName"], $FORM["orderNum"]);

}

・カートに商品を入れる

カートに商品を入れる処理を関数化しcartInとした場合、プログラムは以下のようになる。

**リスト 9. cartIn関数**

function cartIn($itemID, $itemName, $orderNum){

global $CONF;

$data = array();

$data["itemID"] = $itemID;

$data["itemName"] = $itemName;

$data["orderNum"] = $orderNum;

#セッションがなければ

if (!isset($\_SESSION["cart"])){

$\_SESSION["cart"][] = $data;

#セッションがあれば

}else{

#同じ商品がすでにあるかのフラグ

$flg = false;

#同じ商品があれば数量を変更

for($i = 0, $n = count($\_SESSION["cart"]); $i < $n; $i++){

if($\_SESSION["cart"][$i]["itemID"] == #itemID){

$flg = true;

$\_SESSION["cart"][$i]["orderNum"] = $orderNum;

}

}

if(!$flg){

$\_SESSION["cart"[] = $data;

}

}

}

cartIn関数の処理

1.商品ID・商品名・購入数を配列にセット

2.セッションがなければ、新規で記録

3.セッションがあれば、同じ商品が記録されているチェック

4.同じ商品があれば数量を変更

5.同じ商品がなければ、新規で記録

なお、この関数ではセッションを利用することでカートに記録している。

ユーザーがカートに入れるボタンを押したときは、$\_SESSION["cart"]にデータを追加している。"itemID","itemName","orderNum"という3つのキーを持った$dataという連想配列を作成し、$\_SESSION["cart"]に追加。

すでに同じ商品が入っているかどうかをチェックするために、for文で$\_SESSION["cart"]の中身を一つずつ見る。同じ商品があった場合は、orderNumの値を変更。同じ商品がなかった場合、または$\_SESSION["cart"]自体が存在していない場合は、連想配列$dataごと追加。

このセッションはブラウザを閉じると無効になる。

この時点で、プログラム仕様の１、２、３は全て満たしている。

・カートの中身を表示

次に商品リストと商品詳細ページで、カートの中身が表示されるようにする（仕様4）。表示する内容は、商品名 × 数量。カートの中身を表示する処理も関数にしておき、それを呼び出すようにする。

**リスト 10. get\_cartValue関数**

function get\_cartValue(){

$flg = false;

if(isset($\_SESSION["cart"])){

for($i = 0; $n = count($\_SESSION["cart"]); $i < $n; $i++){

if($\_SESSION["cart"][$i]["orderNum"] > 0){

$flg = true;

$cartValue.="<li>".$\_SESSION["cart"][$i]["itemName"]."X".$\_SESSION["cart"][$i]["orderNum"]."点</li>";

}

}

}

if (!$flg){

$cartValue = "<li>現在カートには何も入っていません。</li>";

}

return $cartValue;

}

この関数ではセッション変数が存在する場合は、商品名と数量を変数に連結し、存在しない場合は"何も入っていません"と代入して、returnで返すという処理を行ってる。あとは商品リスト、商品詳細のそれぞれの関数内で、get\_cartValueの実行結果を変数$cartValueで受け取る処理をする。

これで、4つの仕様全てを満たすことになる。

### カートの中身を表示

カートの中身を表示させる処理は、viewCartという関数名で作成する。カートに入れた商品を一覧で表示し、商品ごとの小計金額、全商品の合計金額を表示させる。また各商品の数量を変更できるようにする。カートに何も入っていない場合は、その旨のメッセージを表示させる。

**リスト 11. viewCart関数**

function viewCart(){

global $CONF, $ITEM, $FORM;

#数量を変更

if ($FORM["mode2"] === "orderNumChg"){

cartIn($FORM["itemID"],$FORM["itemName"],$FORM["orderNum"],$FORM["price"]);

}

$code = openFile("\_tmp\_viewCart.html");

$code = implode("", $code);

$code = explode("<!--section-->", $code);

print $code[0];

#カートの中身を表示

$flg = false;

$syokei = 0;

$goukei = 0;

if (isset($\_SESISON["cart"])){

for($i = 0,$n = count($\_SESSION["cart"]); $i < $n; $i++){

if ($\_SESSION["cart"][$i]["orderNum"] > 0){

$flg = true;

$orderNum = $\_SESSION["cart"][$i]["orderNum"];

$itemName = $\_SESSION["cart"][$i]["itemName"];

$itemID = $\_SESSION["cart"][$i]["itemID"];

$price = $\_SESSION["cart"][$i]["price"];

$syoukei = $price \* $orderNum;

$goukei += $syoukei;

$orderNum1 = "";

$orderNum2 = "";

$orderNum3 = "";

$orderNum4 = "";

$orderNum5 = "";

switch ($orderNum){

case 1:

$orderNum1 = 'selected = "selected"';

break;

case 2:

