Advanced OpenEdge REST/Mobile Security

Securing your OpenEdge Web applications

Michael Jacobs
August 2013

PROGRESS software

Legal Disclaimer

- The contents of these materials are confidential information of Progress Software Corporation or its affiliated entities (collectively Progress Software). These materials may also include information about future features, products, technologies and/or services that are under consideration by Progress Software. Progress Software makes no commitments with respect to such future features, products, technologies and/or services. The information contained in these materials is subject to change. Progress Software does not guarantee any release dates or that there will be a release of any future features, products, technologies and/or services (if any) referenced herein.

Session Agenda

- General Web application security overview

Session Agenda

- General Web application security overview
- OpenEdge Web application security specifics
  - REST/Mobile Web applications

Session Agenda

- General Web application security overview
- OpenEdge Web application security specifics
  - REST/Mobile Web applications
- Advanced security considerations

Session Agenda

- General Web application security overview
- OpenEdge Web application security specifics
  - REST/Mobile Web applications
- Advanced security considerations
- Security reference materials

Web Application Environment Assumptions

- All internet data is of high value needing strong security

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall...
  firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall...
  firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall... firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall...
  firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately
  - Anything placed in them is a target

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall... firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately
  - Anything placed in them is a target
  - *Security by obscurity* wastes your time and only servers to annoy them

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall... firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately
  - Anything placed in them is a target
  - *Security by obscurity* wastes your time and only servers to annoy them
- SSL does not prevent all data leaks, but it does pretty good

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall...
  firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately
  - Anything placed in them is a target
  - *Security by obscurity* wastes your time and only servers to annoy them
- SSL does not prevent all data leaks, but it does pretty good
- OE admin, name, & app servicer security is not DMZ quality

Web Application Environment Assumptions

- All internet data is of high value needing strong security
- Your web server is connected to the internet via a firewall... firewalls leak and no web server or Servlet engine (Tomcat) is ever installed with a secure configuration
- Your web application will be discovered and probed < 1 min after deployment
- Attackers know web servers and applications intimately
  - Anything placed in them is a target
  - *Security by obscurity* wastes your time and only servers to annoy them
- SSL does not prevent all data leaks, but it does pretty good
- OE admin, name, & app servicer security is not DMZ quality
- Robots, crawlers, and spiders will harm you

Web Application Security Goals

- Design and build security into my web application from day 1

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
    - Authentication

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
  - Authentication
  - Authorization

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
  - Authentication
  - Authorization
  - Session management

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
  - Authentication
  - Authorization
  - Session management
- Use strong, peer reviewed, industry security technologies

Web Application Security Goals

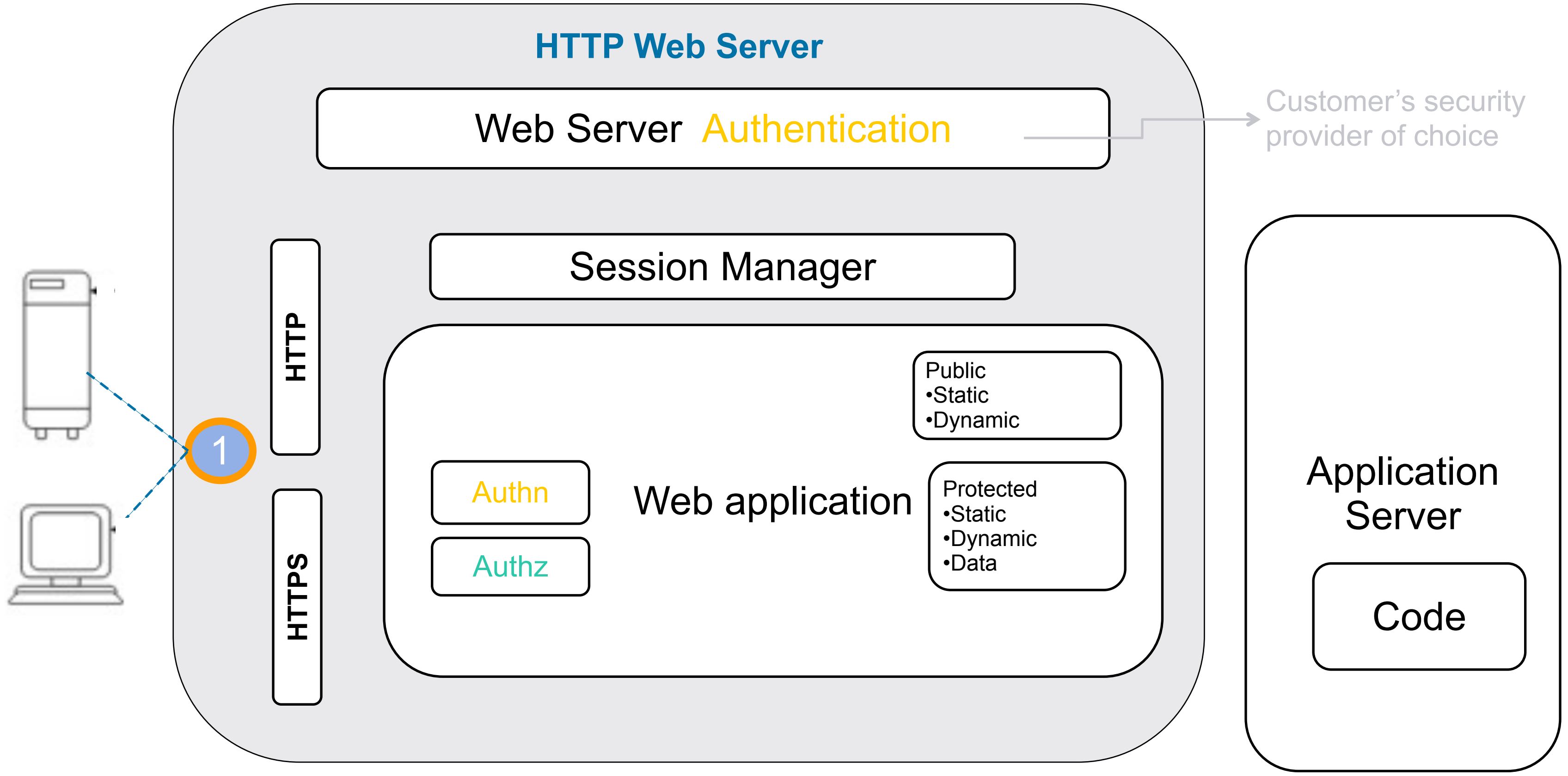
- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
  - Authentication
  - Authorization
  - Session management
- Use strong, peer reviewed, industry security technologies
- Push identity to back-end servers for application authorization

Web Application Security Goals

- Design and build security into my web application from day 1
- Meet OWASP web application security guidelines
- Force all client access through 1 strong perimeter IdM (Identity Management) security process
    - Authentication
    - Authorization
    - Session management
- Use strong, peer reviewed, industry security technologies
- Push identity to back-end servers for application authorization
- Separate user accounts for local & internet access

General Web application Security

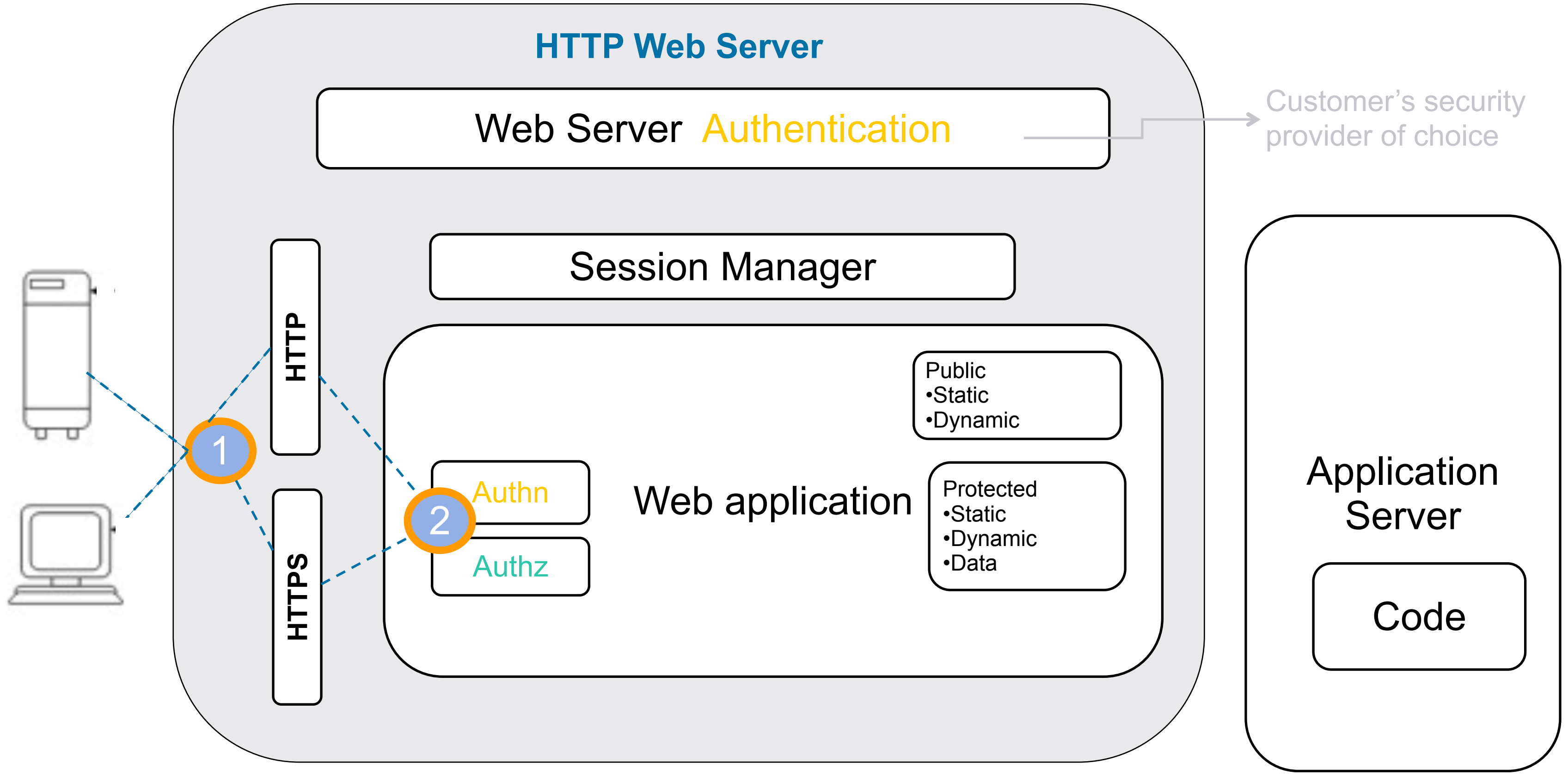
- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



Monday, June 24, 13

General Web application Security

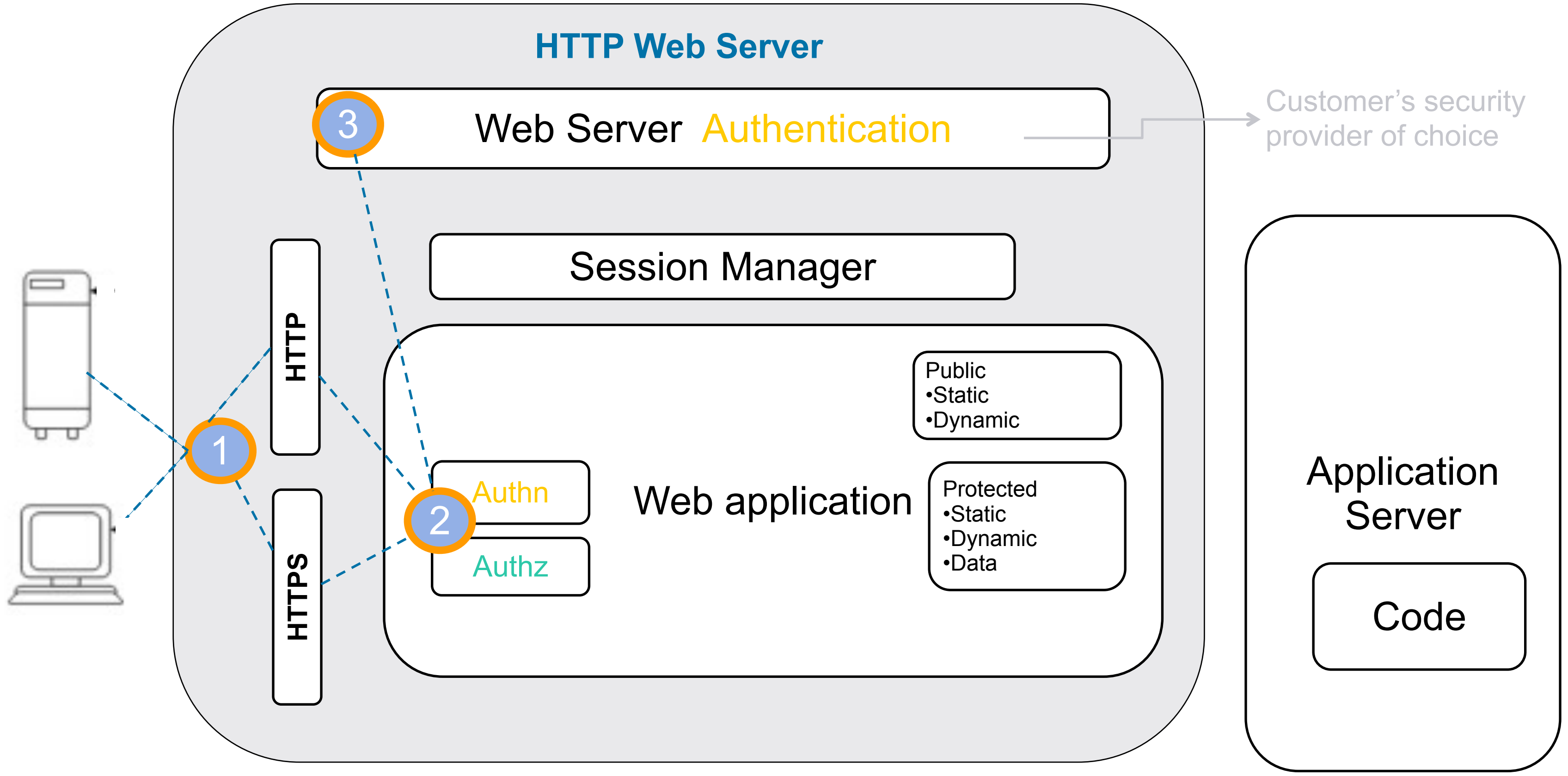
- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



