Security mechanisms in **Java SE 8**

Native security of the Java Platform

JVM "Sandbox"

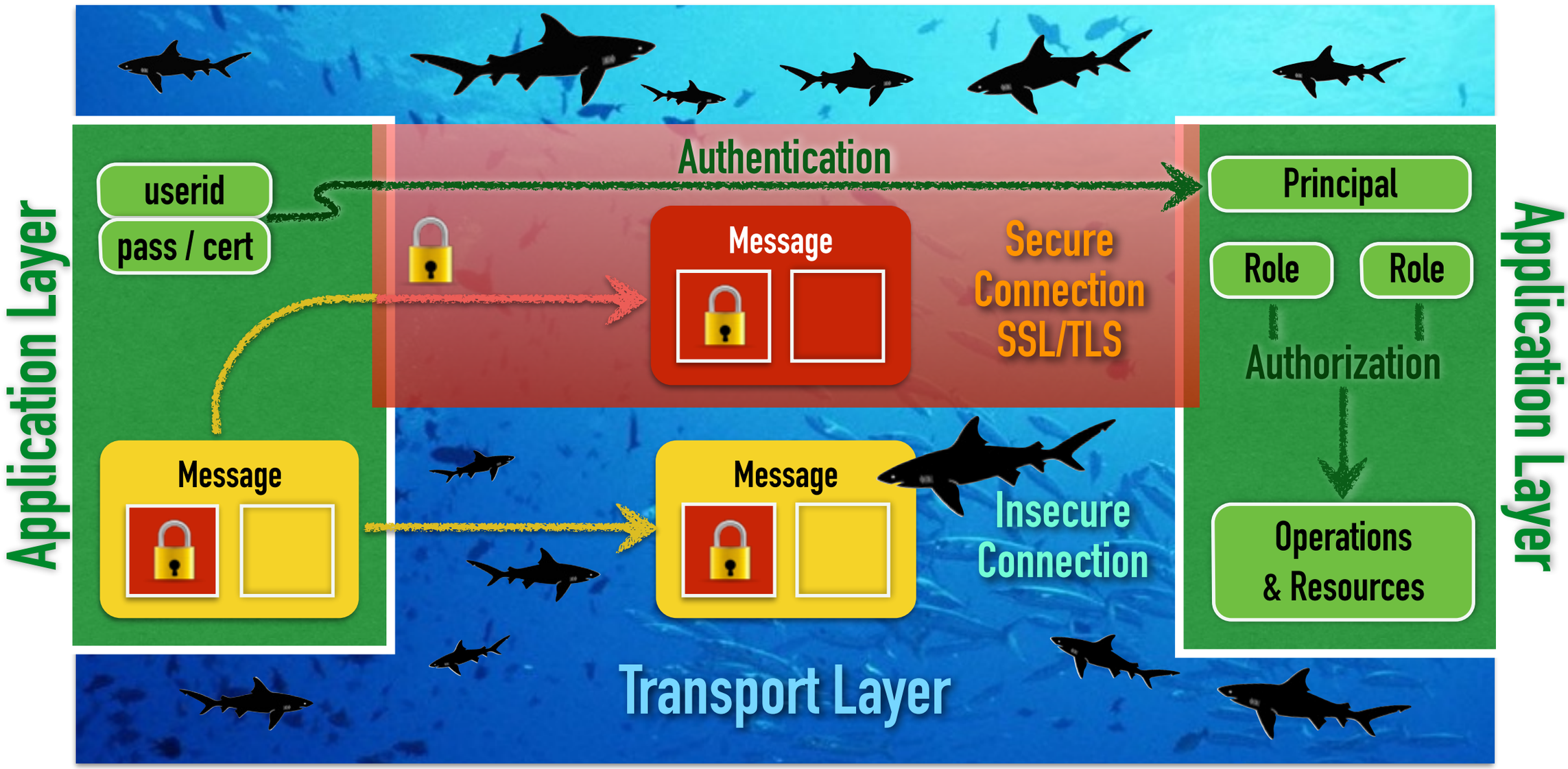
- Classloader
- Garbage collection
- Bytecode verification
- Data-typing

