
Symfony

Security Component Bundle

- Bietet Mechanismen zur Authentifizierung und Authorisierung
- Was bedeuten die Begriffe?

Security Component Bundle

- Bietet Mechanismen zur Authentifizierung und Authorisierung
- Was bedeuten die Begriffe?
- Authentifizierung klärt wer der Benutzer ist.

Security Component Bundle

- Bietet Mechanismen zur Authentifizierung und Authorisierung
- Was bedeuten die Begriffe?
- Authentifizierung klärt wer der Benutzer ist.
- Authorisierung klärt, ob der Benutzer das Recht hat, eine bestimmte Aktion auszuführen

Grundbegriffe

- Encoder: Ein Encoder wird verwendet um Passworte zu hashen und zu vergleichen

Grundbegriffe

- Encoder: Ein Encoder wird verwendet um Passworte zu hashen und zu vergleichen
- Provider: Ein Provider stellt User Objekte bereit

Grundbegriffe

- Encoder: Ein Encoder wird verwendet um Passworte zu hashen und zu vergleichen
- Provider: Ein Provider stellt User Objekte bereit
- Firewall: Eine Firewall definiert einen Authentifizierungs-Workflow für einen Teil der Applikation oder auch die ganze Applikation

Grundbegriffe

- Encoder: Ein Encoder wird verwendet um Passworte zu hashen und zu vergleichen
- Provider: Ein Provider stellt User Objekte bereit
- Firewall: Eine Firewall definiert einen Authentifizierungs-Workflow für einen Teil der Applikation oder auch die ganze Applikation
- Access Control: Access Control Regeln legen Rollen-Anforderungen für bestimmte Teile der Applikation fest.

Encoder

- Encoder: Ein Encoder wird verwendet um Passworte zu hashen und zu vergleichen

```
namespace Symfony\Component\Security\Core\Encoder;

interface PasswordEncoderInterface
{
    function encodePassword($raw, $salt);
    function isPasswordValid($encoded, $raw, $salt);
}
```

Encoder konfigurieren

```
# app/config/security.yml
security:
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
    Mvcokg\UserBundle\User: plaintext
    Sensio\UserBundle\User: sha512
    Liip\UserBundle\User:
      algorithm: sha512
      encode_as_base64: true
      iterations: 5000
    Mvcokg\UserBundle\User:
      id: eigener.encoder.service.id
```

Genau ein(!) Encoder für jeden Provider

Encoder verwenden

- Ein Passwort hashen

```
$user = new User();
$factory = $this->get('security.encoder_factory');
$encoder = $factory->getEncoder($user);
$pwd = $encoder->encodePassword('IsdiwU9', 'salt');
$user->setPassword($pwd);
```

- Ein Passwort vergleichen

```
$encoder->isPasswordValid('e5e[...]ca5', 'IsdiwU9',
                                        'salt');
```

User Provider

- Ein Provider stellt User Objekte zur Verfügung

Eingebaute User Provider

- in-memory
- chain
- entity
- propel

Eingebaute User Provider

- in-memory: Liest Konfigurationsdatei (security.yml)
- chain
- entity
- propel

Eingebaute User Provider

- in-memory: Liest Konfigurationsdatei (security.yml)
- chain: Liest nacheinander aus mehreren Providern
- entity
- propel

Eingebaute User Provider

- in-memory: Liest Konfigurationsdatei (security.yml)
- chain: Liest nacheinander aus mehreren Providern
- entity: Liest aus einem Doctrine Entity Model
- propel

Eingebaute User Provider

- in-memory: Liest Konfigurationsdatei (security.yml)
- chain: Liest nacheinander aus mehreren Providern
- entity: Liest aus einem Doctrine Entity Model
- propel: Liest aus einem Propel Active Record Model

~~Eingebaute~~ User Provider

- in-memory: Liest Konfigurationsdatei (security.yml)
- chain: Liest nacheinander aus mehreren Providern
- entity: Liest aus einem Doctrine Entity Model
- propel: Liest aus einem Propel Active Record Model
- fos_userbundle: Liest aus Doctrine, MongoDB, CouchDB oder Propel