Monday, June 24, 13

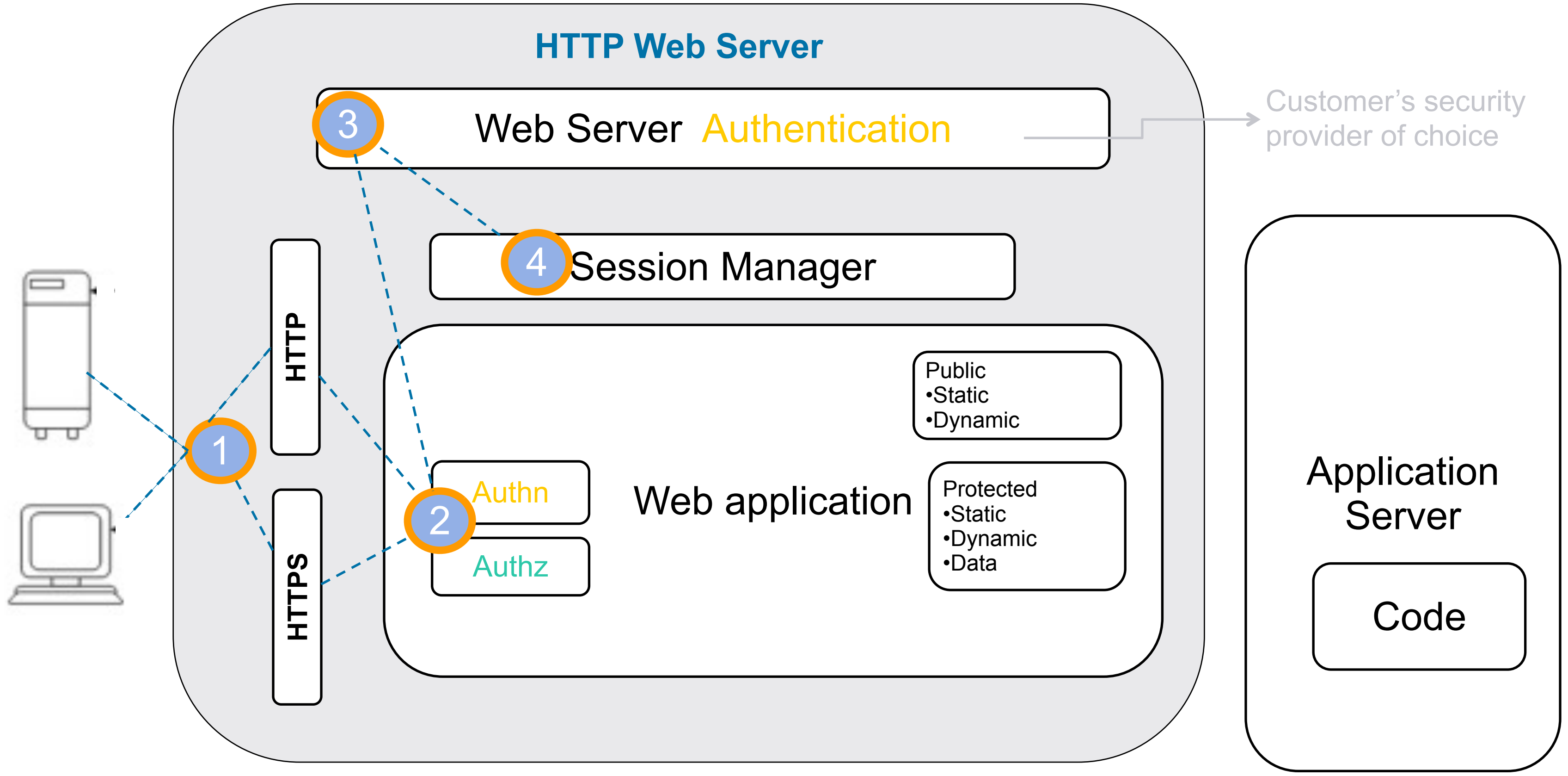
General Web application Security

- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



General Web application Security

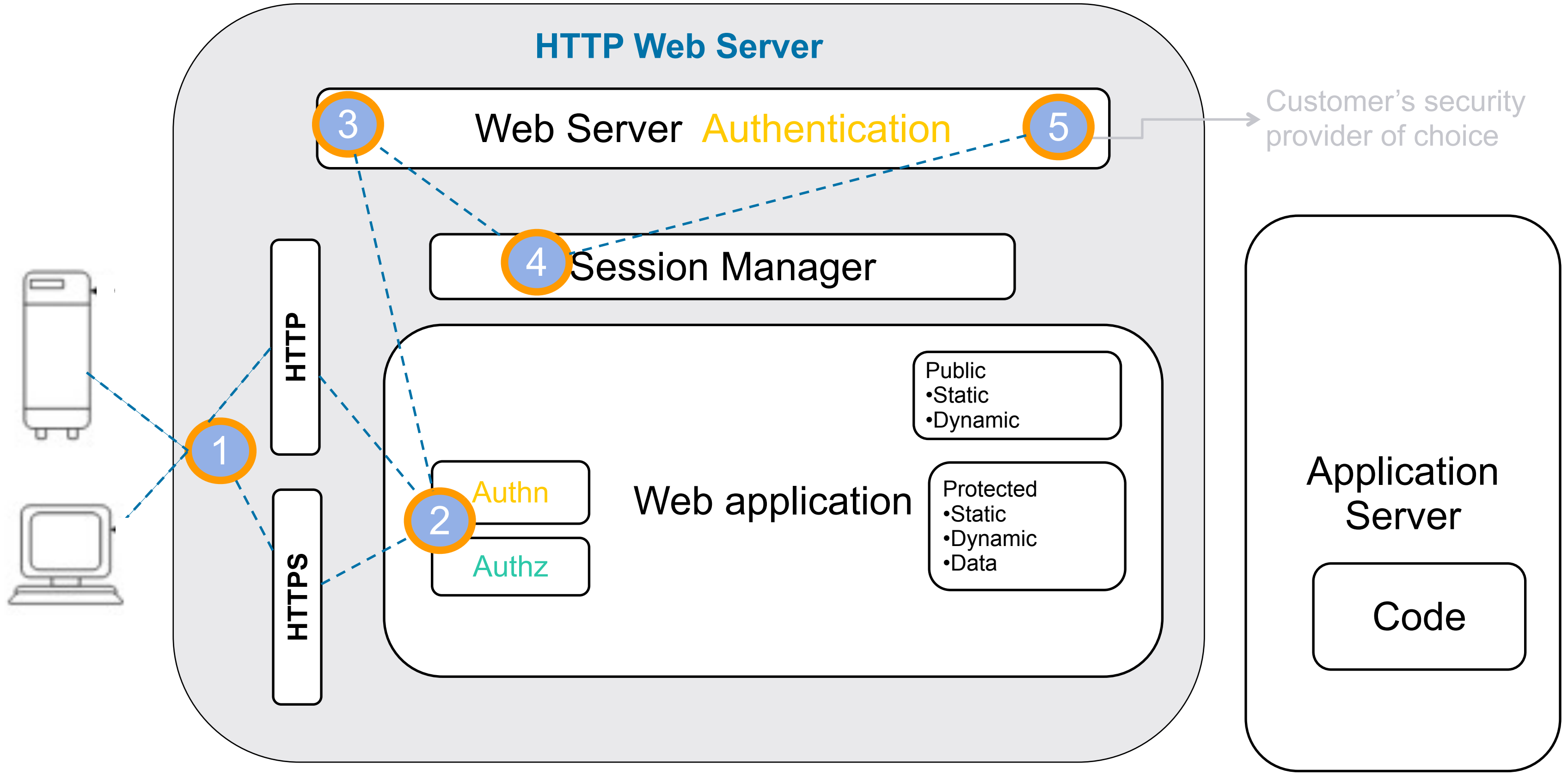
- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



Monday, June 24, 13

General Web application Security

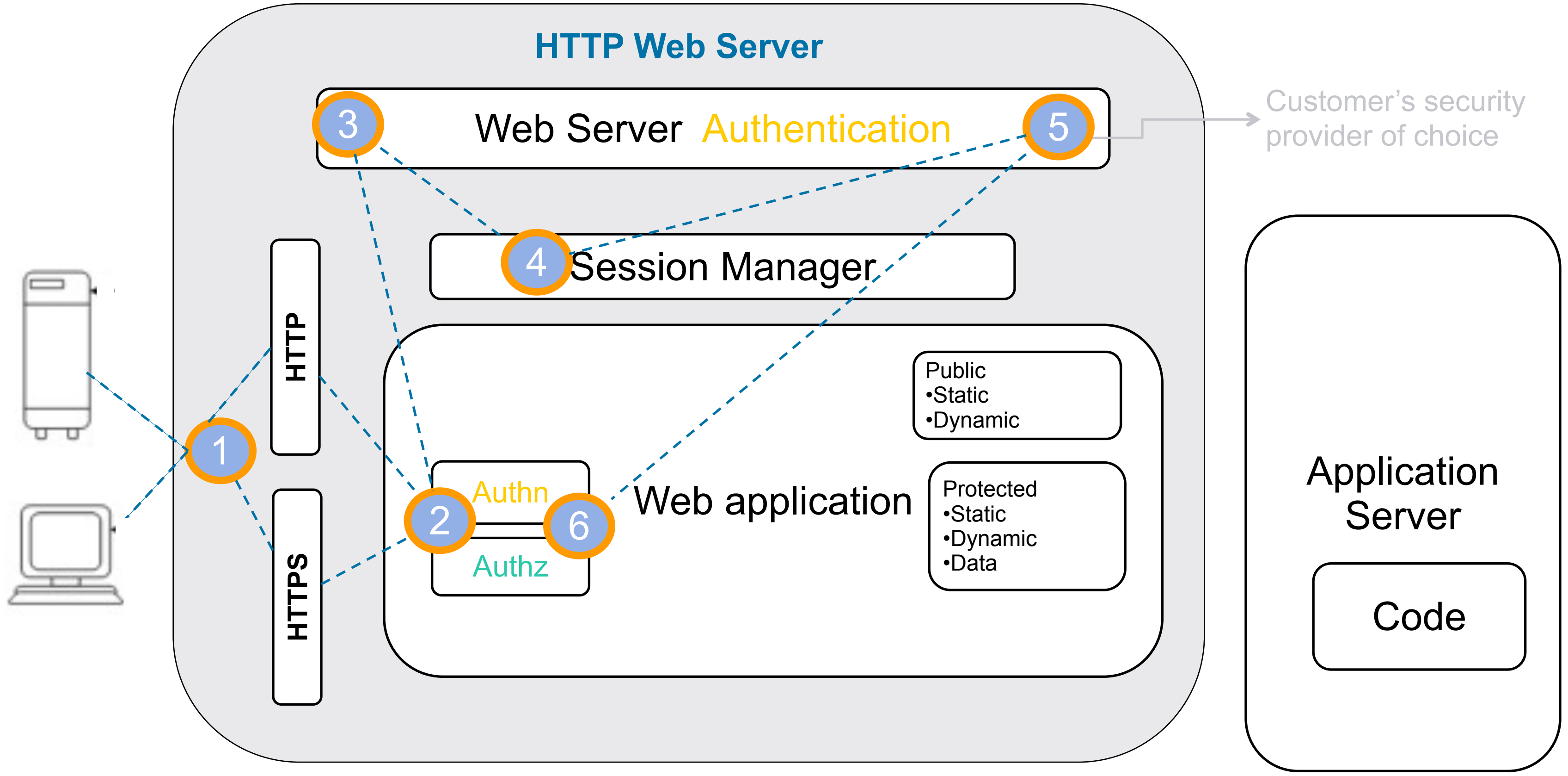
- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



Monday, June 24, 13

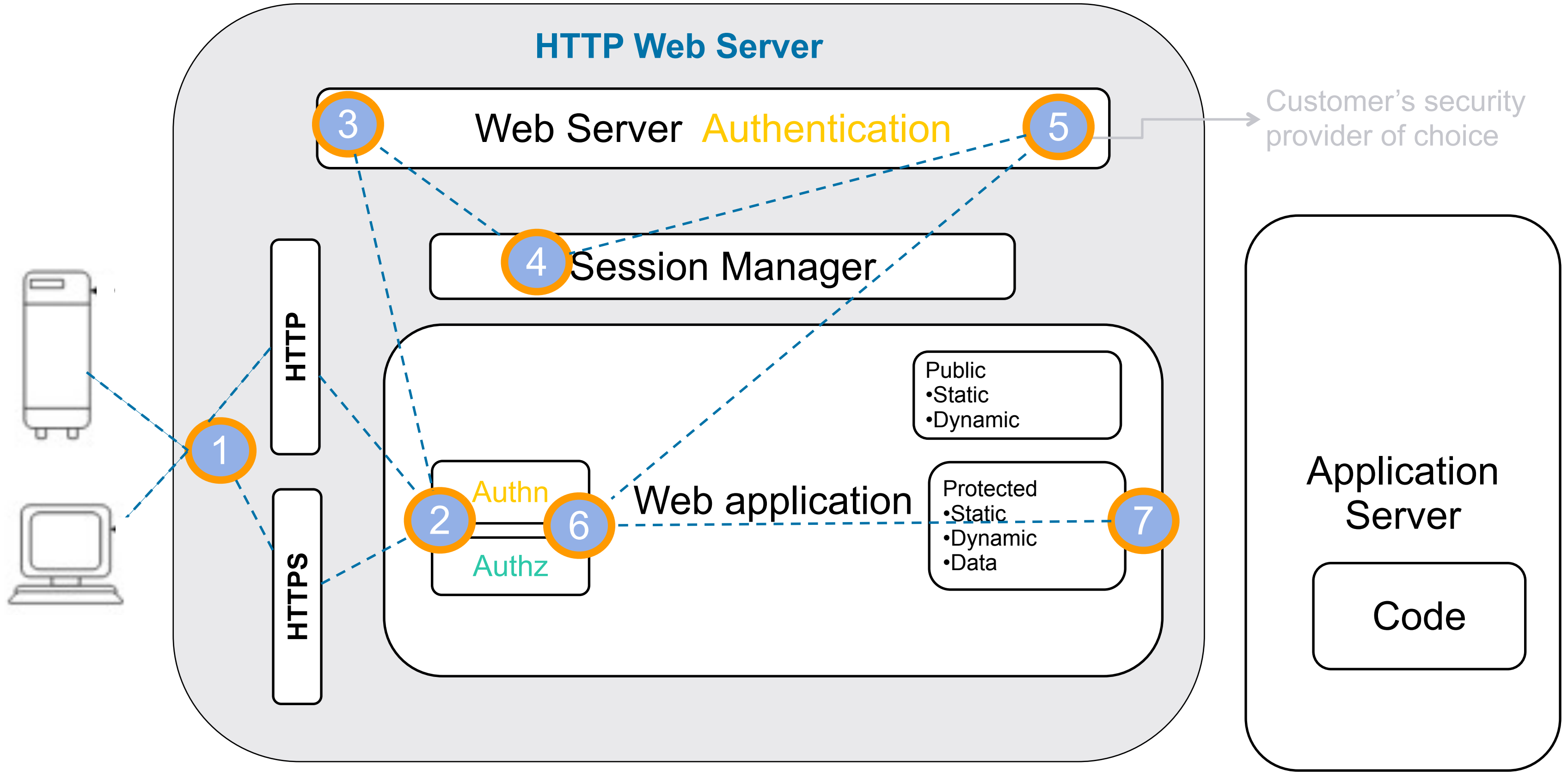
General Web application Security

- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



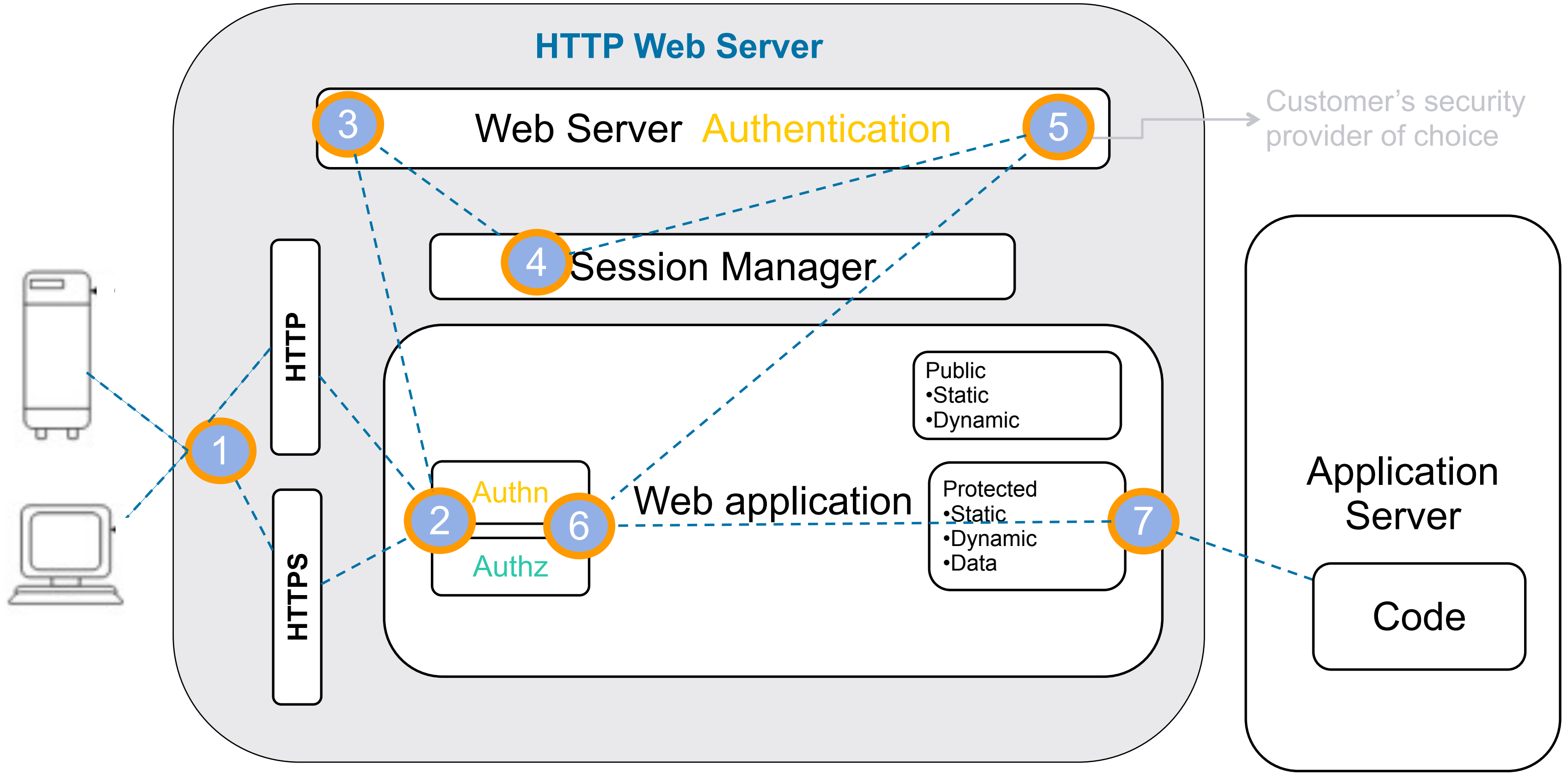
General Web application Security

- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



General Web application Security

- This diagram shows a typical web application authentication and authorization journey
- It is not specific to OpenEdge it address the industry standards that customers will already being using



Monday, June 24, 13

Some Assembly Required...

- OpenEdge (OE) Web applications provide the starting point for you application's security

Some Assembly Required...

- OpenEdge (OE) Web applications provide the starting point for you application’s security
- You will contribute to these layers

1. Data-in-transit security provided by the SSL/TLS for web application’s client to web server, and from web application to an AppServer
2. Web application session management
3. Java container or Web application authentication
4. Web application authorization to HTTP resources
5. OpenEdge AppServer for application level authorization

Some Assembly Required...

