Trabalhando com Servlet

Primeiro Servlet

O primeiro passo a ser dado é criar uma classe Java como qualquer outra. Essa nova classe criada deve estender a classe abstrata HttpServlet pertencente ao pacote javax.Servlet.http.

```
import javax.servlet.http.HttpServlet;

public class PrimeiroServlet extends HttpServlet{

}
```

O exemplo acima pode ser considerado um Servlet, contudo não funcional ainda. Além de importar a classe HttpServlet, devemos importar também as seguintes classes:

```
java.io.IOException;
javax.servlet.ServletException;
javax.servlet.http.HttpServletRequest;
javax.servlet.http.HttpServletResponse;
```

Além de importar as classes acima, o Servlet deve sobrescrever um dos métodos abaixo listados; podendo sobrescrever os dois.

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException;
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException;
```

O método doGet será responsável por tratar as requisições enviadas via get, isto é, os parâmetros são enviados fazendo parte da url da requisição. Esse formato de envio tem a vantagem de ser facilmente criado e entendido, contudo só podem ser passados 255 caracteres de informações.

O método doPost será responsável por tratar as requisições enviadas via post, neste caso a quantidade de informações é praticamente ilimitado de informações, podendo ser também enviados arquivos binário; a desvantagem é que o envio de dados dever ser feito através de formulários HTML.

Caso uma requisição não tenha parâmetros, esta deve ser tratada através do método doGet;

Observe que os dois métodos recebem dois parâmetros que são- um objeto HttpServletRequest que modela as requisições e o HttpServletResponse que modela as respostas.

Nosso Servlet de exemplo está praticamente pronto, o último quesito diz respeito ao registro do Servlet no nosso servidor de aplicação.

Atualmente, existem duas formas de se fazer tal procedimento.

1. O primeiro seria inserir a tag Servlet web.xml, como no exemplo abaixo:

```
<Servlet>
        <servlet-name>primeiroServlet</servlet-name> <!-- nome do Servlet dentro do
web.xml -->
        <servlet-class>PrimeiroServlet</servlet-class> <!-- nome da classe do Servlet --
>
</servlet>

<servlet-mapping>
        <servlet-name>primeiroServlet</servlet-name> <!-- nome do servlet dentro do
web.xml -->
        <url-pattern>meuPrimeiroServlet</url-pattern><!-- nome que será mapeado na
url -->
</servlet-mapping>
```

2. A partir da especificação JSR-315 (Servlet 3), é possível registrar o Servlet através de anotações, então, no lugar de usar arquivo de configuração web.xml poderíamos inserir a seguinte anotação antes da definição da classe:

```
@WebServlet("/meuPrimeiroServlet")
```

Lembrando que para poder utilizar a anotação citada, devemos importar a classe javax.Servlet.annotation.WebServlet.

No caso da utilização de anotações, a versão final do nosso arquivo seria:

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/meuPrimeiroServlet")
public class Teste extends HttpServlet {
        private static final long serialVersionUID = 1L;

        protected void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException {

   }

        protected void doGet(HttpServletRequest request, HttpServletResponse
response)
         throws ServletException, IOException {

   }
}
```

Para executar nosso Servlet, clique no menu Executar/Executar Arquivo.

Observe que a url da nossa aplicação é:

Apesar de plenamente funcional, nosso Servlet ainda não apresenta nenhum resultado; com o intuito de resolvermos esse problema, teremos que "escrever" uma resposta.

Primeiramente, importe as classes:
java.io.PrintWriter
No interior do método doGet insira as linhas:

```
        //Indica que a resposta será text/HTML e utiliza a codificação de caracteres
      UTF-8
        response.setContentType("text/html;charset=UTF-8");
        //Cria o objeto responsável por escrever a resposta para o cliente
PrintWriter out = response.getWriter();
  out.println("<html>");
   out.println("<head>");
  out.println("<title>Meu primeiro Servlet</title>");
  out.println("</head>");
  out.println("<body>");
  out.println("<h1>Meu primeiro Servlet</h1>");
  out.println("</body>");
  out.println("</html>");
out.close();
```

Versão final do nosso Servlet:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/meuPrimeiroServlet")
public class Teste extends HttpServlet {
        private static final long serialVersionUID = 1L;

        protected void doPost(HttpServletRequest request,
                        HttpServletResponse response) throws
ServletException, IOException {

