



Tecnologias de Internet I

Aula 12

- Segurança

Prof. Rafael Garcia Barbosa

Segurança

Introdução

2

- Cada vez mais empresas estão tratando seus negócios pela Internet
- Milhares de pessoas atualmente transmitem dados importantes via Internet:
 - ▣ Comércio eletrônico
 - ▣ Home Banking
 - ▣ E-government
- A segurança de uma aplicação Web passa a ser indispensável

Segurança

Conceitos Básicos

3

- Autenticação
 - ▣ É o processo de identificação da pessoa que está acessando a aplicação
- Autorização
 - ▣ É o processo que determina após identificar um usuário, a quais recursos ele tem acesso
- Integridade de dados
 - ▣ É o processo de garantir que o dado enviado pelo cliente não é alterado antes de chegar ao servidor
- Privacidade de dados
 - ▣ É o processo de garantir que os dados enviados pelo cliente só podem ser lidos pelo servidor

Segurança

Autenticação

4

- Aplicações Servlet podem utilizar quatro mecanismos de autenticação:
 - ▣ HTTP Basic authentication
 - ▣ HTTP Digest authentication
 - ▣ HTTPS Client authentication
 - ▣ FORM-based authentication

Autenticação

HTTP Basic Authentication

5

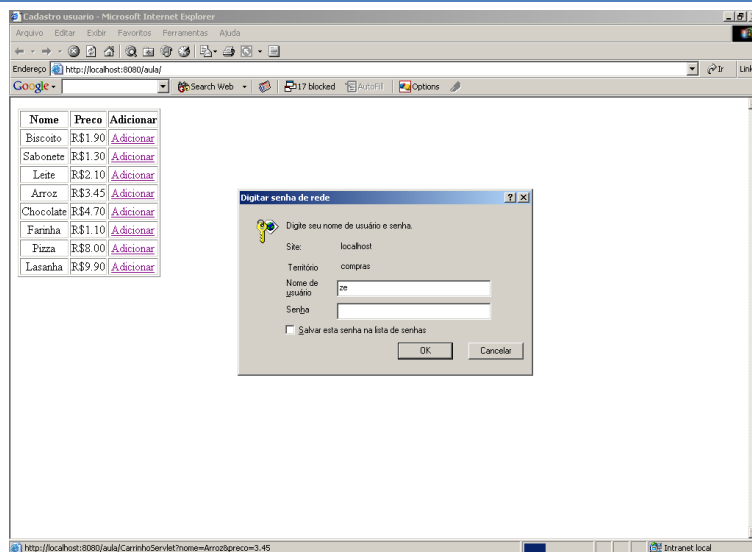
- ❑ Definido no HTTP 1.0
- ❑ Quando o cliente solicita um recurso protegido o servidor envia uma resposta com status 401 (não autorizado):

```
HTTP/1.0 401 Unauthorized
Server: Tomcat/4.0.1
WWW-Authenticate: Basic realm="compras"
```

Autenticação

HTTP Basic Authentication

6



Autenticação

HTTP Basic Authentication

7

- Os dados da autenticação são enviados em novo request
- Dados são codificados em Base64

```
GET /servlet/SalesServlet HTTP/1.0
Authorization: Basic am9objpwamo=
```

- Vantagem:
 - ▣ Fácil de utilizar
- Desvantagem:
 - ▣ Codificação dos dados fácil de quebrar
 - ▣ Não é possível customizar a tela de login

Autenticação

HTTP Basic Authentication – Configurando

8

- A configuração da autenticação é feita no web.xml da aplicação:

```
<web-app>
...
<login-config>
    <auth-method>BASIC</auth-method>
    <realm-name>compras</realm-name>
</login-config>
...
</web-app>
```

Autenticação

HTTP Digest Authentication

9

- Criado para suprir alguns dos defeitos de Basic
- Semelhante ao Basic, exceto que os dados enviados são criptografados
- Vantagens:
 - Mais seguro que o método Basic
- Desvantagens
 - Alguns Servlet containers não suportam, pois a especificação de Servlet não determina
 - Não é possível customizar a tela de login
 - Suscetível a ataques tipo “Man-in-the-middle”

