Tutoriais

Objetos Implícitos
Parte 5: Criando funções que manipulam objetos implícitos

por: Ulisses Telemaco Neto
ulisses@jspbrasil.com.br

Como foi visto em módulos anteriores, é possível definir funções na própria página JSP através das tags <%! e %>. Veja o exemplo abaixo:

```
<%!
String jogueMoeda(){
  String retorno = "";
  int i = (int)(Math.random()*10); //um número entre 0 e 10
  i = i%2;                         //resto da divisao de i por 2
  if(i==0){
     retorno = "cara";
  } else {
     retorno = "coroa";
  }
  return retorno;
}
%>

<html>
<body>

<h1>Deu <%= jogueMoeda() %>!</h1>

</body>
</html>
```

Nesse trecho de código, uma função é definida dentro de uma página JSP. Essa função é bastante simples: simula o lançamento de uma moeda para cima e escolhe, aleatoriamente, entre cara ou coroa.

Imagine agora que queremos que a função simule o lançamento da moeda várias vezes seguidas. Podemos alterar o código anterior e chamarmos a função várias vezes:

```
....
<html>
<body>

<h1>Deu <%= jogueMoeda() %>,
<%= jogueMoeda() %> ...
<%= jogueMoeda() %>!</h1>

</body>
</html>
```

Essa, solução, apesar de funcionar, não é a mais adequada. Podemos reescrever a função para receber como parâmetros a quantidades de jogadas que deverão ser efetuadas e fornecer um meio para que a função possa escrever o conteúdo na página JSP:

```
<%!
void jogueMoeda(int numeroJogadas, javax.servlet.jsp.JspWriter o){
  try{
    for(int k=1; k <= numeroJogadas; k++){
      int i = (int)(Math.random()*10); //número entre 0 e 10
      i = i%2;
      if(i==0){
         o.print("cara");
      } else {
         o.print("coroa");
      }
      if(k==numeroJogadas){
         o.print("!");
      } else {
         o.print(", ");
      }
    }
  } catch(Exception e){}
}
%>

<html>
<body>

<h1>Deu <% jogueMoeda(10,out); %></h1>

</body>
</html>
```

Essa segunda versão, apesar de ter uma funcionalidade bem parecida com a anterior traz algumas novidades:
Passamos como parâmetro um objeto da interface javax.servlet.jsp.JspWriter. É através desse objeto que iremos enviar para a página JSP o resultado da "jogada".
Na chamada da função passamos, além do número de jogadas desejadas, uma referência ao objeto implícito out. É através dessa referência que a função irá imprimir na saída da página JSP o resultado de sua jogada.

É importante aprender que não podemos chamar o objeto out (ou qualquer outro objeto implícito) diretamente em uma função declarada dentro de uma página JSP. Portanto, o seguinte código está errado e apresentará um erro em tempo de compilação:

```
<%!
void jogueMoeda(int numeroJogadas){
  for(int k=1; k <= numerovezes; k++){

    int i = Math.random();
    i = i*10;
    i = i%2;
    if(i==0){
       out.print("cara");
    } else {
       out.print("coroa");
    }
        if(k==numeroJogadas){
           out.print("!");
        } else {
           out.print(", ");
        }
  }
}
%>
<html>
<body>

<h1>Deu <% jogueMoeda(50) %></h1>

</body>
</html>
```

A tabela a seguir exibe como criar e chamar funções que usam os nove objetos implícitos:

Objeto	Assinatura da Função:	Chamada da Função na página JSP:

page	void f(javax.servlet.jsp.HttpJspPage p){
	...
	p.algumaFuncaoDoObjetoPage();
	...
	}
		<%
		f(page);
		%>

config	void f(javax.servlet.Config c){
	...
	c.algumaFuncaoDoObjetoConfig();
	...
	}
		<%
		f(config);
		%>

request	void f(javax.servet.http.HttpServletRequest rq){
	...
	rq.algumaFuncaoDoObjetoRequest();
	...
	}
		<%
		f(request);
		%>

response	void f(javax.servet.http.HttpServletResponse rp){
	...
	rp.algumaFuncaoDoObjetoResponse();
	...
	}
		<%
		f(response);
		%>

out	void f(javax.servet.jsp.JspWriter o){
	...
	o.algumaFuncaoDoObjetoOut();
	...
	}
		<%
		f(out);
		%>

session	void f(javax.servet.http.HttpSesseion s){
	...
	s.algumaFuncaoDoObjetoSession();
	...
	}
		<%
		f(session);
		%>

application	void f(javax.servet.ServletContext a){
	...
	a.algumaFuncaoDoObjetoApplication();
	...
	}
		<%
		f(application);
		%>

pageContext	void f(javax.servet.jsp.PageContext pc){
	...
	pc.algumaFuncaoDoObjetoPageContext();
	...
	}
		<%
		f(pageContext);
		%>

exception	void f(java.lang.Throwable e){
	...
	e.algumaFuncaoDoObjetoException();
	...
	}
		<%
		f(exception);
		%>

Obs.: As funções acima exemplificadas recebem apenas um parâmetro e não têm valor de retorno. Contudo, podemos criar funções que recebem vários parâmetros e tenham algum tipo de retorno.		