Tutoriais

Objetos Implícitos
Parte 2: Objetos Relacionados a OutPut/InPut da página

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

Os objetos classificados nessa categoria se concentram no input (entrada de dados) e output (saída de informação) de uma página JSP. Os três objetos classificados nessa categoria são:
request
response
out

Objeto request

O objeto request representa a solicitação que requisitou a página. O objeto request implementa a interface javax.servlet.http.HttpServletRequest (subinterface de javax.servlet.ServletRequest). Esse objeto, que também é classificado como um objeto contextual (parte 3), é um dos mais complexos e mais utilizados na construção de páginas JSP. Podemos dividir os métodos desse objeto em quatro funcionalidades:

Armazenar e Recuperar valores de atributos:

Método	Descrição
void setAttribute(String key, Object value)	Associa um valor de atributo com um nome.
Enumeration getAttributeNames()	Recupera os nomes de todos os atributos associados com o objeto.
Object getAttribute(String key)	Recupera o valor de atributo associado com a chave.
void removeAttribute(String key)	Remove o valor de atributo associado com a chave.

Recuperar parâmetros de solicitação e cabeçalho de HTTP:

Métodos	Descrição
Enumeration getParameterNames()	Retorna os nomes de todos os parâmetros de solicitação.
String getParameter(String name)	Retorna os primeiro valor (principal) de um único parâmetro de solicitação.
String[] getParameterValues(String name)	Recupera todos os valores para um único parâmetro de solictação.

Recuperar cabeçalhos de solicitação e cabeçalhos de HTTP:

Métodos	Descrição
Enumeration getHeaderNames()	Recupera os nomes de todos os cabeçalhos associados com a solicitação.
String getHeader(String name)	Retorna o valor de um único cabeçalho de solicitação, como uma cadeia.
Enumeration getHeaders(String name)	Retorna todos os valores para um único cabeçalho de solicitação.
int getIntHeader(String name)	Retorna o valor de um único cabeçalho de solicitação, com um número inteiro.
long getDateHeader(String name)	Retorna o valor de um único cabeçalho de solicitação, como uma data.
Cookies[] getCookies()	Recupera todos os cookies associados com a solicitação.

Diversos métodos da interface que javax.servlet.hhtp.HttpServletRequest:

Métodos	Descrição
String getMethod()	Retorna o método de HTTP (e.g, POST, GET, etc) para a solicitação.
String getRequestURI()	Retorna o URL de solicitação (não inclui a cadeia de consulta).
String getQueryString()	Retorna a cadeia de consulta que segue o URL de solicitação, se houver algum.
HttpSession getSession()	Recupera os dados da sessão para a solicitação (i.e, o objeto implícito session).
HttpSession getSession(boolean flag)	Recupera os dados da sessão para a solicitação (i.e, o objeto implícito session), opcionalmente criando-o se ele ainda não existir.
RequestDispatcher getRequestDispatcher(String path)	Cria um dispatcher de solicitação para o URL local indicado.
String getRemoteHost()	Retorna o nome totalmente qualificado do host que enviu a solicitação.
String getRemoteAddr()	Retorna o endereço de rede (IP) do host que enviou a solicitação.
String getRemoteUser()	Retorna o nome do usuário que enviou a solicitação, se conhecido.

Abaixo estão os links para a API que define essas interfaces:
ServletRequest
HttpServletRequest

Exemplo utilizando o objeto implícito request:

```
.....
Seu IP é :<%= request.getRemoteAddr() %><br>
Seu Host é :<%= request.getRemoteHost() %><br>
.....
```

Objeto response

O objeto response representa a resposta que será enviada de volta para o usuário como resultado do processamento da página JSP. Este objeto implementa a interface javax.servlet.http.HttpServletResponse que é uma subinterface de javax.servlet.ServletResponse.

Podemos dividir os métodos desse objeto em quatro funcionalidades:

Especificar o tipo de conteúdo e codificação da resposta:

Métodos	Descrição
void setContentType(String type)	Define o tipo MIME e, opcionalmente, a codificação de caracteres do conteúdo da resposta.
String getCharacterEncoding()	Retorna o conjunto de estilos de codificação de caracteres para o conteúdo da resposta.