- OpenEdge (OE) Web applications provide the starting point for you application's security
- You will contribute to these layers
  1. Data-in-transit security provided by the SSL/TLS for web application's client to web server, and from web application to an AppServer
  2. Web application session management
  3. Java container or Web application authentication
  4. Web application authorization to HTTP resources
  5. OpenEdge AppServer for application level authorization
- Bottom line: OpenEdge's goal is to provide a developer and production administrator with the ability to configure ***their*** Web application's security to suit their needs

Securing an OpenEdge Web Application

- Design stage
  - ✓ Choose web application Authentication & Session model(s)
  - ✓ Design web application RBA (Role Based Access)
  - ✓ Choose single/multiple web application and service architecture

Securing an OpenEdge Web Application

- Design stage
  - ✓ Choose web application Authentication & Session model(s)
  - ✓ Design web application RBA (Role Based Access)
  - ✓ Choose single/multiple web application and service architecture
- Implementation stage
  - ✓ Configure web application Authentication & Session models
  - ✓ Code Authentication & Session models into the client
  - ✓ Configure additional web application Authorization controls (optional)
  - ✓ Code AppServer SSO & authorization (optional)

Securing an OpenEdge Web Application

- Design stage
  - ✓ Choose web application Authentication & Session model(s)
  - ✓ Design web application RBA (Role Based Access)
  - ✓ Choose single/multiple web application and service architecture
- Implementation stage
  - ✓ Configure web application Authentication & Session models
  - ✓ Code Authentication & Session models into the client
  - ✓ Configure additional web application Authorization controls (optional)
  - ✓ Code AppServer SSO & authorization (optional)
- Deployment stage (+ unit testing)
  - ✓ Configure web application source of user accounts
  - ✓ Reconfigure RBA role names to match user account source
  - ✓ Configure web application CORS (for mobile JavaScript clients)

Securing an OpenEdge Web Application

- Design stage
  - ✓ Choose web application Authentication & Session model(s)
  - ✓ Design web application RBA (Role Based Access)
  - ✓ Choose single/multiple web application and service architecture
- Implementation stage
  - ✓ Configure web application Authentication & Session models
  - ✓ Code Authentication & Session models into the client
  - ✓ Configure additional web application Authorization controls (optional)
  - ✓ Code AppServer SSO & authorization (optional)
- Deployment stage (+ unit testing)
  - ✓ Configure web application source of user accounts
  - ✓ Reconfigure RBA role names to match user account source
  - ✓ Configure web application CORS (for mobile JavaScript clients)

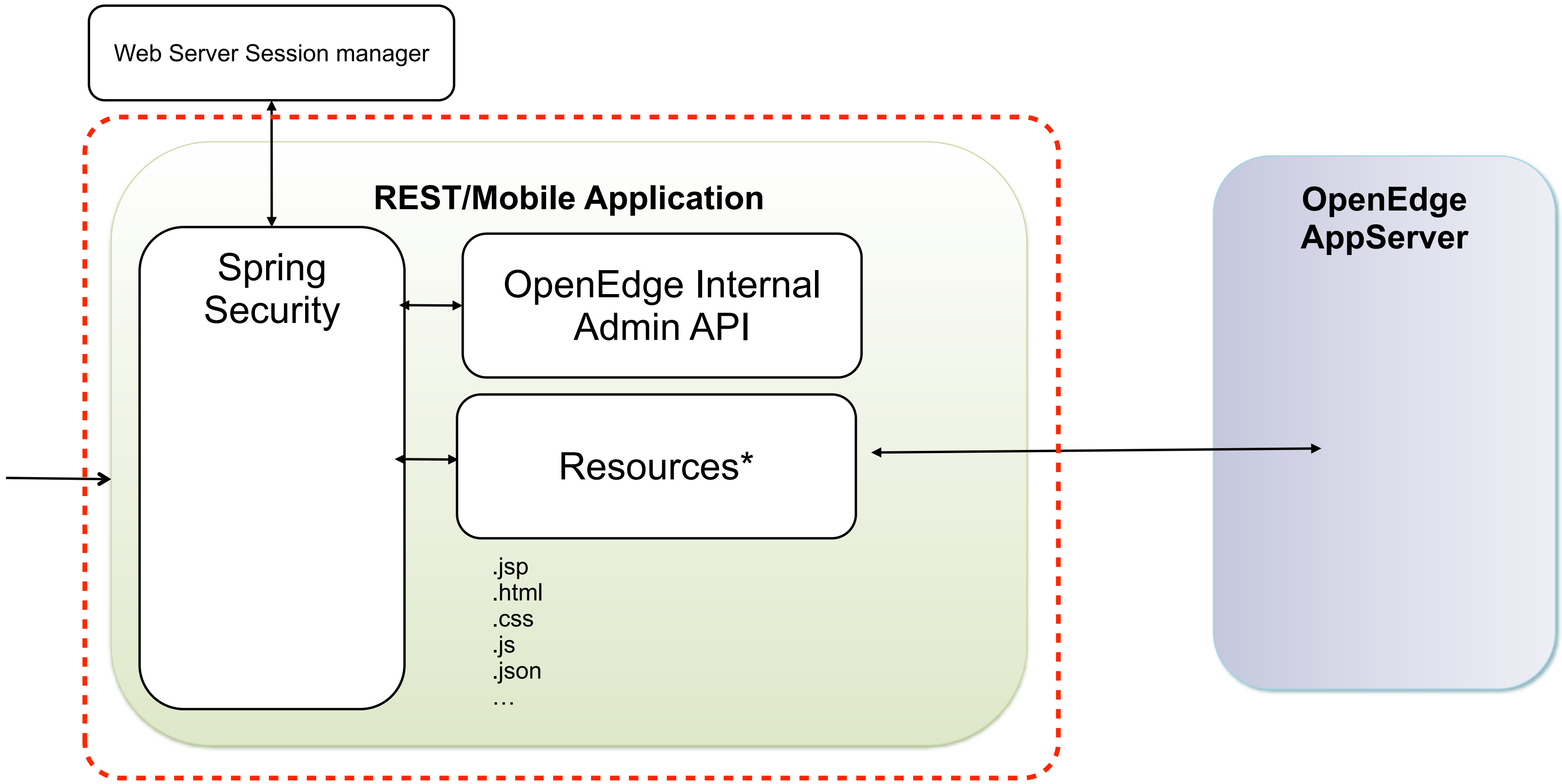
Spring Security Framework version 3.1

- OpenEdge supplements the Java container's security with the industry-recognized Spring security framework
- Spring security is a customizable authentication and access-control framework
- It is one of the industry standards for securing Spring-based applications

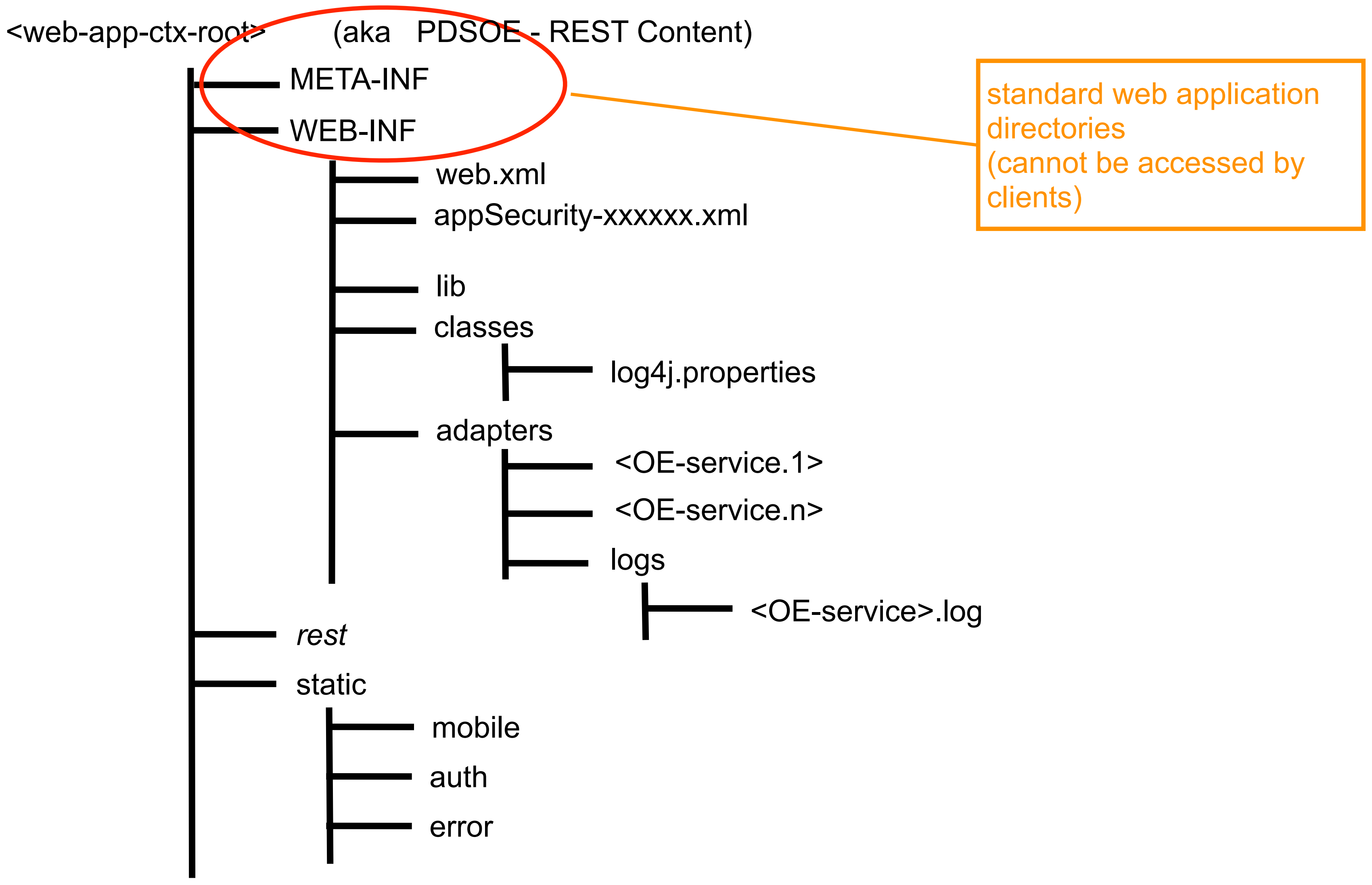
springsource

  - For more information on Spring security framework, see the http://static.springsource.org/spring-security/site/reference.html

Anatomy of an OpenEdge Web Application

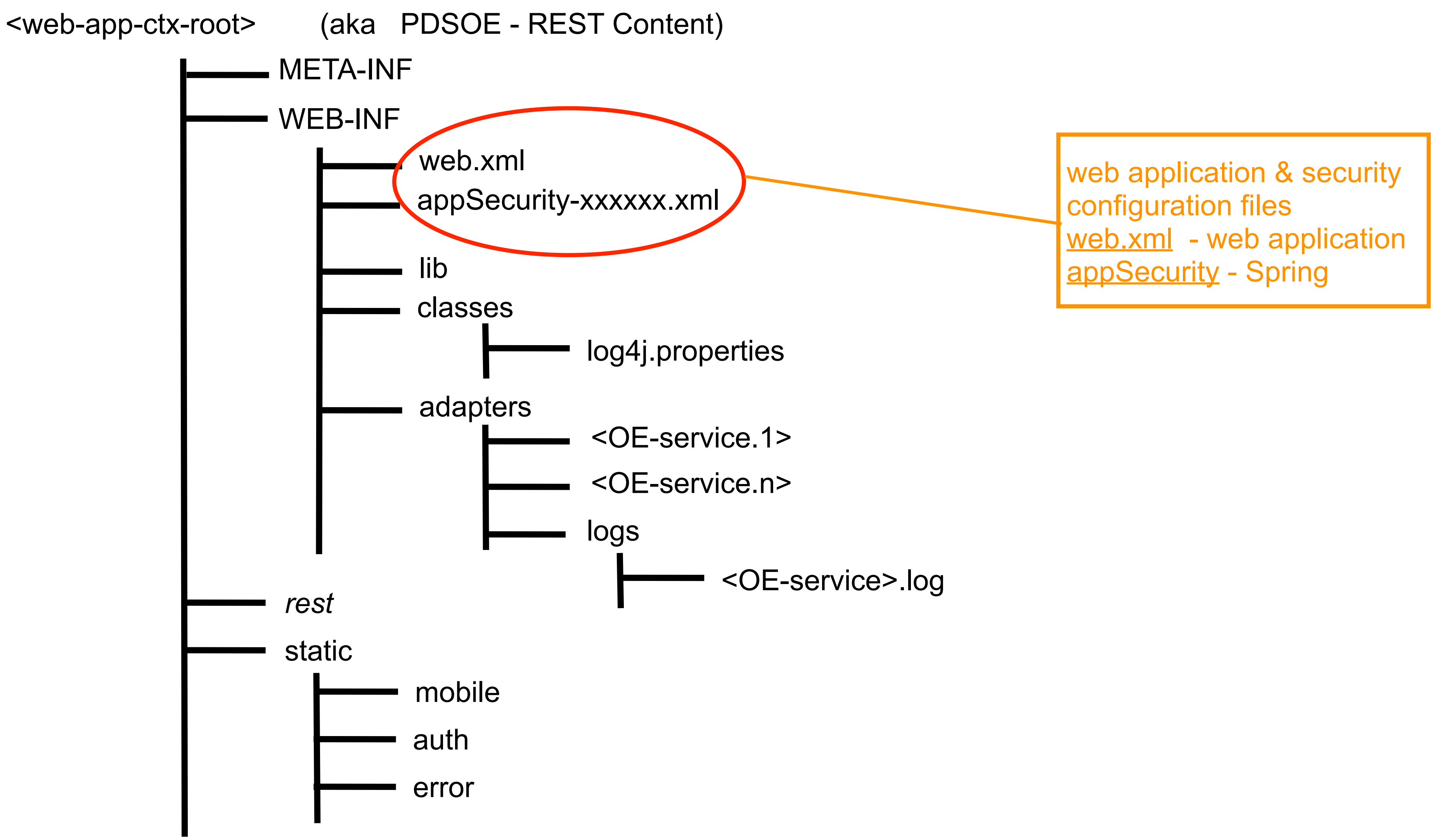


Anatomy of an OpenEdge Web Application (1 of 9)

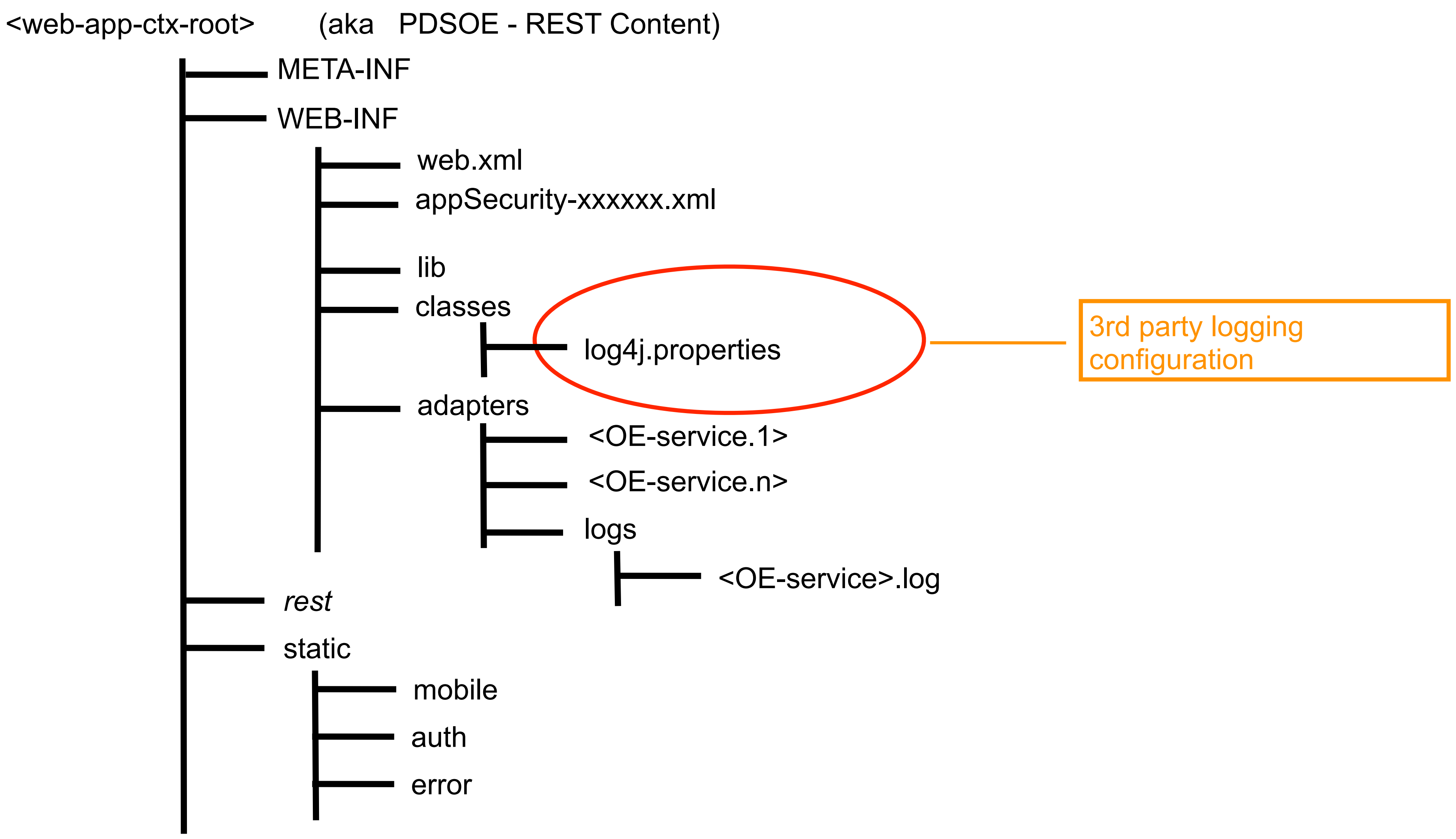


Monday, June 24, 13

Anatomy of an OpenEdge Web Application (2 of 9)

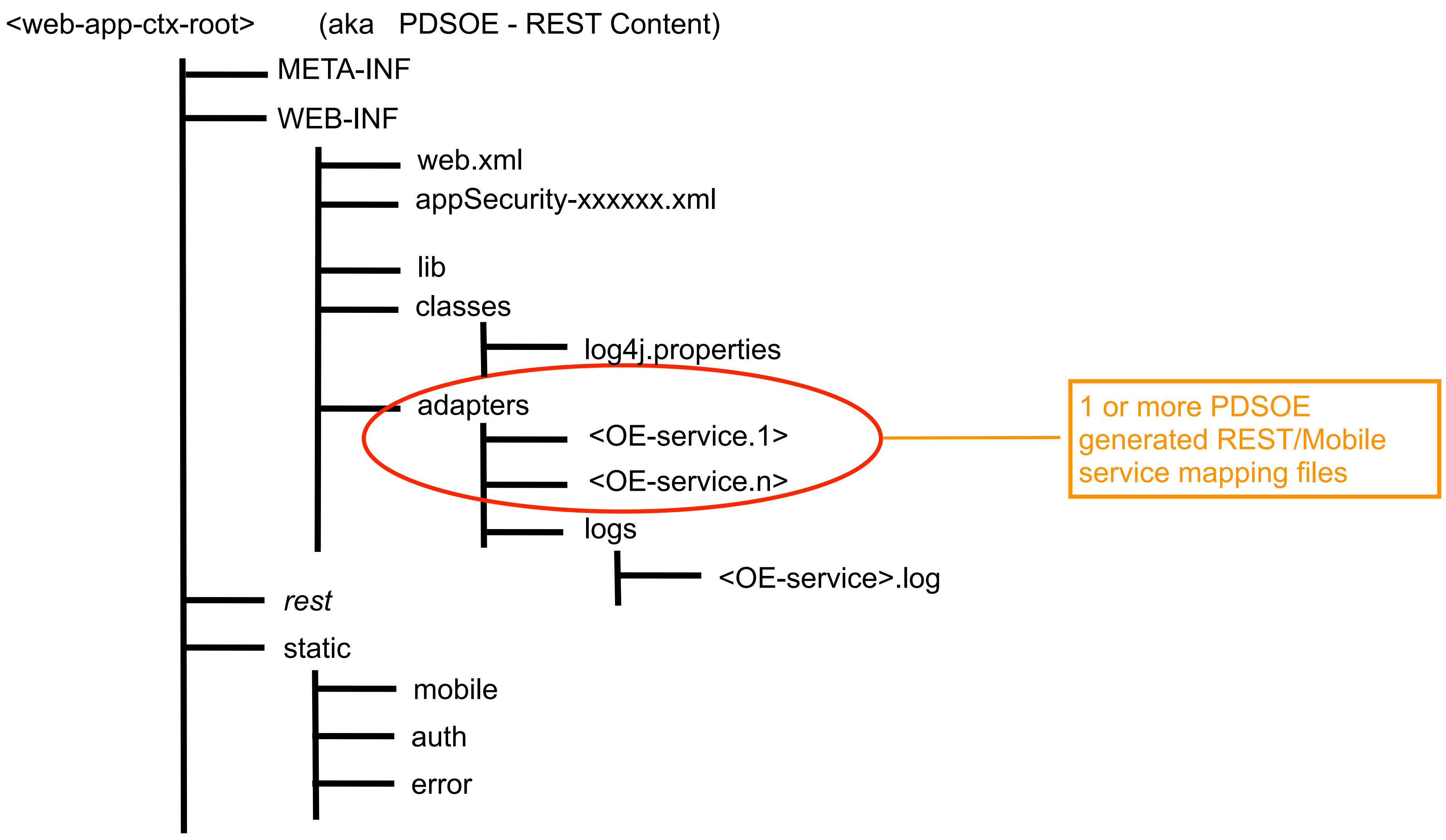


Anatomy of an OpenEdge Web Application (3 of 9)