Autenticação

HTTP Client Authentication

10

- Utiliza HTTPS para enviar os dados da autenticação
- HTTPS
 - HTTP sobre SSL (Secure Socket Layer)
 - HTTP transmite dados criptografados usando mecanismo de chave publica
- Vantagem:
 - O mecanismo mais seguro
- Desvantagem:
 - Necessita de um autoridade de certificação (VeriSign, Entrust ...)
 - O usuário deve possuir uma Public Key Certificate (PKC) emitido por uma autoridade de certificação

Autenticação

FORM-based Authentication

11

- Semelhante a Basic authentication, mas a tela de autenticação pode ser um formulário HTML
- O formulário deve possuir:
 - ▣ A tag `<form action= "j_security_check" >`
 - ▣ Parâmetros
 - `j_username` com o login do usuário
 - `j_password` com a senha do usuário
- Vantagem:
 - ▣ Tela de login configurável
 - ▣ Todos os browser suportam
- Desvantagens:
 - ▣ Deve ser feita conexão HTTPS para que seja segura

Autenticação

FORM-based Authentication

12

```
<html>
  <body>
    <form method="POST" action="j_security_check">
      Login: <input type="text" name="j_username">
      Senha:<input type="password" name="j_password">
      <input type="submit" value="OK">
    </form>
  </body>
</html>
```

Autenticação

FORM-based Authentication – Configurando

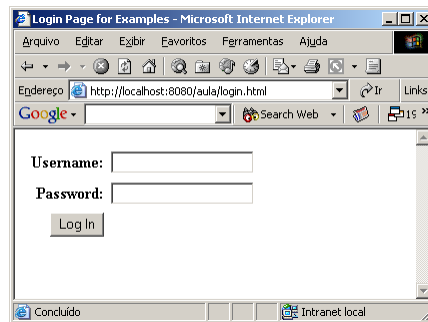
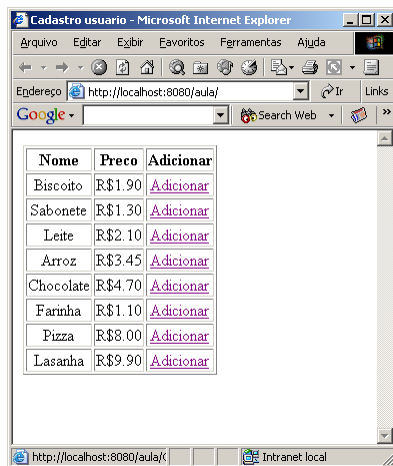
13

```
<web-app>
...
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/formlogin.html</form-login-page>
    <form-error-page>/formerror.html</form-error-page>
  </form-login-config>
</login-config>
...
</web-app>
```

Autenticação

FORM-based Authentication – Configurando

14

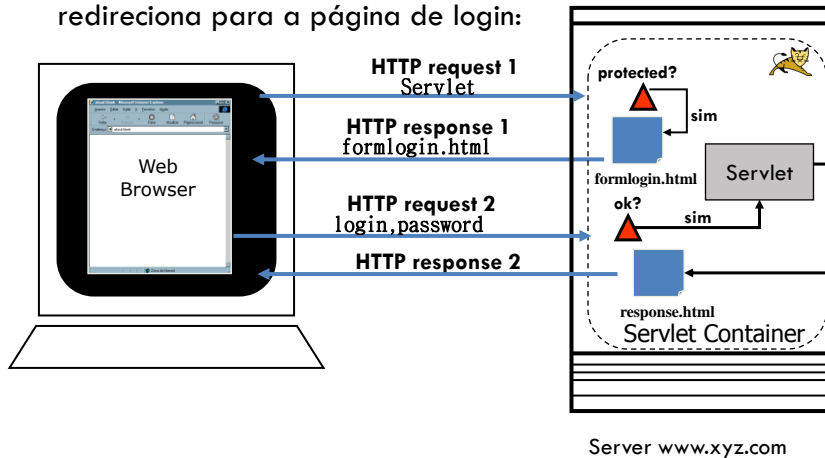


Autenticação

Processo no Container

15

- Ao receber uma requisição para um recurso protegido (Servlet, JSP, HTML ...) o container automaticamente redireciona para a página de login:



Autenticação

Via Container

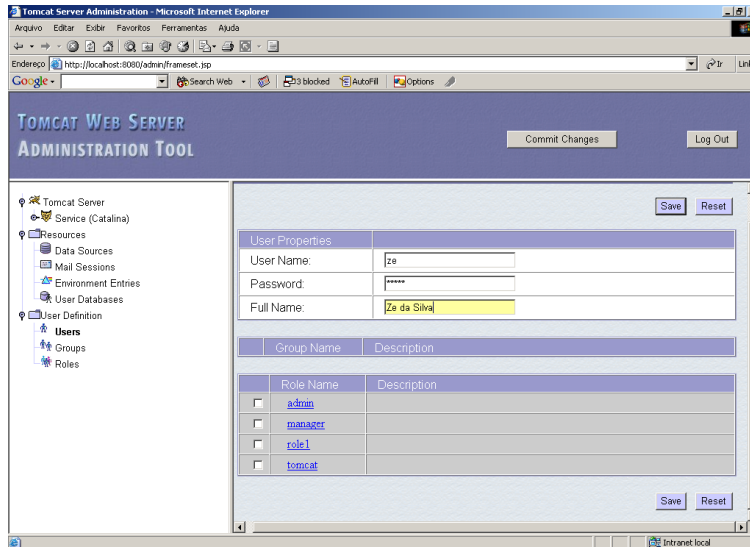
16

- Por default, a autenticação do acesso à aplicação é definida no container
- A configuração consiste em usuários (user) e papéis que os usuários podem assumir (role)
- No Tomcat isso é feito no arquivo:
 - ▣ <CATALINA_HOME>\conf\tomcat-users.xml

```
<tomcat-users>
  <role rolename="cliente"/>
  <role rolename="admin"/>
  <role rolename="manager"/>
  <user username="user" password="senha" roles="cliente"/>
  <user username="admin" password="admin" roles="admin,manager"/>
</tomcat-users>
```


Autenticação Via Container

17



Autenticação Via JDBC

18

- É possível também configurar o container para uma aplicação autenticar a partir de um BD
- Configurar o <Realm> para o contexto:
 - ▣ <CATALINA_HOME>\conf\server.xml
 - ▣ Ou então o server.xml criado pelo Eclipse

```
<Context path="/aula" docBase="aula" debug="0" privileged="true">
  <Realm className="org.apache.catalina.realm.JDBCRealm"
    driverName="com.mysql.jdbc.Driver"
    connectionURL="jdbc:mysql://localhost/test"
    connectionName="root" connectionPassword="root"
    userTable="users" userRoleTable="user_roles"
    userNameCol="nome" userCredCol="senha" roleNameCol="role" />
</Context>
```

Autenticação

Via JDBC

19

- Para o container conectar ao BD o Driver JDBC usado deve ser copiado para:
 - ▣ <CATALINA_HOME>\common\lib
- As duas tabelas indicadas no <realm> devem ser criada para fazer a associação:

USUÁRIO → SENHA		USUÁRIO → PAPEL	
nome	senha	nome	role
ze	zeze	ze	cliente
maria	mama	maria	cliente
pedro	pepe	pedro	gerente

Autenticação

Verificando a Autenticação

20

- É possível ainda verificar em uma aplicação que usuário está logado
- Para isso pode-se usar os métodos da classe `HttpServletRequest` :
 - ▣ `String getRemoteUser()`
 - Retorna o login do usuário que está logado na aplicação e null se não estiver logado
 - ▣ `boolean isUserInRole(String rolename)`
 - retorna true se o usuário logado pertencer ao papel indicado no argumento

