Servidores de Aplicação

Especificaçãon JSP 3.1

13 - Security

- Requisito para a maioria das aplicações
- Os containers oferecem suporte e mecanismo para resolver esses problemas:
  - Autenticação
  - Controle de acesso aos recursos
  - Integridade dos dados
  - Privacidade dos dados
- Caso especial: RequestDispatcher

13 - Security

- Essa segurança pode ser implementada de três formas diferentes:
  - Programaticamente
  - Declarativa
    - Anotações (Annotations)
    - web.xml
- Política *default*
  - Nenhum recurso requer autenticação
  - A camada de transporte segura não é obrigatoria

13 - Security

- Segurança Programática
  - Na interface HttpServletRequest
    - authenticate()
    - login()
    - logout()
    - getRemoteUser()
    - isUserInRole(String role)
    - getUserPrincipal()

13 - Security

- Segurança Declarativa (Annotations)
  - @ServletSecurity

Element	Description	Default
`value`	the HttpConstraint that defines the protection to be applied to all HTTP methods that are NOT represented in the array returned by httpMethodConstraints.	@HttpConstrai nt
`httpMethodConstraints`	the array of HTTP method specific constraints.	{}

13 - Security

- Segurança Declarativa (Annotations)
  - @HttpConstraint

Element	Description	Default
`value`	The default authorization semantic that applies (only) when rolesAllowed returns an-empty array.	PERMIT
`rolesAllowed`	An array containing the names of the authorized roles	{}
`transportGuarantee`	The data protection requirements that must be satisfied by the connections on which requests arrive.	NONE

13 - Security

- Segurança Declarativa (Annotations)
  - @HttpMethodConstraint

Element	Description	Default
value	The HTTP protocol method name	
`emptyRoleSemantic`	The default authorization semantic that applies (only) when rolesAllowed returns an empty array.	PERMIT
`rolesAllowed`	An array containing the names of the authorized roles	{}
`transportGuarantee`	The data protection requirements that must be satisfied by the connections on which requests arrive.	NONE

13 - Security

- Segurança Declarativa (Annotations)
  - Exemplos

```
@ServletSecurity
public class Example1 extends HttpServlet {
}

@ServletSecurity(@HttpConstraint(transportGuarantee =
TransportGuarantee.CONFIDENTIAL))
public class Example2 extends HttpServlet {
}

@ServletSecurity(@HttpConstraint(rolesAllowed = "R1"))
public class Example4 extends HttpServlet {
}
```

13 - Security

- Segurança Declarativa (Annotations)
  - Exemplos

```
@ServletSecurity((httpMethodConstraints = {
        @HttpMethodConstraint(value = "GET", rolesAllowed = "R1"),
        @HttpMethodConstraint(value = "POST", rolesAllowed = "R1",
        transportGuarantee = TransportGuarantee.CONFIDENTIAL)
    })
public class Example5 extends HttpServlet {
}
```

```
@ServletSecurity(value = @HttpConstraint(rolesAllowed = "R1"),
       httpMethodConstraints = @HttpMethodConstraint(value="TRACE",
emptyRoleSemantic = EmptyRoleSemantic.DENY))
public class Example7 extends HttpServlet {
}
```

13 - Security

- Segurança Declarativa (web.xml)
  - Principais diretivas:
    - security-constraint
      - web-resource-collection
        - url-pattern
        - http-method-omission
        - http-method
      - auth-constraint
        - role-name
      - user-data-constraint
        - transport-guarantee

13 - Security

- Mapeamento de annotations para web.xml

```
@ServletSecurity(value=@HttpConstraint(rolesAllowed = "Role1"),
    httpMethodConstraints = @HttpMethodConstraint(value = "TRACE",
<security-constraint>
    <web-resource-collection>
        <url-pattern>...</url-pattern>
        <http-method-omission>TRACE</http-method-omission>
    </web-resource-collection>
    <auth-constraint>
        <role-name>Role1</role-name>
    </auth-constraint>
</security-constraint>
<security-constraint>
    <web-resource-collection>
    <url-pattern>...</url-pattern>
    <http-method>TRACE</http-method>
</web-resource-collection>
<auth-constraint/>
</security-constraint>
```

13 - Security

- Mapeamento de annotations para web.xml
  - Sua vez

```
@ServletSecurity(@HttpConstraint(rolesAllowed = "Role1"))
```

13 - Security

- Mapeamento de annotations para web.xml
  - Sua vez

```
@ServletSecurity(@HttpConstraint(rolesAllowed = "Role1"))
```

```
<security-constraint>
    <web-resource-collection>
        <url-pattern>...</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>Role1</role-name>
    </auth-constraint>
</security-constraint>
```

13 - Security

- Roles
  - Grupos lógicos criados pela desenvolvedor
  - Deve ser mapeado no Container (Servidor de aplicação para ser usado
  - Ver tomcat/conf/tomcat-users.xml
  - Exemplos:
    - Admin,Visitante
    - Nivel1,Nivel2,Nivel3
    - 1,2,3,4,5,6

13 - Security

- Formas de Autenticação
  - HTTP Basic Authentication
  - HTTP Digest Authentication
  - Form Based Authentication
  - HTTPS Client Authentication

13 - Security

- Formas de Autenticação
  - HTTP Basic Authentication
    - Válido desde o HTTP 1.0
    - Baseado em:
      - username
      - password
    - Não é seguro
      - Encode base64
      - Pode-se usar HTTPS para dar segurança

13 - Security

- Formas de Autenticação
  - HTTP Digest Authentication
    - Válido desde o HTTP 1.0
    - Baseado em:
      - username
      - password
    - Seguro
      - Utiliza encriptamento

13 - Security

- Formas de Autenticação
  - Form Based Authentication
    - Deve ter uma página de erro
    - Dever ter uma página com formulário e os campos:
      - j_username
      - j_password (com a propriedade autocomplete=”off”)
    - O formulário deve enviar para
      - j_security_check
    - Seguro apenas quando o formulário é enviado sobre HTTPS

13 - Security

- Formas de Autenticação
  - HTTPS Client Authentication
    - Mais segura
    - Envolve certificados digitais

13 - Security

- **Exercícios**

1) Implemente a forma de autenticação baseada em formulario para acessar uma determinada Servlet.

2) Implemente uma Servlet que só será executado por usuários que possuam a Regra “admin”.