

JSP & Servlets

Beveiliging van websites

Bron:

Cursus Java Webcomponenten, Noël Vaes

Inhoud

Beveiliging van web-applicaties

1. Algemeen

2. Authenticatie

- a. Algemeen
- b. Basic Authentication
- c. Digest Authentication
- d. Formulier gebaseerde authenticatie
- e. Programmatorische authenticatie
- f. HTTPS Client Certificate

3. Autorisatie

- a. configuratie via web.xml
- b. configuratie via annotaties

4. Encryptie

5. Programmatorische beveiliging

Algemeen

Meerdere technieken mogelijk

Zelf ontwikkelen

Servlets voorzien van een beveiligingsmechanisme. Hierbij vragen we eerst een gebruikersnaam en paswoord op.

Dezelfde techniek gebruiken in een filter, zodat we deze niet telkens hoeven te herhalen in de servlets

...

Declaratieve beveiliging

We laten de beveiliging over aan de webcontainer

Onderdelen beveiliging via de webcontainer

1. Authenticatie

Gebruikers moeten zich met de nodige credentials kenbaar maken aan de webcontainer

2. Autorisatie

Bepalen welke gebruikers toegang hebben tot welke onderdelen

3. Encryptie

Aangeven of een verbinding al dan niet versleuteld moet zijn. Deze versleuteling gebeurt door de webcontainer

Authenticatie

Algemeen

Wat is authenticatie?

Zich kenbaar maken aan de webcontainer met de nodige bewijzen (= credentials)

Credentials

- Gebruikersnaam en paswoord

- Een certificaat

Soorten

- Basic authentication

- Digest authentication

- Client-Certification authentication

- Form authentication

Gebruikers, groepen en rollen

Om een web-applicatie, of gedeelten ervan te beveiligen dienen we eerst gebruikers, groepen en rollen te definiëren.

1. Gebruikers

zijn de **individuele personen** die toegang moeten krijgen tot de web-applicatie.

Iedere gebruiker heeft zijn eigen wachtwoord.

2. Groepen

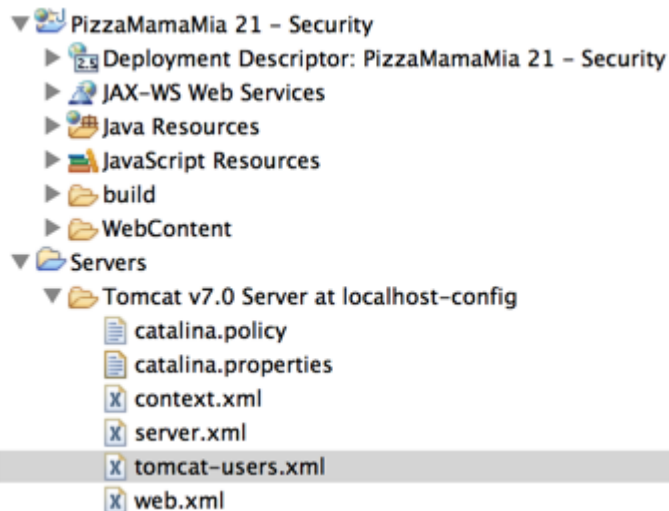
zijn **verzamelingen van gebruikers** die dezelfde toegangsrechten moeten krijgen

3. Rollen

bepalen **de functies** die gebruikers en groepen in de web-applicatie vervullen.

Configuratie van gebruikers, groepen en rollen

1. verschilt per webserver
2. bij **Tomcat**:
via het configuratiebestand **tomcat-users.xml** in de map **conf** van de Tomcat-installatie



Bij de integratie in **Eclipse** bevindt een kopie van het bestand zich in het project **Servers** dat bij het opstarten van Tomcat automatisch gekopieerd wordt

tomcat-users.xml

```
<role rolename="admin"></role>  
<role rolename="guest"></role>  
<user username="kenneth" password="kenneth" roles="admin, guest"></user>  
<user username="test" password="test" roles="guest"></user>
```

web.xml

Het authenticatie-mechanisme wordt in web.xml geconfigureerd met de tag

<login-conf>

Tag	Omschrijving
<login-conf>	Definieert het authenticatie-mechanisme
<auth-method>	Definieert de vier mogelijke authenticatie-methoden: BASIC, DIGEST, CLIENT-CERT of FORM
<realm-name>	De naam van de realm. Is enkel van toepassing voor de basic authentication.
<form-login-config>	Definieert de URL van de login-pagina die gebruikt wordt voor de formulier gebaseerde authenticatie

Basic Authentication

Basic Authentication

Werking

De browser **toont zelf een dialoogvenster** waarbij de gebruiker zijn username en password kan invullen.