User Provider Interface

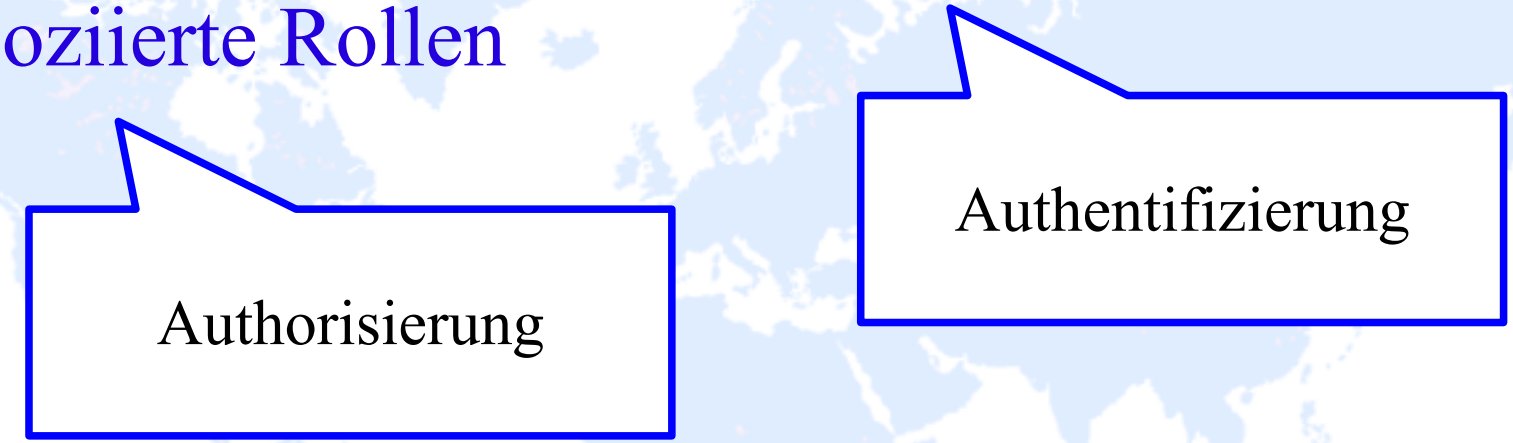
```
interface UserProviderInterface
{
    // Loads the user for the given username.
    function loadUserByUsername($username);

    // Refreshes the user properties like credentials.
    function refreshUser(UserInterface $user);

    // Whether this provider supports the given user class
    function supportsClass($class);
}
```

User im Security Layer

- User Objekte speichern Zugangsdaten und assoziierte Rollen

User im Security Layer

- User Objekte speichern Zugangsdaten und assoziierte Rollen



Das User Interface

```
interface UserInterface
{
    function getRoles();
    function getPassword();
    function getSalt();
    function getUsername();
    function eraseCredentials();
}
```

Das Advanced User Interface

```
interface AdvancedUserInterface extends UserInterface
{
    function isEnabled();
    function isCredentialsNonExpired();
    function isAccountNonLocked();
    function isAccountNonExpired();
}
```

Das Advanced User Interface

```
interface AdvancedUserInterface extends UserInterface
{
    function isEnable  ();
    function isCreden   lsNonExpired();
    function isAccount    cked();
    function isAccountN     ed();
}
```

Die BaseUser Klasse des FOS-Userbundles implementiert dieses Interface!

In Memory Provider

```
# app/config/security.yml
security:
  providers:
    in_memory:
      memory:
        users:
          timon:  { password: sd, roles: ['ROLE_ADMIN' ] }
```

Doctrine Entity Provider

```
# app/config/security.yml
security:
  providers:
    in_db:
      entity: { class: MvcokgErpBundle:User, property: username }
```

Chain Provider

```
# app/config/security.yml
security:
  providers:
    chain_provider:
      chain:
        providers: [in_memory, in_db]
      mvcokg_in_memory:
        memory:
          users:
            timon:  { password: sd, roles: ['ROLE_ADMIN' ] }
      mvcokg_in_db:
        entity: { class: MvcokgErpBundle:User, property: username }
```

Übermittlung der Zugangsdaten zur Firewall

- http-basic: HTTP authentication
- http-digest: HTTP authentication mit clientseitig gehashtem Passwort
- x.509: Verwendet ein x.509 Zertifikat
- form-based: Verwendet ein Formular um Benutzername und Passwort abzufragen

Firewall

```
# app/config/security.yml
security:
# ...
  firewalls:
    admin:
      pattern:    ^/admin
      provider:   in_memory
      form_login:
         check_path: /admin/auth
         login_path: /login
         default_target_path: /admin
         always_use_default_target_path: true
```

Authentifizierungs Workflow

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad

Firewall

```
# app/config/security.yml
security:
# ...
    firewalls:
        admin:
            provider:    in_memory
            pattern:     ^/admin
            form_login:
                check_path: /admin/check
                login_path: /login
        general:
            pattern:     ^/cms # ...
```

Start mit Check Path, Firewall finden, Provider finden

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft.

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

In Memory Provider

```
# app/config/security.yml
security:
  providers:
    in_memory:
      memory:
        users:
          timon:  { password: pAsswrt, roles: ['ROLE_ADMIN' ] }
```

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

3) Mit dem UserProvider ist genau ein Encoder verknüpft:

Encoder konfigurieren

```
# app/config/security.yml
security:
  encoders:
    Symfony\Component\Security\Core\User\User: plaintext
    Mvcokg\UserBundle\User: plaintext
    Sensio\UserBundle\User: sha512
    Liip\UserBundle\User:
      algorithm: sha512
      encode_as_base64: true
      iterations: 5000
    Mvcokg\UserBundle\User:
      id: eigener.encoder.service.id
```

memory Provider ist ein Spezialfall, hat immer diese Klasse.