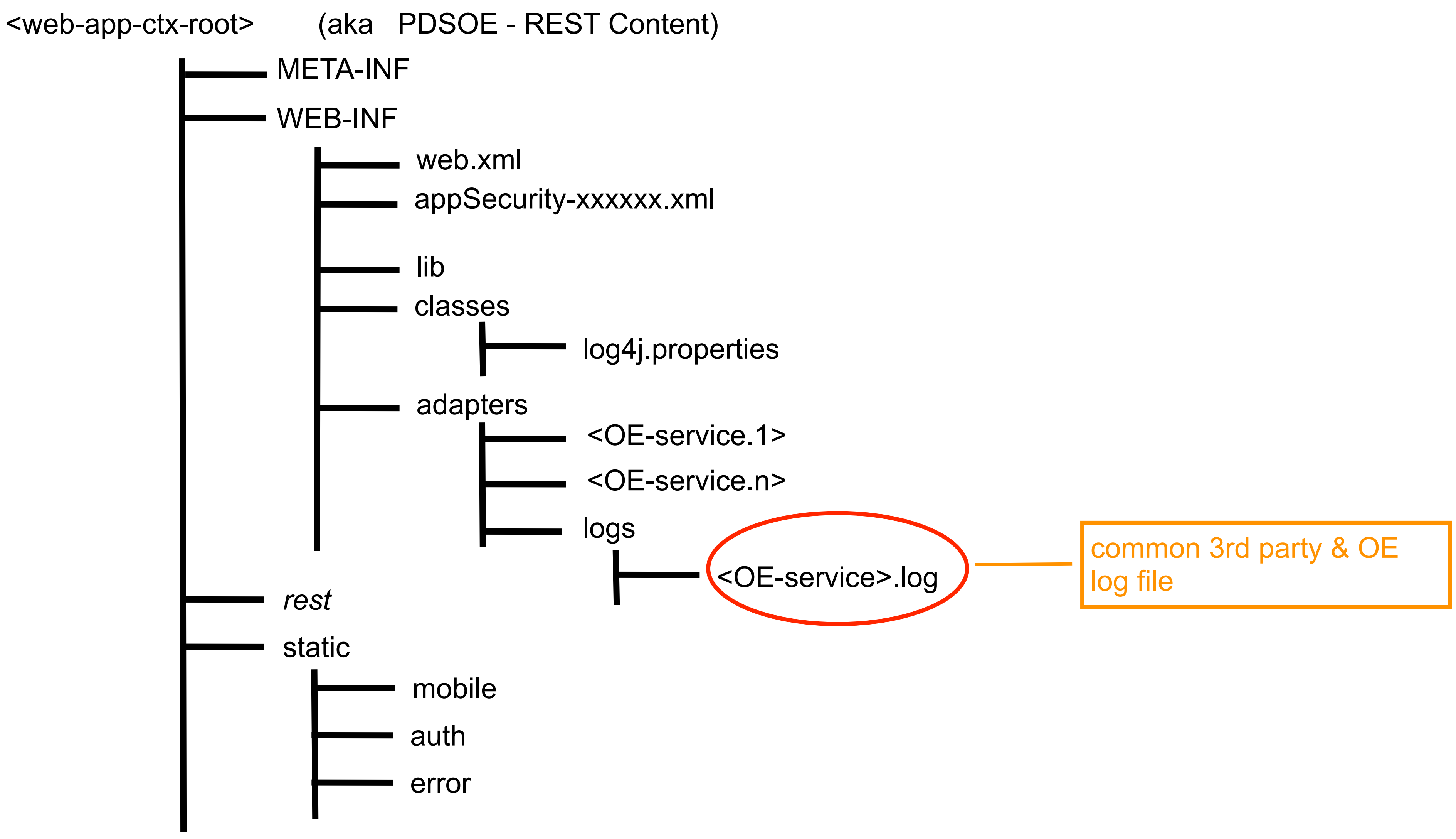
Monday, June 24, 13

Anatomy of an OpenEdge Web Application (4 of 9)

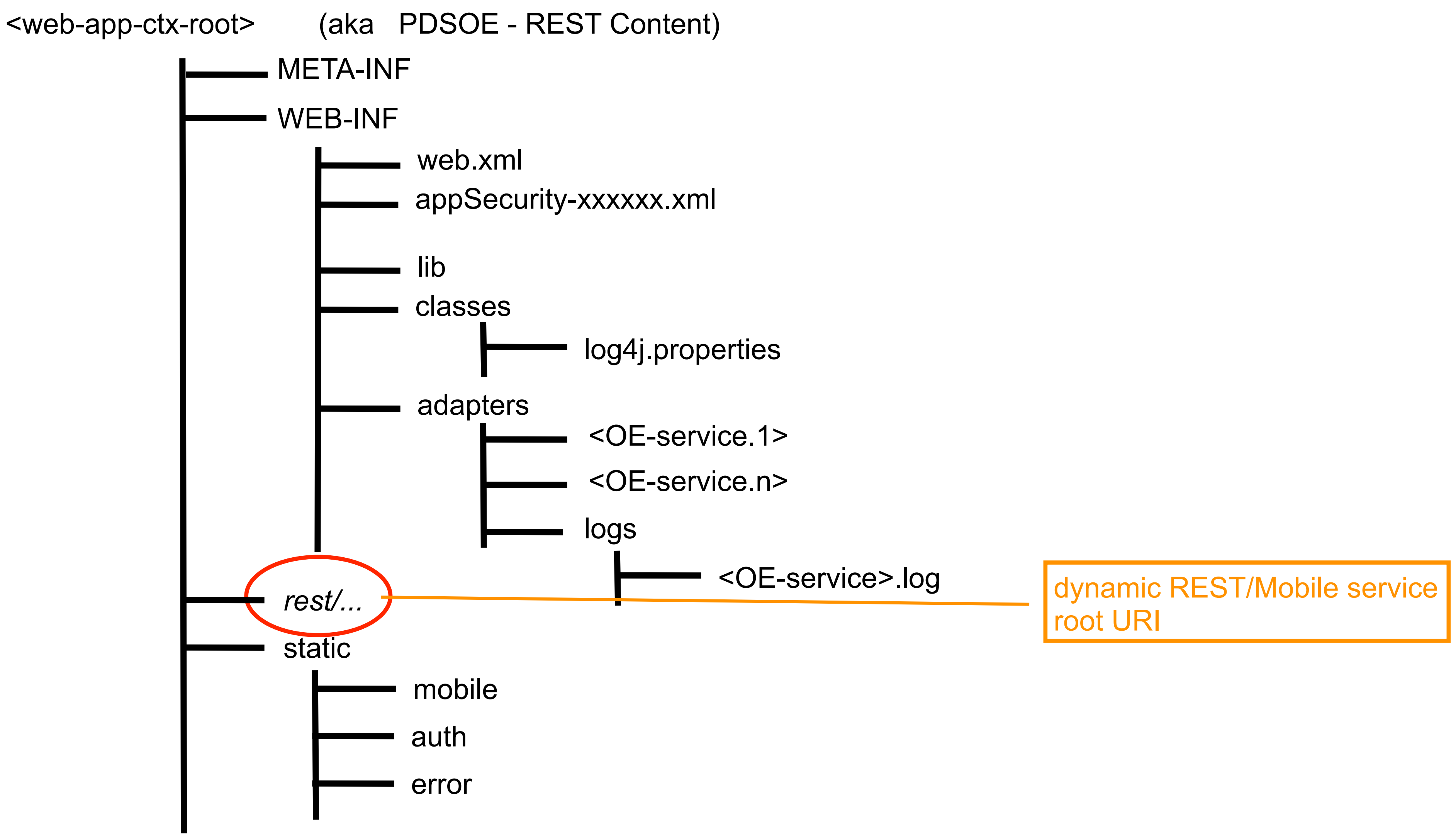


Monday, June 24, 13

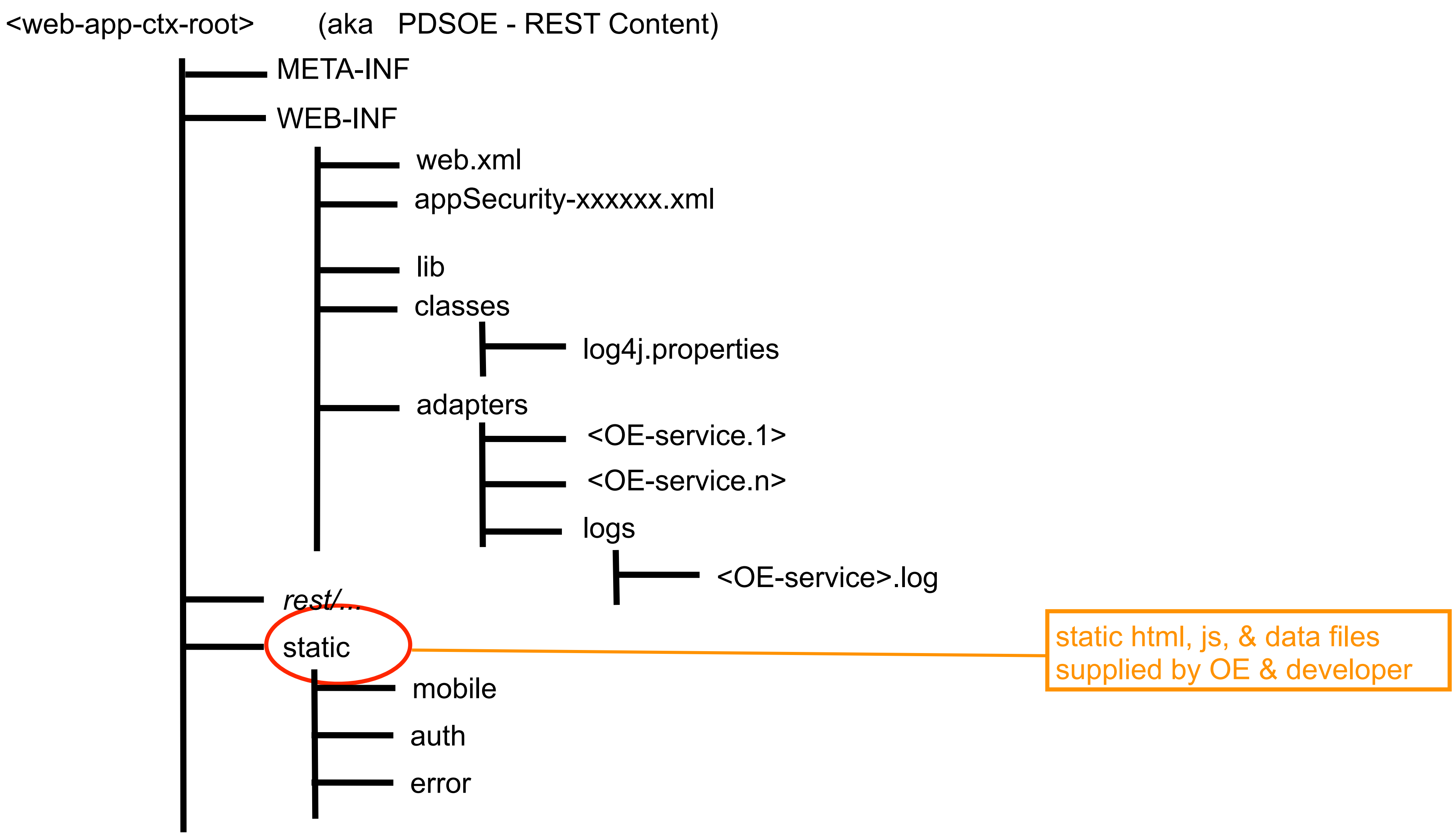
Anatomy of an OpenEdge Web Application (5 of 9)



Anatomy of an OpenEdge Web Application (6 of 9)



Anatomy of an OpenEdge Web Application (7 of 9)



Monday, June 24, 13

Anatomy of an OpenEdge Web Application (8 of 9)

```
<web-app-ctx-root>      (aka   PDSOE - REST Content)
    ├── META-INF
    ├── WEB-INF
    │     ├── web.xml
    │     ├── appSecurity-xxxxxx.xml
    │     ├── lib
    │     ├── classes
    │     │     └── log4j.properties
    │     └── adapters
    │           ├── <OE-service.1>
    │           ├── <OE-service.n>
    │           └── logs
    │                 └── <OE-service>.log
    ├── rest/...
    └── static
          ├── mobile
          ├── auth
          └── error
```

static html & java script for browser UI + catalog for Mobile service

Anatomy of an OpenEdge Web Application (9 of 9)

```
<web-app-ctx-root>      (aka   PDSOE - REST Content)
    ├── META-INF
    ├── WEB-INF
    │     ├── web.xml
    │     ├── appSecurity-xxxxxx.xml
    │     │
    │     ├── lib
    │     ├── classes
    │     │     └── log4j.properties
    │     └── adapters
    │           ├── <OE-service.1>
    │           ├── <OE-service.n>
    │           └── logs
    │                 └── <OE-service>.log
    ├── rest/...
    └── static
          ├── mobile
          ├── auth
          └── error
```

OE supplied login/logout and error pages for testing (need to be replaced and added to by developer)

Monday, June 24, 13

Choose your Authentication Model

- OpenEdge supplies security template files for various Authentication and user account source combinations
- Default [anonymous] security configuration is development time only and has no authentication/authorization
- Web application and client need to share the same model chosen at development time
- Security template file names are in the format

appSecurity-<*authn-model*>[-<*provider*>].xml

Web application authentication models *

- **Anonymous** — The *no user authentication or login session*
- **HTTP Basic Authentication** — *Client sends base64 encoded user name/password to web application in each http request*
  - HTTP header: *Authorization*
- ***HTTP Form Authentication** — The client logs into and out of the web application once per session*
  - Login: *The client obtains user credentials and POSTs them to the web application*
    - *URI: /static/auth/j_spring_security_check*
    - *Body: j_username=xxxx&j_password=yyyy&submit=Submit+Query*
    - *Cookie: JSESSIONID*
  - Logout: *The client uses a GET request to log out*
    - *URI: /static/auth/j_spring_security_logout*

* Other authentication models available - not certified

Common web application authentication providers *

- Supplies user account information to common Spring authentication process
  - Name; password; roles; state; locked; expired; ...
- Can change provider type at production deployment time
- Provider choices
  - In-memory — user accounts in configuration file
  - Local file — user accounts in clear-text file
  - Container — user account authenticated by Java container
  - LDAP / AD — user accounts in a *Directory Service* *(11.2.1)*
  - OERealm — OpenEdge AppServer service *(11.3)*

* Other authentication providers available - not certified

Choose your user login session model

- Client and server configurations must agree
- Sessions controlled by Web Server's session manager (**not** OE web application or your AppServer)
- Session ID stored in cookie & sent to client - client returns the cookie with EACH request (JSESSIONID cookie)
- Session ID cookies cannot be seen by client JavaScript
- Login sessions timeout - default 30 minutes
- Login sessions are not automatically shared across multiple web applications
  - PDSOE wants to deploy Mobile UI and its service as TWO different OE web applications

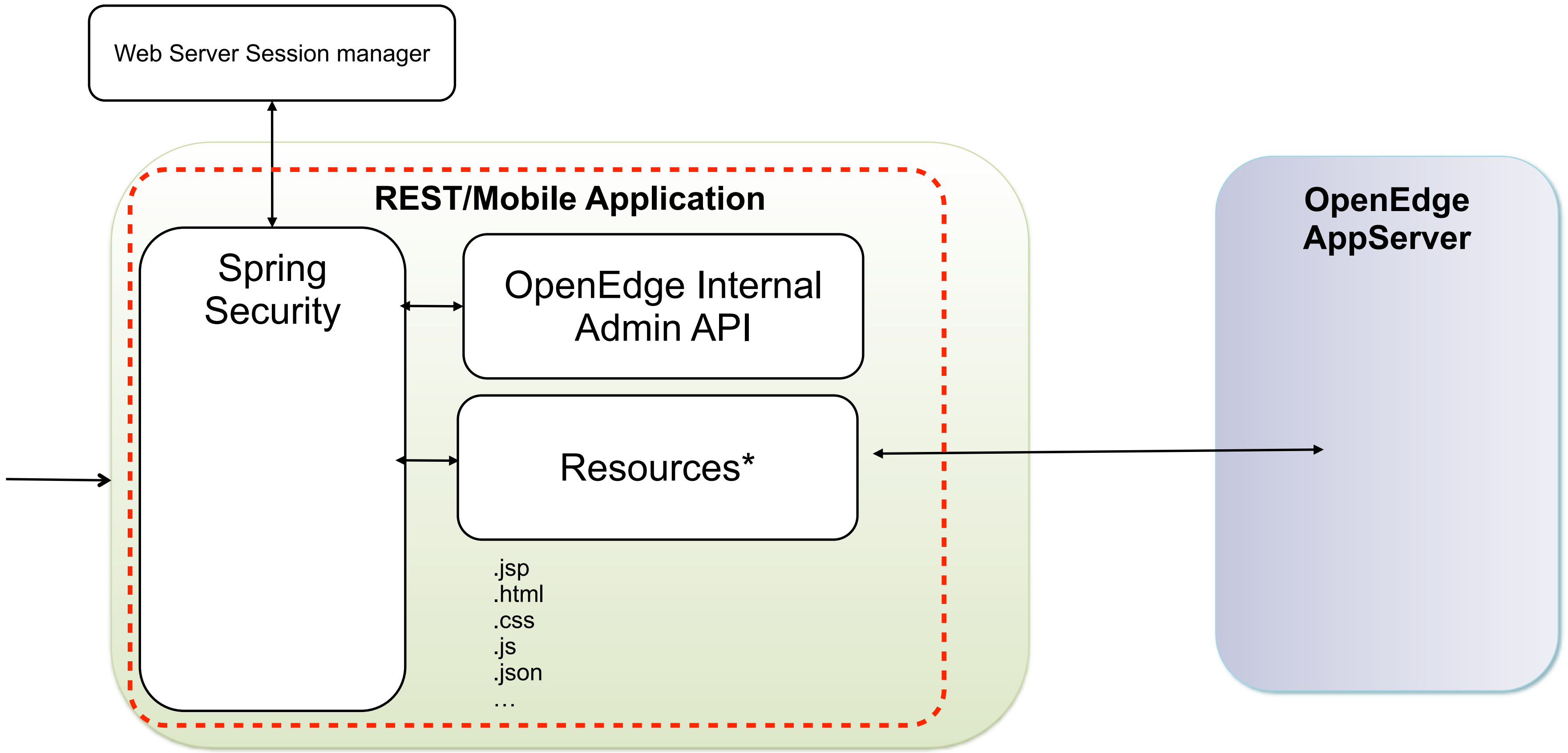
Plan for Role Based Access (RBA)

