# Spring Security

# 3 Key Areas in Security

- Authentication
- Authorization
- Exception Handling

# User Database

| Username | Password | Authorities |
|---|---|---|
| joe@example.com | password | ROLE_USER |
| rob@example.com | password | ROLE_USER |
| admin@example.com | password | ROLE_USER, ROLE_ADMIN |

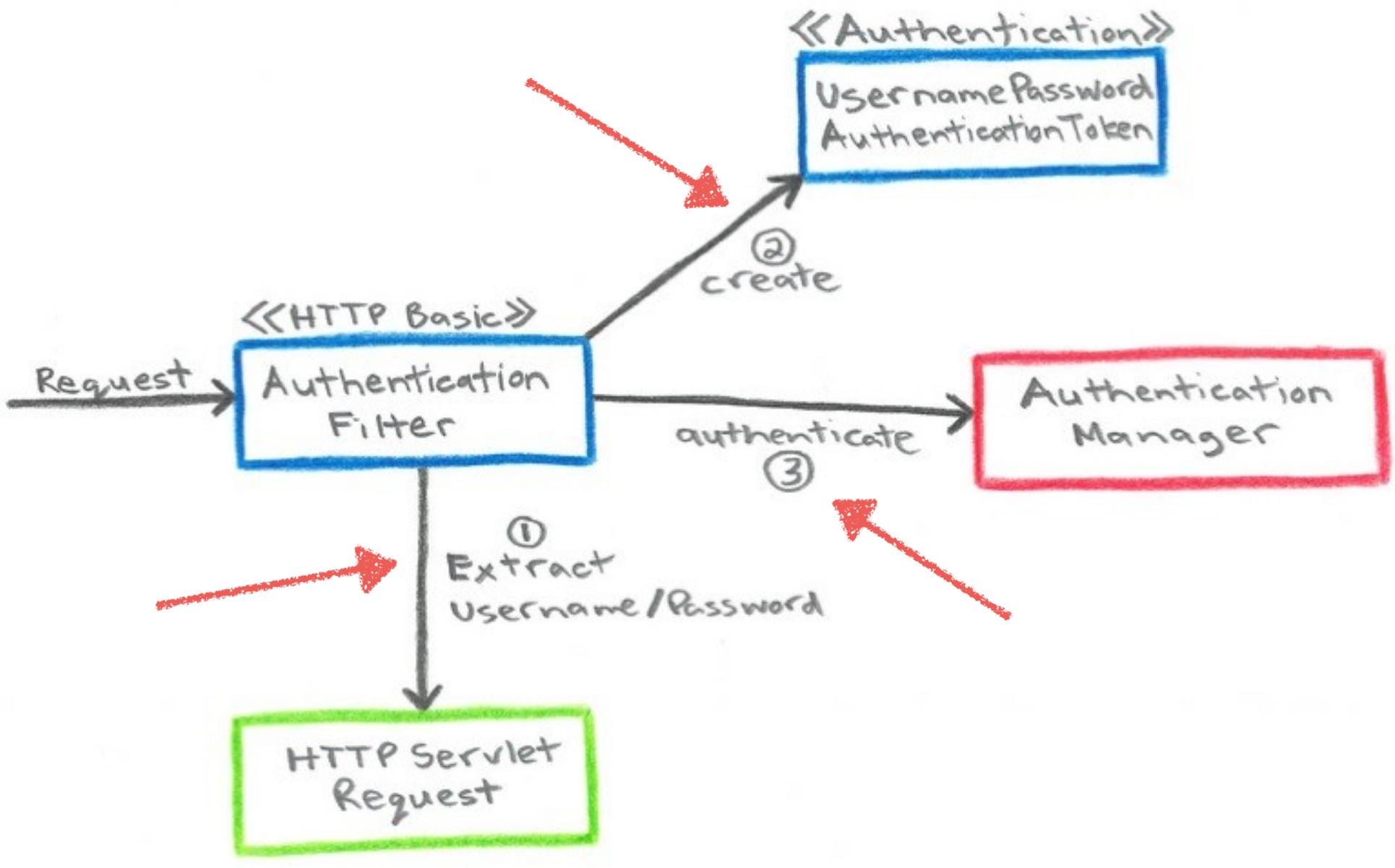
# Demo

**Authentication**

Authorization

Exception Handling

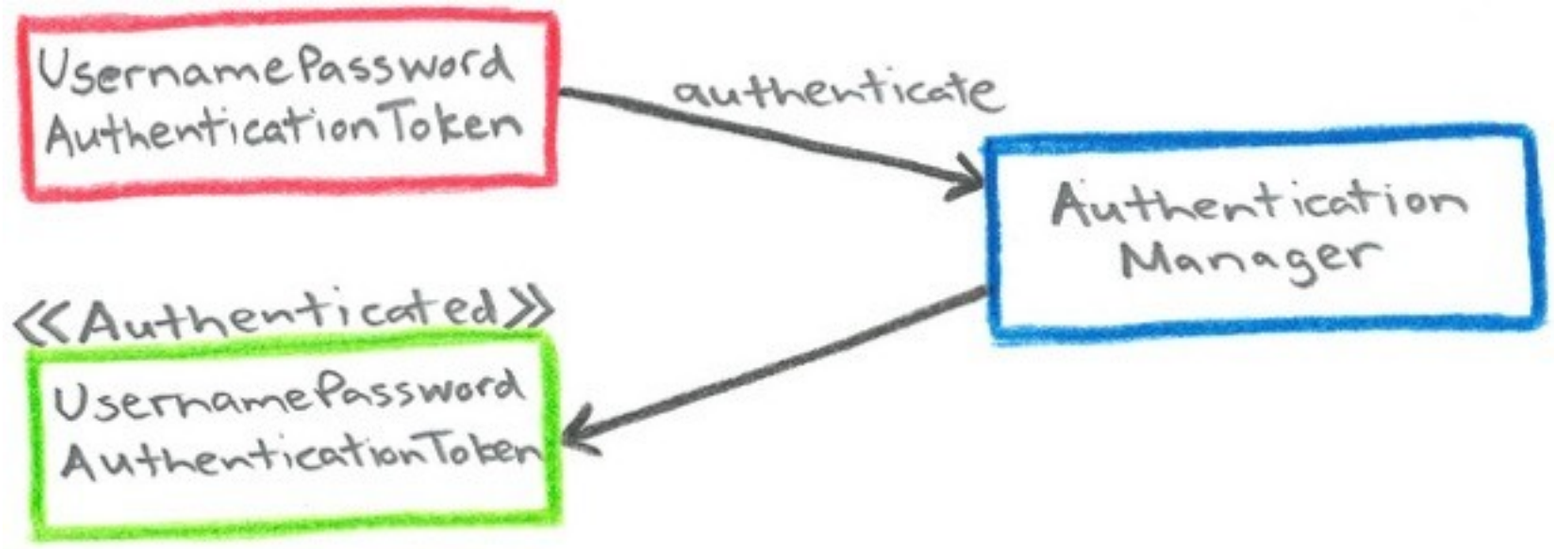
# Authentication Filter



# Authentication

| Authentication | |
|---|---|
| **Principal:** | joe@example.com |
| **Credentials:** | password |
