Segurança e Controle de erros

Helder da Rocha (helder@acm.org)
www.argonavis.com.br

Assuntos abordados

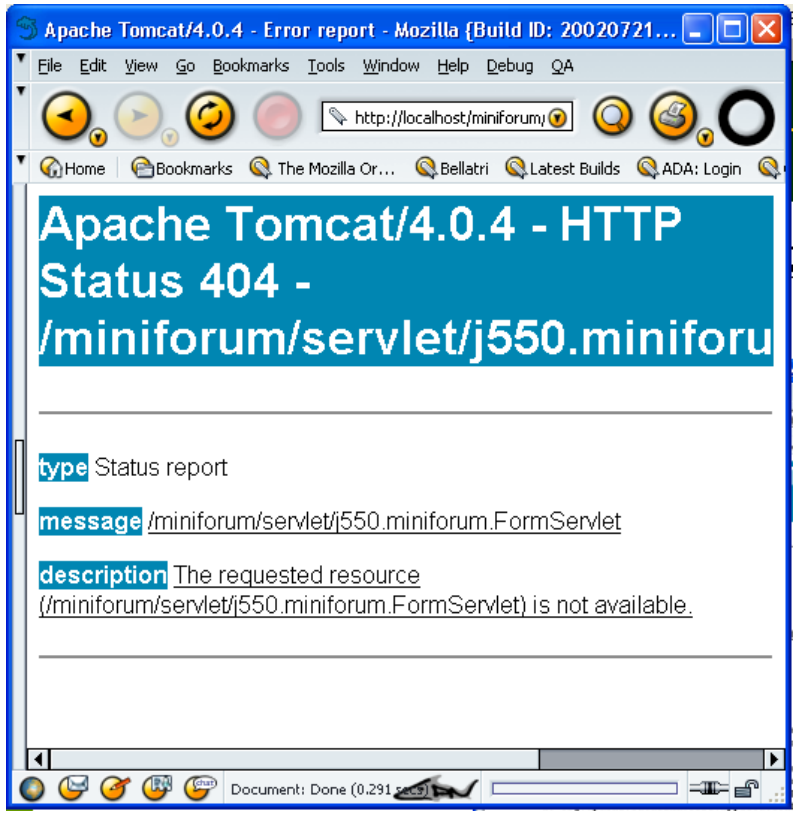
- Este módulo trata de dois assuntos
  - Como mapear erros HTTP e exceções Java a servlets ou páginas HTML e JSP, criando respostas personalizadas e servlets que possam lidar com situações de erro
  - Como implementar autenticação e autorização baseada em perfis do usuário usando servlets, permitindo que determinados recursos fiquem protegidos por senha e com acesso limitado a usuários que pertençam a determinado perfil

