03

Mostrando e resolvendo o erro

Transcrição

Encontrando-se o erro, há algumas maneiras de resolvê-lo. Sabemos que a coluna "nome" no banco de dados ("mysql") possui limite de `50` caracteres... Antes de resolvermos, porém, é importante notar que, muitas vezes, o problema poderia ser resolvido mais rapidamente se nos atentassemos às boas práticas na hora de desenvolver o código.

O programador utilizou *Try/Catch*, mas não tratou a exceção. O mínimo que deveria ser feito seria lançá-la, printando-a no console do Tomcat, a nossa container. No entanto, o ideal seria enviar uma mensagem ao usuário, pois ele ficou desamparado diante do erro, sem saber o que estava acontecendo, gerando uma situação bem ruim.

Para mostrar que já temos um `servlet` de erro em `ErrorController.java`, após seu recebimento precisamos enviá-lo à tela do usuário. Idealmente, lançaremos a exceção até o `servlet` para que se possa capturar o erro, jogando-o (`throw`) de volta para o usuário, informando-o sobre o que está acontecendo. Então, o código fica assim:

```
public void salvar(Convidado convidado){

  try {
    //...

  } catch (SQLException e) {
    e.printStackTrace();
    throw new SQLException();
  }

}
```

Com o mouse sobre `throw`, clicaremos na opção "*Add throws declaration*", ou seja, vamos lançar a exceção a este método, em `ConvidadoRepository.java`. O arquivo `ConvidadoService.java` precisa tratar este erro na hora de salvar, ou então lançá-lo. Com esta aba aberta, e o mouse em cima de `ConvidadoRepository()`, selecionaremos a opção "*Add throws declaration*" de novo, adicionando a exceção à declaração. O próximo método que capturar isto precisa tratar a exceção. Vamos salvar e abrir a aba de `ConvidadoServlet.java`. Trataremos a exceção acrescentando-se um "*Try/Catch*", mantendo o mouse sobre `convidadoService.salvar(novoConvidado);` e selecionando "*Surround with try/catch*".

Não iremos somente printar o erro, e sim mostrar que a exceção foi lançada, chegando ao `servlet` de erro, ou seja, lançaremos uma exceção de outra maneira, usando `throw new ServletException("Deu erro, e o erro é " + e);` na aba `ConvidadoServlet.java`:

ConvidadoServlet.java | ConvidadoService.java | ConvidadoRepository.java | ErrorController.java

```java
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;

@WebServlet("/convidado")
public class ConvidadoServlet extends HttpServlet {

    private static final long serialVersionUID = -2949873189744763662L;

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

        String nome = req.getParameter("nome");
        String email = req.getParameter("email");
        String telefone = req.getParameter("telefone");

        Convidado novoConvidado = new Convidado(nome, email, telefone);

        ConvidadoService convidadoService = new ConvidadoService();
        try {
            convidadoService.salvar(novoConvidado);
        } catch (SQLException e) {
            throw new ServletException("Deu erro, e o erro é: " + e);
        }

        List<Convidado> convidados = new ConvidadoService().obterTodos();

        req.setAttribute("convidados", convidados);
        req.getRequestDispatcher("/lista.jsp").forward(req, resp);
```

Agora sim, já temos a exceção a ser capturada e mostrada ao usuário. Quando houverem exceções, mapeamos ao `servlet` de erro a partir de `web.xml` em "listavipServlet > WebContent > WEB-INF > `web.xml` ", à qual incluiremos uma página de erro, com duas tags, `<exception-type>` e `<location>` :