$orderNum2 = 'selected = "selected"';

break;

case 3:

$orderNum3 = 'selected = "selected"';

break;

case 4:

$orderNum4 = 'selected = "selected"';

break;

case 5:

$orderNum5 = 'selected = "selected"';

break;

}

$line = $code[1];

$line = str\_replace("#itemID#", $itemID, $line);

$line = str\_replace("#itemName#", $itemName, $line);

$line = str\_replace("#orderNum1#", $orderNum1, $line);

$line = str\_replace("#orderNum2#", $orderNum2, $line);

$line = str\_replace("#orderNum3#", $orderNum3, $line);

$line = str\_replace("#orderNum4#", $orderNum4, $line);

$line = str\_replace("#orderNum5#", $orderNum5, $line);

$line = str\_replace("#price#", $price, $line);

$line = str\_replace("#syoukei#", $syoukei, $line);

print $line;

}

}

}

if ($flg){

$code[2] = str\_replace("#goukei#", number\_format($goukei), $code[2]);

print $code[2];

}else{

print $code[3];

}

print $code[4];

exit;

}

viewCart関数の処理

1.数量を変更ボタンが押された場合は、cartIn関数を実行

2.htmlファイルを分解し上部を表示

3.$\_SESSION["cart"]の要素数ループ処理を行い、商品を表示

4.カートに商品がない場合はメッセージを表示

・カート内の商品を表示

カート内に商品があれば、$\_SESSION["cart"]にデータが入っている。この$\_SESSION["cart"]に対して、for分を使ってループ処理を行う。商品の購入数が0より大きい場合は、<!--section-->で区切られた以下のHTML部分を読み込んで変数を置換して表示する。

**リスト 12. ループ処理中に表示するHTML**

<!--section-->

<form action = "ファイル名" method = "POST">

<input type="hidden" name="mode" value="viewCart" />

<input type="hidden" name="mode2" value="orderNumChg" />

<input type="hidden" name="itemID" value="#itemID#" />

<input type="hidden" name="itemName" value="#itemName#" />

<input type="hidden" name="price" value="#price#" />

<table>

<tr>

<td><img src="イメージURL" alt="#itemName#" /></td>

<td>#itemName#</td>

<td>#priceF#円</td>

<td>

<select name = "orderNum">

<option value="0">0個</option>

<option value="1" #orderNum1#>1個</option>

<option value="2" #orderNum2#>2個</option>

<option value="3" #orderNum3#>3個</option>

<option value="4" #orderNum4#>4個</option>

<option value="5" #orderNum5#>5個</option>

</select>

<input type = "submit" value = "数量変更" />

</td>

<td>#syoukei#円</td>

</tr>

</table>

</form>

<!--section-->

カート内の商品1つで一行分のデータとなり、各商品ごとに<form>を用意している。これは数量変更ボタンを押した際に、どの商品の数量を変更するのかを簡単に判別させるためである。また各商品の小計と全商品の合計を計算している。小計は単純に単価×購入数の値で、合計は小計を加算していくだけの処理。購入数の<select>フォームでは、現在の購入数が選択済みになるように、各<option>タグ内に変数を入れている。その変数は、現在の購入数の変数のみselected="selected"の文字列と置き換わる（viewCart関数にて）。

数量変更ボタンを含む<form>内には、以下のタグを埋め込んでいる。

<input type="hidden" name="mode2" value="orderNumChg" />

つまり$FORM["mode2"]にorderNumChgという値が入っていた場合、数量変更ボタンが押されたことを意味し、このときはviewCart関数で再度cartIn関数を呼び出す。cartIn関数ではすでに同じ商品を代入した場合は数量を変更するように処理をしているので、商品のIDと新たに選択された購入数のみを渡すだけで、$\_SESSION["cart"]内の値を変更してくれる。

カートの中身を確認後は購入画面へ進む。

# **参考文献**

[1] PHP と RSS: その仕組みを理解する, <http://www.ibm.com/developerworks/jp/xml/library/x-phprss/index.html>

[2]ハーシー(2005), 速効！図解プログラミング PHP＋MySQL, （株）毎日コミュニケーションズ

[3]サルにもわかる正規表現入門, <http://www.mnet.ne.jp/~nakama/>

[4]メールの送信 | PHP Labo, <http://www.php-labo.net/tutorial/php/mail.html>

[5]離れPHP島, <http://masago.kir.jp/php20030617.php>

[6] ショッピングカートとは【shopping cart】 - 意味/解説/説明/定義 ： IT用語辞典, <http://e-words.jp/w/E382B7E383A7E38383E38394E383B3E382B0E382ABE383BCE38388.html>

[7] PHPプログラム実習 ショッピングカート - PHPプログラムを作る・学ぶ・頼むなら about::PHP by Frogstone, <http://php.frogstone.jp/cart/index.html>