Security Patterns im Praxiseinsatz

Mike Wiesner
SpringSource Germany

Über mich

- Senior Consultant bei SpringSource Germany
- Spring-/Security-Consulting
- Trainings
- IT-Security Consulting / Reviews
- mike.wiesner@springsource.com

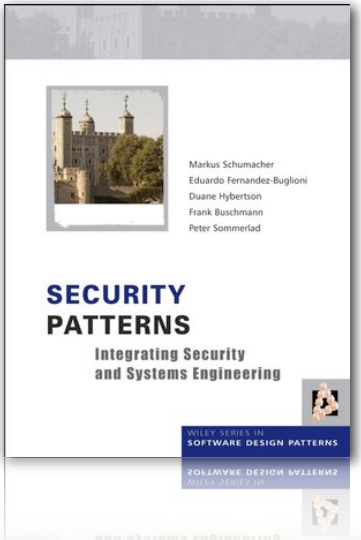
Agenda

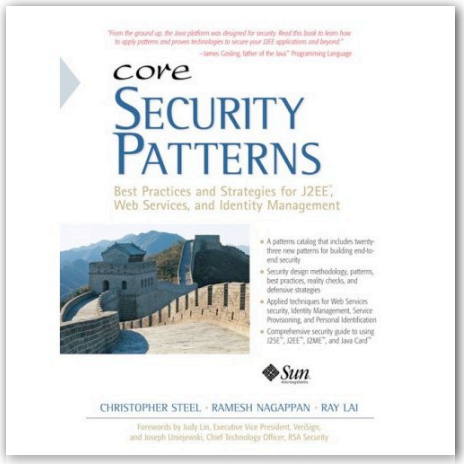
- Security-Patterns?
- Ungeprüfte Eingaben
- Authentifizierung
- Berechtigungsprüfung
- Vertrauensgrenzen

Entwurfsmuster

- Lösungsansatz für ein Probleme in einem bestimmten Kontext
- zeigt das Problem auf
- zeigt die Nachteile auf
- ist erprobt und bewährt
- ist generisch (Sprachunabhängig)

Literatur





Probleme

- Keine konkrete Implementierung
- Keine Einbeziehung von Frameworks/Libs
- Keine Verwendung von modernen Technologien
- Es gibt sehr viele davon ;-)

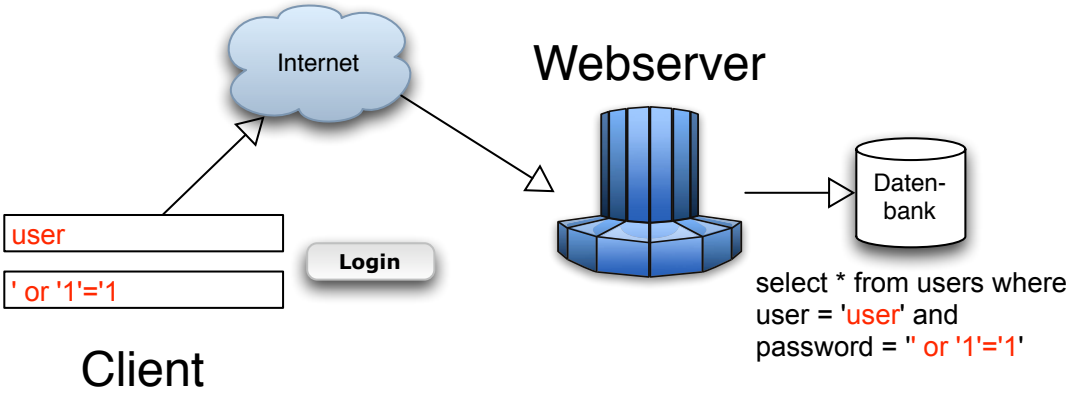
Agenda

- Security-Patterns?
- Ungeprüfte Eingaben
- Authentifizierung
- Berechtigungsprüfung
- Vetrauensgrenzen

Ungeprüfte Eingaben

- Grund für sehr viele Schwachstellen
- ungeprüft ins Backend
  - SQL
  - LDAP
  - XPath
- ungeprüft zum Client
  - Cross-site scripting (XSS)
  - Cross-site request-forgery (XSRF)

SQL Injection



JDBC Prepared Statement?

