springone 2GX
WASHINGTON DC
OCTOBER 15-18, 2012

Getting Started with Spring Security 3.1

Rob Winch, Sr Software Engineer, Spring Security Lead

@rob_winch

springone 2GX

About Me

- Spring Security Lead at Spring Source / VMware
- 15+ years web experience
- 10+ years of Java experience
- Previous Employment
  - Health Care Security at Cerner (7 years)
  - Grid Computing at Argonne National Labs (1 year)
  - Proteomics Research at Loyola University Chicago (1 year)
  - Started professional career as PERL contractor in High School

Agenda

- What is Spring Security
- Setting up Spring Security
- How the basic Spring Security flow works
- Simple customizations of Spring Security
  - Log In Page
  - Custom Log In Controller
- Global Method Security
- Q&A

Tell me about Spring Security

- Formerly known as Acegi Security
- Authentication
  - Database, LDAP, CAS, OpenID, Pre-Authentication, custom, etc
- Authorization
  - URL based, Method Based (AOP)
- Extensions
- Simple yet powerful

Java Servlet Filter Review - Dispatcher

web.xml

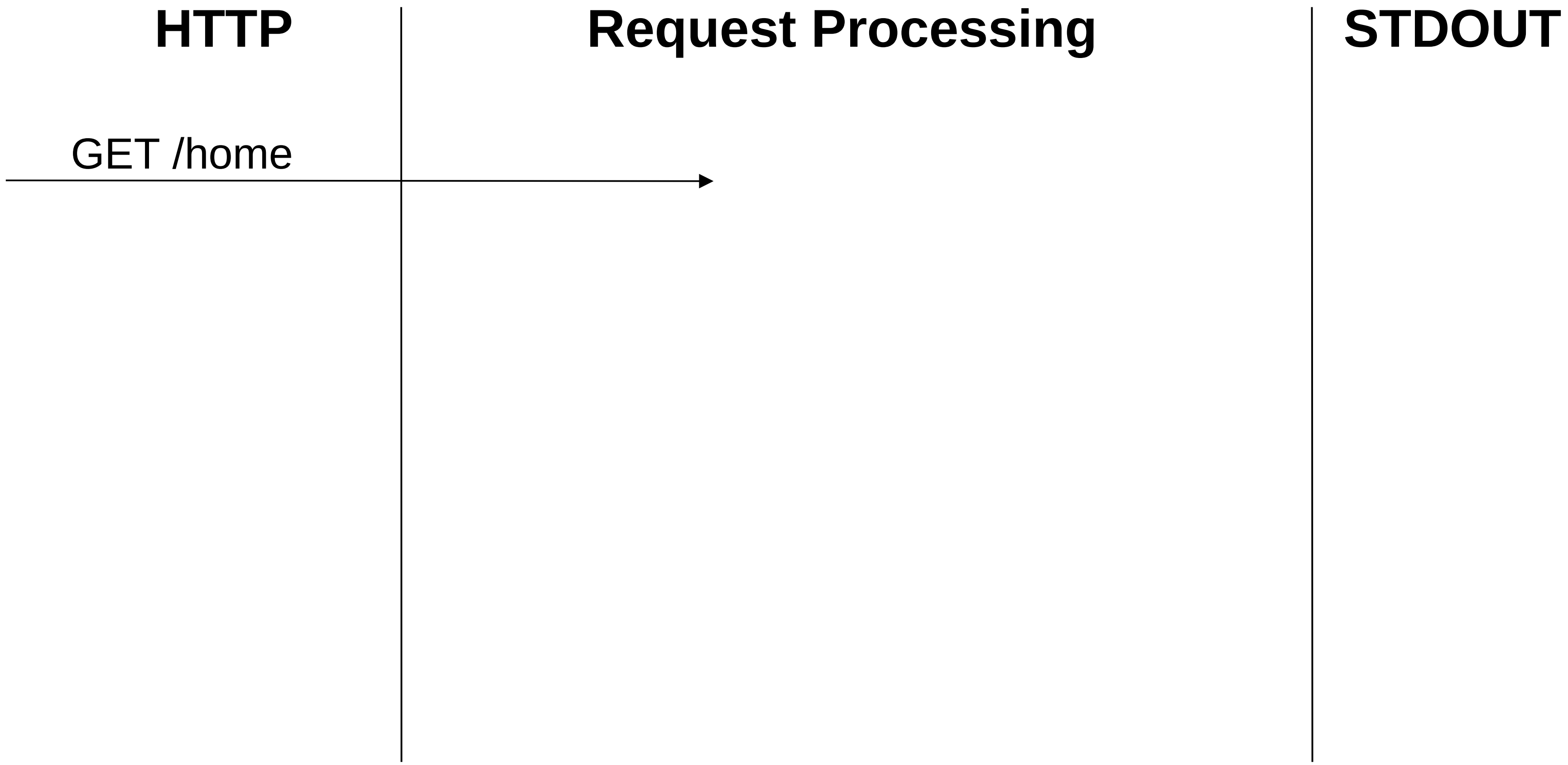
```
<filter>
  <filter-name>filter1</filter-name>
  <filter-class>Filter1</filter-class>
</filter>
<filter-mapping>
  <filter-name>filter1</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

Java Servlet Filter Review - Dispatcher

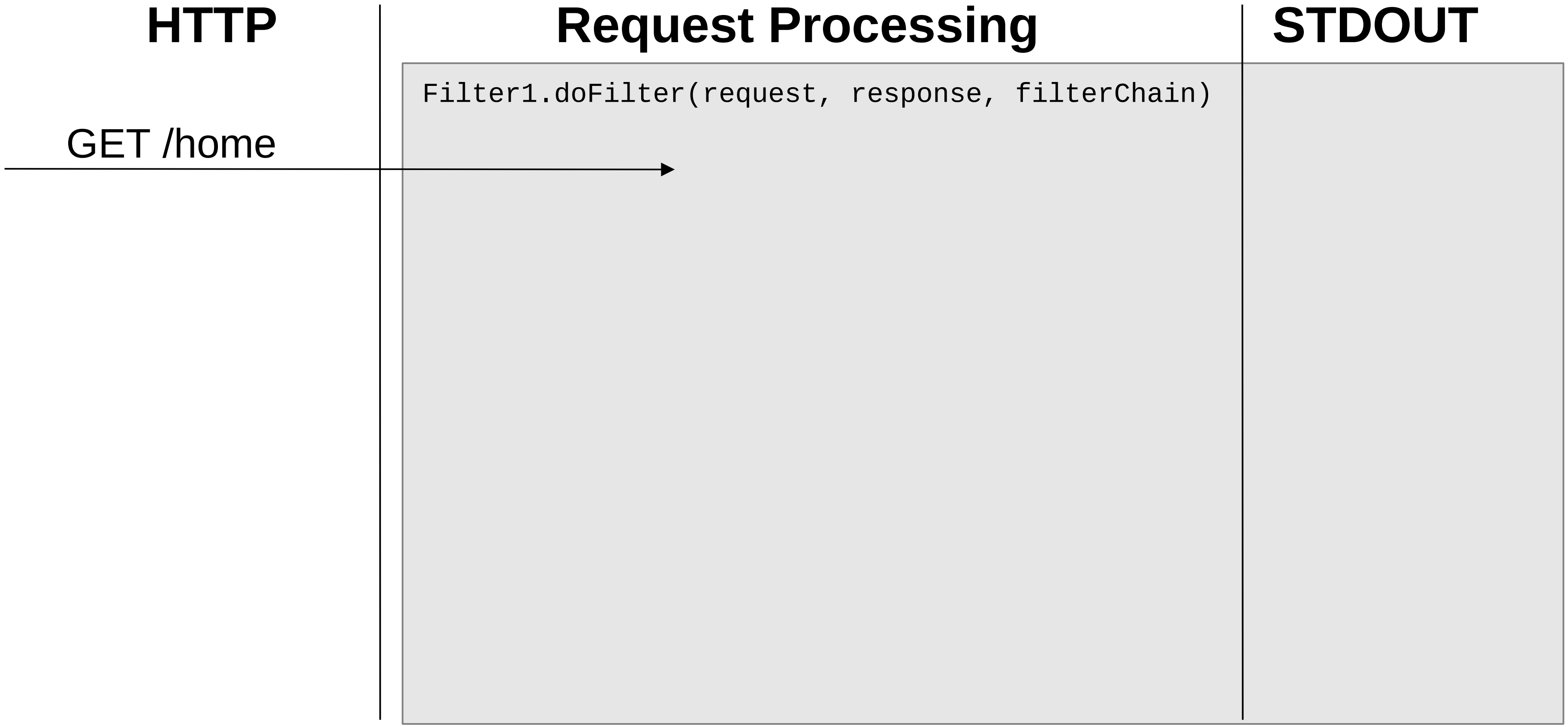
Filter1.java

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {
  …
  // do something before
  System.out.println("> " + requestUrl);
  // run rest of application
  filterChain.doFilter(request, response);
  // cleanup
  System.out.println("< " + requestUrl);
}
```

Java Servlet Filter Review - Dispatcher

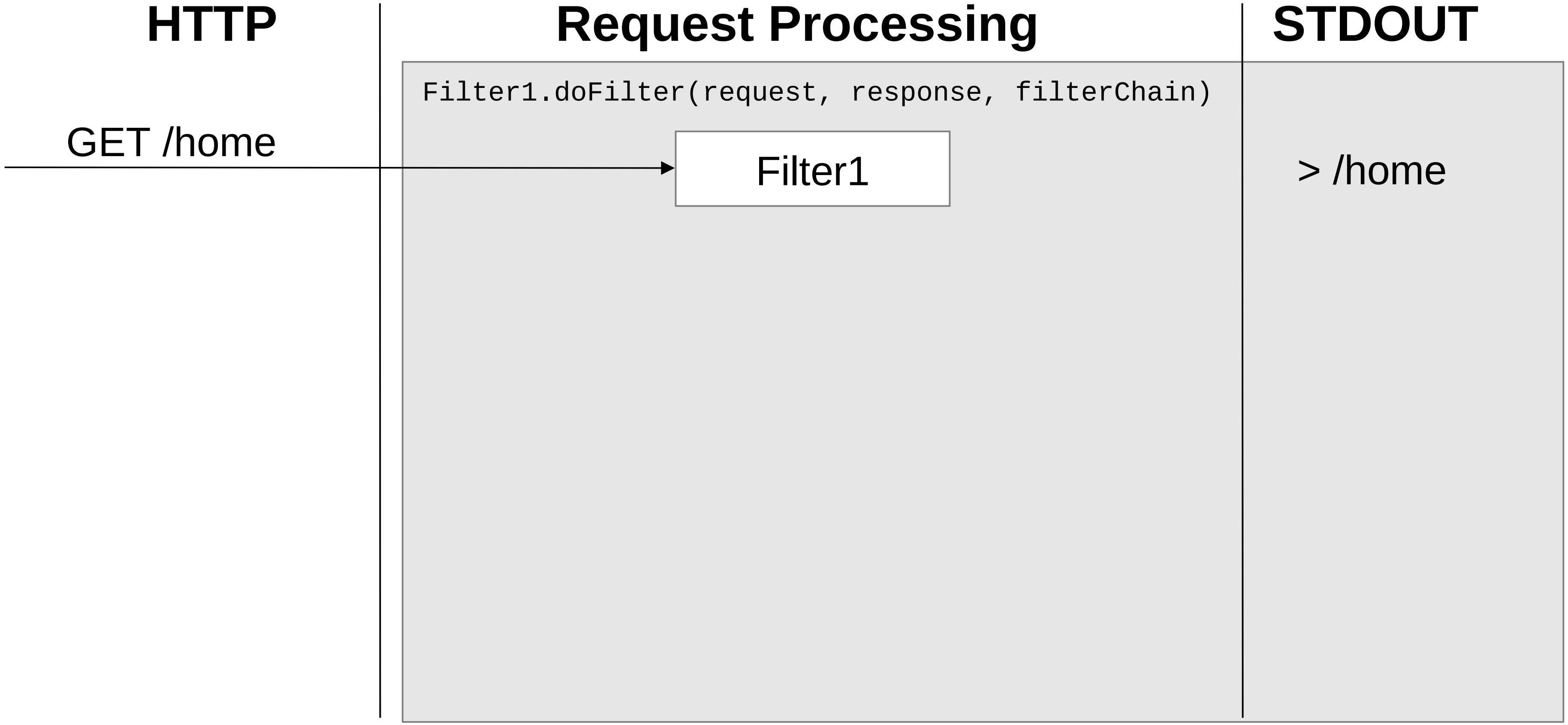


Java Servlet Filter Review - Dispatcher

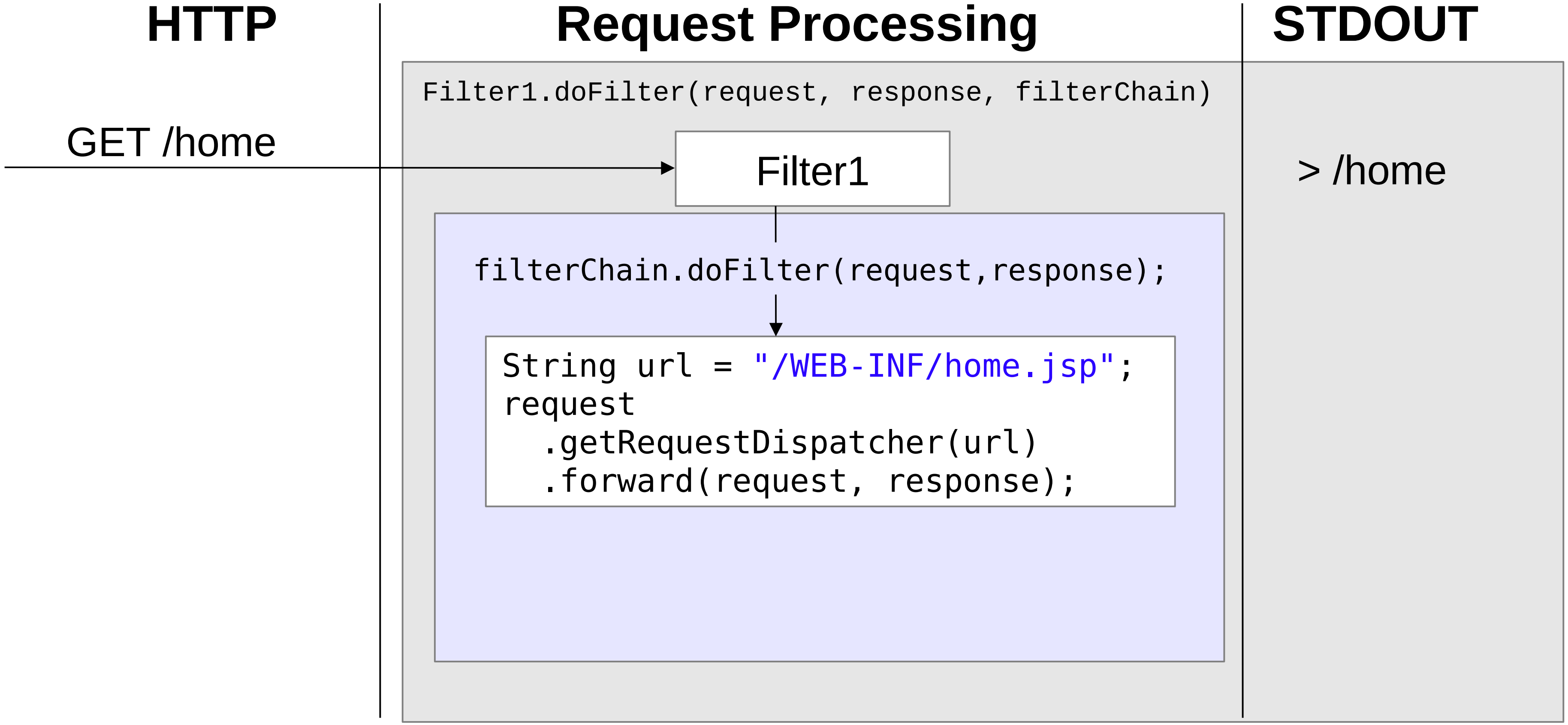


```
Filter1.doFilter(request, response, filterChain)
```

Java Servlet Filter Review - Dispatcher



Java Servlet Filter Review - Dispatcher

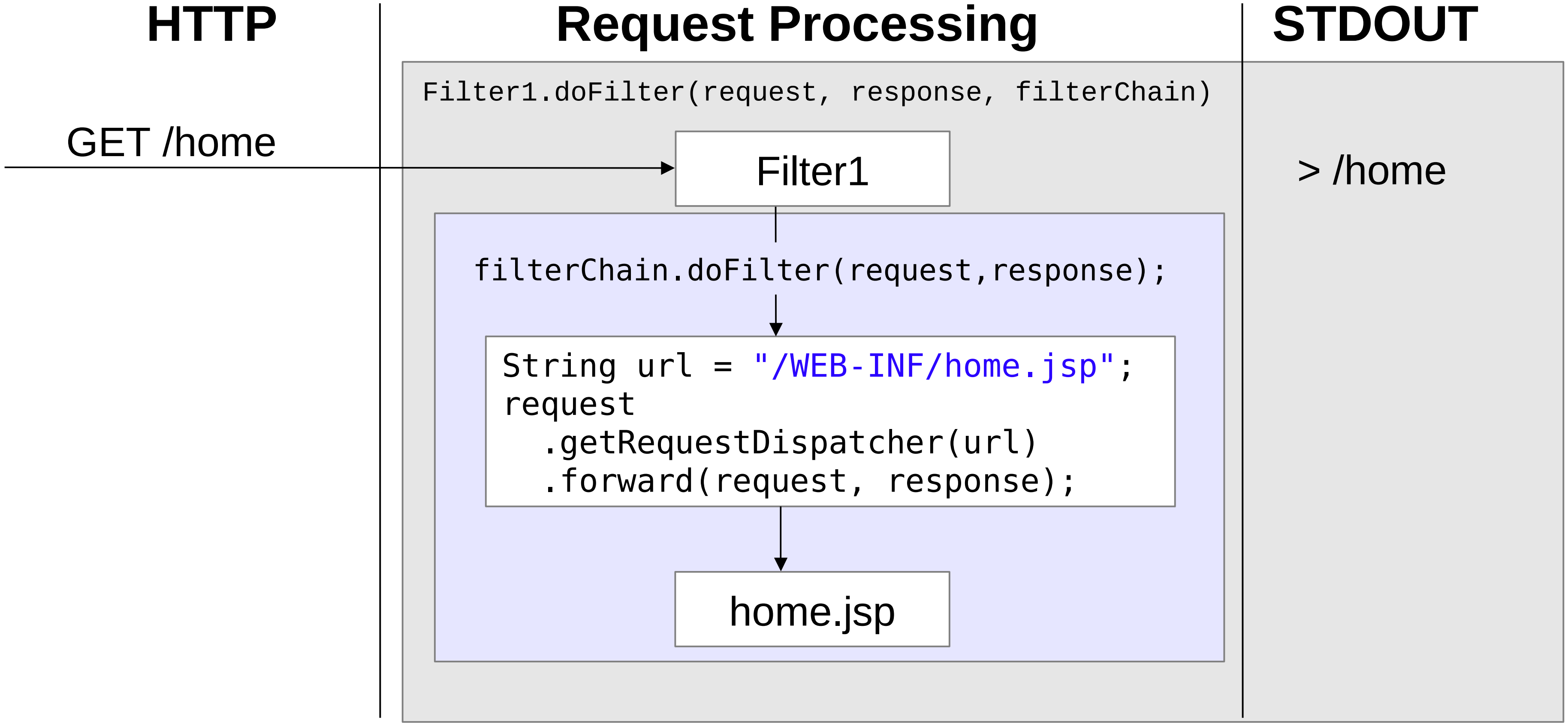


```
Filter1.doFilter(request, response, filterChain)
```

```
filterChain.doFilter(request,response);
```

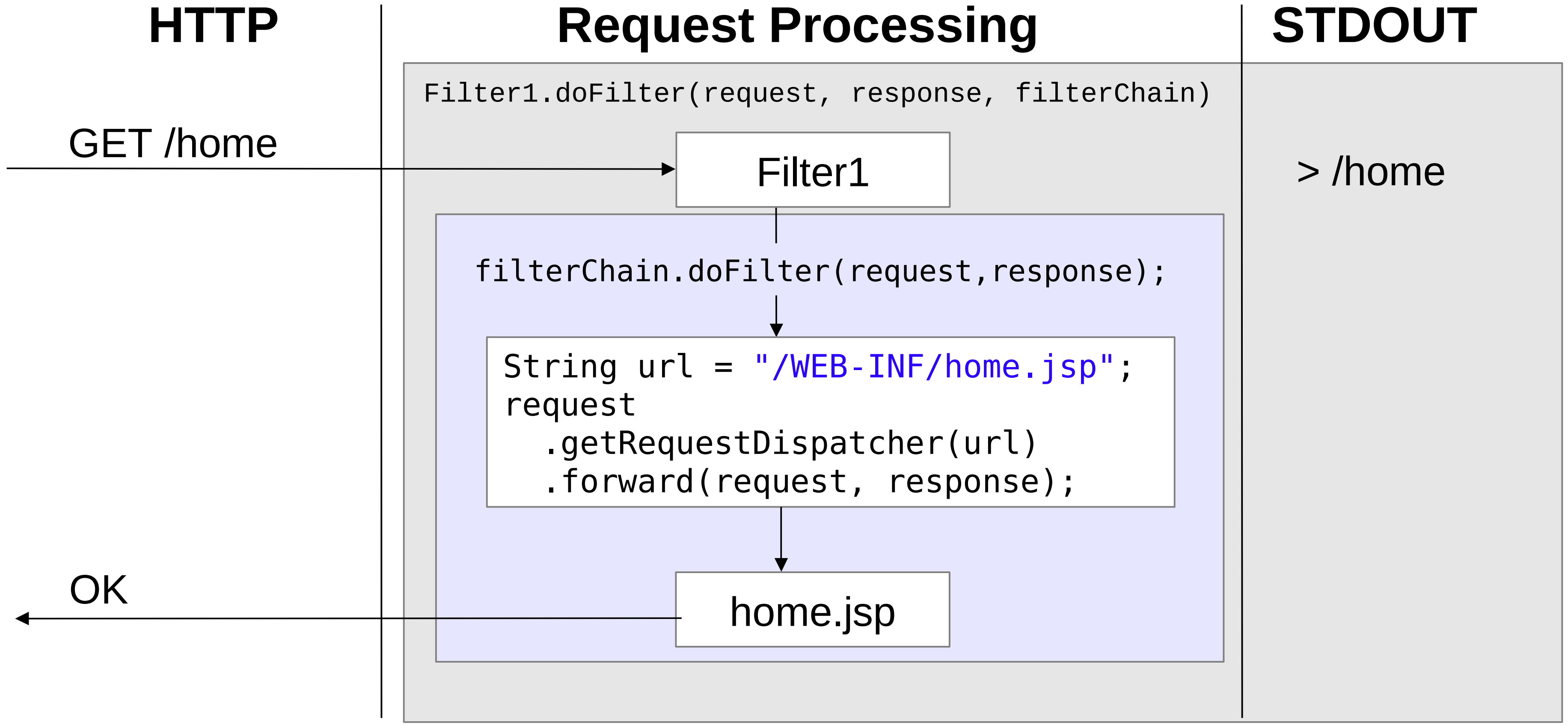
```
String url = "/WEB-INF/home.jsp";
request
  .getRequestDispatcher(url)
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher


HTTP
Request Processing
STDOUT
GET /home
Filter1.doFilter(request, response, filterChain)
Filter1
filterChain.doFilter(request,response);
home.jsp
> /home

```
String url = "/WEB-INF/home.jsp";
request
  .getRequestDispatcher(url)
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher

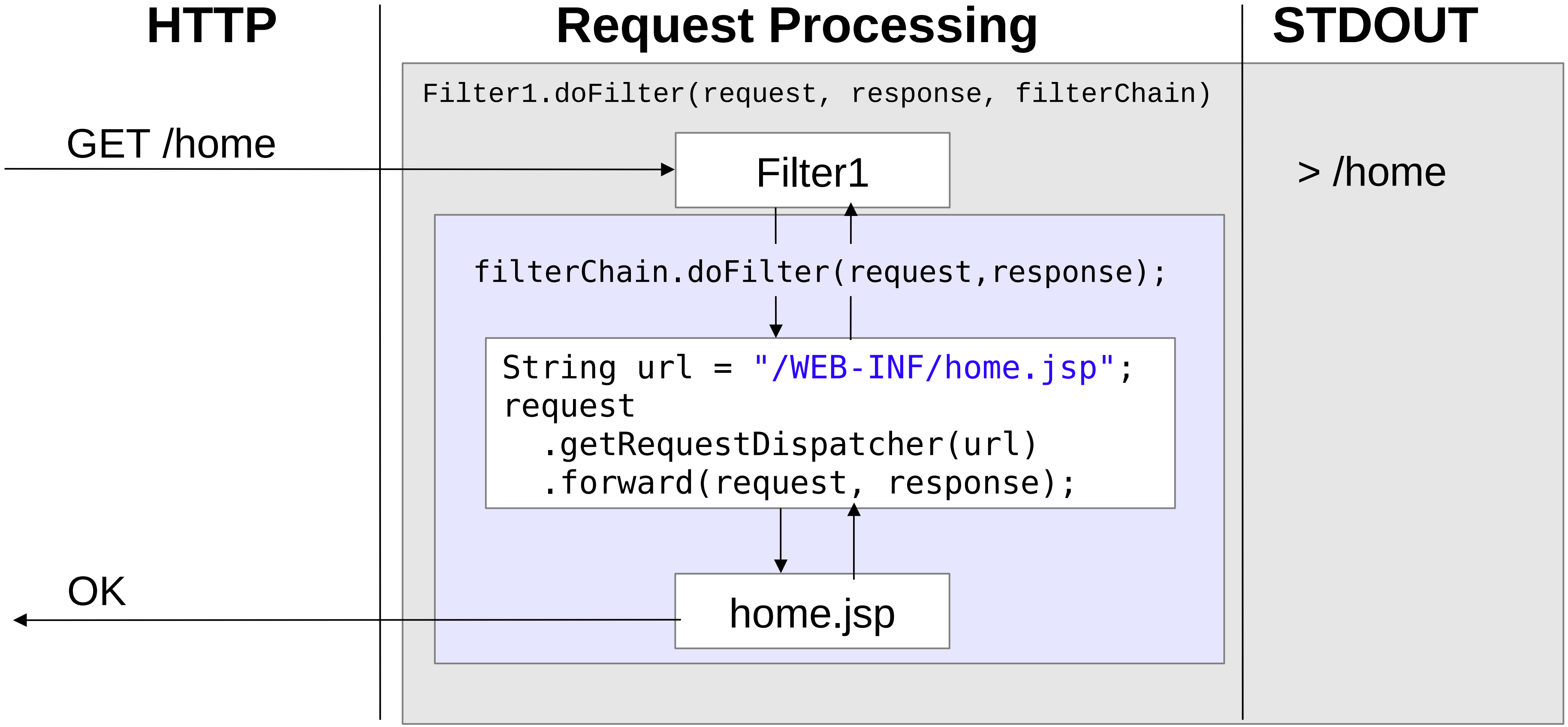


```
Filter1.doFilter(request, response, filterChain)
```

```
filterChain.doFilter(request,response);
```

```
String url = "/WEB-INF/home.jsp";
request
  .getRequestDispatcher(url)
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher

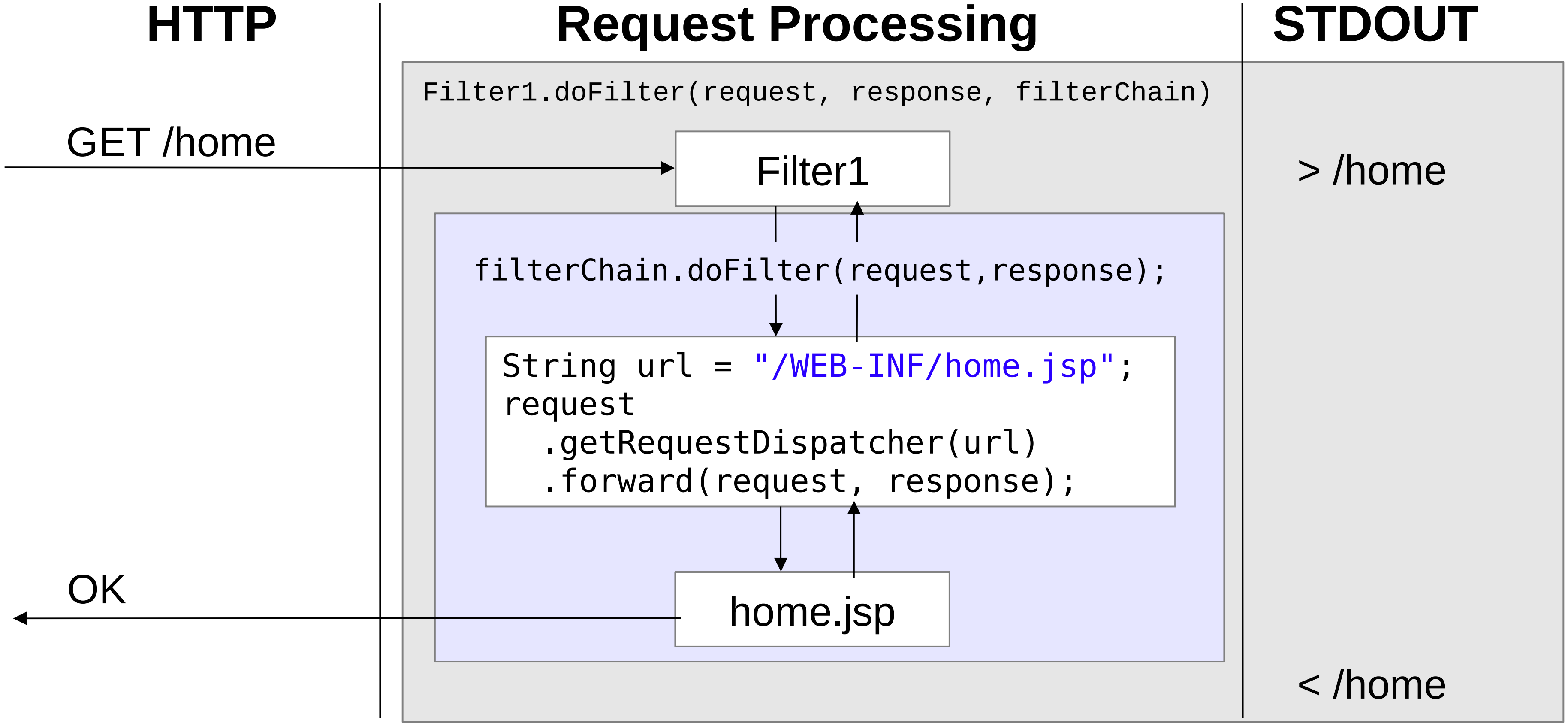


```
Filter1.doFilter(request, response, filterChain)
```

```
filterChain.doFilter(request,response);
```

```
String url = "/WEB-INF/home.jsp";
request
  .getRequestDispatcher(url)
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



```
Filter1.doFilter(request, response, filterChain)
```

```
filterChain.doFilter(request,response);
```

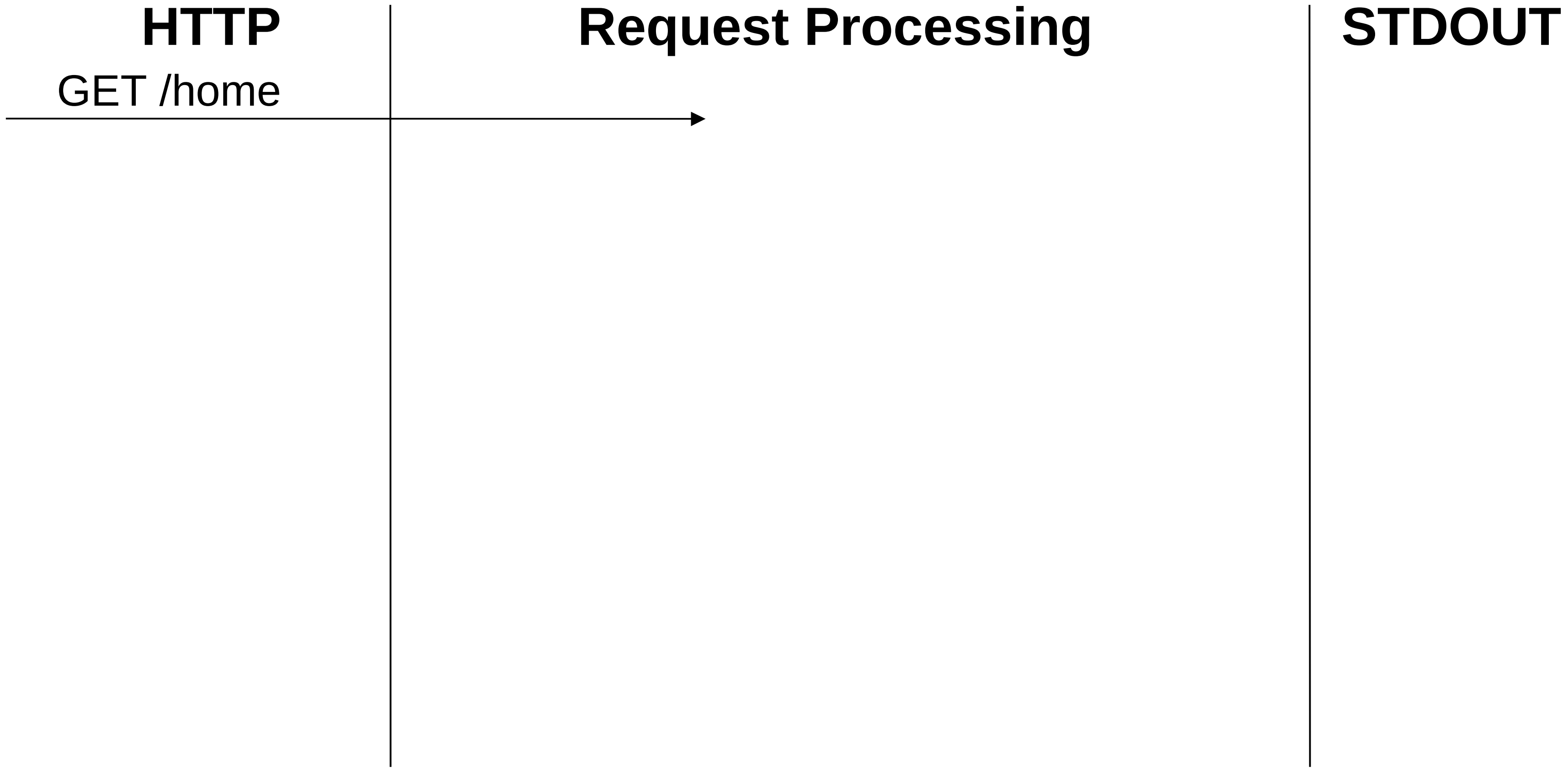
```
String url = "/WEB-INF/home.jsp";
request
  .getRequestDispatcher(url)
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher

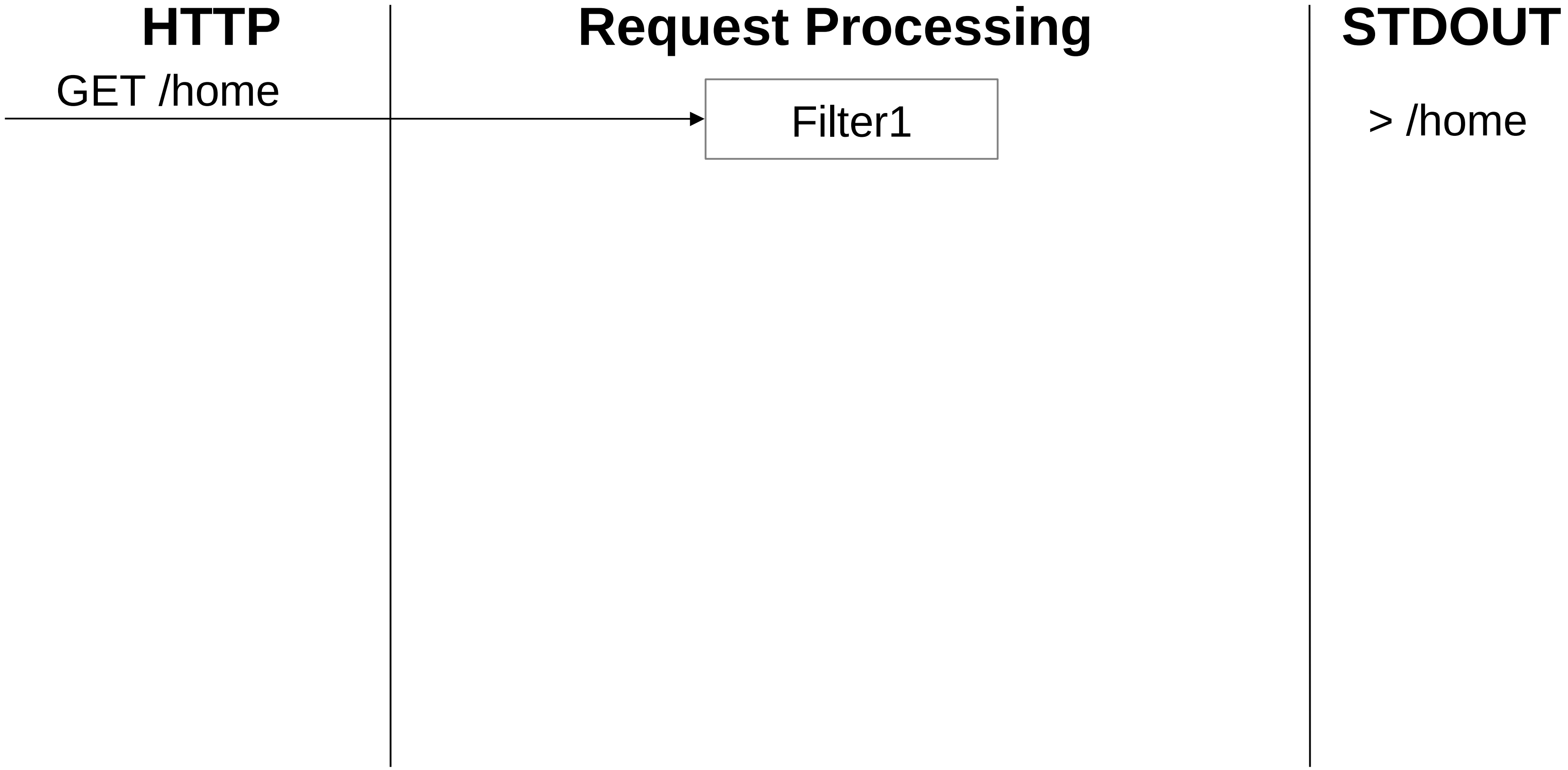
web.xml

```
<filter>
  <filter-name>filter1</filter-name>
  <filter-class>Filter1</filter-class>
</filter>
<filter-mapping>
  <filter-name>filter1</filter-name>
  <url-pattern>/*</url-pattern>
  <dispatcher>REQUEST</dispatcher>
  <dispatcher>FORWARD</dispatcher>
</filter-mapping>
```

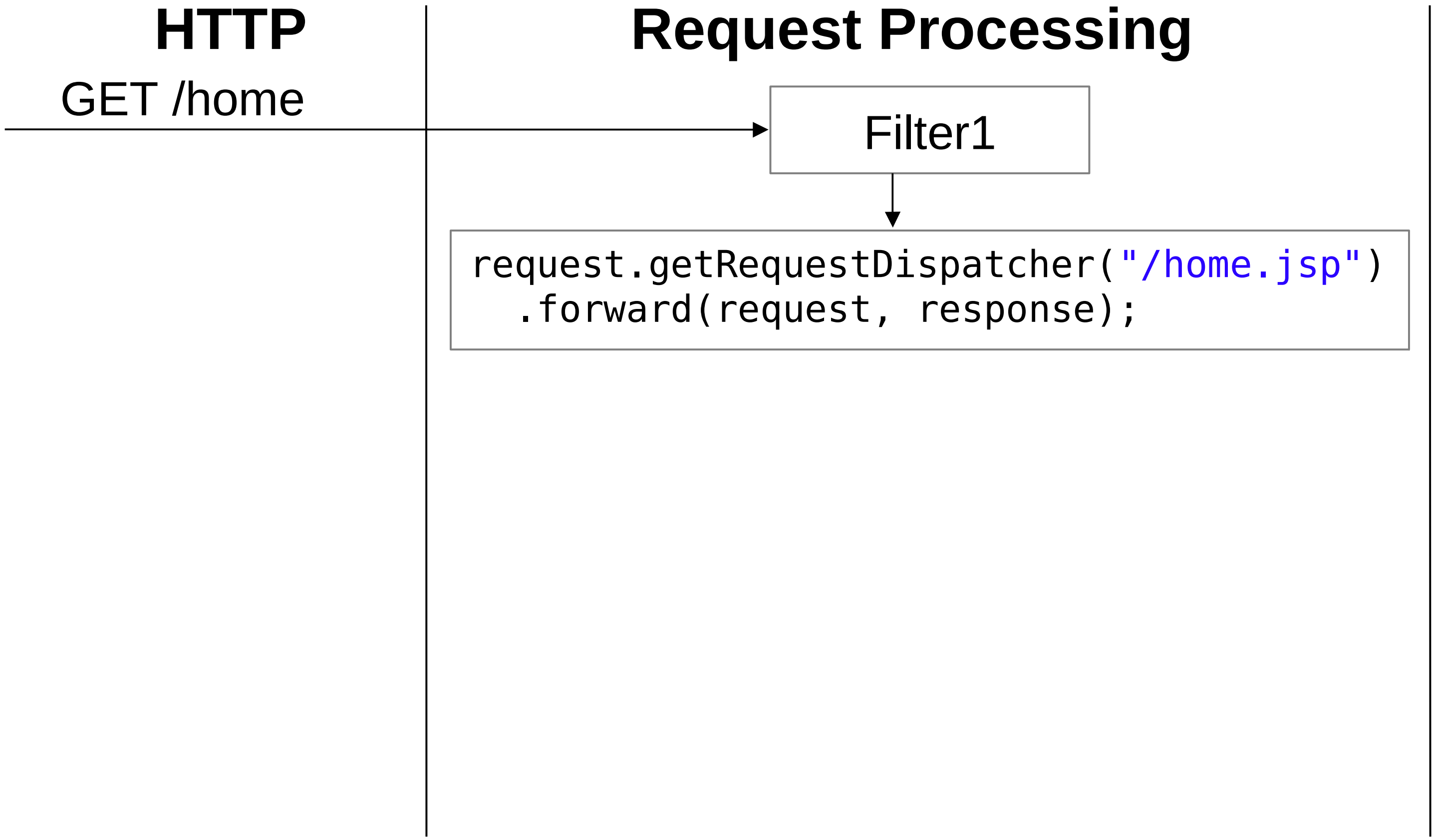
Java Servlet Filter Review - Dispatcher



Java Servlet Filter Review - Dispatcher

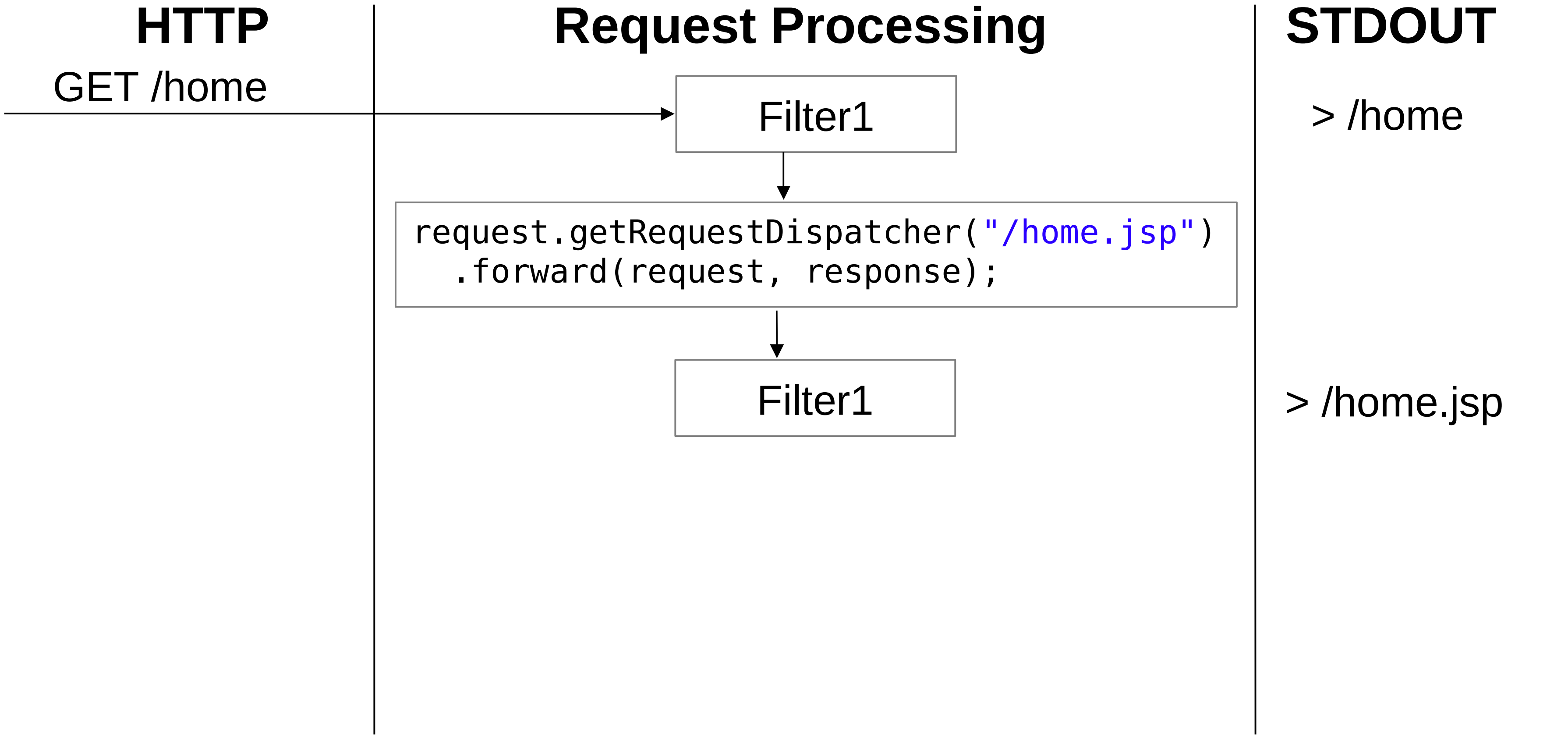


Java Servlet Filter Review - Dispatcher



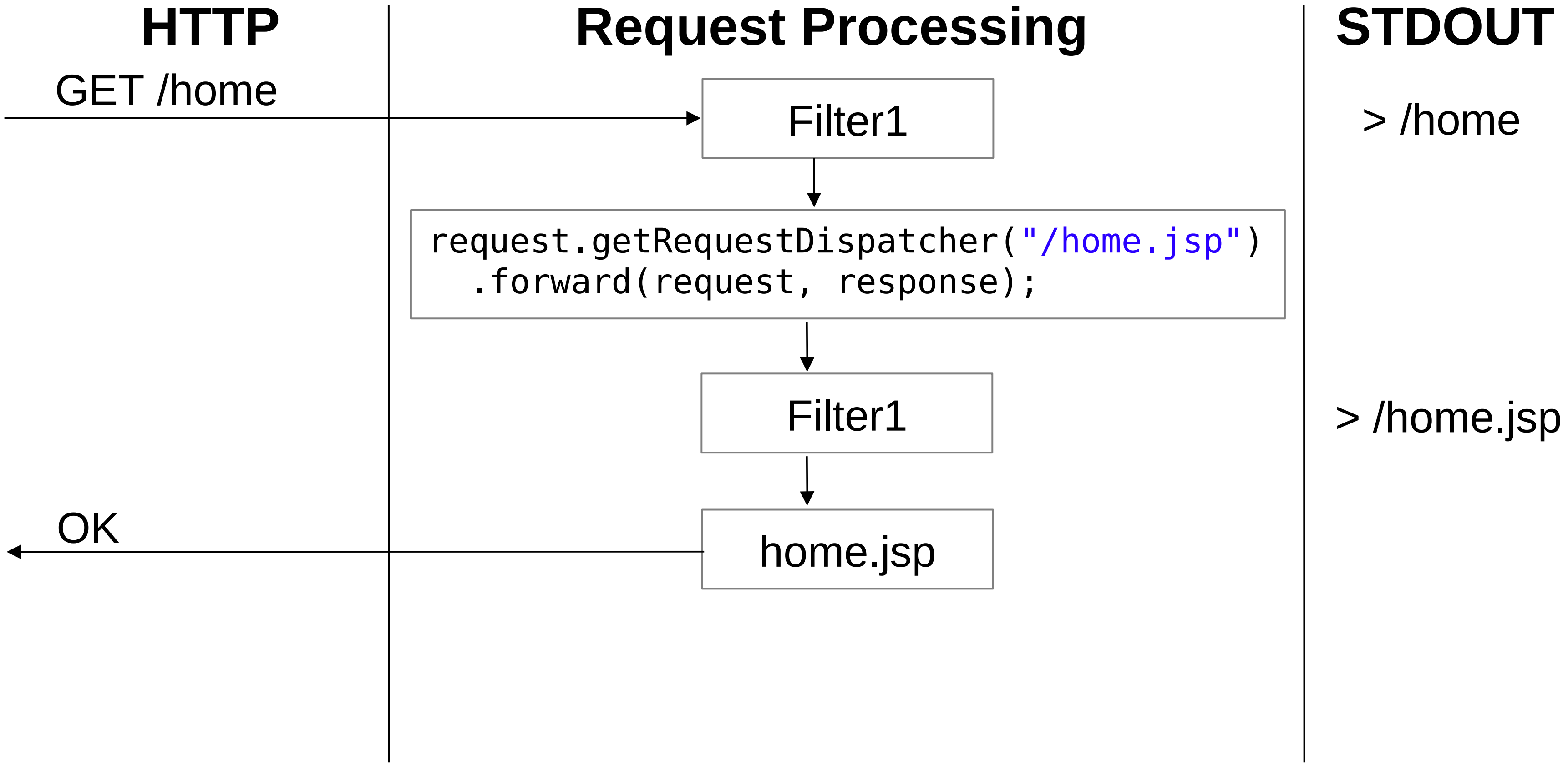
```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



