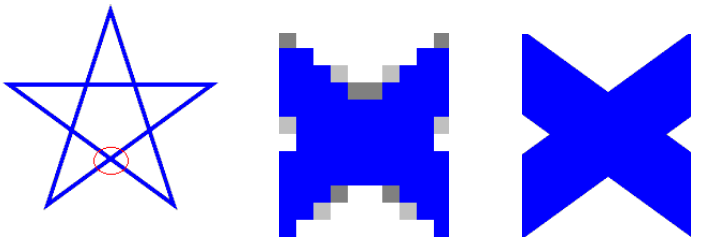
第7章　サーバサイド画像生成

Servlet は HTML 以外のデータを生成することも当然可能である。Web ブラウザで閲覧可能なデータ形式としては例えばラスター画像形式の JPEG, PNG, GIF、ベクター画像形式の PDF, SWF などがある。これらはすべてバイナリ形式であるため、何らかのツールやライブラリの助けなしに Servlet で生成することは骨が折れる。一方、ベクター画像形式の SVG や 3 次元グラフィックスの X3D、数式を記述するための MathML などは、XHTML と同じように XML の一種であり、テキスト形式であるため、Servlet で生成するのは比較的容易である。この章では、Servlet による SVG の生成を取り上げる。

7.1　SVG とは

SVG とは、Scalable Vector Graphics の略で、2 次元のベクター（ベクトル）グラフィックスのための言語仕様である。SVG も HTML と同じく W3C という団体で制定されている。

2 次元画像のファイル形式は大きくラスター形式とベクター形式にわけられる。JPEG、GIF、PNG などの WWW で良く使われている画像形式は、すべてラスター形式に分類される。ラスター形式と言うのは、画像を縦横に分割して、格子（画素）の集まりとして考え、各画素の色を基本要素として扱う形式である。この画素の数を解像度といい、640×480, 320×240 などと表す。ラスター形式で表されている画像の細部を拡大すると、当然この画素が目に見えるようになって、斜めの線などはギザギザになってしまう。



元の画像　　拡大（ラスター）　拡大（ベクター）

一方、ベクター形式というのは、画像の輪郭を直線・曲線の集まりとして捉え、線の代表点（端点・角など）の列として画像を表現する方式である。ベクター形式の場合、画像をいくら拡大しても、線の輪郭は綺麗なままである。

写真などの自然画像はラスター形式でなければ表現することが難しいが、イラストなどの人工的な画像は、本来ベクター形式の方が保存に適している。Adobe 社の PostScript™ 形式や PDF 形式、Macromedia 社の Flash™ が使用する SWF 形式などが代表的なベクター形式の規格である。

SVG は、W3C で仕様が策定され一般に公開されている[1]。ベクター形式のファイルフォーマットで、XHTML と同じく **XML** の一種である。すなわちテキスト形式である、という特徴がある。

[1] http://www.w3.org/TR/SVG/

現時点では Netscape や IE などのブラウザは、標準では SVG をサポートしていないので、プラグインをインストールする必要がある。Windows 用の SVG のプラグインは Adobe の Web サイト（http://www.adobe.co.jp/svg/viewer/install/）または、Corel の Web サイト（http://www.corel.com/svgviewer/）からダウンロードできる。

まずは、SVG を使用している Web サイトを訪れて、プラグインがインストールされていることを確認する。

- Adobe のサイト（http://www.adobe.co.jp/svg/）
- takesato.com “SVG Developers Center”（http://www.takesato.com/svg/）

などを挙げておく。最後のサイトは SVG に関するリンクが豊富である。
試しに次のようなファイルを作成してみる。

```
<?xml version='1.0' ?>
<!DOCTYPE svg PUBLIC '-//W3C/DTD SVG 1.0//EN'
                     'http://www.w3.org/TR/SVG/DTD/svg10.dtd'>
<svg width='300' height='200'>
  <text x='115' y='110' stroke='red' fill='orange' font-size='24'>
    Hello!</text>
  <rect stroke='blue' fill='none'
     x='50' y='50' width='200' height='100' rx='35' ry='25' />
</svg>
```

ファイル名は何でも良いが、拡張子は .svg にしておく。ここでは、first.svg という名前をつけることにする。

一見するとタグ< ... >の使い方などは HTML に似ている。ただし、従来の HTML はアルファベットの大文字と小文字は区別しなかったが、SVG は XML の一種なので大文字と小文字を厳密に区別する。

このファイルを Netscape または IE で表示すると次のようになる。

text は文字列を描くための命令である[2]。<text ... >と</text>の間に描画する文字列を挿入し、... の部分に位置や色などの様々な属性を指定する。rect は長方形を描くための命令である。この命令の rx と ry は長方形の角のまるみを表す属性である。rect と同じような基本図形のためのタグは、この他に

[2]なお、文字列に日本語を使う時は、注意が必要である。現時点では、Adobe SVG Viewer 3.0 は文字コードとして Unicode（UTF-8 と UTF-16）しかサポートしていないようである。

<line ... />	直線を描く
<circle ... />	円を描く
<ellipse ... />	楕円を描く
<polyline ... />	折れ線を描く
<polygon ... />	多角形を描く

