# Security Technologien in Java EE 6

**Java Forum Stuttgart 2010**

Sebastian Glandien

Acando GmbH

sebastian.glandien@acando.de

# Agenda

I. Einleitung

II. Java Authentication SPI for Containers (JSR-196)
   I. Message Processing Model
   II. Profile
   III. Demo

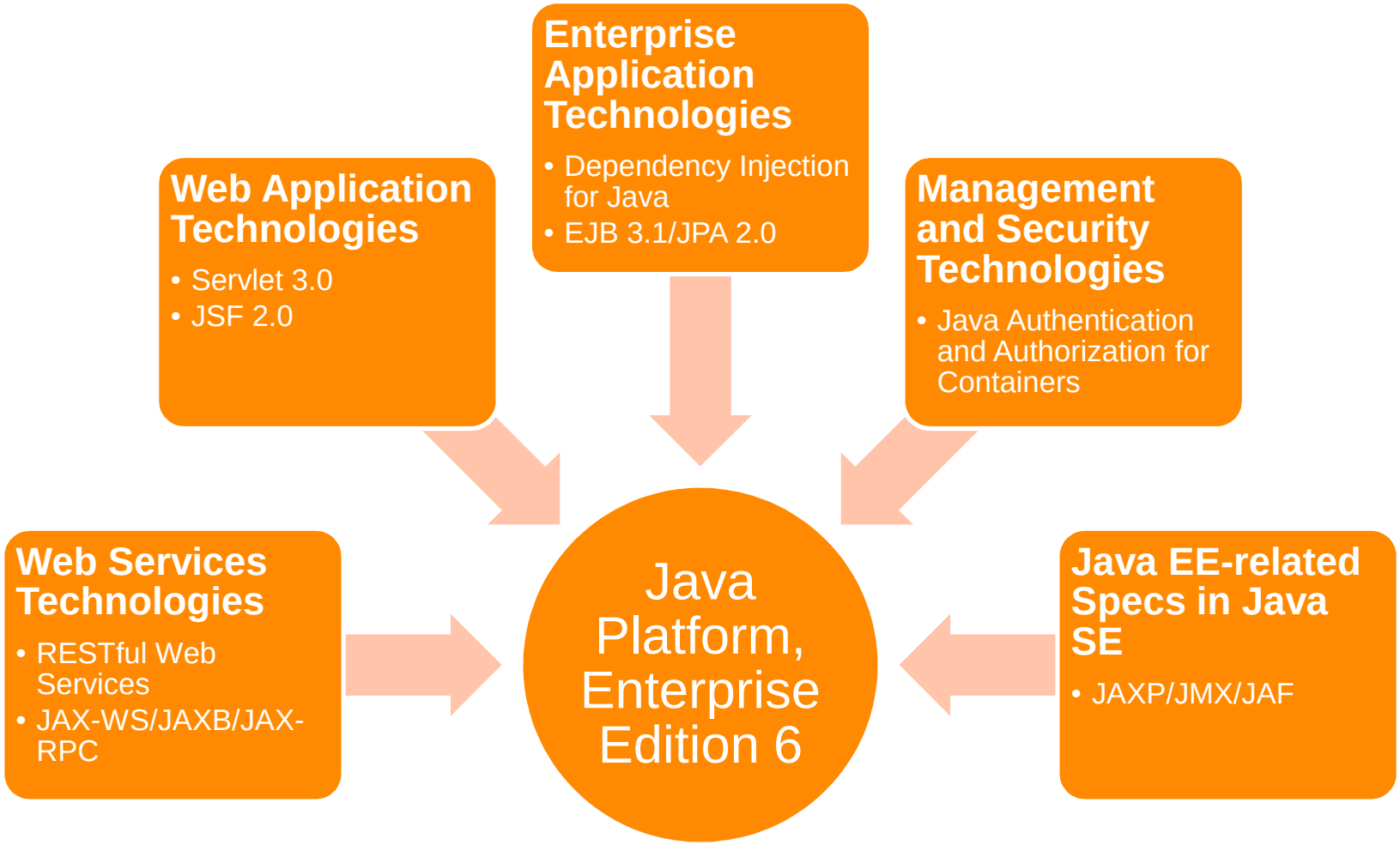
III. Java Autorization Contract for Containers (JSR-115)
   I. Klassendiagramm
   II. Subcontracts
   III. Demo