Autenticação

Verificando a Autenticação

21

```
public void doGet (HttpServletRequest request,
                  HttpServletResponse response)
    throws ServletException, IOException {
    PrintWriter out;
    response.setContentType("text/html");
    String user = request.getRemoteUser();
    out = response.getWriter();
    if(user != null){
        out.println("<H1> Bem vindo "+ user +"!! </H1>");
    } else {
        out.println("<H1> Bem vindo !! </H1>");
    }
    ...
}
```

Autorização

Introdução

22

- A autorização define que usuários tem acesso a cada recurso da aplicação
- Por default, todos os recursos de uma aplicação estão autorizadas para qualquer usuário
- Para restringir o acesso a parte da aplicação é preciso definir:
 - ▣ Coleção de recursos protegidos
 - ▣ Grupos de usuários autorizados
 - ▣ Forma com que os dados serão transmitidos

Autorização

Uso

23

- A definição da autorização é feita no arquivo web.xml da aplicação.
- A tag `<security-constraint>` define:
 - ▣ `<web-resource-collection>`: Define a coleção de recursos protegidos
 - ▣ `<auth-constraint>`: Define grupos de usuários autorizados
 - ▣ `<user-data-constraint>`: Define como serão transmitidos os dados

Autorização

A Tag `<web-resource-collection>`

24

```

<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>produtos</web-resource-name>
    <url-pattern>/compras/*</url-pattern>
  </web-resource-collection>
...
</security-constraint>
...
</web-app>

```

Nome pelo qual será referenciada a coleção de recursos

Padrões de URL que identificam os recursos

Autorização

A Tag <auth-constraint>

25

```

<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>produtos</web-resource-name>
    <url-pattern>/compras/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>cliente</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>cliente</role-name>
</security-role>
...

```

Define papéis que terão acesso à coleção de recursos declarados

Declaração dos papéis utilizados pela aplicação

Autorização

A Tag <user-data-constraint>

26

```

<web-app>
...
<security-constraint>
  <web-resource-collection>
    <web-resource-name>produtos</web-resource-name>
    <url-pattern>/compras/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>cliente</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
...

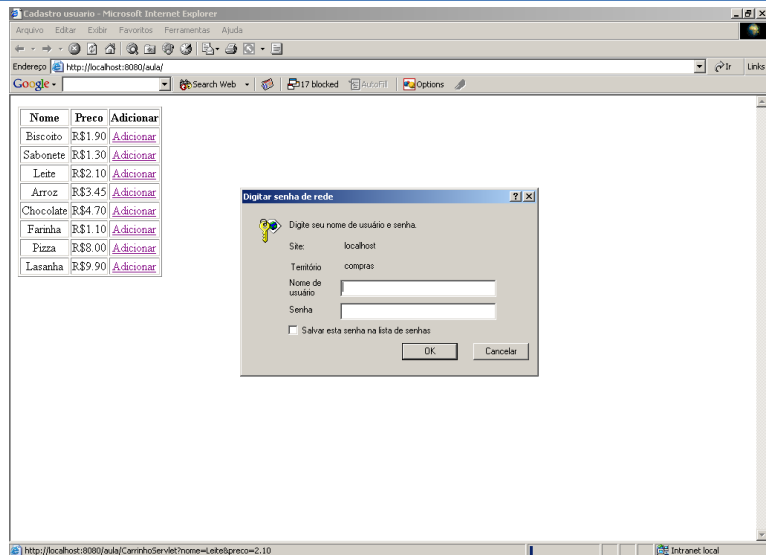
```

Define como esses recursos serão transmitidos: NONE, INTEGRAL ou CONFIDENTIAL

Autorização

Exemplo

27



HTTPS

Introdução

28

- ❑ HTTP sobre SSL (Secure Socket Layer)
- ❑ HTTP transmite dados criptografados usando mecanismo de chave publica
- ❑ É preciso gerar as chaves de criptografia armazenadas em um arquivo .keystore
- ❑ Será usada para transmitir parte da aplicação cujo transporte foi definido como INTEGRAL ou COFIDENTIAL