        }

        protected void doGet(HttpServletRequest request,
                        HttpServletResponse response) throws
ServletException, IOException {

                response.setContentType("text/html;charset=UTF-8");
                PrintWriter out = response.getWriter();
                out.println("<html>");
                out.println("<head>");
                out.println("<title>Meu primeiro Servlet</title>");
                out.println("</head>");
                out.println("<body>");
```

```
                out.println("<h1>Meu primeiro Servlet</h1>");
                out.println("</body>");
                out.println("</html>");
                out.close();

            }
        }
```

Recebendo parâmetros
Os principais métodos utilizados para recebimento e manipulação de parâmetros são:
String getParameter(String nomeDoParametro)
Retorna o valor associado ao determinado parâmetro.

String[] getParameterValues(String nomeDoParametro)
Retorna todos os valores associados ao determinado parâmetro. (List boxes, Check boxes)

Com o intuito de exibir a utilização dos dois métodos citados, iremos criar duas páginas HTML e um Servlet.

A primeira página possui um formulário que permite ao usuário gerar jogos aleatórios para Mega Sena.
Para isso deve ser criada em Páginas Web uma arquivo HTML com o nome loteria.html. Abaixo, podemos ver esse arquivo.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Loteria</title>
</head>
<body>

    <h3>Sistema de geração de números para loteria</h3>

    <form action="/SegundoServlet">
        <table>
            <tr>
                <td>Nome:</td>
                <td><input type="text" name="nome" />
                </td>
            </tr>
            <tr>
                <td>Quantos números você quer jogar:</td>
                <td><input type="text" name="quantidadeDeNumeros" />
                </td>
            </tr>
            <tr>
                <td>Quantas apostas você quer fazer:</td>
                <td><input type="text" name="quantidadeDeApostas" />
                </td>
            </tr>
            <tr>
                <td colspan="2"><input type="submit" value="Enviar" />
                </td>
            </tr>
```

```
            </table>
        </form>
</body>
</html>
```

Como não foi informado em nosso formulário o método de submissão, este será o padrão que é post.
O código a seguir apresenta nosso segundo Servlet, que criará a resposta adequada para a requisição recebida.

```
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


@WebServlet("/SegundoServlet")
public class SegundoServlet extends HttpServlet{

        protected void doGet(HttpServletRequest req, HttpServletResponse resp)
                        throws ServletException, IOException {

                //O primeiro passo é capturar os atributos

                String nome = req.getParameter("nome");
                String quantidadeDeNumeros =
req.getParameter("quantidadeDeNumeros");
                String quantidadeDeApostas = req.getParameter("quantidadeDeApostas");

                //Note que o método getParameter trata todos os valores recebidos como
String

                //Temos agora que converter esses valores em inteiros


                int quantidadeDeNumerosInt = Integer.parseInt(quantidadeDeNumeros);
                int quantidadeDeApostasInt = Integer.parseInt(quantidadeDeApostas);

                //String que receberá o conteúdo a ser exibido no body da resposta
                String resposta = "<h1> "+nome+" suas apostas</h1>";

                //Bloco onde serão gerados os números
                for(int i = 0; i < quantidadeDeApostasInt; i++){
                        resposta += "<b>Aposta "+ (i+1) + "</b>: ";
                        for(int j = 0; j < quantidadeDeNumerosInt; j++){
                                resposta += ((int) (Math.random()*59))+1; ;
                                if(j==quantidadeDeNumerosInt-1){
                                        resposta += "<br/>";
                                }else{
                                        resposta += ";";
                                }
                        }
```

```
            }

            resp.setContentType("text/html;charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Loteria</title>");
        out.println("</head>");
        out.println("<body>");
        out.println(resposta);
        out.println("</body>");
        out.println("</html>");
        out.close();
          }
}
```