IV. Weitere Standards

V. Zusammenfassung

# Java EE 6 Technologies

## Einleitung



# Management and Security Technologies

## Einleitung

- Security Technologies:
  - Java Authentication SPI for Containers (JSR-196)
  - Java Authorization Contract for Containers (JSR-115)
- Management Technologies:
  - Java EE Application Deployment 1.2 (JSR 88)
  - J2EE Management 1.1 (JSR 77)

# Sicherheitsmodelle für JEE-Anwendungen

## Einleitung

### Programmatische Sicherheit

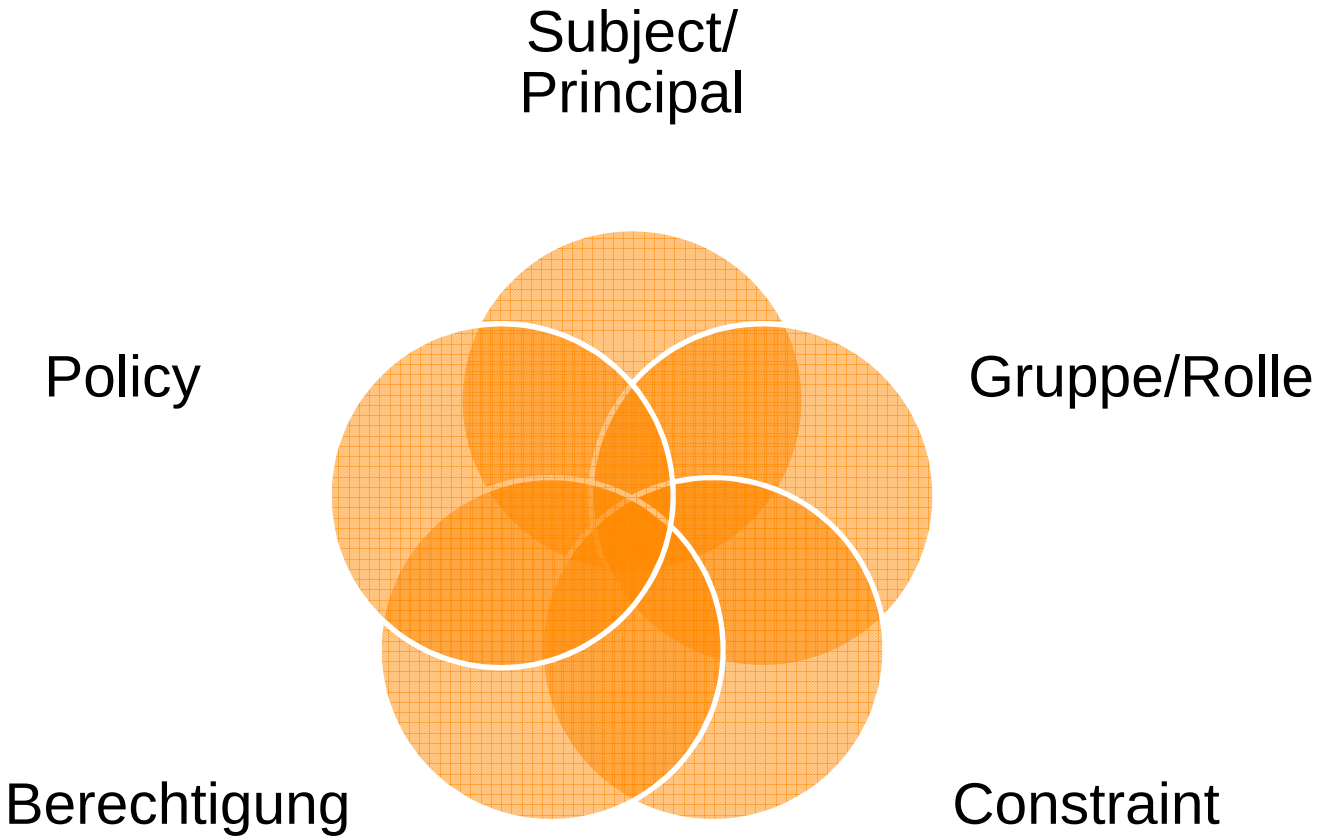
- Steuerung des Zugriffs im Programmcode
- Definition von Rollen und Berechtigungen innerhalb des Codes
- Keine Trennung von Busineslogik und Sicherheitaspekten
- Problem: Überprüfung der Bedingungen erst zur Laufzeit