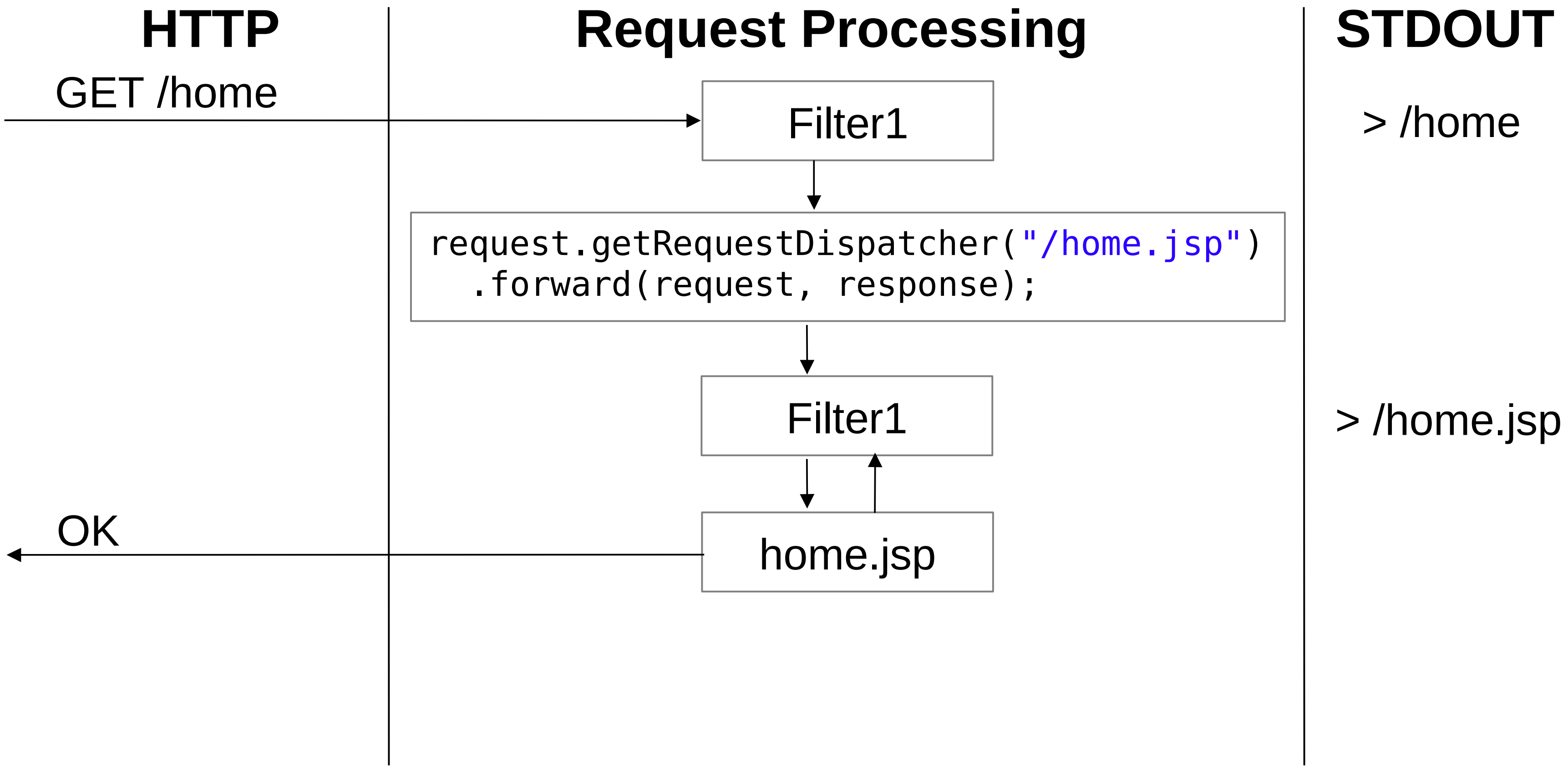
```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



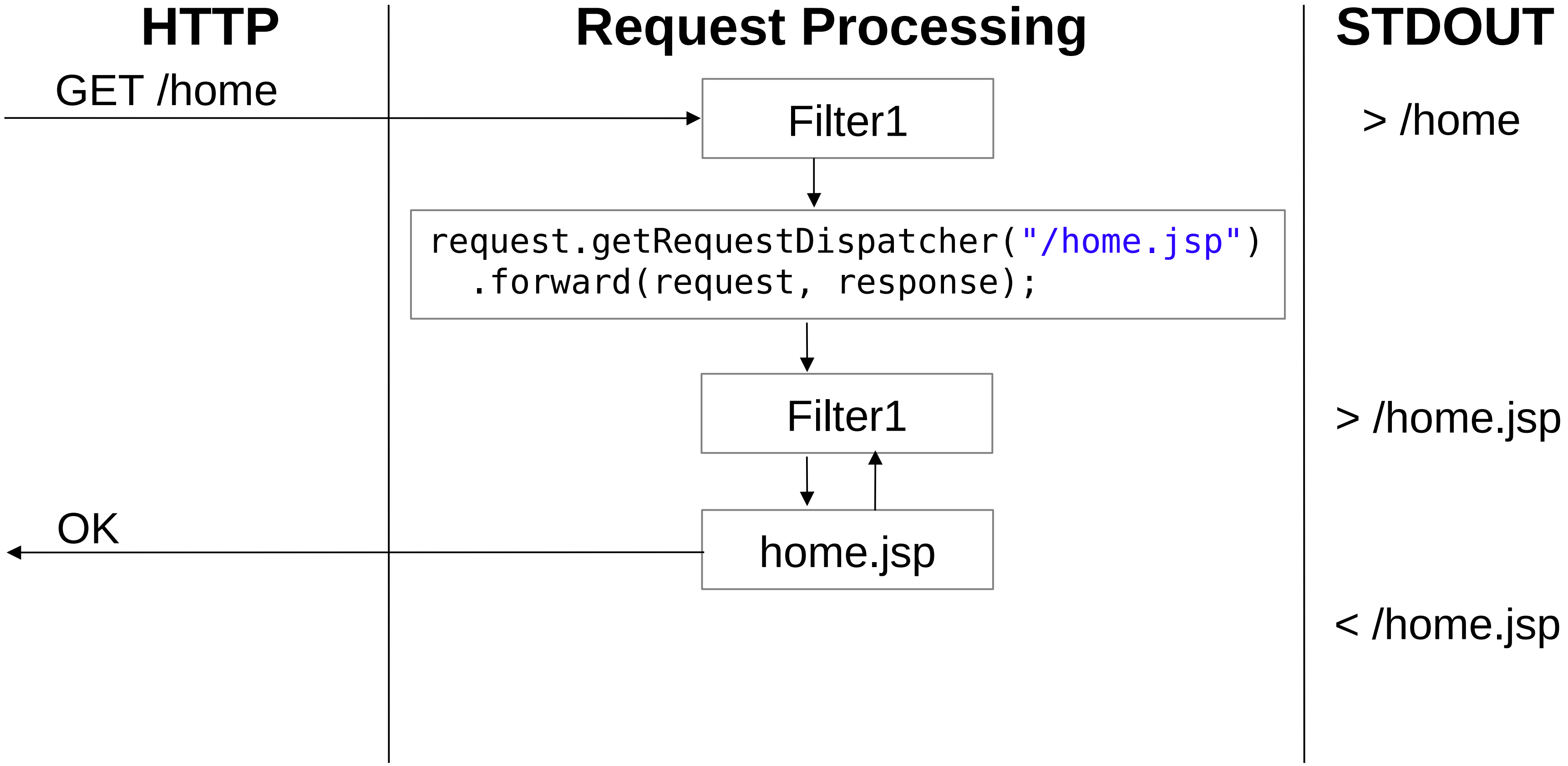
```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



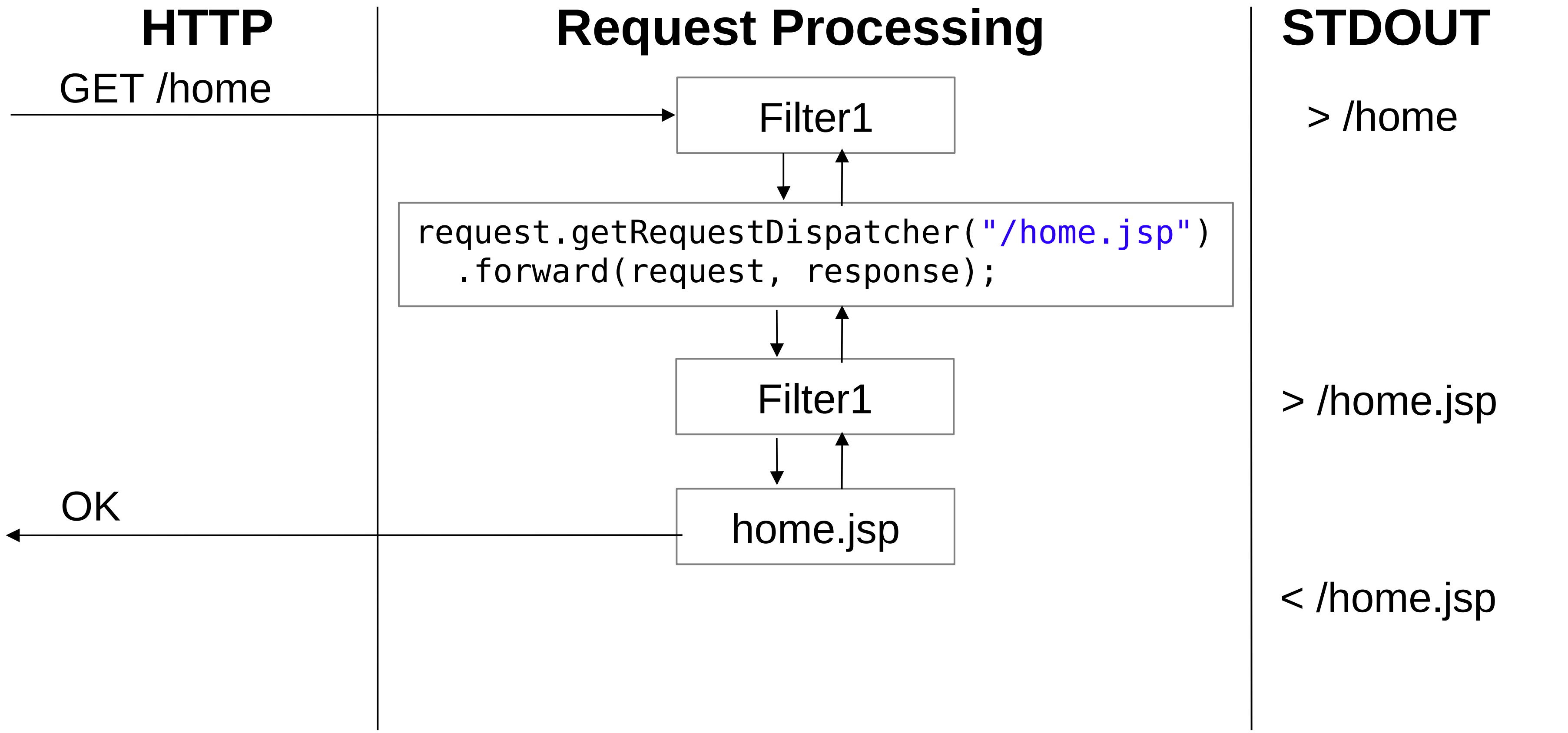
```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



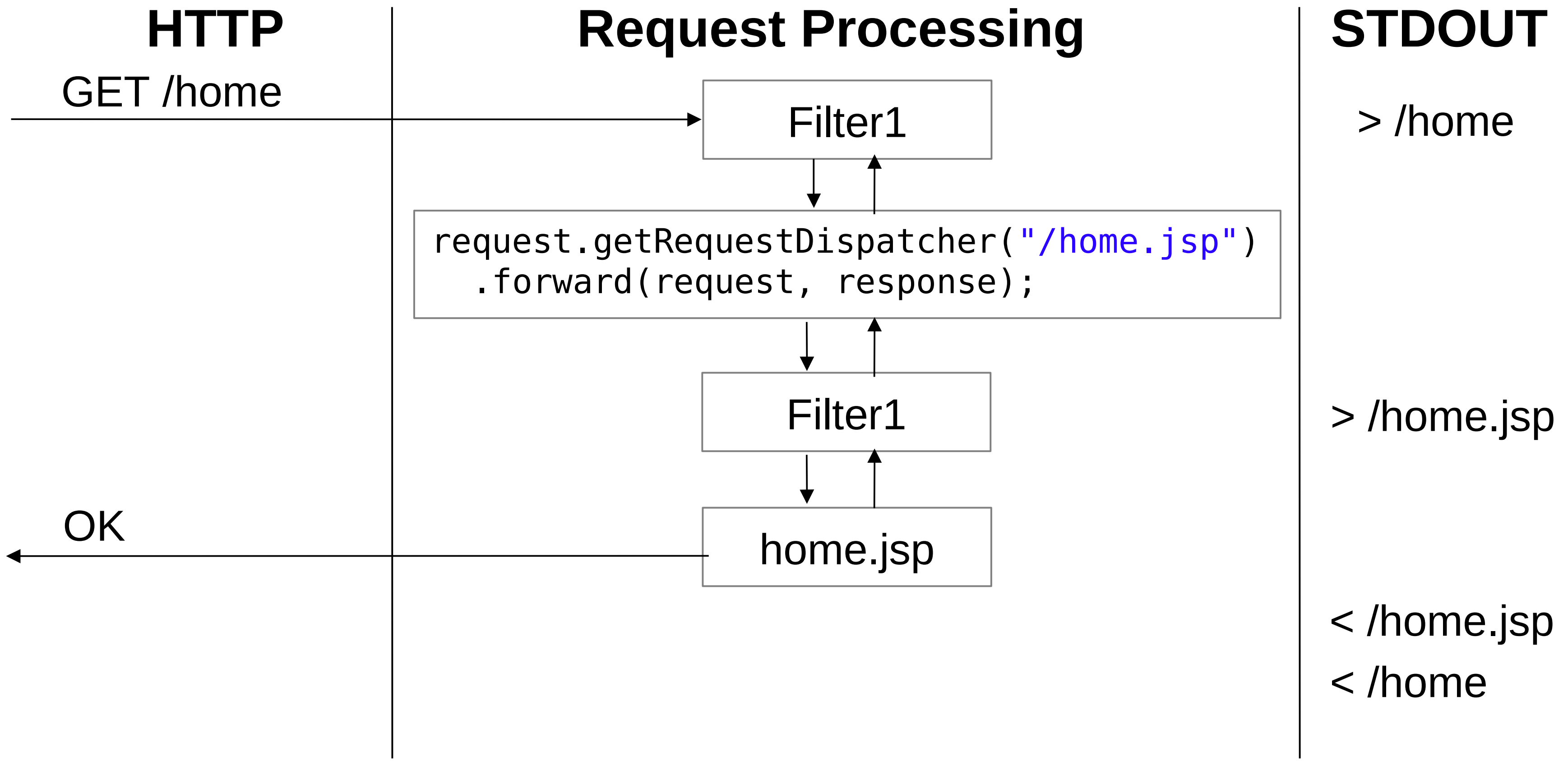
```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher



```
request.getRequestDispatcher("/home.jsp")
  .forward(request, response);
```

Java Servlet Filter Review - Dispatcher

- Spring Security is based on Servlet Filters
- Rare to process other dispatcher types, but important to be aware of them
- Ensure to include the necessary dispatcher elements
- Other possible dispatcher values include
  - REQUEST (default)
  - INCLUDE
  - FORWARD
  - ERROR

Java Servlet Filter Review - FilterChain

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {

    // run rest of application
    filterChain.doFilter(request, response);









}
```

Java Servlet Filter Review - FilterChain

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {

    // run rest of application
    ~~filterChain.doFilter(request, response);~~
    securityFilter.doFilter(request, response, filterChain);





}
```

Java Servlet Filter Review - FilterChain

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {

    // run rest of application
    ~~filterChain.doFilter(request, response);~~
    ~~securityFilter.doFilter(request, response, filterChain);~~
    servlet.service(request, response);






}
```

Java Servlet Filter Review - FilterChain

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {
  try {
    // run rest of application
    ~~filterChain.doFilter(request, response);~~
    ~~securityFilter.doFilter(request, response, filterChain);~~
    servlet.service(request, response);
  }
  catch (SecurityException e) {
    // handle error by sending to login page
  }
  finally {
    // cleanup
  }
}
```

Demo Messages Application

Setting up Spring Security

Basic Spring Security Setup

- Add Spring Security Dependencies
- Update web.xml
- Create Spring Security Configuration

Specifying Dependencies with Maven

```
<dependencies>
  ...
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-core</artifactId>
    <version>3.1.3.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-web</artifactId>
    <version>3.1.3.RELEASE</version>
  </dependency>
  <dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>spring-security-config</artifactId>
    <version>3.1.3.RELEASE</version>
  </dependency>
</dependencies>
```

Specifying Dependencies with Gradle

```
dependencies {
  compile "org.springframework:spring-context:$springVersion",
    ...
    "org.springframework.security:spring-security-web:3.1.3.RELEASE",
    "org.springframework.security:spring-security-config:3.1.3.RELEASE",
    "org.springframework.security:spring-security-core:3.1.3.RELEASE"
}
```

Update web.xml - ContextLoaderListener

web.xml

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
     /WEB-INF/spring/*.xml
  </param-value>
</context-param>
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

What is the ContextLoaderListener

- Not Specific to Spring Security
- Creates a Spring ApplicationContext using the Spring Configurations (i.e. the value of contextConfigLocation)
- Can be used to lookup objects in ApplicationContext
- Rare to interact with ApplicationContext directly

ContextLoaderListener pseudocode

```
// init ApplicationContext
XmlWebApplicationContext applicationContext =
    new XmlWebApplicationContext();
applicationContext.setConfigLocation("/WEB-INF/spring/*.xml");
applicationContext.refresh();

// Use ApplicationContext
Filter filter =
    applicationContext.getBean("springSecurityFilterChain",
                               Filter.class);
```

ContextLoaderListener pseudocode

```
// init ApplicationContext
XmlWebApplicationContext applicationContext =
    new XmlWebApplicationContext();
applicationContext.setConfigLocation
applicationContext.refresh();

// Use Applica
Filter filter
    applicatio
```

```
"/WEB-INF/spring/*.xml"
```

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>
     /WEB-INF/spring/*.xml
  </param-value>
</context-param>
```

ContextLoaderListener pseudocode

```
// init ApplicationContext
XmlWebApplicationContext applicationContext =
    new XmlWebApplicationContext();
applicationContext.setConfigLocation("/WEB-INF/spring/*.xml");
applicationContext.refresh();

// Use ApplicationContext
Filter filter =
    applicationContext.getBean("springSecurityFilterChain",
                               Filter.class);
```

Update web.xml - springSecurityFilterChain

web.xml

```
<filter>
  <filter-name>springSecurityFilterChain</filter-name>
  <filter-class>
    org.springframework.web.filter.DelegatingFilterProxy
  </filter-class>
</filter>
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

DelegatingFilterProxy pseudocode

```
public class DelegatingFilterProxy implements Filter {
  public void init(FilterConfig config) throws ServletException {
    // applicationContext is obtained from ContextLoaderListener
    this.delegate =
      applicationContext.getBean("springSecurityFilterChain",
                                 Filter.class);
  }
  public void doFilter(...) throws ... {
    this.delegate.doFilter(request, response, filterChain);
  }
  public void destroy() {
    // this may not be invoked depending on the settings
    this.delegate.destroy();
  }
  private Filter delegate;
}
```

DelegatingFilterProxy pseudocode

```
public class DelegatingFilterProxy implements Filter {
  public void init(FilterConfig config) throws ServletException {
    // applicationContext is obtained from ContextLoaderListener
    this.delegate =
      applicationContext.getBean( springSecurityFilterChain ,
                                  Filter.class);
  }
```

```
<filter-mapping>
  <filter-name>springSecurityFilterChain</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

```
    this.delegate.destroy();
  }
  private Filter delegate;
}
```

DelegatingFilterProxy pseudocode

```
public class DelegatingFilterProxy implements Filter {
  public void init(FilterConfig config) throws ServletException {
    // applicationContext is obtained from ContextLoaderListener
    this.delegate =
      applicationContext.getBean("springSecurityFilterChain",
                                 Filter.class);
  }
  public void doFilter(...) throws ... {
    this.delegate.doFilter(request, response, filterChain);
  }
  public void destroy() {
    // this may not be invoked depending on the settings
    this.delegate.destroy();
  }
  private Filter delegate;
}
```

Create security.xml

- The file location should be src/main/webapp/WEB-INF/spring/security.xml to match the contextConfigLocation
- Need to ensure to get the xml namespace declaration correct
- Spring Tool Suite (STS) can help with adding namespace decloarations

Create security.xml with Spring Tool Suite

- In STS right click `src/main/webapp/WEB-INF/spring/`
- **New → Spring Bean Configuration File**
- Enter `security.xml` as the file name
- Click **Next**
- Select Security
- Click **Finish**

src/main/webapp/WEB-INF/security.xml

Add the following between the `<beans>` tags

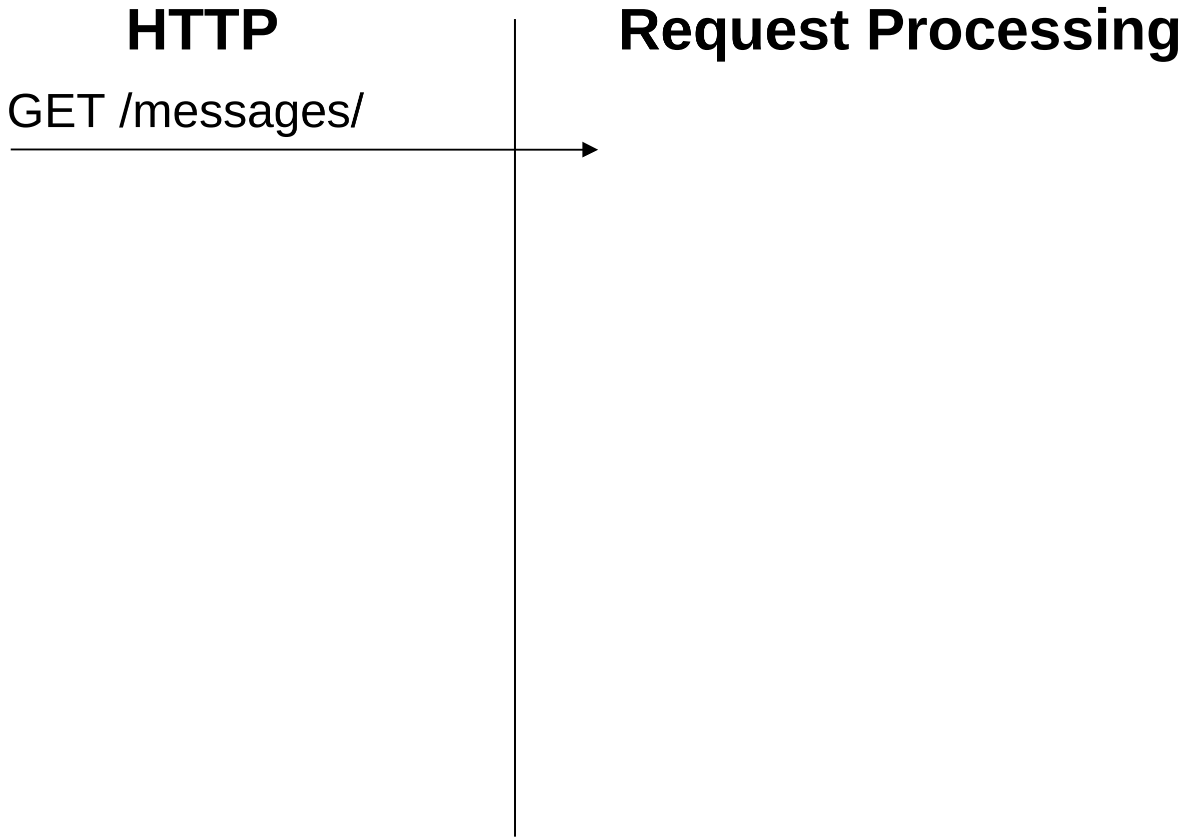
```
<security:http use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
  <security:form-login />
</security:http>
<security:authentication-manager>
  <security:authentication-provider>
    <security:user-service>
      <security:user name="user"
          password="password"
          authorities="ROLE_USER"/>
    </security:user-service>
  </security:authentication-provider>
</security:authentication-manager>
```

Demo Basic Spring Security

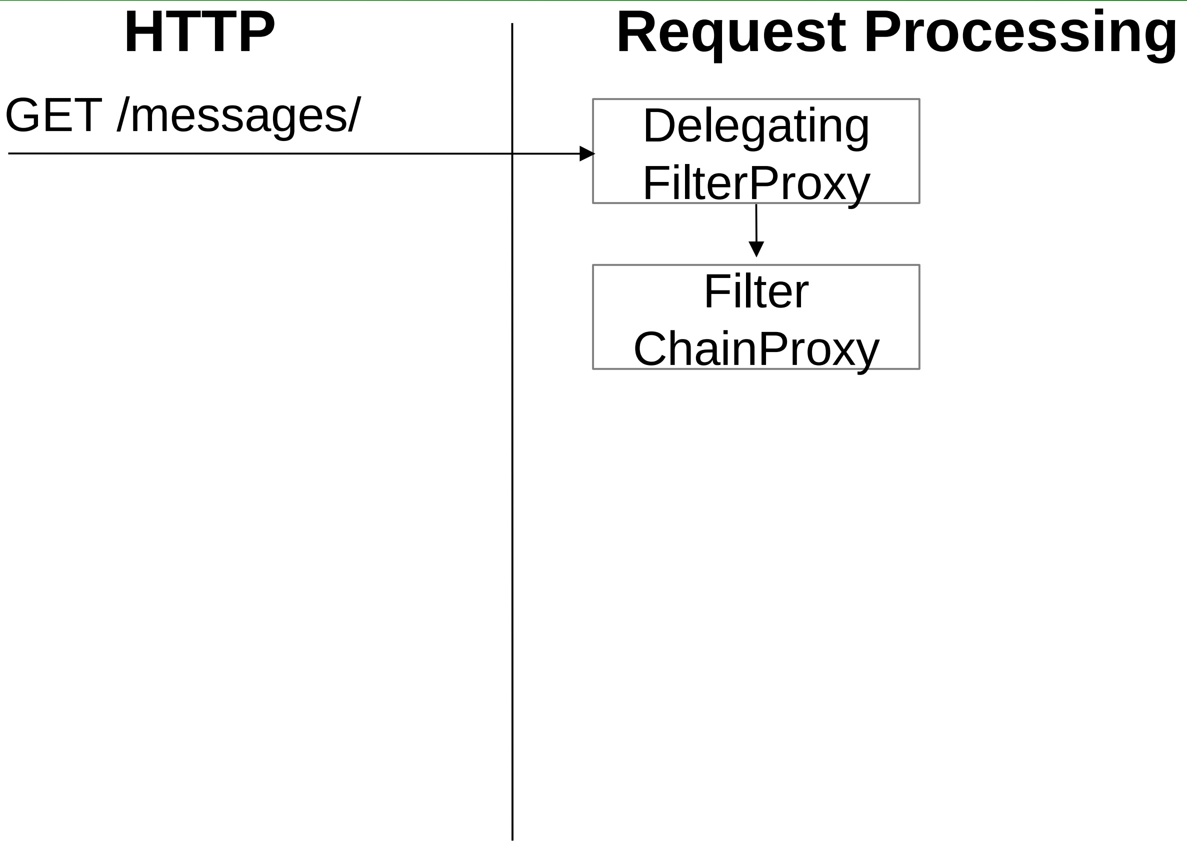
FilterChainProxy (springSecurityFilterChain) Pseudocode

```
public void doFilter(ServletRequest request,
    ServletResponse response, FilterChain filterChain) … {

  Filter[] delegates = lookupDelegates(request);
  for(Filter delegate : delegates) {
    delegate.doFilter(request, response, chain);

    if(delegate does not invoke filterChain.doFilter)
      return;
  }

  filterChain.doFilter(request, response);
}
```

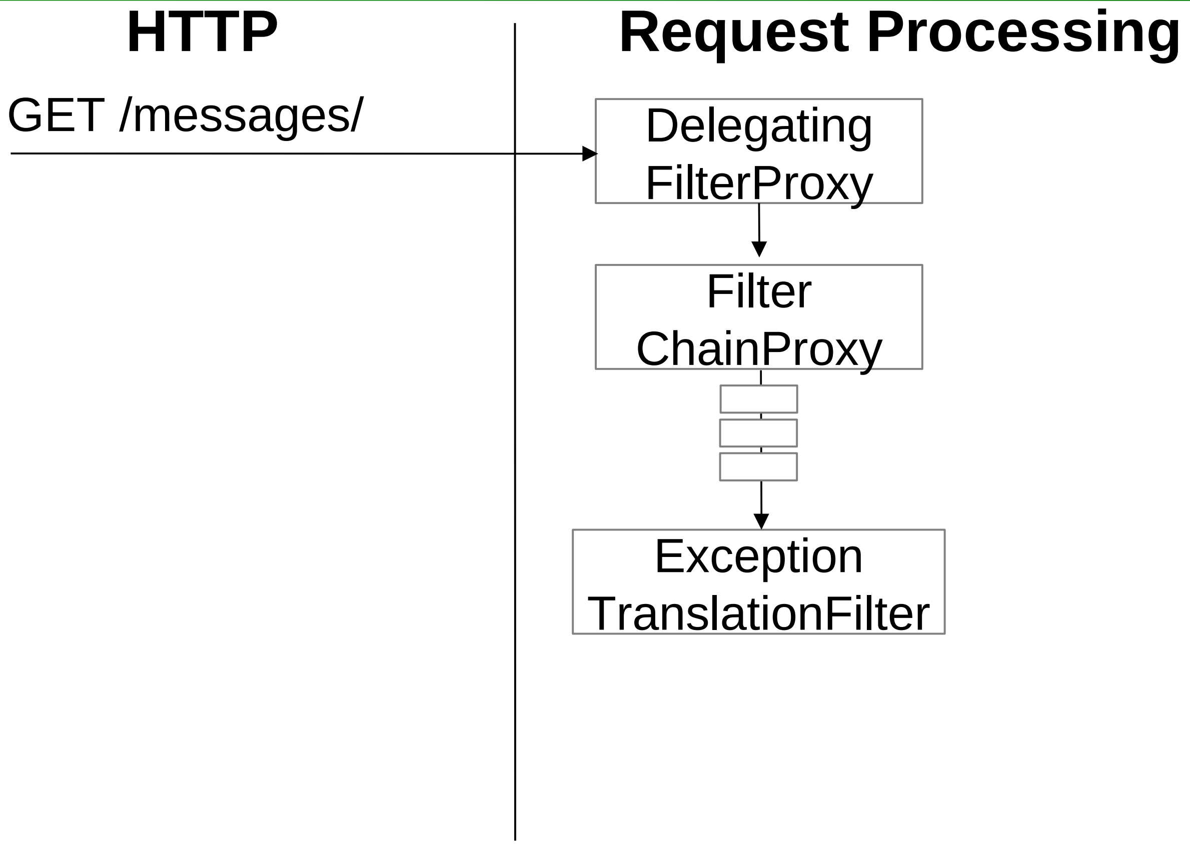
Unauthenticated Request to Protected Resource



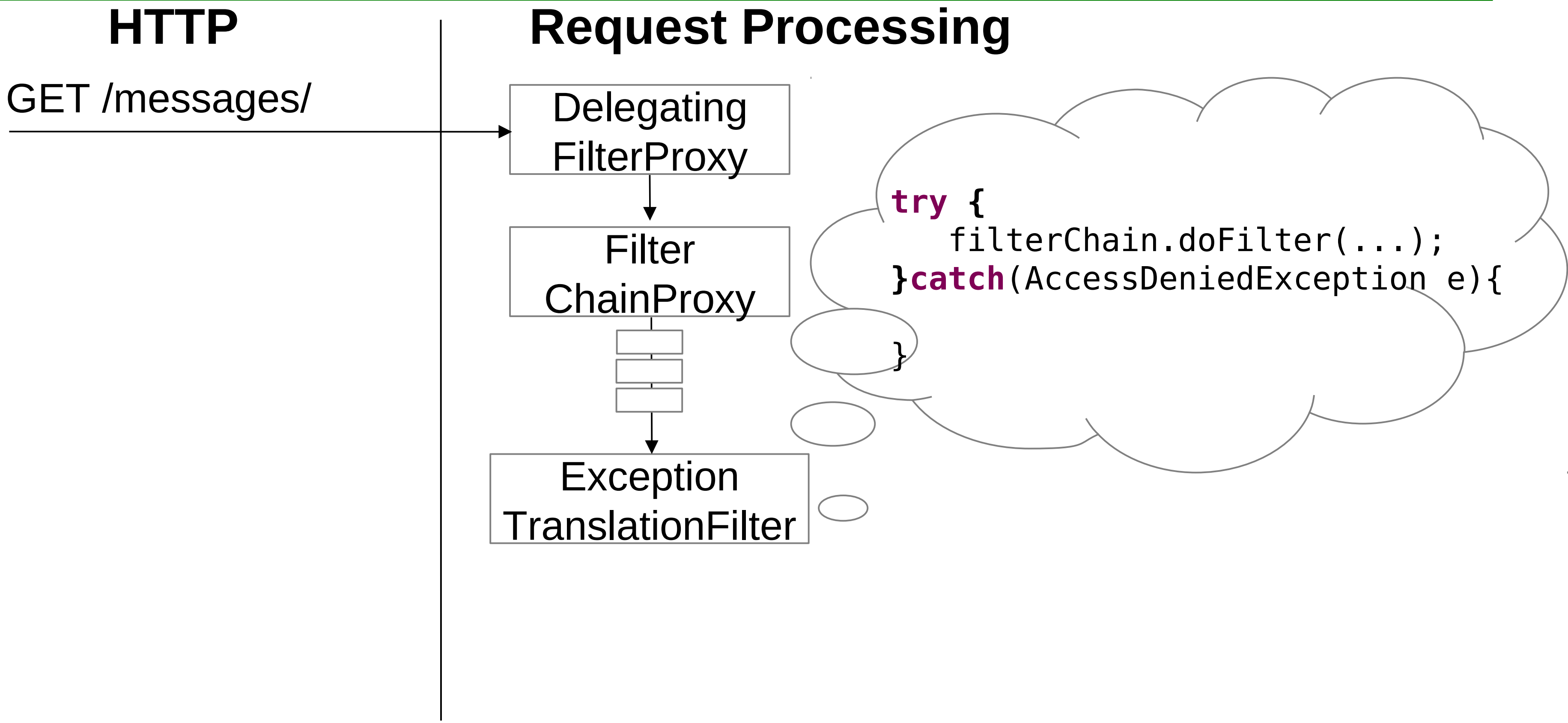
Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource

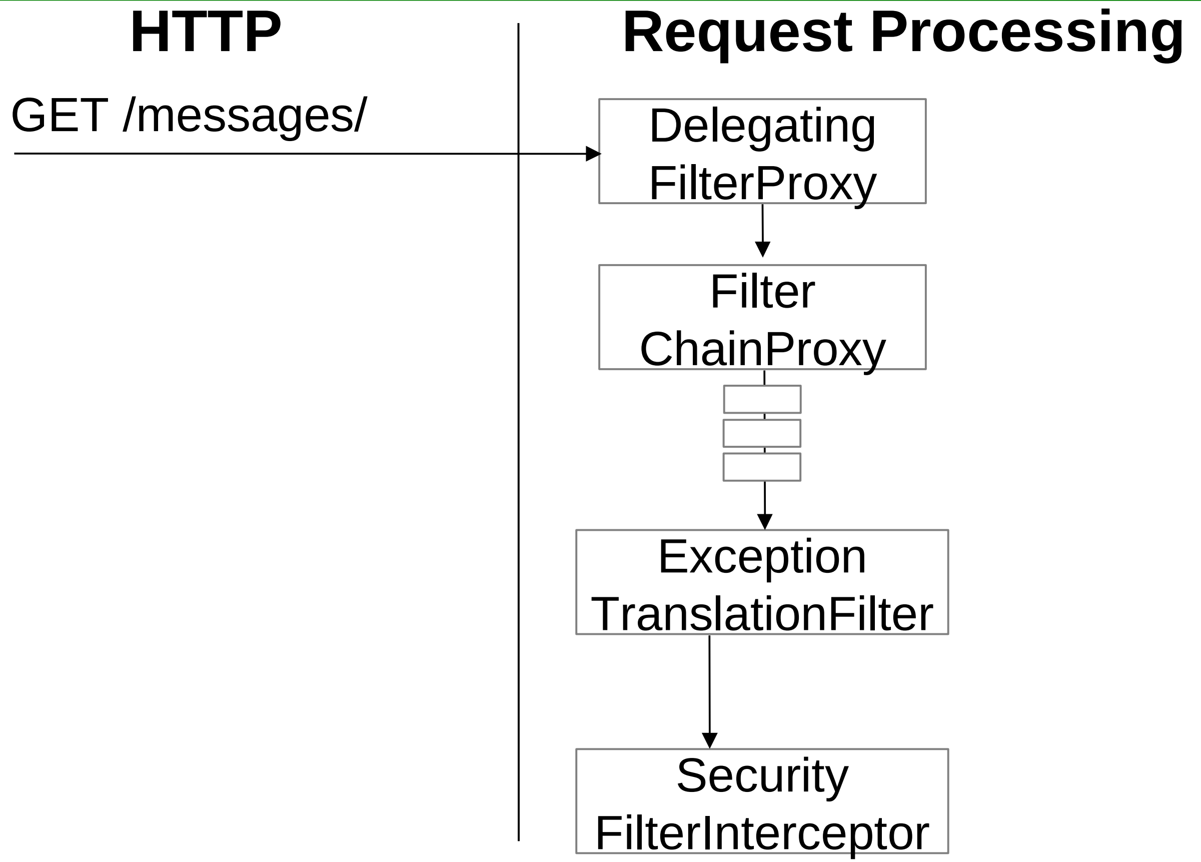


Unauthenticated Request to Protected Resource

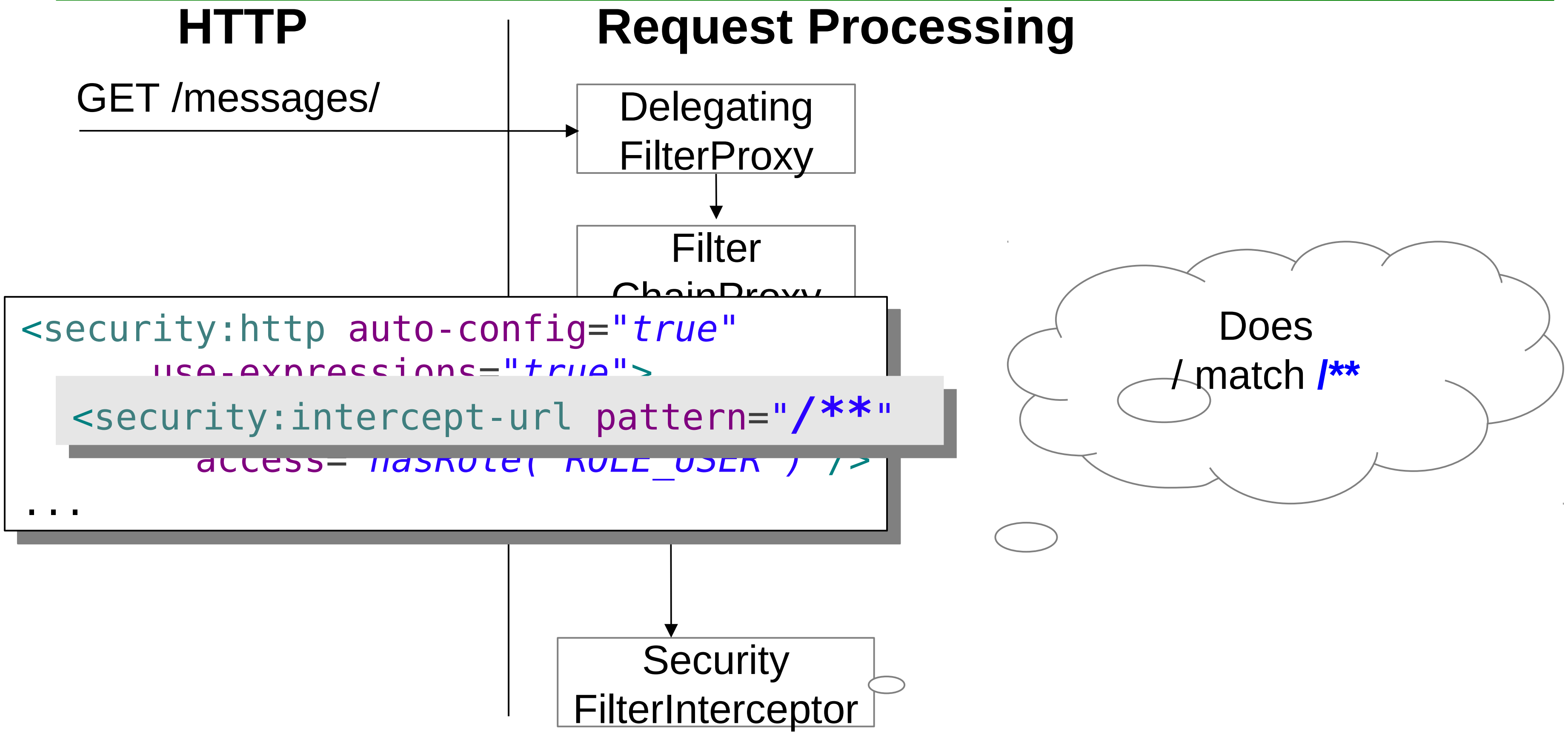