+

Cryptography	JCA JCE Java XML DSig (JSR 105)
Authentication & Authorization	Policy / Security Manager JAAS JarSigner TSA/TSP (RFC 3161)
Public Key Infrastructure (PKI)	X.509, CRLs & CertPath API OCSP (RFC 2560) PKCS#11
Secure Communications	JSSE (SSL/TLS) SASL (RFC 2222) GSS-API (RFC 2853)

by Helder da Rocha (www.argonavis.com.br)

Security mechanisms in **Java EE 7**



Seven ways to

get **principals** & test **roles**

in Java EE 7

WebServlets, Facelets, WebFilters

getUserPrincipal()

isUserInRole()

javax.servlet.http.
HttpServletRequest

EJBs

getCallerPrincipal()

isCallerInRole()

javax.ejb.
EJBContext

SOAP Web Services

getUserPrincipal()

isUserInRole()

javax.xml.ws.
WebServiceContext

JSF backing beans

getUserPrincipal()

isUserInRole()

javax.faces.context.
ExternalContext

CDI

@Inject
java.security.Principal

WebSockets

getUserPrincipal()

javax.websocket.
Session

RESTful Web Services

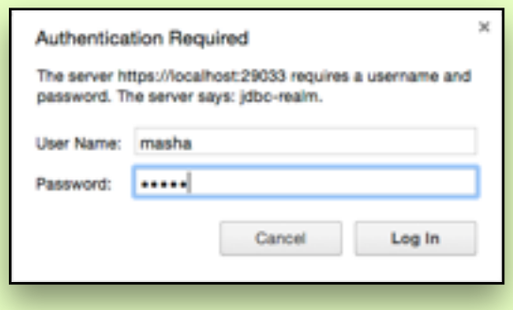
getUserPrincipal()

isUserInRole()

javax.ws.rs.core.
SecurityContext

Authentication & Authorization in the **Web** Layer

Browser



Request 1: Ask for restricted resource
GET /app/faces/biblioteca.xhtml

HTTP

Response 1: Ask for credentials
401 Unauthorized
WWW-Authenticate: Basic realm="jdbc-realm"

HTTPS

HTTPS

Request 2: Send credentials
GET /app/faces/biblioteca.xhtml
Authorization: Basic bWFzaGE6MTIzNDU=