| **Authorities:** | —— |
| **Authenticated:** | FALSE |

| Authentication | |
|---|---|
| **Principal:** | UserDetails |
| **Credentials:** | —— |
| **Authorities:** | ROLE_USER |
| **Authenticated:** | TRUE |




```
public interface Authentication extends Principal, Serializable
{
   Object getPrincipal();

   Object getCredentials();

   Collection<? extends GrantedAuthority> getAuthorities();

   . . .
}
```

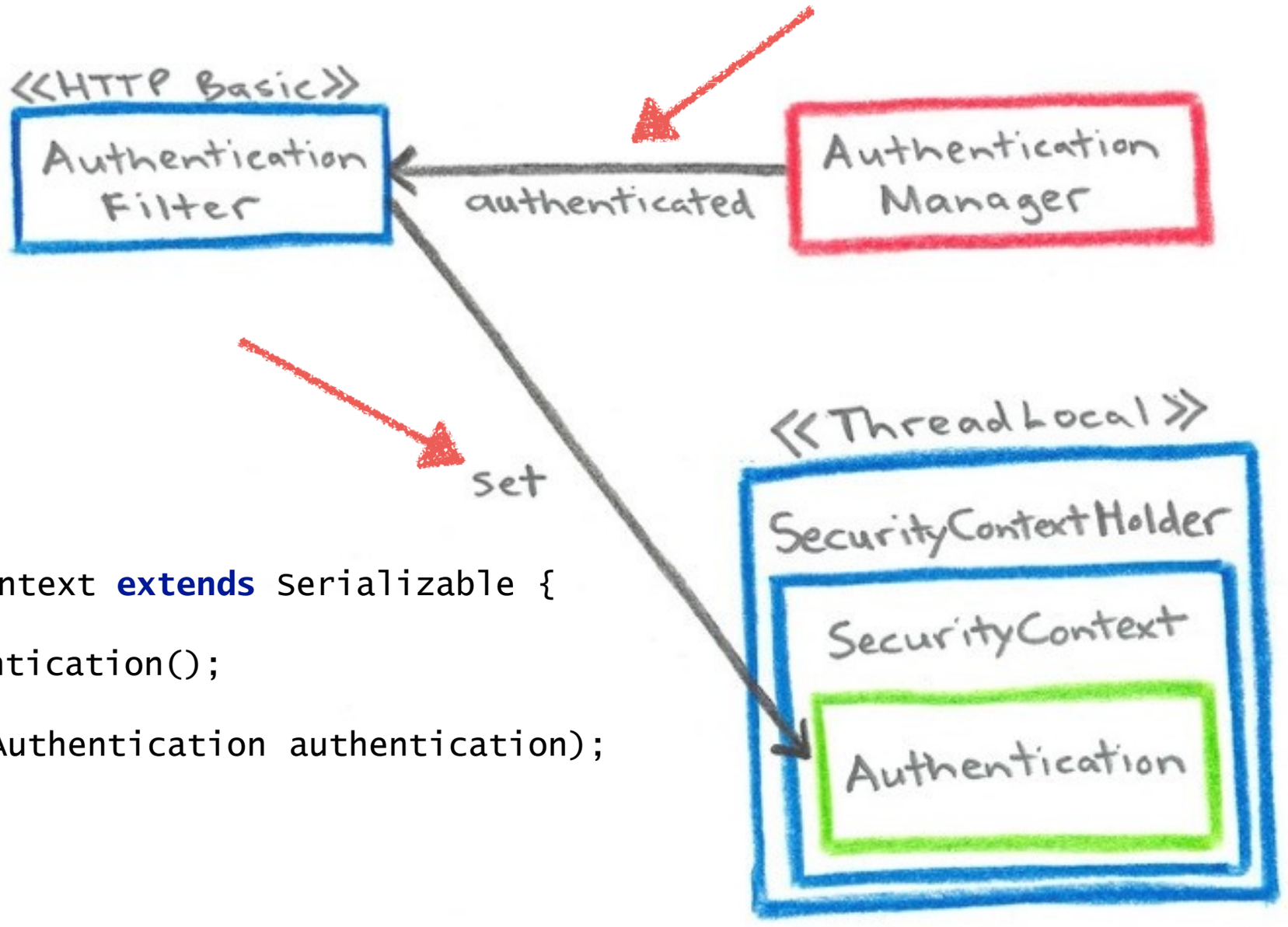
# UserDetails / Service

```
public interface UserDetailsService {

   UserDetails loadUserByUsername(String username)
         throws UsernameNotFoundException;

}
```