### Deklarative Sicherheit

- Verwendung ohne Eingriff in den Quellcode
- ausschließliche Definition im Deployment Descriptor
- Vergabe von Zugriffsrechten einzig über Rollen
- Hintergrund: Sicherheit ist nicht Aufgabe des Progammieres

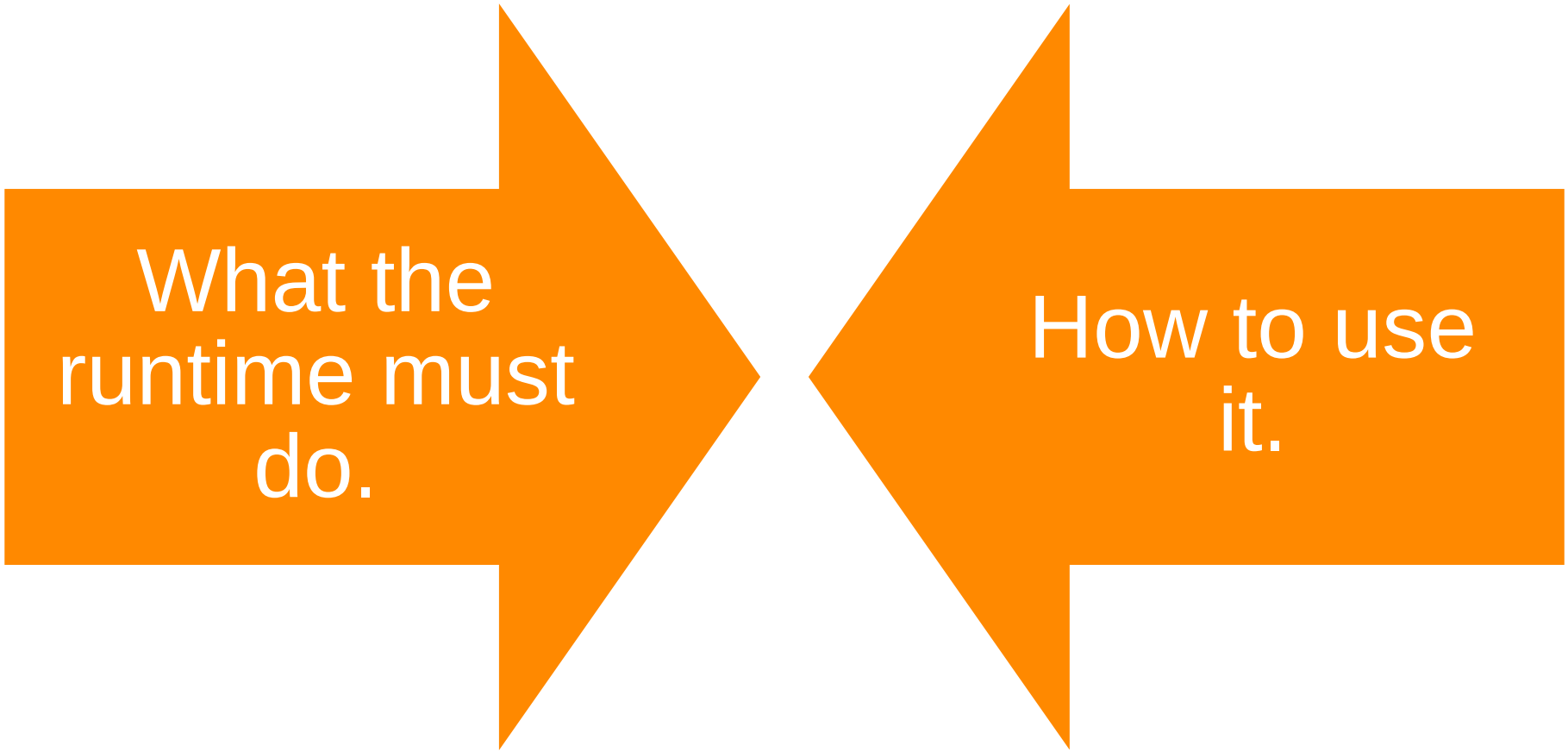
# Begriffsklärung

## Einleitung



# Blickwinkel auf die Standards

## Einleitung

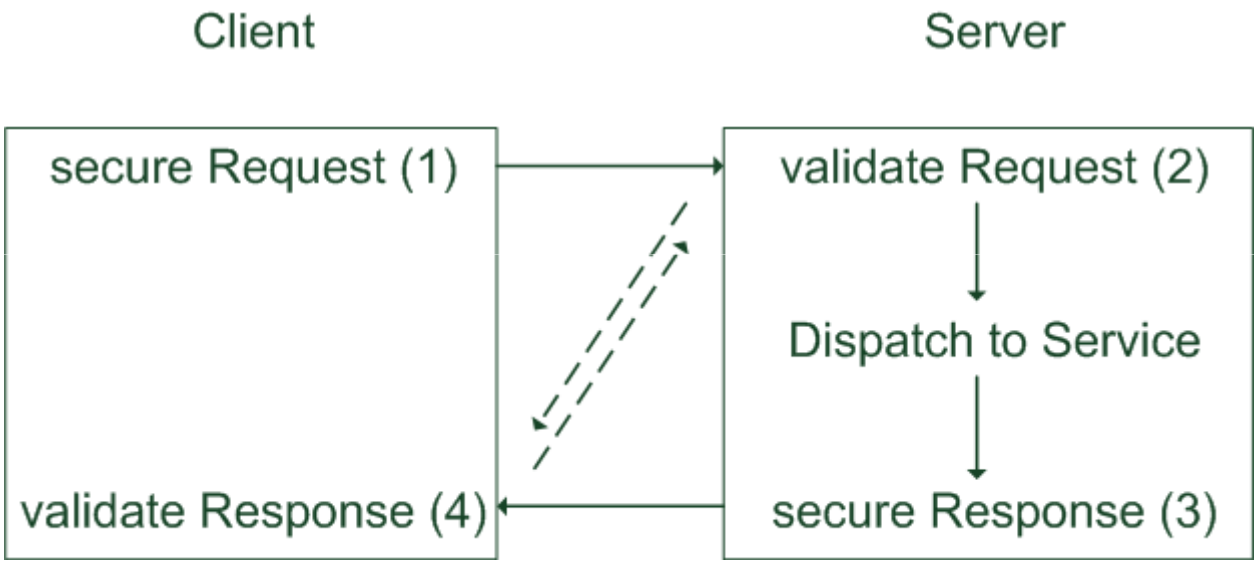


# Java Authentication SPI for Containers (JSR-196)

- Standard Service-Provider Interface (SPI) zur Integration eines Authentifizierungsmechanismus in message processing runtimes
- Definition von Profilen zur Verwendung des SPI in einem spezifischen Kontext
- Notwendige Interfaces für Authentifizierungsmodule:
  - `javax.security.auth.message.module.ClientAuthModule`
  - `javax.security.auth.message.module.ServerAuthModule`

# Message Processing Model

## Java Authentication SPI for Containers



# Servlet Container Profile

## Java Authentication SPI for Containers

- Definition der Verwendung von JSR-196 in einem Servlet Container
- Fokus liegt auf den Punkten 2 und 3 des Message Processing Models
  - Verhalten in 1 und 4 ist nicht spezifiziert
- Message Layer Identifier: **HttpServlet**
- Verarbeitung von Objekten des Typs `javax.servlet.http.HttpServletRequest` und `javax.servlet.http.HttpServletResponse`

# SOAP Profile

## Java Authentication SPI for Containers

- gültig für SOAP Versionen 1.1 und 1.2
- Unterteilung der Anforderungen des Profils in:
  - Clientseitig
  - Serverseitig
- Message Layer Identifier: **SOAP**
- Verarbeitung von Objekten des Typs `javax.xml.soap.SOAPMessage`