```
try {
    filterChain.doFilter(...);
}catch(AccessDeniedException e){

}
```

Unauthenticated Request to Protected Resource

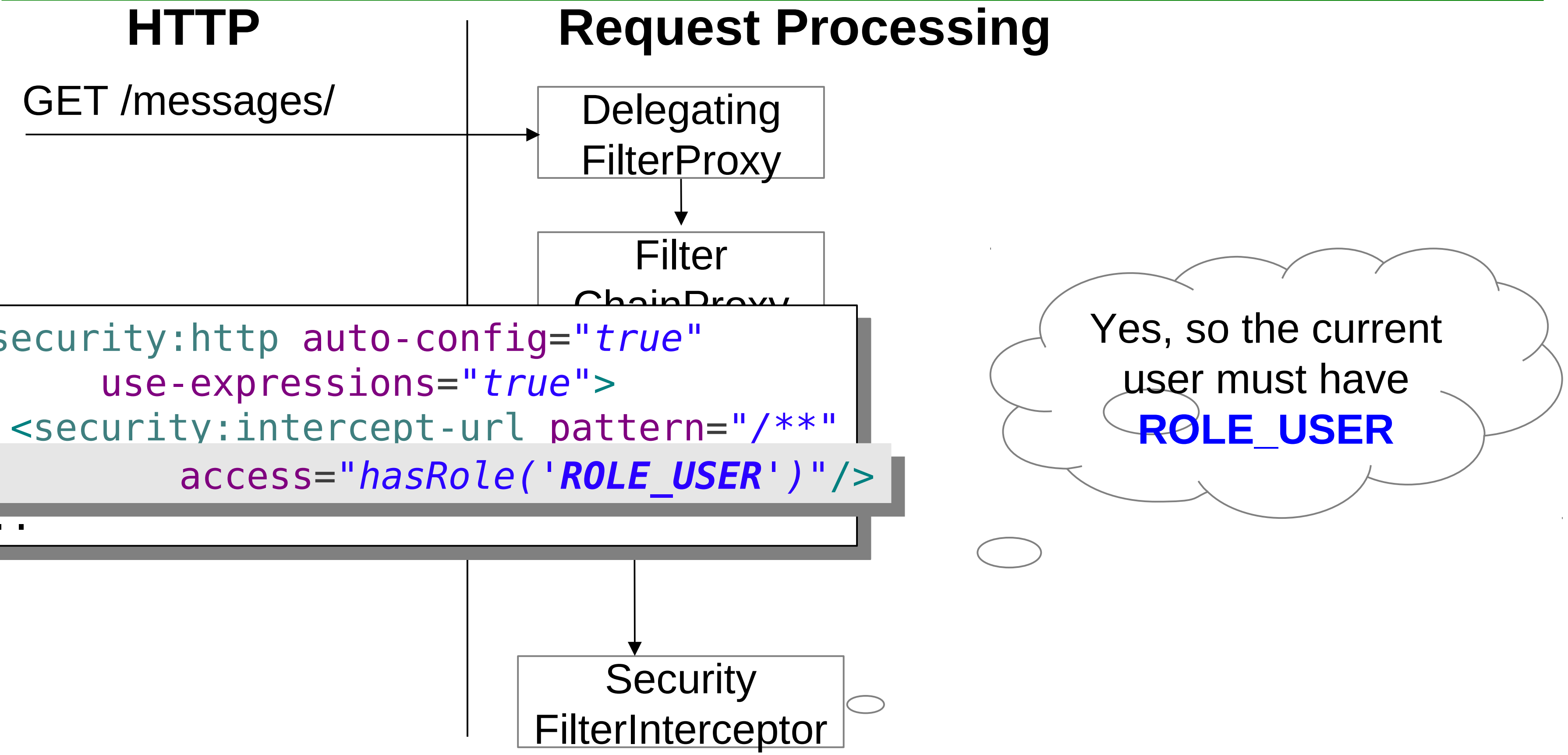


Unauthenticated Request to Protected Resource



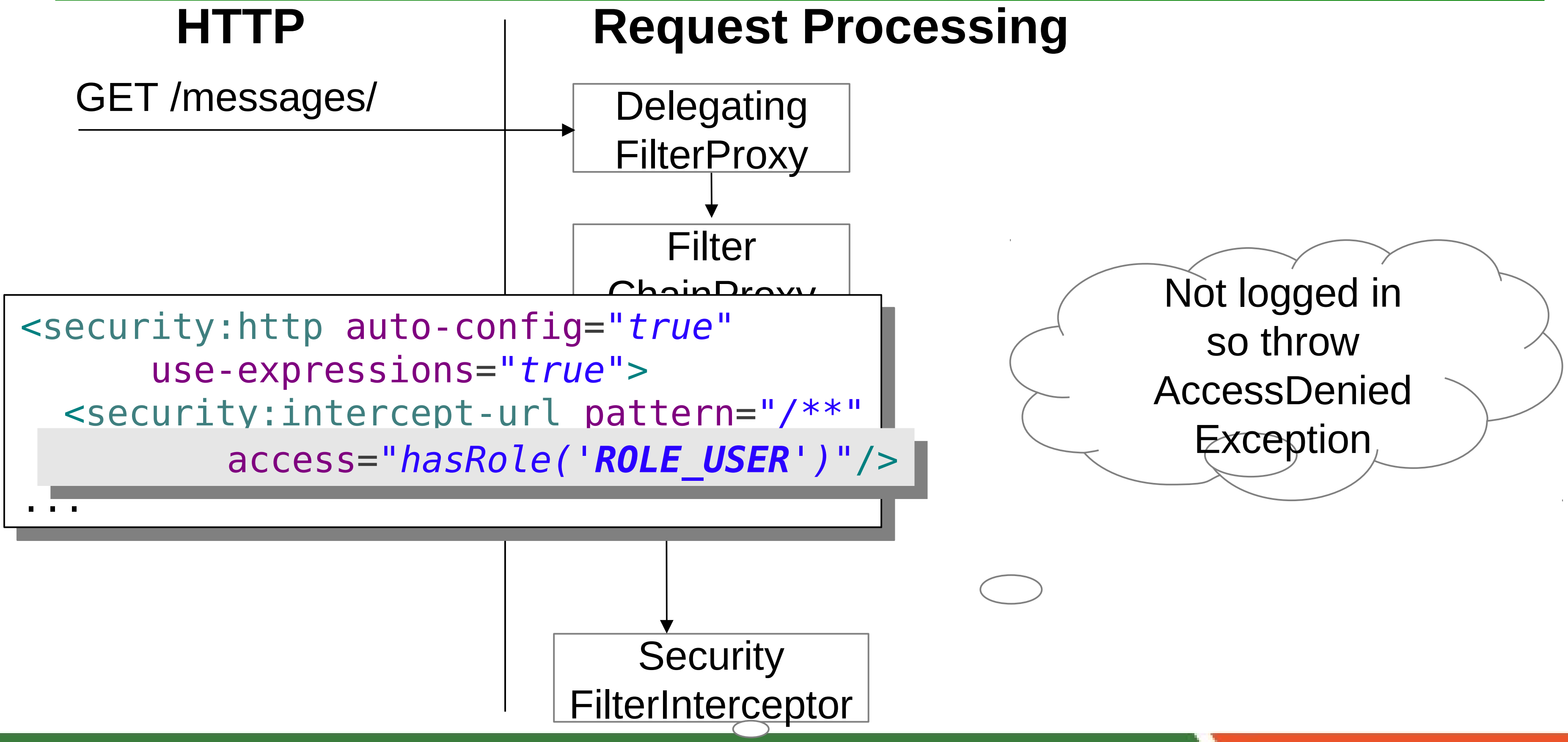
```
<security:http auto-config="true"
        use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')" />
...
```

Unauthenticated Request to Protected Resource



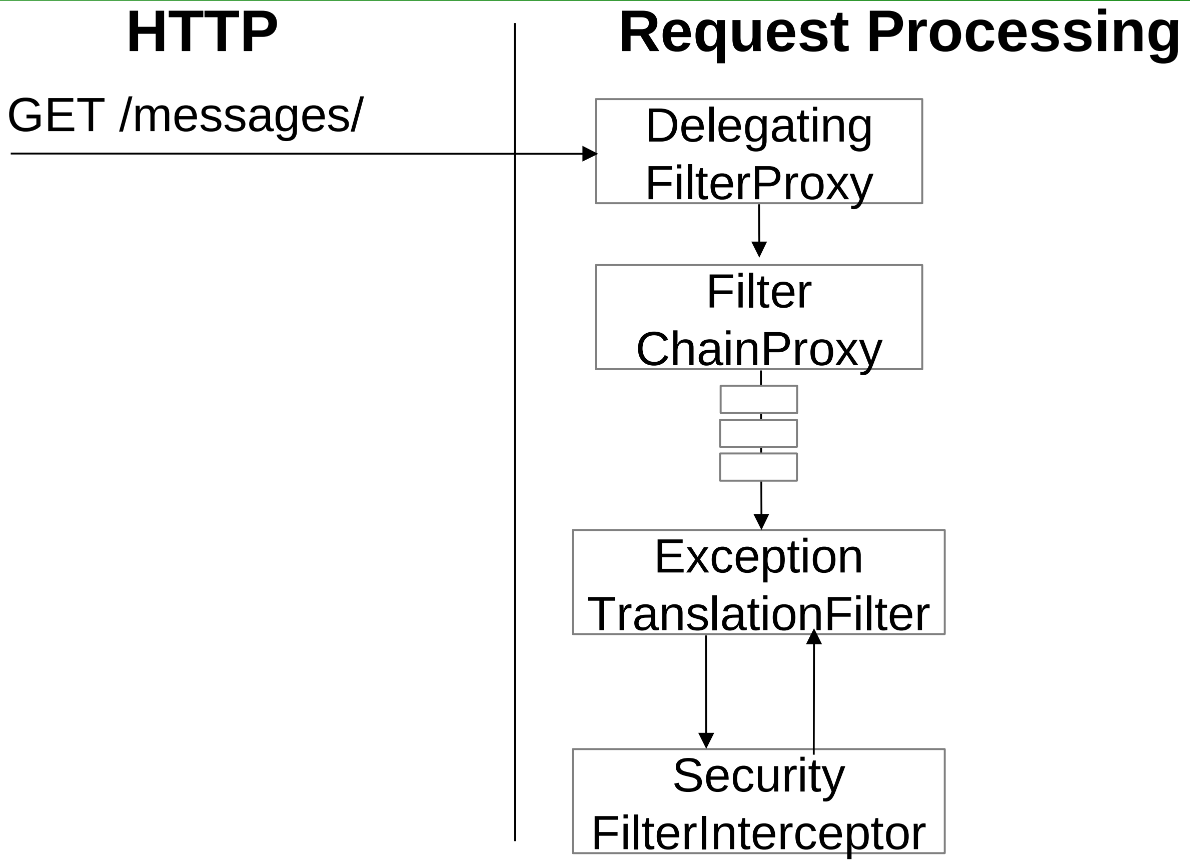
```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
          access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

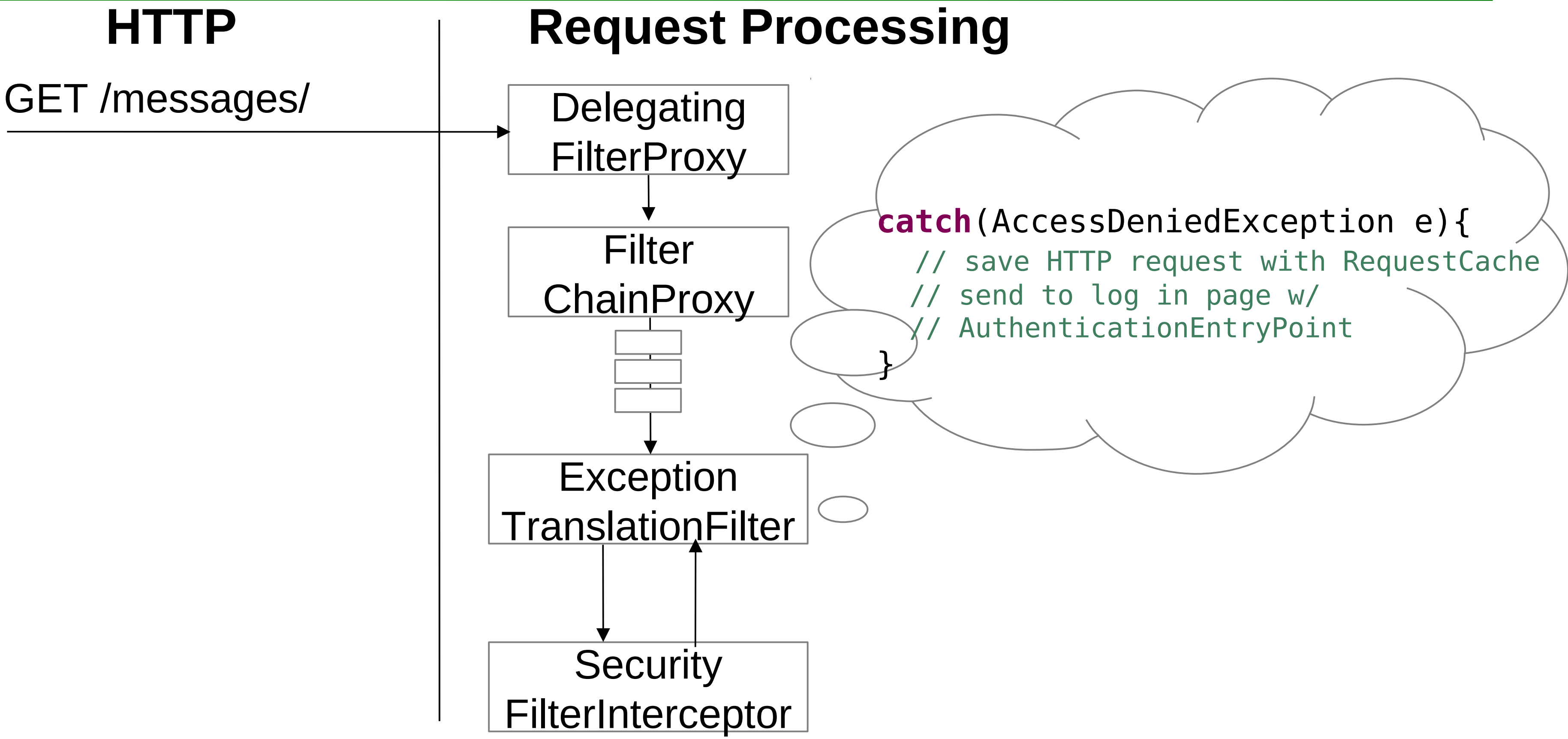


```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
           access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

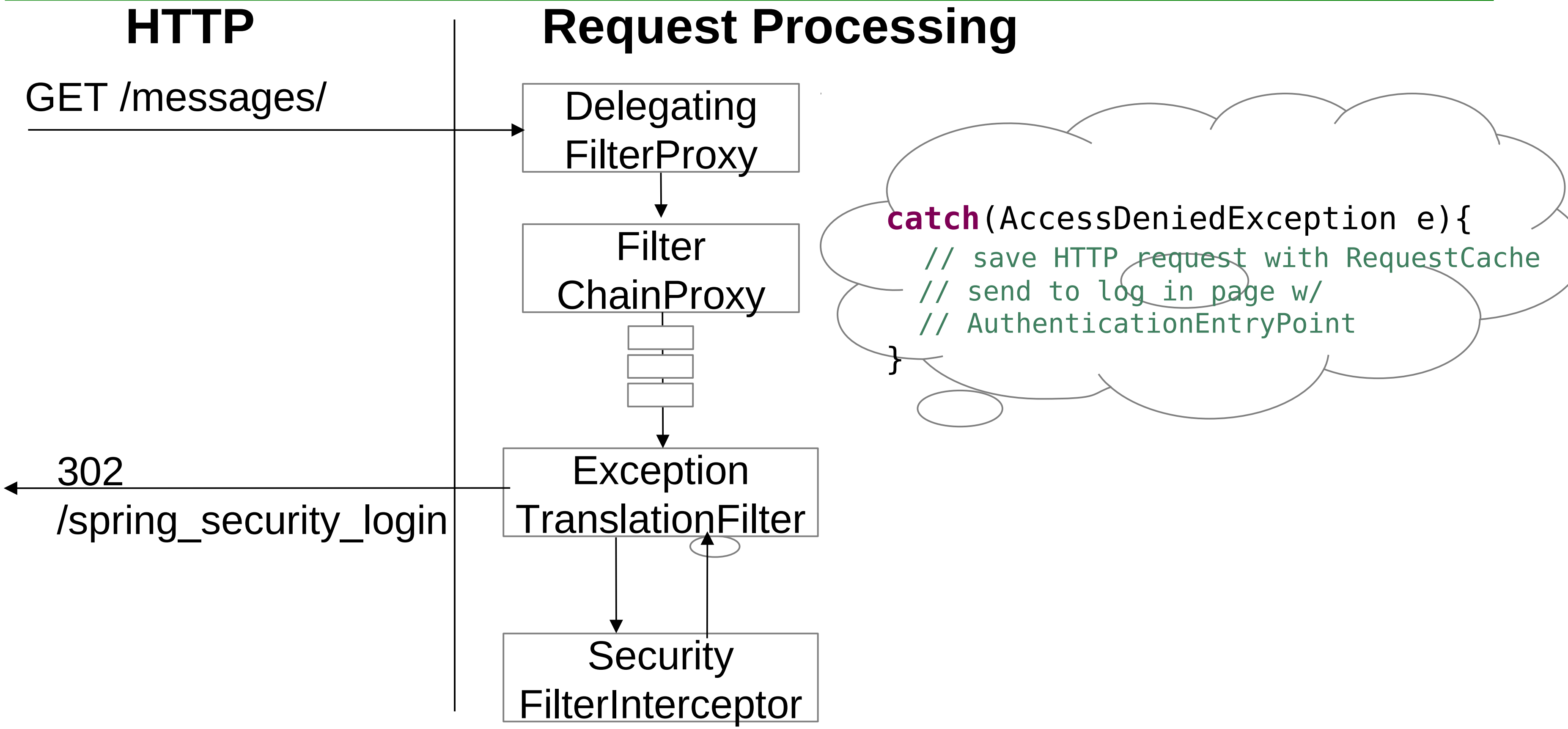


Unauthenticated Request to Protected Resource



```
catch(AccessDeniedException e){
   // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