- All of Spring authorization uses ROLES
- You need to configure at least one role
- Add additional roles appropriate for your application's security
- The user's roles are obtained from the same location as the user account information
- If no user authentication is performed for the request Spring assigns a default user account
  - anonymousUser
  - ROLE_ANONYMOUS

Choose Your Application Architecture

- The number of web applications affects your client
- Single web application
  - One user login - one large set of access controls
  - Cross Site Resource access not a problem
  - Everything enabled/disabled at one time
- Multiple web applications
  - Multiple user logins - multiple sets of access controls
  - Cross Site Resource access configured everywhere
  - Some applications enabled, others disabled

Anatomy of an Web Application Configuration



Web Application Configuration: web.xml (1 of 4)

<context-param />

contextConfiguration parameter for choosing Spring security template

<listener>

<filter>

<filter-mapping>

<servlet>

<session-config>

<security-constraints>

<welcome-file-list>

<error-page>

Web Application Configuration: web.xml (2 of 4)

<context-param />

<listener>
<filter>
<filter-mapping>
<servlet>

OE web application declarations
DO NOT EDIT

<session-config>

<security-constraints>
<welcome-file-list>
<error-page>

Web Application Configuration: web.xml (3 of 4)

<context-param />

<listener>

<filter>

<filter-mapping>

<servlet>

<session-config>

Edit web server user-session timeouts, http-only cookies, etc

<security-constraints>

<welcome-file-list>

<error-page>

Web Application Configuration: web.xml (4 of 4)

<context-param />

<listener>

<filter>

<filter-mapping>

<servlet>

<session-config>

<security-constraints>

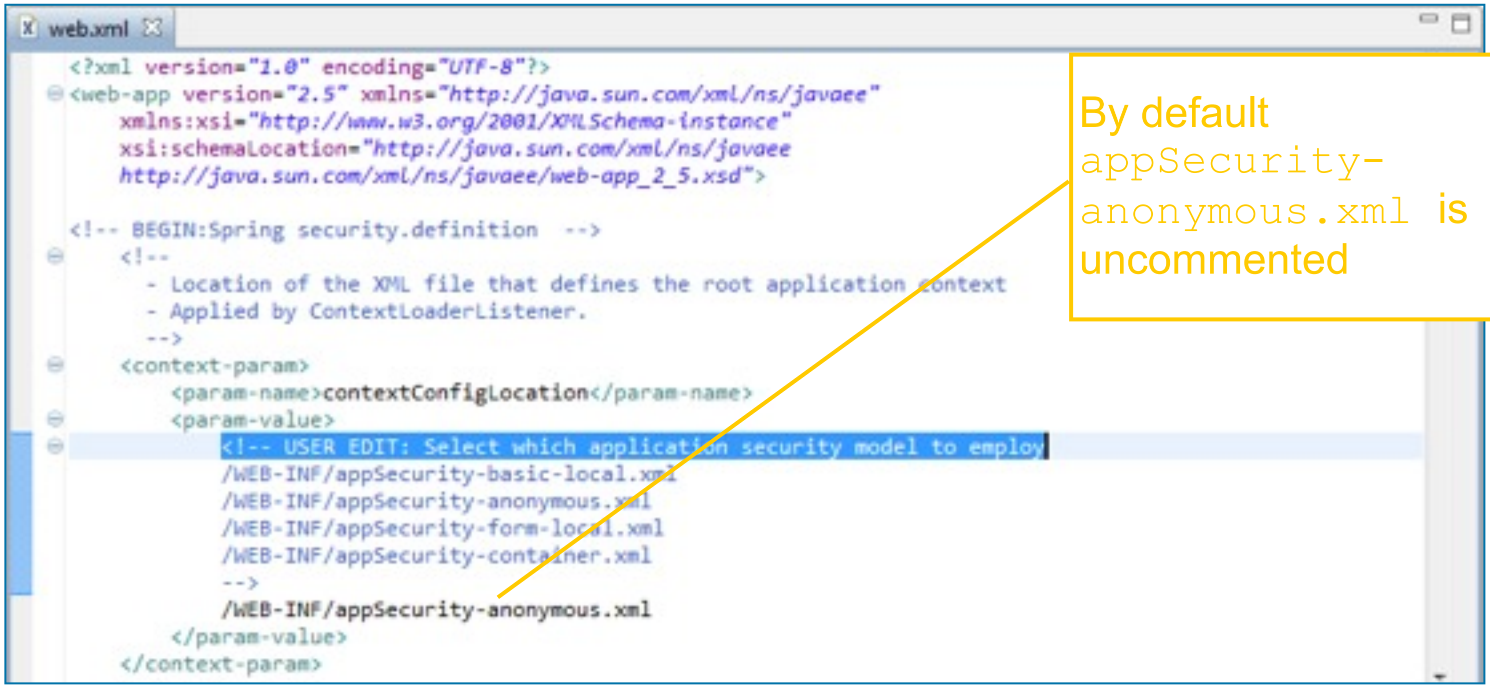
<welcome-file-list>

<error-page>

Edit web server level security for “container” Spring authn model

Example: Choosing the Spring Security template

- You edit the `web.xml` file to set the security configuration
  - Default location
    - `C:\Progress\OpenEdge\rest\server\WEB-INF`
  - See `param-values` in the <!--USER EDIT section for `contextConfigLocation`

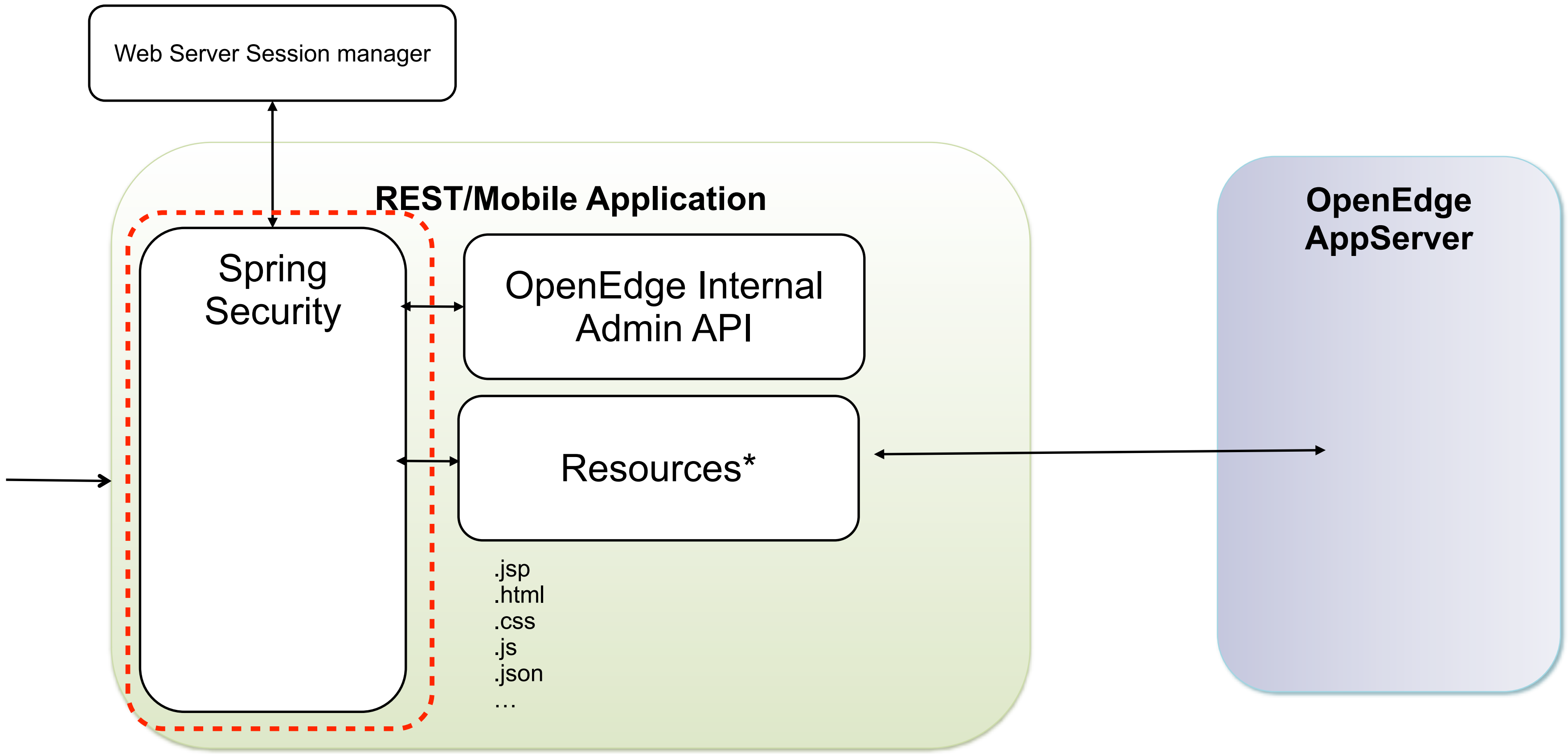


```
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
    http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd">

<!-- BEGIN:Spring security.definition  -->
    <!--
      - Location of the XML file that defines the root application context
      - Applied by ContextLoaderListener.
      -->
    <context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
            <!-- USER EDIT: Select which application security model to employ
            /WEB-INF/appSecurity-basic-local.xml
            /WEB-INF/appSecurity-anonymous.xml
            /WEB-INF/appSecurity-form-local.xml
            /WEB-INF/appSecurity-container.xml
            -->
            /WEB-INF/appSecurity-anonymous.xml
        </param-value>
    </context-param>
```

By default `appSecurity-anonymous.xml` is uncommented

Monday, June 24, 13

Anatomy of an Spring Security Configuration



Spring Security Configuration File Tour (1 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

contains the authn & authz process controls like session, realm, ... for a specific application URI space

Spring Security Configuration File Tour (2 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

OpenEdge additions to Spring security process

Spring Security Configuration File Tour (2 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

URI access controls:
what, how, & who

Spring Security Configuration File Tour (3 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

controls the authn model implementations - only one in effect at a time

Spring Security Configuration File Tour (4 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

modifies the session management found in <http>

Spring Security Configuration File Tour (5 of 6)

```
<http   “attributes” >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

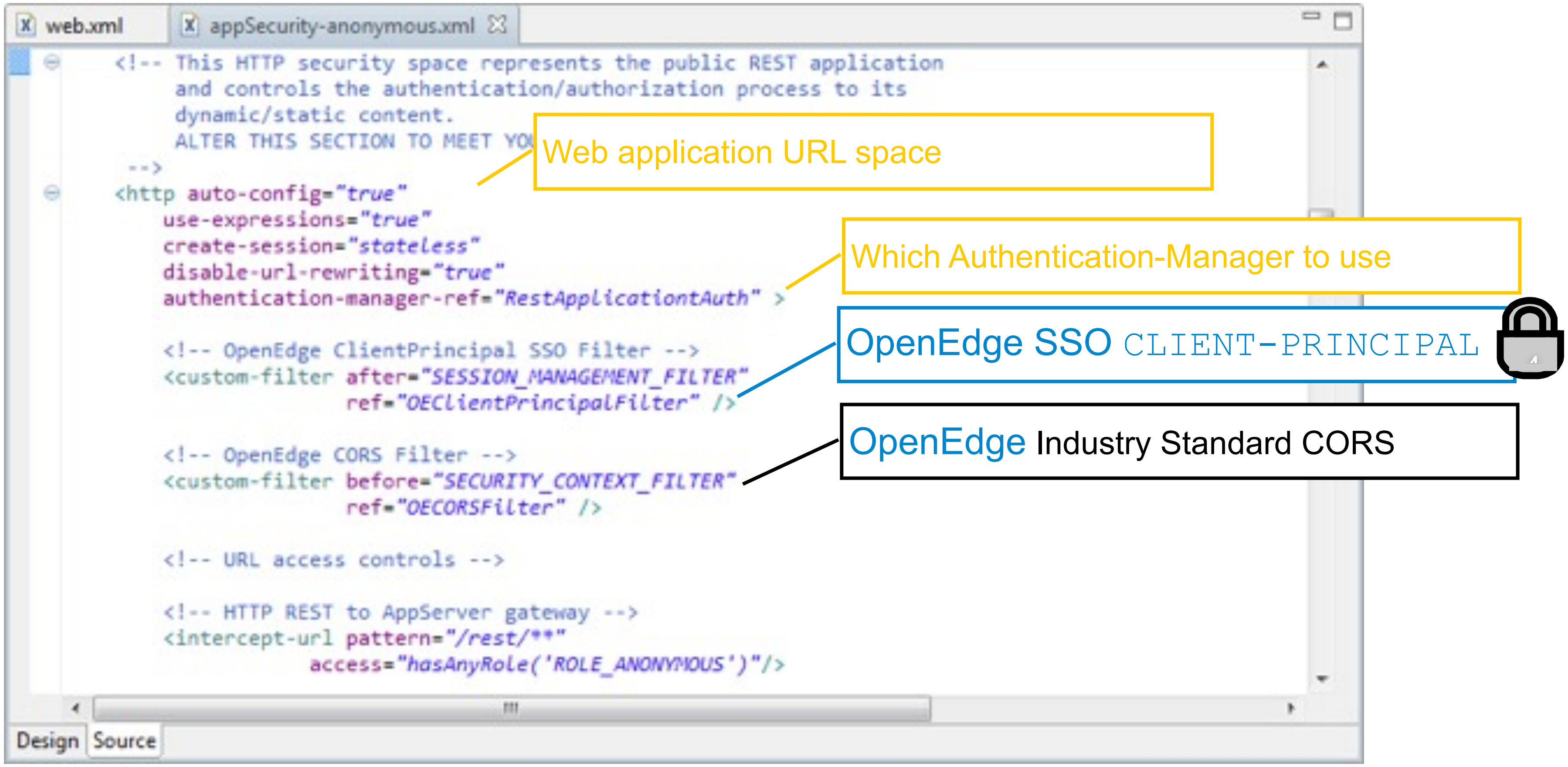
Spring bean plug-ins for a wide range of authn and process related operations

Spring Security Configuration File Tour (6 of 6)

```
<http   "attributes" >
      <custom-filter>
      <intercept-url>
      <basic>
      <form-login>
      <logout>
      <jee>
      <session-management>
</http>

<b:bean>   </b:bean>

<authentication-manager>
      <authentication-provider>
      </authentication-provider>
</authentication-manager>
```

configure how and where user accounts are accessed for Spring's authn process

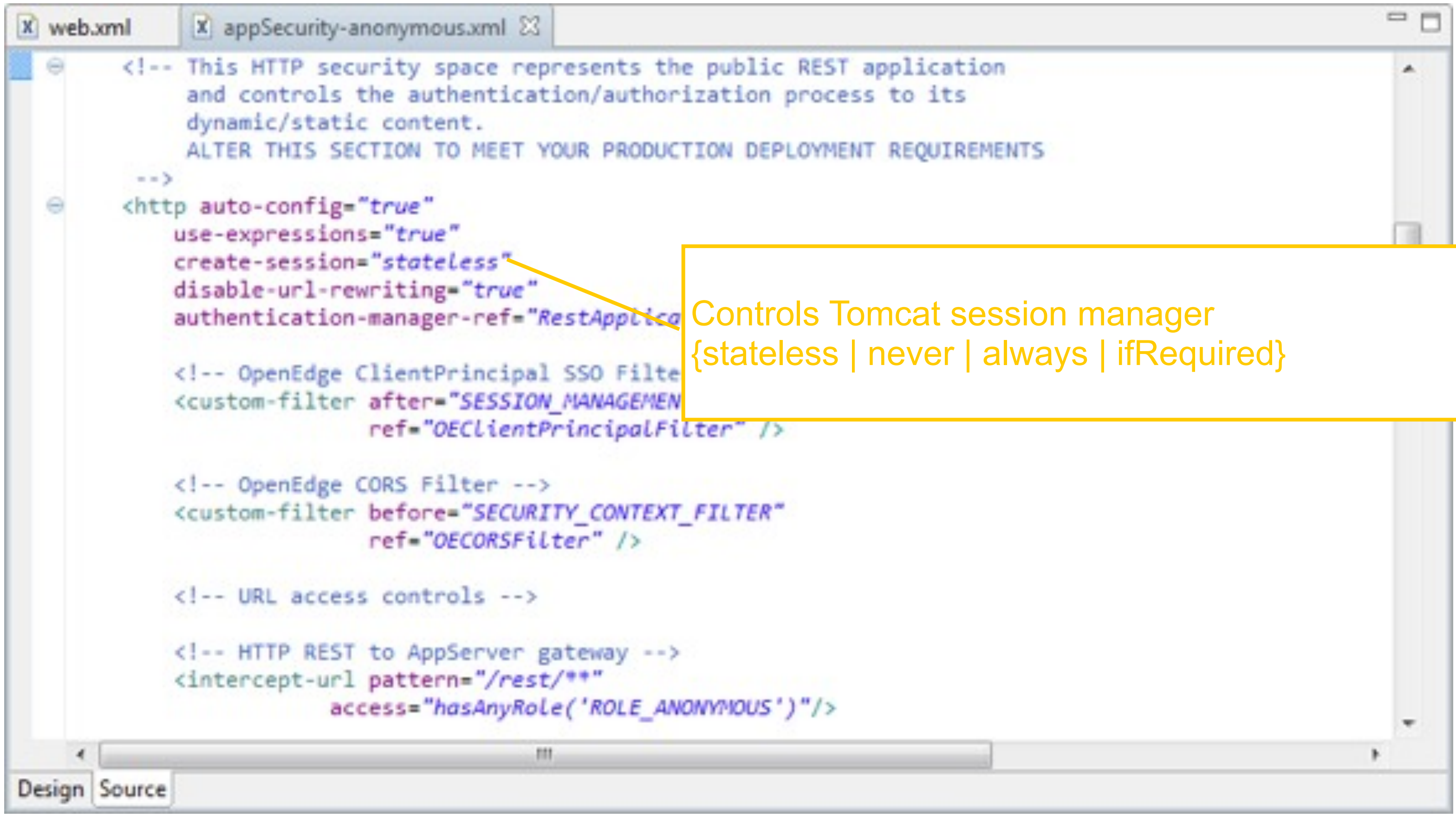
Example: <http> and OpenEdge Spring extensions


web.xml
appSecurity-anonymous.xml
Web application URL space
Which Authentication-Manager to use
OpenEdge SSO CLIENT-PRINCIPAL
OpenEdge Industry Standard CORS
Design
Source

