



# Fundamentos básicos de Segurança

*Helder da Rocha*  
[www.argonavis.com.br](http://www.argonavis.com.br)

- *Este capítulo apresenta*
  - *Noções básicas de segurança em J2EE*
  - *Exemplos de aplicações*
  - *Usos típicos de JAAS*
  - *Configuração básica de aplicações J2EE com recursos de segurança*
- *A abordagem deste assunto neste curso é superficial. Para maiores detalhes consulte*
  - *Livros-texto (MEJB contém versão original do exemplo mostrado usando outro modelo de autenticação; J2EE Tutorial mostra exemplo usando container do J2EE-RI)*
  - *Código-exemplo (comentado)*

# Controle de acesso

- J2EE define uma especificação para controle de acesso e autenticação de clientes
  - Identidade é representada por objeto `java.security.Principal`
  - Não há uma recomendação formal para outros aspectos importantes de segurança, tais como criptografia e comunicação segura
- O controle de acesso em aplicações J2EE é feito de forma declarativa nos arquivos `web.xml` e `ejb-jar.xml`
  - Declara-se papéis (roles) lógicos que podem ser mapeados a usuários e grupos em tempo de deployment: `<security-role>`
  - Papéis são associados a métodos para determinar quem tem autorização para executar: `<method-permission>`
  - Métodos podem ser executados em outros beans por um Principal diferente daquele que executa o bean atual: `<security-identity>`

# Exemplo

```
<assembly-descriptor>
  <security-role>
    <role-name>BankCustomer</role-name>
  </security-role>
  <security-role>
    <role-name>BankAdmin</role-name>
  </security-role>
  <method-permission>
    <unchecked />
    <method>
      <ejb-name>CustomerEJB</ejb-name>
      <method-name>getPrimaryKey</method-name>
    </method>
  </method-permission>
  <method-permission>
    <role-name>BankAdmin</role-name>
    <method>
      <ejb-name>CustomerEJB</ejb-name>
      <method-name>*</method-name>
    </method>
  </method-permission>
</assembly-descriptor>
```

ejb-jar.xml

← Outros métodos de CustomerEJB  
só podem ser chamados por  
usuários que assumem o papel  
de BankAdmin

- *Java Authentication and Authorization Service*
  - *Conjunto de pacotes para autenticação de usuários e autorização*
  - *Maior parte implementada pelo servidor*
- *Principais classes*
  - *javax.security.auth.**Subject***
  - *javax.security.**Principal***
  - *javax.security.auth.callback.**Callback***
  - *javax.security.auth.callback.**CallbackHandler***
  - *javax.security.auth.login.**Configuration***
  - *javax.security.auth.login.**LoginContext***
  - *javax.security.auth.spi.**LoginModule***

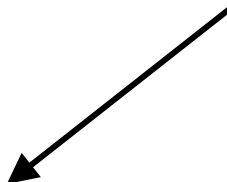
# Implementação do JAAS no JBoss (JBossSX)

- A implementação do JAAS no JBoss consiste de
  - *JAASecurityManager*
  - Módulos de configuração do servidor (*login-config.xml*) e do cliente (*auth.conf*)
  - Implementações de *javax.security.auth.spi.LoginModule*
- *ClientLoginModule*
  - Implementação de *LoginModule* no cliente
- *UsersRolesLoginModule*
  - Uma das implementações de *LoginModule* no servidor
  - Par de arquivos de texto para definir usuários e grupos
- *jboss.xml* e *jboss-web.xml*
  - Declaram domínio JAAS a ser usado através do elemento `<security-domain>` de *jboss.xml* e *jboss-web.xml*

# Exemplo: ejb-jar.xml

```
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>SecureHelloEJB</ejb-name>
      <home>examples.HelloHome</home>
      <remote>examples.Hello</remote>
      <ejb-class>examples.HelloBean</ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
    </session>
  </enterprise-beans>
  <assembly-descriptor>
    <security-role>
      <role-name>TestRole</role-name>
    </security-role>
    <method-permission>
      <role-name>TestRole</role-name>
      <method>
        <ejb-name>SecureHelloEJB</ejb-name>
        <method-name>*</method-name>
      </method>
    </method-permission>
  </assembly-descriptor>
</ejb-jar>
```

*Todos os métodos  
acessíveis a TestRole*



*Veja código exemplo em cap15/src*

# Serviço JBossSX com UsersRolesLoginModule

- Módulo simples que utiliza par de arquivos de texto
  - É a configuração default em login-config.xml

```
<policy>  (...)
  <application-policy name="other">
    <authentication>
      <login-module
        code="org.jboss.security.auth.spi.UsersRolesLoginModule"
        flag="required" />
    </authentication>
  </application-policy>
</policy>
```

Trecho de server/default/conf/login-config.xml

users.properties (nome=senha)

```
mickey=mouse
niquel=nausea
```

Use preferencialmente  
senha criptografada!

roles.properties (nome.Grupo=Role,...,Role)

```
mickey.Roles=TestRole, AdminRole
niquel.Roles=NoRole, MouseRole
```

Coloque no CLASSPATH  
do servidor (pode ser no EJB-JAR)