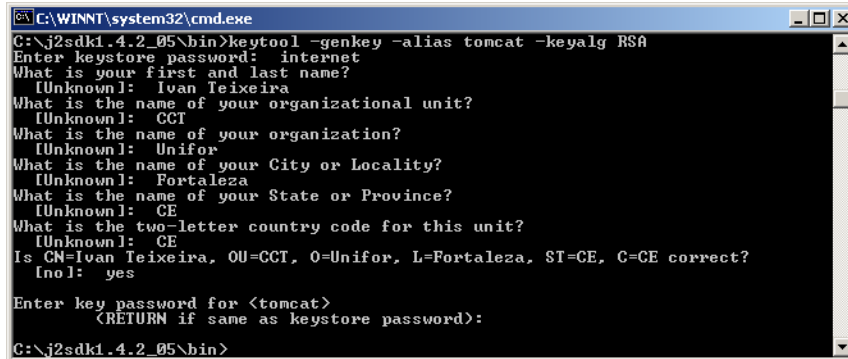
HTTPS

Gerando Chaves

29

- Gerando as chaves com o JDK:

`<JAVA_HOME>\bin\keytool -genkey -alias tomcat -keyalg RSA`



```
C:\WINNT\system32\cmd.exe
C:\j2sdk1.4.2_05\bin>keytool -genkey -alias tomcat -keyalg RSA
Enter keystore password: internet
What is your first and last name?
[Unknown]: Ivan Teixeira
What is the name of your organizational unit?
[Unknown]: CCT
What is the name of your organization?
[Unknown]: Unifor
What is the name of your City or Locality?
[Unknown]: Fortaleza
What is the name of your State or Province?
[Unknown]: CE
What is the two-letter country code for this unit?
[Unknown]: CE
Is CN=Ivan Teixeira, OU=CCT, O=Unifor, L=Fortaleza, ST=CE, C=CE correct?
[no]: yes
Enter key password for <tomcat>
<RETURN if same as keystore password>:
C:\j2sdk1.4.2_05\bin>
```

HTTPS

Usando Container

30

- Para habilitar o container a utilizar o protocolo HTTPS configure na tag `<Service>` no arquivo `server.xml`

```
<Service ... >
<!-- Define a SSL Coyote HTTP/1.1 Connector on port 8443 -->
  <Connector port="8443"
    maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
    enableLookups="false" disableUploadTimeout="true"
    acceptCount="100" debug="0" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS" keystorePass="internet"/>
```

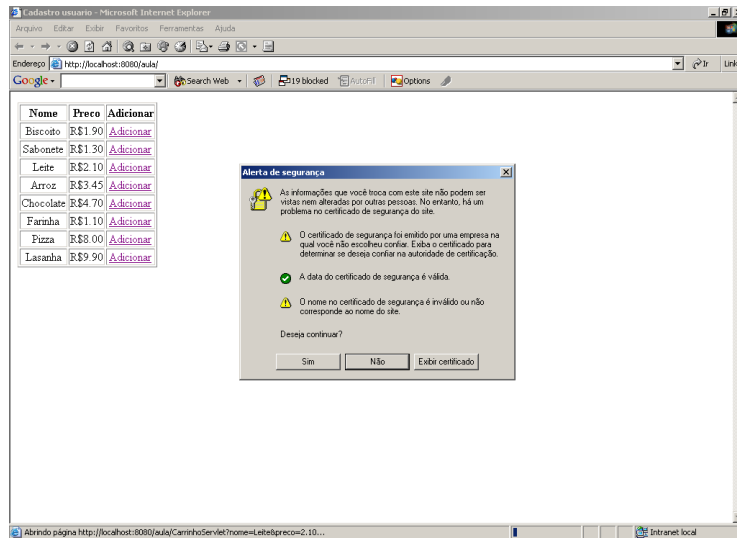
...

Senha digitada ao criar a chave

HTTPS

Exemplo

31



HTTPS

Exemplo

32

- Por último, para evitar um ataques do tipo “Man-in-the-middle”, obtenha uma Certificado Digital para a sua chave publica junto à uma Autoridade de Certificação