| Authentication | |
|---|---|
| **Principal:** | UserDetails |
| **Credentials:** | —— |
| **Authorities:** | ROLE_USER |
| **Authenticated:** | TRUE |

```
public interface UserDetails extends Serializable {

   String getUsername();

   String getPassword();

   Collection<? extends GrantedAuthority>
   getAuthorities();

   . . .
}
```

# Security Context




```
public interface SecurityContext extends Serializable {

    Authentication getAuthentication();

    void setAuthentication(Authentication authentication);

}
```

```
SecurityContextHolder.getContext().setAuthentication(authenticated);
```

# Authentication Recap

- **Authentication Filter** creates an *"Authentication Request"* and passes it to the **Authentication Manager**

- Authentication Manager delegates to the **Authentication Provider**

- Authentication Provider uses a **UserDetailsService** to load the **UserDetails** and returns an *"Authenticated Principal"*

- Authentication Filter sets the **Authentication** in the **SecurityContext**

Authentication

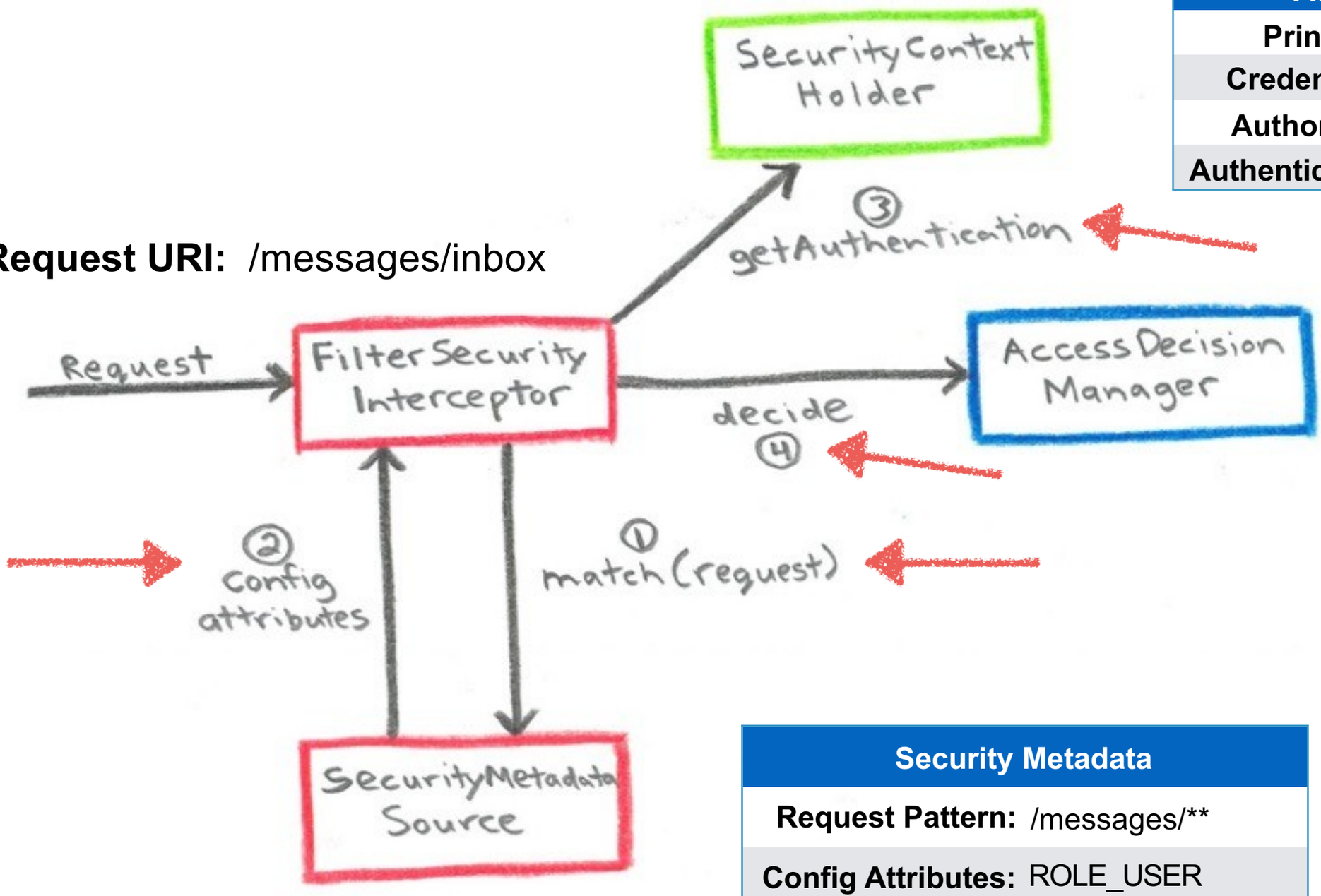
**Authorization**

Exception Handling

# Filter Security Interceptor

| Authentication | |
|---|---|
| **Principal:** | UserDetails |
| **Credentials:** | —— |
| **Authorities:** | ROLE_USER |
| **Authenticated:** | TRUE |