Unauthenticated Request to Protected Resource


HTTP
Request Processing
GET /messages/
Delegating FilterProxy
Filter ChainProxy
Exception TranslationFilter
Security FilterInterceptor
302 /spring_security_login

```
catch(AccessDeniedException e){
  // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

Ant Patterns

Spring Security uses an AntPathRequestMatcher to determine if a URL matches the current URL. The following rules are used when matching:

- Query parameters are not included in the match
- The context path is not included in the match
- ? matches one character
- * matches zero or more characters (not a directory delimiter i.e. /)
- ** matches zero or more 'directories' in a path

Ant Patterns - Examples

Ant Pattern examples that assume a context path of /messages

Pattern	Description	Full Path	Path to Match
/**	Matches any URL		
/*	Matches anything in root folder	/messages/1	**/1**
		/messages/2?a=b	**/2**
		/messages/1/	**/1/**

Ant Patterns - Examples

Ant Pattern examples that assume a context path of /messages

Pattern	Description	Full Path	Path to Match
/1/**	Matches anything that starts with /1	/messages/1	**/1**
		/messages/1?a=b	**/1**
		/messages/1/	**/1/**
		/messages/1/view	**/1/view**
		/messages/other/	**/other/**
		/messages/2/view	**/2/view**

Ant Patterns - Examples

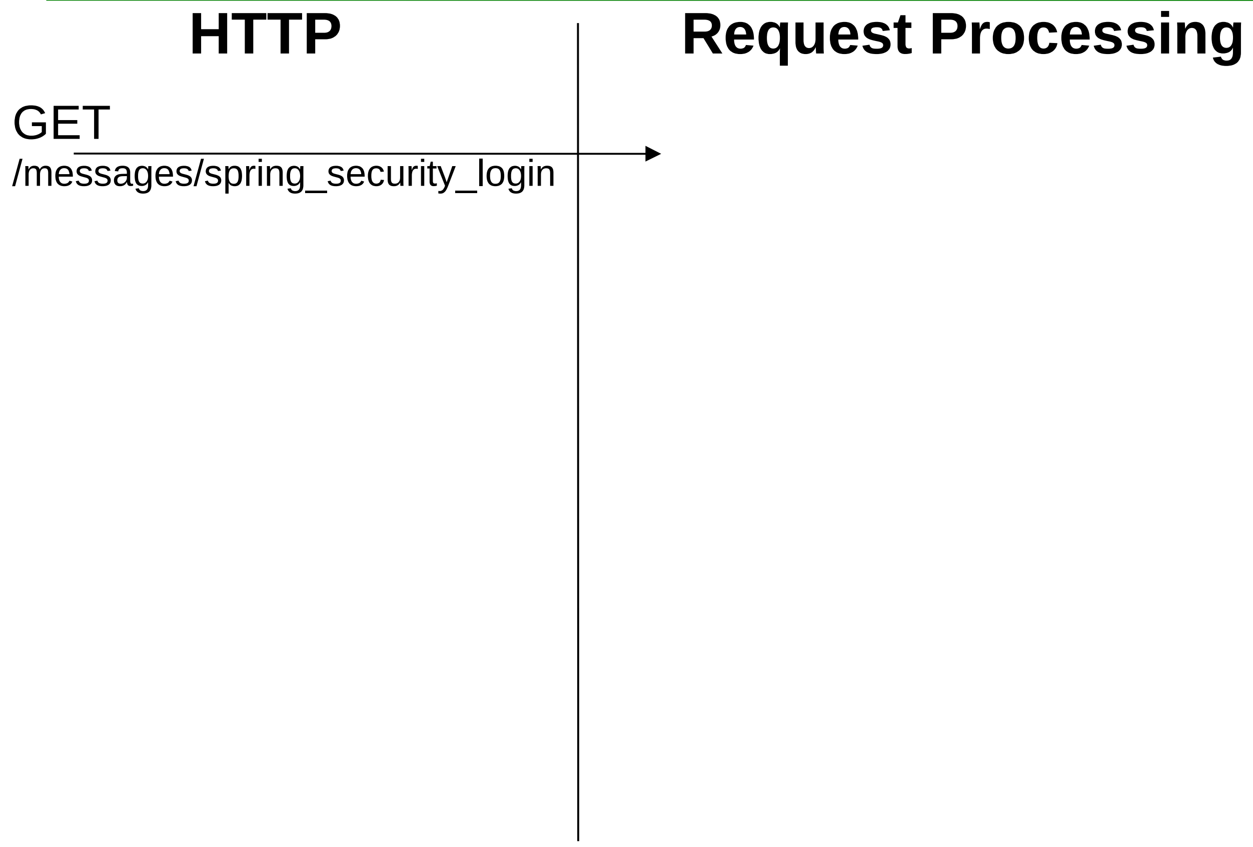
Be careful when using pattern matching

Pattern	Description	Full Path	Path to Match
/**/*.css	Matches anything that ends with .css	/messages/styles/main.css	**/styles/main.css**
		/messages/1	**/1**
		/messages/1.css	**/1.css**

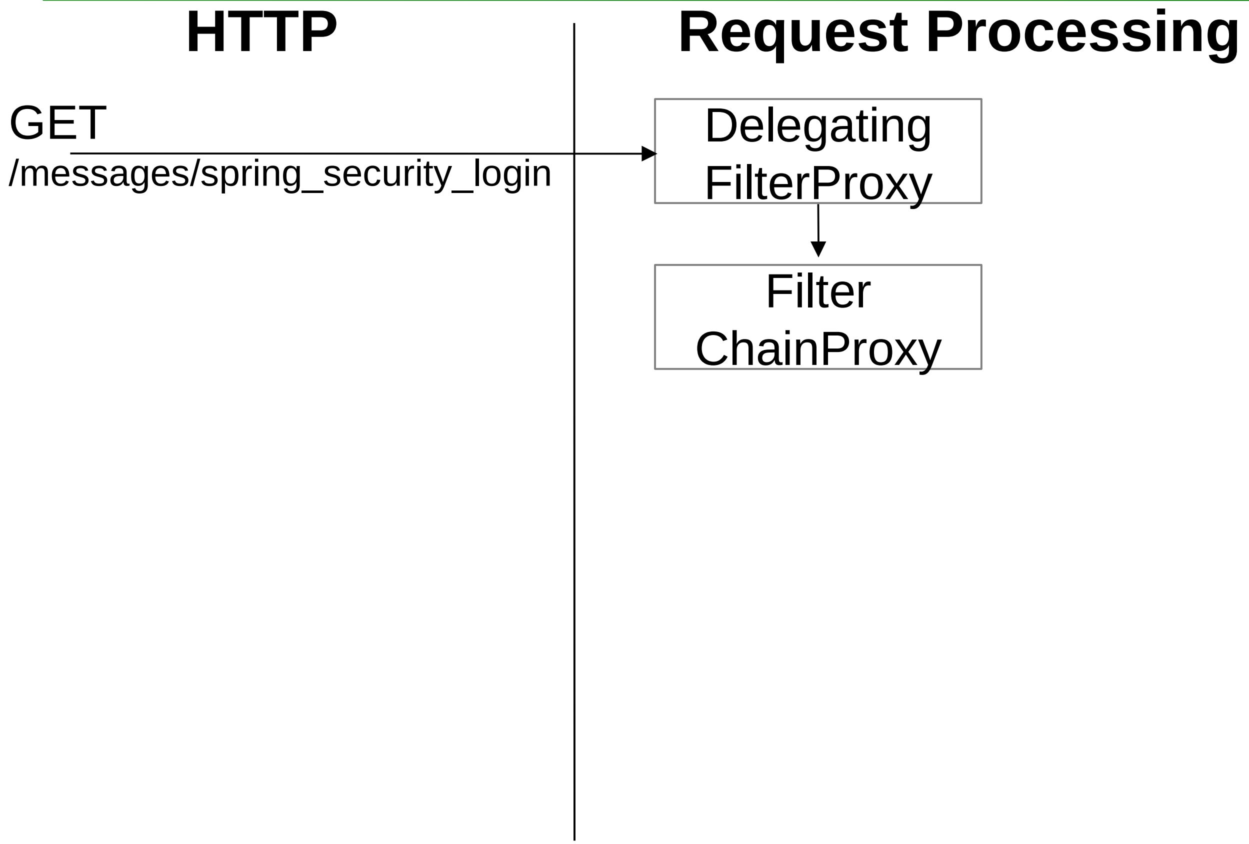
Ant Patterns

- The less restrictive the mapping the easier it is for a malicious user to bypass
- Spring MVC will treat /1.css the same as /1, so a malicious user can use this to bypass security constraints
- Other ways to bypass URL based security (i.e. path variables, non-normalized URLs, etc). Spring Security does have things in place to help protect you (i.e. HttpFirewall)
- Best to combine URL Security with Method Security to provide defense in depth

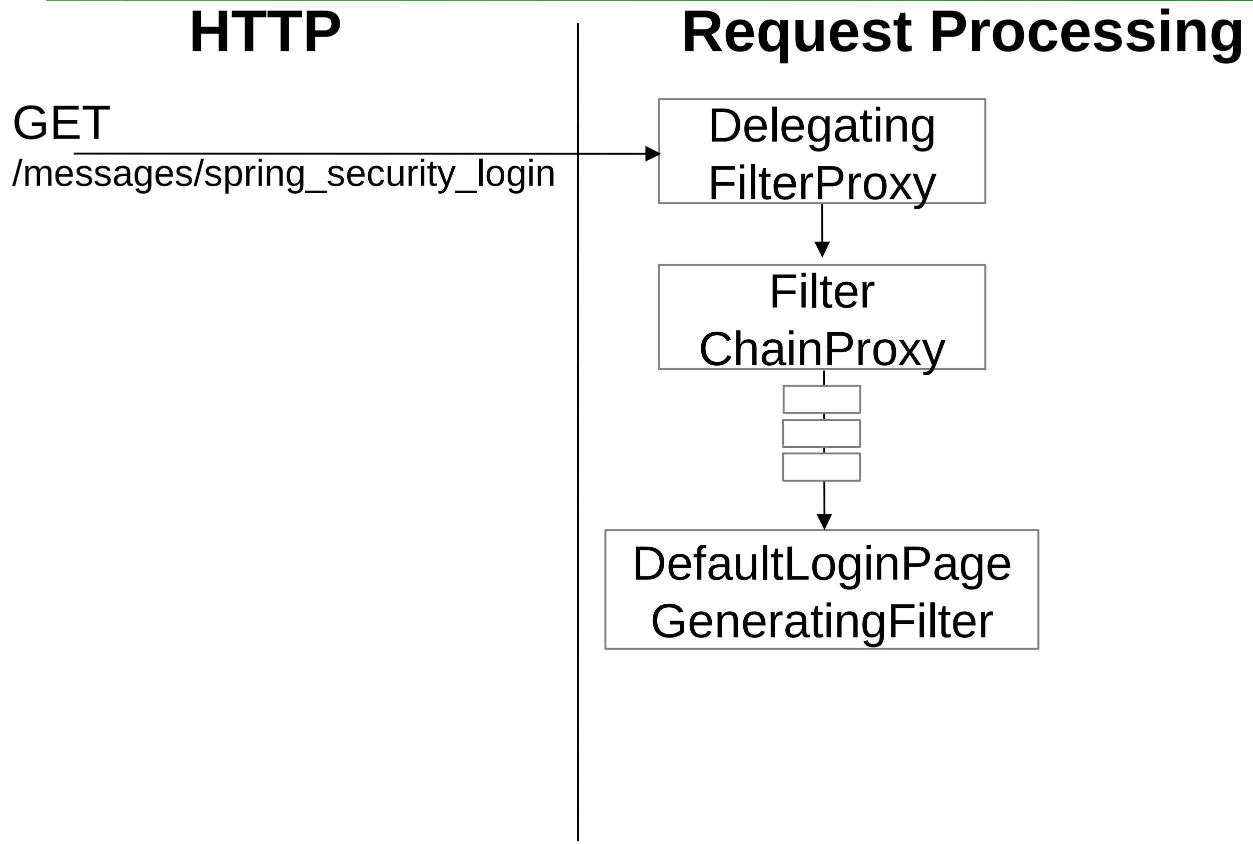
Requesting log in page



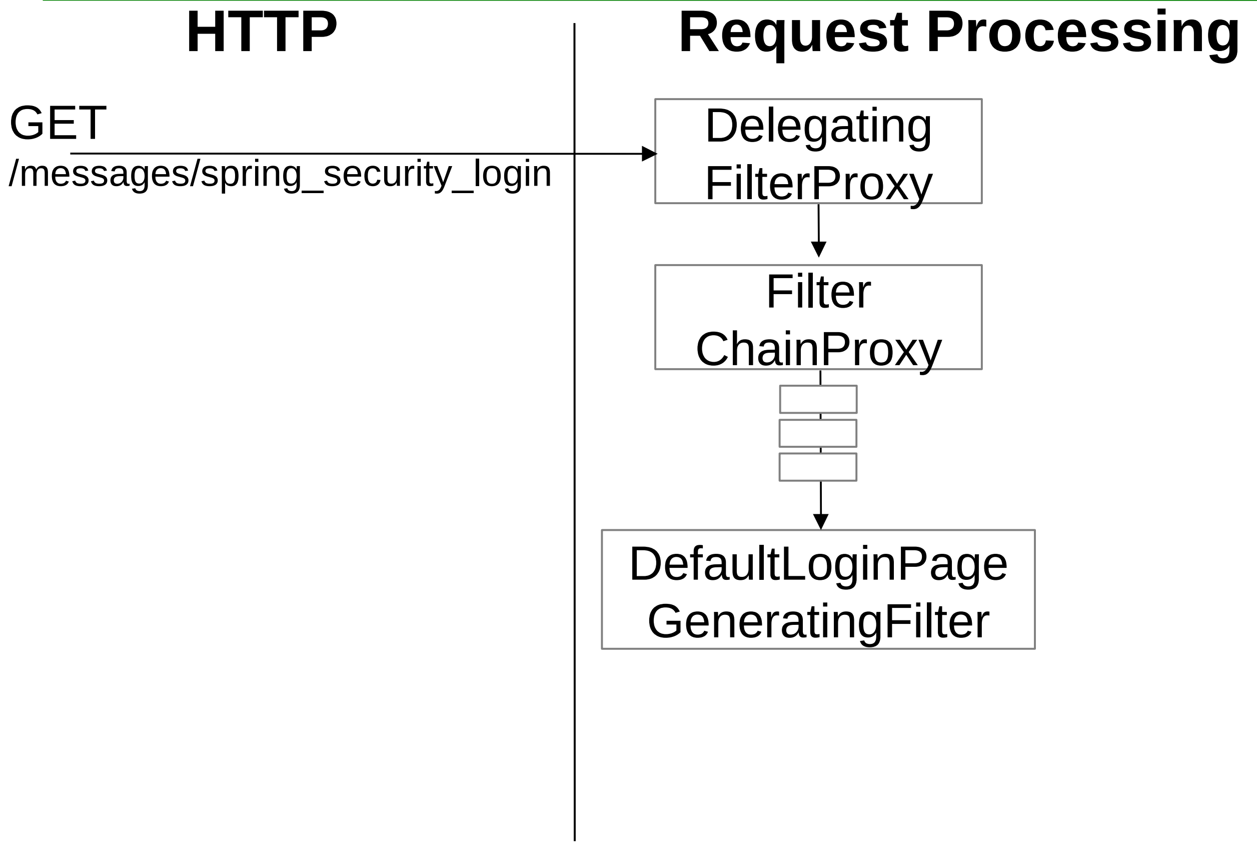
Requesting log in page



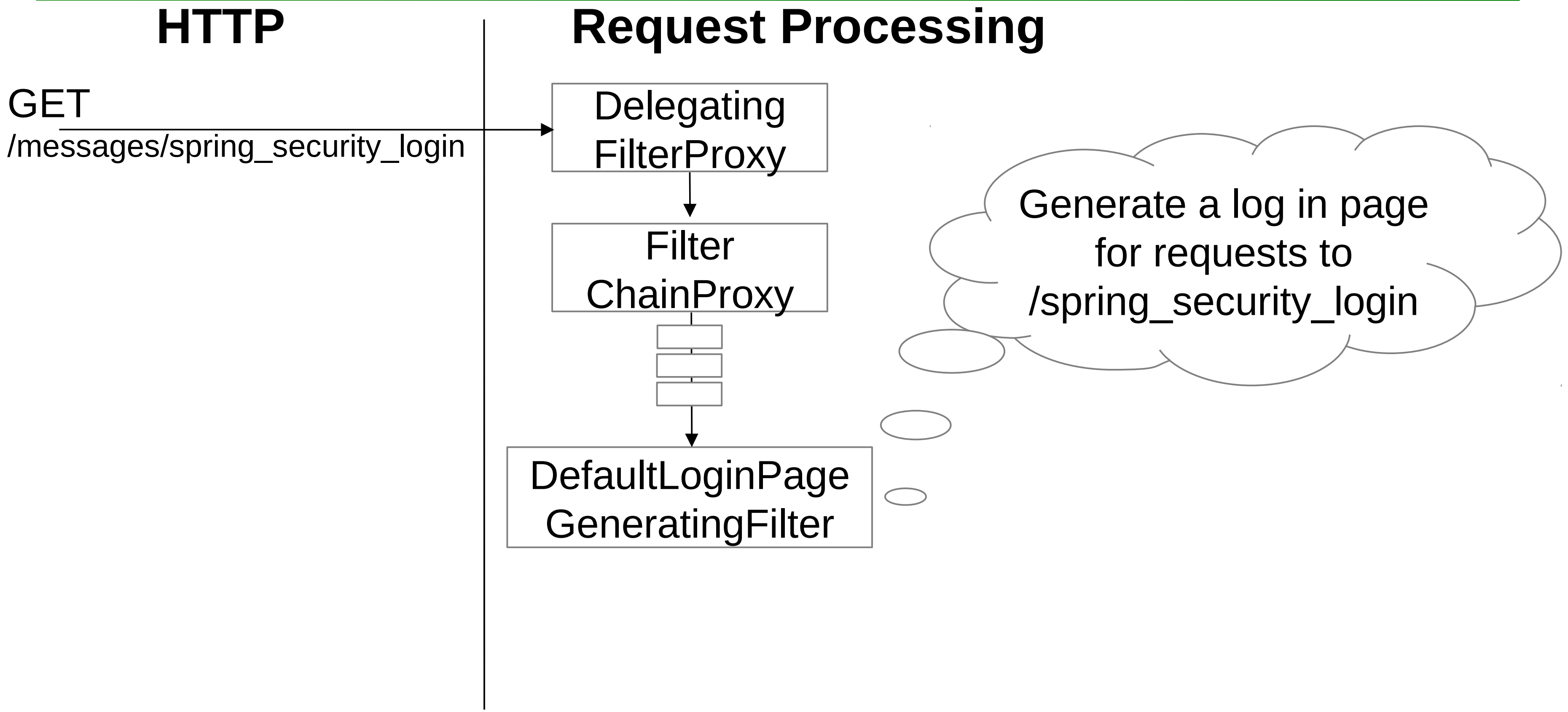
Requesting log in page


HTTP
Request Processing
GET
/messages/spring_security_login
Delegating FilterProxy
Filter ChainProxy
DefaultLoginPage GeneratingFilter

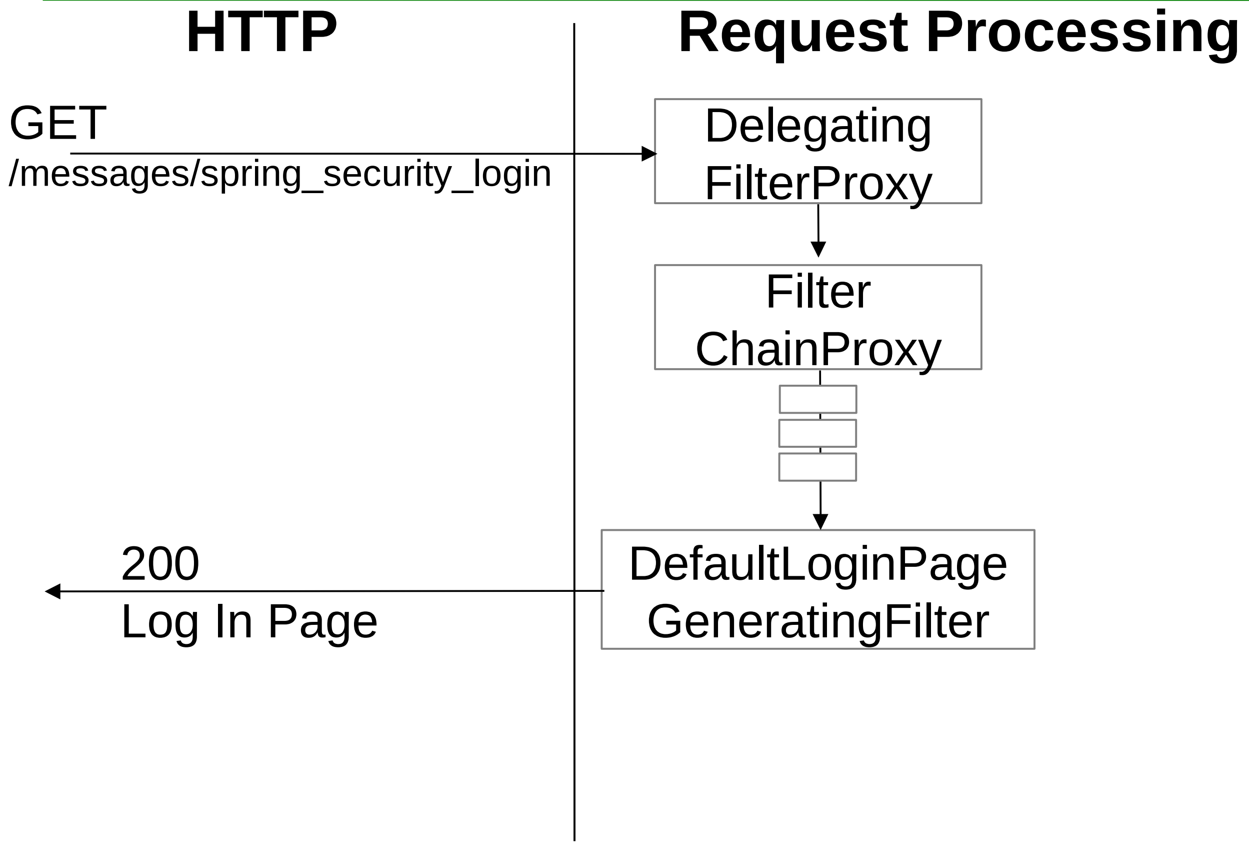
Requesting log in page


HTTP
Request Processing
GET
/messages/spring_security_login
Delegating FilterProxy
Filter ChainProxy
DefaultLoginPage GeneratingFilter

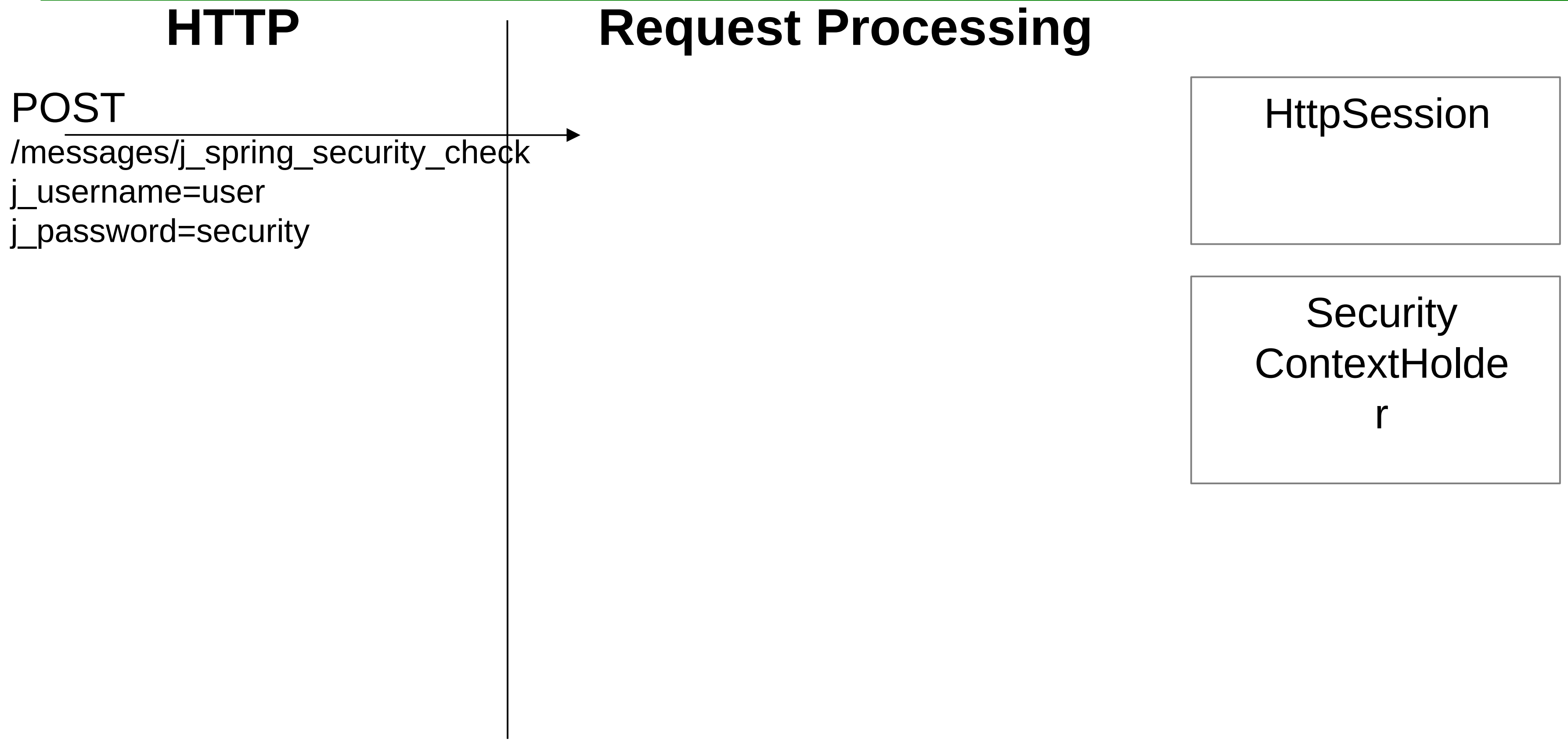
Requesting log in page


HTTP
Request Processing
GET /messages/spring_security_login
DelegatingFilterProxy
FilterChainProxy
DefaultLoginPageGeneratingFilter
Generate a log in page for requests to /spring_security_login

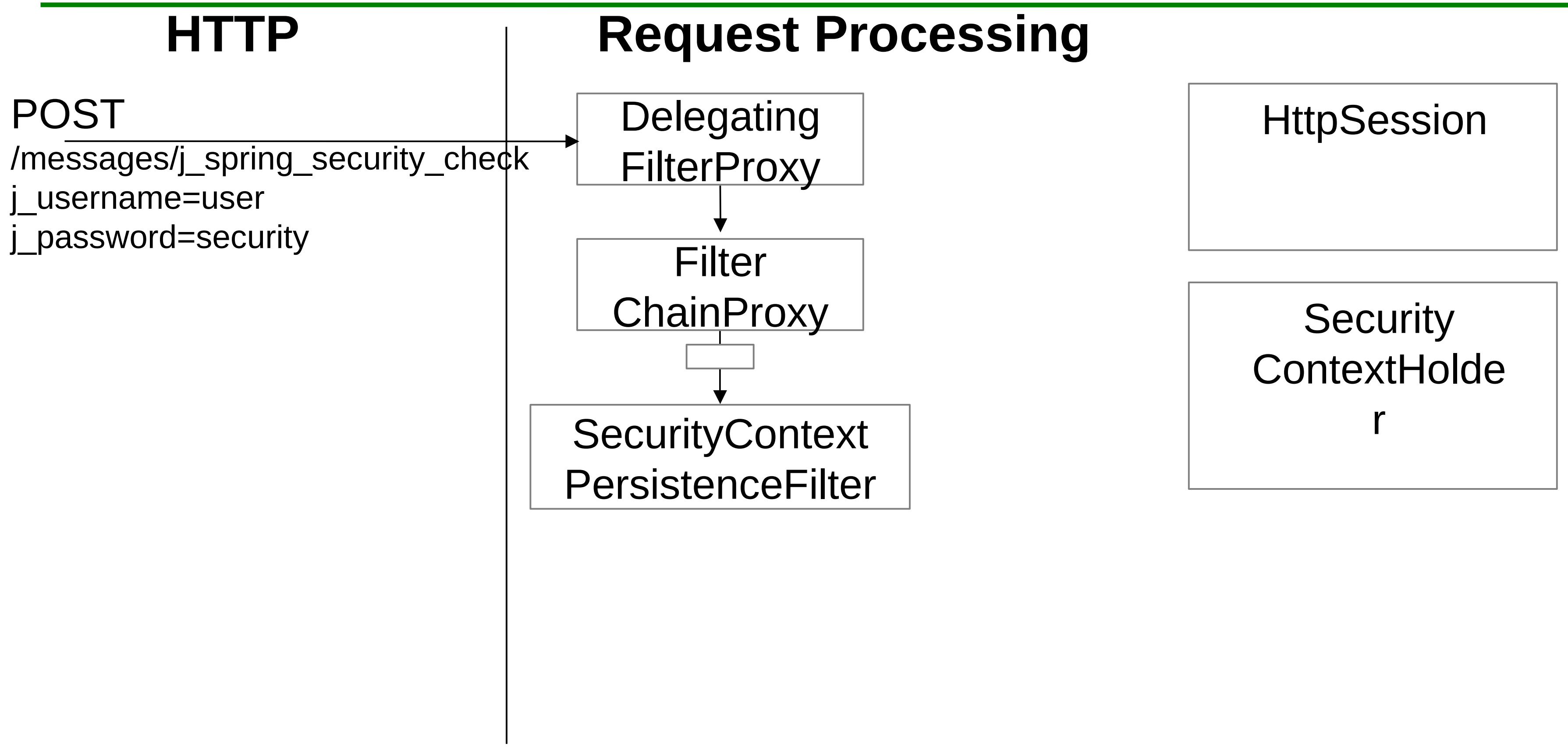
Requesting log in page



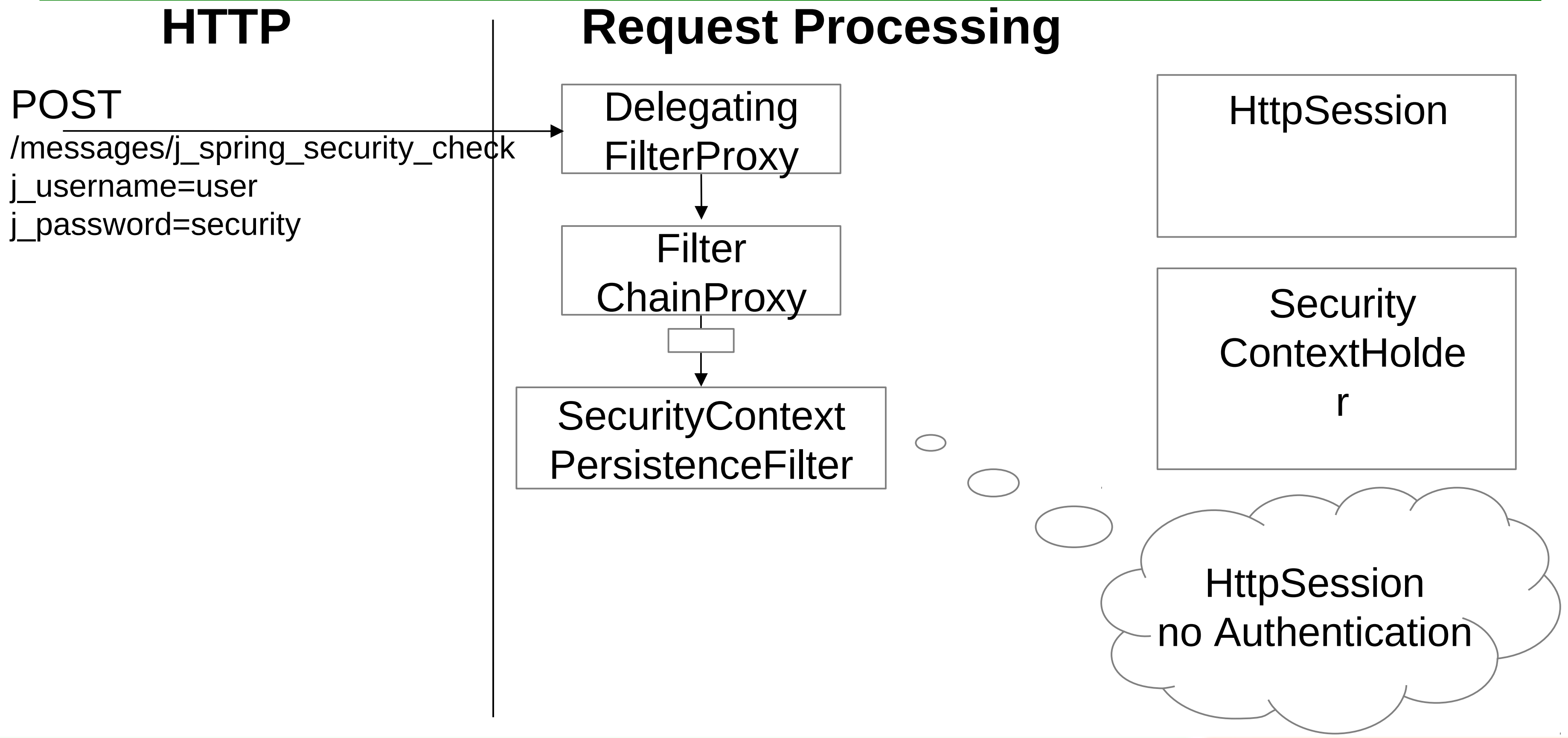
Authenticating via username & password



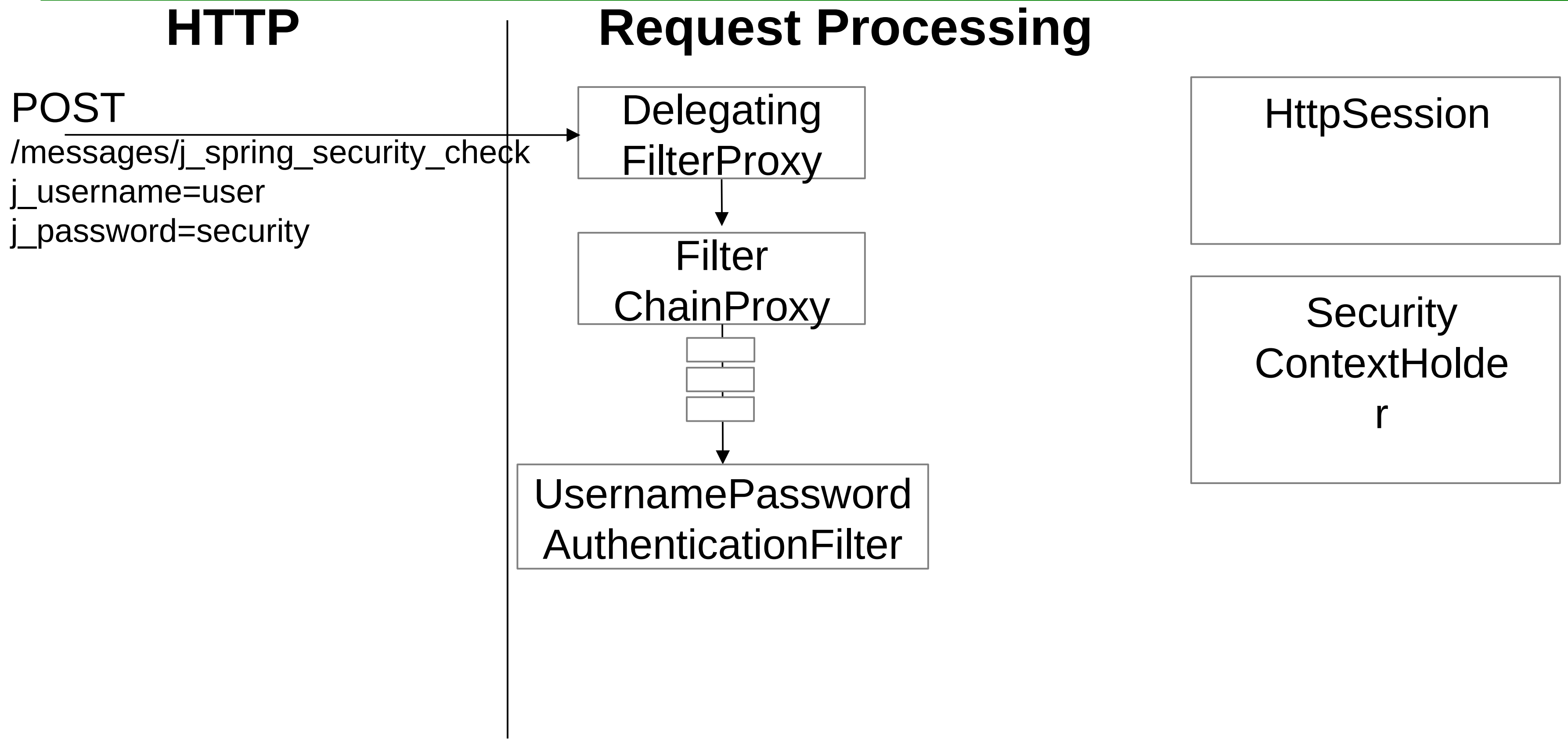
Authenticating via username & password



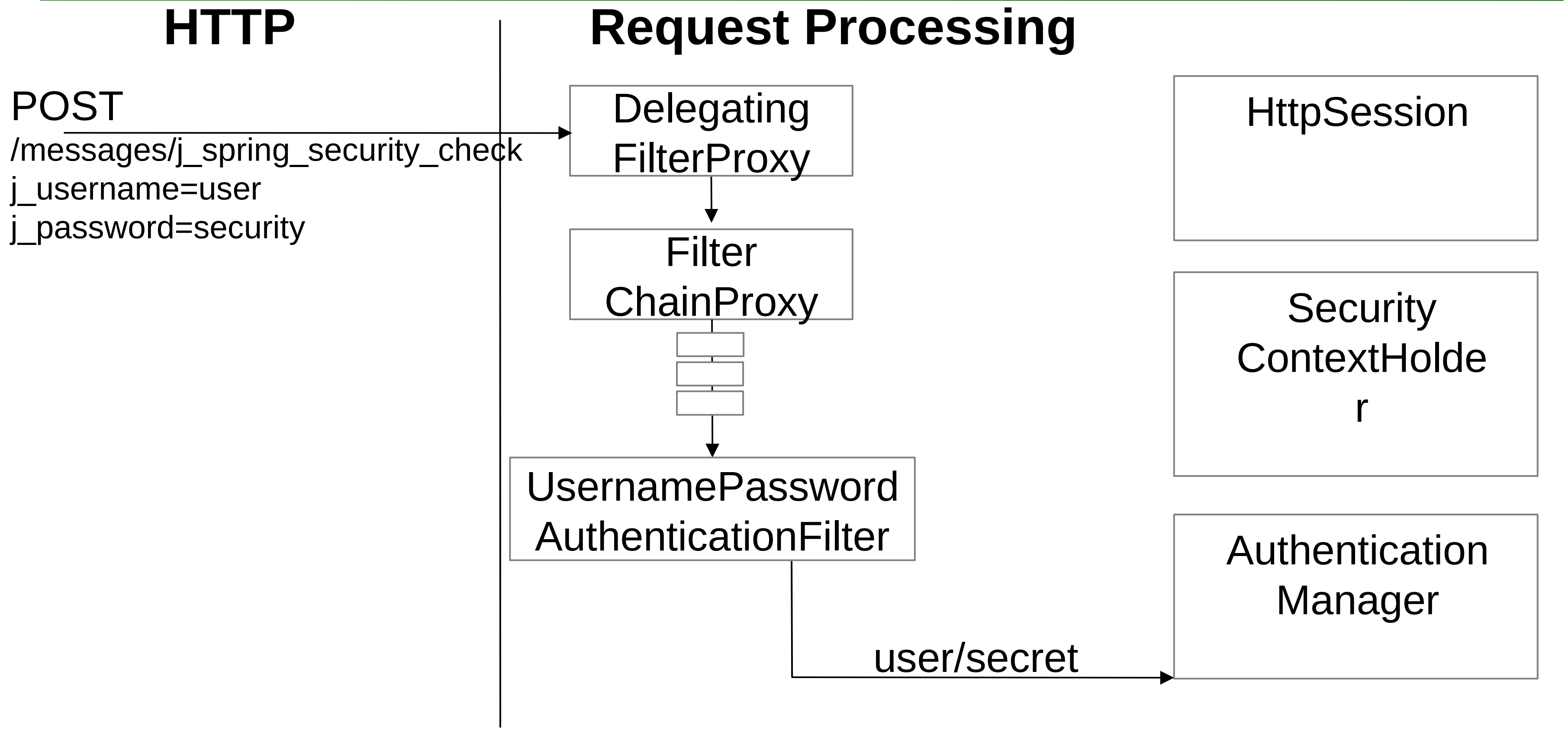
Authenticating via username & password



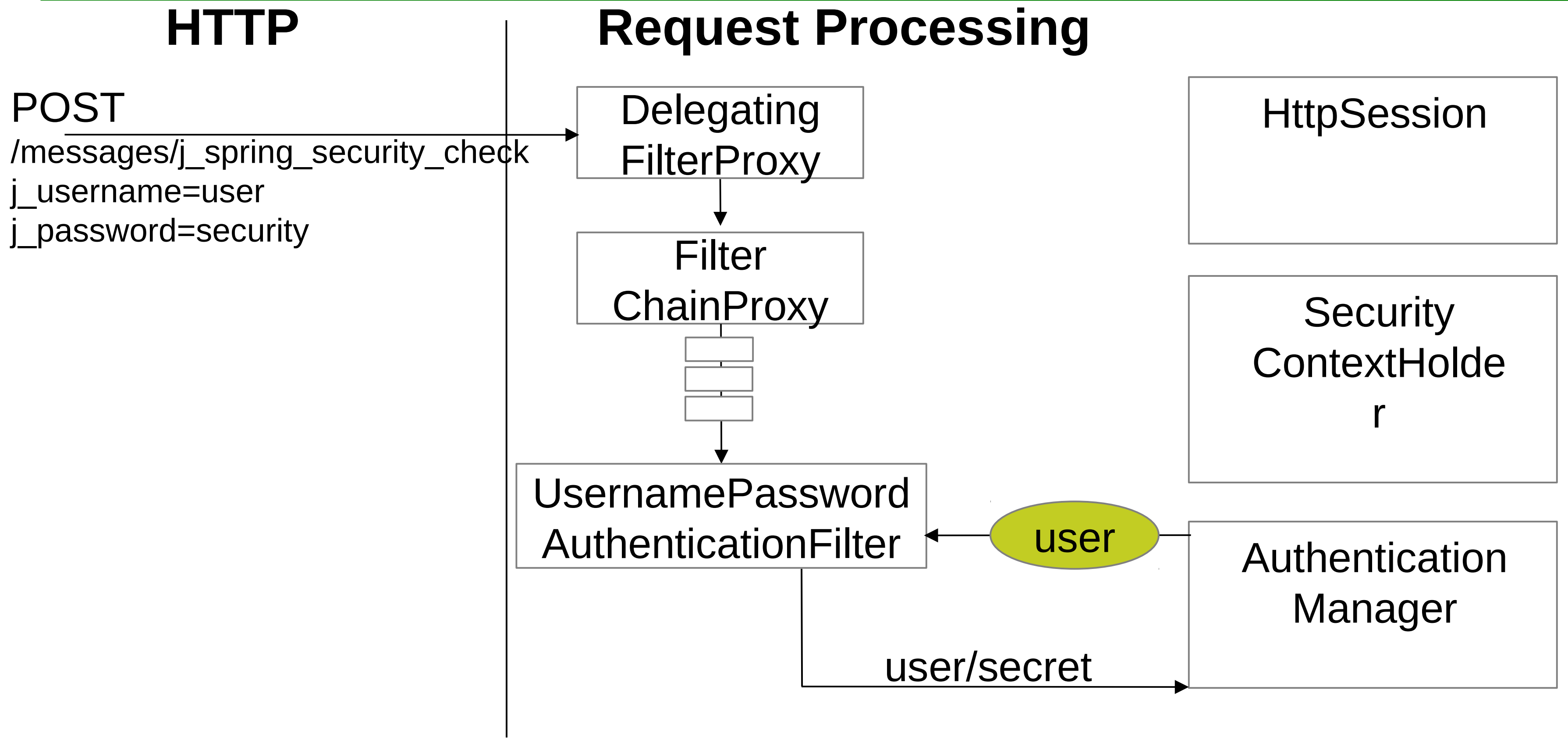
Authenticating via username & password



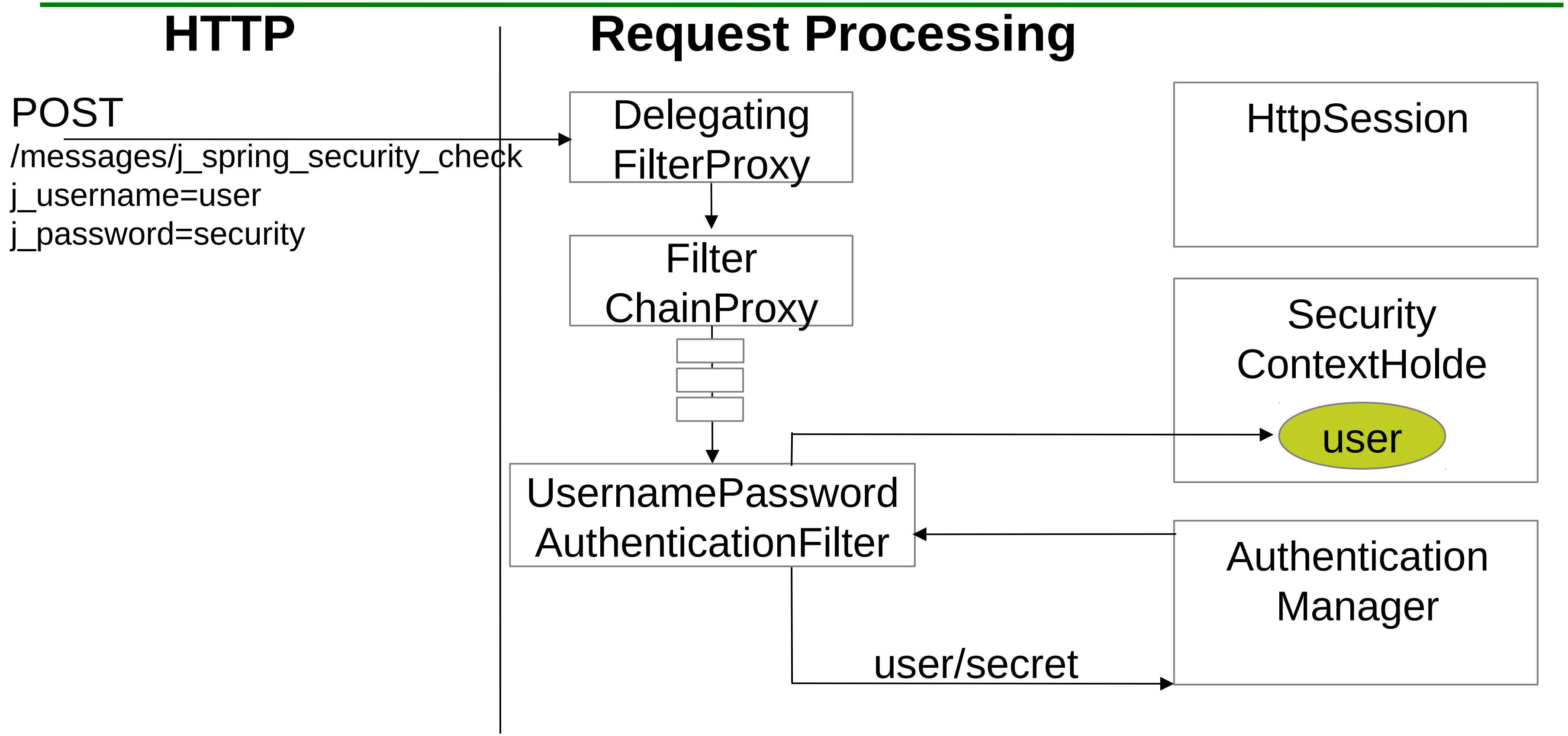
Authenticating via username & password



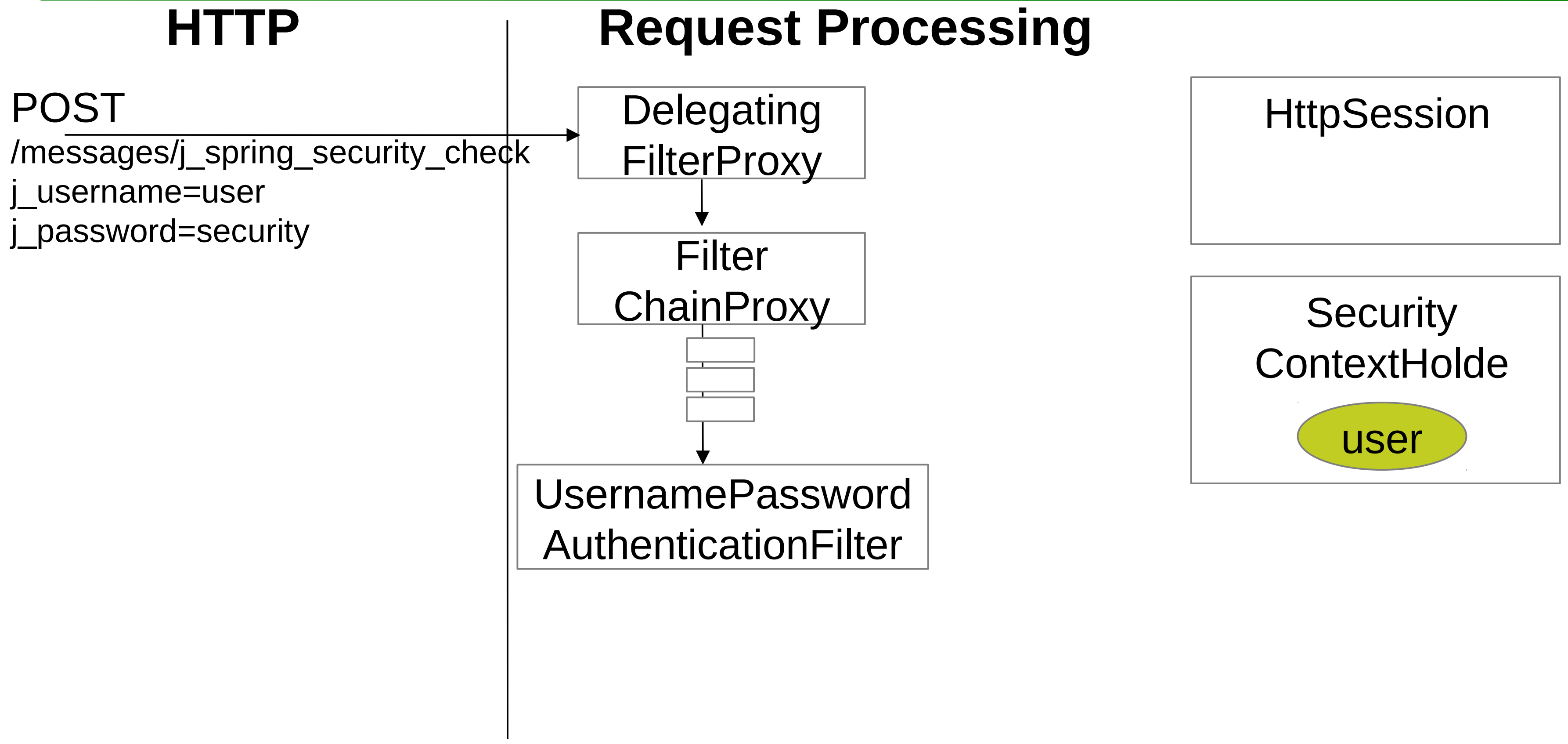
Authenticating via username & password



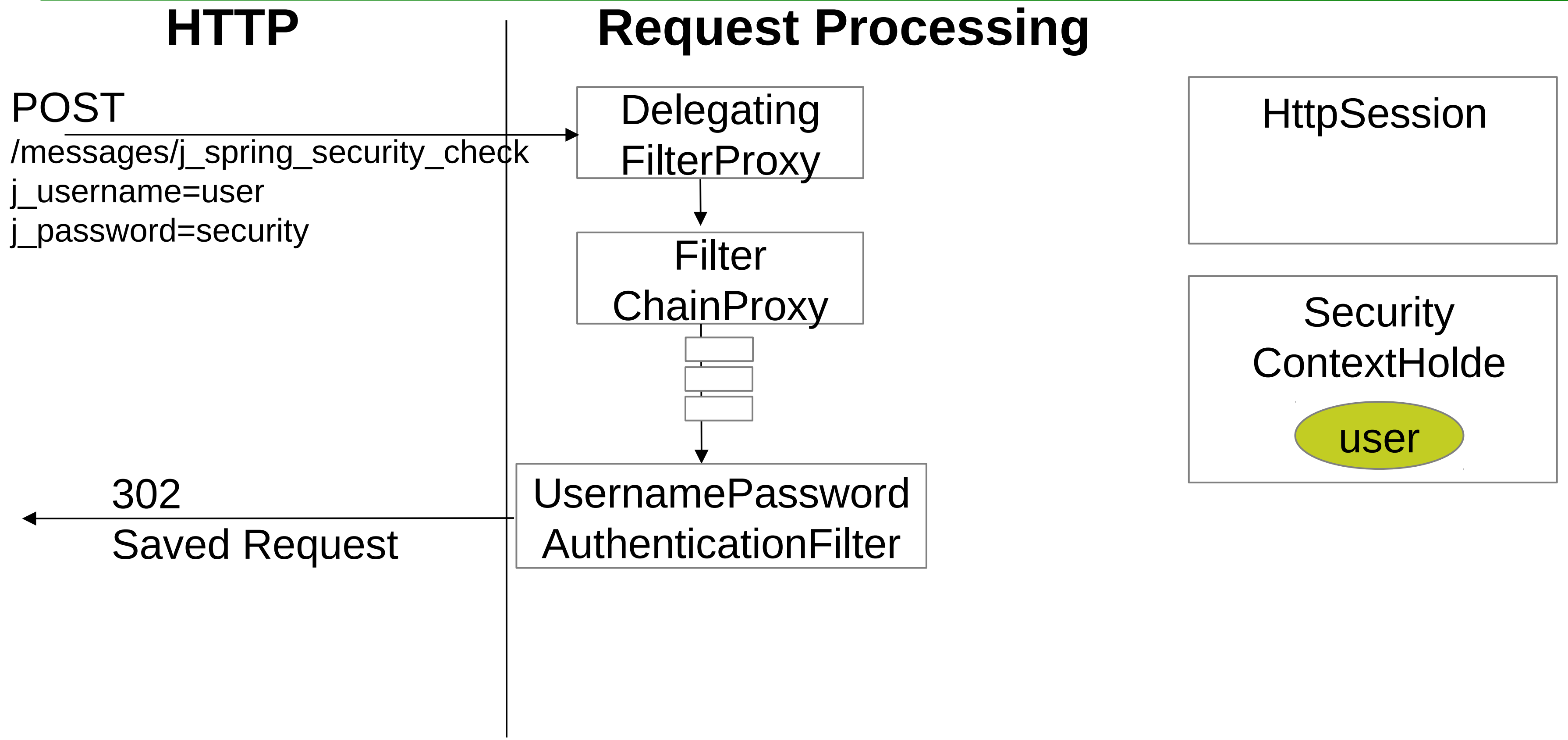
Authenticating via username & password



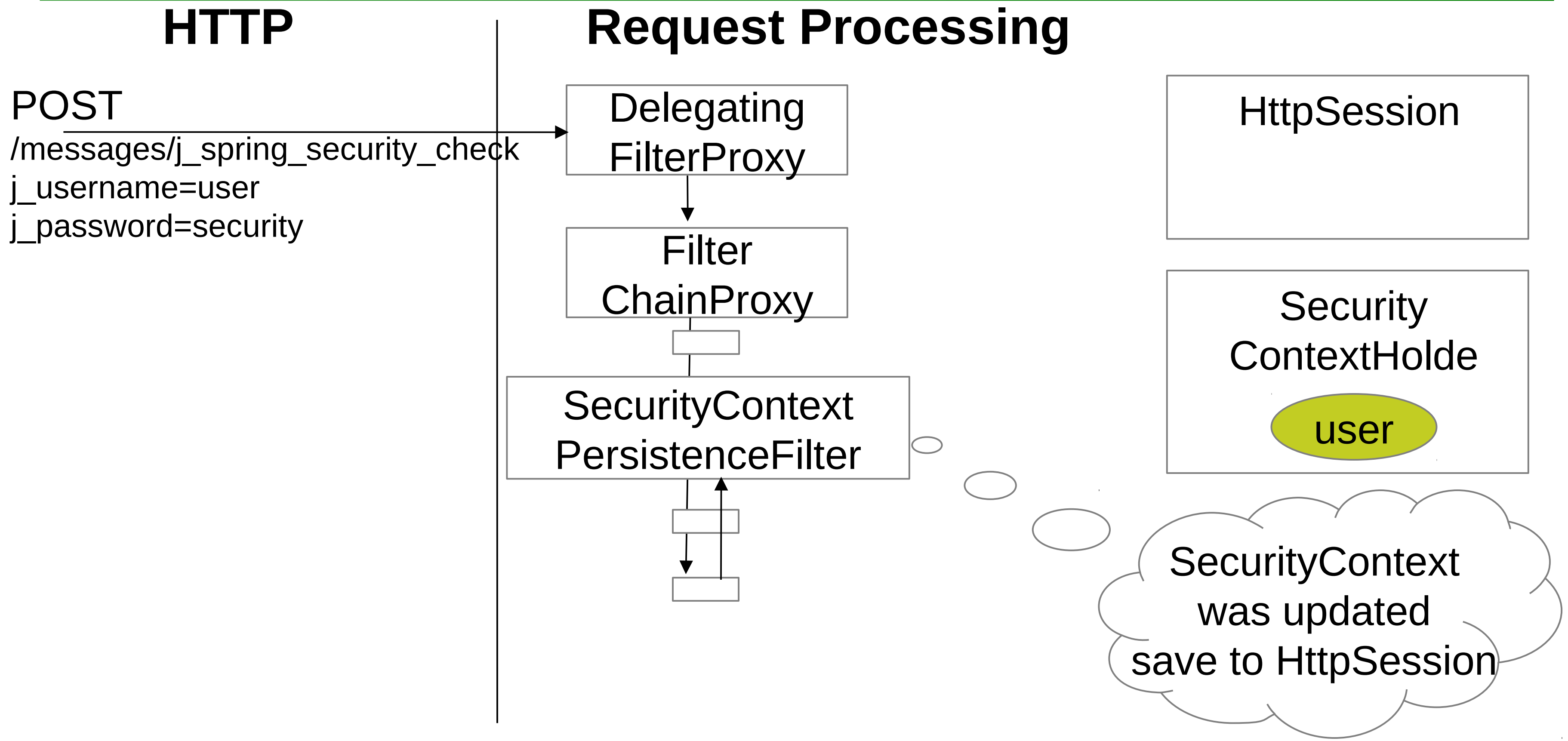
Authenticating via username & password



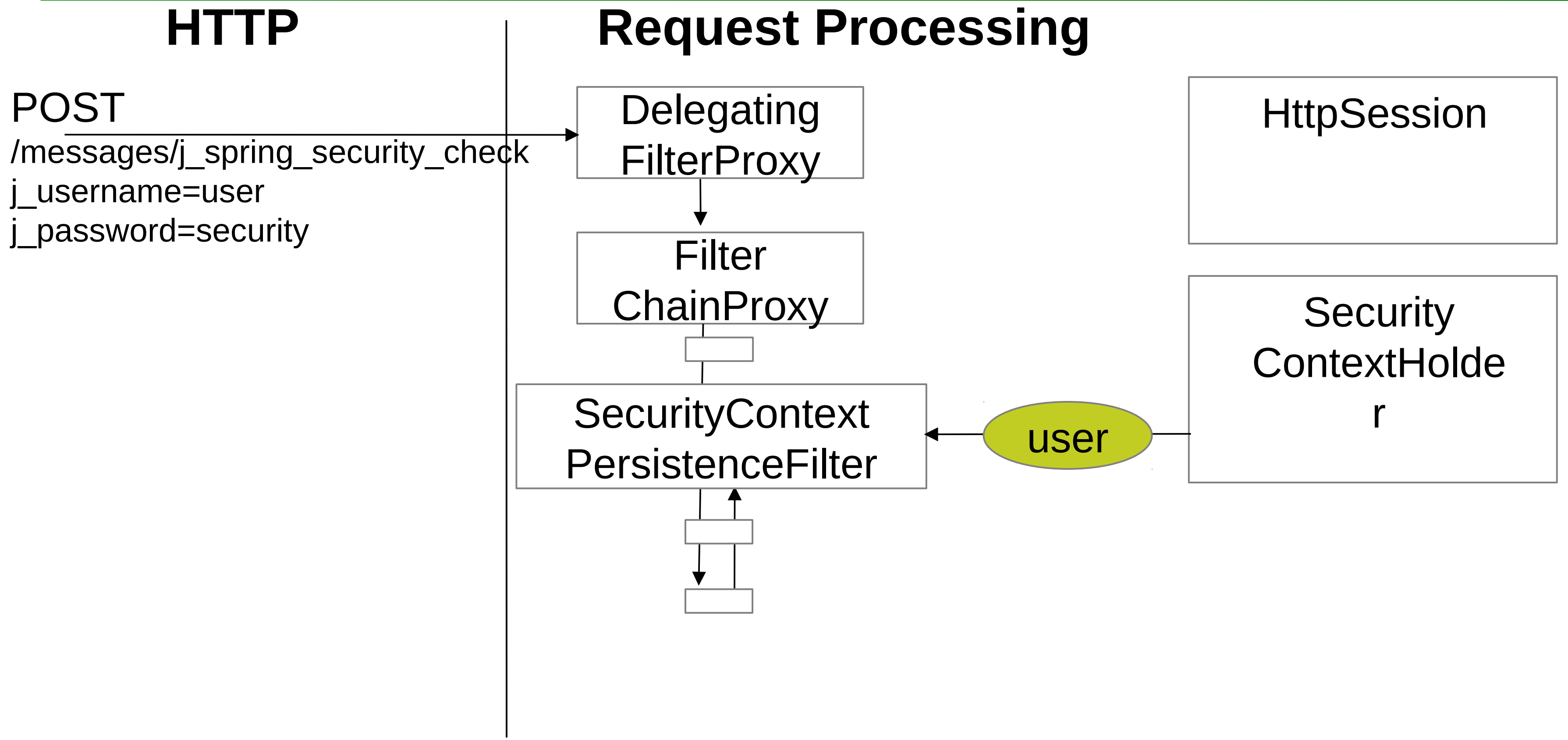
Authenticating via username & password



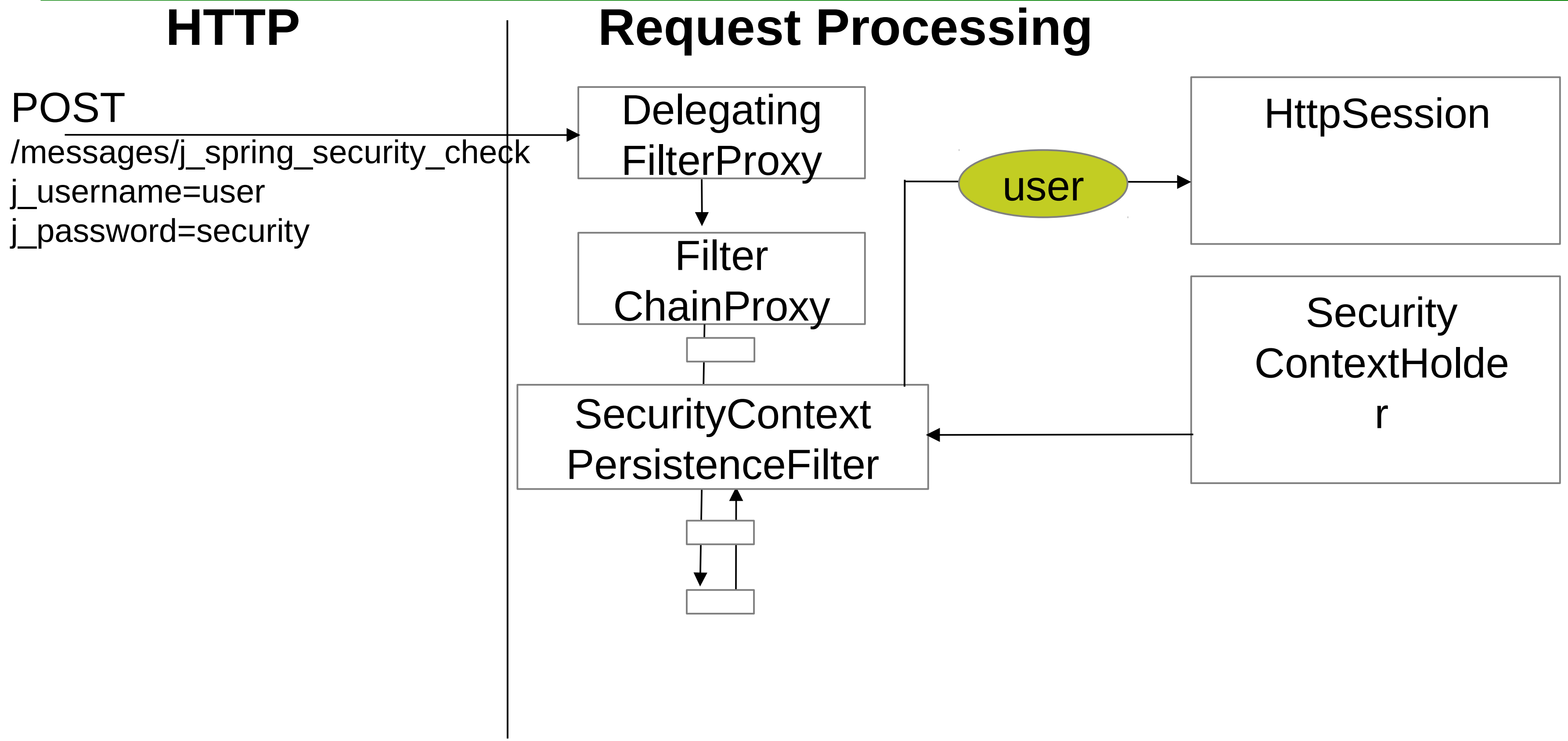
Authenticating via username & password



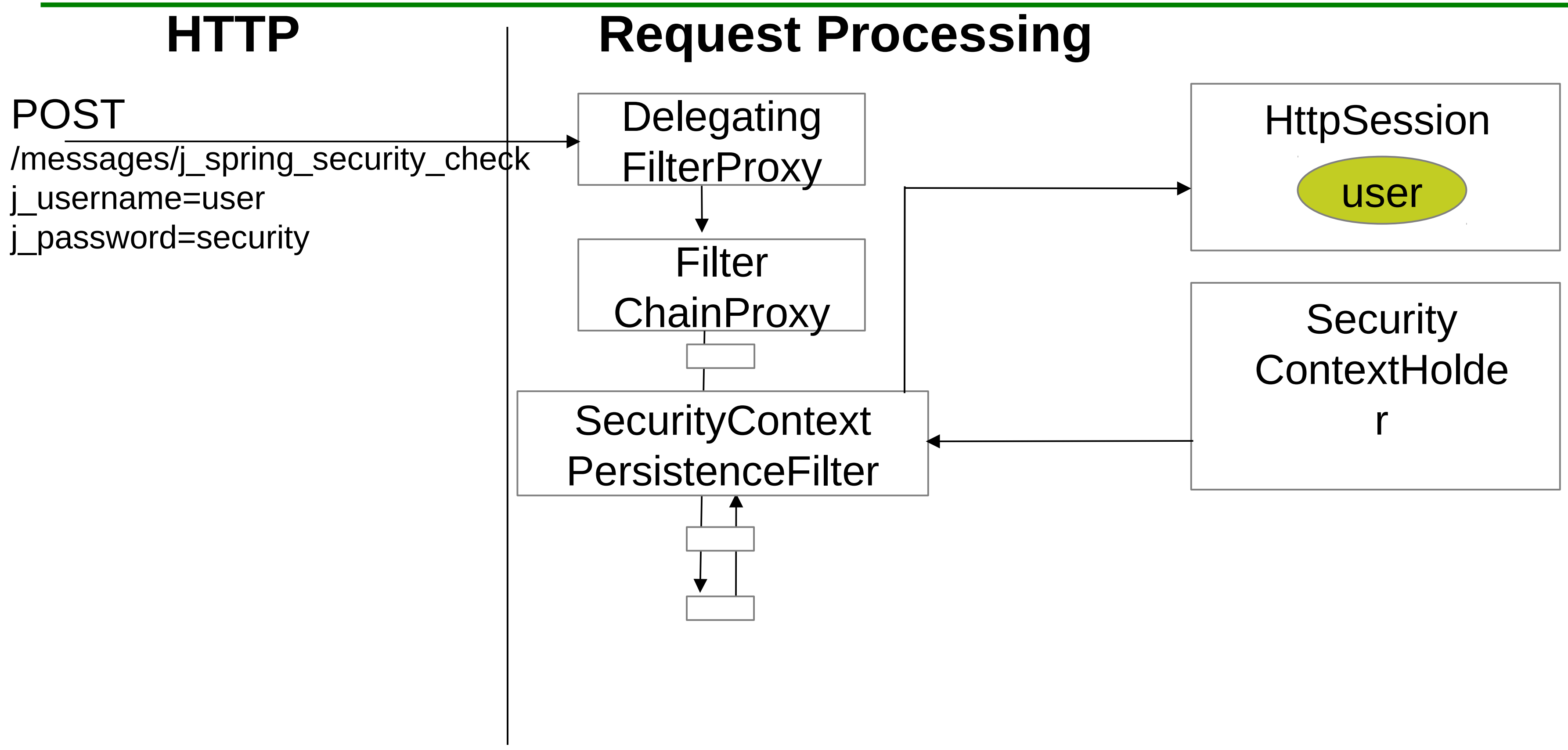
Authenticating via username & password



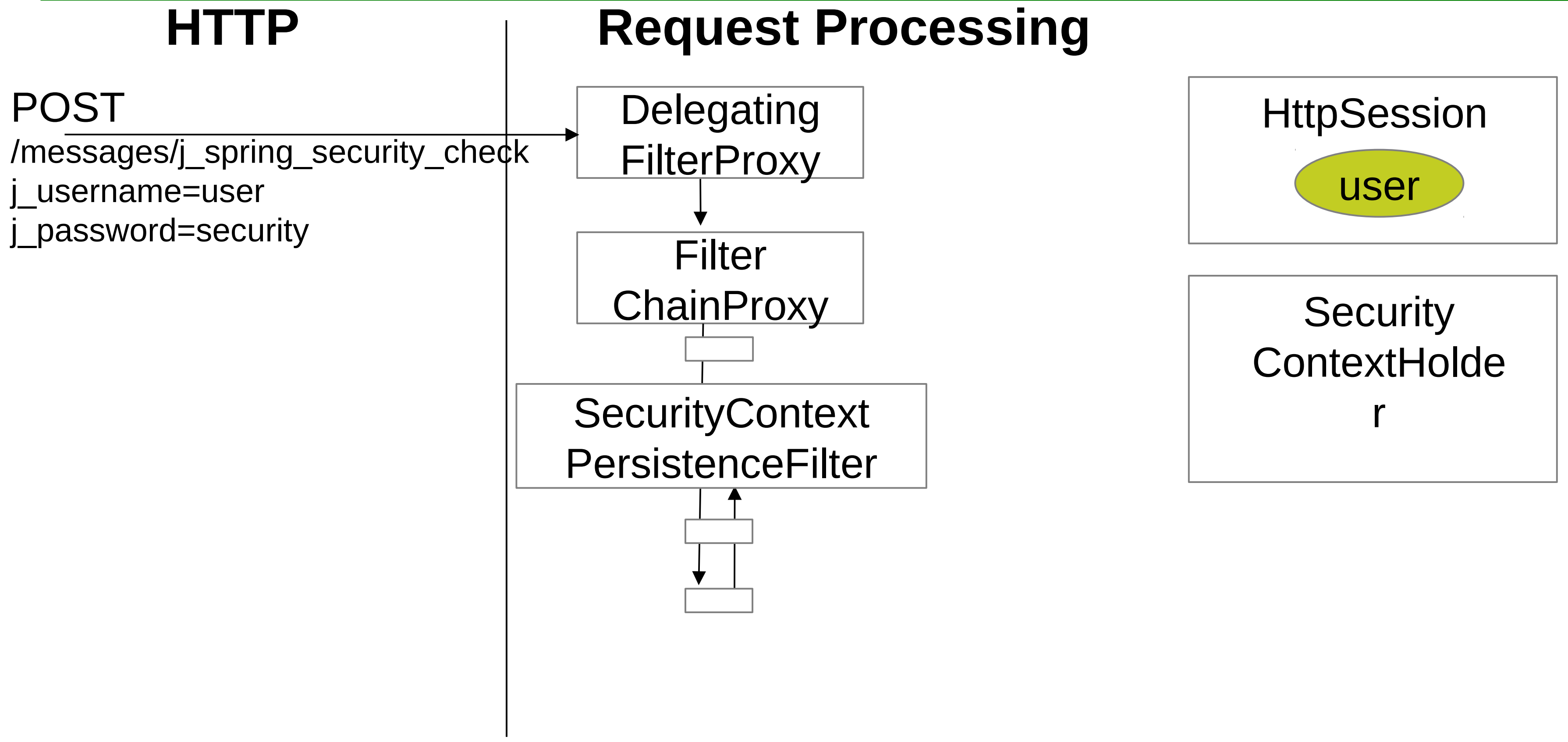
Authenticating via username & password



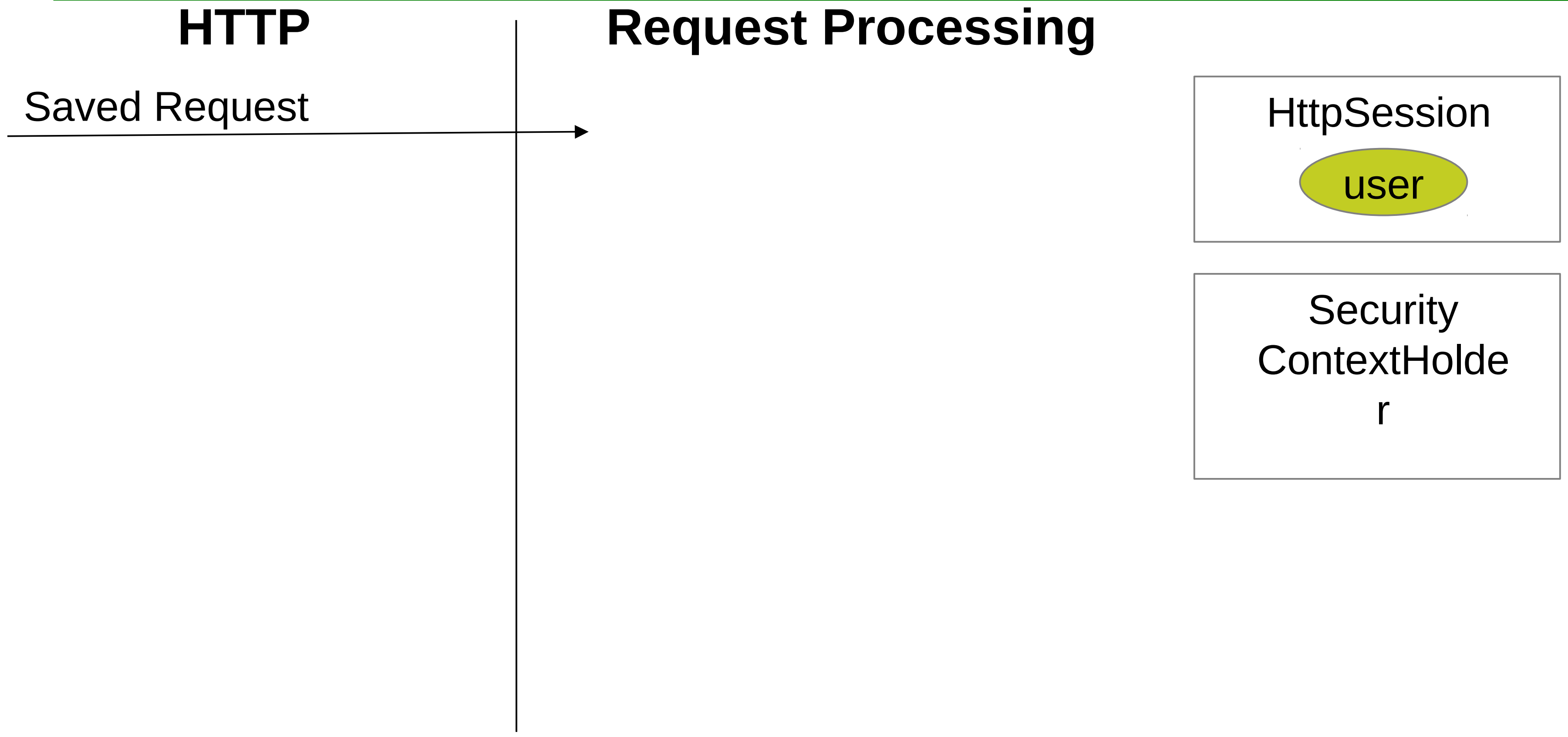
Authenticating via username & password



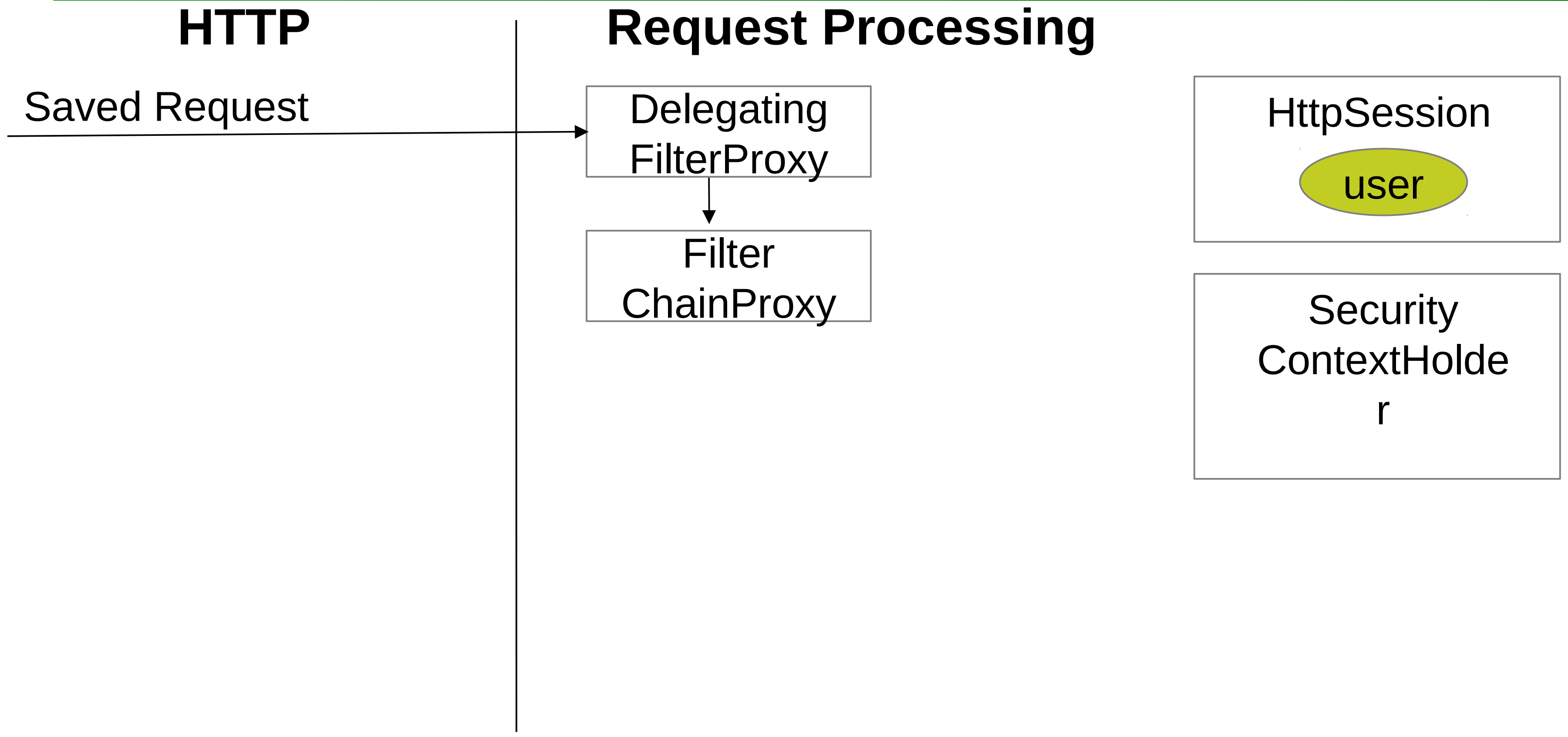
Authenticating via username & password



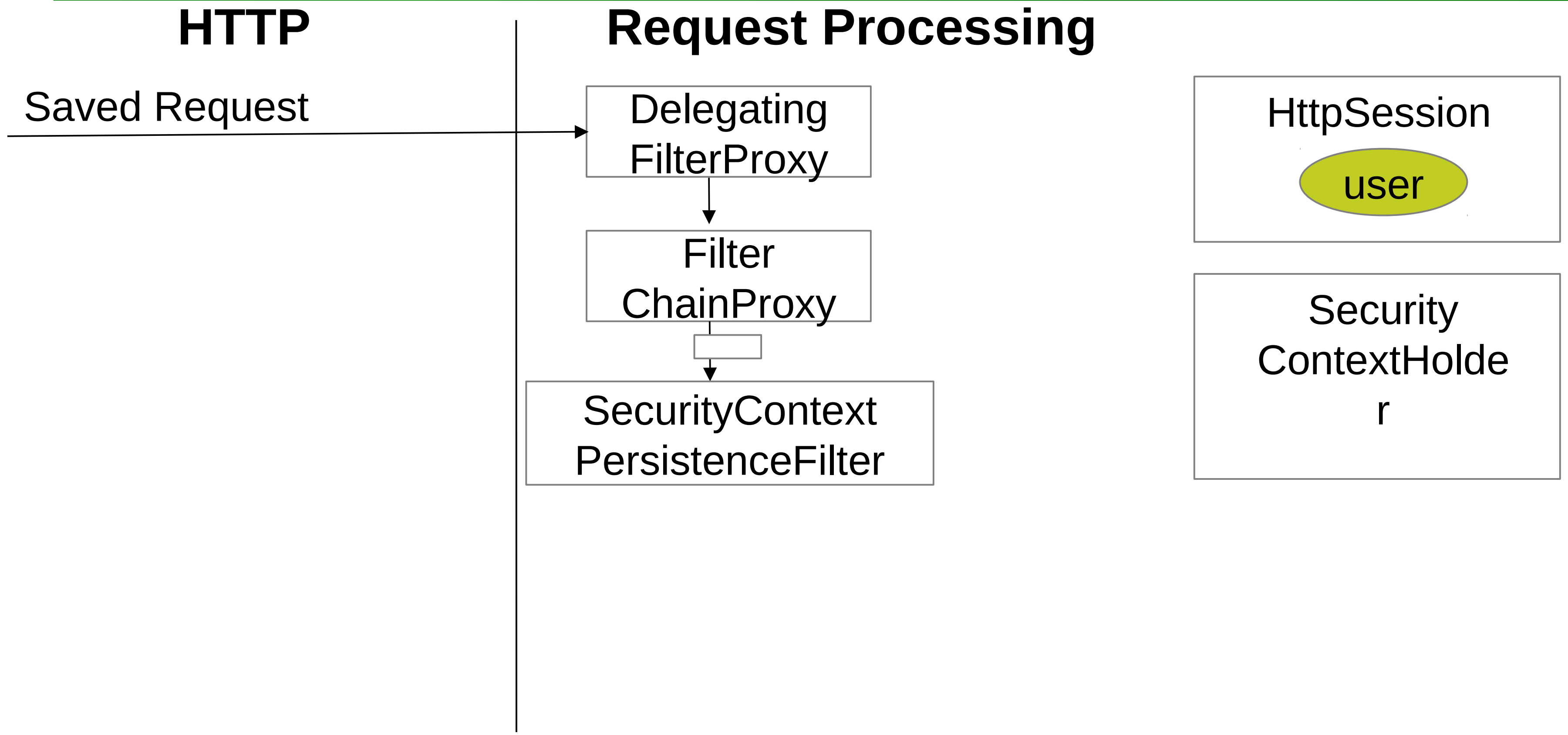
Requesting Protected Resource while Authenticated



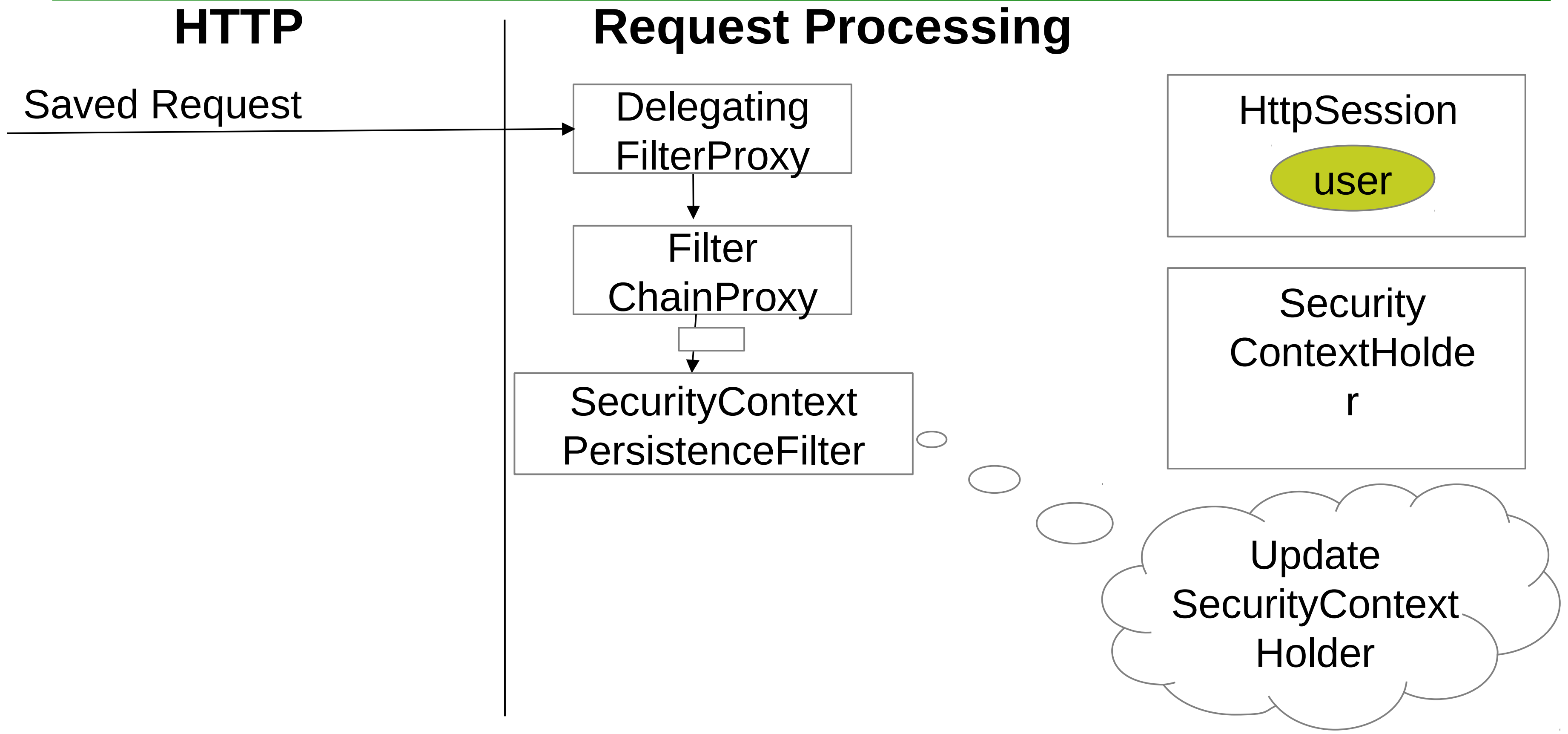
Requesting Protected Resource while Authenticated



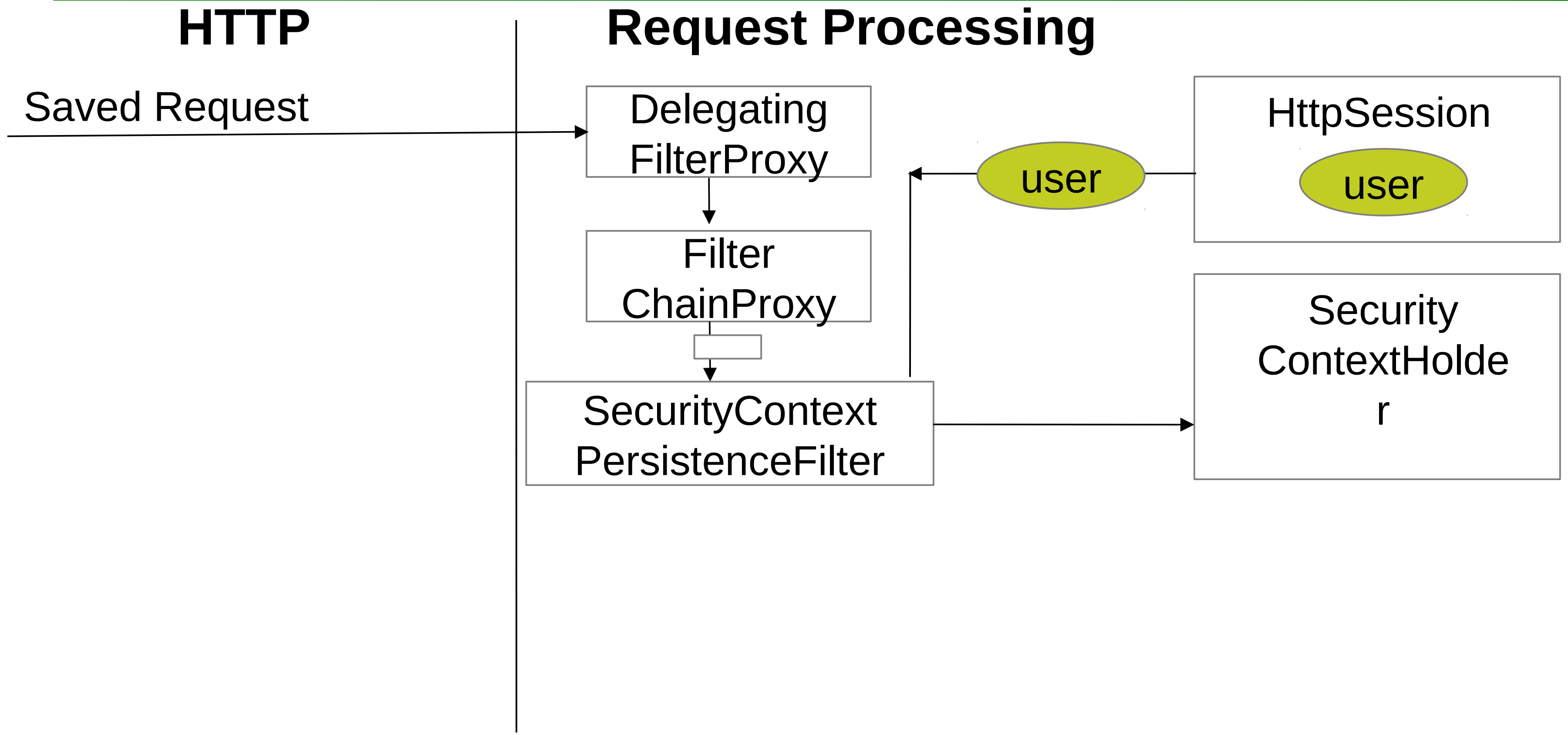
Requesting Protected Resource while Authenticated



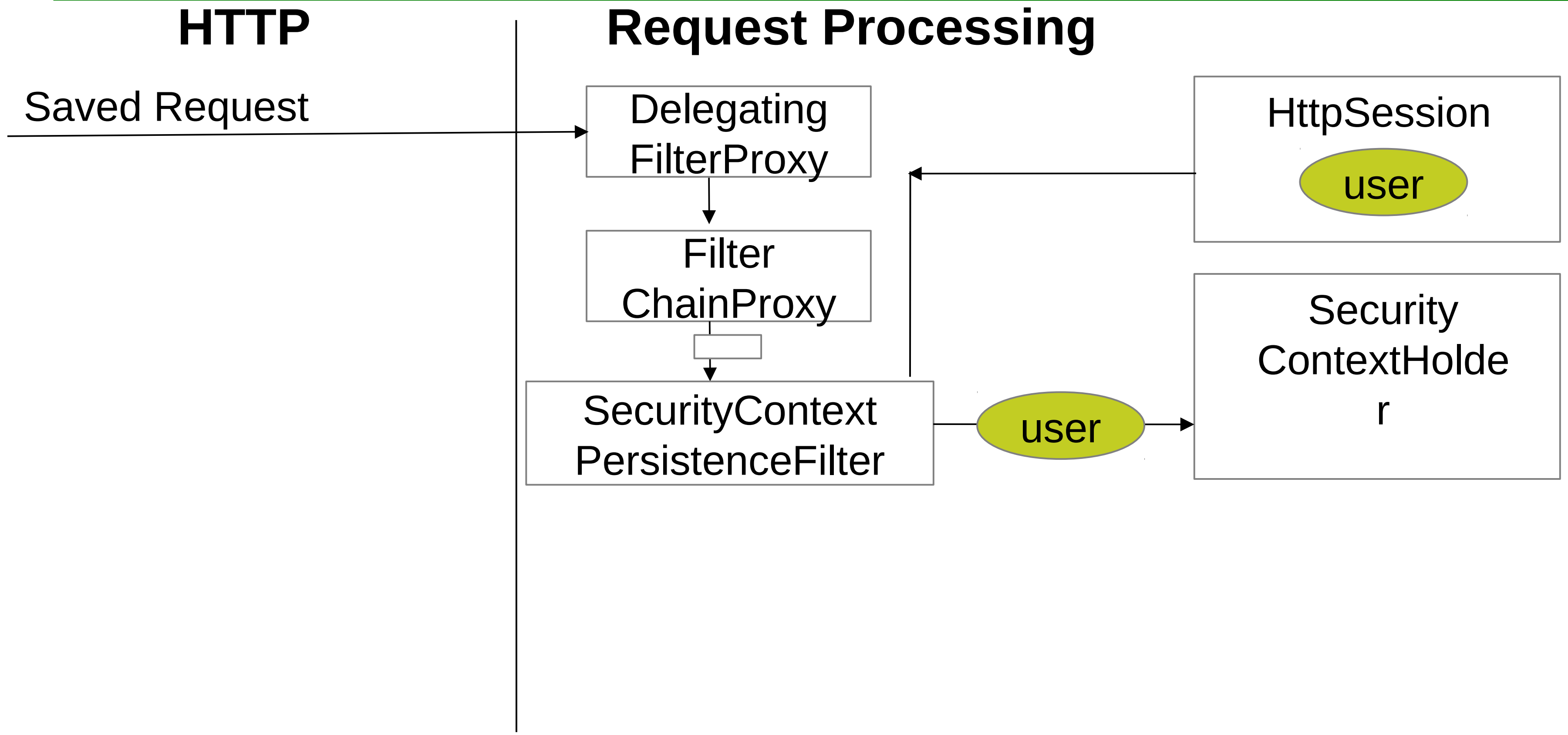
Requesting Protected Resource while Authenticated



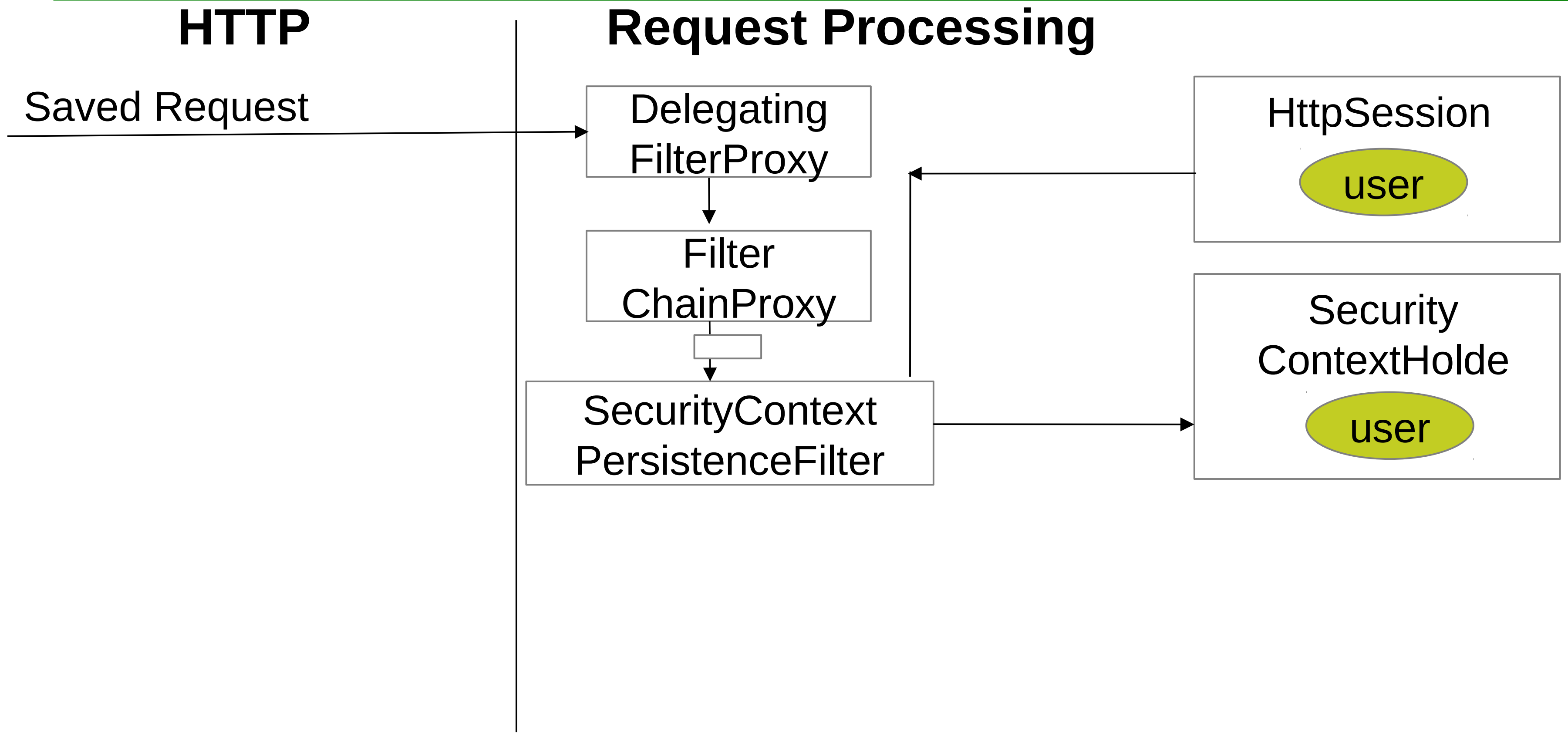
Requesting Protected Resource while Authenticated



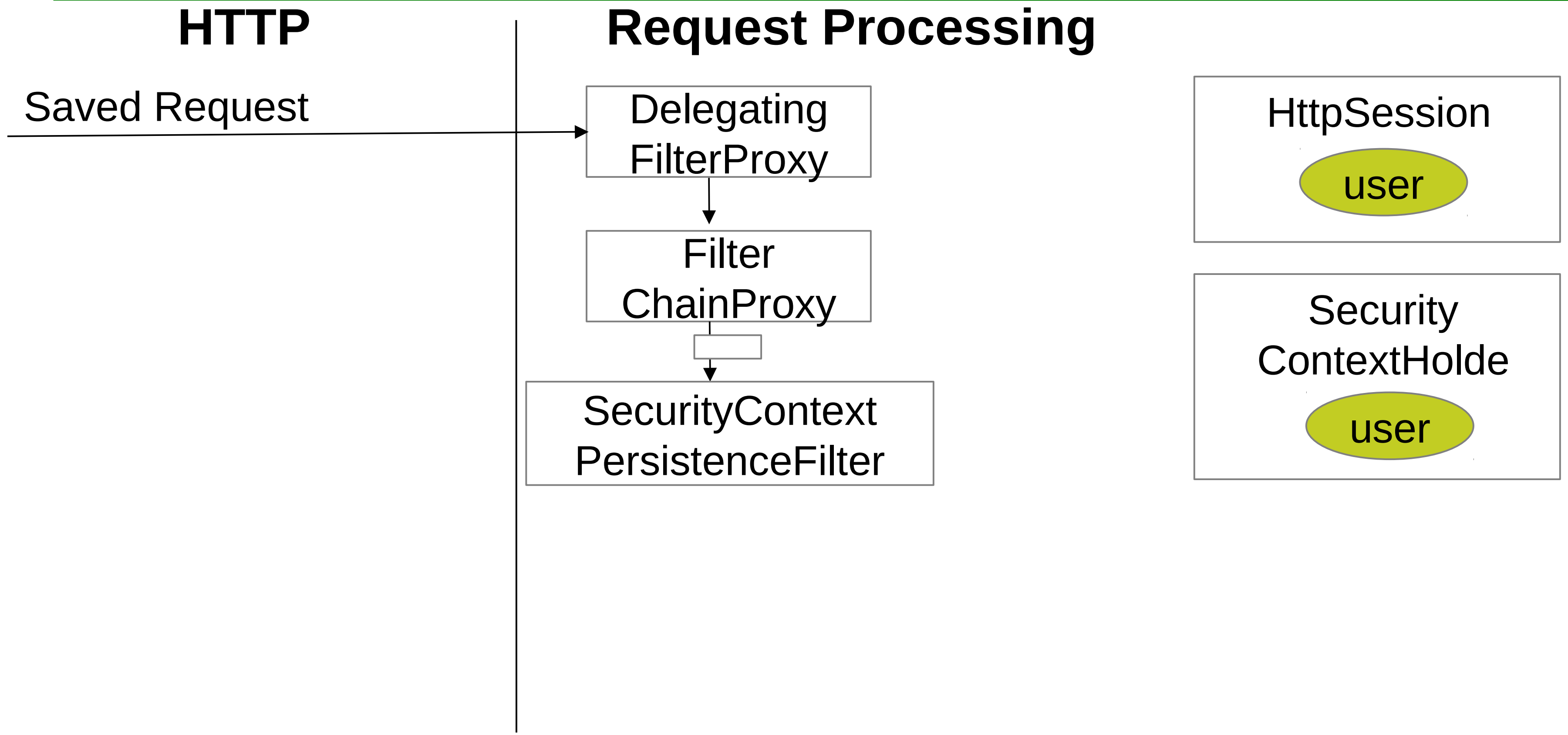
Requesting Protected Resource while Authenticated



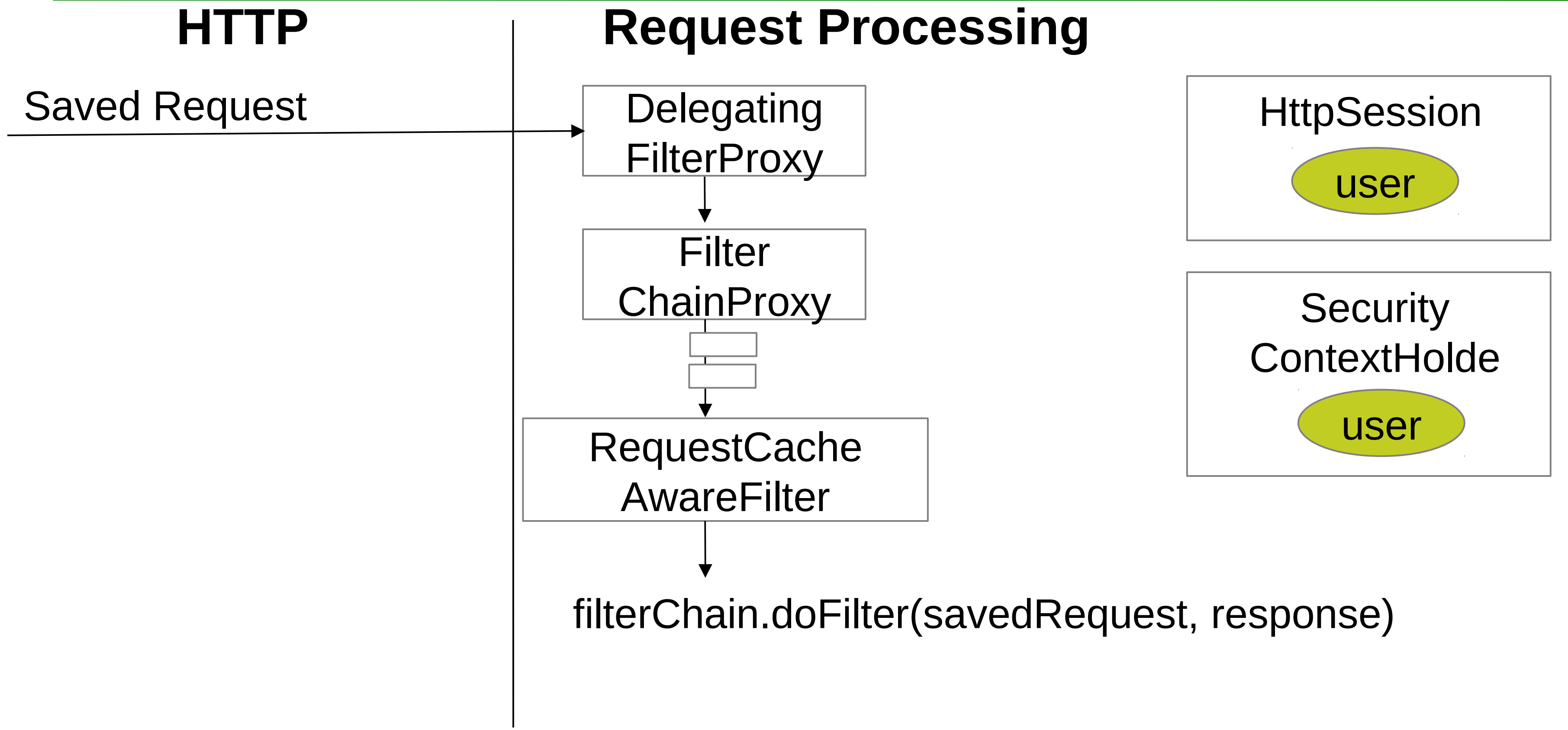
Requesting Protected Resource while Authenticated



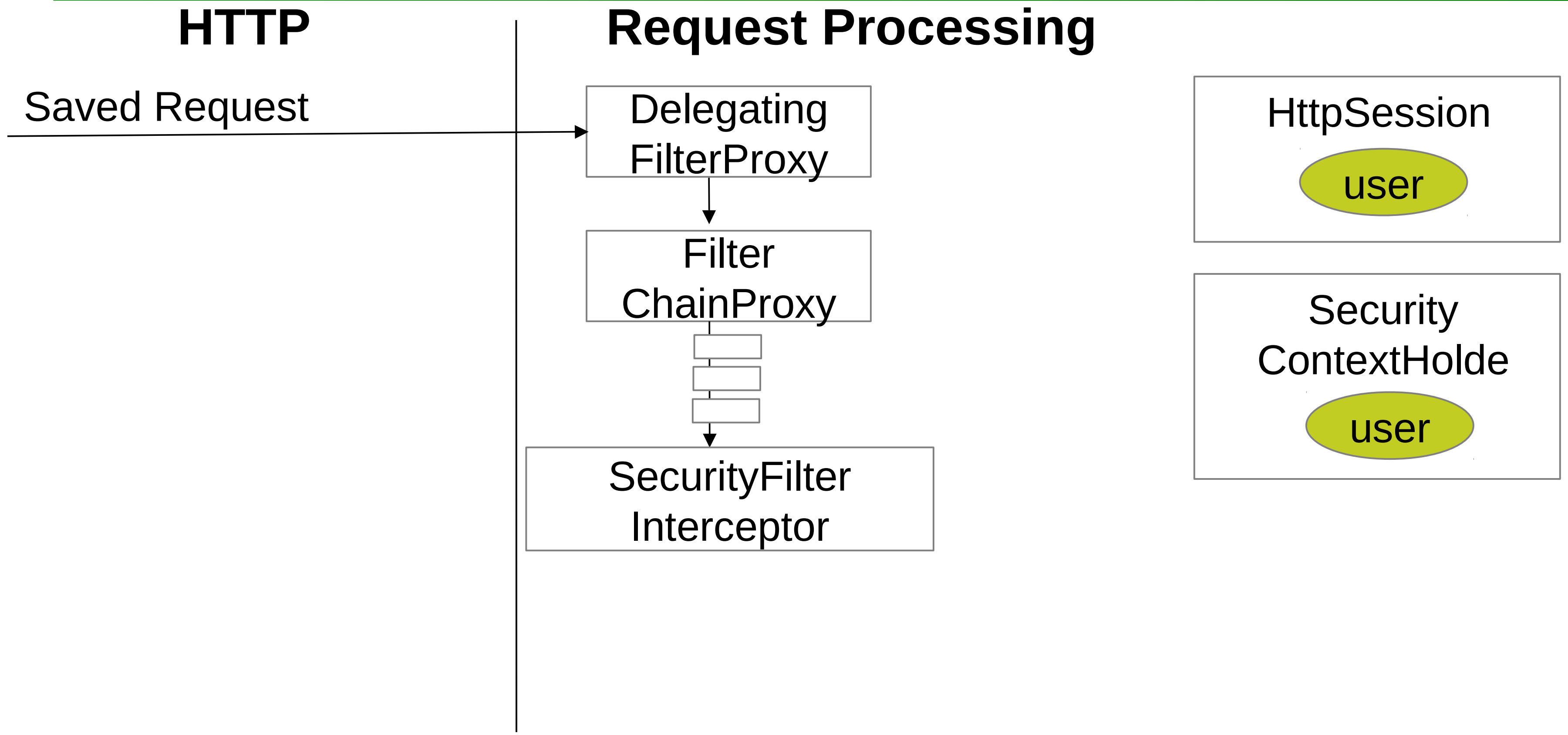
Requesting Protected Resource while Authenticated



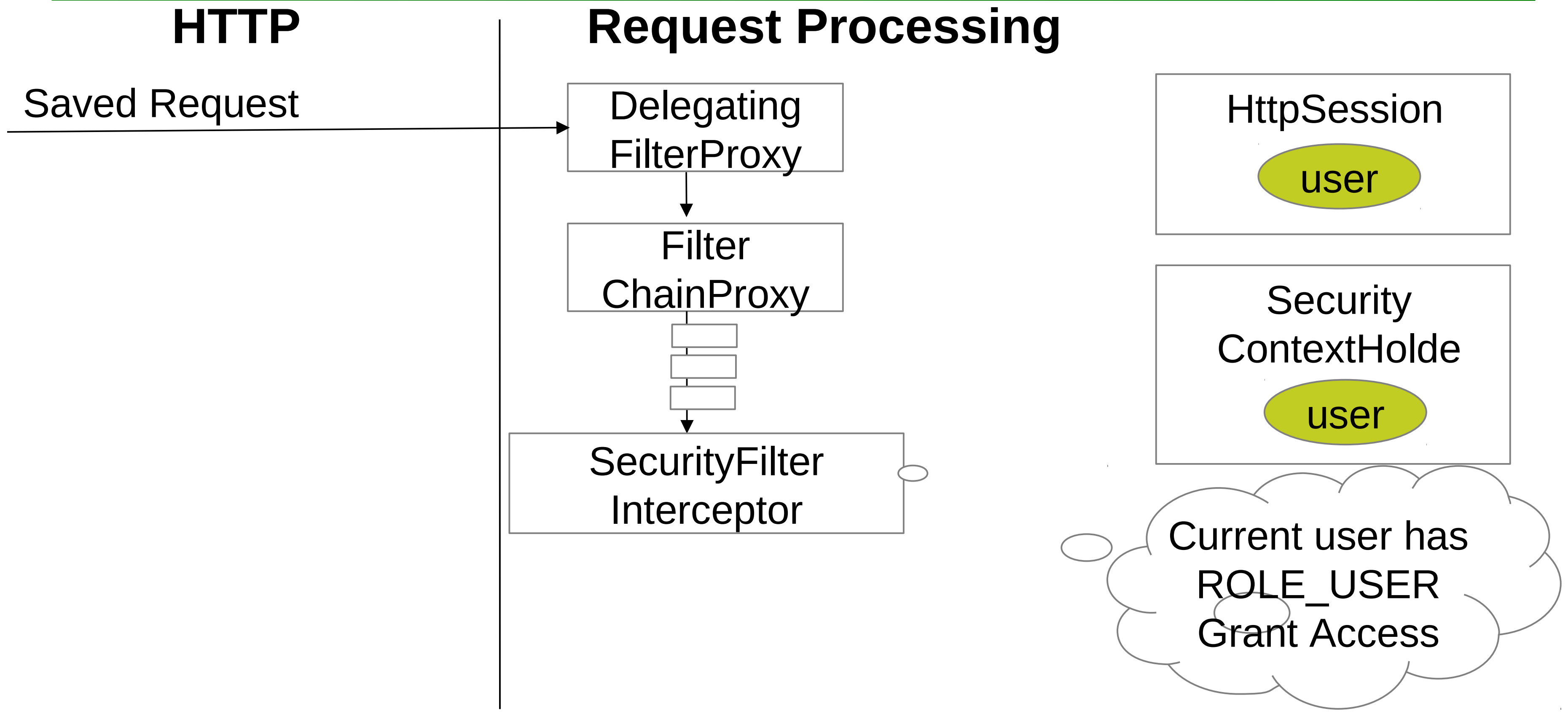
Requesting Protected Resource while Authenticated



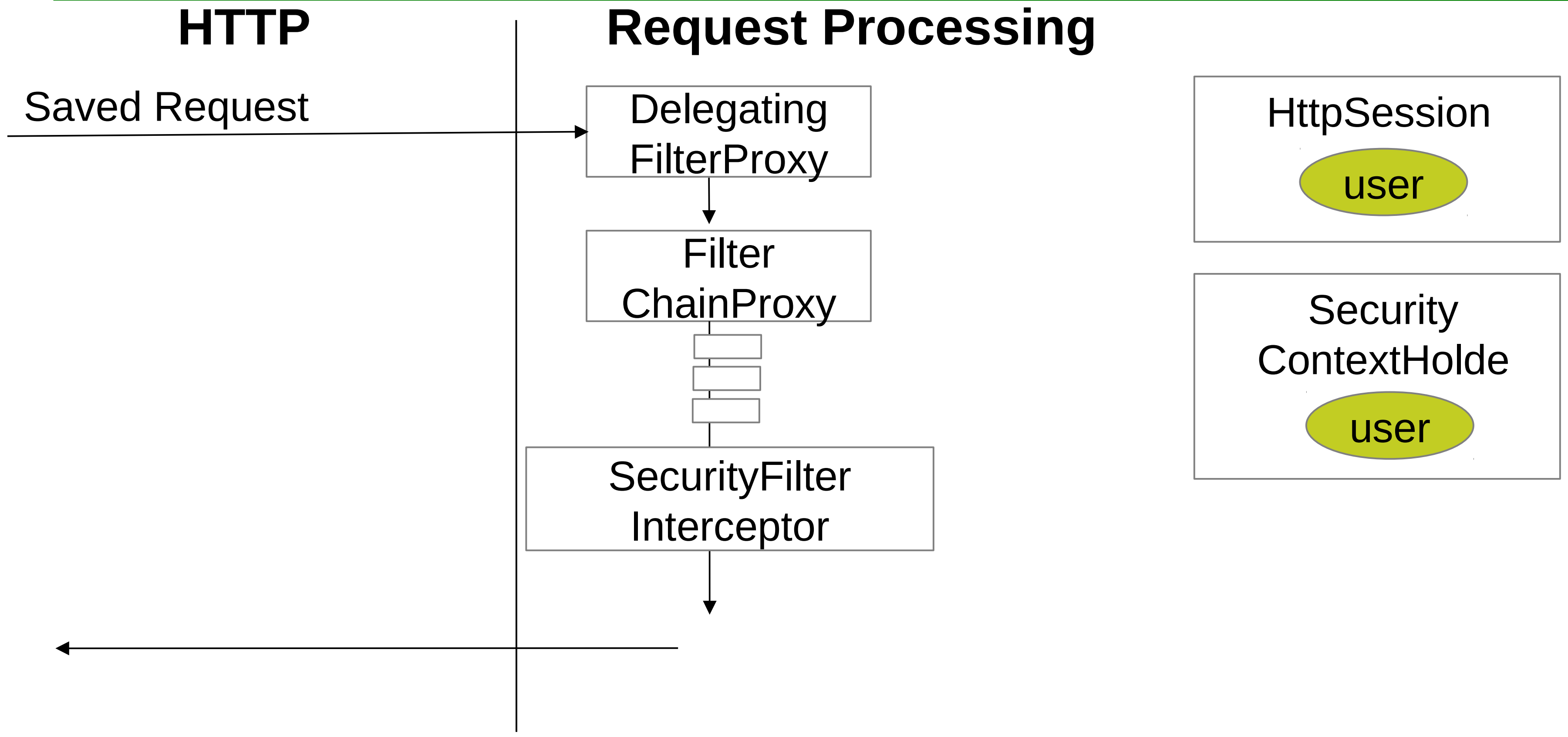
Requesting Protected Resource while Authenticated



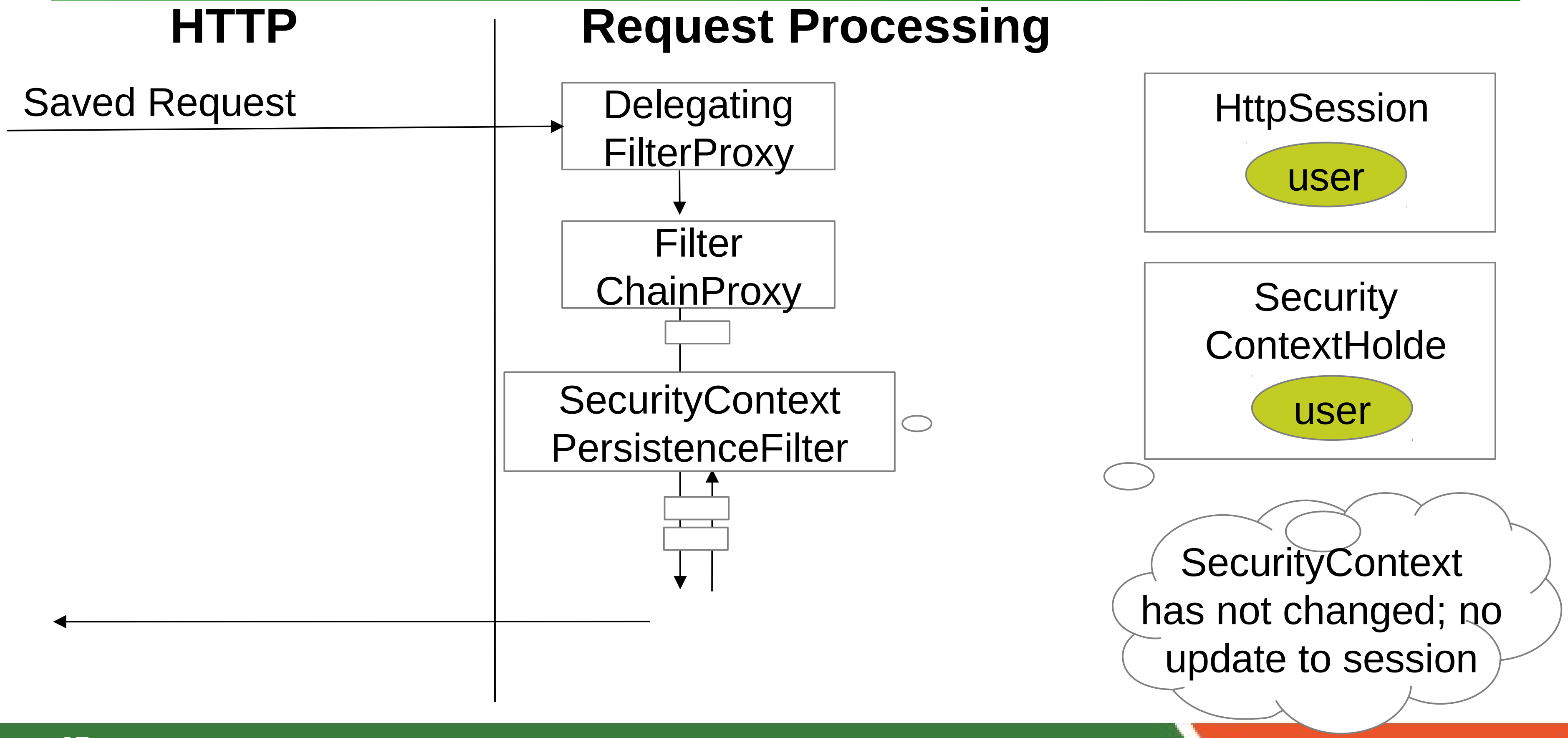
Requesting Protected Resource while Authenticated



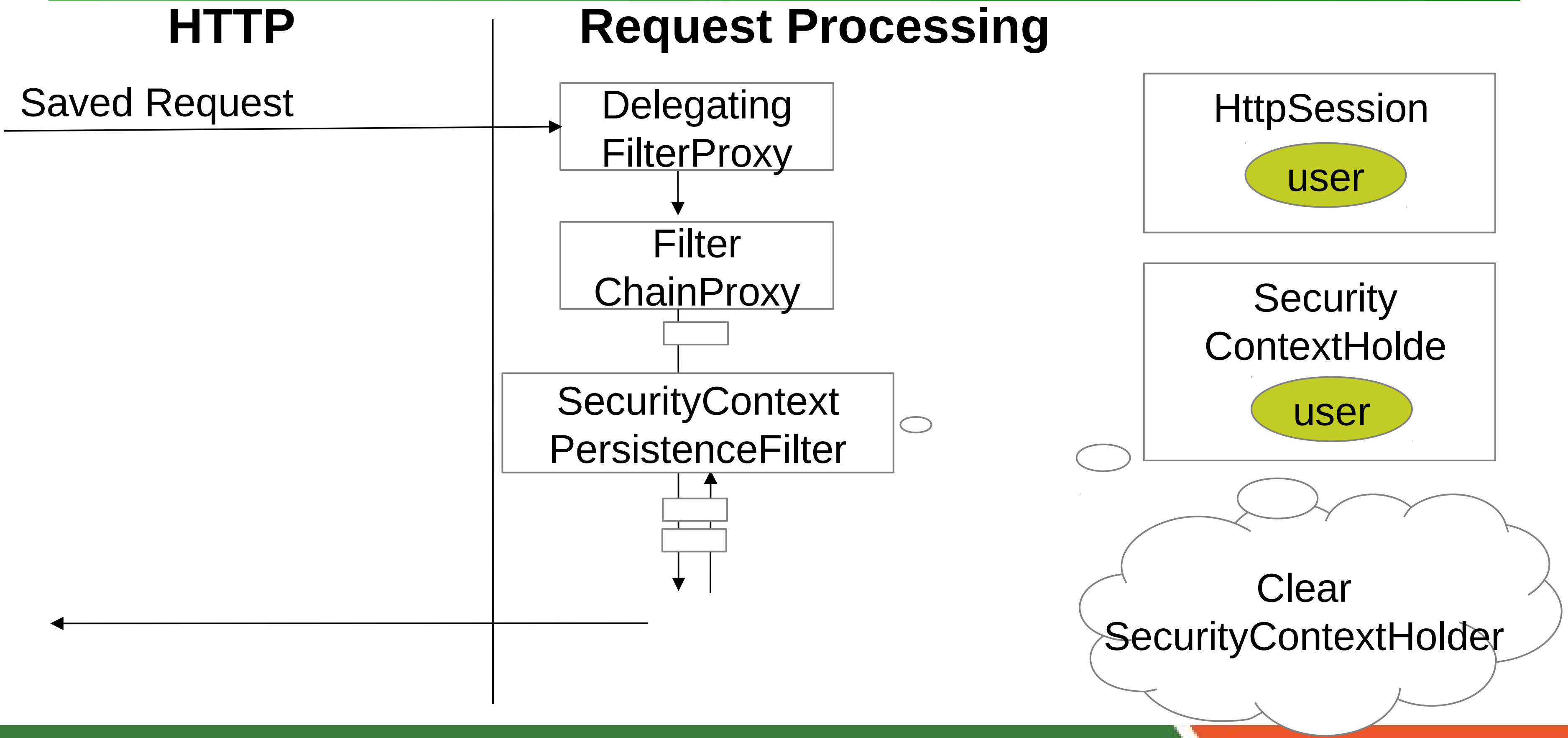
Requesting Protected Resource while Authenticated



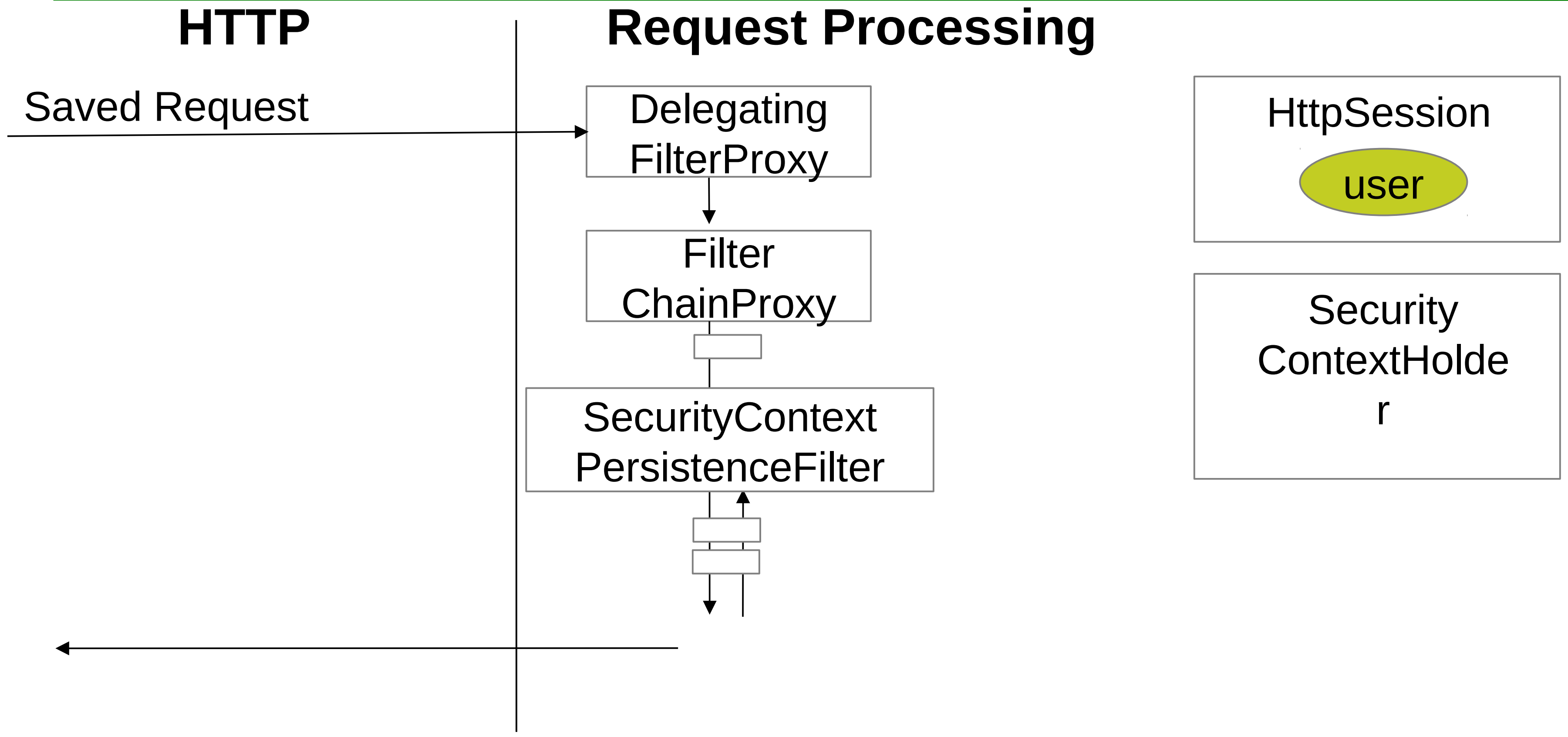
Requesting Protected Resource while Authenticated



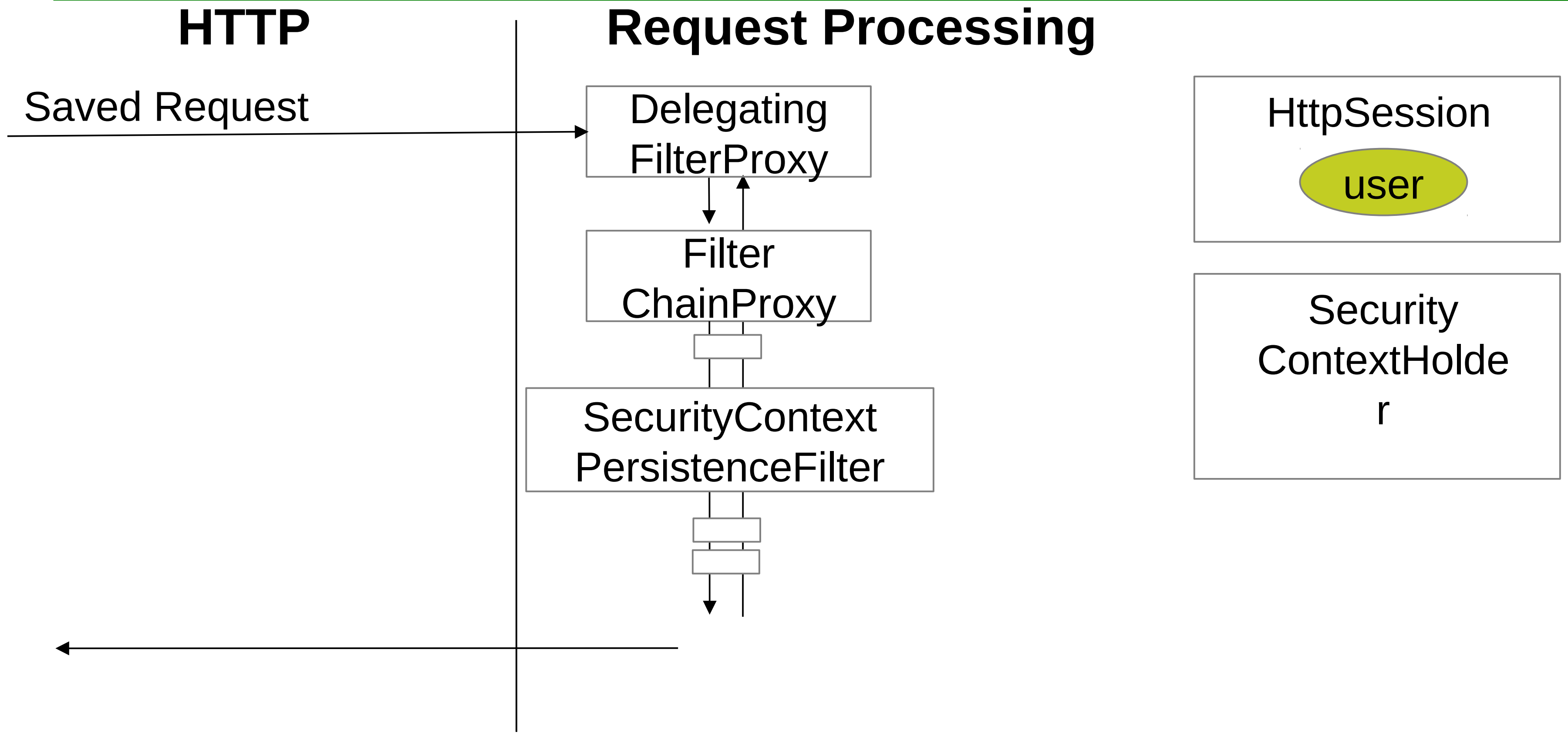
Requesting Protected Resource while Authenticated



Requesting Protected Resource while Authenticated



Requesting Protected Resource while Authenticated



Spring Security Filters

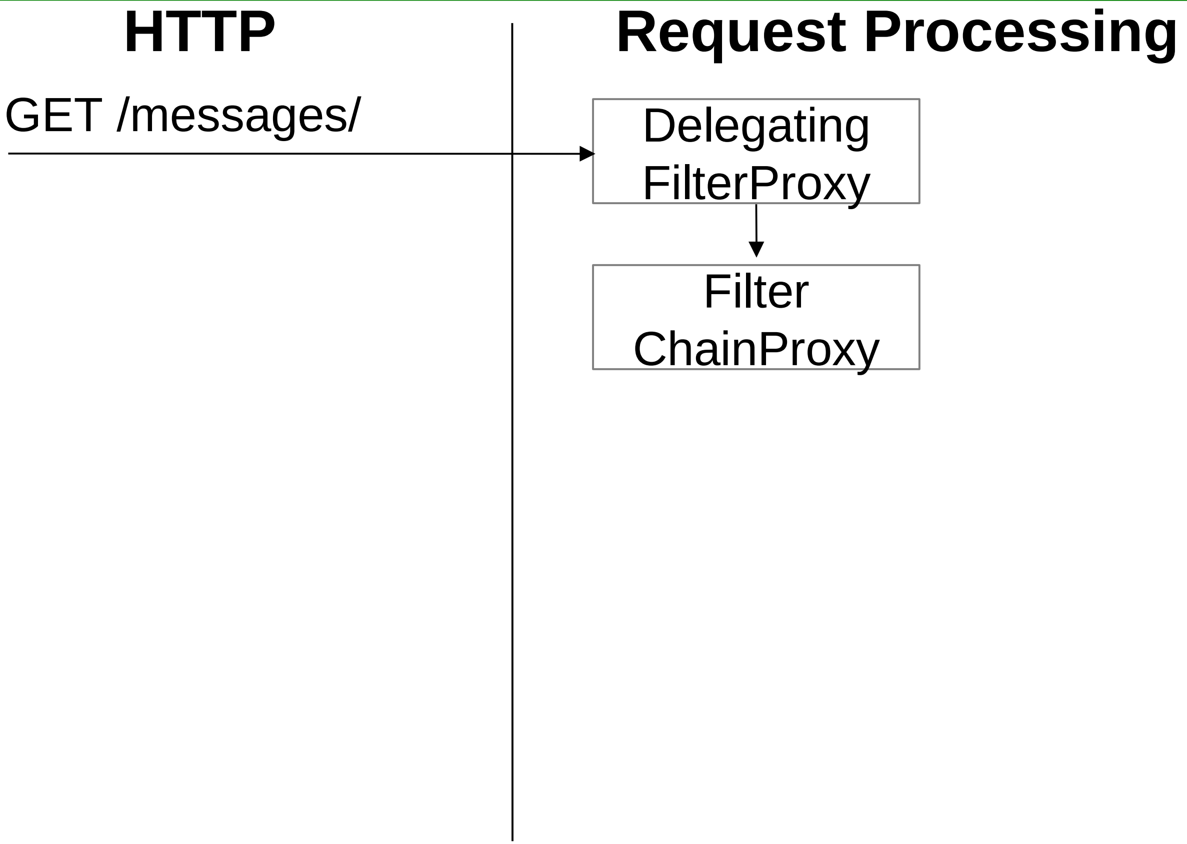
- Each Filter has a specific task
- Each Filter acts as a controller
- Logic in Filter's can be implemented in a controller of the framework of your choice

Custom Log In Page

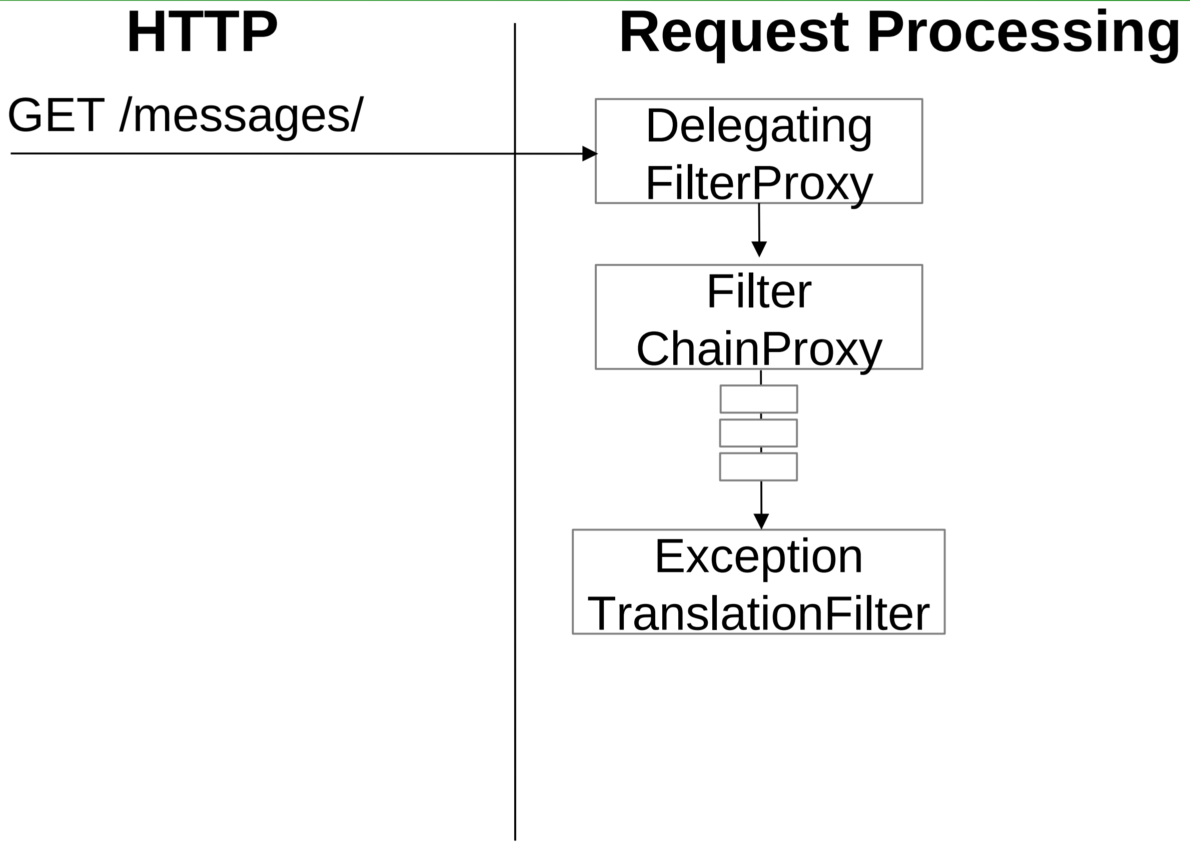
src/main/webapp/WEB-INF/security.xml

