Transcrição

Seguiremos o projeto e aprenderemos mais funcionalidades do Servlets. Teremos mais algumas explicações teóricas e configurações, mas parte a parte vamos apreendendo o conteúdo.

Um pequeno resumo do que fizemos até o momento: usamos o navegador para enviar uma requisição, o Servlet foi mapeado e devolveu uma resposta que criamos no código Java. Construímos a comunicação remota, isto é, nosso navegador poderia ser aberto em outro computador e acessar o Tomcat por meio da rede.

Retomaremos também a construção do nosso código: Em `OiMundoServlet.java`, temos a configuração `@WebServlet(urlPatterns="/oi")`, a URL que evoca o Servlet. Temos o método `service()` que recene os objetos `HttpServletRequest` e `HttpServletResponse`, isto é, referências fundamentais de requisição e resposta. A partir da `resp`, usaremos o `getWriter()` para de fato devolver algum conteúdo ao navegador.

Por fim, utilizamos o `System.out.println()` para garantirmos que o Servlet é chamado.

```
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;

//oi
@WebServlet(urlPatterns="/oi")
public class OiMundoServlet extends HttpServlet {

    @Override
    protected void service(HttpServletRequest req, HttpServletResponse resp) th

        PrintWriter out = resp.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("oi mundo, parabens vc escreveu o primeiro servlets.");
        out.println("</body>");
        out.println("</html>");

        System.out.println("o servlet OiMundoServlet foi chamado");
    }
}
```

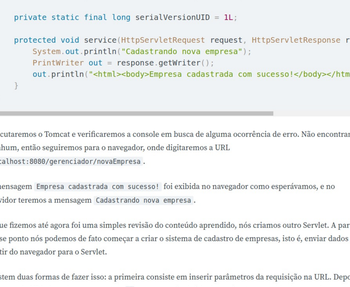
A próxima meta é criar um sistema de cadastro para empresas por meio do Servlet. Passaremos a enviar dados para este pequeno servidor e precisaremos devolver uma resposta de confirmação de cadastro.

Faremos um novo Servlet. Na pasta "gerenciador > Java Resources > src > br.com.alura.gerenciador", clicaremos com o botão direito e selecionaremos a opção "New > Servlet", ou seja, por meio do Eclipse podemos criar um Servlet. Caso essa opção não esteja visível para você, selecione "New > Other" e digite no campo "Wizard" a palavra "Servlet".

Na caixa de diálogo aberta, chamaremos a nova classe de `NovaEmpresaServlet`. A "Superclass" será estendida, isto é `javax.servlet.http.HttpServlet`. Pressionaremo o botão "Next" para conhecer mais algumas configurações.

Teremos a opção "URL mappings", isto é, o mapeamento da URL pode ser modificado. Clicaremos sobre a opção "Edit". e modificaremos o nome de `/NovaEmpresaServlet` para `/novaEmpresa`.

Pressionaremos o botão "Next". Nessa nova área, teremos como ecolher métodos que podem ser previamente implementados pelo Eclipse. Iremos desmarcar a opção de construtor padrão "Constructors fro superclass"; e desmarcaremos também os métodos "DoPost" e "doGet".



Dessa forma, teremos nossa classe criada:

```
package br.com.alura.gerenciador.servlet;

import java.io.IOException;

/**
*Servlet implementation class NovaEmpresaServlet
*/
@WebServlet("/novaEmpresa")
public class NovaEmpresaServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#service(HttpServletRequest request, HttpServletResponse
     */
    protected void service(HttpServletRequest request, HttpServletResponse resp
        //TODO Auto-generated method stub
    }
}
```

Removeremos os comentários. Teremos, ainda, o `serialVersionUID` para que desapareça um *warning* relacionado ao Java IO, mas isto é totalmente opcional.