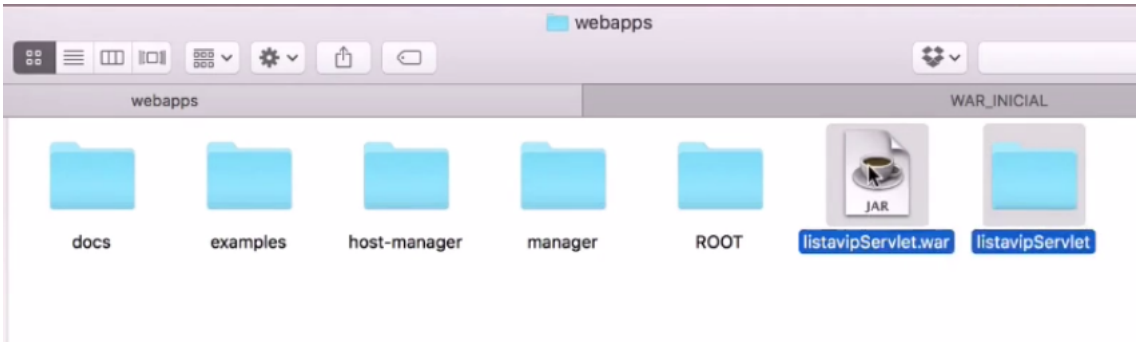
```
<error-page>
  <exception-type>java.lang.Exception</exception-type>
  <location>/error</location>
</error-page>
```

Salvaremos estas alterações, e agora todas as exceções serão lançadas ao `servlet` . Ao alterarmos o código, podemos salvá-lo e a alteração será refletida no Tomcat. Porém, como modificamos o `web.xml` , precisaremos fazer um *deploy* novo.

Para isto, precisamos gerar o `.war` , artefato a ser colocado no Tomcat, para realizarmos um *deploy* em produção. Vamos clicar com o lado direito do mouse na pasta `listavipServlet` , exibido ao lado esquerdo da página, selecionando "*Export* > *WAR file*". Na janela que se abriu, criaremos uma pasta nova de destino, a qual denominaremos "novo_deploy", clicando em "*Finish*" depois.

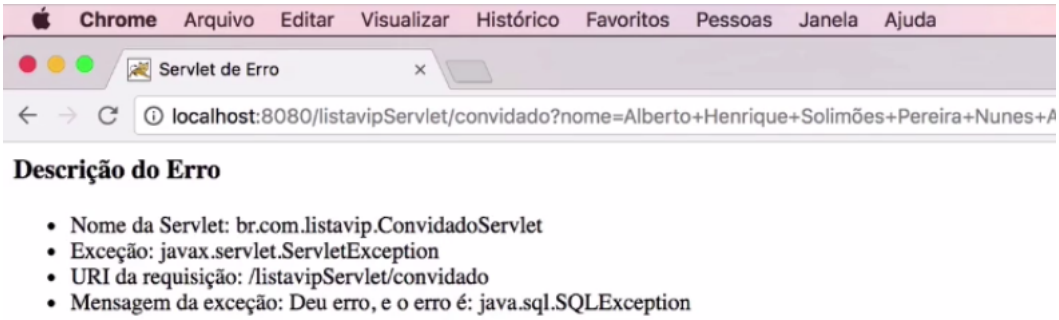
Iremos ao Tomcat, pausaremos o servidor, limparemos, colocaremos o `.war` novo, após o qual subiremos o servidor novamente. No terminal, nos encontramos na pasta "logs". Subiremos um nível e entraremos na pasta "bin" e, com o comando `./shutdown.sh` , derrubaremos o servidor.

Voltando ao "Finder" (Arquivos), selecionaremos os arquivos abaixo, deletando-os em seguida:



Acessando-se "Mesa > projetoWeb > novo_deploy", encontraremos o arquivo `listavipServlet.war` atualizado. Copiaremos e colaremos em "webapps". Vamos subir o servidor novamente utilizando o JPDA, como visto em vídeo anteior. No terminal, digitaremos `./catalina.sh jpda start`. Verificaremos estas alterações no *browser*, abrindo a url localhost:8080/listavipServlet.

O link está funcionando corretamente, então vamos tentar simular o cadastro novamente para verificar se desta vez a exceção é lançada ao usuário. Ao completarmos o formulário e clicarmos em "Convidar", somos redirecionados a esta página de erro:



Trata-se de uma tela de erro em desenvolvimento, sendo que não é recomendável colocar este tipo de informação à disposição, pois pode ocorrer de ser um erro de segurança, por exemplo, informação que é melhor evitar deixar vulnerável.