```
<security:http use-expressions="true">
  <security:intercept-url pattern="/**"
      access="hasRole('ROLE_USER')"/>
  <security:form-login login-page="/login"
      authentication-failure-url="/login?error"/>
</security:http>
```

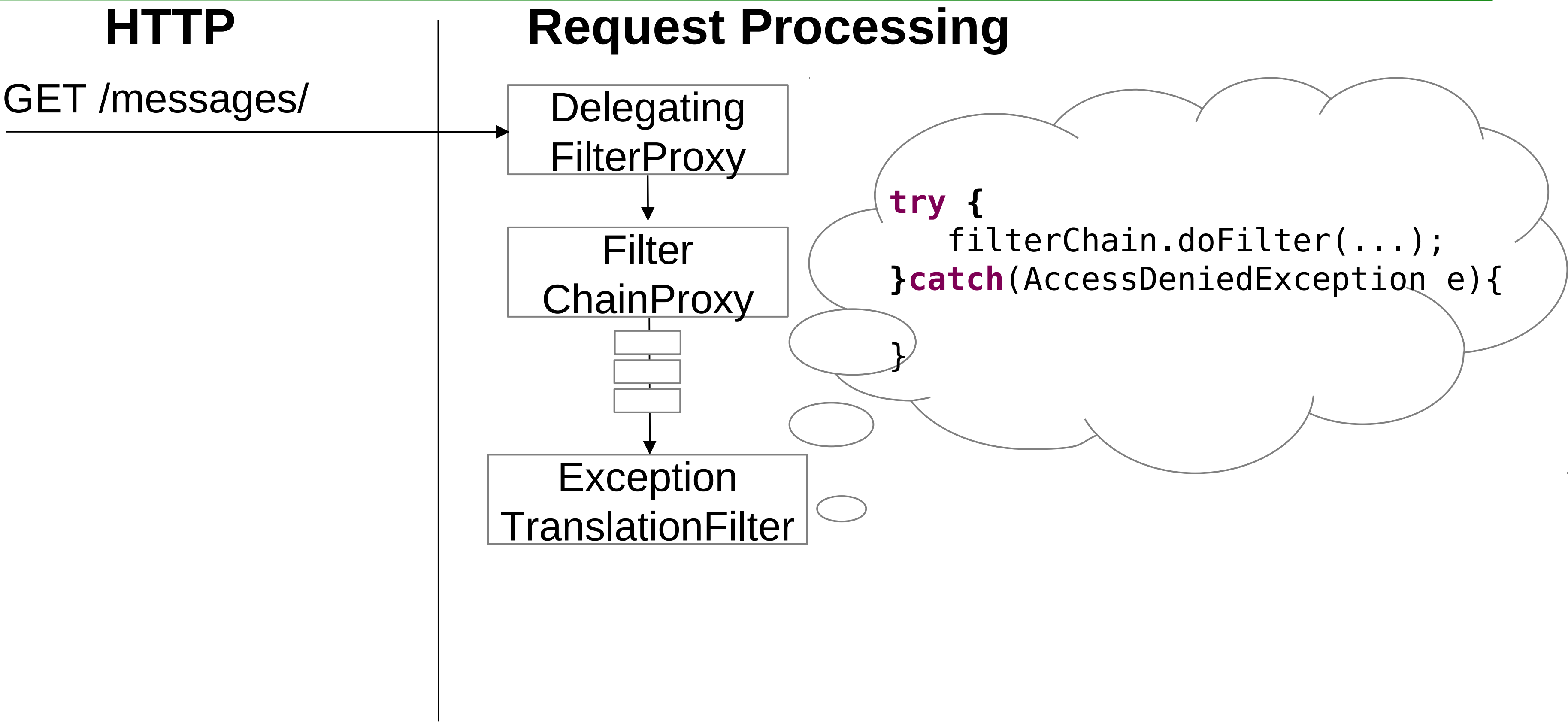
Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource

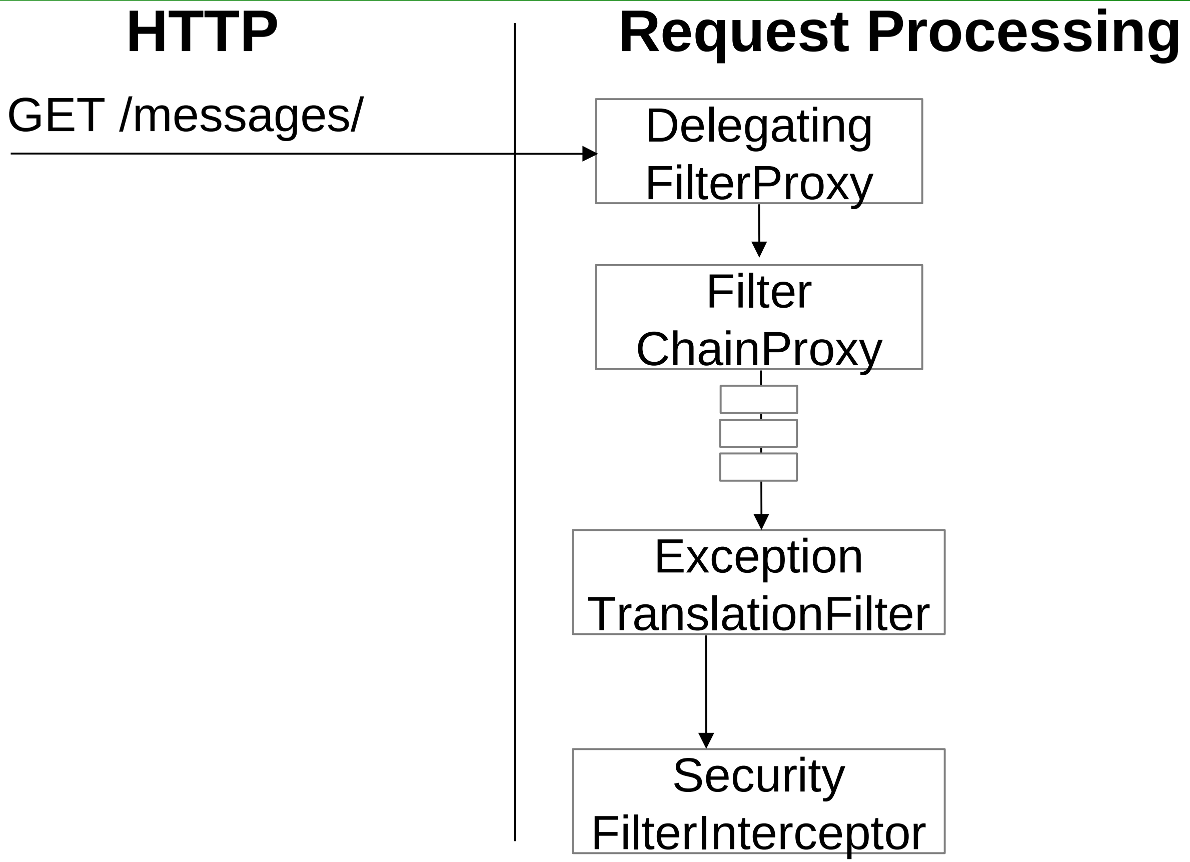


Unauthenticated Request to Protected Resource

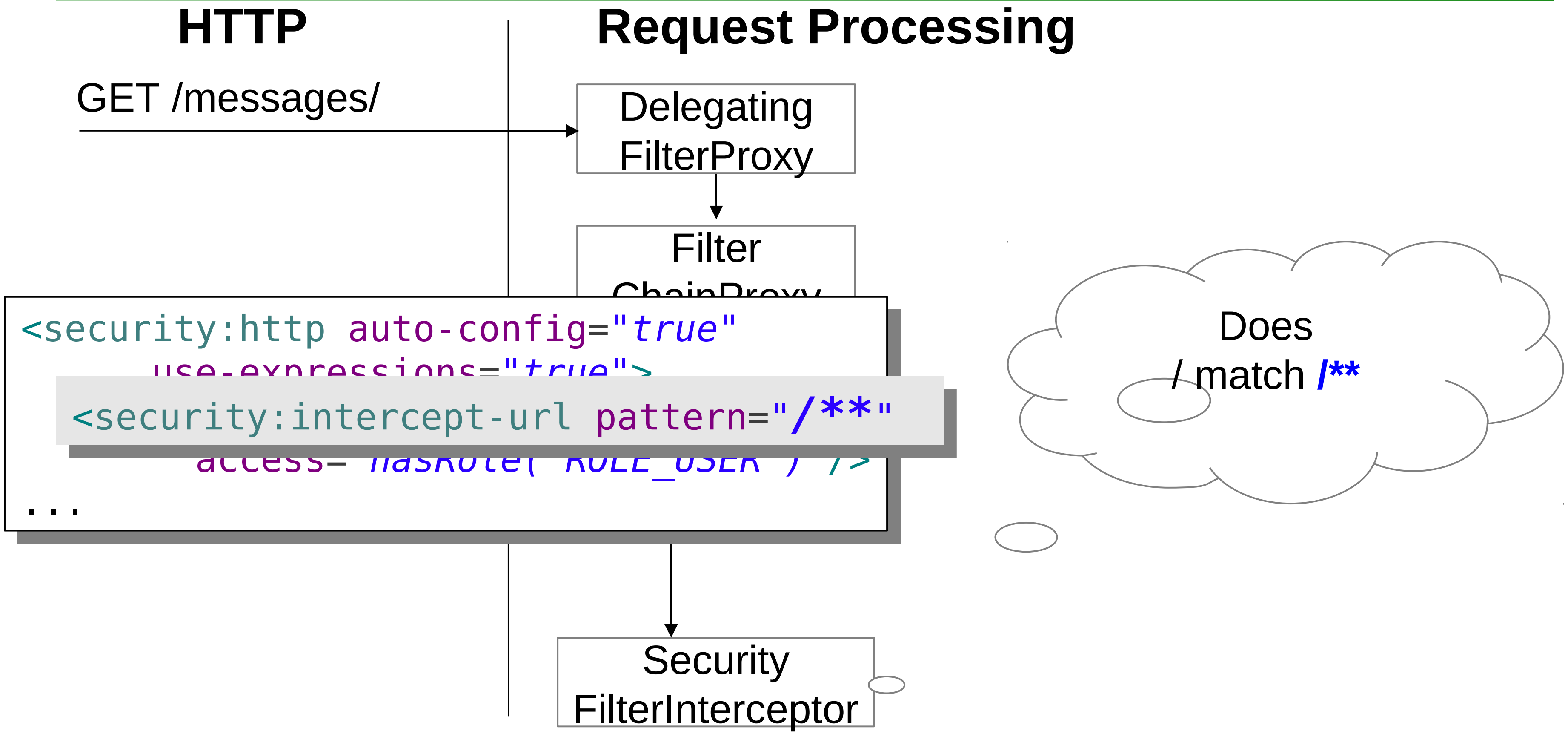