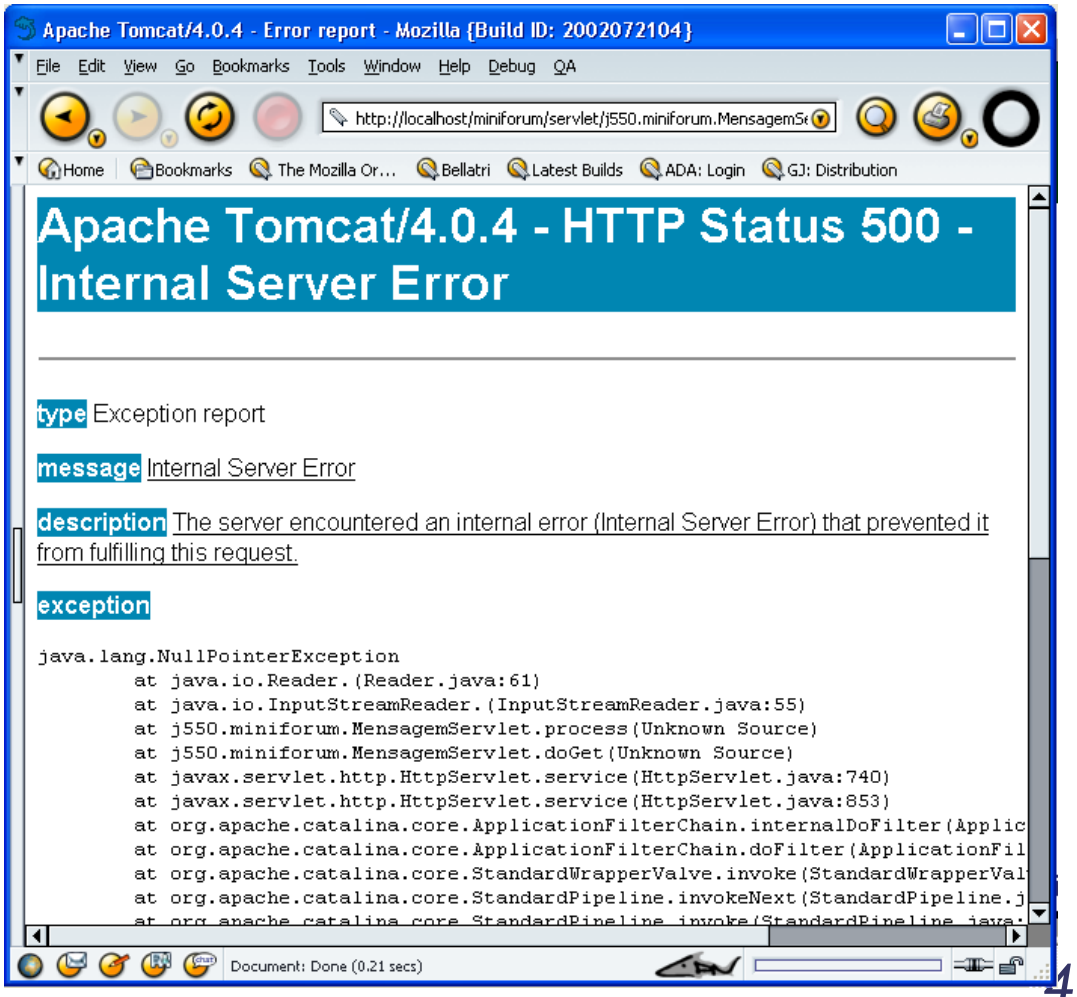
Erros

- Dois tipos de erros podem ocorrer em servlets
  - Erros HTTP
  - Exceções
- Erros HTTP são identificados pelo servidor através de um código de status
  - Códigos de status que começam com 1, 2 ou 3 não indicam erro
  - Códigos que começam com 4 (404, 401, etc.) indicam erro originado no cliente (que, em tese, poderia ser corrigido pelo browser - ex: desviar para outra página)
  - Códigos que começam com 5 (500, 501, etc.) indicam erro no servidor (não adianta o browser tentar evitá-lo)

Comportamento default

- *Quando acontece um erro do servidor, causado ou não por uma exceção, o servidor mostra uma página default*





Páginas de erro

- Configurando-se o web.xml, pode-se estabelecer páginas de erro customizadas para cada código de erro HTTP e cada exceção ocorrida
- O exemplo abaixo usa <error-page> para informar uma página HTML que irá tratar erros 404 e um servlet que irá receber FileNotFoundException

```
<error-page>
  <error-code>404</error-code>
  <location>/notFound.html</location>
</error-page>

<error-page>
  <exception-type>java.io.FileNotFoundException</exception-type>
  <location>/servlet/j550.error.IOExceptionServlet</location>
</error-page>
```

Pode haver qualquer número de elementos <error-page> mas apenas um para cada tipo de exceção específica ou código de erro HTTP

Se você define <error-page> de exceções não defina um <error-page> para o erro HTTP 500: este é o código para todas as exceções!

Atributos especiais

- *Destinos de erro recebem dois atributos na requisição que podem ser lidas se forem páginas geradas dinamicamente (servlet ou JSP) para melhor exibir informações de erro*
  - *Nome: javax.servlet.error.request_uri. Tipo: String*
  - *Nome: javax.servlet.error.exception. Tipo: Throwable*
- *A exceção recebida corresponde à exceção embutida dentro de ServletException (a causa)*

```
public void doGet(... request, ... response) {
  Throwable exception = (Throwable)
    request.getAttribute("javax.servlet.error.exception");
  String urlCausadora = (String)
    request.getAttribute("javax.servlet.error.request_uri");

  // Faça alguma coisa com os dados recebidos

}
```

Recursos de segurança

- A especificação Servlet 2.3 permite a configuração de segurança em dois domínios
  - Autenticação: o processo de verificação da identidade do usuário que solicita um recurso - quem é?
  - Autorização: processo de verificação das permissões que um usuário autenticado possui - o que ele pode fazer?
- Os dois recursos podem ser configurados para cada contexto no web.xml definindo
  - Login Configuration: configuração de métodos de autenticação utilizados
  - Security Roles: perfis de usuário para autorização
  - Security Constraint: URLs e métodos de acesso permitidos e perfis de segurança necessários para acessá-los

Configuração de Login

- *Escolha uma dentre quatro técnicas de autenticação*
  - *BASIC: mostra uma janela do browser que recebe nome e senha. Os dados são enviados em conexão insegura*
  - *FORM: igual a BASIC mas em vez de usar uma janela do browser, pemite o uso de um formulário HTML*
  - *DIGEST: usa criptografia fraca para enviar os dados*
  - *CLIENT-CERT: requer certificado X-509 para funcionar e usa criptografia forte (128-bit)*
- *BASIC, FORM e DIGEST são seguros se usados com SSL*

```
<login-config>
  <auth-method>
    BASIC
  </auth-method>
</login-config>
```