# Domínio de segurança (servidor)

- Deve ser declarado em *jboss.xml* (ou *jboss-web.xml*)

```
<jboss>
  <security-domain>
    java:/jaas/TestServerDomain</security-domain>
  <session>
    <ejb-name>SecureHelloEJB</ejb-name>
    <jndi-name>login/HelloHome</jndi-name>
  </session>
</jboss>
```

*jboss.xml*

- Deve coincidir com nome de um `<application-policy>` em *login-config.xml*

```
<policy> (...) <application-policy name="TestServerDomain">
  <authentication>
    <login-module code="MyLoginModule" .../>
```

- Se não houver domínio com mesmo nome em *login-config.xml*, será usado o domínio default "other", previamente configurado para usar *UsersRolesLoginModule* (veja slide anterior)

- Coloque no Classpath do cliente
  - *arquivo de configuração* (indica no mínimo a implementação de login module usada)

arquivo de configuração (default)

```
other {  
    // JBoss LoginModule implementation  
    org.jboss.security.ClientLoginModule required;  
};
```

- JARs do JBossSX: *jbossx-client.jar* (além dos outros JARs necessários para o cliente JBoss)

# Cliente

```
public class HelloClient {
    public static void main(String[] args) throws Exception {
        LoginContext loginCtx = null;
        try {
            // Cria CallbackHandler
            CallbackHandler handler = new HelloCallbackHandler();
            // Carrega configuração
            loginCtx = new LoginContext("other", handler);
            // Faz o login
            loginCtx.login();
        } catch (LoginException e) {
            System.out.println("Login failed"); System.exit(1);
        }
        // Executa ação privilegiada propagando contexto de segurança
        Context ctx = new InitialContext(System.getProperties());
        Object obj = ctx.lookup("SecureHelloEJB");
        HelloHome home = (HelloHome)
            PortableRemoteObject.narrow(obj, HelloHome.class);
        Hello hello = home.create();
        System.out.println(hello.hello());
    }
}
```

*Encapsula dados de autenticação*

*Contexto default do JBossSX (veja slide anterior)*

*Faz o login aqui*

*Chama métodos privilegiados*

# CallbackHandler

```
import javax.security.auth.*;
import javax.security.auth.callback.*;
import javax.security.auth.login.*;

public class HelloCallbackHandler implements CallbackHandler {
    private String username;
    private String password;

    public HelloCallbackHandler() {
        LoginDialog dialog = new LoginDialog("User Authentication Required");
        this.username = dialog.getData()[0];
        this.password = dialog.getData()[1];
    }

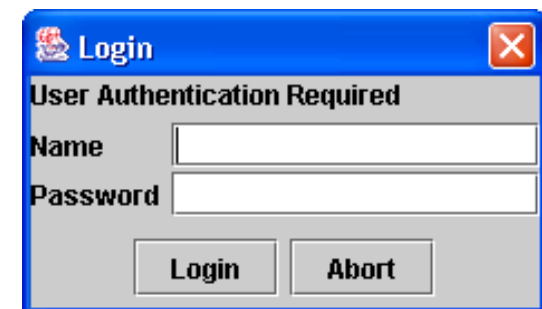
    public void handle(Callback[] callbacks)
        throws java.io.IOException, UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof NameCallback) {
                NameCallback nc = (NameCallback) callbacks[i];
                nc.setName(username);
            } else if (callbacks[i] instanceof PasswordCallback) {
                PasswordCallback pc = (PasswordCallback) callbacks[i];
                pc.setPassword( this.preparePassword(password) );
            } else throw new UnsupportedCallbackException(callbacks[i], "Error");
        }
    }

    private char[] preparePassword(String word) {
        return word.toCharArray(); // ou rotina de criptografia
    }
}
```

*GUI simples para entrada de texto*

# Como executar

- *Diretório cap15/mejb*
- *Configuração*
  - *Configure build.properties*
  - *Edite usuários e grupos em lib/users.properties e lib/roles.properties*
- *Use targets do Ant*
  - > **ant jboss.deploy** (para instalar)
  - > **ant run.jboss.client** (para rodar)
- *Digite nome e senha compatíveis com arquivos de configuração*
  - *Tente logar como usuário válido não autorizado (niquel)*



- 1. a) Crie uma aplicação contendo um session bean com dois métodos
  - *listarNomes()*
  - *adicionarNome(String nome)*
- 1. b) Crie dois papéis (roles): **User** e **Admin** e defina as permissões dos métodos
  - Admin: pode *listarNomes()* e *adicionarNome()*
  - User: pode *listarNomes()*
- 1. c) Implemente um cliente que faça o login e chame um dos dois métodos
- 1. d) Configure usuários do sistema, associe-os aos papéis existentes e rode a aplicação

Use como esqueleto os exemplos fornecidos!

- [1] JBoss Group. *JBoss User's Manual. Chapter 8: The JBoss Security Extension Framework.*
- [2] Erik Jendrock. *Security.* Sun J2EE Tutorial.
- [3] Richard Monson-Haefel, *Enterprise JavaBeans, 3rd. Edition*, O'Reilly and Associates, 2001
- [4] Ed Roman. *Mastering EJB 2.*

*helder@ibpinet.net*

***www.argonavis.com.br***