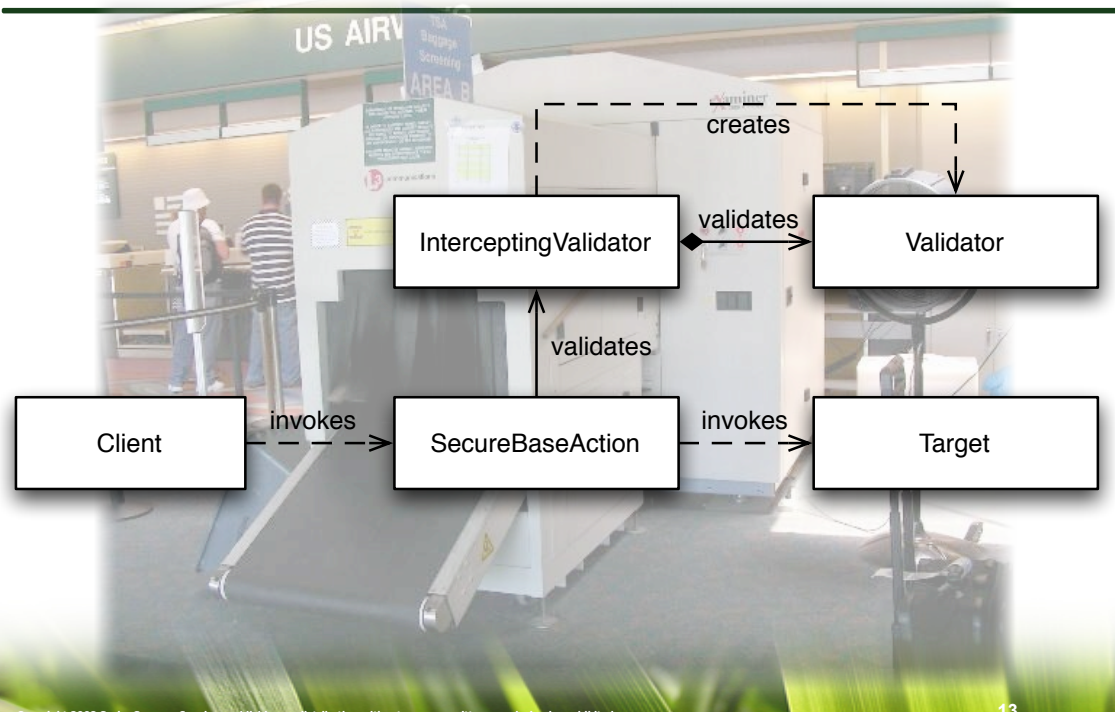
- Maskiert üblicherweise gefährliche Zeichen
- Steht aber so nicht in der JDBC-Spec
- Erwischt evtl. nicht alle
  - Unterschiedliche Zeichensätze
  - Unterschiedliche Versionen
  - Bugs
- Kein Lösung, sondern Out-Sourcing

Abhilfe

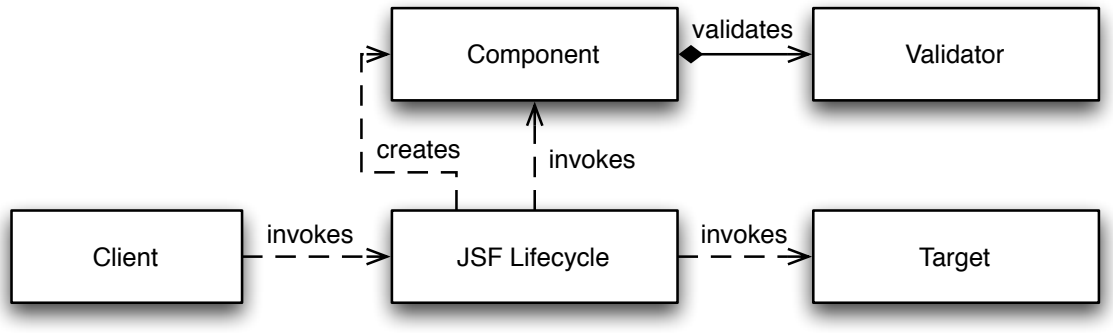
- Blacklist :-(
- Zeichen maskieren :-|
- Whitelist :-)
  - –und dann Zeichen maskieren
- Keine Eingaben erlauben ;-)
  - –z.B. durch Vorselektion