**Request URI:** /messages/inbox



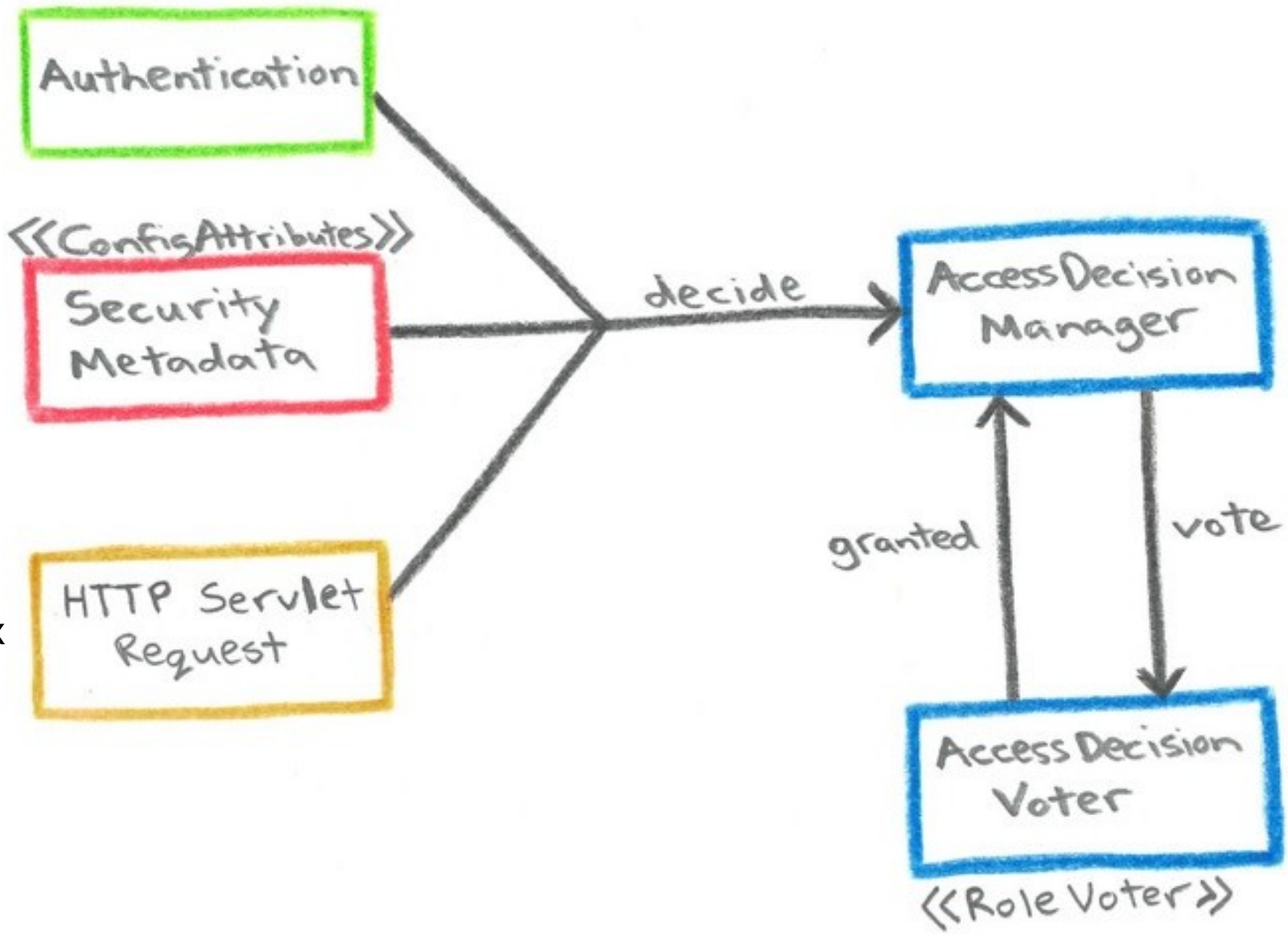

| Security Metadata | |
|---|---|
| **Request Pattern:** | /messages/** |
| **Config Attributes:** | ROLE_USER |

# Access Decision

| Authentication | |
|---|---|
| **Principal:** | UserDetails |
| **Credentials:** | —— |
| **Authorities:** | ROLE_USER |
| **Authenticated:** | TRUE |

| Security Metadata | |
|---|---|
| **Request Pattern:** | /messages/** |
| **Config Attributes:** | ROLE_USER |

**Request URI:** /messages/inbox



# Authorization Recap

- **FilterSecurityInterceptor** obtains the ***"Security Metadata"*** by matching on the current request

- FilterSecurityInterceptor gets the current **Authentication**

- The Authentication, Security Metadata and Request is passed to the **AccessDecisionManager**

- The AccessDecisionManager delegates to it's **AccessDecisionVoter(s)** for decisioning