# Weitere Profile

## Java Authentication SPI for Containers

- LoginModule Bridge Profile
  - Delegation der Sicherheitsvalidierung an JAAS-Loginmodul (`javax.security.auth.spi.LoginModule`)
- Future Profiles
  - JMS Profile
  - RMI/IIOP Portable Interceptor Profile

# Java Authentication SPI im JBoss 5

1. Registrierung und Konfiguration des JSR-196 Modul in jaspi-webform-jboss-beans.xml

```
…org.jboss.web.tomcat.security.jaspi.modules.HTTPFormServerAuthModule…
```

2. Konfiguration eines Valve-Elements in context.xml

```
<Context>
    <Valve className="org.jboss.web.tomcat.
                                security.jaspi.TomcatJASPIAuthenticator" />
</Context>
```

3. Konfiguration der Security Domain der Webapplikation in jboss-web.xml

```
<jboss-web>
    <security-domain>java:/jaas/jaspi-test</security-domain>
</jboss-web>
```

# Registrierung und Konfiguration des JSR-196 Moduls

```
<deployment xmlns="urn:jboss:bean-deployer:2.0">
   <application-policy xmlns="urn:jboss:security-beans:1.0„ name="jaspi-test">
      <authentication-jaspi>
         <login-module-stack name="lm-stack">
            <login-module
  code="org.jboss.security.auth.spi.UsersRolesLoginModule“
             flag="required">
            </login-module>
         </login-module-stack>
         <auth-module
  code="org.jboss.web.tomcat.security.jaspi.modules.HTTPFormServerAuthModule“
  login-module-stack-ref="lm-stack" />
      </authentication-jaspi>
   </application-policy>
</deployment>
```

# JAAS vs. JSR-196

## JAAS

- Authentifizierungs-API für Pluggable und Sequentielle Authentifzierung
- Standard zur Konfiguration und Verbindung von Loginmodulen mit Applikationen
- Callback-Fähigkeit zwischen Services und Applikationen
- Protokollunabhängige Verarbeitung
- User = Subject

## JSR-196

- Nachrichtenbasierte Verarbeitung von Authentifizierungsinformationen
  - SSO-Support verfügbar
- Verwendung der deklarativen JEE-Security
- Verarbeitung von protokollspezifischen Nachrichten
- User = Principal

# Spring Security (Servlet Filter) vs. JSR-196

## Spring Security (Servlet Filter)

- Konfiguration als Teil der Applikation (via web.xml)
- Aufruf des Servlet Filters nachdem Authentifizierung und Autorisierung verarbeitet wurde
- Vorteile:
  - Spring-styled Konfiguration
  - Aspekt-orientierte Programmierung
  - Inversion-of-Control Mechanismus