Definir cabeçalhos da resposta:

Métodos	Descrição
void addCookies(Cookie cookie)	Adiciona o cookie especificado.
boolean containsHeader(String name)	Verifica se a resposta inclui o cabeçalho.
void setHeader(String name, String value)	Atribui o valor definido pela variável "value" ao cabeçalho especificado por "name"
void setIntHeader(String name, int value)	Atribui o valor de número inteiro especificado por "value" ao cabeçalho especificado por "name"
void setDateHeader(String name, long date)	Atribui o valor de data especificado por "value" ao cabeçalho especificado por "name"
void addHeader(String name, String value)	Adiciona o valor definido por "value" ao cabeçalho especificado por "name"
void addIntHeader(String name, int value)	Adiociona o valor de número inteiro especificado por "value" ao cabeçalho especificado por "name"
void addDateHeader(String name, long date)	Adiciona o valor de data especificado por "value" ao cabeçalho especificado por "name"

Definir códigos de resposta:

Métodos	Descrição
void setStatus(int code)	Define o código de status para a resposta (para cisrcuntâncias sem erro)
void sendError(int status, String msg)	Define o código de status e mensagem de erro para a resposta.
void sendRedirect(String url)	Envia uma resposta para o navegador indicando que ele deveria solicitar um URL alternativo (absoluto)

Reescrita da URL:

Métodos	Descrição
String encodeRedirectURL(String url)	Codifica um URL para uso com o método sendRedirect() para incluir informações de sessão.
String encodeURL(String url)	Codifica um URL usado em um link para incluir informações de sessão.

Abaixo estão os links para a API que define essas interfaces:
ServletResponse
HttpServletResponse

O exemplo abaixo ilustra uma das utilidades desse objeto. Vários cabeçalhos são definidos para evitar que a página seja armazenada em cache por um navegador.

```
<%
response.setHeader("Expires", 0);
response.setHeader("Pragme, "no-cache");
if(request.getProtocol().equals("HTTP/1.1")){
    response.setHeader("Cache-Control", "no-cache");
}
%>
```

Este script primeiro define o cabeçalho Expires para uma data no passado. Isto significa que o conteúdo da página já expirou, como uma dica que seu conteúdo não deve ser armazenado em cache.

Objeto out

Este objeto implícito representa o fluxo de saída para a página, cujo conteúdo será enviado para o navegador com o corpo de sua resposta. O objeto out é uma intância da classe javax.servlet.jsp.JspWriter.
Esse objeto implementa todos os métodos print() e println() definidos por java.io.Writer. Por exemplo, o objeto out pode ser usado dentro de um script para adicionar conteúdo à página gerada. Veja o exemplo abaixo:

```
<%
int i = (int)(Math.random()*10);
if(i%2==0){
    out.print("O Número escolhido "+ i +" é par!");
}
else {
    out.print("O Número escolhido "+ i +" é impar!");
}
%>
```

Esse objeto é muito utilizado para para gerar conteúdo dentro do corpo de um script, sem ter que fechá-lo temporariamente para inserir conteúdo de página estático. Contudo, deve-se evitar usar os métodos print() ou println() para inserir cadeias de caracteres muito grandes. No próximo caso, é mais aconselhável fechar o script e inserir o conteúdo estático. Veja o exemplo abaixo:

```
Não Aconselhável:
<%
if(i == 1){
out.print("<h6>"+
"<font face='verdana'>"+
"Guga viaja nesta sexta"+
"para a Suíça para"+
"jogar"+
"</font>"+
"</h6>");
}
%>

Aconselhável:
<% if(i == 1) {%>
<h6>
<font face='verdana'>
Guga viaja nesta sexta
para a Suíça para
jogar
</font>
</h6>
<% } %>
```

O links abaixo refere-se a API que define a interfaces que o objeto out implementa: JspWriter