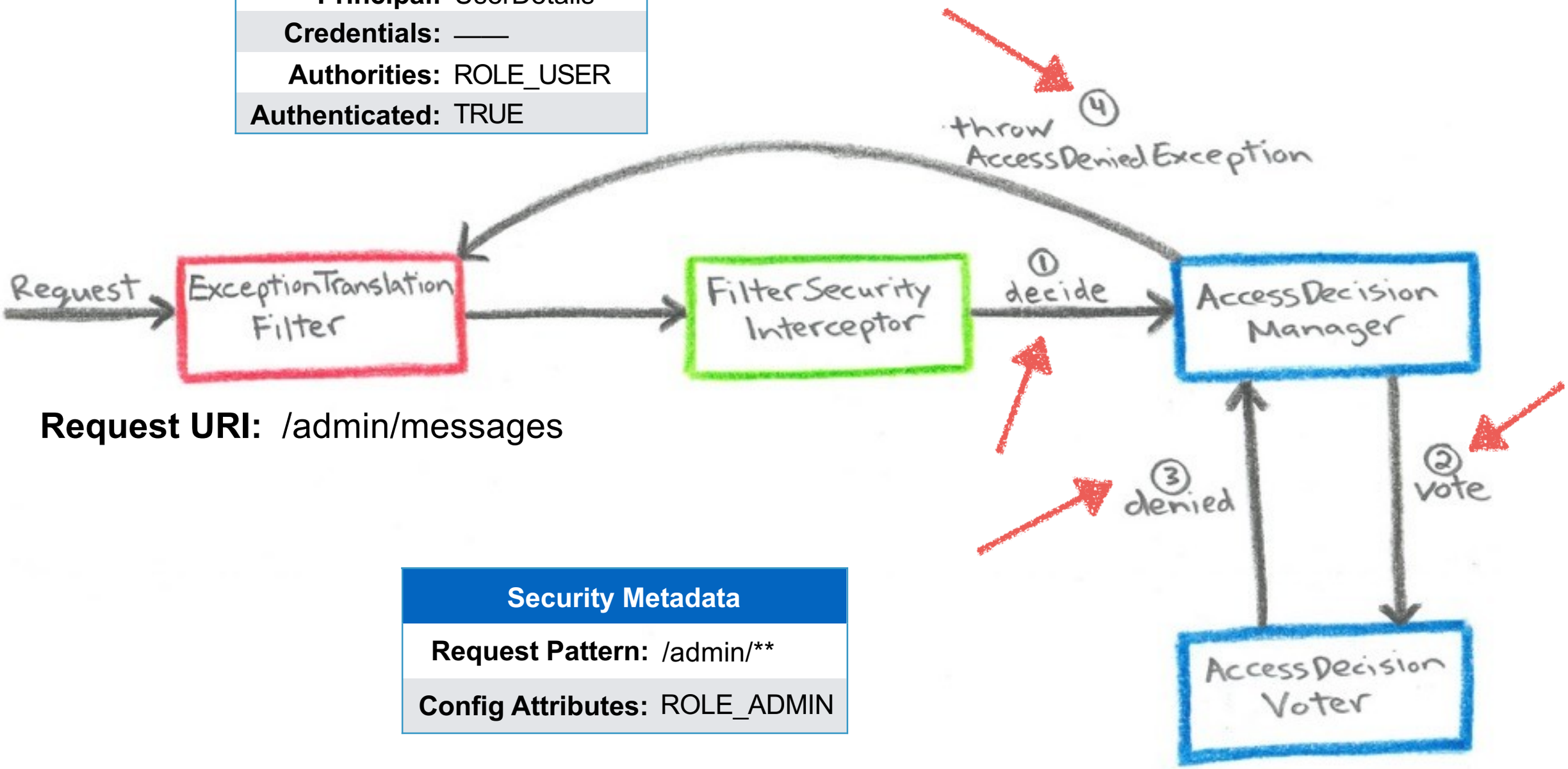
Authentication

Authorization

# Exception Handling

# Access Denied

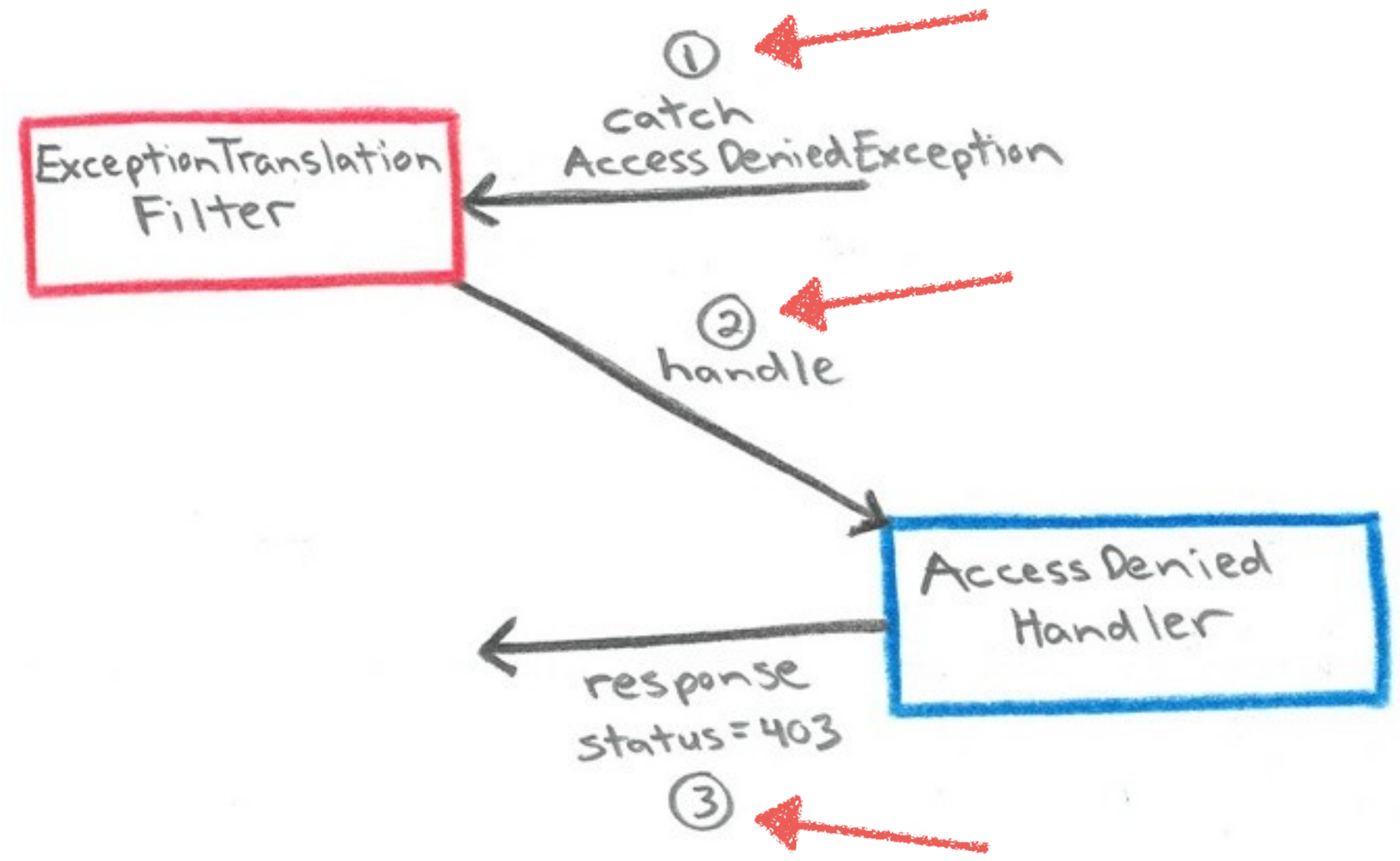
| Authentication | |
|---|---|
| **Principal:** | UserDetails |
| **Credentials:** | —— |
| **Authorities:** | ROLE_USER |
| **Authenticated:** | TRUE |




**Request URI:** /admin/messages

| Security Metadata | |
|---|---|
| **Request Pattern:** | /admin/** |
| **Config Attributes:** | ROLE_ADMIN |