## JSR-196

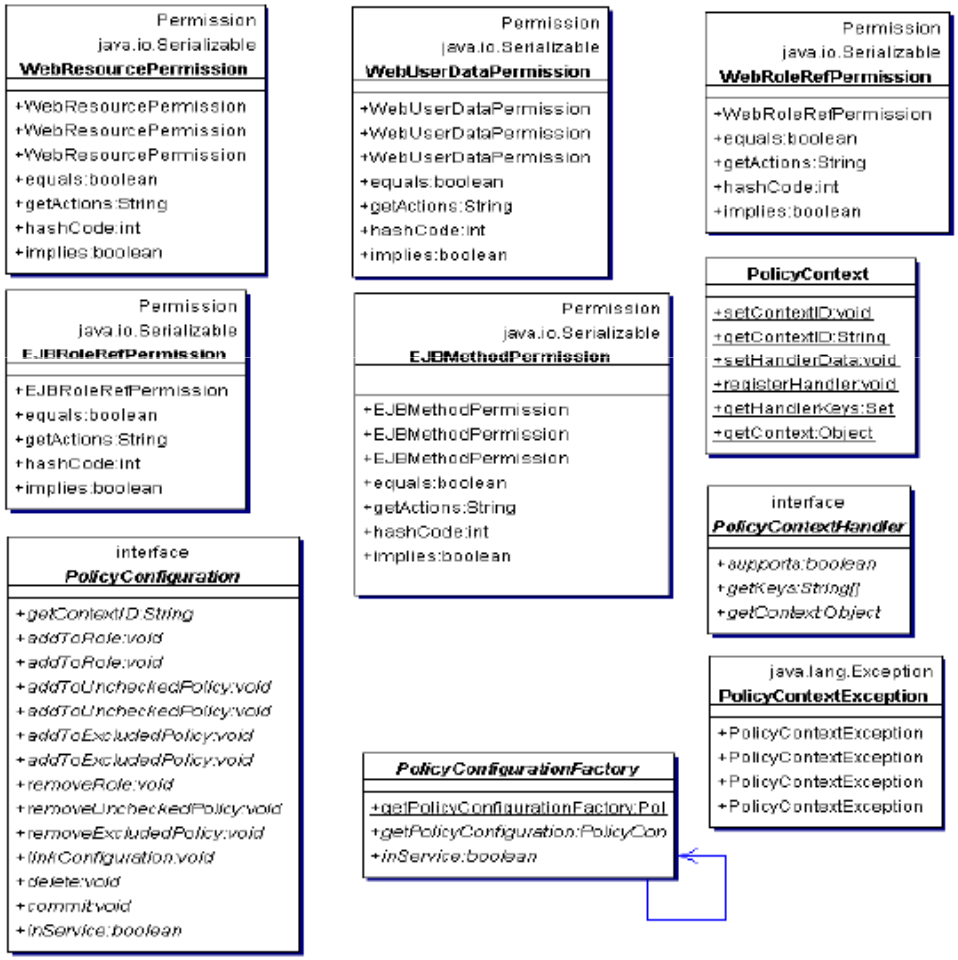
- Nachrichtenbasierte Verarbeitung von Authentifizierungsinformationen
  - SSO-Support verfügbar
- Verwendung der deklarativen JEE-Security
- Verarbeitung von protokollspezifischen Nachrichten
- Principal = User

# Java Authorization Contract for Containers (JSR-115)

- Definition eines Standardmechanismus für Authorization Provider zur Verwendung im JEE-Container
- Einführung neuer `java.security.Permission` Klassen zur Java EE Autorisierung
- Erfüllung neuer JEE-Anforderung zur Autorisierung:
  - Definition von Rollen als Sammlung von Berechtigungen
  - Zuweisung von Berechtigungen zu Principal mittels Rollen
  - `isCallerInRole`
  - Definition von Identifier zu Rollenmapping