などがある。

上記のタグに共通の属性として次のようなものがある。

属性	意味	例	備考
stroke	線の色	stroke='red'	
fill	塗りつぶしの色	fill='#ffff00'	line は除く
stroke-width	線の幅	stroke-width='2'	

stroke と fill には none という値を与えることも可能である。その場合、それぞれ線無し、塗りつぶし無し、という意味になる。

この他にも、一次変換を行なう transform などの属性がある。

また各タグに特有の属性のうち良く使うものには次のようなものがある。

属性	意味
line の属性	
x1	始点の x 座標
y1	始点の y 座標
x2	終点の x 座標
y2	終点の y 座標
rect の属性	
x	左上の点の x 座標
y	左上の点の y 座標
width	幅
height	高さ
rx	角の丸みの x 方向の半径
ry	角の丸みの y 方向の半径
polyline の属性	
points	点列

属性	意味
circle の属性	
cx	中心の x 座標
cy	中心の y 座標
r	半径
ellipse の属性	
cx	中心の x 座標
cy	中心の y 座標
rx	x 方向の半径
ry	y 方向の半径
polygon の属性	
points	点列
text の属性	
x	始点の x 座標
y	始点の y 座標
font-size	フォントのサイズ

polyline と polygon の点列は次の例のように:

$$x_1,\ y_1\ \ x_2,\ y_2\ \ \cdots\ \ x_n,\ y_n$$

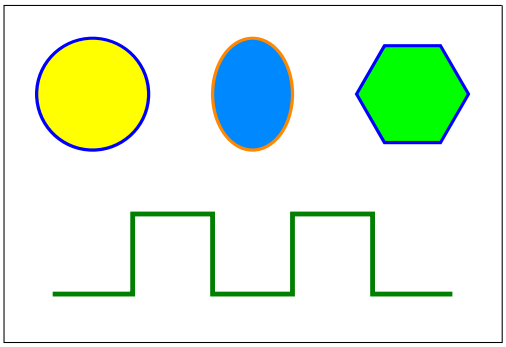
x 座標と y 座標を ,（コンマ）で区切り、さらに各点を空白で区切る、という形式で記述する。

これらのタグを使用した例をあげておく。（最初の灰色の背景の部分は決まり文句である。Adobe の SVG Viewer では、この部分は省略可能のようである。）

ファイル sample.svg

```
<?xml version='1.0' ?>
<!DOCTYPE svg PUBLIC '-//W3C/DTD SVG 1.0//EN'
                     'http://www.w3.org/TR/SVG/DTD/svg10.dtd'>
<svg width='300' height='200'>
<circle stroke='blue' stroke-width='2' fill='yellow' cx='50' cy='50'  r='35' />
<ellipse stroke='#ff8800' stroke-width='2' fill='#0088ff'
         cx='150' cy='50' rx='25' ry='35' />
<polyline fill='none' stroke='green' stroke-width='3'
          points=' 25,175  75,175  75,125 125,125 125,175
                   175,175 175,125 225,125 225,175 275,175' />
<polygon fill='lime' stroke='blue' stroke-width='2'
         points='285, 50  267.5, 80.31  232.5, 80.31
                 215, 50  232.5, 19.69  267.5, 19.69' />
</svg>
```

このファイルは次のように表示される。



また、SVG にはこのような基本図形の他にも、ベジエ（Bezier）曲線という滑らかな曲線を描くための path 命令や、フィルタ効果、グラデーション、アニメーションなどのさまざまな機能がある。

7.2 Servlet による SVG 生成

Servlet 自体の書き方は SVG を生成する場合でも、HTML の場合と大きく変わるところはない。次の例は現在の時刻を SVG の text タグを用いて表示している。

ファイル MyDateSVG.java

```
import java.io.*;
import java.util.Date;
import javax.servlet.*;
import javax.servlet.http.*;

public class MyDateSVG extends HttpServlet {
  public void doGet(HttpServletRequest req, HttpServletResponse res)
              throws IOException {
    res.setContentType("image/svg+xml");
    PrintWriter out = res.getWriter();
    out.println("<svg width='500' height='100'>");
    out.print("<text x='10' y='60' font-size='28'");
    out.println(" stroke='red' fill='orange'>");
    Date d = new Date();
    out.println(d.toString());
    out.println("</text>");
    out.println("</svg>");
    out.close();
  }
}
```

ただし、ブラウザにデータの内容を示す ContentType は、HTML の場合、

```
res.setContentType("text/html; charset= ... ");
```

であったのが、SVG の場合、

```
res.setContentType("image/svg+xml");
```

とする必要がある。生成する SVG データの中で最初の決まり文句の

```
<?xml version='1.0' ?>
<!DOCTYPE svg PUBLIC '-//W3C/DTD SVG 1.0//EN'
                     'http://www.w3.org/TR/SVG/DTD/svg10.dtd'>
```

の部分は、クライアント側で Adobe の SVGViewer を使っている場合は省略可能なようである。そのため、このプリントのこの後の例では、この部分を省略してある。svg タグは width, height 属性を含めて必要である。