```
try {
   filterChain.doFilter(...);
}catch(AccessDeniedException e){

}
```

Unauthenticated Request to Protected Resource

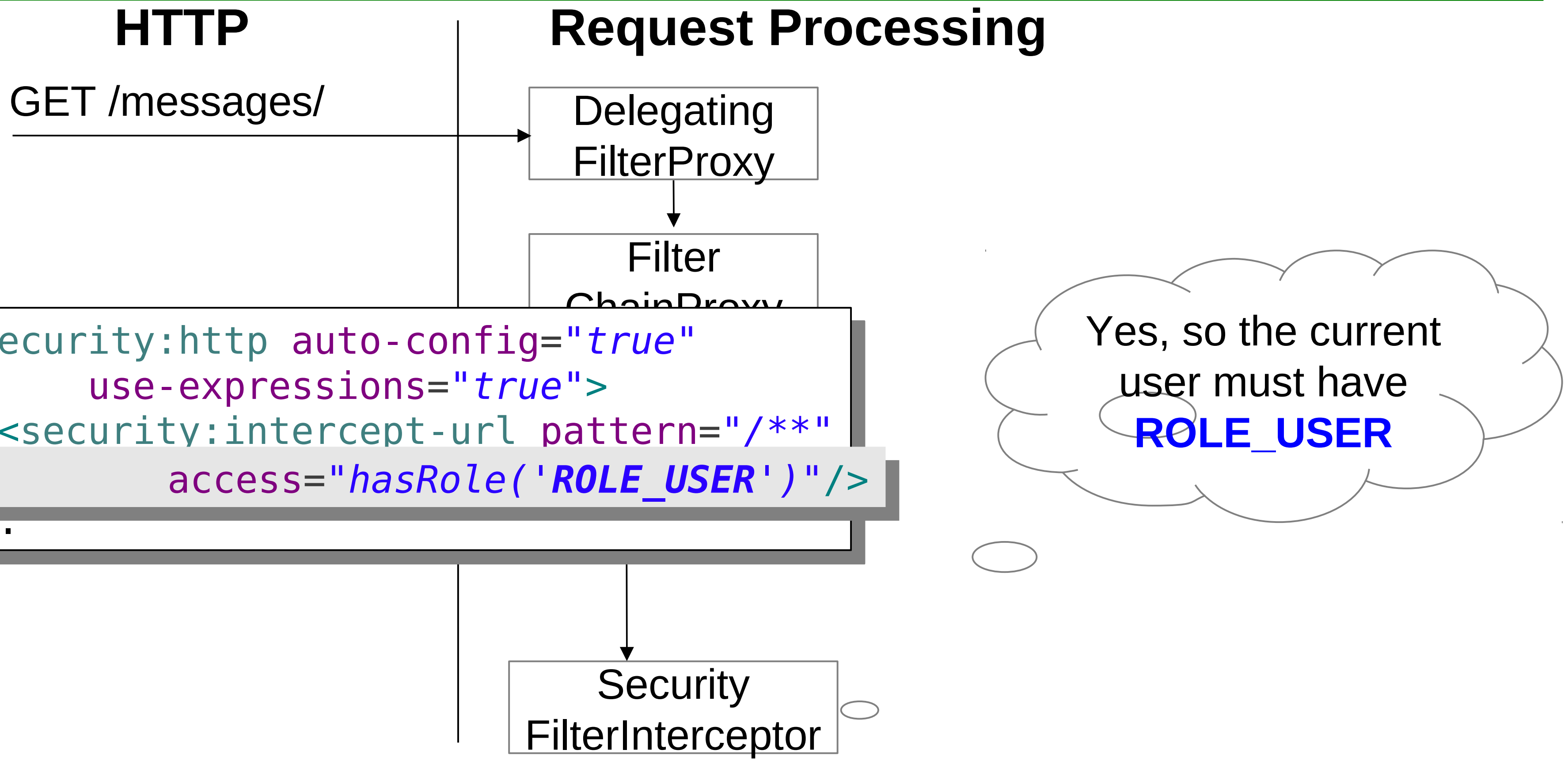


Unauthenticated Request to Protected Resource



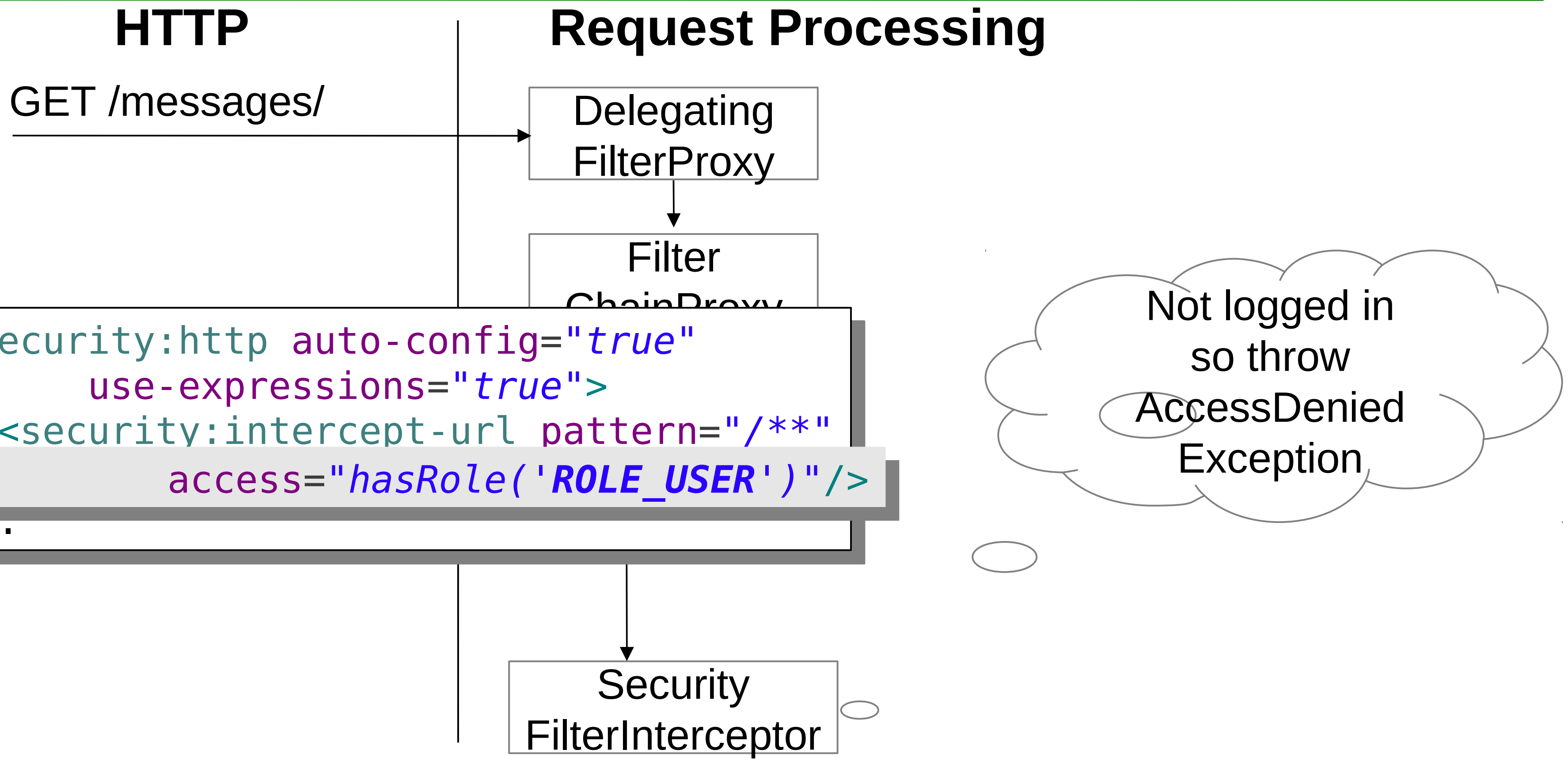
```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')" />
...
```

Unauthenticated Request to Protected Resource



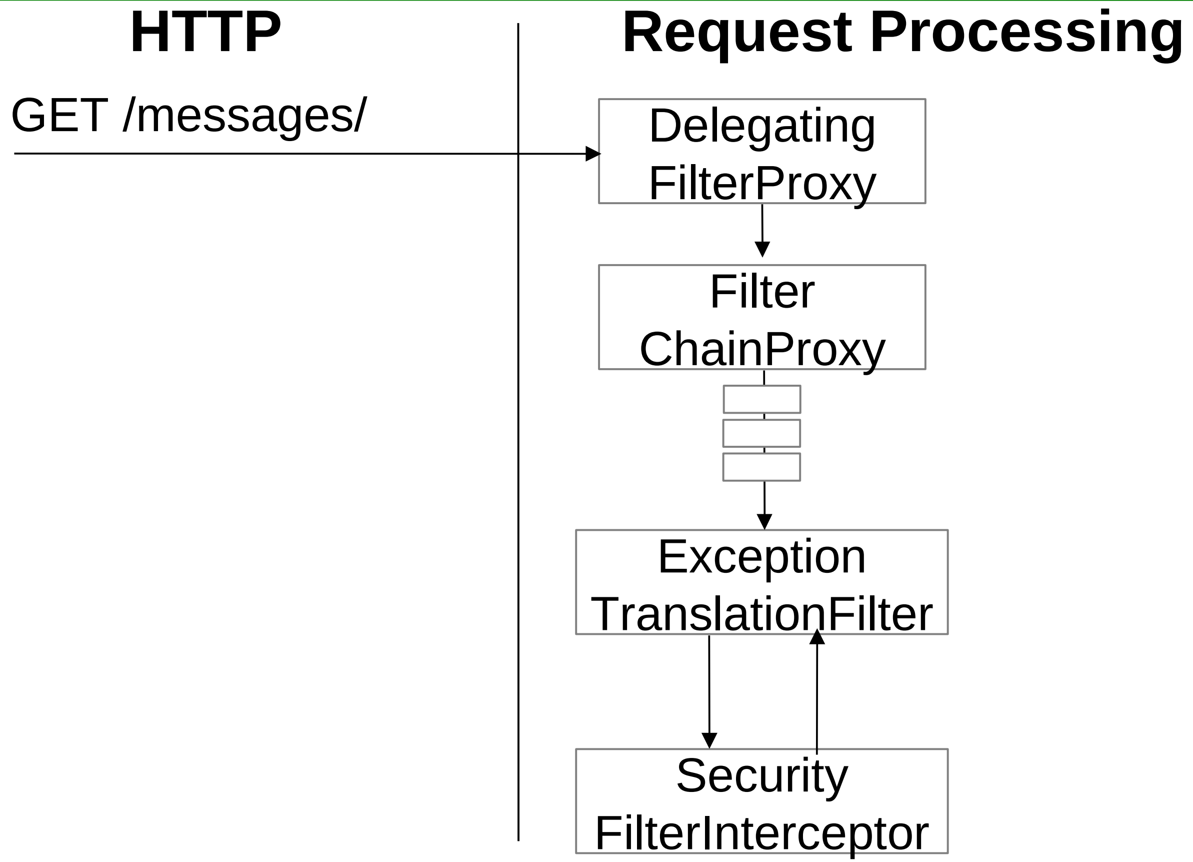
```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
            access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

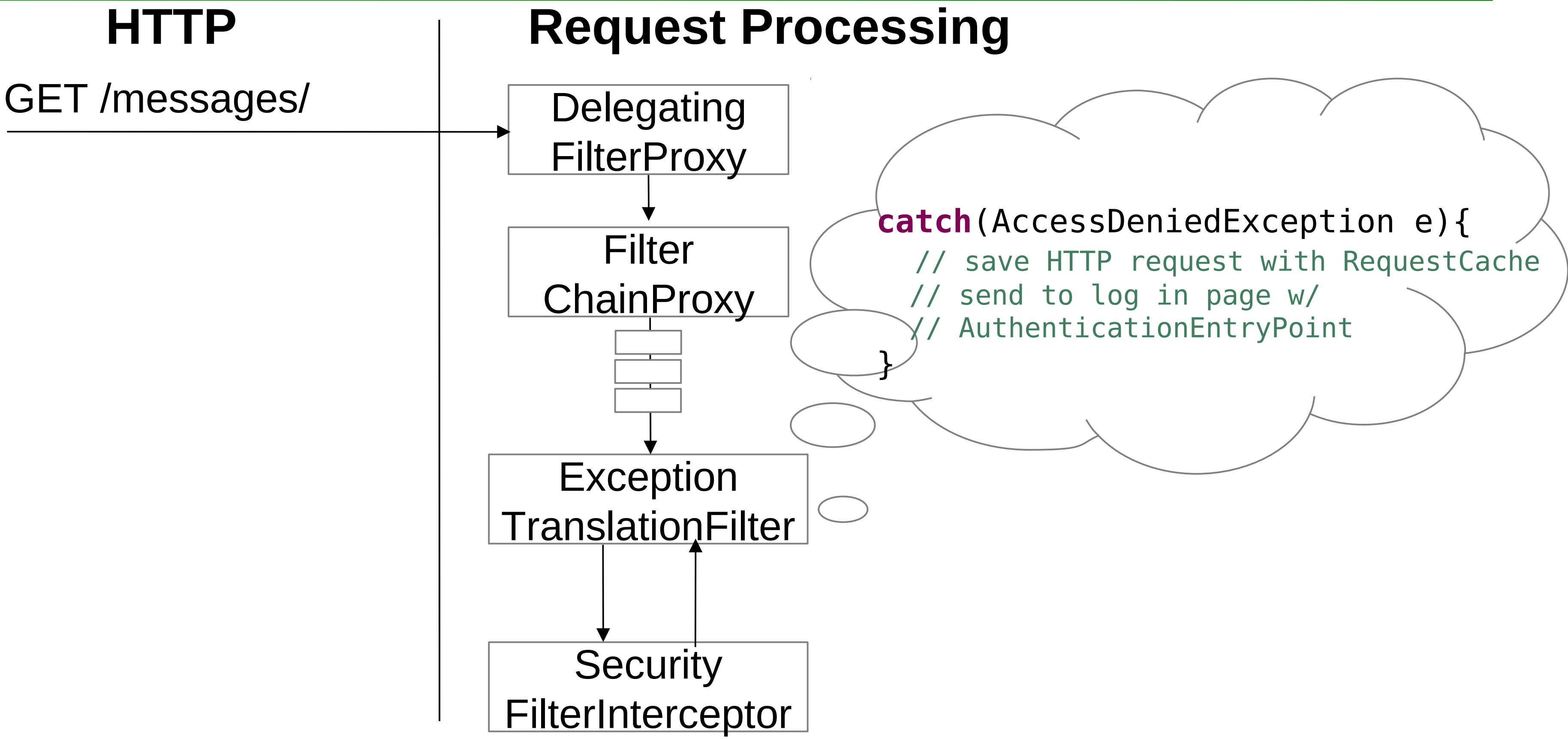


```
<security:http auto-config="true"
   use-expressions="true">
 <security:intercept-url pattern="/**"
     access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

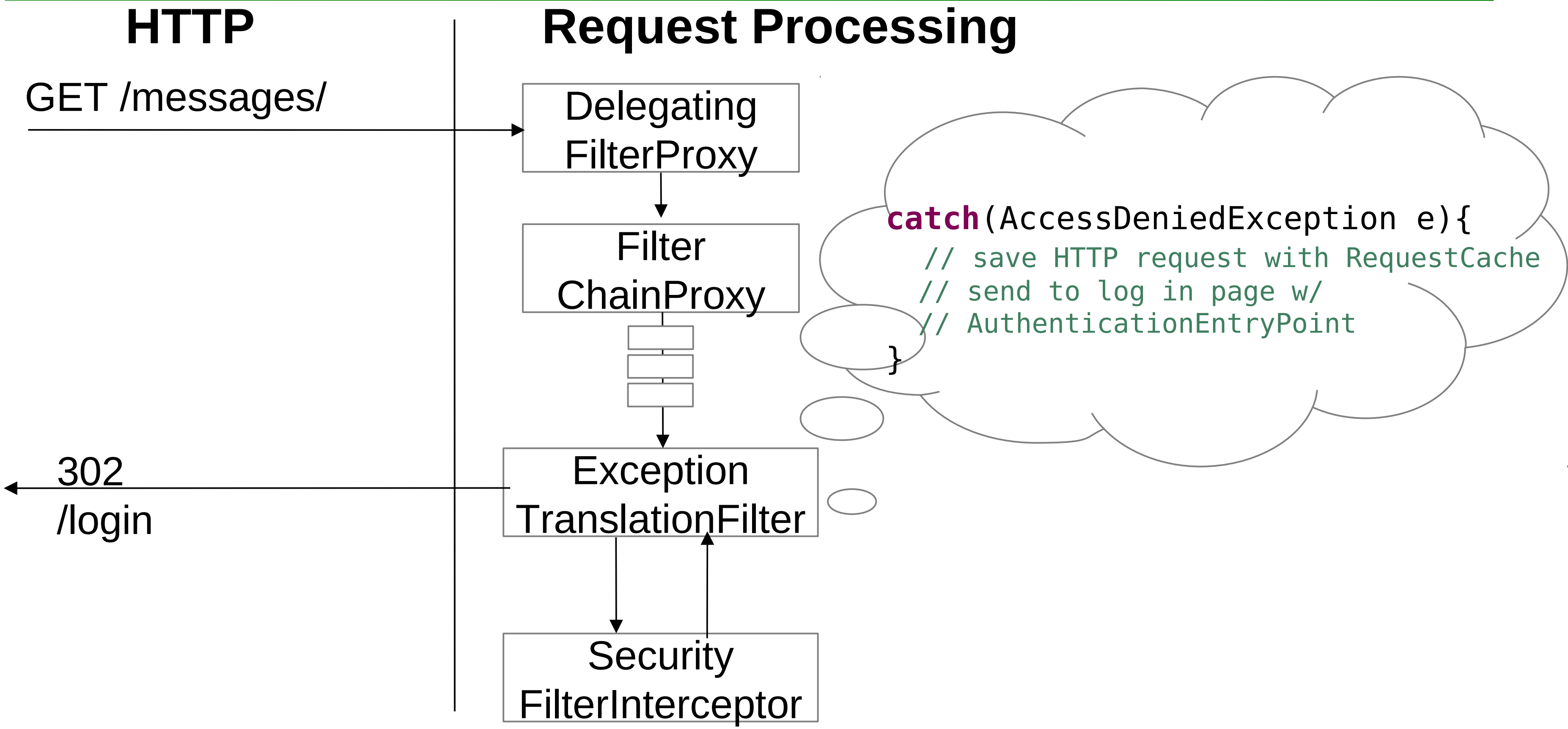


Unauthenticated Request to Protected Resource



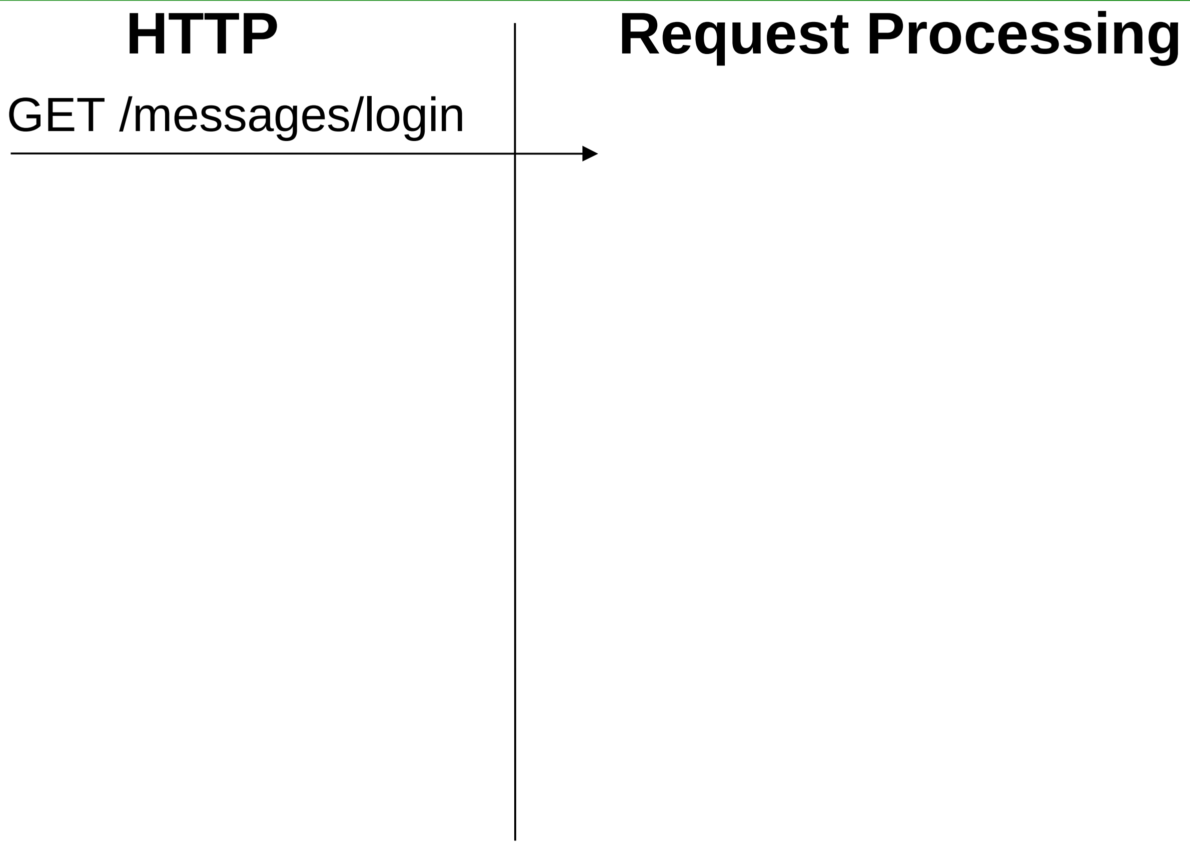
```
catch(AccessDeniedException e){
  // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