Und wo wird das gemacht?

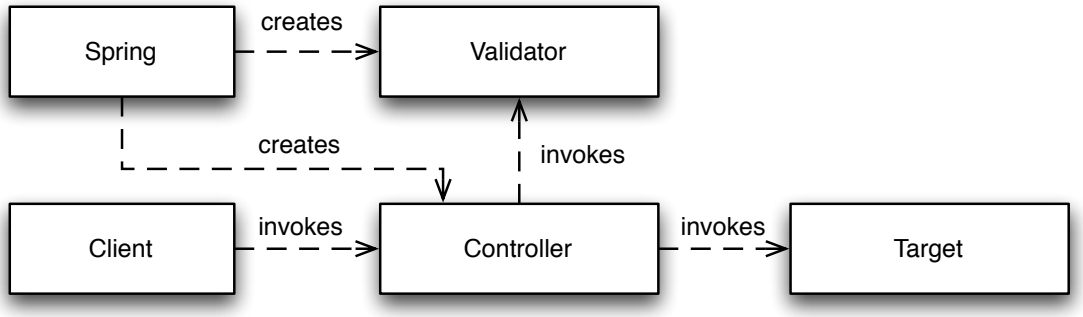
Intercepting Validator



Validation mit JSF



Validation mit Spring



Und ohne Web?

- Viele Validatoren sind Web-Unabhängig (Spring, Commons Validator, ...)
- Alternative:
  - Annotations und AOP

Demo

Validation mit Annotations/AOP

JSR 303: Bean Validation

- Annotationen auf den Domänenobjekten

```
@ZipCodeCityCoherenceChecker
public class Address {

  @NotNull @Length(max=30)
  private String addressline1;

  @Length(max=30)
  private String addressline2;

}
```

JSR 303 - Constraints

```
@Documented
@ValidatorClass(NotNullConstraint.class)
@Target({METHOD, FIELD})
@Retention(RUNTIME)
public @interface NotNull {
  String message() default "{beancheck.notNull}";
  String[] groups() default {};
}
```


JSR-303 - Aufruf

- Set<InvalidConstraint<T>>
  validate(T object, String... groups);
- Oder über einen eigenen Aspekt

JSR-303

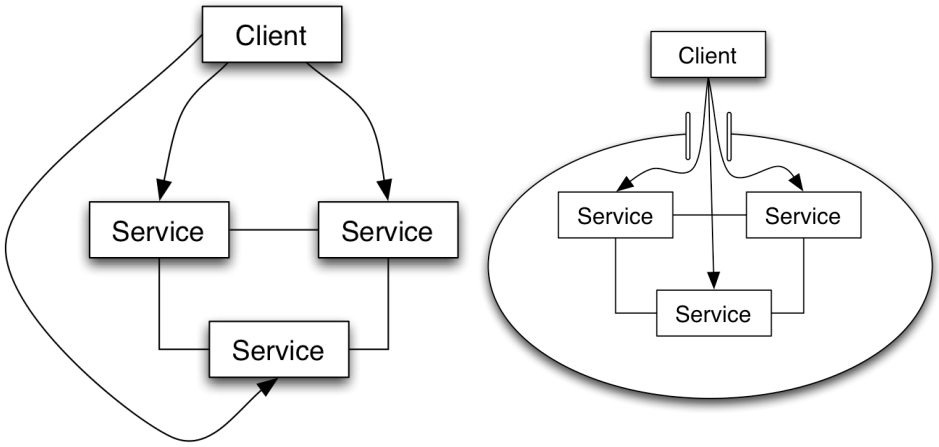
- Validierungsinformationen direkt im Domänenobjekt
- Erweiterbar mit eigenen Annotations
- Aber: Derzeit noch nicht final