Daarna controleert de server deze en verleent al dan niet toegang.

Voordelen

eenvoudig
door alle browsers ondersteund

Nadelen

niet echt veilig
geen aanpassingen mogelijk



The image shows a standard Basic Authentication dialog box. On the left is a circular icon with a blue background and three white horizontal lines. To the right of the icon, the text reads: 'Om deze pagina weer te geven, moet u inloggen bij het volgende gedeelte op 'localhost:8080':'. Below this, it says 'Authentication required' and 'Uw wachtwoord wordt ongecodeerd verstuurd.'. There are two input fields: 'Naam:' followed by a text box, and 'Wachtwoord:' followed by a password box. Below the password box is a checkbox labeled 'Bewaar wachtwoord in mijn sleutelhanger'. At the bottom right are two buttons: 'Annuleer' and 'Log in'.

Om deze pagina weer te geven, moet u inloggen bij het volgende gedeelte op 'localhost:8080':

Authentication required

Uw wachtwoord wordt ongecodeerd verstuurd.

Naam:

Wachtwoord:

☐ Bewaar wachtwoord in mijn sleutelhanger

Annuleer Log in

Geen toegang verleend...



configuratie in de deployment descriptor.

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

```
<security-constraint>  
  <web-resource-collection>  
    <web-resource-name>welkom pagina</web-resource-name>  
    <url-pattern>/welkom.htm</url-pattern>  
  </web-resource-collection>  
  <auth-constraint>  
    <role-name>administrators</role-name>  
  </auth-constraint>  
</security-constraint>
```

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

Digest authentication

Digest authentication

Werking

Identiek aan Basic authentication

Gebruikersnaam en paswoord worden echter nu versleuteld

Voordelen

veiliger dan Basic authentication

Nadelen

Enkel ondersteund door Microsoft Internet Explorer vanaf versie 5

Wordt niet door alle servlet containers ondersteund omdat de servlet specificatie het neit vereist.

Web.xml

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

Formulier gebaseerde authenticatie

Formulier gebaseerde authenticatie

Werking

Identiek aan Basic authentication

Zelf ontworpen inlogformulier mogelijk

Voordelen

De authenticatie kan volledig geïntegreerd worden in de web-applicatie d.m.v. een loginscherm

Nadelen

Dezelfde nadelen als bij de Basic authenticatie: het wachtwoord wordt in klartekst over het netwerk verzonden en kan dus onderschept worden. Ook maakt de combinatie met HTTPS dit mechanisme wel veilig.

Formulier gebaseerde authenticatie

Configuratie (web.xml):

```
<login-config>
  <auth-method>FORM</auth-method>
  <form-login-config>
    <form-login-page>/WEB-INF/JSP/login.jsp</form-login-page>
    <form-error-page>/WEB-INF/JSP/error.jsp</form-error-page>
  </form-login-config>
</login-config>
```

Opmerking

De login-pagina moet een formulier bevatten dat zijn gegevens doorstuurt naar de volgende URL: **j_security_check**.

Het formulier moet tevens volgende velden bevatten:

Gebruikersnaam: **j_username**

Wachtwoord: **j_password**

Voorbeeld van een inlog formulier

DEMO

Programmatrische authenticatie

Programmatrische authenticatie

Ter vervanging van formulier gebaseerde authenticatie

Met de methods **login()** en **logout()** van het **HttpServletRequest** object

Zelfde voor- en nadelen van een formulier gebaseerde authenticatie

HTTPS Client Certificate

HTTPS Client Certificate

Werking

Maakt deel uit van het **HTTPS protocol**

Alle gegevens tussen browser en webserver worden **versleuteld d.m.v. SSL** (Secure Socket Layer)

Werkt met **certificaten**

Voordelen

Meest veilig. De gebruiker moet over een certificaat (bv. op een smartcard) beschikken om zich aan te melden bij de server.

Nadelen

vraagt **veel inspanning** wat betreft configuratie en onderhoud. Enkel nuttig bij hoge beveiligingseisen

Autorisatie

Wat is autorisatie ?

Tweede deel van de beveiliging

Bepalen welke onderdelen van de website voor welke rollen toegankelijk zijn.