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

3) Mit dem UserProvider ist genau ein Encoder verknüpft:

```
$encoder = $factory->getEncoder($user)
```

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

3) Mit dem UserProvider ist genau ein Encoder verknüpft:

```
$encoder = $factory->getEncoder($user)
```

4) Mit dem Encoder prüft die Firewall das Passwort

```
$encoder->isPasswordValid("hash","pAsswrt","salt")
```

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

3) Mit dem UserProvider ist genau ein Encoder verknüpft:

```
$encoder = $factory->getEncoder($user)
```

4) Mit dem Encoder prüft die Firewall das Passwort

```
$encoder->isPasswordValid("hash","pAsswrt","salt")
```

5) Passwort korrekt. Weiterleitung entspr. Firewall Config:

Firewall

```
# app/config/security.yml
security:
# ...
    firewalls:
        admin:
            pattern:    ^/admin
            provider:   in_memory
            form_login:
                check_path: /admin/check
                login_path: /login
                default_target_path: /admin
                always_use_default_target_path: true
```

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in deren Geltungsbereich /login-check liegt, ist genau ein UserProvider verknüpft:

```
$user = $provider->loadUserByUsername("timon")
```

3) Mit dem UserProvider ist genau ein Encoder verknüpft:

```
$encoder = $factory->getEncoder($user)
```

4) Mit dem Encoder prüft die Firewall das Passwort

```
$encoder->isPasswordValid("pAsswrt")
```

5) Passwort korrekt. Weiterleitung. Ziel Erreicht.

Authentifizierungs Workflow

1) Timon füllt Login-Formular aus und sendet "timon" + "pAsswrt" an Login-check Pfad
2) Mit der Firewall, in de[illegible] Geltungsbereich /login-check liegt, ist genau ein User[illegible]der verknüpft:

```
$user = $provider->
```

3) Mit dem UserProvide[illegible]ipft:

```
$encoder = $factory
```

4) Mit dem Encoder prü[illegible]
```
$encoder->isPasswordValid("pAsswrt")
```

5) Passwort korrekt. Weiterleitung. Ziel Erreicht.

Damit der Benutzer auf diesen Pfad im Bereich der Firewall zugreifen kann gibt es `IS_AUTHENTICATED_ANONYMOUSLY` Andere Pfade fordern höhere Rollen.

Rollenhierarchie

```
# app/config/security.yml
security:
    role_hierarchy:
        ROLE_ADMIN:       ROLE_USER
        ROLE_TRAINER:     ROLE_USER
        ROLE_SUPERADMIN:
            - ROLE_USER
            - ROLE_ADMIN
            - ROLE_ALLOWED_TO_SWITCH
```

Access Control

- Access Control Regeln legen Rollen-Anforderungen (und ggf. weitere Anforderungen) für bestimmte Teile der Applikation fest.

```
# app/config/security.yml
security:
  access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    - { path: ^/admin, roles: [ROLE_ADMIN, ROLE_MANAGER] }
    - { path: ^/,      roles: ROLE_USER}
```

Access Control

- Access Control Regeln legen Rollen-Anforderungen (und ggf. weitere Anforderungen) für bestimmte Teile der Applikation fest.

```
# app/config/security.yml
security:
  access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    -
      path: ^/admin
      roles: [ROLE_ADMIN, ROLE_MANAGER]
    -
      path: ^/
      roles: ROLE_USER
```

Access Control

```
# app/config/security.yml
security:
  access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    -
      path: ^/admin
      roles: [ROLE_ADMIN, ROLE_MANAGER]
    -
      path: ^/
      roles: ROLE_USER
```

Access Control

```
# app/config/security.yml
security:
  access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    -
      path: ^/admin
      roles: [ROLE_ADMIN, ROLE_MANAGER]
    -
      path: ^/
      roles: ROLE_USER
```

Access Control

```
# app/config/security.yml
security:
  access_control:
    - { path: ^/login, roles: IS_AUTHENTICATED_ANONYMOUSLY}
    -
      path: ^/admin
      ip: 192.169.24.0/8
      roles: [ROLE_ADMIN, ROLE_MANAGER]
    -
      path: ^/admin
      roles: ROLE_ADMIN
      requires_channel: https
    -
      path: ^/
      roles: ROLE_USER
```

Rollenbasiert agieren: Controller

```
public function editAction($id)
{
  $securityContext = $this->get('security.context');
  if (!$securityContext->isGranted('ROLE_ADMIN') {
    throw new AccessDeniedException([...]);
  }
  // Zugriff gewährt ...
}
```

Rollenbasiert agieren: Twig

```
{% if is_granted("ROLE_ADMIN") %}
    <a href="...">Delete</a>
{% endif %}
```

Benutzer ermitteln

- Benutzer abfragen im Controller

```
$context = $this->get('security.context');
$name = $context->getToken()->getUsername();
```

- Benutzer abfragen im Twig Template

```
{{ app.user.username }}
```

Vielen Dank
für Eure Aufmerksamkeit!


Fragen?
Ideen, Wünsche, Anmerkungen?