Agenda

- Security-Patterns?
- Ungeprüfte Eingaben
- Authentifizierung
- Berechtigungsprüfung
- Vetrauensgrenzen

Single Access Point

Single Access Point



Single Access Point im Web

- Servlet-Filter mit Redirect zum Login
- z.B. mit Spring Security:

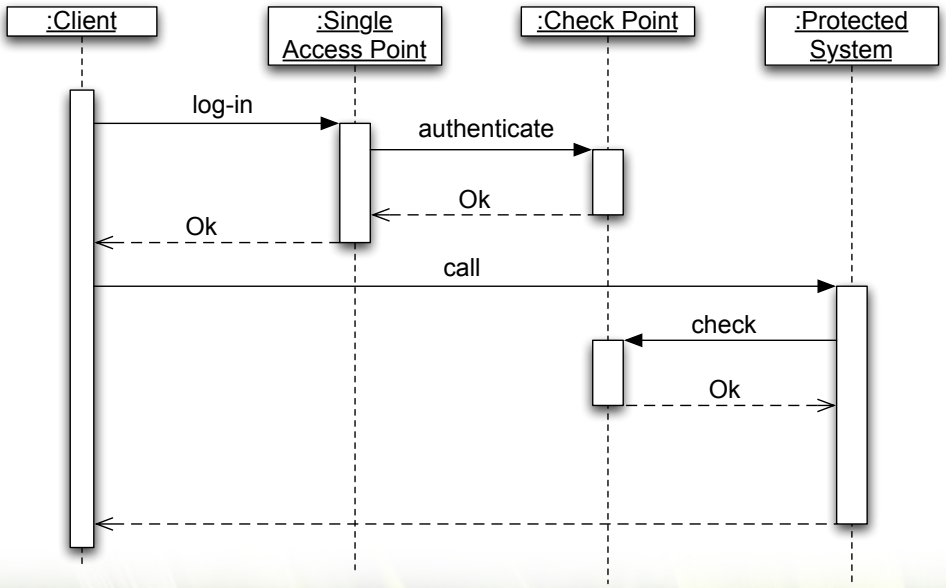
```
<http>
   <intercept-url pattern="/myWebApp/**"
                  access="IS_AUTHENTICATED_FULLY" />

   <form-login login-page="/login.jsp"
               authentication-failure-url="/login.jsp" />
</http>
```


Agenda

- Security-Patterns?
- Ungeprüfte Eingaben
- Authentifizierung
- Berechtigungsprüfung
- Vetrauensgrenzen