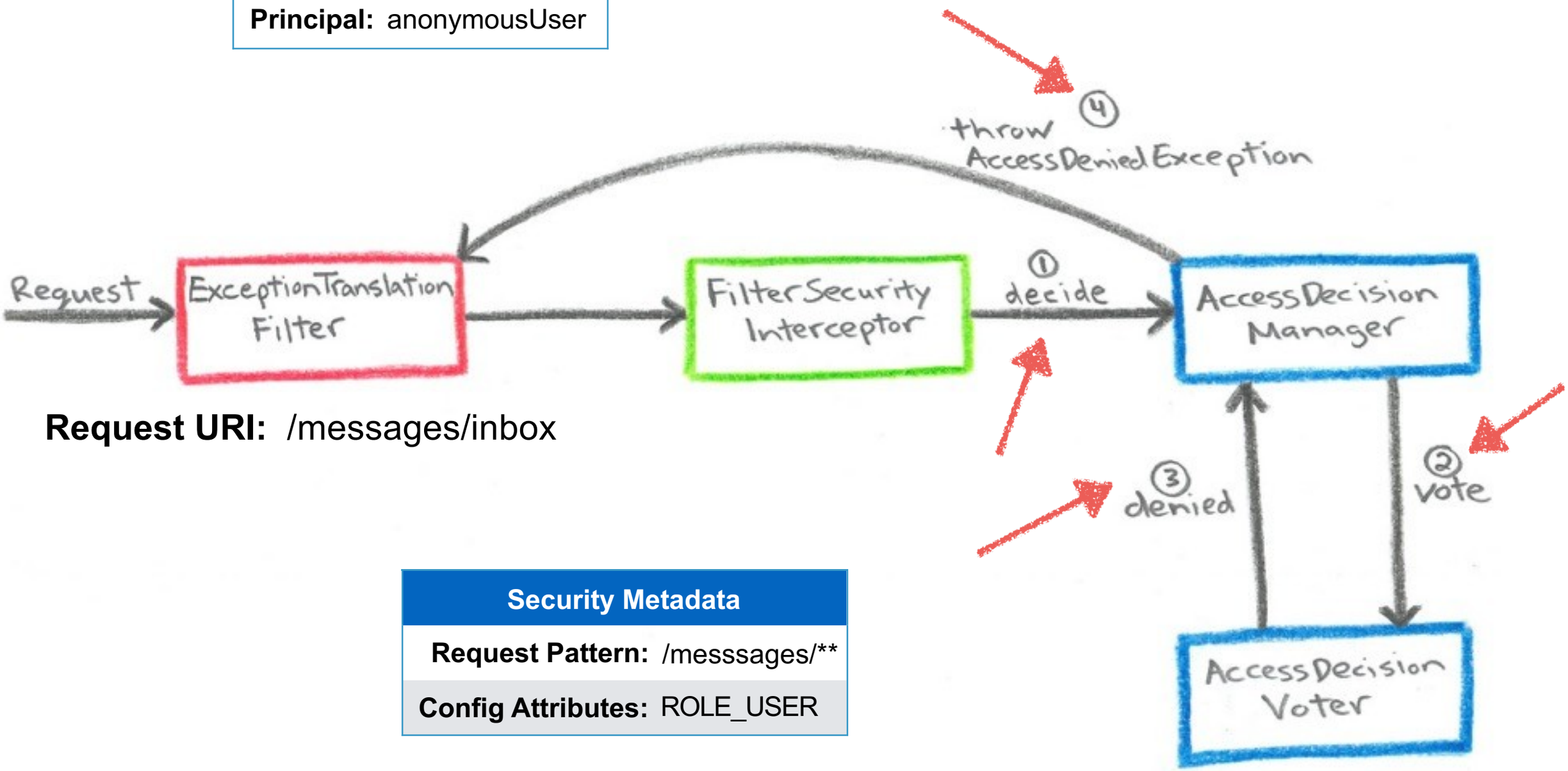
# Access Denied Handler




```
public interface AccessDeniedHandler {

  void handle(HttpServletRequest request, HttpServletResponse response,
       AccessDeniedException accessDeniedException) throws IOException, ServletException;

}
```