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/form.html</form-login-page>
    <form-error-page>/erro.html</form-error-page>
  </form-login-config>
</login-config>
```

Autorização: perfis de usuário

- *Uma vez definida a forma de autenticação, é preciso definir perfis de usuário habilitados a acessar os recursos*
- *A declaração de perfis de usuários é feita no web.xml usando <security-role>. A associação desses perfis com usuários reais é dependente de servidor*

```
<security-role>
   <role-name>administrador</role-name>
</security-role>

<security-role>
   <role-name>membro</role-name>
</security-role>

<security-role>
   <role-name>visitante</role-name>
</security-role>
```

Associação de perfis com usuário no Tomcat

- *Para definir domínios de segurança* ***no Tomcat*** *veja a documentação do servidor sobre security realms: http://localhost:8080/tomcat-docs/realm-howto.html*
  - *Há três níveis diferentes de configuração. Um grava os dados em banco relacional (JDBC Realm), outro em LDAP via JNDI (JNDI Realm) e o mais simples usa um par de arquivos (Memory Realm)*
- *Para usar Memory Realm, localize o arquivo tomcat-users.xml no diretório conf/ em* ***$TOMCAT_HOME*** *e acrescente os usuários, senhas e perfis desejados*

```
<tomcat-users>
  <user name="einstein" password="e=mc2" roles="visitante" />
  <user name="caesar"   password="rubicon"
                        roles="administrador, membro"  />
  <user name="brutus"   password="tomcat" roles="membro" />
</tomcat-users>
```

Web-Resource Collection

- *A coleção de recursos Web protegidos e os métodos de acesso que podem ser usados para acessá-los é definido em um bloco <web-resource-collection> definido dentro de <security-constraint>*
  - *Inclui URL-patterns <url-pattern> que indicam quais os mapeamentos que abrangem a coleção*
  - *Inclui um <http-method> para cada método permitido*

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Seção de Assinantes</web-resource-name>
    <url-pattern>/membros/*</url-pattern>
    <url-pattern>/preferencias/config.jsp</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>
  ...
</security-constraint>
```

Security Constraint

- *O Web Resource Collection faz parte do Security Constraint que associa perfis de usuário (papéis lógicos) à coleção*

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>Seção de Assinantes</web-resource-name>
    <url-pattern>/membros/*</url-pattern>
    <url-pattern>/preferencias/config.jsp</url-pattern>
    <http-method>GET</http-method>
    <http-method>POST</http-method>
  </web-resource-collection>

  <auth-constraint>
    <role-name>administrador</role-name>
  </auth-constraint>
</security-constraint>
...
<security-constraint>
        ... outras coleções e constraint ...
</security-constraint>
```

Form-based Login

- *Para autenticação por formulário, é preciso definir no <login-config> FORM como <auth-method>*
- *Adicionalmente, é informada a página que implementa o formulário. Esta página deve conter um elemento <FORM> HTML com algumas restrições*
  - *Atributo ACTION de <FORM> deve conter: j_security_check*
  - *Campo <INPUT> do nome deve conter: j_username*
  - *Campo <INPUT> da senha deve conter: j_password*

```
<form action="j_security_check" method="POST">
  <p>Digite seu nome de usuário:
  <input type="text" name="j_username">
  <p>Digite sua senha:
  <input type="password" name="j_password">
  <input type="submit">
</form>
```

Exercícios

- 1. Mapeamentos de erro: crie uma página especial para erros 404 na sua aplicação.
- 2. Mapeamentos de exceção: redirecione qualquer Exception para uma página especial que imprima a classe da exceção ocorrida
  - Use getClass() para obter o nome da classe do objeto.
- 3. Configure os usuários e perfis mostrados nos exemplos no seu Tomcat e implemente um login BASIC para que
  - Apenas membros possam mandar mensagens para o forum
  - Apenas usuários visitantes registrados possam vê-las
  - Qualquer um possa ver a página inicial (index.html)
  - Administradores tenham acesso a todo o sistema
- 4. Crie um formulário de login em uma página HTML e altere o programa do exercício anterior para utilizá-lo.

Exercício

- *3. Implemente uma tela de login para a aplicação da Loja Virtual alterando LojaServlet para que peça além do e-mail, uma senha*
  - *Se o usuário for administrador, redirecione-o para AdminLojaServlet*
  - *Se usuário foi registrado, redirecione-o para ComprasServlet*
  - *Se usuário não for registrado, envie-o para página onde possa se registrar**
  - *Use BASIC ou FORM para autenticação*

** Não precisa implementar o registro. Se desejar, manipule o arquivo XML do Tomcat usando a API JAXP (javax.xml) ou use o JDBC Security Realm do Tomcat para guardar as informações em um banco de dados*

helder@acm.org

argonavis.com.br