Caso você queira que outra página de erro seja mostrada, basta ir à aba `ErrorController.java` e fazer outro redirecionamento, ou alterar o texto referente, o que achar melhor, apenas tomando cuidado com questões de segurança do site - existem cursos sobre isto na Alura, a partir dos quais é possível obter maiores informações sobre o assunto.

Voltando ao Eclipse, podemos arrumar o erro tendo em vista as seguintes opções: colocamos um `limit` ao `input` para que não seja ultrapassado os `50` caracteres disponíveis, ou, caso tenhamos acesso ao banco de dados como neste caso (algo que não ocorre sempre, uma vez que nem sempre a DBA permite este acesso), podemos aumentar a coluna "nome" para abrigar mais de `50` caracteres.

Inicialmente, incluiremos `maxlength` no `input`, depois, acessaremos a pasta `lista.jsp` e buscando esta tag, em que definiremos o limite de caracteres:

Lembrando que em "mysql", a coluna "nome" possui "varchar" de `50` caracteres, digitaremos a linha abaixo no terminal:

```
ALTER TABLE `listavip`. `convidado` CHANGE COLUMN `nome` `nome` VARCHAR(200) NULL DEFAULT NULL
```

Após o qual rodaremos `desc convidado;` digitando-o no terminal, e verificamos que o "varchar" realmente passou de `50` para `200`. Como alteramos o arquivo `.jsp`, precisaremos refazer o *deploy* no Tomcat, repetindo os procedimentos feitos anteriormente: clicando-se na pasta do projeto com o lado direito do mouse, selecionando "*Export > WAR file*", tendo como destino a mesma pasta `novo_deploy`, pois sobrescreveremos os arquivos.

O `.war` novo foi gerado, pausaremos o Tomcat, copiaremos e colaremos este arquivo na pasta "webapps", substituindo as versões anteriores, subindo novamente o servidor (por meio de `./catalina.sh jpda start`). Vamos ao navegador verificar a alteração feita. Na raiz, apertaremos o botão "Clique aqui para ver a lista de convidados", colocando os dados do usuário no formulário.

O cadastro foi feito com sucesso, e os dados aparecem na lista em cima do formulário. Resolvemos o problema! No entanto, para além de consertarmos bugs, vamos tomar cuidado com boas práticas para sermos cada vez melhores programadores, o que inclui atenção ao suporte ao cliente, à experiência de uso no programa que desenvolvemos, entre outros.

Para mostrar que já temos um `servlet` de erro em `ErrorController.java`, após seu recebimento precisaremos enviá-lo à tela do usuário. Idealmente, teremos que lançar a exceção até o `servlet` para que se possa capturar o erro, jogando-o (`throw`) de volta para o usuário, para que ele saiba o que está acontecendo. Então, o código fica assim:

```
public void salvar(Convidado convidado){

  try {
    //...

  } catch (SQLException e) {
    e.printStackTrace();
    throw new SQLException();
  }

}
```

Com o mouse sobre `throw`, clicaremos na opção "*Add throws declaration*", com isto lançaremos a exceção a este método, em `ConvidadoRepository.java`. O arquivo `ConvidadoService.java` tratará o erro na hora de salvar, ou então lançá-lo. Com esta aba aberta, e o mouse em cima de `ConvidadoRepository()`, selecionaremos a opção "*Add throws declaration*" de novo, adicionando a exceção à declaração.

O próximo método que capturar tratará a exceção. Vamos salvar e abrir a aba de `ConvidadoServlet.java`. Trataremos a exceção acrescentando-se um "*Try/Catch*", mantendo o mouse sobre `convidadoService.salvar(novoConvidado);` e selecionando "*Surround with try/catch*".