# “Unauthenticated”

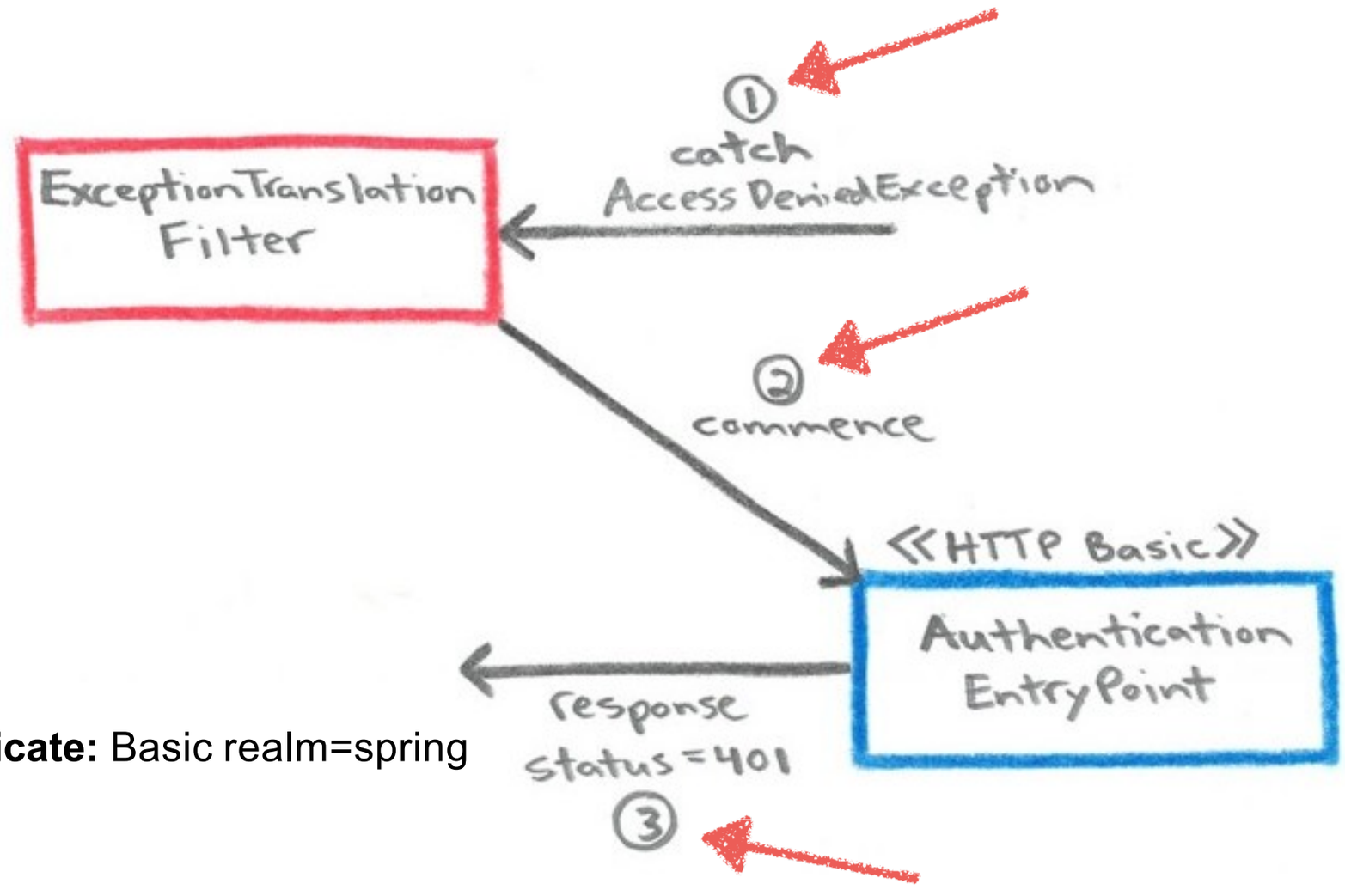
| Authentication |
| --- |
| **Principal:** anonymousUser |




**Request URI:** /messages/inbox

| Security Metadata |
| --- |
| **Request Pattern:** /messsages/** |
| **Config Attributes:** ROLE_USER |

# Start Authentication




**WWW-Authenticate:** Basic realm=spring

```
public interface AuthenticationEntryPoint {

    void commence(HttpServletRequest request, HttpServletResponse response,
            AuthenticationException authException) throws IOException, ServletException;

}
```

# Exception Handling Recap

- When *"Access Denied"* for current Authentication, the
- **ExceptionTranslationFilter** delegates to the **AccessDeniedHandler**, which by default, returns a 403  Status.

- When current Authentication is *"Anonymous"*, the **ExceptionTranslationFilter** delegates to the **AuthenticationEntryPoint** to start the Authentication process.

Summary

Authentication

Authorization

Exception Handling

# Oauth 2

- Protocol and framework for providing access to HTTP services
- Often used for third-party access
- Can be used for system-to-system communications in standalone or on behalf of a user

# Parts of OAuth2

- Resource owner - often the user
- Client - application requesting access
- Resource server - hosts protected data and accounts
- Authorization server - service that grants tokens

# Token Types

- Access token - the secret and often short-lived token that identifies a user
- Refresh token - longer-lived token used to renew access token when it expires
- Scopes provide for rights associated with the access token

# Grants

- Several grant types that impact flows
- Authorization code grant is most common
- Implicit is common in web apps and mobile apps
- Client credentials grant is useful in system-to-system comms

# Spring Support for OAuth

# CommonOAuth2Provider

- Provides native support for Okta, Google, GitHub, and Facebook
- Property-based configuration in Spring Boot
- Client-side OAuth integration

# Authorization Server

- Provides authorization services to the system
- @EnableAuthorizationServer
- AuthorizationServerConfigurerAdapter used to configure it
- Supports various grant types

# Resource Server

- Provides the resources being protected
- @EnableResourceServer

# OAuth Client

- Full client-side support
- @EnableOauth2Client
- Oauth2RestTemplate provides much of the scaffolding
- Supports various grant types