この例ではさらに text タグを使用して、文字列を表示している。輪郭が赤色で、内部が橙色の、大きさ 28 の文字となる。

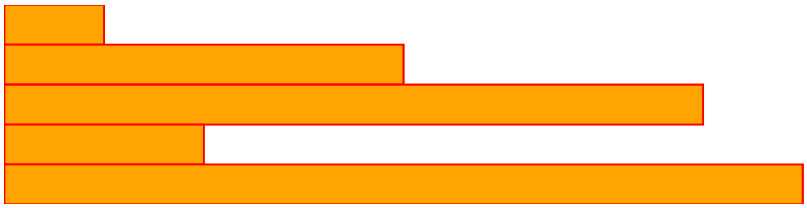
次のプログラムは、QueryString として数字の並び（_で区切られている）を受け取り、その棒グラフを作成する Servlet である。

ファイル GraphSVG.java

```
import java.io.*;
import java.util.StringTokenizer;
import eseservlet.*;

public class GraphSVG extends HttpServlet {
  final private static int ht = 20, unit = 5;
  public void doGet(HttpServletRequest req, HttpServletResponse res) {
    res.setContentType("image/svg+xml");
    String str = req.getQueryString();
    StringTokenizer st = new StringTokenizer(str, "_");
    int c = st.countTokens();

    PrintWriter out = res.getWriter();
    out.println("<svg width='"+(unit*100)+"' height='"+(c*ht)+"'>");

    int i;
    for (i=0; i<c; i++) {
        out.print("<rect stroke='red' fill='orange'");
        int w = Integer.parseInt(st.nextToken());
        out.print(" x='0' y='"+(i*ht)+"' width='"+(unit*w)+"'");
        out.println(" height='"+ht+"' />");
    }
    out.println("</svg>");
    out.close();
  }
}
```

このプログラムでは rect タグを利用して長方形を描画している。

例えば、10_40_70_20_80 という文字列を QueryString として受け取ると、次のようなグラフを生成する。



問 7.2.1 上の例題のグラフの向きを *90* 度回転して、縦方向に伸びるグラフを生成する *Servlet* を作成せよ。

問 7.2.2 （アナログ時計）
現在の時刻から、アナログ（針式）時計の図形を生成する *Servlet* を作成せよ。美的センスは問わないが、長針と短針、秒針ははっきりと区別できるようにする。文字盤の部分（時刻を表す *1*〜*12* の数字や目盛）は必要ない。また、表示した時の時刻を表示できれば、それ以降は “動く” 必要はない。

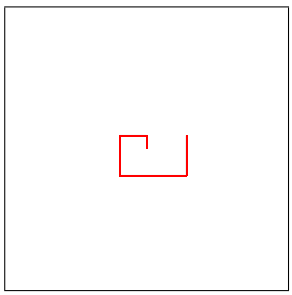
問 7.2.3 （タートルグラフィックス）
物体 *X* は数字と命令文字が交互に現れる文字列を受け取り移動する。命令文字は次のような意味を持ち、

L	*R*	*F*
左回転	右回転	前進

命令文字の直前の数字は回転や移動の量を表す。命令文字と数字は区切り文字_で区切られているとする。

この物体 *X* への指令文字列を *QueryString* として受取り、物体 *X* の軌跡を *SVG* 形式で図示するサーブレットを作成せよ。ただし物体 *X* ははじめ画面の中央にあり、真上を向いているものとする。

例えば、`10_F_90_L_20_F_90_L_30_F_90_L_50_F_90_L_30_F` という *QueryString* を受け取ると、



のような図を作成する。

時間があれば、さらに色や線の太さなどを変更する命令を追加せよ。

問 7.2.4 （度数分布グラフ）
QueryString として上の例題 `GraphSVG.java` と同じような数の列を受け取り、度数分布グラフ（ヒストグラム）を作成する *Servlet* を生成せよ。受け取る数の範囲は *0*〜*100* とし、範囲は *10* ごとに区切るものとする。また、グラフの平均値のところに印をつけよ。

問 7.2.5 （熱方程式）
一次元の（針金のような）物体上の熱の伝播を表す方程式は、次のように離散化して近似することができる。

$$a_i^{t+1} = a_i^t + \Delta t \cdot \alpha \cdot \frac{a_{i-1}^t - 2a_i^t + a_{i+1}^t}{\Delta x^2}$$

a_i^t は端から $i \cdot \Delta x$ メートルの地点の $t \cdot \Delta t$ 秒後の温度である。

ここで針金の長さを *0.64m*、Δx が *0.01m*、Δt が *1* 秒、熱伝導率が 2×10^{-5}、針金の両端が *25* 度と固定されている時、初期温度分布を *64* 個の数値として、*Form* から受け取り、*0*〜*60* 秒の間の熱分布を画像として表示する *Servlet* を作成せよ。

横軸に針金の左端からの距離、縦軸に時間を取り、温度と色の対応を決めて表示する *Servlet* を作成せよ。

キーワード :

SVG, XML, ラスター形式, ベクター形式, X3D, MathML,