HTTPS

Response 2: Send restricted resource
200 OK

Web Container

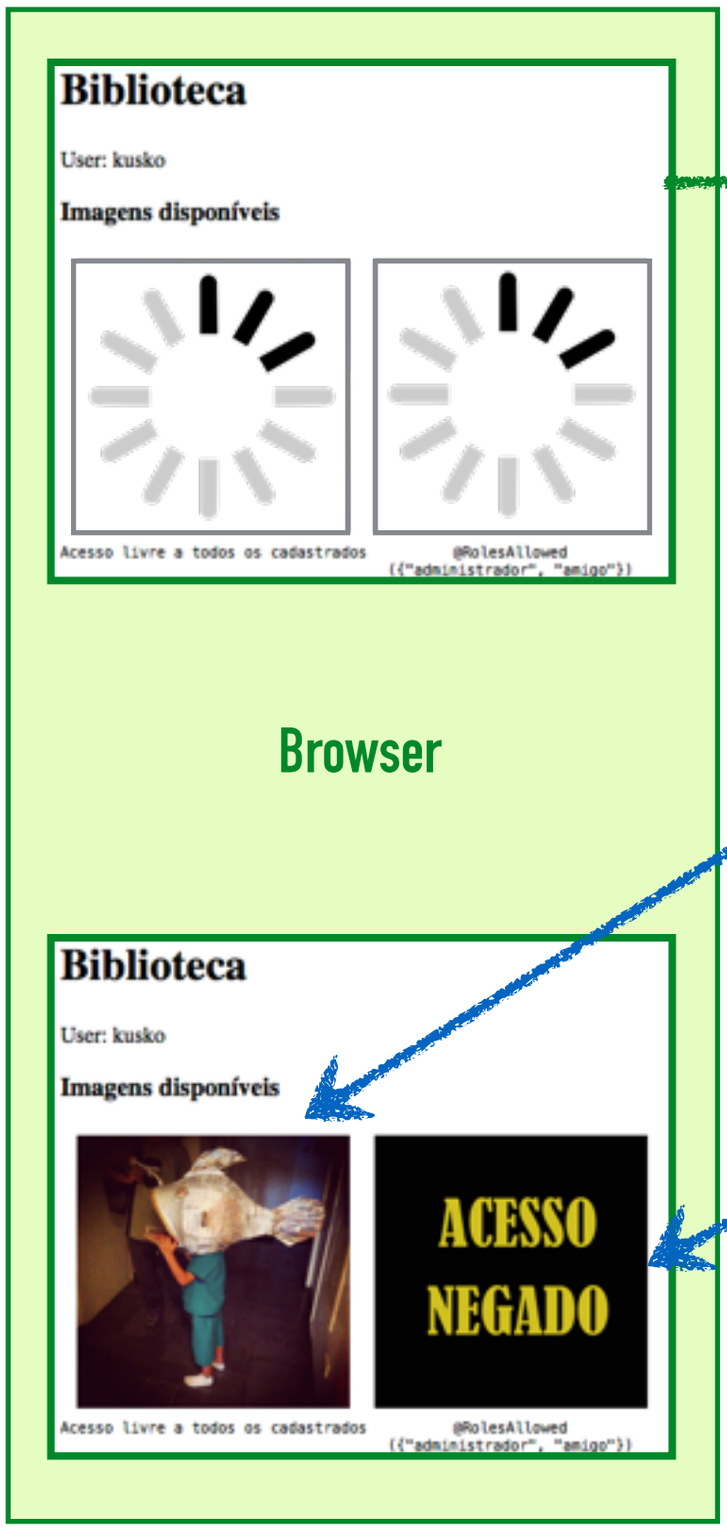
Configuration (web.xml)
- HTTPS 2-way (CONFIDENTIAL)
- Authentication: jdbc-realm
- Methods: GET, POST
- Roles: all logged in (**)

Authentication
- Do authentication
- Authenticated? Create credential
  - Go to Authorization
- Not authenticated?
  - Return 401 Unauthorized

Authorization
- Check permissions
- Authorized?
  - Return requested resource
- Not authorized?
  - Return 403 Forbidden

by Helder da Rocha (www.argonavis.com.br)

Authorization in the **EJB** Layer



Req2

Res2

Web Container

Web Context – Credentials
Principal: kusko
Roles: {"other"}

Servlet calls EJB method 1

```
try {
   b.getPicture1();
} catch (EJBAccessException e) {
   return "acessonegado.png";
}
```

Servlet calls EJB method 2

```
try {
   b.getPicture2();
} catch (EJBAccessException e) {
   return "acessonegado.png";
}
```

EJB Container

EJB Context (propagated)
Principal: kusko
Roles: {"outro"}

Authorization

```
@RolesAllowed("**")
public String getPicture1() {
   return "minifish.png";
}
```

```
@RolesAllowed("friend")
public String getPicture2() {
   return "lupita.png";
}
```

Return link

EJBAccessException

BASIC (RFC 2069 / 2617)