Kan ofwel via de **deployment descriptor** (web.xml) of **via annotations** gebeuren

Configuratie via web.xml

Configuratie via web.xml

Wanneer via web.xml ?

Indien we ook **gewone HTML**-pagina's en **andere bronnen** willen beveiligen.
Het gebruik van annotations is daar namelijk niet mogelijk

Hoe ?

1. Met de tag **<security-constraints>** welke bestaat uit volgende delen:

Tag	Omschrijving
<display-name>	Naam (optioneel)
<web-resource-collection>	Beschrijving van de te beveiligen bronnen
<auth-constraint>	Beschrijven van de rollen die toegang krijgen

2. Tenslotte moeten we met de tag **<security-role>** nog eens de rollen apart definiëren

Configuratie via web.xml

<web-resource-collection>

Het bepalen van de bronnen die beschermd moeten worden.

Tag	Omschrijving
<web-resource-name>	Naam van deze verzameling gegevens
<description>	Beschrijving van deze verzameling (optioneel)
<url-pattern>	URL patroon voor de gegevens die beveiligd moeten worden. Men kan meerdere patronen na elkaar opgeven.
<http-method>	De HTTP methode waarvoor deze beveiliging geldt. Men kan meerdere methoden opgeven

Configuratie via web.xml

<auth-constraints>

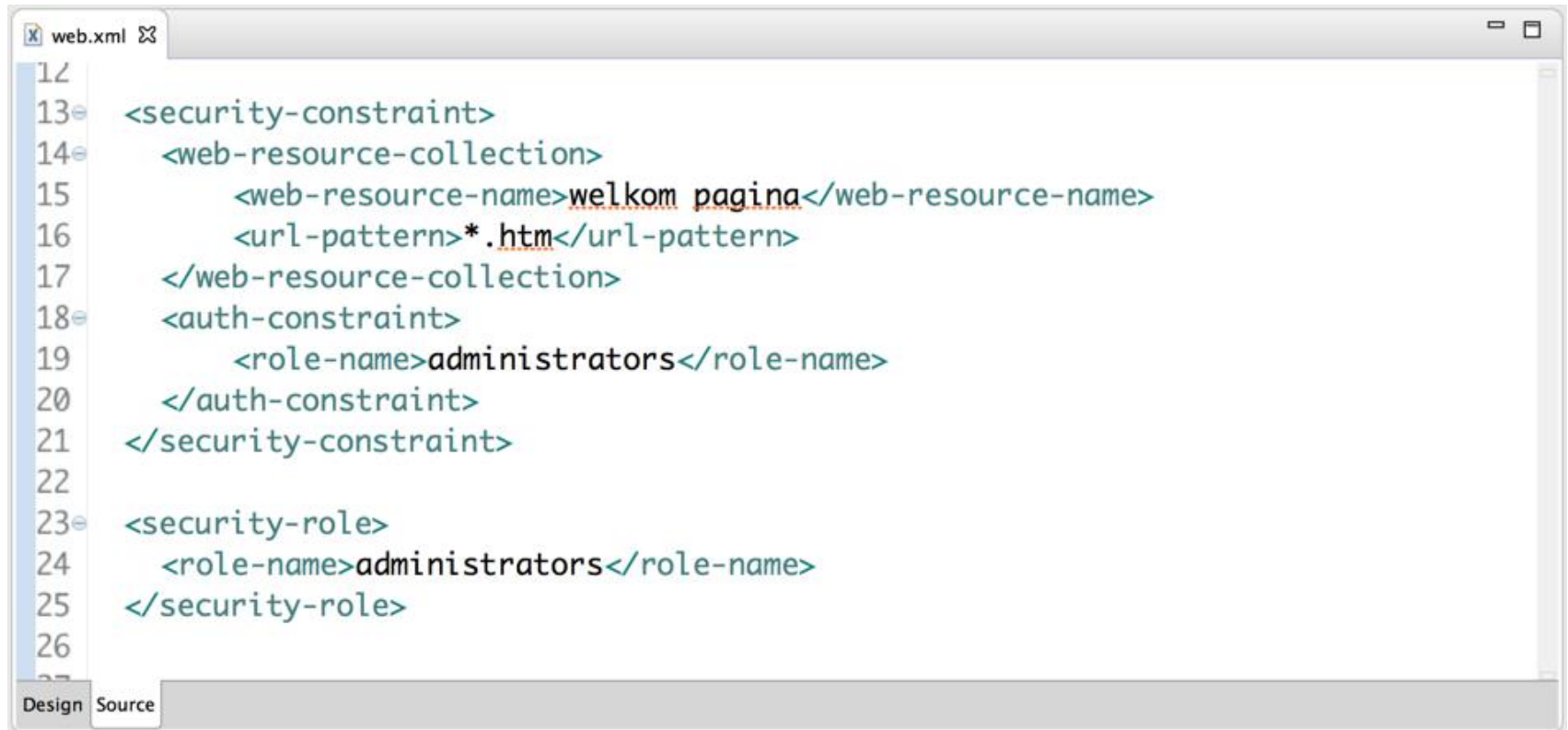
Aangeven welke rollen toegang krijgen tot deze bronnen.