```
<!-- This HTTP security space represents the public REST application
     and controls the authentication/authorization process to its
     dynamic/static content.
     ALTER THIS SECTION TO MEET YO
 -->
<http auto-config="true"
    use-expressions="true"
    create-session="stateless"
    disable-url-rewriting="true"
    authentication-manager-ref="RestApplicationAuth" >

    <!-- OpenEdge ClientPrincipal SSO Filter -->
    <custom-filter after="SESSION_MANAGEMENT_FILTER"
                   ref="OEClientPrincipalFilter" />

    <!-- OpenEdge CORS Filter -->
    <custom-filter before="SECURITY_CONTEXT_FILTER"
                   ref="OECORSFilter" />

    <!-- URL access controls -->

    <!-- HTTP REST to AppServer gateway -->
    <intercept-url pattern="/rest/**"
                access="hasAnyRole('ROLE_ANONYMOUS')"/>
```

Monday, June 24, 13

Example: Choosing the session management model

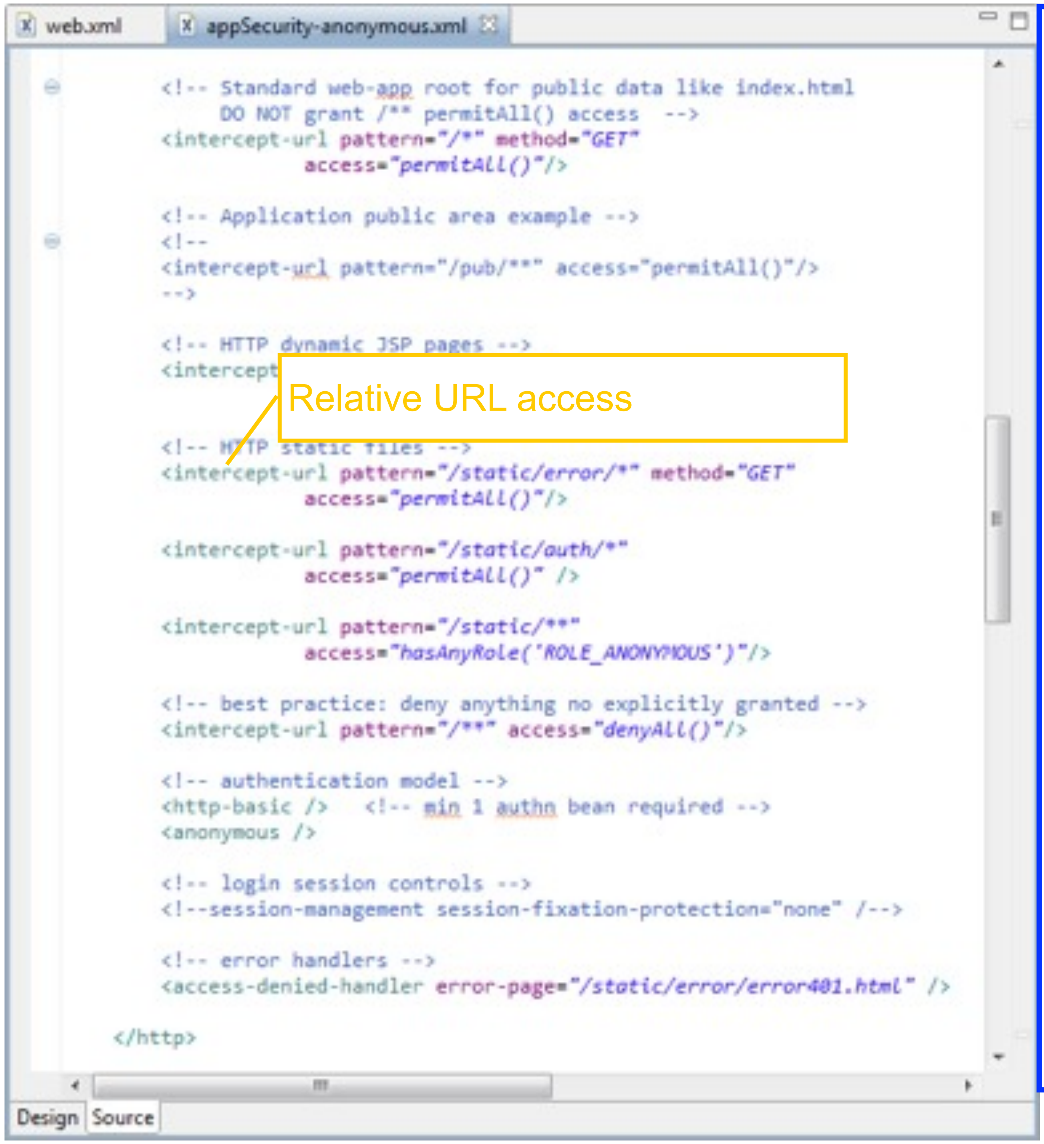


```
<!-- This HTTP security space represents the public REST application
     and controls the authentication/authorization process to its
     dynamic/static content.
     ALTER THIS SECTION TO MEET YOUR PRODUCTION DEPLOYMENT REQUIREMENTS
 -->
<http auto-config="true"
    use-expressions="true"
    create-session="stateless"
    disable-url-rewriting="true"
    authentication-manager-ref="RestApplica

    <!-- OpenEdge ClientPrincipal SSO Filte
    <custom-filter after="SESSION_MANAGEMEN
                   ref="OEClientPrincipalFilter" />

    <!-- OpenEdge CORS Filter -->
    <custom-filter before="SECURITY_CONTEXT_FILTER"
                   ref="OECORSFilter" />

    <!-- URL access controls -->

    <!-- HTTP REST to AppServer gateway -->
    <intercept-url pattern="/rest/**"
                access="hasAnyRole('ROLE_ANONYMOUS')"/>
```

Controls Tomcat session manager
{stateless | never | always | ifRequired}

Monday, June 24, 13

Example: controlling URI access



```
      <!-- Standard web-app root for public data like index.html
           DO NOT grant /** permitAll() access  -->
      <intercept-url pattern="/*" method="GET"
                     access="permitAll()"/>

      <!-- Application public area example -->
      <!--
      <intercept-url pattern="/pub/**" access="permitAll()"/>
      -->

      <!-- HTTP dynamic JSP pages -->
      <intercept

      <!-- HTTP static files -->
      <intercept-url pattern="/static/error/*" method="GET"
                     access="permitAll()"/>

      <intercept-url pattern="/static/auth/*"
                     access="permitAll()" />

      <intercept-url pattern="/static/**"
                     access="hasAnyRole('ROLE_ANONYMOUS')"/>

      <!-- best practice: deny anything no explicitly granted -->
      <intercept-url pattern="/**" access="denyAll()"/>

      <!-- authentication model -->
      <http-basic />   <!-- min 1 authn bean required -->
      <anonymous />

      <!-- login session controls -->
      <!--session-management session-fixation-protection="none" /-->

      <!-- error handlers -->
      <access-denied-handler error-page="/static/error/error401.html" />

  </http>
```

```
<intercept-url
        pattern="uri"
        access="role-list"
        [ method="method" ]
        [ requires-channel="https" ]/>
```

role-list: permitAll()
denyAll()
hasAnyRole('roles')

roles: 'role' [, 'role' [,...]]

role: 'ROLE_provider-role'

provider-role: account provider role name

method: {GET|PUT|POST|DELETE}

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care
- Use a GRANT model - **the last intercept url pattern is ALWAYS - deny everybody access to everything**

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care
- Use a GRANT model - **the last intercept url pattern is ALWAYS - deny everybody access to everything**
- Put the most used resource patterns first

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care
- Use a GRANT model - **the last intercept url pattern is ALWAYS - deny everybody access to everything**
- Put the most used resource patterns first
- Put explicit resource patterns before wildcard patterns

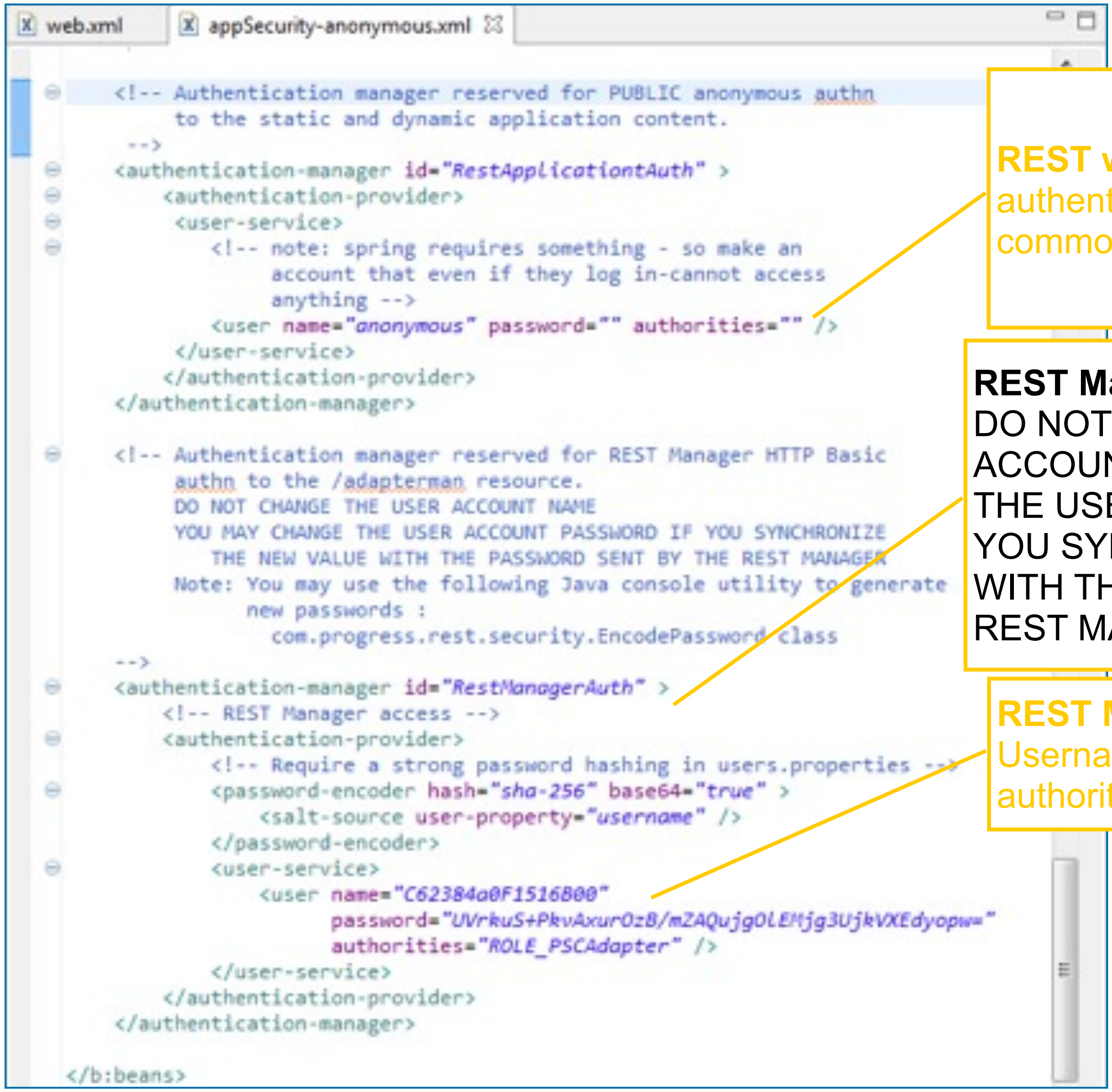
Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care
- Use a GRANT model - **the last intercept url pattern is ALWAYS - deny everybody access to everything**
- Put the most used resource patterns first
- Put explicit resource patterns before wildcard patterns
- If you add extra resources to a web application - add the intercept url pattern immediately

Intercept-url Tips

- Evaluated in the order they appear in the Spring configuration
- The first pattern match ends the search
- The more patterns to match, the slower it runs, use wildcards liberally but with great care
- Use a GRANT model - **the last intercept url pattern is ALWAYS - deny everybody access to everything**
- Put the most used resource patterns first
- Put explicit resource patterns before wildcard patterns
- If you add extra resources to a web application - add the intercept url pattern immediately
- The default pattern matching is ANT (* & ** wildcards) (REGEX is configurable if needed)

Authentication Managers (continued 7 of 7)



```
<!-- Authentication manager reserved for PUBLIC anonymous authn
     to the static and dynamic application content.
 -->
<authentication-manager id="RestApplicationtAuth" >
    <authentication-provider>
     <user-service>
        <!-- note: spring requires something - so make an
             account that even if they log in-cannot access
             anything -->
        <user name="anonymous" password="" authorities="" />
     </user-service>
    </authentication-provider>
</authentication-manager>

<!-- Authentication manager reserved for REST Manager HTTP Basic
     authn to the /adapterman resource.
     DO NOT CHANGE THE USER ACCOUNT NAME
     YOU MAY CHANGE THE USER ACCOUNT PASSWORD IF YOU SYNCHRONIZE
        THE NEW VALUE WITH THE PASSWORD SENT BY THE REST MANAGER
     Note: You may use the following Java console utility to generate
           new passwords :
             com.progress.rest.security.EncodePassword.class
-->
<authentication-manager id="RestManagerAuth" >
    <!-- REST Manager access -->
    <authentication-provider>
        <!-- Require a strong password hashing in users.properties -->
        <password-encoder hash="sha-256" base64="true" >
            <salt-source user-property="username" />
        </password-encoder>
        <user-service>
            <user name="C62384a0F1516B00"
                  password="UVrkuS+PkvAxurOzB/mZAQujgOLEMjg3UjkVXEdyopw="
                  authorities="ROLE_PSCAdapter" />
        </user-service>
    </authentication-provider>
</authentication-manager>

</b:beans>
```

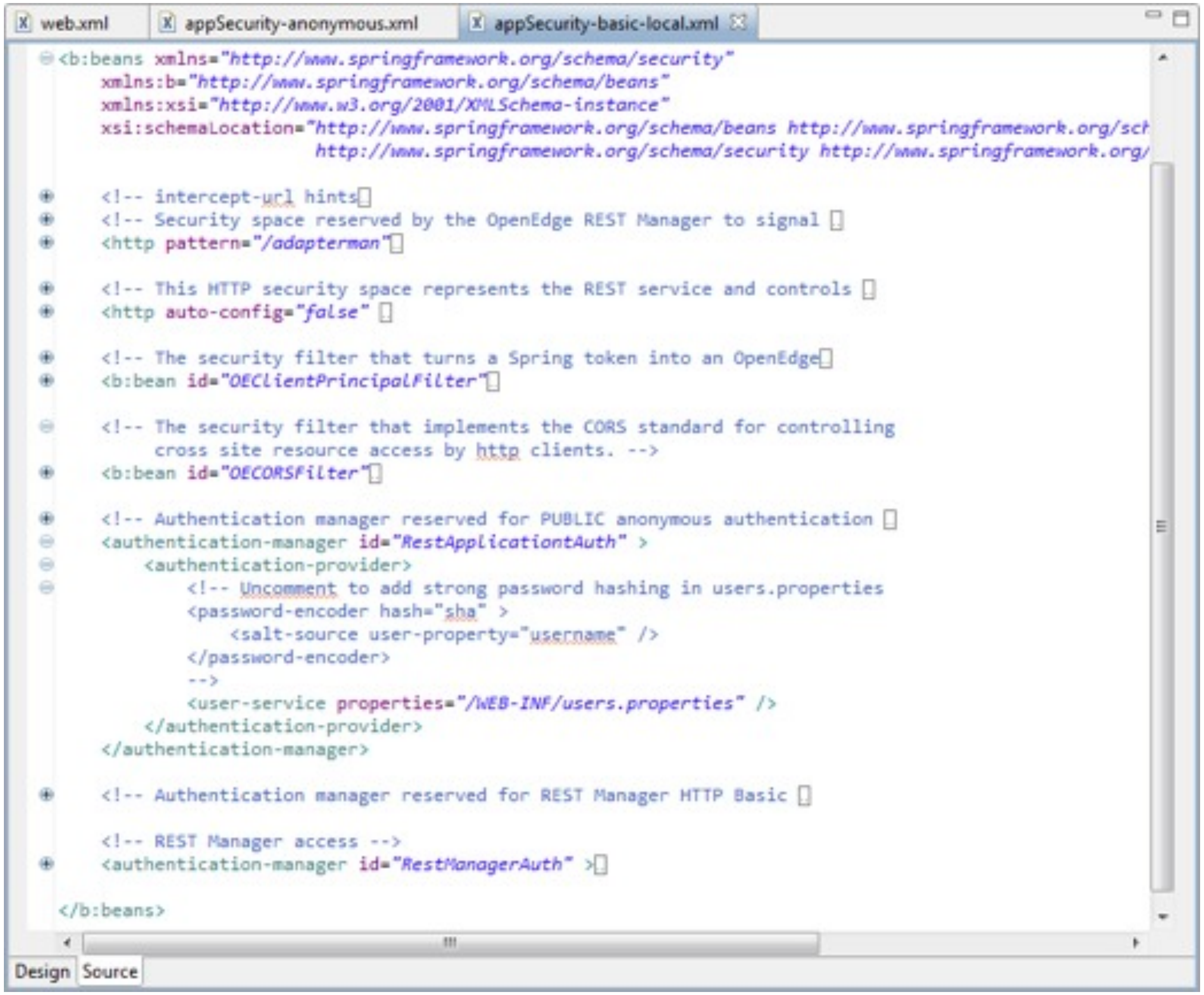
REST web application
authentication manager: controls the common user authentication process

REST Manager
DO NOT CHANGE THE USER ACCOUNT NAME YOU MAY CHANGE THE USER ACCOUNT PASSWORD IF YOU SYNCHRONIZE THE NEW VALUE WITH THE PASSWORD SENT BY THE REST MANAGER

REST Manager
Username and encoded password and authorities

Monday, June 24, 13

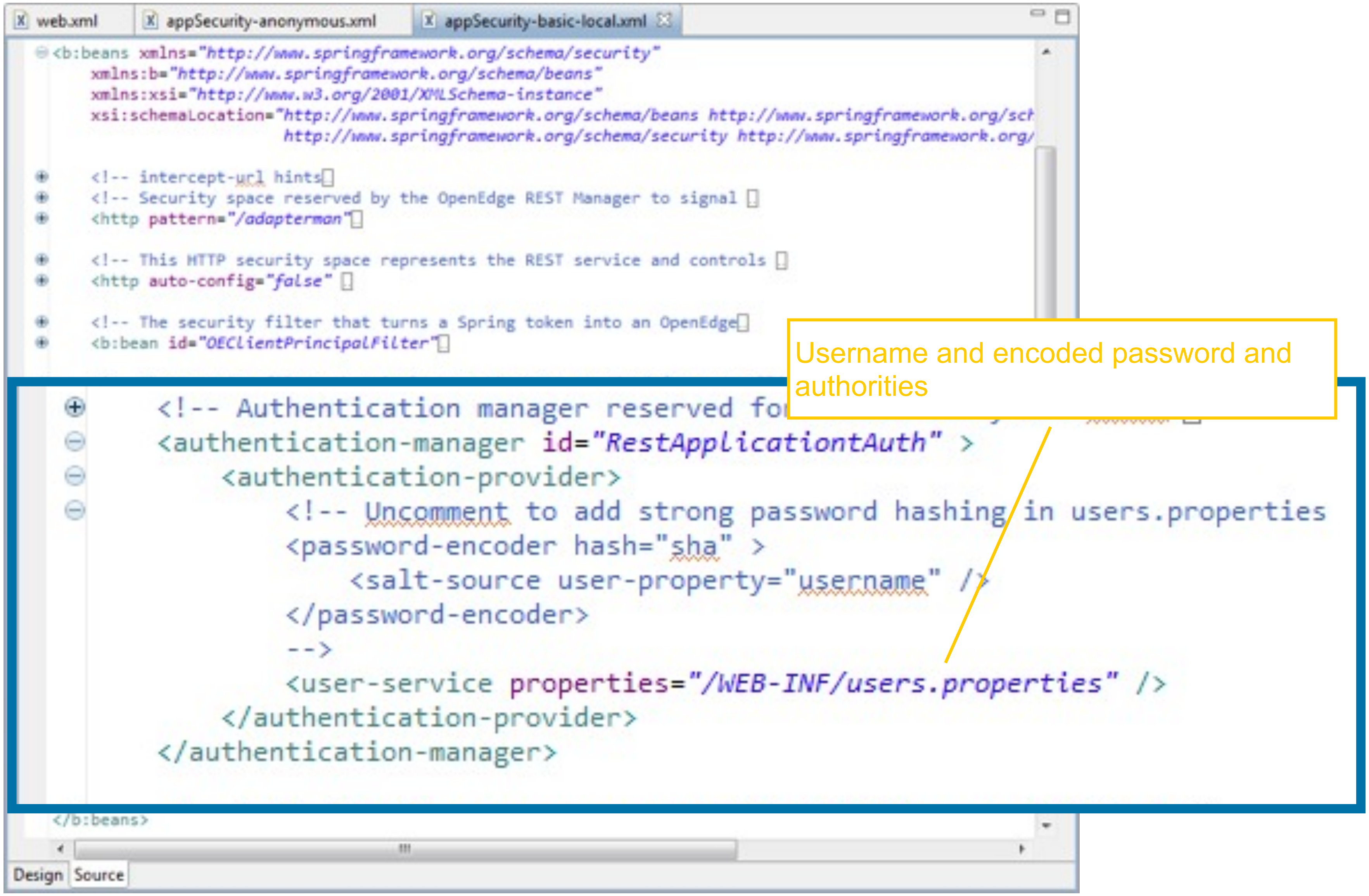
Example: User account authentication control