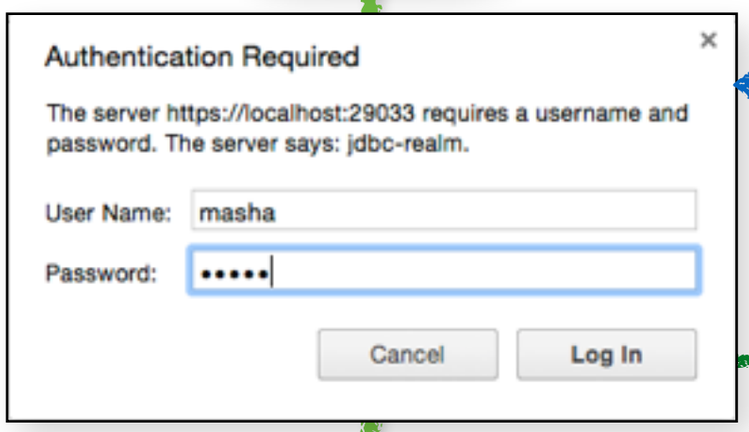
```
<login-config>
   <auth-method>BASIC</auth-method>
   <realm-name>jdbc-realm</realm-name>
</login-config>
```

configuration in **web.xml**

GET /app/secret

401 Unauthorized

WWW-Authenticate: Basic realm="jdbc-realm"



Credentials: Base64 encoding

GET /app/secret

Authorization: Basic bWFzaGE6MTIzNDU=

200 OK

To logout: exit the browser!

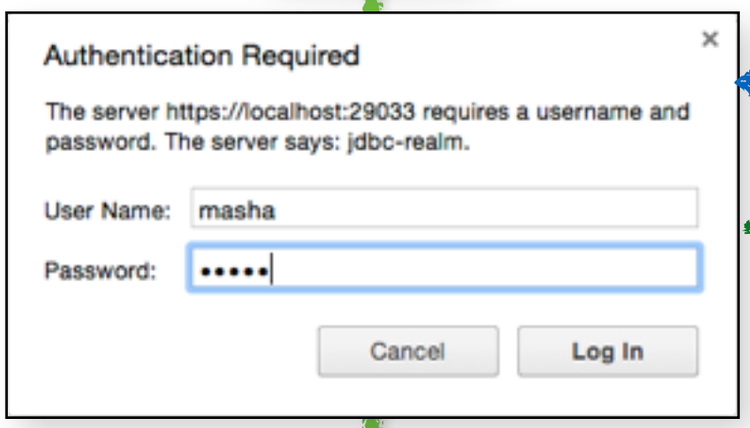
DIGEST (RFC 2069 / 2617)

```
<login-config>
   <auth-method>DIGEST</auth-method>
   <realm-name>jdbc-realm</realm-name>
</login-config>
```

configuration in **web.xml**

GET /app/secret

401 Unauthorized
WWW-Authenticate: Digest realm="jdbc-realm",
qop="auth", nonce="143...064", opaque="DF...5C"



GET /app/secret
Authorization: Digest username="masha", realm="jdbc-realm",
nonce="143...f89", uri="/app/faces/biblioteca.xhtml",
response="2c40...df", opaque="DF...5C", qop=auth,
nc=00000001, cnonce="66...948b"

Credentials: MD5 hash

200 OK

To logout: exit the browser!

FORM

```
<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>
```

configuration in **web.xml**

GET /app/secret

200 OK

Set-Cookie: JSESSIONID=aac...44; Path=/app; Secure; HttpOnly

```
<form action="j_security_check"
        method="POST">
 <input type="text" name="j_username">
 <input type="password" name="j_password">
  <input type="submit">
</form>
```

Login:

User name: masha

Password: •••••

Submit Reset

POST https://localhost:29033/app/faces/j_security_check

Referer: https://localhost:29033/app/faces/biblioteca.xhtml

Cookie: JSESSIONID=aaab5...b1f6

Authorization:Basic Y2VyZWJybzoxMjM0NQ==

Base64 encoding

j_username:masha

j_password:12345

Protection depends on SSL/TLS layer

Biblioteca

User: masha

Imagens disponíveis

Acesso livre a todos os cadastrados

@RolesAllowed ({"administrador", "amigo"})