O Servlet está pronto, para testá-lo adicionaremos o `System.out.println()` com a mensagem `Cadastrando nova empresa`, que será impressa somente na console. Em seguida, estruturaremos também a mensagem HTML por meio do `PrintWriter()`, especializado no envio de caracteres. Usaremos a referência `out`, e o coletaremos por meio de `response.getWriter()`. Devemos importar `PrintWriter` para evitar erros de compilação.

Escreveremos a mensagem HTML a ser lida pelo navegador em apenas uma linha.

```
package br.com.alura.gerenciador.servlet;

import java.io.IOException;

@WebServlet("/novaEmpresa")
public class NovaEmpresaServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse resp
        System.out.println("Cadastrando nova empresa");
        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa cadastrada com sucesso!</body></html>"
    }
}
```

Executaremos o Tomcat e verificaremos a console em busca de alguma ocorrência de erro. Não encontramos nenhum, então seguiremos para o navegador, onde digitaremos a URL `localhost:8080/gerenciador/novaEmpresa`.

A mensagem `Empresa cadastrada com sucesso!` foi exibida no navegador como esperávamos, e no servidor teremos a mensagem `Cadastrando nova empresa`.

O que fizemos até agora foi uma simples revisão do conteúdo aprendido, nós criamos outro Servlet. A partir desse ponto nós podemos de fato começar a criar o sistema de cadastro de empresas, isto é, enviar dados a partir do navegador para o Servlet.

Existem duas formas de fazer isso: a primeira consiste em inserir parâmetros da requisição na URL. Depois do nome da URL, inserimos o caractere `?` que marcará o início do parâmetro, seguido de seu valor. Vejamos um exemplo hipotético, cujo nome do parâmetro será `nome` e o valor `Alura`. Para adicionarmos mais um parâmetro adicionaremos o caractere `&`, seguindo do `cnpj` com valor `123`.

```
localhost:8080/gerenciador/novaEmpresa?nome=Alura&cnpj=123
```

Por enquanto usaremos apena um parâmetro, e digitaremos no nosso navegador `localhost:8080/gerenciador/novaEmpresa?nome=Alura`, e os parâmetros foram enviados. A página continua funcionando perfeitamente. Ao analisarmos o servidor no Eclipse, veremos que ele foi chamado novamente, pois a mensagem `Cadastrando nova empresa` foi impressa no console.

Gostaríamos de ler o novo parâmetro enviado. A `request` representa o objeto que recebeu os dados da requisição do navegador, portanto podemos acessar os parâmetros dessa requisição. Escreveremos `request.getParameter()`, que recebe uma string. Colocaremos o nome do parâmetro, no caso, `nome`. Será devolvida uma string `nomeEmpresa`.

Por fim, concatenaremos na mensagem HTLM a variável `nomeEmpresa`.

```
package br.com.alura.gerenciador.servlet;

import java.io.IOException;

@WebServlet("/novaEmpresa")
public class NovaEmpresaServlet extends HttpServlet {

    private static final long serialVersionUID = 1L;

    protected void service(HttpServletRequest request, HttpServletResponse resp
        System.out.println("Cadastrando nova empresa");
        String nomeEmpresa = request.getParameter("nome");
        PrintWriter out = response.getWriter();
        out.println("<html><body>Empresa cadastrada "+ nomeEmpresa +" com suces
    }
}
```

Salvaremos as modificações e executaremos o Tomcat, talvez tenhamos de esperar um pouco para que o servidor faça o *reload* das informações. Feito isso, escreveremos a URL `localhost:8080/gerenciador/novaEmpresa?nome=Alura`, e teremos a mensagem `Empresa Alura cadastrada com sucesso!`, caso modifiquemos a URL para `localhost:8080/getenciador/novaEmpresa?nome=Caleum`, a mensagem será atualizada para `Empresa Caelum cadastrada com sucesso!`.

Chamamos o Servlet, enviamos um parâmetro na requisição, lemos esse parâmetro e depois o concatenamos com a resposta exibida no navegador. Podemos fazer esse processo ser mais elaborado, mas a primeira parte já conseguimos completar, enviamos os dados por meio da requisição HTTP.