```
<b:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:b="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/sch
                        http://www.springframework.org/schema/security http://www.springframework.org/

    <!-- intercept-url hints
    <!-- Security space reserved by the OpenEdge REST Manager to signal 
    <http pattern="/adapterman"

    <!-- This HTTP security space represents the REST service and controls 
    <http auto-config="false" 

    <!-- The security filter that turns a Spring token into an OpenEdge
    <b:bean id="OEClientPrincipalFilter"

    <!-- The security filter that implements the CORS standard for controlling
         cross site resource access by http clients. -->
    <b:bean id="OECORSFilter"

    <!-- Authentication manager reserved for PUBLIC anonymous authentication 
    <authentication-manager id="RestApplicationtAuth" >
        <authentication-provider>
            <!-- Uncomment to add strong password hashing in users.properties
            <password-encoder hash="sha" >
                <salt-source user-property="username" />
            </password-encoder>
            -->
            <user-service properties="/WEB-INF/users.properties" />
        </authentication-provider>
    </authentication-manager>

    <!-- Authentication manager reserved for REST Manager HTTP Basic 

    <!-- REST Manager access -->
    <authentication-manager id="RestManagerAuth" >

</b:beans>
```

Monday, June 24, 13

Example: User account authentication control

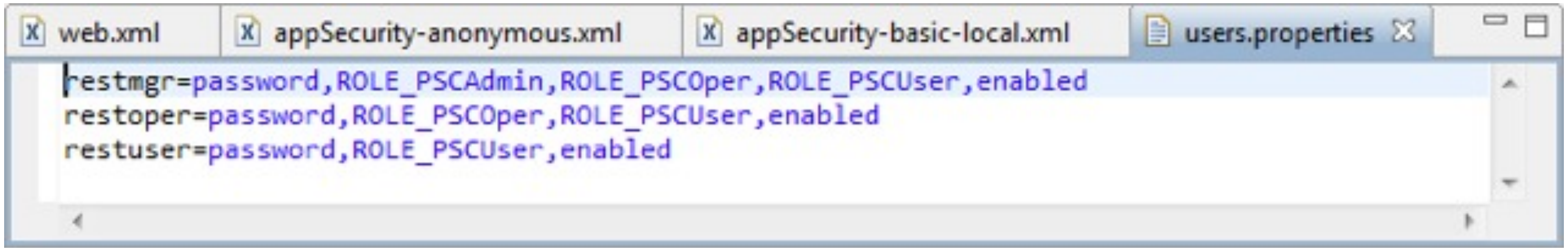

web.xml
appSecurity-anonymous.xml
appSecurity-basic-local.xml
Design
Source
Username and encoded password and authorities

```xml
<b:beans xmlns="http://www.springframework.org/schema/security"
    xmlns:b="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/sch
                        http://www.springframework.org/schema/security http://www.springframework.org/

    <!-- intercept-url hints
    <!-- Security space reserved by the OpenEdge REST Manager to signal
    <http pattern="/adapterman"

    <!-- This HTTP security space represents the REST service and controls
    <http auto-config="false"

    <!-- The security filter that turns a Spring token into an OpenEdge
    <b:bean id="OEClientPrincipalFilter"

    <!-- Authentication manager reserved for
    <authentication-manager id="RestApplicationAuth" >
        <authentication-provider>
            <!-- Uncomment to add strong password hashing in users.properties
            <password-encoder hash="sha" >
                <salt-source user-property="username" />
            </password-encoder>
            -->
            <user-service properties="/WEB-INF/users.properties" />
        </authentication-provider>
    </authentication-manager>

</b:beans>
```

Monday, June 24, 13

user.properties

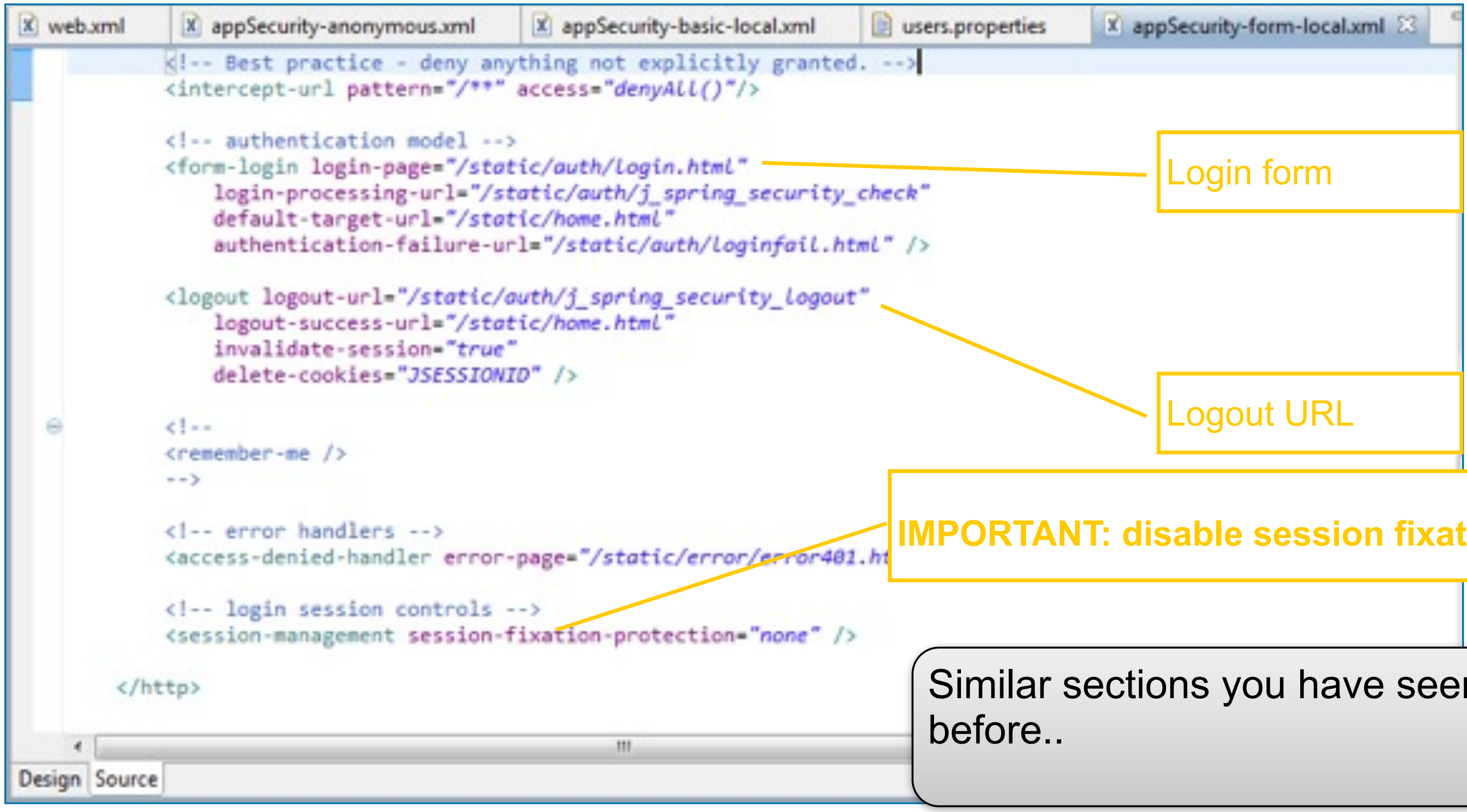
- Using the `users.properties` file, you can modify the user roles and privileges using the appropriate security configuration file
- You perform a similar set of edits regardless of which `appSecurity-XXX.xml` is used



```
restmgr=password,ROLE_PSCAdmin,ROLE_PSCOper,ROLE_PSCUser,enabled
restoper=password,ROLE_PSCOper,ROLE_PSCUser,enabled
restuser=password,ROLE_PSCUser,enabled
```

Example: controlling FORM login

- Similar to the **Basic configuration** Model
- The main difference is that it uses HTTP forms or messages to pass user credentials for authentication



web.xml
appSecurity-anonymous.xml
appSecurity-basic-local.xml
users.properties
appSecurity-form-local.xml
Design
Source

```
    <!-- Best practice - deny anything not explicitly granted. -->
    <intercept-url pattern="/**" access="denyAll()"/>

    <!-- authentication model -->
    <form-login login-page="/static/auth/login.html"
        login-processing-url="/static/auth/j_spring_security_check"
        default-target-url="/static/home.html"
        authentication-failure-url="/static/auth/loginfail.html" />

    <logout logout-url="/static/auth/j_spring_security_logout"
        logout-success-url="/static/home.html"
        invalidate-session="true"
        delete-cookies="JSESSIONID" />

    <!--
    <remember-me />
    -->

    <!-- error handlers -->
    <access-denied-handler error-page="/static/error/error401.ht

    <!-- login session controls -->
    <session-management session-fixation-protection="none" />

</http>
```

Login form

Logout URL

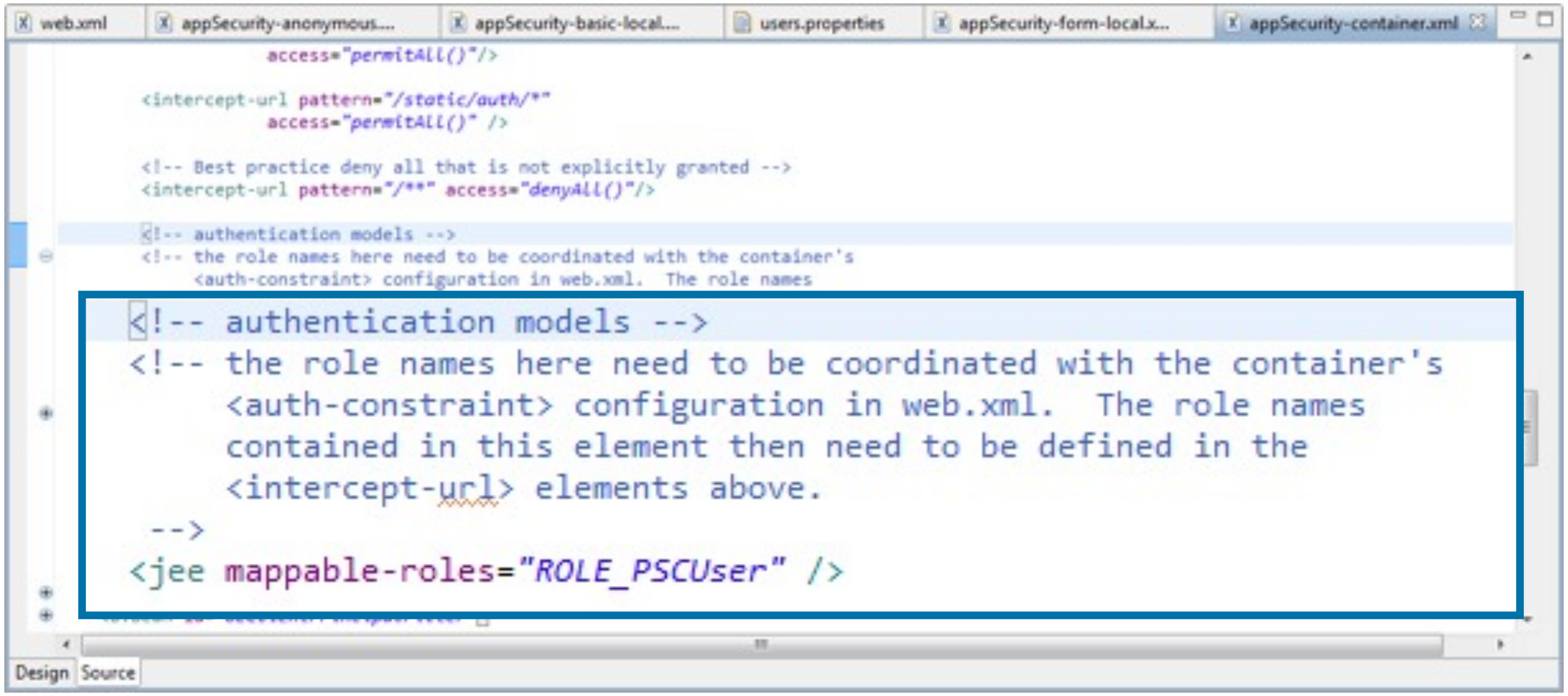
IMPORTANT: disable session fixation

Similar sections you have seen before..

Monday, June 24, 13

Example: controlling which container granted roles Spring will use for authorization

- This model integrates Spring Security framework with the authentication service of the Java container
- The Java container authenticates, and the Spring Security framework controls the authorization to REST application resources



web.xml
appSecurity-anonymous...
appSecurity-basic-local...
users.properties
appSecurity-form-local.x...
appSecurity-container.xml
Design
Source

```
                access="permitAll()"/>

    <intercept-url pattern="/static/auth/*"
                access="permitAll()" />

    <!-- Best practice deny all that is not explicitly granted -->
    <intercept-url pattern="/**" access="denyAll()"/>

    <!-- authentication models -->
    <!-- the role names here need to be coordinated with the container's
         <auth-constraint> configuration in web.xml.  The role names
```