Além de printarmos o erro, mostraremos que a exceção foi lançada, chegando ao `servlet` de erro, ou seja, lançaremos uma exceção de outra maneira, por meio de `throw new ServletException("Deu erro, e o erro é " + e);` na aba `ConvidadoServlet.java`:

ConvidadoServlet.java | ConvidadoService.java | ConvidadoRepository.java | ErrorController.java

```java
import org.apache.commons.mail.Email;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.SimpleEmail;

@WebServlet("/convidado")
public class ConvidadoServlet extends HttpServlet {

    private static final long serialVersionUID = -2949873189744763662L;

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

        String nome = req.getParameter("nome");
        String email = req.getParameter("email");
        String telefone = req.getParameter("telefone");

        Convidado novoConvidado = new Convidado(nome, email, telefone);

        ConvidadoService convidadoService = new ConvidadoService();
        try {
            convidadoService.salvar(novoConvidado);
        } catch (SQLException e) {
            throw new ServletException("Deu erro, e o erro é: " + e);
        }

        List<Convidado> convidados = new ConvidadoService().obterTodos();

        req.setAttribute("convidados", convidados);
        req.getRequestDispatcher("/lista.jsp").forward(req, resp);
```

Agora sim, já temos a exceção a ser capturada e mostrada ao usuário. Quando houverem exceções, mapeamos ao `servlet` de erro a partir de `web.xml` em "listavipServlet > WebContent > WEB-INF > `web.xml` ", à qual incluiremos uma página de erro, com duas tags, `<exception-type>` e `<location>` :

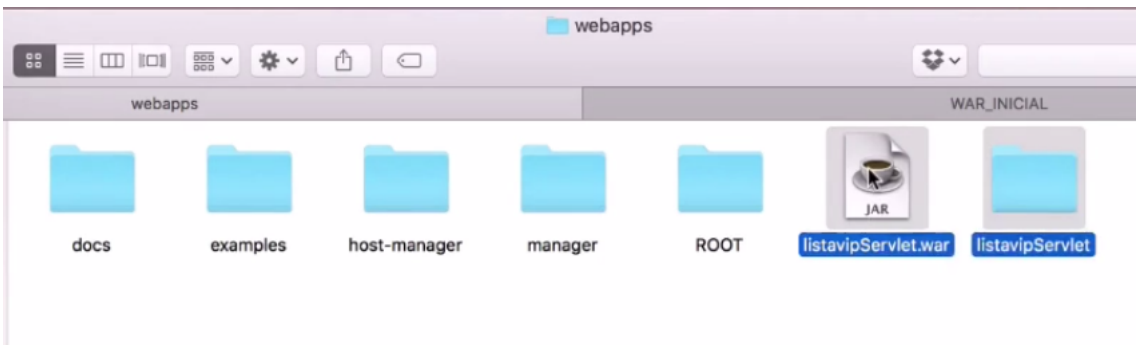
```
<error-page>
  <exception-type>java.lang.Exception</exception-type>
  <location>/error</location>
</error-page>
```

Salvaremos estas alterações, e agora todas as exceções serão lançadas ao `servlet` . Ao alterarmos o código, podemos salvá-lo, e a alteração é refletida no Tomcat. Porém, como modificamos o `web.xml` , faremos um *deploy* novo.

Para isto, precisaremos gerar o `.war` , artefato a ser colocado no Tomcat, para realizarmos um *deploy* em produção. Vamos clicar com o lado direito do mouse na pasta `listavipServlet` , exibido ao lado esquerdo da página, selecionando "*Export* > *WAR file*". Na janela que se abriu, criaremos uma pasta nova de destino, a qual denominaremos "novo_deploy", clicando em "*Finish*" depois.

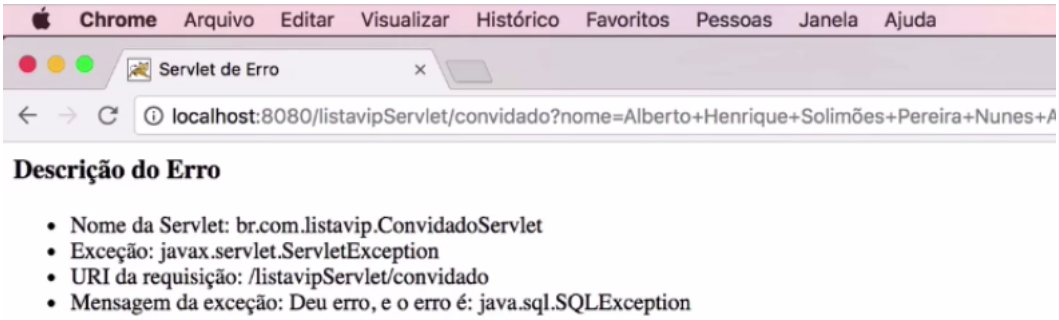
Iremos ao Tomcat, pausaremos o servidor, limparemos, colocaremos o `.war` novo, após o qual subiremos o servidor novamente. No terminal, nos encontramos na pasta "logs". Subiremos um nível e entraremos na pasta "bin" e, com o comando `./shutdown.sh` , derrubamos o servidor.