Unauthenticated Request to Protected Resource

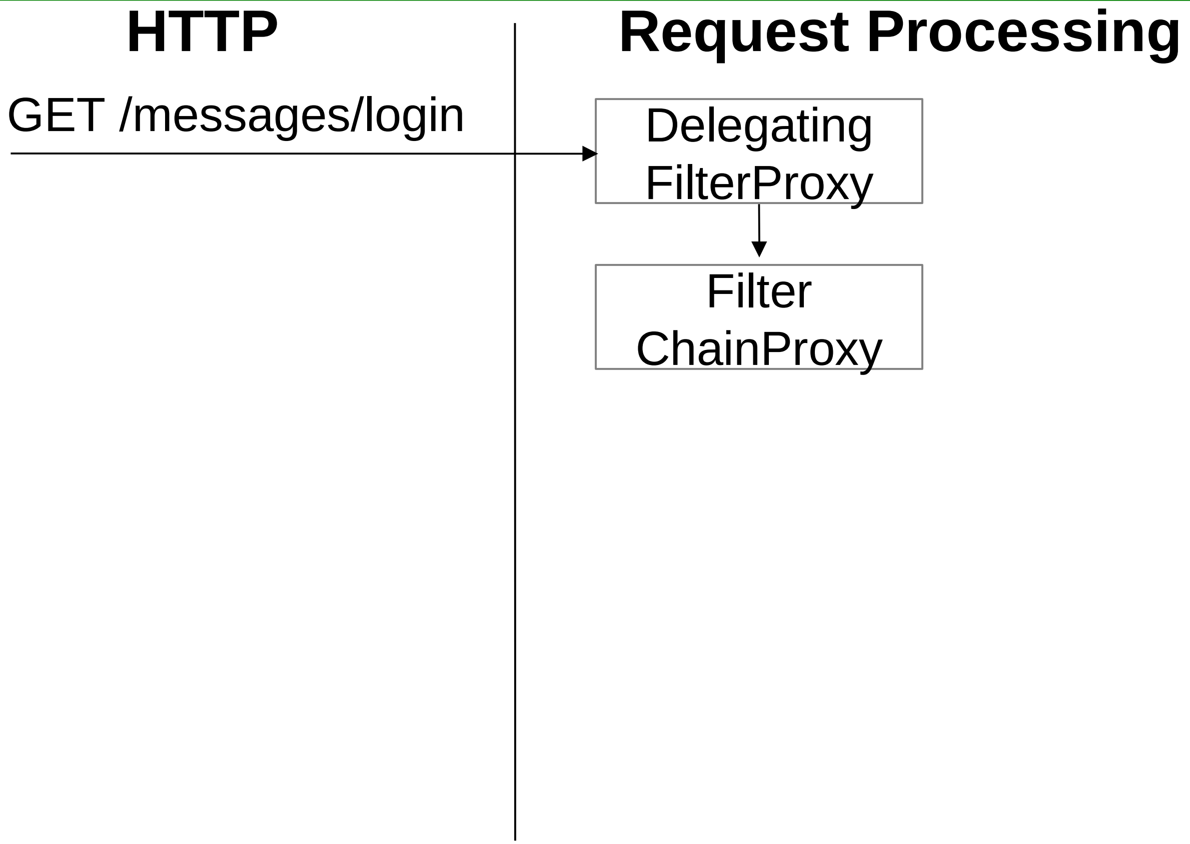


```
catch(AccessDeniedException e){
  // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

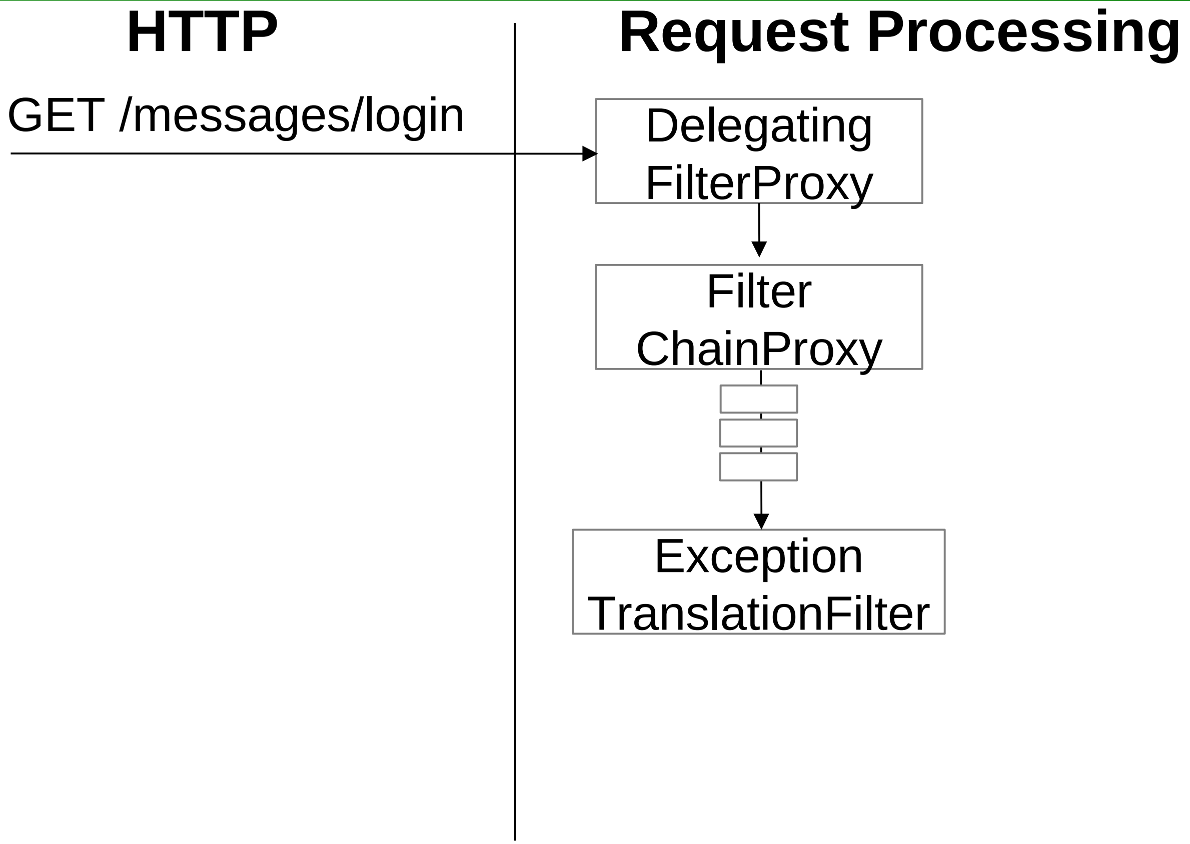
Unauthenticated Request to Protected Resource



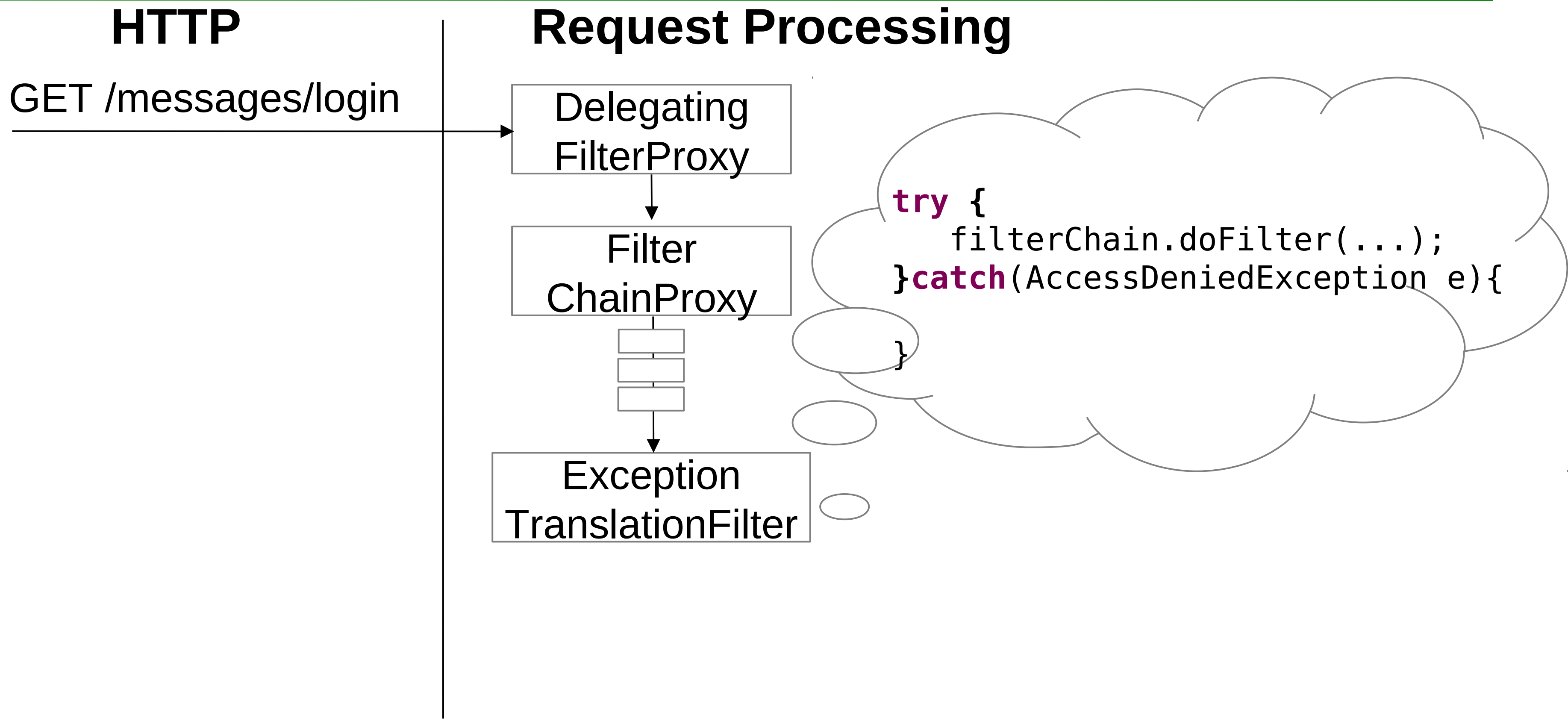
Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource

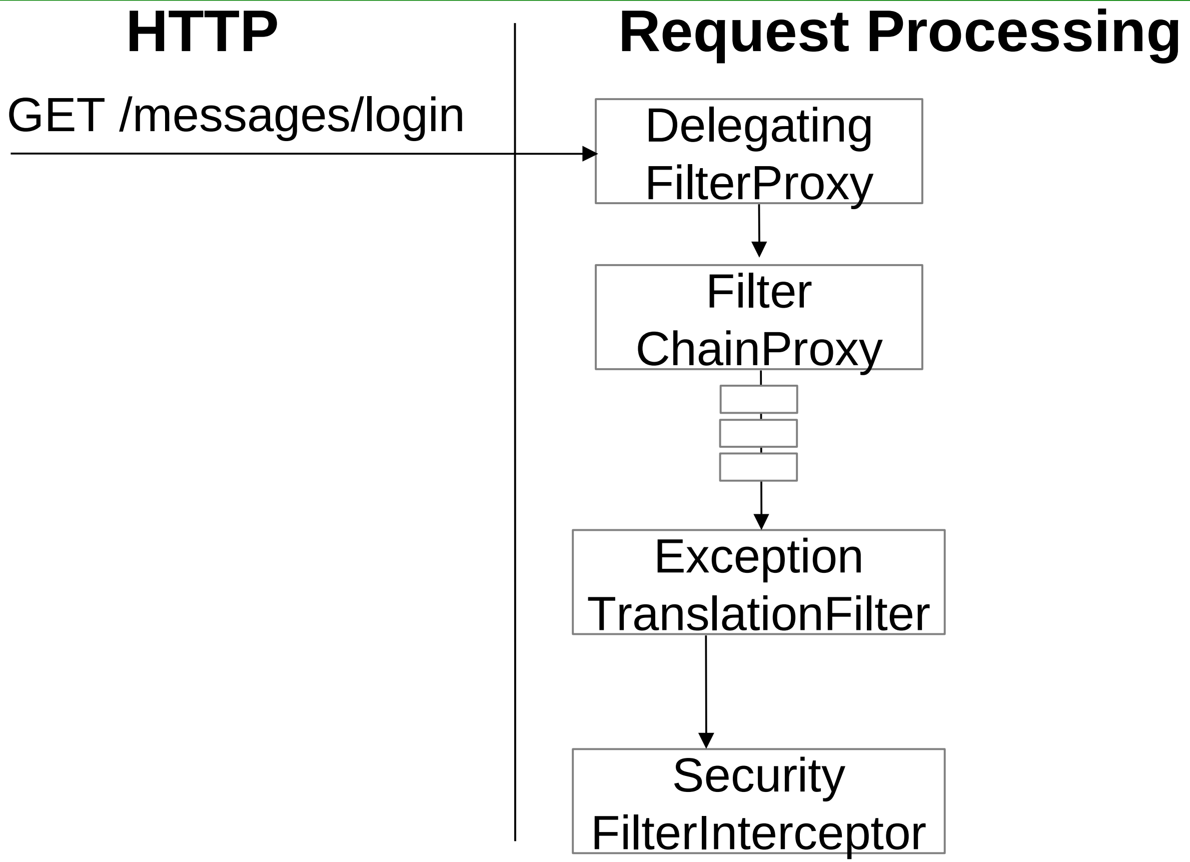


Unauthenticated Request to Protected Resource

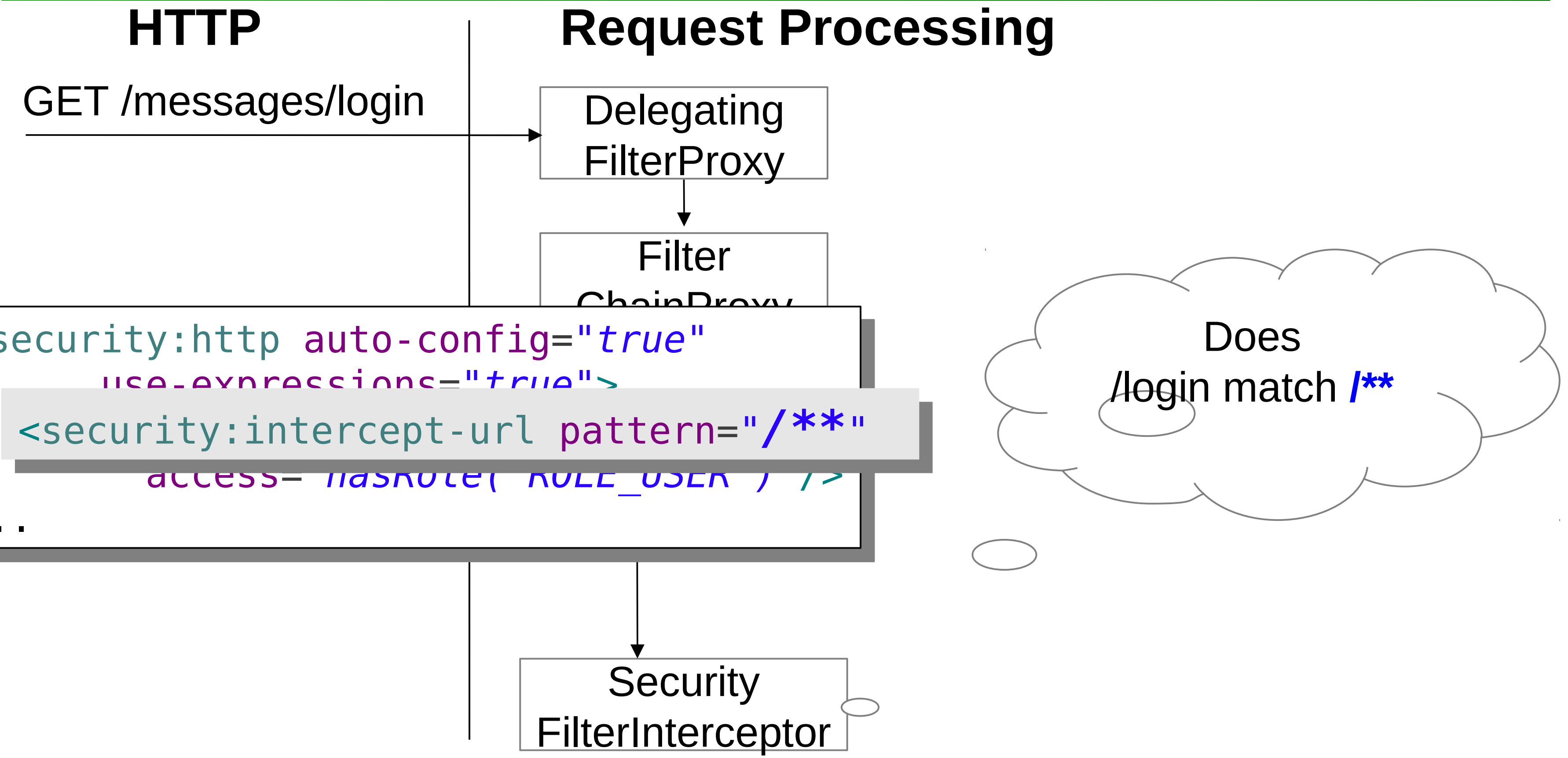


```
try {
   filterChain.doFilter(...);
}catch(AccessDeniedException e){

}
```

Unauthenticated Request to Protected Resource

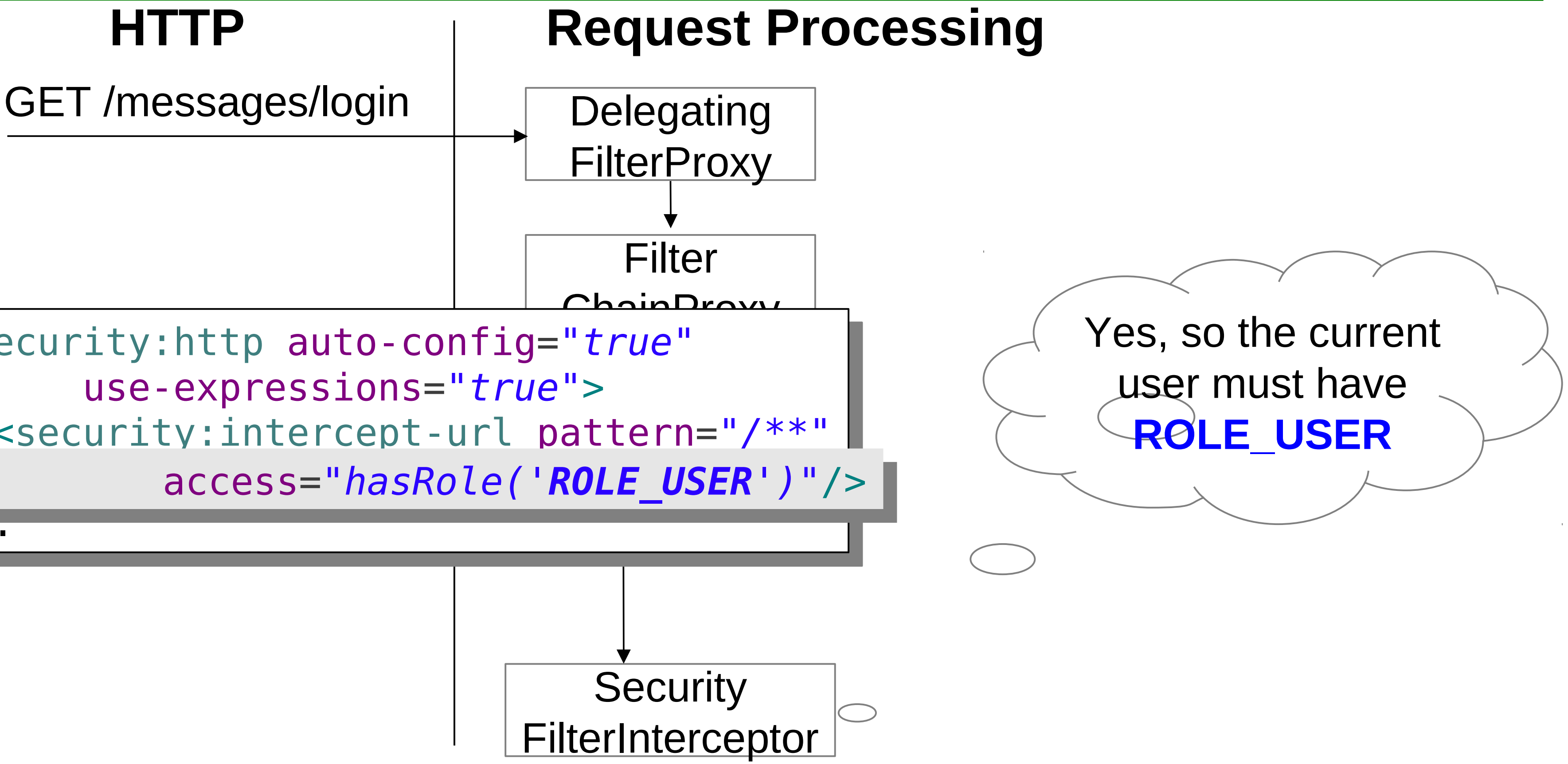


Unauthenticated Request to Protected Resource



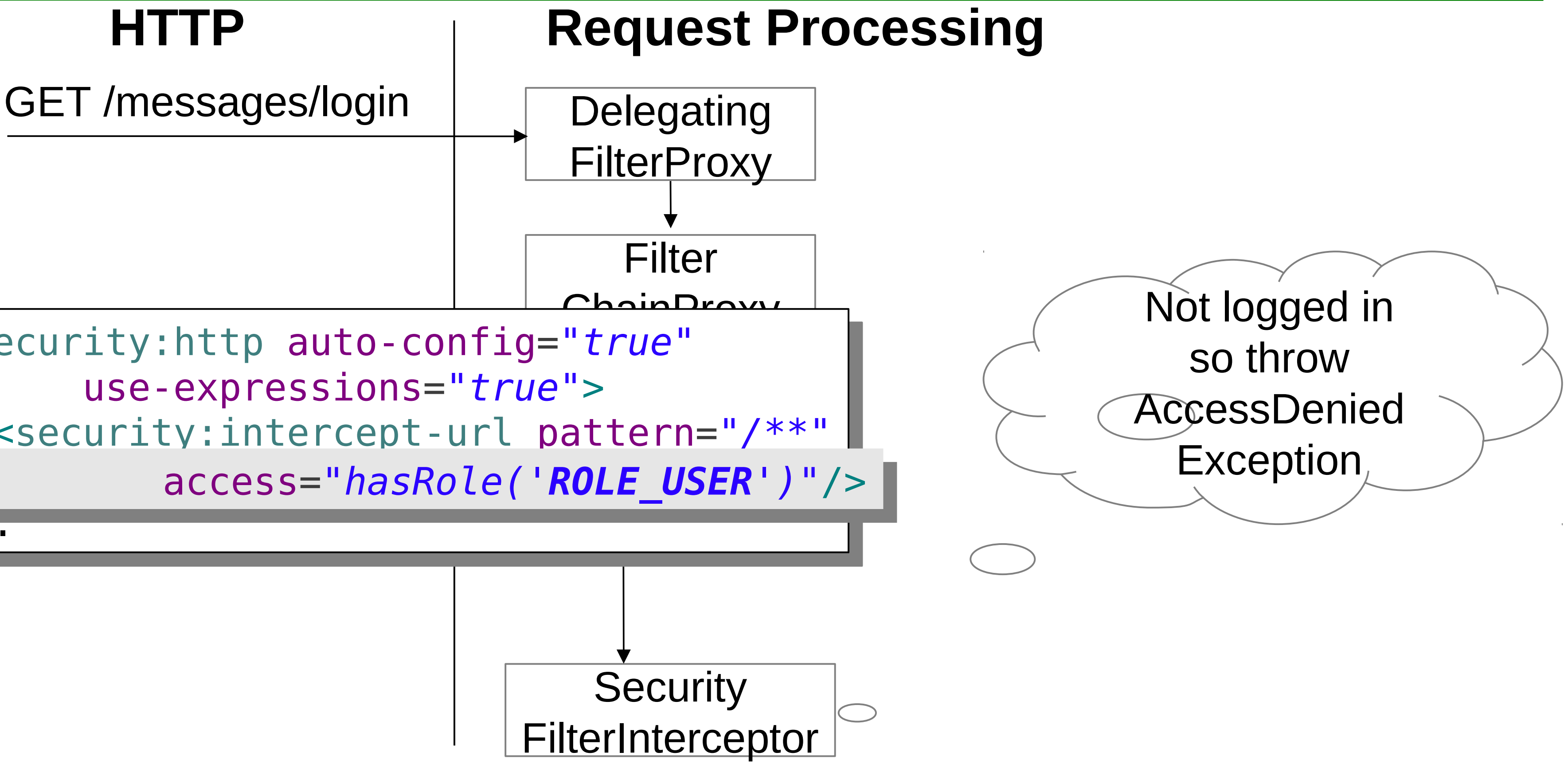
```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')" />
...
```

Unauthenticated Request to Protected Resource



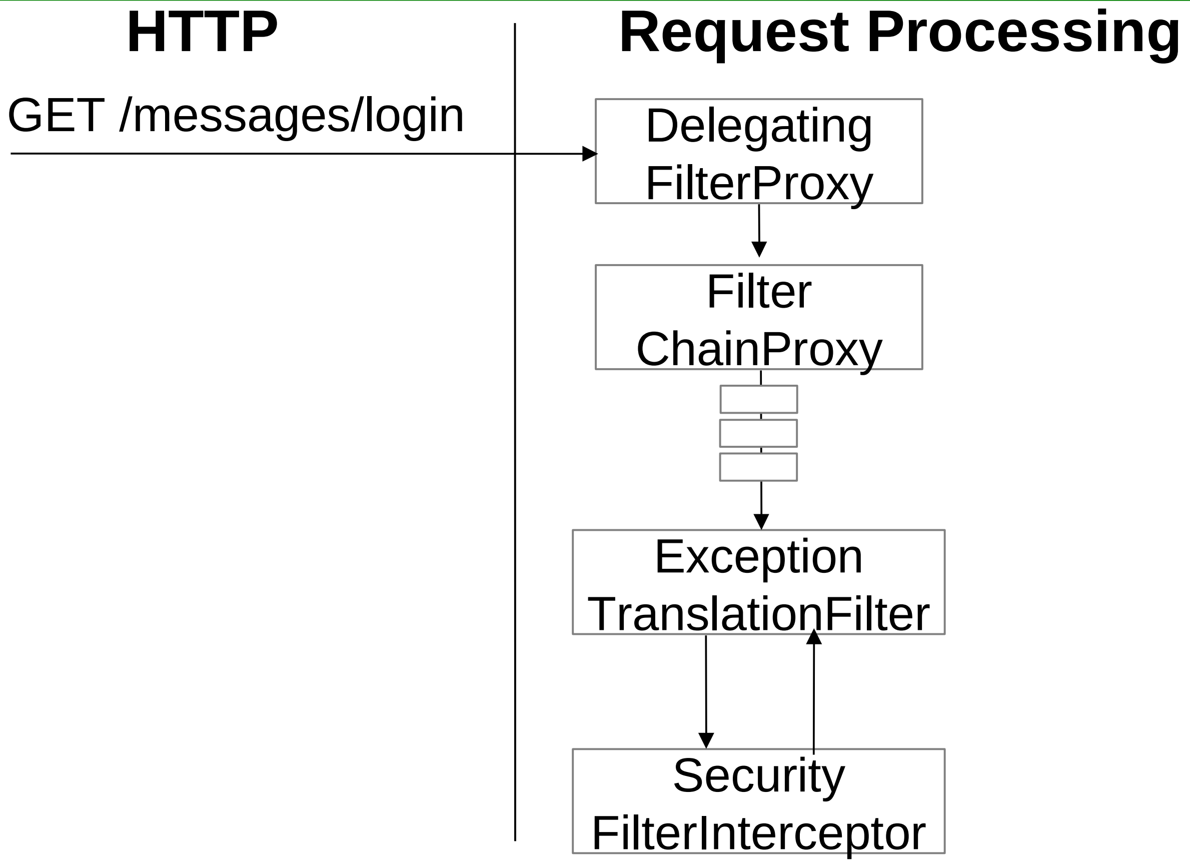
```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
            access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

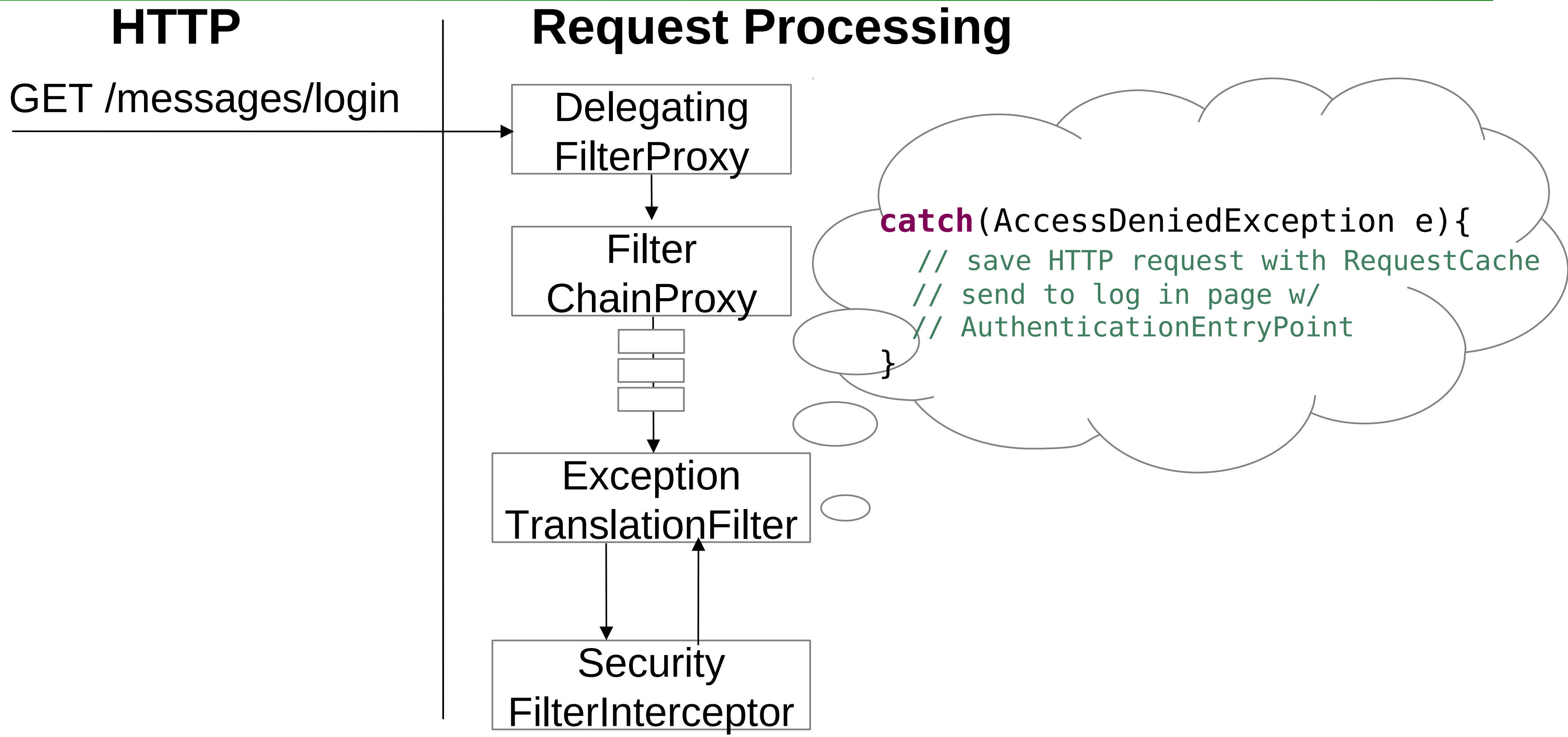


```
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
           access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

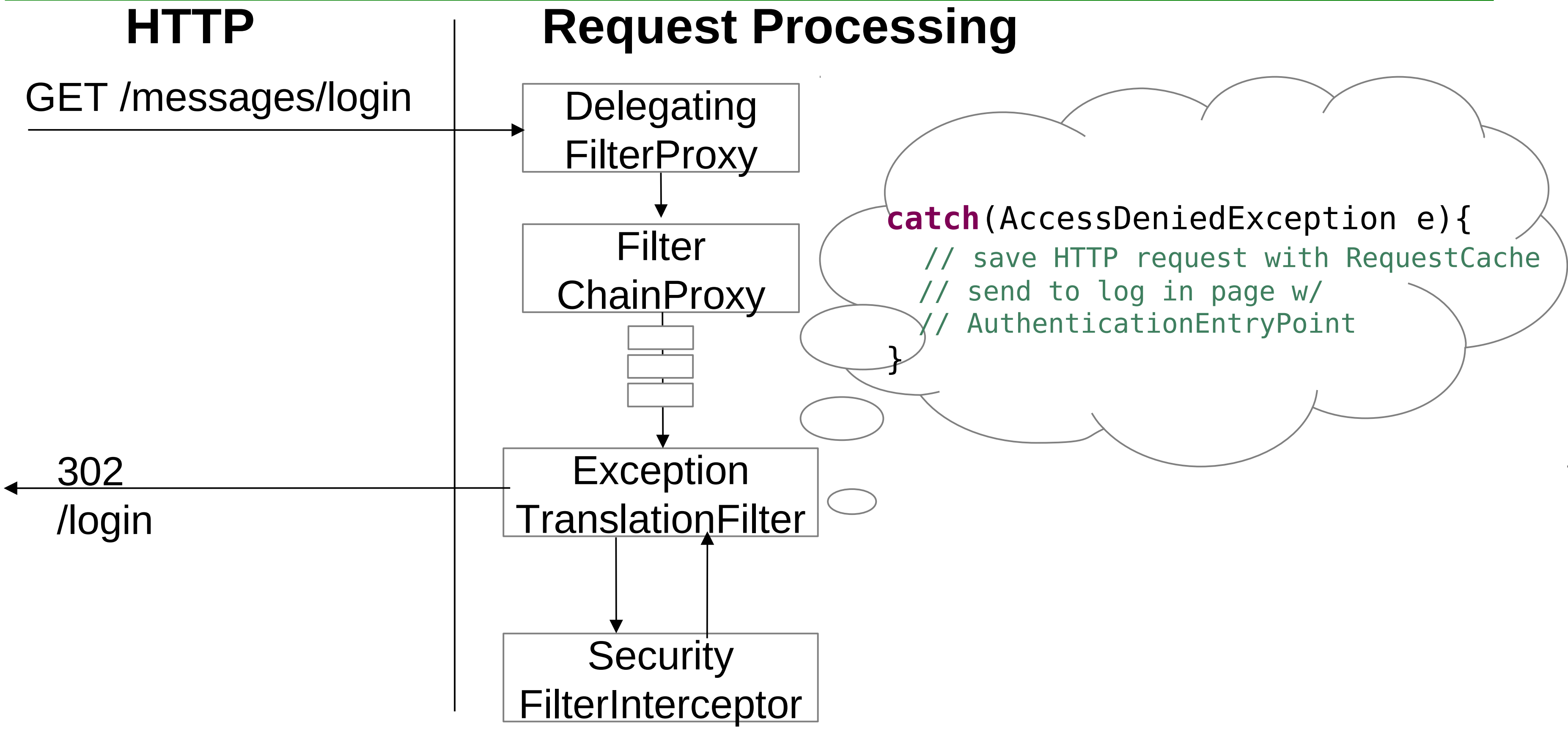


Unauthenticated Request to Protected Resource



```
catch(AccessDeniedException e){
  // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

Unauthenticated Request to Protected Resource



```
catch(AccessDeniedException e){
  // save HTTP request with RequestCache
  // send to log in page w/
  // AuthenticationEntryPoint
}
```

src/main/webapp/WEB-INF/security.xml

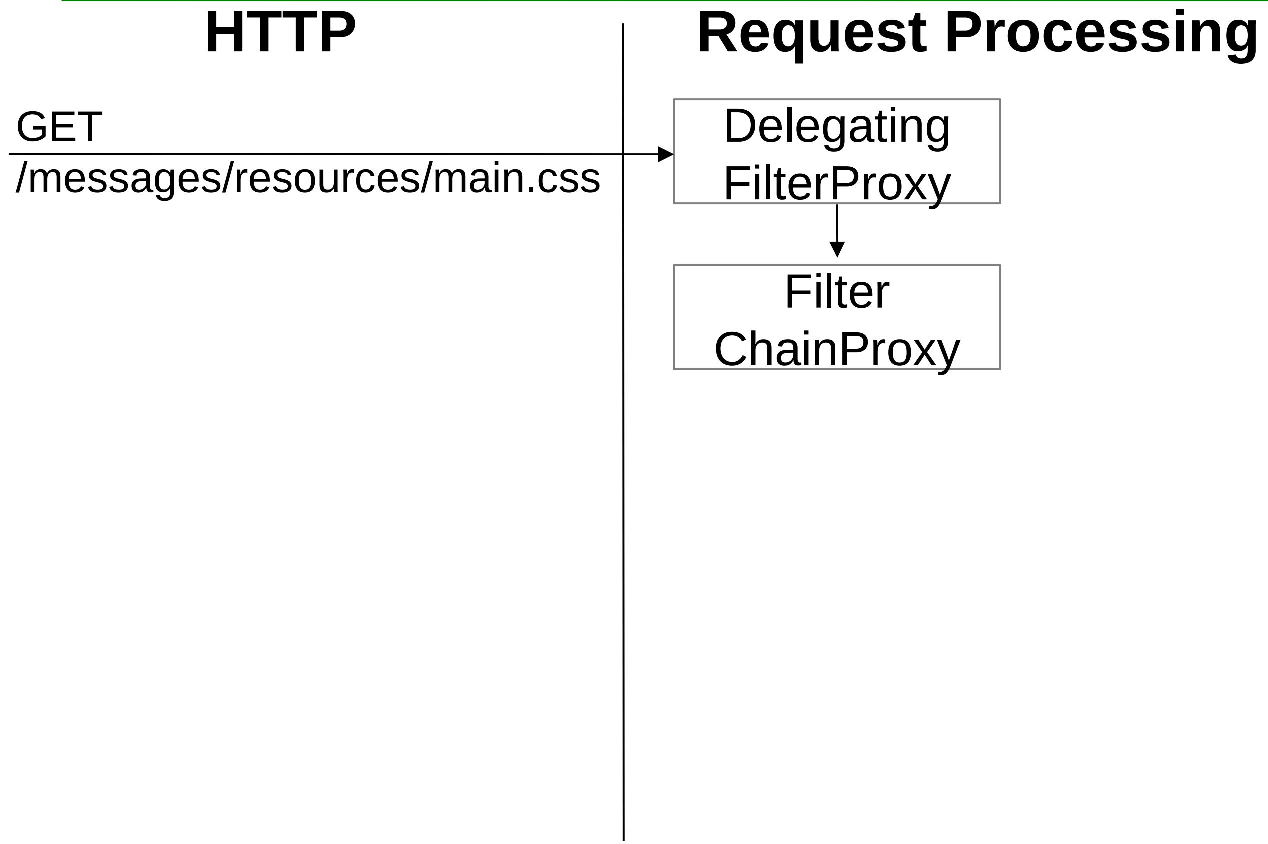
```
<security:http use-expressions="true">
   <security:intercept-url pattern="/login"
         access="permitAll"/>
   <security:intercept-url pattern="/**"
         access="hasRole('ROLE_USER')"/>
  <security:form-login login-page="/login"
         authentication-failure-url="/login?error"/>
</security:http>
```

src/main/webapp/WEB-INF/views/login.jspx

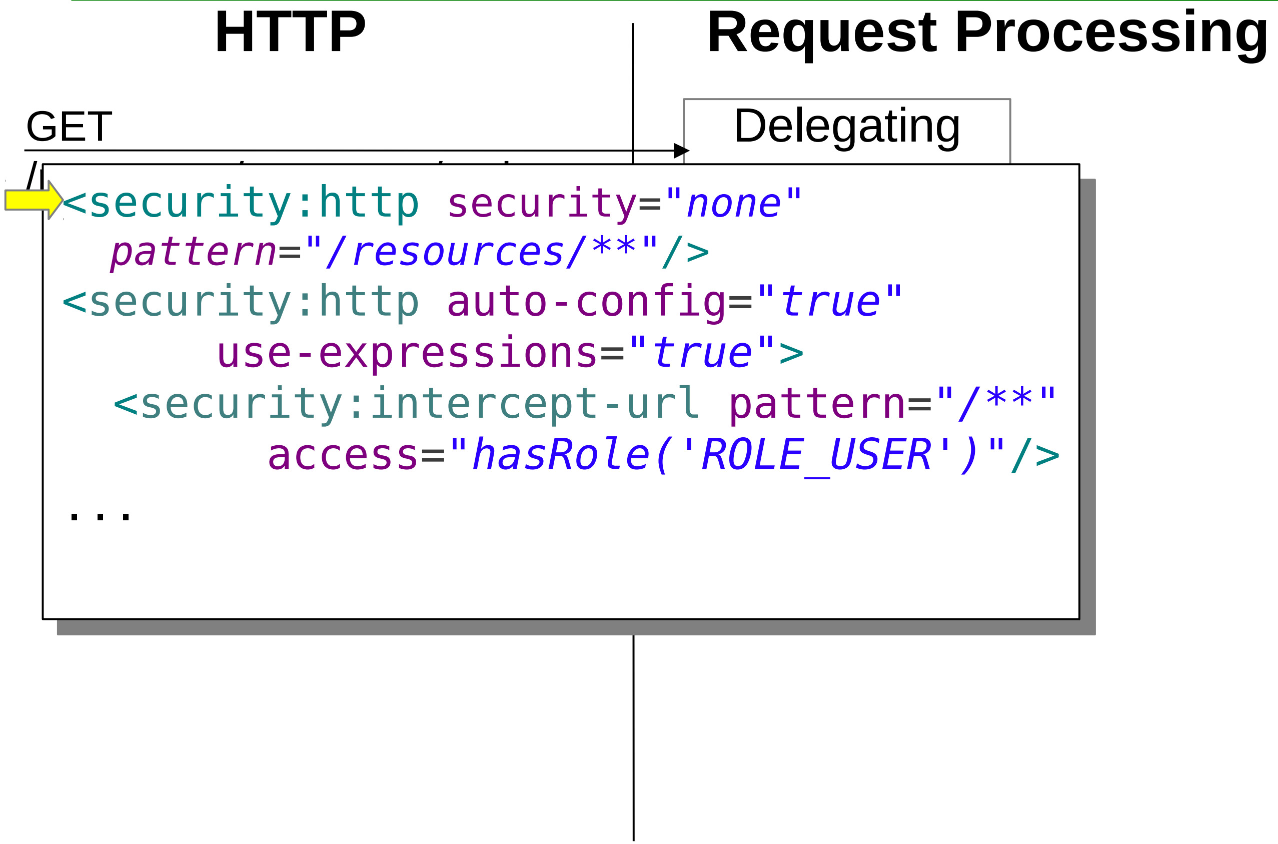
```
<c:url value="/j_spring_security_check" var="loginUrl"/>
<form action="${loginUrl}" method="post">
    <c:if test="${param.error != null}">
        <div class="alert alert-error">
            Failed to login.
            <c:if test="${SPRING_SECURITY_LAST_EXCEPTION != null}">
              Reason: <c:out value="${SPRING_SECURITY_LAST_EXCEPTION.message}" />
            </c:if>
        </div>
    </c:if>
    <label for="username">Username</label>
    <input type="text" id="username" name="j_username"/>
    <label for="password">Password</label>
    <input type="password" id="password" name="j_password"/>
    <div class="form-actions">
        <input id="submit" class="btn" name="submit" type="submit" value="Login"/>
    </div>
</form>
```

Multiple <http> Support

Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

HTTP

Request Processing

GET

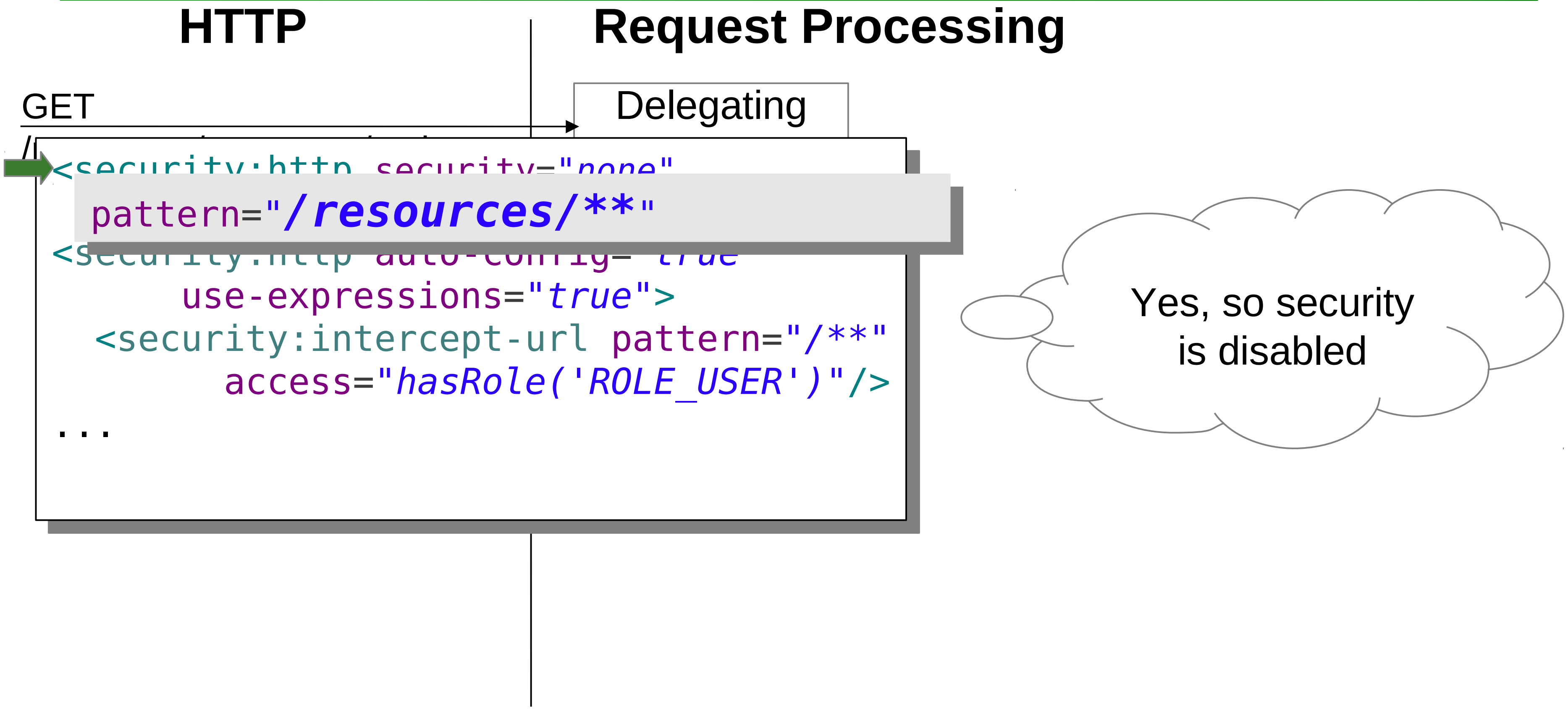
/

Delegating

```
<security:http security="none"
  pattern="/resources/**"
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

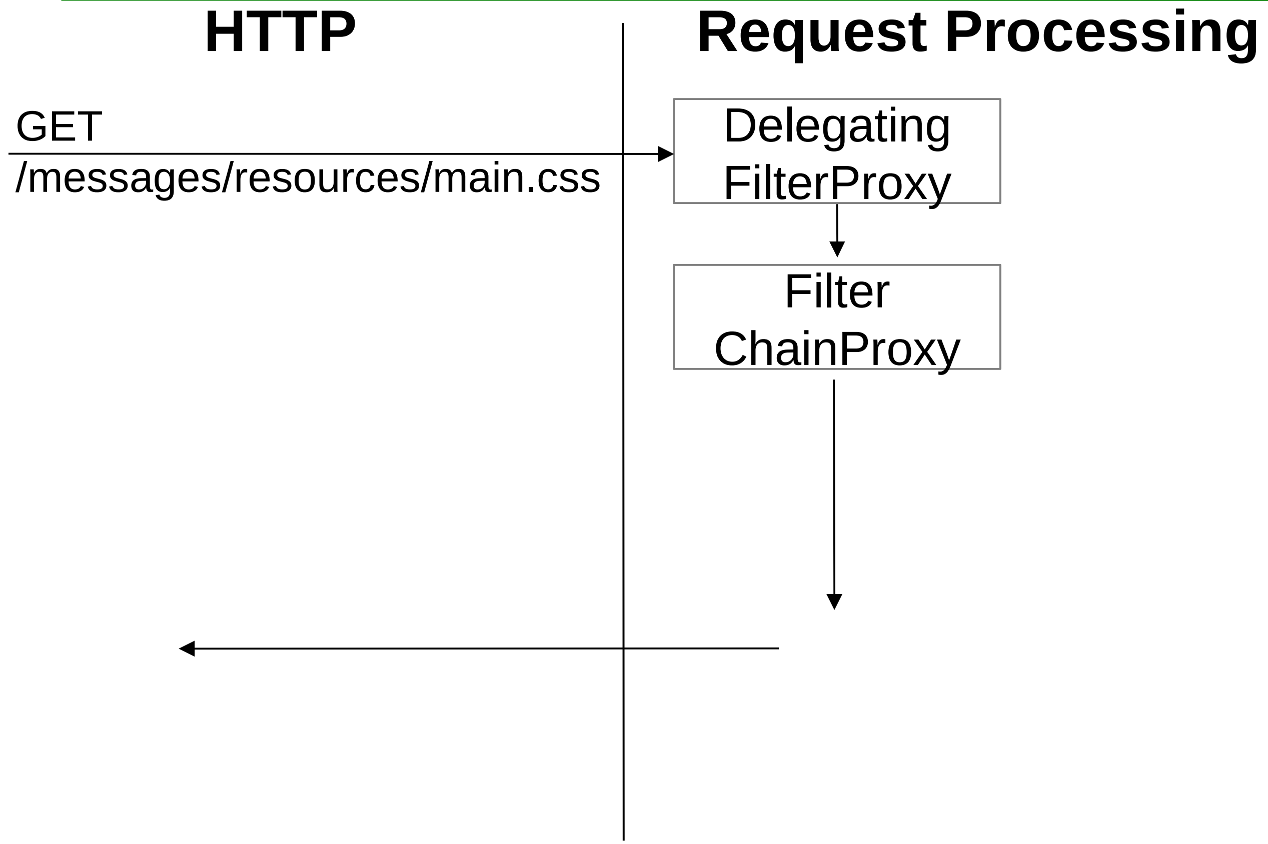
Does
/resources/main.css
match ***/resources/*****

Unauthenticated Request to Protected Resource

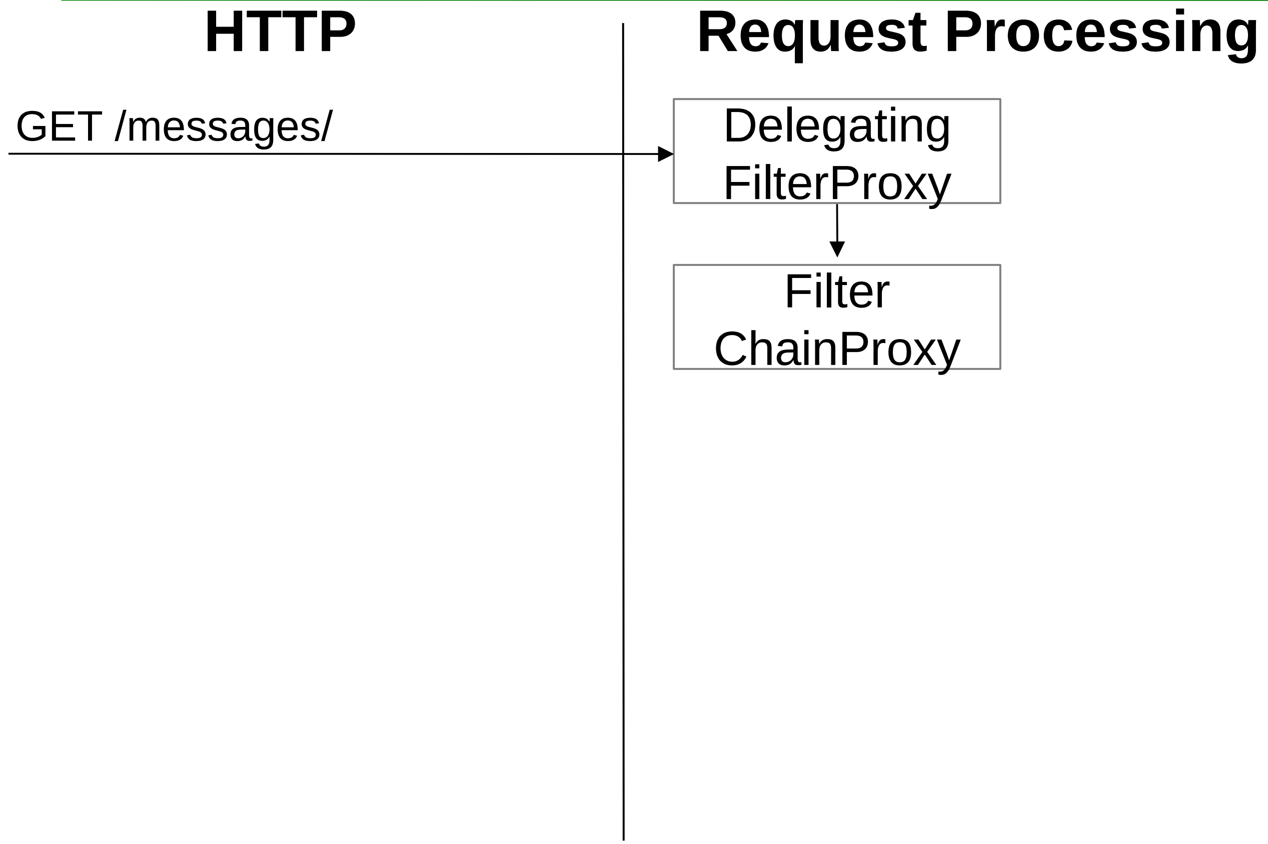


```
<security:http security="none"
  pattern="/resources/**"
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