Tag	Omschrijving
<description>	Beschrijving (optioneel)
<role-name>	Naam van de rol die toegang krijgt tot deze verzameling gegevens. Er kunnen meerdere rollen opgesomd worden. Om alle rollen aan te geven, gebruikt men *

<security-role>

Afzonderlijk definiëren van de rollen die door deze web-applicatie erkend worden (zie voorbeeld verder...)

Configuratie via web.xml



```
12
13 <security-constraint>
14   <web-resource-collection>
15     <web-resource-name>welkom pagina</web-resource-name>
16     <url-pattern>*.htm</url-pattern>
17   </web-resource-collection>
18   <auth-constraint>
19     <role-name>administrators</role-name>
20   </auth-constraint>
21 </security-constraint>
22
23 <security-role>
24   <role-name>administrators</role-name>
25 </security-role>
26
27
```

Design Source

Configuratie via annotaties

Configuratie via annotations

Wanneer via annotations ?

Indien de autorisatie enkel servlets betreft.

Hoe?

De annotation `@ServletSecurity` aan de servlet toevoegen.

@ServletSecurity

Element	Omschrijving
value	Beschrijft de toegangsrechten via de annotatie @HttpConstraint
httpMethodConstraints	De toegangsrechten via een reeks annotaties van het type @HttpMethodConstraint . Men kan hiermee ook aangeven voor welke HTTP-method dit geldig is.