Voltando ao "Finder" (Arquivos), selecionaremos os arquivos abaixo, deletando-os em seguida:



Acessando-se "Mesa > projetoWeb > novo_deploy", encontramos o arquivo `listavipServlet.war` atualizado. Copiaremos e colaremos em "webapps". Vamos subir o servidor novamente utilizando o JPDA, como visto anteriormente. No terminal, digitaremos `./catalina.sh jpda start` . Verificamos estas alterações no *browser*, abrindo a url localhost:8080/listavipServlet.

O link está funcionando corretamente, então vamos tentar simular o cadastro novamente para verificar se desta vez a exceção é lançada ao usuário. Ao completarmos o formulário e clicarmos em "Convidar", somos redirecionados a esta página de erro:



Trata-se de uma tela de erro em desenvolvimento, sendo que não é recomendável colocar este tipo de informação à disposição, pois pode ocorrer de ser um erro de segurança, por exemplo, informação que é melhor evitar deixar vulnerável.

Caso você queira que outra página de erro seja mostrada, basta ir à aba `ErrorController.java` e fazer outro redirecionamento, ou alterar o texto referente, o que achar melhor, apenas tomando cuidado com questões de segurança do site - existem cursos sobre isto na Alura, a partir dos quais é possível obter maiores informações sobre o assunto.

Voltando ao Eclipse, podemos arrumar o erro tendo em vista as seguintes opções: colocarmos um `limit ao input` para que não se ultrapasse os `50` caracteres disponíveis, ou, caso tenhamos acesso ao banco de dados como neste caso (algo que não ocorre sempre, uma vez que nem sempre a DBA permite este acesso), podemos aumentar a coluna "nome" para abrigar mais de `50` caracteres.

Inicialmente, incluiremos `maxlength` no `input` , abrindo-se a pasta `lista.jsp` e buscando esta tag, em que definiremos o limite de caracteres:

Lembrando que em "mysql", a coluna "nome" possui "varchar" de `50` caracteres, digitaremos a linha abaixo no terminal:

```
ALTER TABLE `listavip`. `convidado` CHANGE COLUMN `nome` `nome` VARCHAR(200) NULL DEFAULT NULL
```

Após o qual rodaremos `desc convidado;` digitando-o no terminal, e verificamos que o "varchar" realmente passou de `50` para `200`. Como alteramos o arquivo `.jsp`, precisaremos refazer o *deploy* no Tomcat, repetindo os procedimentos feitos anteriormente: clicando na pasta do projeto com o lado direito do mouse, selecionaremos "*Export* > *WAR file*", tendo como destino a mesma pasta `novo_deploy`, pois sobrescreveremos os arquivos.

O `.war` novo foi gerado, pausaremos o Tomcat, copiaremos e colaremos este arquivo na pasta "webapps", substituindo as versões anteriores, subindo novamente o servidor (por meio de `./catalina.sh jpda start`). Vamos ao navegador verificar a alteração feita. Na raiz, apertaremos o botão "Clique aqui para ver a lista de convidados", colocando os dados do usuário no formulário.

O cadastro foi feito com sucesso, e os dados aparecem na lista em cima do formulário. Resolvemos o problema! No entanto, para além de consertarmos bugs, vamos tomar cuidado com boas práticas para sermos cada vez melhores programadores, o que inclui atenção ao suporte ao cliente, à experiência de uso no programa que desenvolvemos, entre outros.