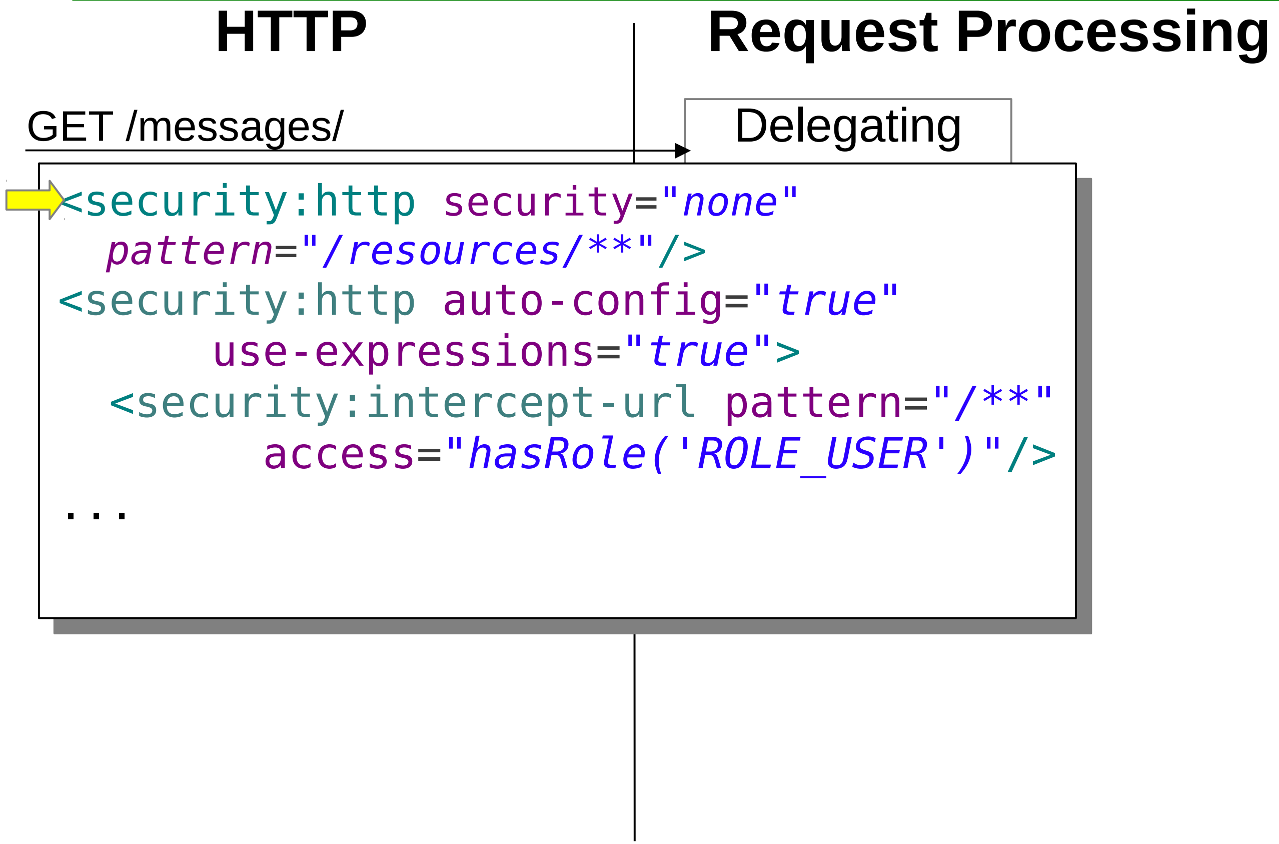
Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource

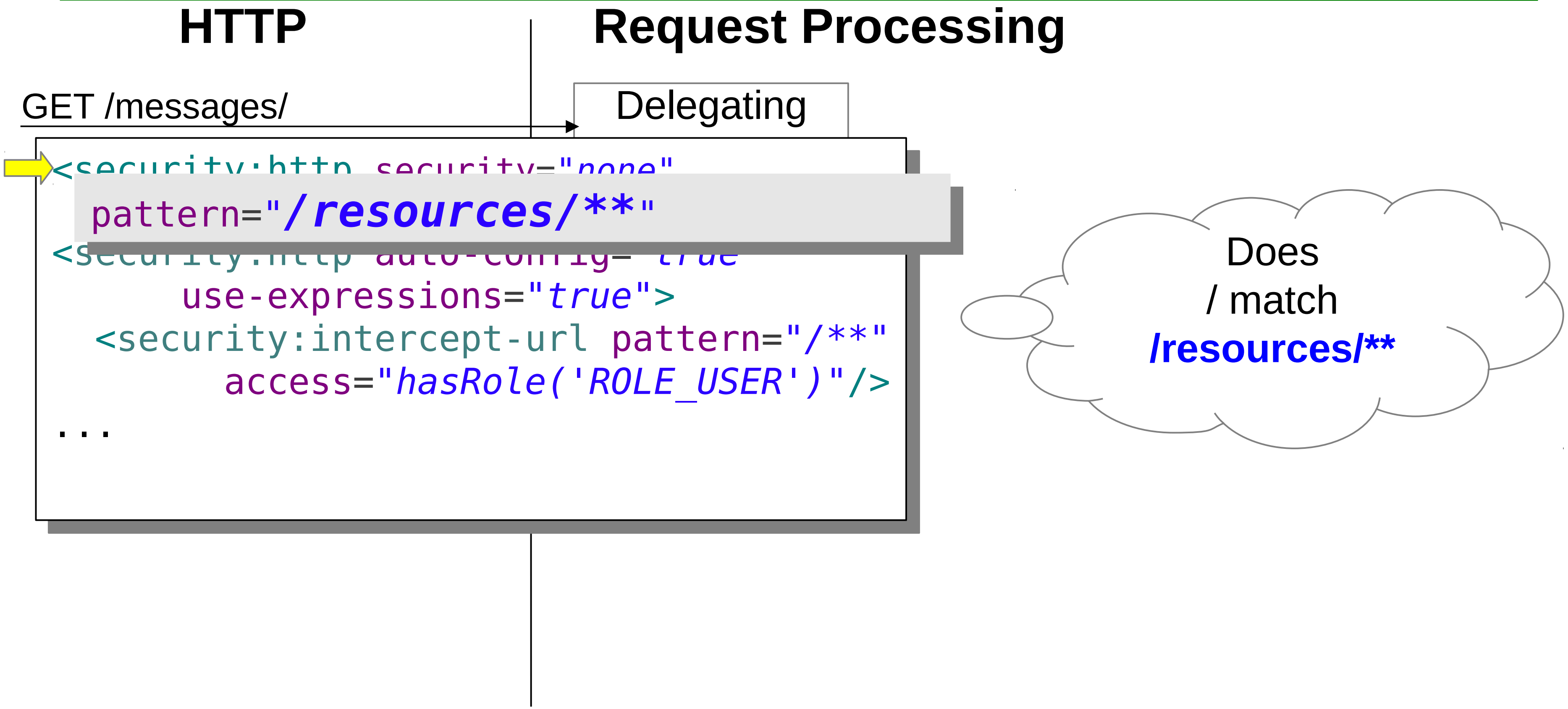


Unauthenticated Request to Protected Resource



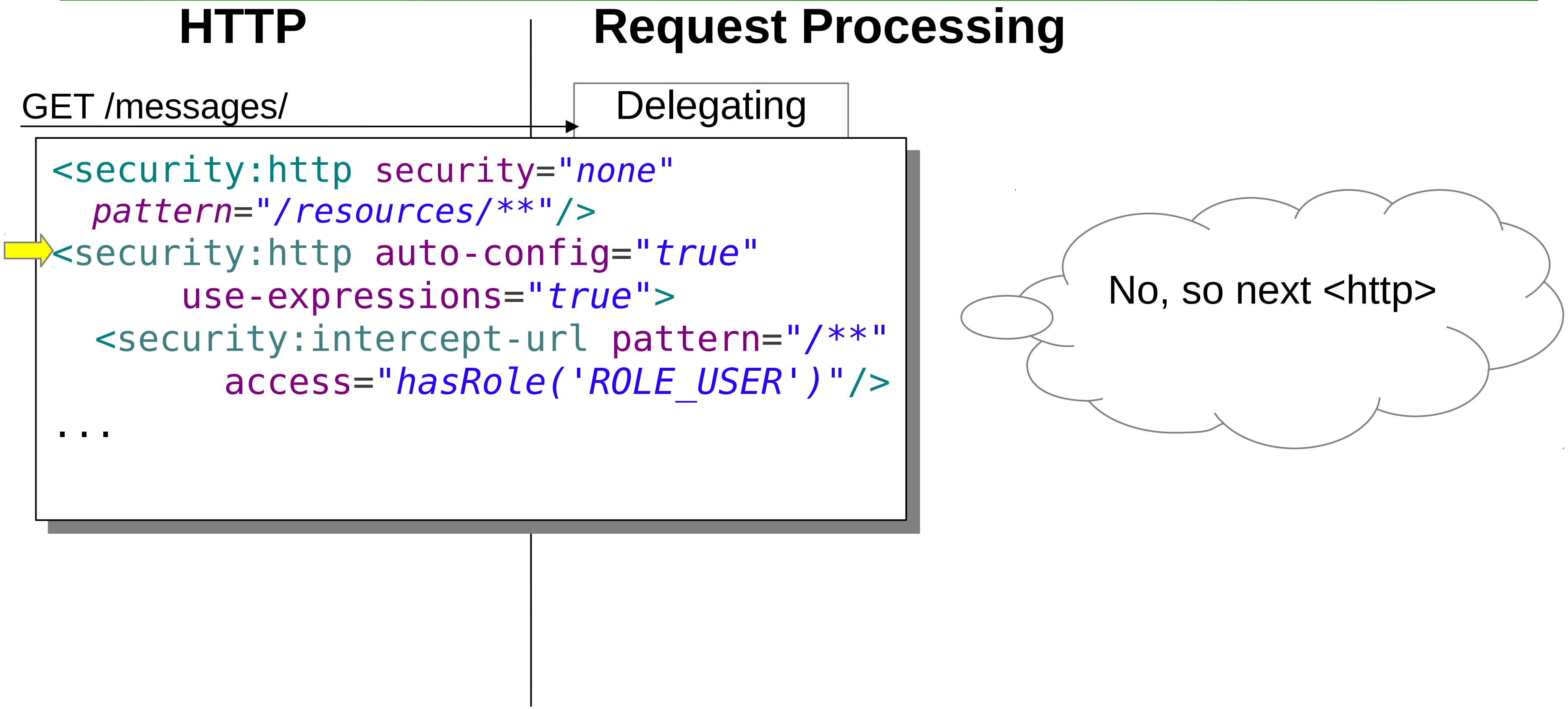
```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
   <security:intercept-url pattern="/**"
         access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource



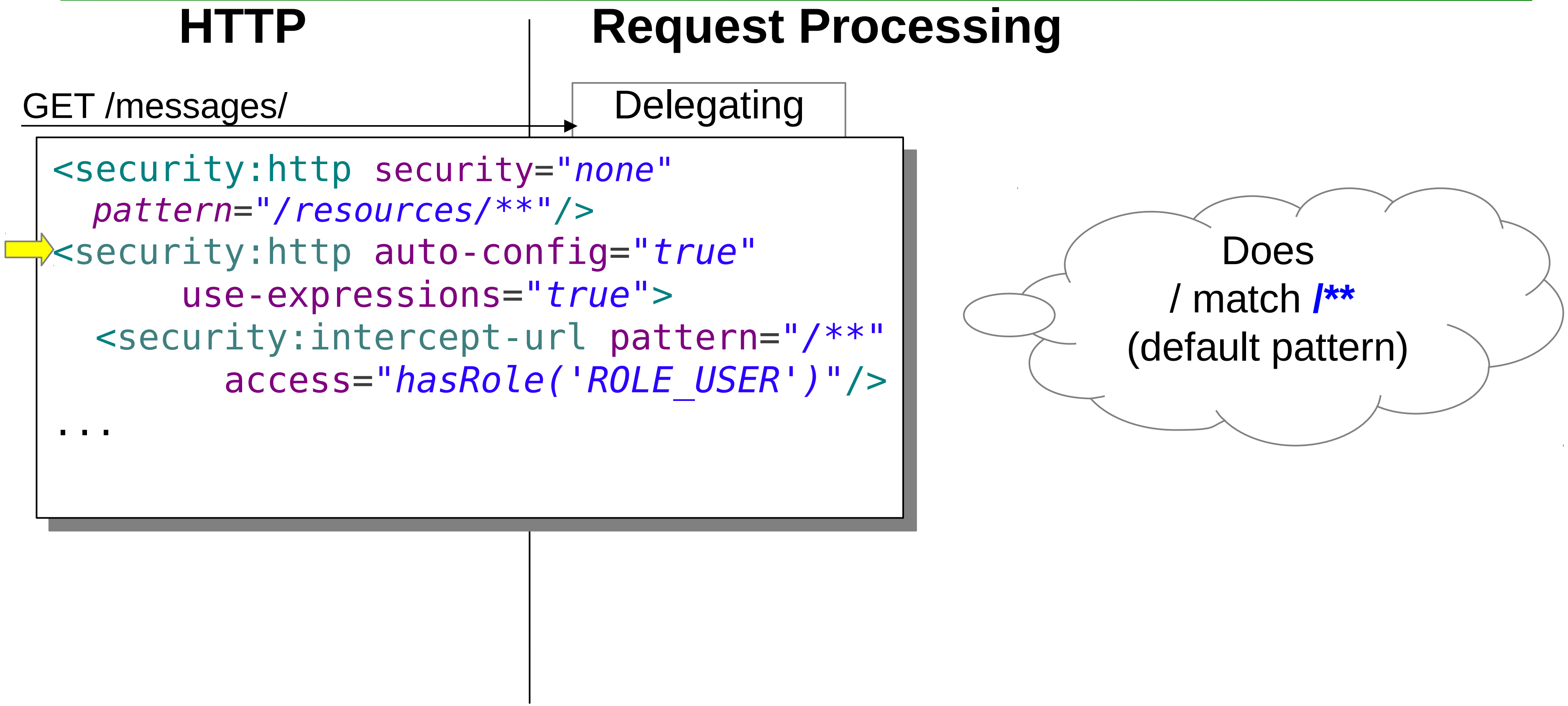
```
<security:http security="none"
  pattern="/resources/**"
<security:http auto-config="true"
      use-expressions="true">
   <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource



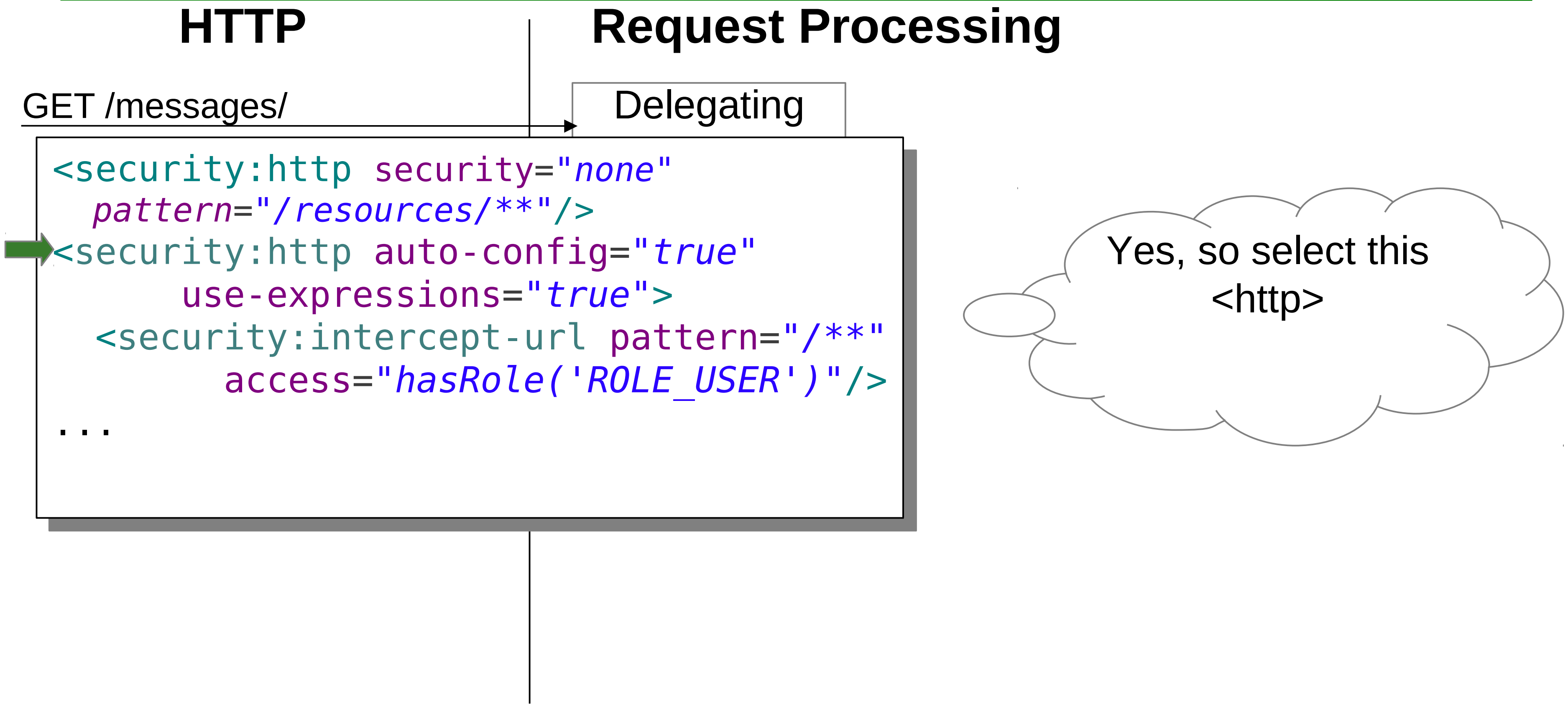
```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

HTTP

Request Processing

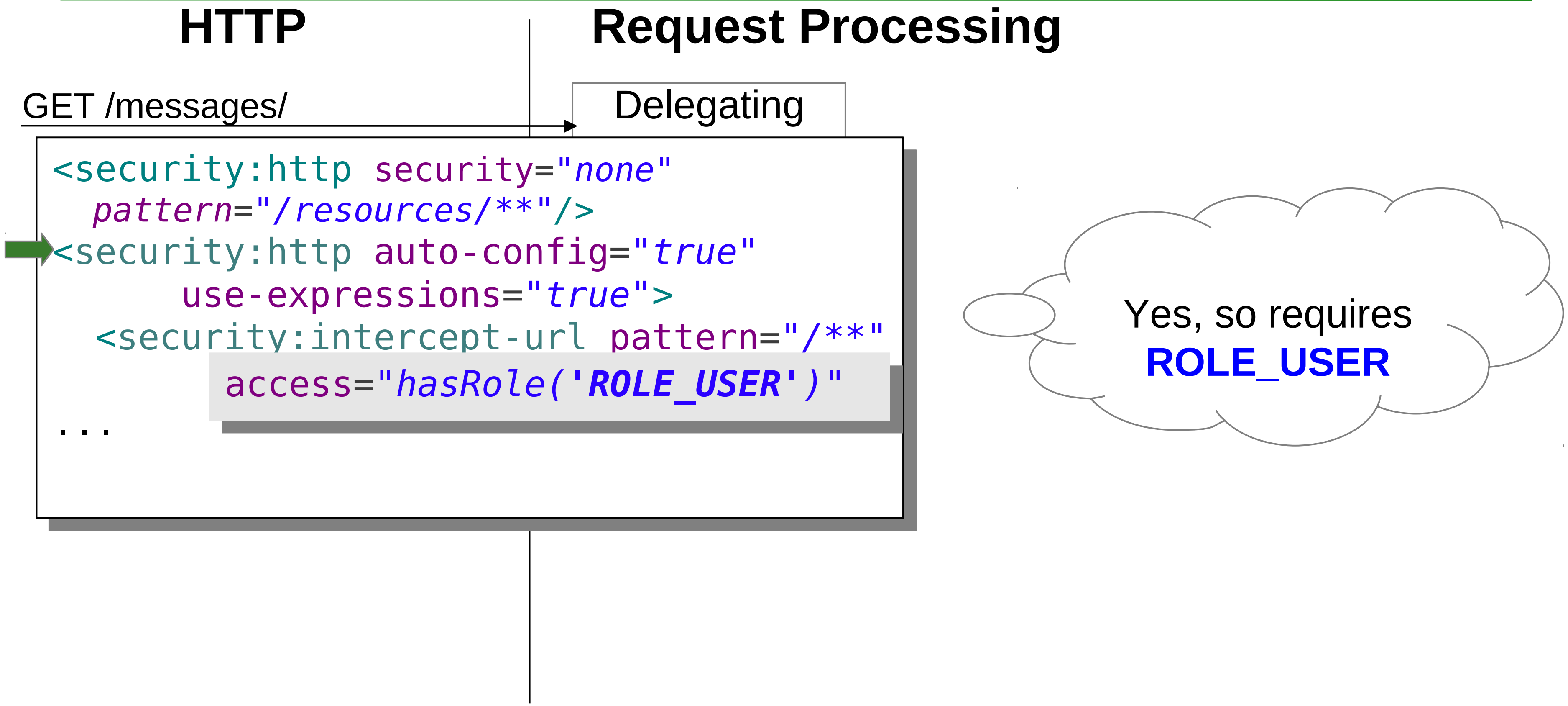
GET /messages/

Delegating

```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-ur pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

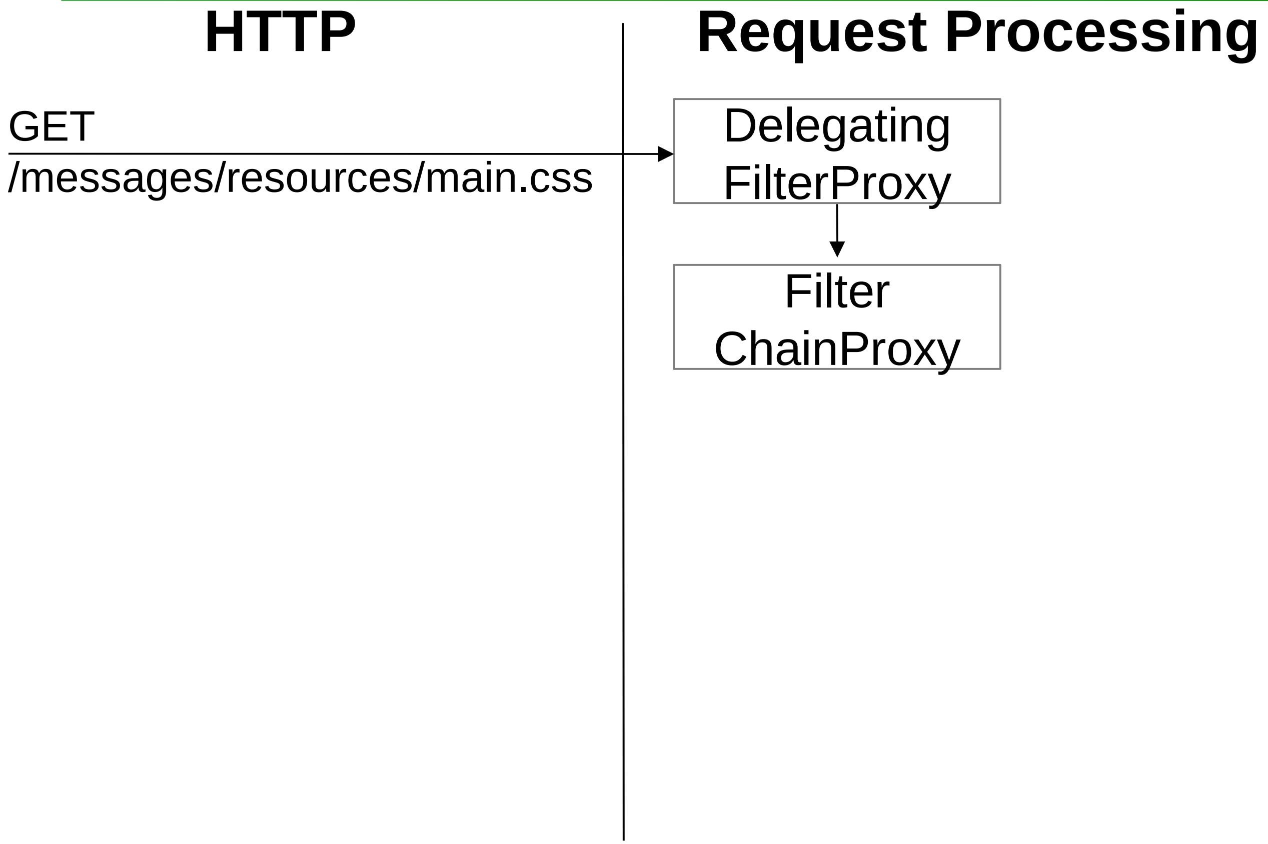
Does / match /**

Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**"/>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"
...
```

Unauthenticated Request to Protected Resource



Unauthenticated Request to Protected Resource

HTTP | **Request Processing**

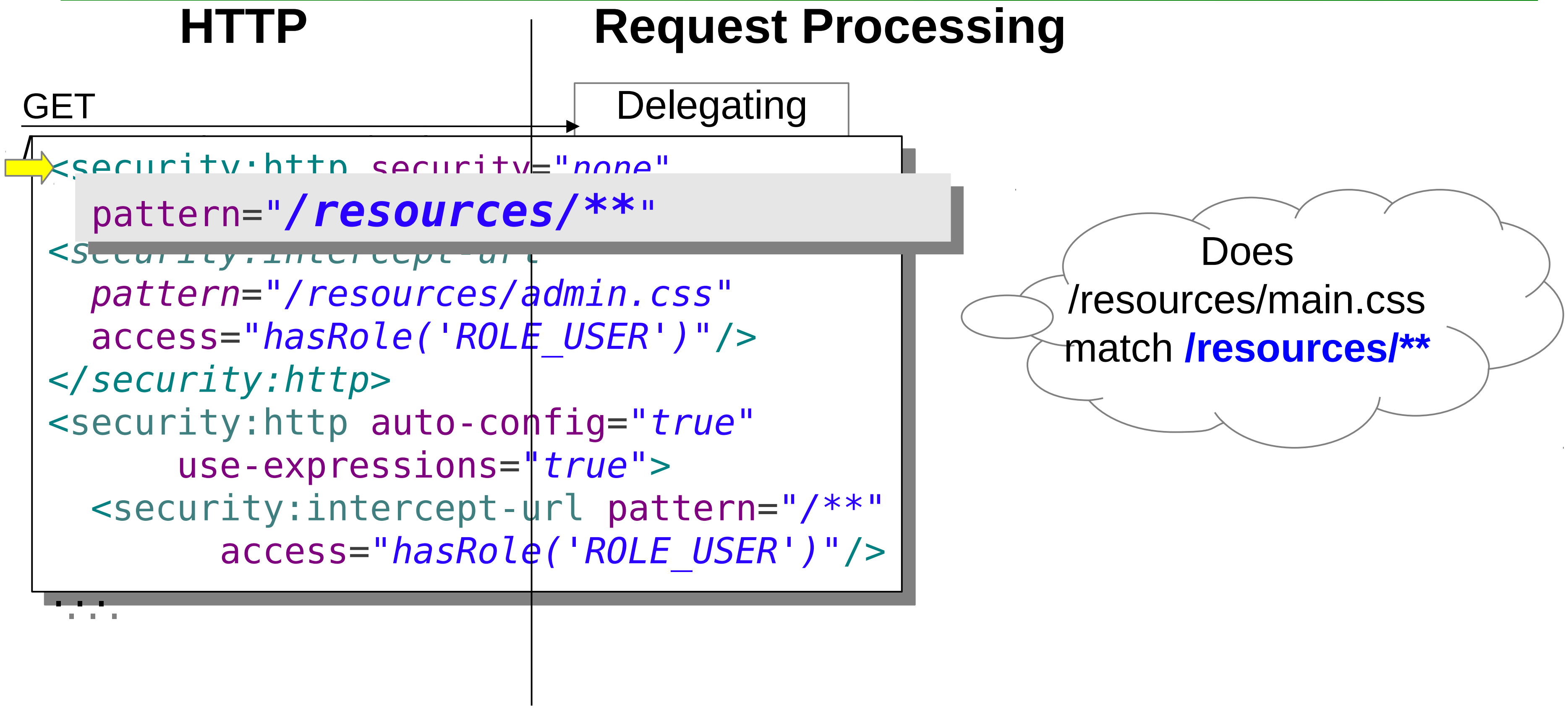
GET

/r

Delegating

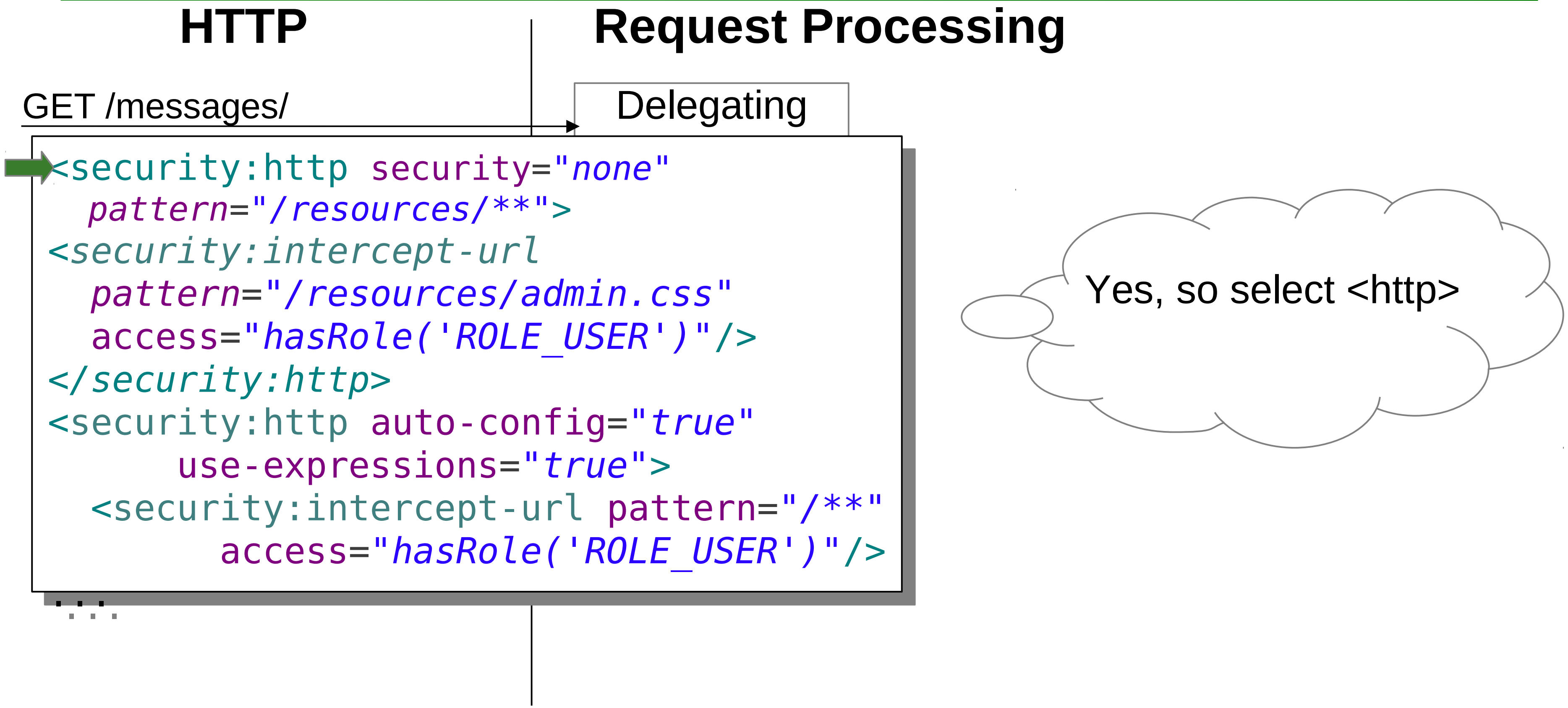
```
<security:http security="none"
  pattern="/resources/**">
<security:intercept-url
  pattern="/resources/admin.css"
  access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
```

Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**"
<security:intercept-url
  pattern="/resources/admin.css"
  access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**">
<security:intercept-url
  pattern="/resources/admin.css"
  access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
        access="hasRole('ROLE_USER')"/>
...
```

Unauthenticated Request to Protected Resource

HTTP | **Request Processing**

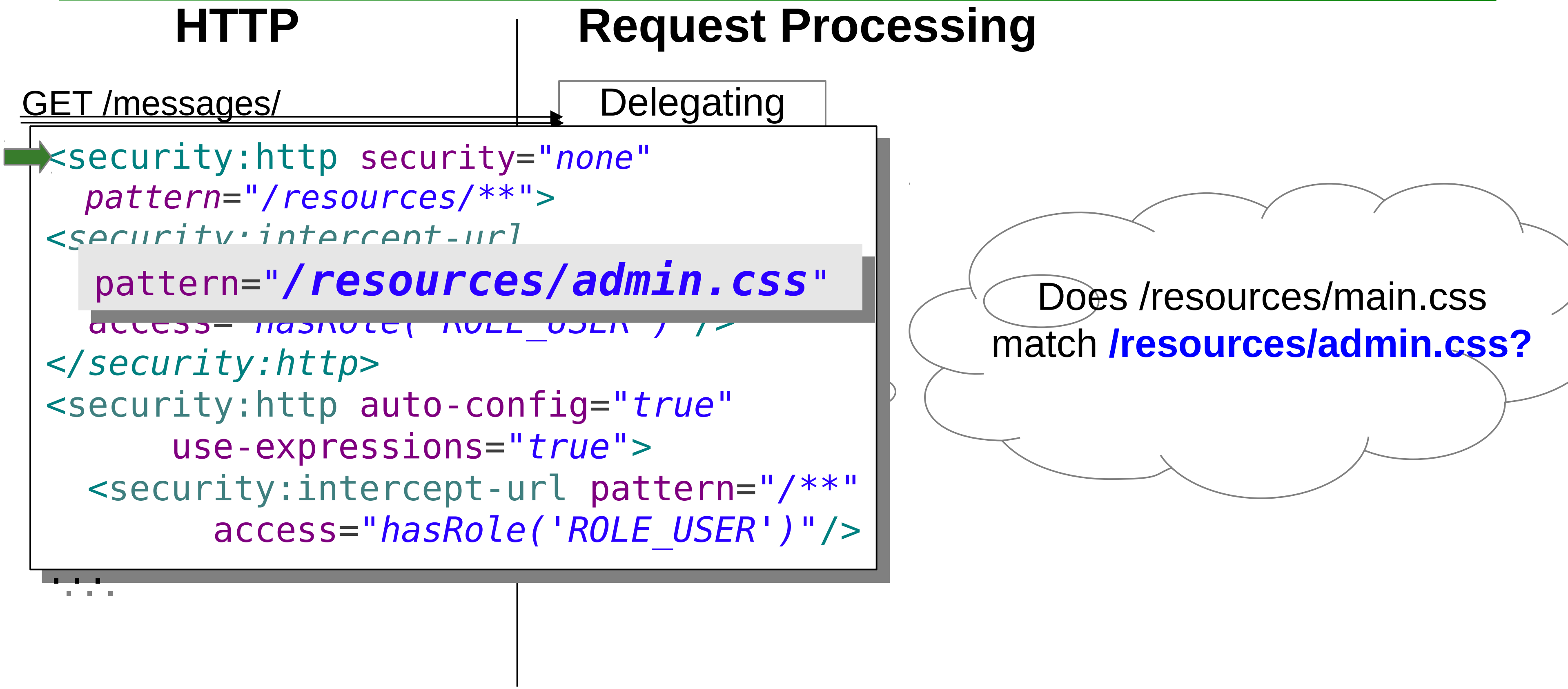
GET /messages/

Delegating

```
<security:http security="none"
  pattern="/resources/**">
<security:intercept-url
  pattern="/resources/admin.css"
  access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
        use-expressions="true">
  <security:intercept-url pattern="/**"
          access="hasRole('ROLE_USER')"/>
...
```

Does /resources/main.css match ***/resources/admin.css?***

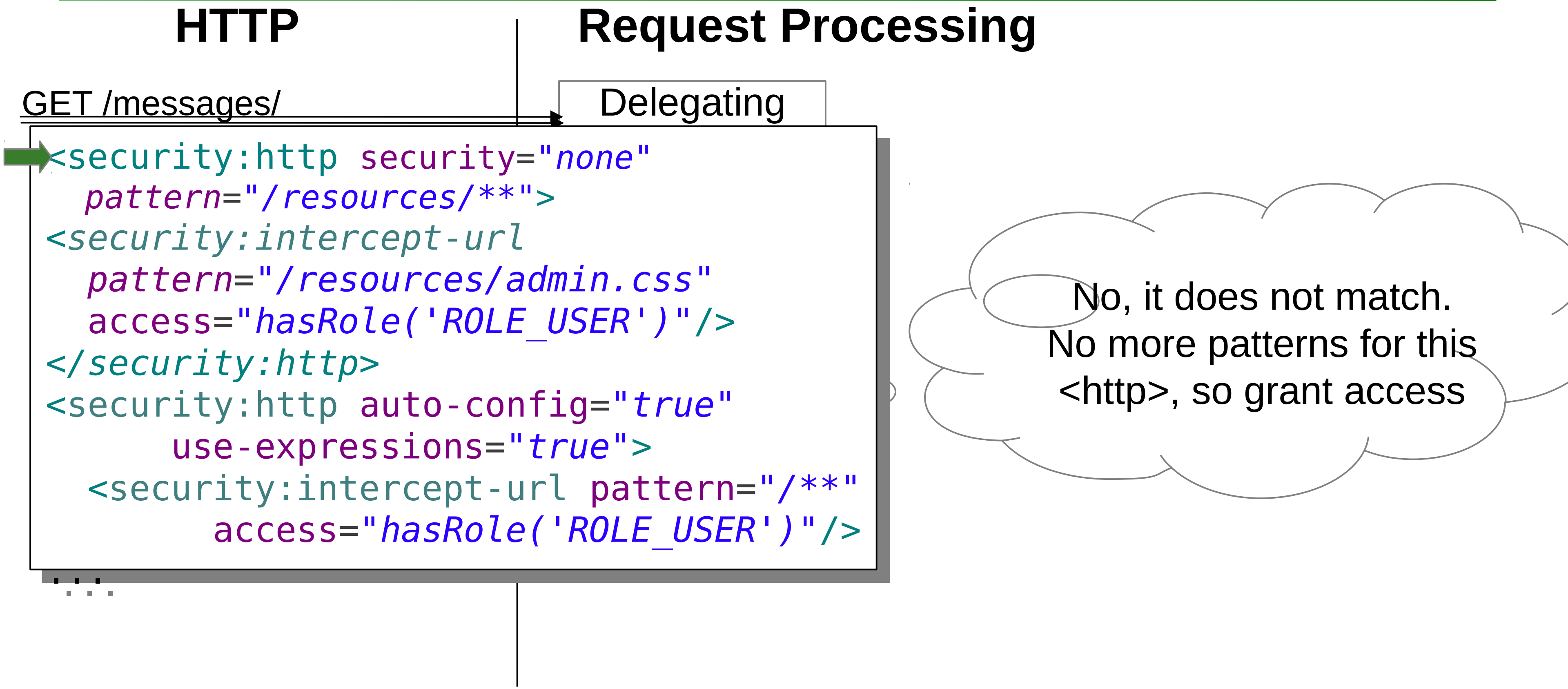
Unauthenticated Request to Protected Resource



```
<security:http security="none"
  pattern="/resources/**">
<security:intercept-url
   pattern="/resources/admin.css"
  access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
      use-expressions="true">
  <security:intercept-url pattern="/**"
         access="hasRole('ROLE_USER')"/>
....
```

Does /resources/main.css match ***/resources/admin.css?***

Unauthenticated Request to Protected Resource



```
<security:http security="none"
   pattern="/resources/**">
<security:intercept-url
   pattern="/resources/admin.css"
   access="hasRole('ROLE_USER')"/>
</security:http>
<security:http auto-config="true"
       use-expressions="true">
  <security:intercept-url pattern="/**"
         access="hasRole('ROLE_USER')"/>
```

No, it does not match.
No more patterns for this
<http>, so grant access

Log Out

src/main/webapp/WEB-INF/security.xml

```xml
<security:http use-expressions="true">
  ...
  <security:logout />
</security:http>
```

JSP Tags

src/main/webapp/WEB-INF/views/header.jspx

```
<jsp:root …
xmlns:sec="http://www.springframework.org/security/tags">
  <sec:authorize access="authenticated">
    … is the user authenticated …
    <span class="navbar-text">
      Welcome,
      <sec:authentication
          property="name"/>
    </span>
  </sec:authorize>
  …
</jsp:root>
```

Method Level Security

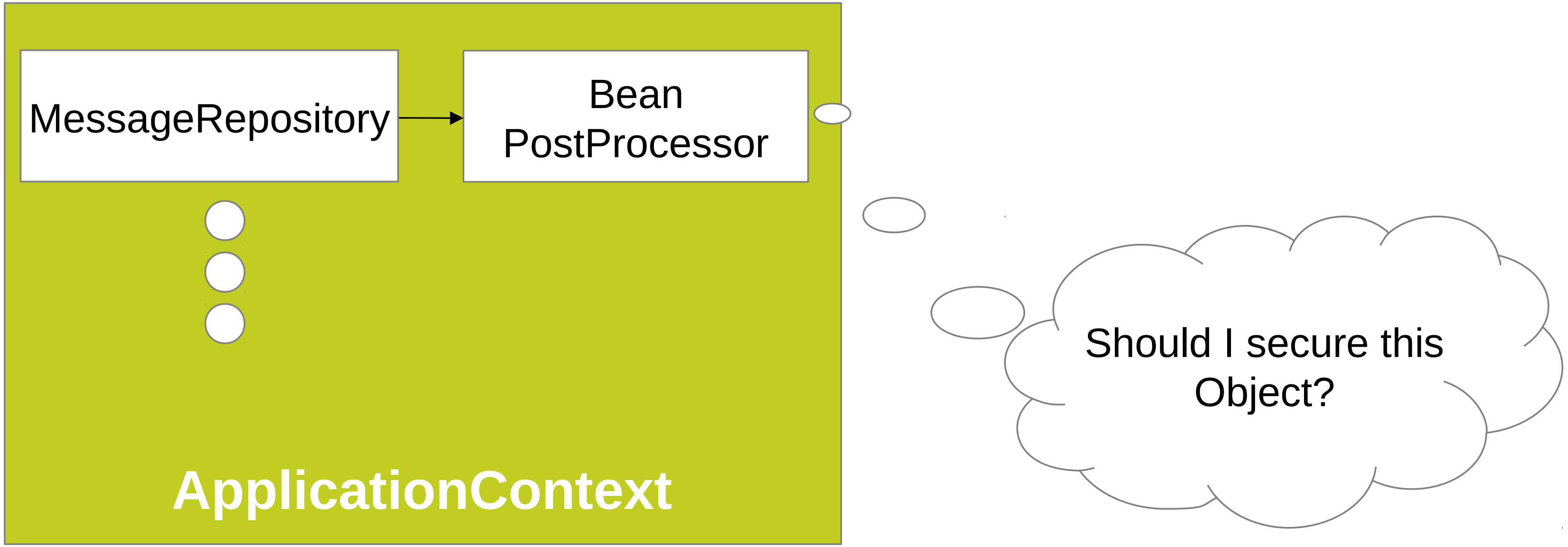
src/main/webapp/WEB-INF/security.xml

```xml
<security:global-method-security
    pre-post-annotations="true"/>
```

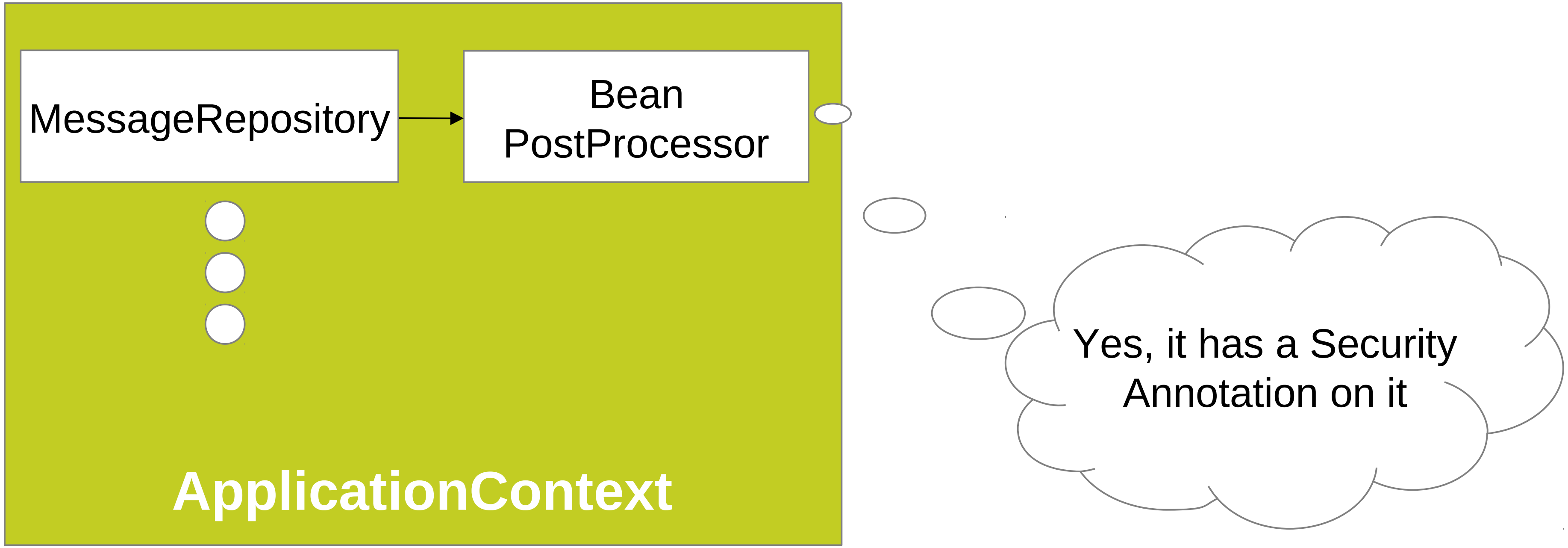
MessageRepository.java

```
    @PostAuthorize("hasRole('ROLE_USER')")
    Message findOne(Long id);
```

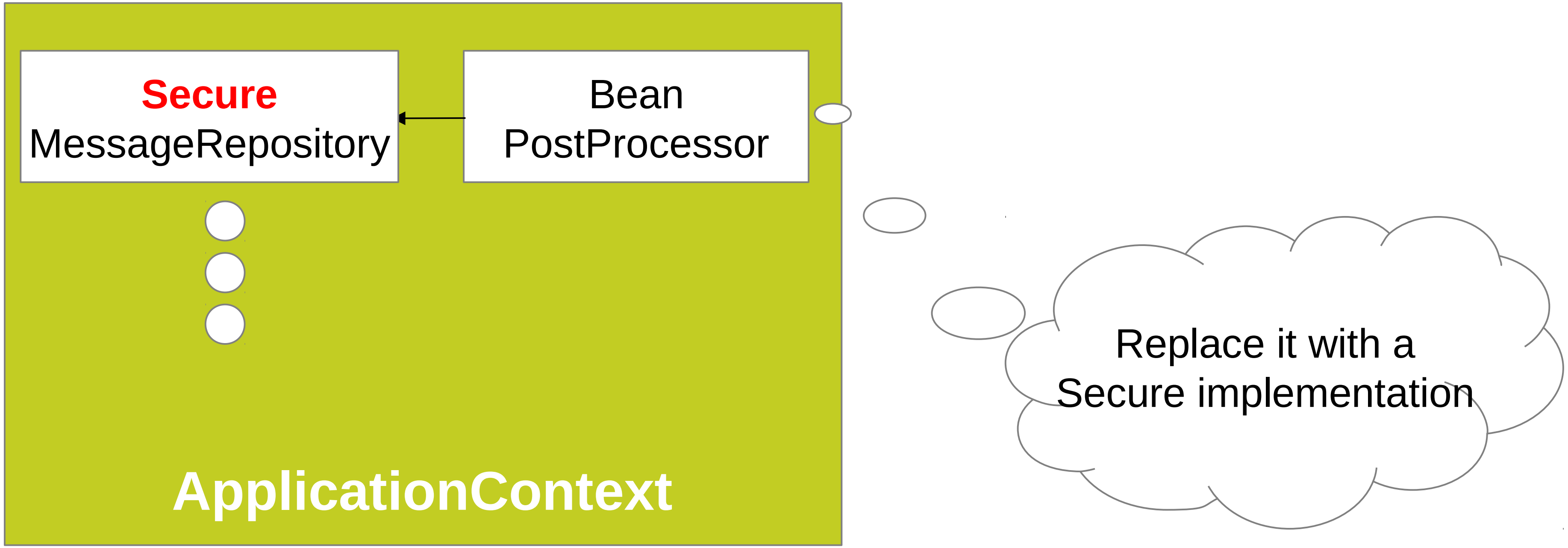
Method Level Security



Method Level Security



Method Level Security



SecureMessageRepository

```
public class SecureMessageRepository implements
      MessageRepository {
  public Message findOne(Long id) {
    // PreAuthorize checks
    Message result = delegate.findOne(id);
    // PostAuthorize checks
    return result;
  }
  …
  // delegate = original MessageRepository
  private MessageRepository delegate;
}
```

Learn More. Stay Connected.

At SpringOne 2GX:

- When and why would I use OAuth2?
- Making Connections with Spring Social

Web: springsource.org

- Github: github.com/rwinch/getting-started-springsecurity-31
- Newsletter: springsource.org/news-events
- Twitter: twitter.com/SpringSecurity twitter.com/rob_winch
- YouTube: youtube.com/user/SpringSourceDev
- LinkedIn: springsource.org/linkedin