Para exemplificar a utilização do método getParameterValues, usaremos outro formulários HTML, que irá questionar que itens de consumo ele possui em casa. Nesse exemplo, passaremos os atributos por post.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
<title>Pesquisa</title>
</head>
<body>
<form action="SegundoServlet" method="post">
<fieldset>
<legend>Pesquisa</legend>
None:<input type="text" name="nome"/><br/>
Telefone:<input type="text" name="fone"/><br/>
Quais dos itens você tem em casa?<br/>
<input type="checkbox" name="iten" value="Aparelho de DVD" /> Aparelho de
DVD<br/>
<input type="checkbox" name="iten" value="Aparelho de som" /> Aparelho de
som<br/>
<input type="checkbox" name="iten" value="Geladeira Duplex" /> Geladeira
Duplex<br/>
<input type="checkbox" name="iten" value="Microondas" /> Microondas<br/>
<input type="checkbox" name="iten" value="Máquina de lavar " />Máquina de lavar
<br/>
<input type="checkbox" name="iten" value="Telefone sem fio" />Telefone sem fio
<br/>
<input type="checkbox" name="iten" value="TV de LCD" />TV de LCD <br/>
<input type="checkbox" name="iten" value="Notebook"/>Notebook<br/>
<input type="submit" value="Enviar">
</fieldset>
</form>
</body>
</html>
```

Para tratar a requisição enviada para pelo formulário acima descrito, sobrescreveremos o método doPost, como a seguir:

```
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
                    throws ServletException, IOException {

                //O primeiro passo é capturar os atributos únicos
                String nome = req.getParameter("nome");
                String telefone = req.getParameter("fone");

                //Para o caso dos itens selecionados devemos usar o método
                //getParameterValues que devolve
                // uma Array de Strings com os valores selecionados
                String itens[] = req.getParameterValues("iten");

                //String que receberá o conteúdo a ser exibido no body da resposta
                String resposta = "<h1> "+nome+" você selecionou os itens:
</h1><br/>";

                //Bloco onde serão gerados os números

                for(String s: itens){
                        resposta += s + "<br/>";
                }

                resposta +=
                "<br/><br/><h3>Entraremos em contato através do telefone:
                "+telefone+"</h3>";

                resp.setContentType("text/html;charset=UTF-8");
        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Pesquisa</title>");
        out.println("</head>");
        out.println("<body>");
        out.println(resposta);
        out.println("</body>");
        out.println("</html>");
        out.close();
        }
```

Redirecionamento no fluxo de páginas

Para se fazer redirecionamento com Servlet, temos, principalmente, duas formas. A primeira utiliza o método sendRedirect, existente nos objetos HttpServletResponse; a outra forma é utilizar o método forward dos objetos RequestDispatcher. A diferença entre os dois métodos é que o segundo pode repassar a requisição recebida para um outro Servlet, enquanto no primeiro a requisição é perdida.

Vamos criar uma página html de erro (erro.html), que será alvo do redirecionamento, caso no nosso primeiro exemplo o usuário digite um valor numérico que gere uma exceção o fluxo será direcionado para essa página de erro.

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1" />
<title>Erro</title>
</head>
<body>
<h1 style="color: red;">Ocorreu uma exceção!</h1>
</body>
</html>
```

Para que o fluxo seja redirecionado, devemos tratar as conversões de números existentes no método doGet, como se segue.

```
int quantidadeDeNumerosInt = 0;
int quantidadeDeApostasInt = 0;
try{
quantidadeDeNumerosInt =
Integer.parseInt(quantidadeDeNumeros);
quantidadeDeApostasInt = Integer.parseInt(quantidadeDeApostas);
}catch (Exception e) {
        //Redirecionamos caso ocorra uma exceção
        resp.sendRedirect("erro.html");
}
```

A segunda forma de redirecionamento será demonstrado no método doPost. Nesse caso, redirecionaremos para um novo Servlet Erro, que exibirá uma mensagem caso o usuário não selecione pelo menos um item.

Devemos importar a classe: javax.Servlet.RequestDispatcher.

```
if(itens == null){
        RequestDispatcher rd = req.getRequestDispatcher("Erro");
        rd.forward(req, resp);
}
```

Nosso Servlet de Erro:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/Erro")
public class Erro extends HttpServlet {

        protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
                response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<head>");
        out.println("<title>Pesquisa</title>");
        out.println("</head>");
        out.println("<body>");
       //Nome que o nome passado para Segundo Servlet está
         //disponível pois a requisição em que o atributo estava inserido foi
```

```
  //repassado para o Servlet Erro
out.println(request.getParameter("nome"));
out.println("</body>");
out.println("</html>");
out.close();
  }

}
```