200 OK

To logout: **HttpSession#invalidate()**

CLIENT-CERT

```
<login-config>
   <auth-method>CLIENT-CERT</auth-method>
</login-config>
```

configuration in **web.xml**

GET /app/secret

Receives certificate from server

```
Owner: CN=Self-Signed, OU=Oracle
Corporation, O=Glassfish
Issuer: CN=Self-Signed, OU=Oracle
Corporation, O=Glassfish
Serial number: 3E:56:D0:C9
Valid from: Jul 3 00:12:13 BRT 2015 until:
Jun 30 00:12:13 BRT 2025
Certificate fingerprints:
   MD5:  F5:A9:21:E3:59:30:E8:...:C3:8C:DE
   SHA1: BE:F5:1A:C3:8C:DE:68:...:83:72:A4
   SHA256: 1E:2D:F1:96:8B:C2:...:C2:C7:6E
```

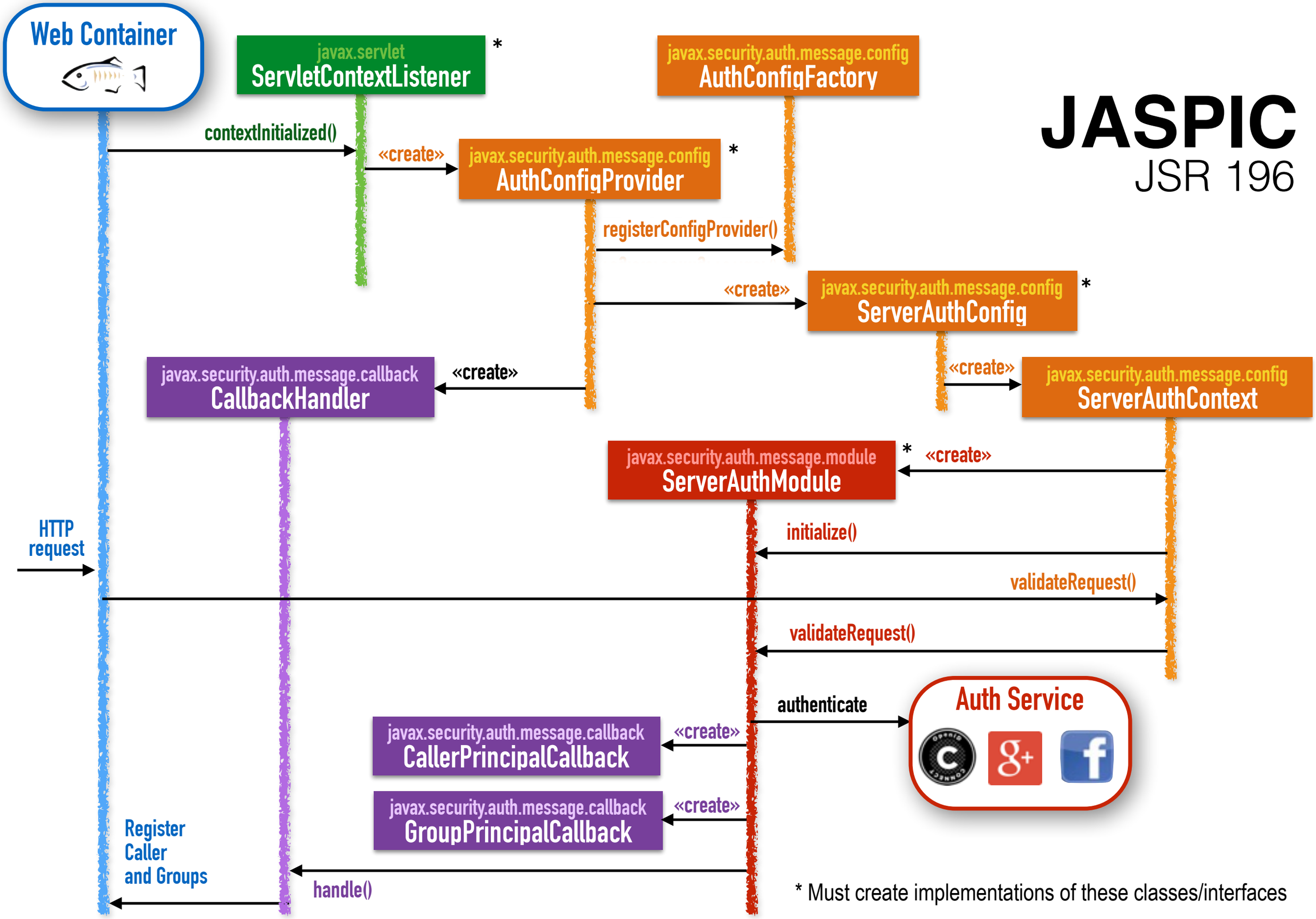
```
Owner: CN=Helder da Rocha, OU=Argonavis,
O=Argonavis, L=Sao Paulo, ST=SP, C=BR
Issuer: CN=Helder da Rocha, OU=Argonavis,
O=Argonavis, L=Sao Paulo, ST=SP, C=BR
Serial number: 156fb994
Valid from: Tue Jul 14 00:28:45 BRT 2015
until: Mon Oct 12 00:28:45 BRT 2015
Certificate fingerprints:
   MD5:  F2:B7:23:7B:61:35:D6:...:27:62:7C
   SHA1: EC:52:88:9B:4F:63:9D:...:EC:D2:17
   SHA256: 2E:35:2B:CF:41:30:...:28:0F:B3
```