```
<!-- authentication models -->
<!-- the role names here need to be coordinated with the container's
     <auth-constraint> configuration in web.xml.  The role names
     contained in this element then need to be defined in the
     <intercept-url> elements above.
 -->
<jee mappable-roles="ROLE_PSCUser" />
```

Monday, June 24, 13

REST Application Manager Security Considerations

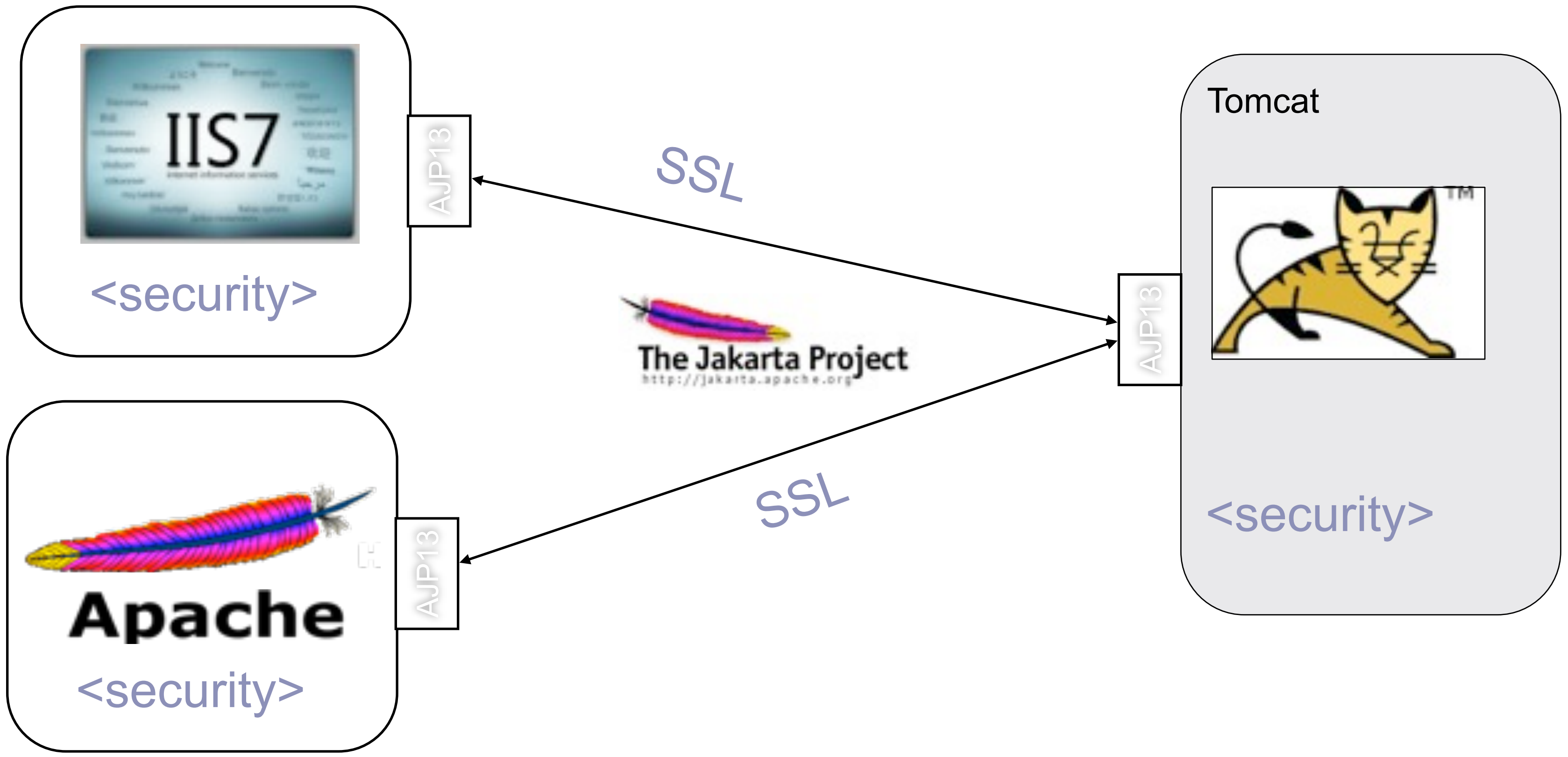
REST Application Manager Security Configuration

- Same Spring Security framework
- Same types of security configuration files
- DO NOT change the <intercept-url> elements and their patters
- YOU MAY change the
  - Authentication provider
  - ROLE names to be consistent with the authentication provider

Advanced Security Considerations

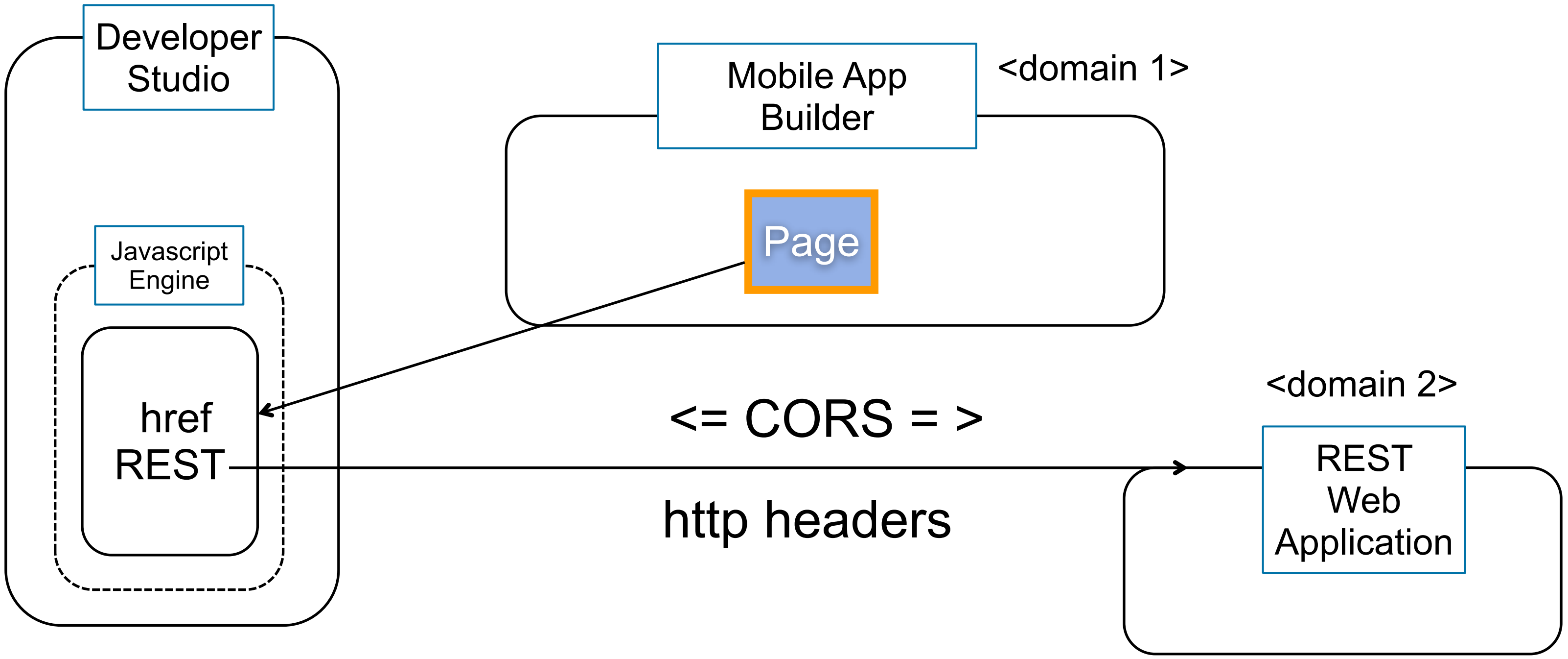
More Advance General Web Security – No Progress based tooling – Customer preferences

Integrating Microsoft ISS or Apache with Tomcat



CORS - Cross Origin Resource Scripting

- CORS is a W3C group standard that allows a *Mobile App's Java Script to access Web application resources in a DNS domain different from the one the current HTTP page and JavaScript were loaded from*
  - It works by adding new HTTP headers that allow servers to control resource access to permitted origin domains.



What is CORS (Cross Origin Resource Sharing)

- A CORS enabled server or web application classifies all HTTP requests as:
  - A CORS request that contains the HTTP ***Origin*** header
  - A **preflight request that contains the Access-Control-Request-Method header** in an OPTIONS request
  - A **generic request that does not contain any CORS HTTP headers**
- For more information on CORS standard and the advances in the standard, see the documentation at http://www.w3.org/TR/cors/

Example: OpenEdge CORS support

1. Identify and open the security configuration you applied to your REST application

2. In the security configuration file, `appSecurity-XXX.xml`, uncomment *only the required properties and you must assign a value to those properties*

```
<b:bean id="OECORSFilter"
      class="com.progress.rest.security.OECORSFilter" >
      <b:property name="allowAll" value="false" />
      <b:property name="allowDomains" value="*" />
      <b:property name="allowSubdomains" value="false" />
      <b:property name="allowMethods" value="GET,PUT,POST,DE
      <b:property name="messageHeaders" value="Accept,
             Accept-Language, Content-Language, Content-Type,
            X-CLIENT-CONTEXT-ID, Origin, Pragma, Cache-Contr
            Access-Control-Request-Headers,
            Access-Control-Request-Method" />
      <b:property name="responseHeaders" value="Cache-Control,
      Content-Language, Content-Type, Expires,
      X-CLIENT-CONTEXT-ID" />
      <b:property name="supportCredentials" value="true" />
      <b:property name="maxAge" value="-1" />
</b:bean>
```

OECORSFilter

allowAll*

allowDomains*

MessageHeaders

repsonseHeaders

Monday, June 24, 13

OECORSFilter properties

Property name	Description	Data types	Default	Range
allowAll[1]	Specifies that CORS filter allow every client request. If this property is set to **true,** all the other CORS properties values are ignored by the CORS filter.	Boolean	true	true or false
allowDomains[2]	Specifies the domains that can make server requests.	String	*	{ "*" | "domain1 [,domain2 ...]“ }
allowSubdomains	Specifies if subdomains of the permitted domains be allowed to make server requests.	Boolean	false	true or False
allowMethods[3]	Specifies valid HTTP method names.	String	GET, PUT, POST, DELETE	Valid HTTP methods in upper case.
messageHeaders[4]	Specifies the message header to be passed as a header to the server. If you are passing multiple messages, you must specify a comma-separated	String	Refer to the footnote.	Any valid string

Monday, June 24, 13

OECORSFilter properties (continued 2 of 2)

Property name	Description	Data types	Default	Range
responseHeaders[5]	Specifies the message header to be received by the client as a header from the server. If you are passing multiple messages, you must specify a comma-separated list of messages.	String	Refer to the footnote.	Any valid string
supportCredentials[6]	Controls whether the CORS filter allows the client to send user credentials in the form of a COOKIE	Boolean	true	true or false
maxAge	Specifies the maximum time (in seconds) for an application resource to be granted on request. After the specified time, the resource grant is revoked and the client must request access	Integer	-1	{ -1 | +n }

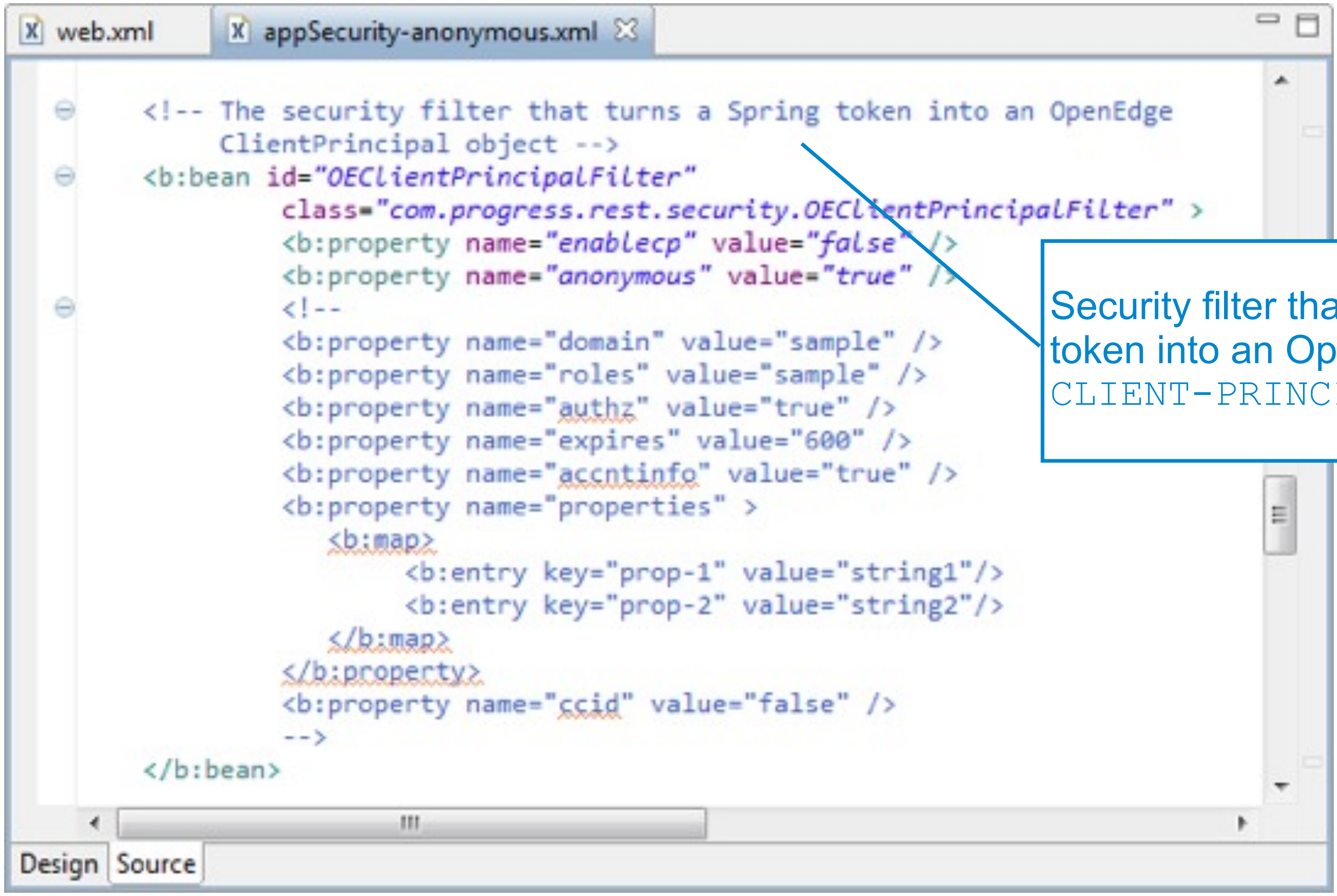
Monday, June 24, 13

AppServer Single Sign-On

- ClientPrincipal authentication token created from Spring authentication token
- ClientPrincipal passed with each request to Agent
- Request context information available via
  session:current-request-info:GetClientPrincipal().
  session:current-request-info:clientContextID.
  session:current-request-info:procedureName.
- ABL Client-Principal handle can be UNKNOWN
  - Anonymous security model
- ABL Client-Principal SESSION-ID attribute can be zero (0)
  - Stateless session configuration in REST web application
- Client-Principal SESSION-ID equals clientContextID attribute
- Client-Principal STATE attribute is SSO
  (represents authentication token generated by another system)

Example: OpenEdge Client-Principal Single Sign-On

AppServer Single Sign-On



```
<!-- The security filter that turns a Spring token into an OpenEdge
     ClientPrincipal object -->
<b:bean id="OEClientPrincipalFilter"
         class="com.progress.rest.security.OEClientPrincipalFilter" >
         <b:property name="enablecp" value="false" />
         <b:property name="anonymous" value="true" />
         <!--
         <b:property name="domain" value="sample" />
         <b:property name="roles" value="sample" />
         <b:property name="authz" value="true" />
         <b:property name="expires" value="600" />
         <b:property name="accntinfo" value="true" />
         <b:property name="properties" >
            <b:map>
                 <b:entry key="prop-1" value="string1"/>
                 <b:entry key="prop-2" value="string2"/>
            </b:map>
         </b:property>
         <b:property name="ccid" value="false" />
         -->
</b:bean>
```

Security filter that turns a Spring token into an OpenEdge `CLIENT-PRINCIPAL`

Monday, June 24, 13

OEClientPrincipalFilter properties

Property name	Description	Data types	Default	Range
enablecp	Enables/disables sending CP to AppServer	Boolean	true	true or false
domain	The OpenEdge domain into which to insert the foreign system's user account ID	String	web app context name	"<valid-string>"
roles	Comma separated list of static role names to grant the foreign user account	String	""	"r1[,r2[,r3...]]]"
expires	Optional ClientPrincipal expiration delta time (in seconds) after which it will not be validated for use in OpenEdge	integer	0 (none)	0 - max positive int
accntinfo	Insert the Spring token's account properties into the ClientPrincipal's PROPERTIES attribute	Boolean	FALSE	true or false

OEClientPrincipalFilter properties (continued 2 of 2)

Property name	Description	Data types	Default	Range
<prop-name>	Adds static ClientPrincipal PROPERTIES attributes	String	“”	“<valid-string>”
ccid	Pass Spring session-id to http client in the X-CLIENT-CONTEXT-ID header	Boolean	FALSE	true or false
anonymous	Pass anonymous user-id as ClientPrincipal to AppServer	Boolean	FALSE	true or false

Security Resources – Self Study

Industry Resources

- OWASP web application security best practices (checklist)

  See http://code.google.com/p/owasp-testing-checklist/
- Tomcat

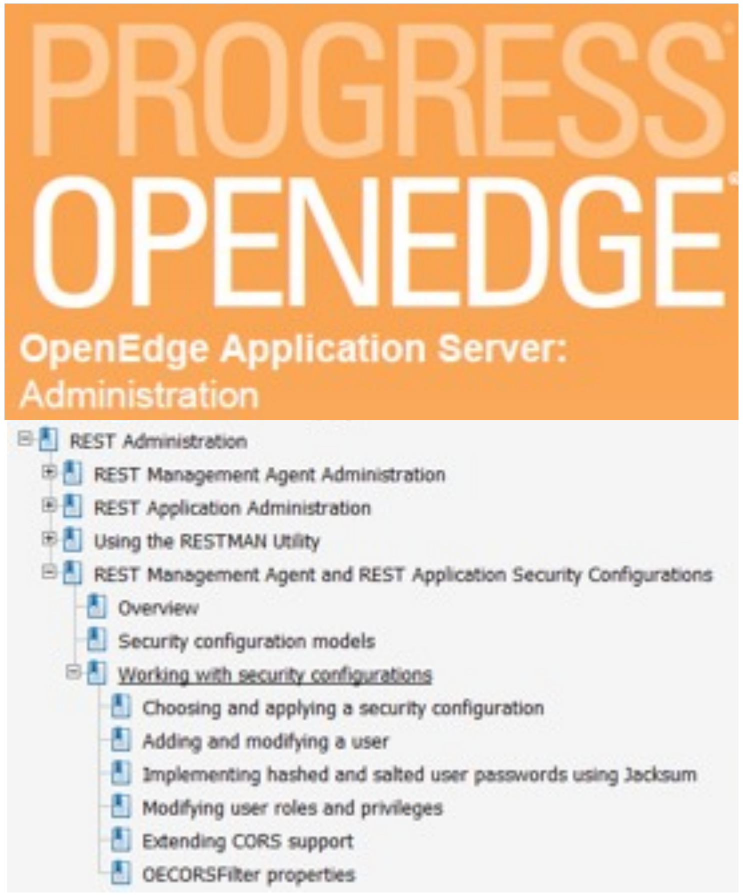
  See http://www.coreservlets.com/Apache-Tomcat-Tutorial/
- AJP13 IIS/Apache to Tomcat

  See http://tomcat.apache.org/tomcat-4.0-doc/config/ajp.html
- Web application Spring Security

  See http://static.springsource.org/spring-security/site/reference.html
- CORS

  See http://www.html5rocks.com/en/tutorials/cors/

Open Edge Resources


PROGRESS OPENEDGE

OpenEdge Application Server: Administration

- REST Administration
  - REST Management Agent Administration
  - REST Application Administration
  - Using the RESTMAN Utility
  - REST Management Agent and REST Application Security Configurations
    - Overview
    - Security configuration models
    - Working with security configurations
      - Choosing and applying a security configuration
      - Adding and modifying a user
      - Implementing hashed and salted user passwords using Jacksum
      - Modifying user roles and privileges
      - Extending CORS support
      - OECORSFilter properties

Monday, June 24, 13

PROGRESS
software