Configuratie via annotations

@HttpConstraint en @HttpMethodConstraint

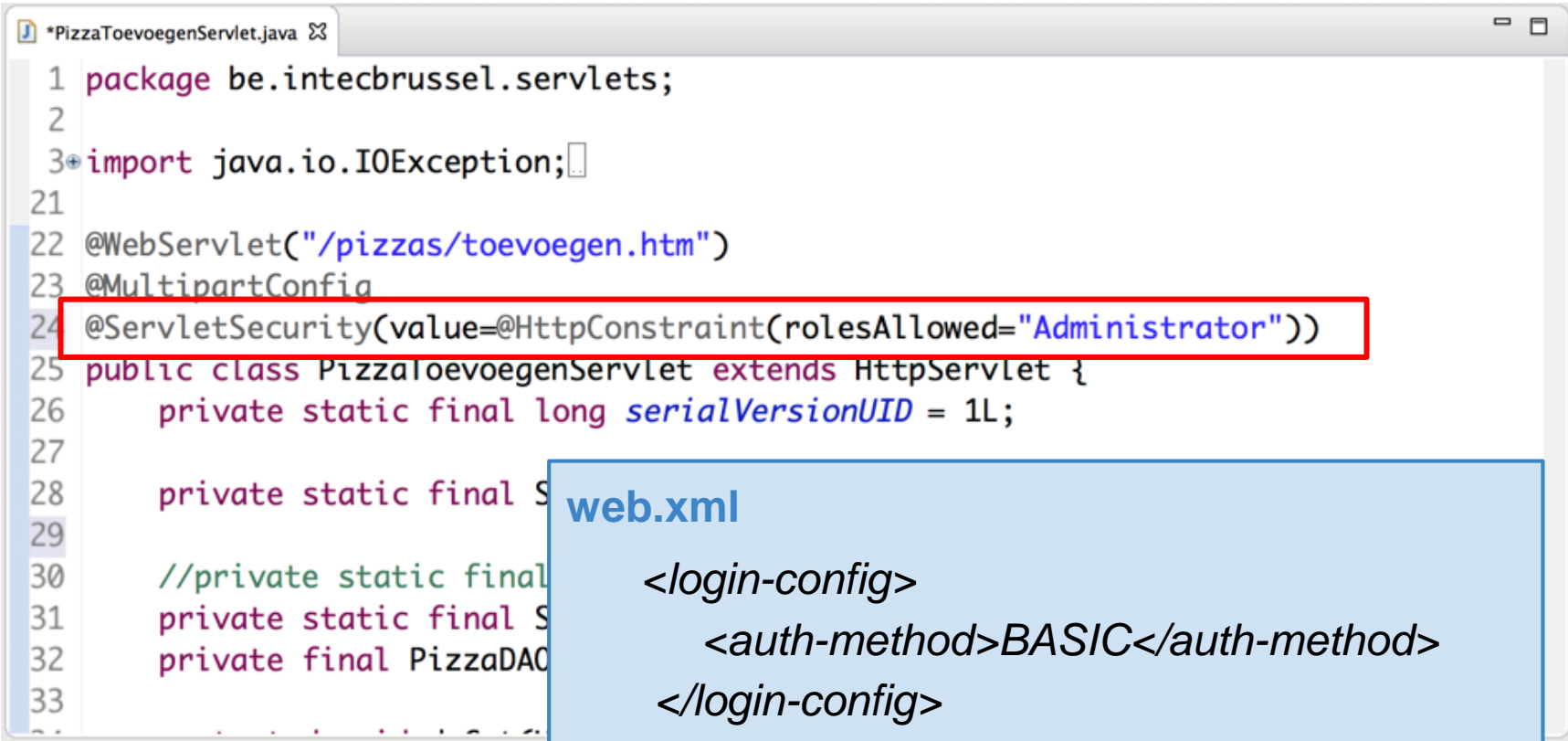
@HttpConstraint

Element	Omschrijving
rolesAllowed	Reeks van rollen die toegang hebben.

@HttpMethodConstraint

Element	Omschrijving
rolesAllowed	Reeks van rollen die toegang hebben.
value	De HTTP-methode waarvoor deze regel geldt: GET, POST ...

Voorbeeld van een configuratie via annotaties



```
1 package be.intecbrussel.servlets;
2
3 import java.io.IOException;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 @WebServlet("/pizzas/toevoegen.htm")
23 @MultipartConfig
24 @ServletSecurity(value=@HttpConstraint(rolesAllowed="Administrator"))
25 public class PizzaToevoegenServlet extends HttpServlet {
26     private static final long serialVersionUID = 1L;
27
28     private static final String
29
30     //private static final
31     private static final String
32     private final PizzaDAO
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

web.xml

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

Encryptie

Encryptie

Derde deel van de beveiliging: **de versleuteling**

Verbinding tussen browser en webcontainer wordt hierbij beveiligd

Kan ofwel in **web.xml** ofwel via **annotations** gebeuren

In web.xml

We gebruiken hiervoor de tag **<user-data-constraint>** die een subtag is van **<security-constraint>**. Deze tag heeft op zijn beurt volgende subtags:

Tag	Omschrijving
<description>	Beschrijving (optioneel)
<transport-garantee>	NONE: geen vereisten (standaardwaarde) INTEGRAL: integriteit moet gewaarborgd zijn CONFIDENTIAL: de confidentialiteit moet gewaarborgd zijn: volledige versleuteling.

Encryptie

Bij gebruik van annotaties:

Uitbreiding op de @ServletSecurity annotatie van daarnet:

```
23 @WebServlet("/pizzas/toevoegen.htm")
24 @MultipartConfig
25 @ServletSecurity(value=@HttpConstraint(rolesAllowed="Administrator",
26                                     transportGuarantee=TransportGuarantee.CONFIDENTIAL))
27
```

Opmerking

In het geval van INTEGRAL of CONFIDENTIAL wordt gebruik gemaakt van HTTPS voor de communicatie tussen client en server. In dit geval dient de webcontainer over een certificaat te beschikken

Programmatorische beveiliging

Programmatoreische beveiliging

Biedt volledige controle

Het **HttpServletRequest** object beschikt ondermeer over volgende methoden:

Methode	Omschrijving
login()	Meldt een gebruiker aan met username en password
logout()	Meldt de huidige gebruiker af
authenticate()	Lanceert indien nodig het authenticatiemechanisme van de webserver. Dit vervangt de configuratie via web.xml of annotatie
getRemoteUser()	Geeft de naam van de gebruiker die aangemeld is
isUserInRole()	Geeft aan of de huidige gebruiker deel uitmaakt van een bepaalde rol