Check Point

Check Point



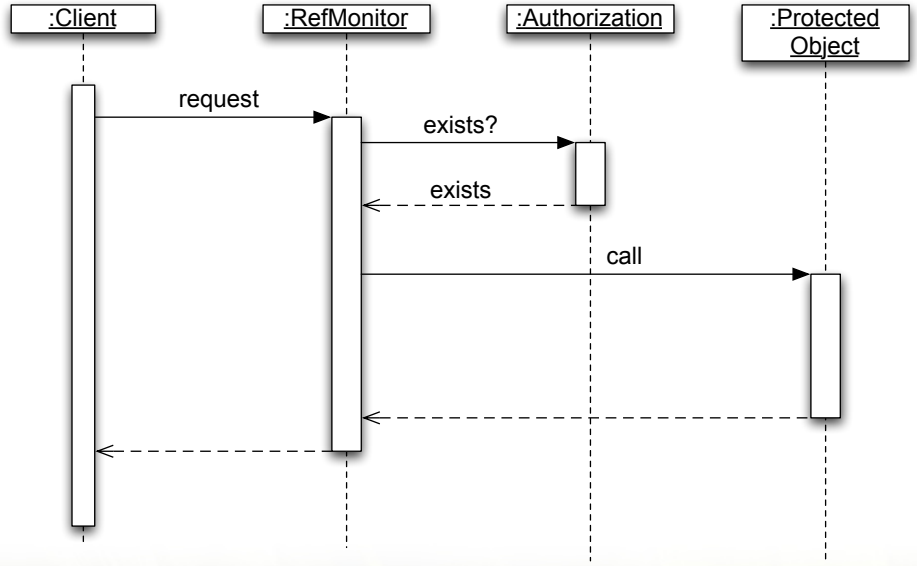
Check Point Code

```
SecurityManager sm = System.getSecurityManager();
if (sm.isUserInRole("ROLE_USER") != true) {
    throw new AccessDeniedException(„No Access“)
}
// do something
```

- Implementierungen müssen vorhanden sein
- Wird immer mitgetestet
- Abhängigkeit zur API
- Kann vergessen werden

Reference Monitor

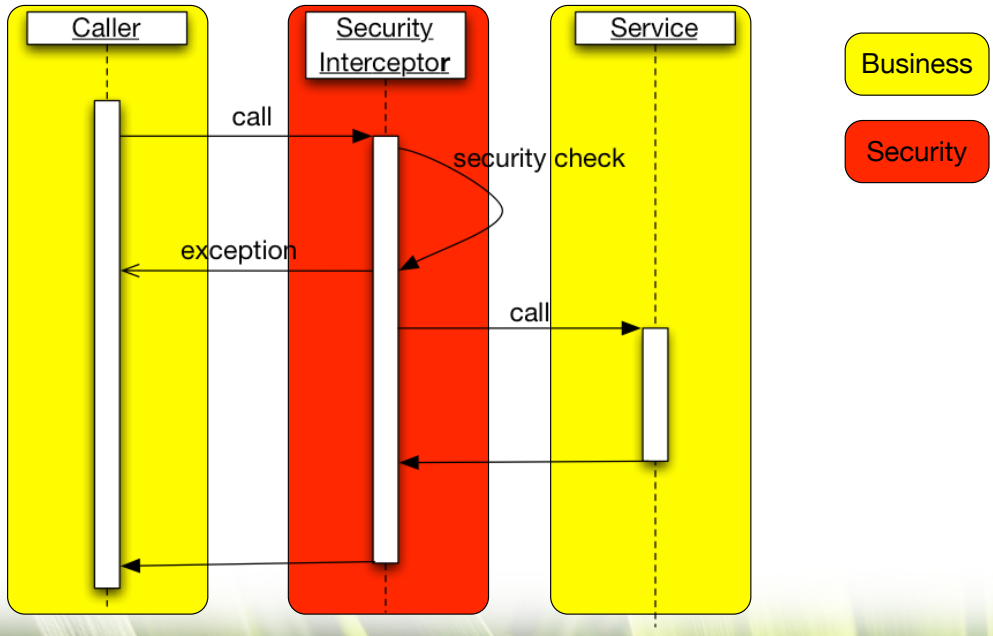
Reference Monitor



Reference Monitor Code

- Command Pattern
  - keine Typsicherheit
- Decorator
  - Muss für jedes Interface erstellt werden

Reference Monitor mit AOP



Code mit Spring Security

```
<global-method-security>
   <protect-pointcut
      expression="execution(* admin.*.*(..))"
      access="PERM_ADMIN_OP"/>
   <protect-pointcut
      expression="execution(* admin.User.delete(..))"
      access="PERM_DELETE_USER"/>
</global-method-security>

package admin
public class User {
  public void delete(User user);
  public void show(long id);
}
```

Code mit Spring Security

```
@Secured("PERM_DELETE_USER")
public void deleteUser(User user);

<global-method-security secured-annotations="enabled"/>
```

... oder ...