Sends client certificate

200 OK

Configuration is platform-dependent

JASPIC
JSR 196



* Must create implementations of these classes/interfaces

Authorization in Java EE

Annotations

Deployment descriptors

JACC (JSR 115)

Web Layer

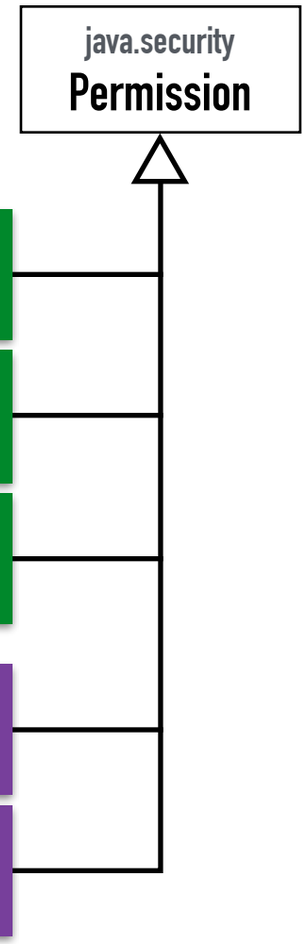
```
@ServletSecurity(
   @HttpConstraint(
      transportGuarantee = ...,
      rolesAllowed = {...}
   )
)
public class MyServlet... {...}
```

```
<security-constraint>                                   web.xml
     <web-resource-collection>
     ...
    </web-resource-collection>
     <auth-constraint>
        <role-name>...</role-name>
     </auth-constraint>
     <user-data-constraint>
        <transport-guarantee>...</transport-guarantee>
     </user-data-constraint>
</security-constraint>
```

EJB Layer

```
@RolesAllowed({...})
public class Bean {...
   @DenyAll
   public void m1() {}
   @PermitAll
   public void m2() {}
```

```
<method-permission>                                     ejb-jar.xml
     <role-name>...</role-name>
     <method>
        <ejb-name>...</ejb-name>
       <method-name>...</method-name>
      </method>
</method-permission>
```



by Helder da Rocha (www.argonavis.com.br)