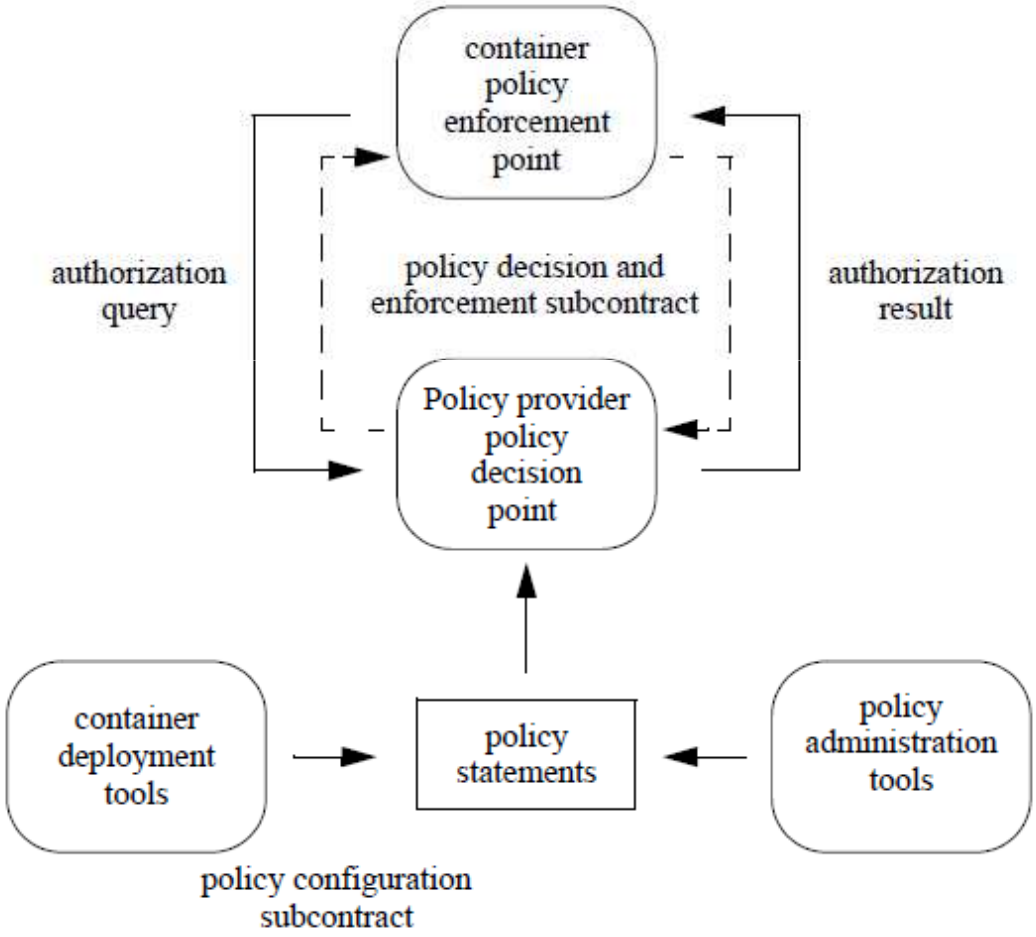
# javax.security.jacc Klassen Diagramm

## Java Authorization Contract for Containers



# Policy Configuration and Enforcement Subcontracts

## Java Authorization Contract for Containers



# Provider Configuration Subcontract

## Java Authorization Contract for Containers

- Festlegung der Berechtigungsverarbeitung (Policy) in Teilspezifikation

- Definition der Policy Provider Klassen:
  - policy.provider → java.security.Policy
  - auth.policy.provider → javax.security.auth.Policy
  - javax.security.jacc.policy.provider → …
  - …
- Konfiguration in Java Security Properties-Datei

# Policy Configuration Subcontract

## Java Authorization Contract for Containers

- Festlegung der Interaktion zwischen Container deployment tools und Providern
- Übersetzung der deklarativen JEE Autorisierungsinformationen in Policyregeln des Java Policy Provider
- Verarbeitung der Policy Deskriptoren:
  - `security-constraint`
  - `method-permission`
  - `security-role-ref`
  - `exclude-list`

# Policy Decision and Enforcement Subcontract

## Java Authorization Contract for Containers

- Policy Enforcement durch
  - Servlet Container
  - EJB Container
- Evaluation der Berechtigungen
  - Validierung von URL Pattern/EJB Methode
  - Integrierter Privilege Test
- AccessControlContext unabhängige Grants

# Java Authorization Contract im JBoss 5

```
<security-constraint>
   <display-name>Secured files constraint</display-name>
   <web-resource-collection>
      <web-resource-name>Secured files</web-resource-name>
      <url-pattern>/secure/*</url-pattern>
   </web-resource-collection>
   <auth-constraint>
      <role-name>admin</role-name>
   </auth-constraint>
   <user-data-constraint>
      <transport-guarantee>INTEGRAL</transport-guarantee>
   </user-data-constraint>
</security-constraint>
```

# Weitere Standards

- JSR 250 – Security Annotations (`javax.annotation.security….`)
  - `RunAs`
  - `RolesAllowed`
  - `PermitAll`
  - `DenyAll`
  - `DeclareRoles`
- seit JSR 154 – Servlet 2.5 (JEE5)
  - `web-resource-collection`
  - `security-constraint`
  - `auth-constraint`

# Zusammenfassung

- Deklaratives und portables Security Model
- Anwendbar für Web- und Enterprise JavaBeans (EJB) Module und Applikationen
- Standardisierte Authentifizierung und Autorisierung
- Anpassbar für diverse Transportprotokolle
- Konfiguration auf Container- und Verwendung auf Applikationsebene

# Kontakt

…vielen Dank für Ihre Aufmerksamkeit!

Sebastian Glandien
IT-Consultant
Geschäftsstelle Hamburg
Millerntorplatz 1, Hamburg
sebastian.glandien@acando.de