JSR-250 Common Annotation

```
@RolesAllowed("PERM_DELETE_USER")
public void deleteUser(User user);

<global-method-security jsr250-annotations="enabled"/>
```


Role-Based Access Control

```
@Secured("ROLE_ADMIN")
public void deleteUser(User user)
```

Wo passiert diese Zuordnung?


```
@Secured("PERM_DELETE_USER")
public void deleteUser(User user)
```

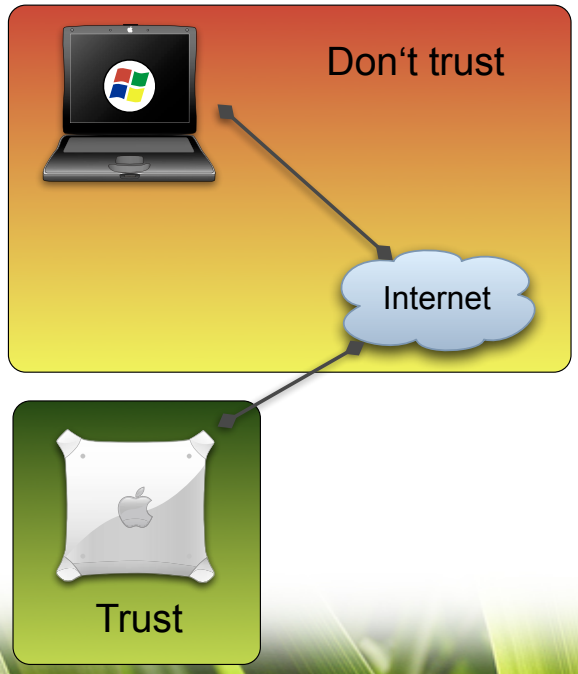
Was ist ein Recht?

- Die Berechtigung eine bestimmte Aktion auszuführen
- I.d.R. pro Use-Case ein Recht
- Code weiß nichts von Rollen
- Neue Rollen können dynamisch angelegt werden
- Rechtezuweisungen zur Laufzeit
- Siehe auch Role Rights Definition Pattern

Agenda

- Security-Patterns?
- Ungeprüfte Eingaben
- Authentifizierung
- Berechtigungsprüfung
- Vetrauensgrenzen

Vertrauensgrenzen



Vertrauensgrenzen

- Prüfungen (Validierung, Berechtigung) immer im Trust-Bereich
- Das gilt nicht nur für Webanwendungen

- Folgendes reicht nicht aus:
  - JavaScript-Validierung
  - Ausblenden von Links/Menüpunkten
  - Berechtigungsprüfung auf dem Client
  - Authentifizierung auf dem Client

FAZIT

Fazit

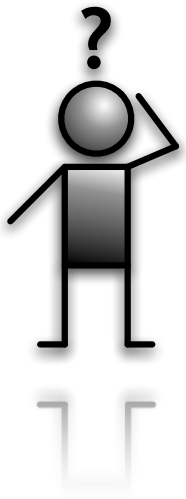
- Security Patterns sind gute Ideengeber
- Pattern sind abstrakt, konkrete Umsetzung erfordert Security Know-How
- Spring Security hilft z.B. bei der Umsetzung
- Hier geht‘s weiter:
  - Silcher-Saal:
    Just-In-Time Security: Sicherheit im Entwicklungsprozess

Fragen?

Mike Wiesner
SpringSource Germany

mike.wiesner@springsource.com
Skype: mikewiesner

http://www.springsource.com/de
http://www.mwiesner.com



Credits

In diesem Vortrag wurde Fotos von folgenden Usern verwendet:

http://www.flickr.com/photos/jurek_durczak/
http://www.flickr.com/photos/skeglovitz/
http://www.flickr.com/photos/willpalmer/
http://www.flickr.com/photos/nedrichards
http://www.flickr.com/photos/freecat
http://www.flickr.com/photos/geminidustin
http://www.flickr.com/photos/crobj