What’s new in PASOE
Spring Security for OE 11.7+
Making it easier

Cheryl LaBarge, Ruben Dröge, Chris Skeldon
Progress

What will you learn:

- Review of Spring security
- Making it easier to…
  - Customize
  - Test
  - Troubleshoot
  - Change authentication providers
Review of Spring security?

Why Spring Security?

Spring security is a powerful and highly customizable authentication and access-control framework.

- **Built into PASOE**
  - Spring is automatically started when you start an instance

- **All requests** must go through the Spring security process to generate a security token
**Why tokens are useful?**

- An object that contains both **user credentials** and **additional information** about the user's roles and capabilities
  - Offers flexibility - I decide how to give them out
    - Anonymous or check a flat file, LDAP, AD, OERealm, OpenEdge Authentication Gateway, etc.
  - Sealed – You can't transfer it to another person
  - Expires* - limits the chance that others will use that credential

**Spring security is always on!**

*For every request!*
1. Check your request – What happens?

- Spring applies industry standard filters
  - HTTPS filters
  - HTTPS [SSL/TLS] client login filters
  - CORS (Cross-origin resource sharing) filters
  - CSRF (Cross site request forgery) filters

**Bottom line: Is the request compliant and reliable?**
- For more detail on these topics, search the web. Spring is checking so you don’t have to check!
  - Additional steps are required to enable TLS, CORS, etc.

2. Authenticate - What happens?

- Spring checks your configuration files to decide
  - What type of authentication?
    - Direct login (user name & password)
      - Where do I find the user provider?
    - Single Sign On (SSO) still check verify
      - Already authenticated

**Bottom line: Do I check the users credentials? If so, where to I find the right list to verify the users are who they say they are**
3. Authorize – What happens?

- Based on your configuration, Spring checks the following:
  - Do we limit access to different types of application (REST, Web, etc.)
  - Do we limit access to parts of the application (Accounting, Finance, Human Capital)

- Bottom line: Am I limiting access to my application based on the user's role in the organization

4. Generate CLIENT-PRINCIPAL – Why?

- Spring generates a security token
- OpenEdge generates a sealed CLIENT-PRINCIPAL
  - Domains
  - Domain access keys

- Bottom line: We transforms the Spring token into a CLIENT-PRINCIPAL to add ABL specific details
Which applications can use Spring security?

User requests access

Authentication Provider

- APSV
- SOAP
- REST
- WEB
- Static

Where can I get user account information?

User requests access

Authentication Provider

- None
- users.properties
- LDAP - Lightweight Directory Access Protocol
- Active Directory
- OERealm
- Single Sign-On
- Container
- 11.7
  - ExtLocal encrypted users.properties (11.7)
- 11.7.2
  - Single Sign-on extending to HTTP
  - OpenEdge Authentication Gateway (STS)
  - OAuth2
How do I customize Spring Security

- **Configuration files**
  - 11.6 uses .xml files
  - 11.7.x uses two files
    - oeablSecurity.properties
    - oeablSecurity.csv
  - **Lesson learned**
    - xml files are error prone
    - .properties and .csv file are easier to
      - Edit
      - Debug
      - Migrate when Spring changes schema files

Managing security prior to 11.7

- **Edit** web.xml
  - Find the `<context-param>` tag
  - Uncomment one of the `<param-value>` from the list of security configurations in the `<param-value>` code block.
Importing 11.6 to 11.7 automatically migrates files

New files and backup available
Benefits of Spring to OpenEdge

- First line of defense for access control
  - Java industry standard
    - DLC/servers/pasoe/common/lib
  - Always on, always runs first
  - Sits outside of your application
  - Blocks request if authentication and authorization fail
  - Always creates a security token

- Easy to use in existing code
  - Automatically generates a CLIENT-PRINCIPAL from the Spring security token so you can use it in your existing applications

- 11.7.x Easy to configure
  - Using properties files and .csv files
    - OpenEdge handles 99% of details
  - Can switch Authentication Providers easily
  - Avoids issues when schema changes

How do I customize it
11.7 Best Practice

- Use `.properties` and `.csv` files
  - If you read about Spring Security you will find that Spring uses xml files to change behavior
    - Don’t use .xml
    - Our updates could overwrite any changes in your xml files

- Bottom line:
  - *Keep it simple, always look through the properties we recommend if you need to change something… chances are…*
    - *there’s a property for that!*

How do I find the right property?

- Each oeabl[.war] web application’s URI space is divided into 5 individually configured Spring Security sub-spaces, arranged by transport
  - APSV
  - SOAP
  - **REST** (Highlighted)
  - WEB
  - Static

  ![Diagram](image.png)
How do I find the right property?

- Your transport determines has potential values for properties

- Example:
  - `client.login.model` - method client uses to authenticate via HTTP messages

<table>
<thead>
<tr>
<th>Transport</th>
<th>URI Path</th>
<th>Login Models</th>
</tr>
</thead>
<tbody>
<tr>
<td>APSV</td>
<td>apsv/**</td>
<td>[ none ]; basic</td>
</tr>
<tr>
<td>SOAP</td>
<td>soap/**</td>
<td>[ none ]; basic</td>
</tr>
<tr>
<td>REST</td>
<td>rest/**</td>
<td>[ anonymous ]; basic; form; container; sso; oAuth2</td>
</tr>
<tr>
<td>WEB</td>
<td>web/**</td>
<td>[ anonymous ]; basic; form; container; sso; oAuth2</td>
</tr>
<tr>
<td>Static files</td>
<td>**</td>
<td>[ anonymous ]; basic; form; container; sso; oAuth2</td>
</tr>
</tbody>
</table>

- 11.7 added extension to sso and 11.7.2 added oAuth2

How do I learn more about a specific properties?

- Readme file
  - ![Install]/servers/pasoe/conf
    - oeablSecurity.properties.readme

- Outlines
  - Hierarchy of .properties files
  - Rules for setting property values
  - Search for your property

Documentation

*Spring security bean properties definition file for all oeml-user based web applications found in a PASO instance. The properties values found in this file constitute the full set of Spring security configuration properties and their values.*

- *Hierarchy of .properties files:
  - Not for use resolves property settings by loading multiple .properties files and by using the first declared value it finds. The .properties files and their values are:

  - # specifying properties and values:
    - # The properties and their values follow the Java properties file format, where each line contains a single property-name and property-value pair. The properties are set in the following directories:
    - # separated by and a equals sign delimiter.
    - # Rules:
      - # Property values are generally set in hierarchical name-spaces.

---

*http.all.auth.manager string value [local][remote][ldap][ad][kerberos]*

- Specifies the name of the user accounts used to validate user authentication (sub-accounts). Ignored when the client.login.model is (anonymous|basic) on container.
- Applies to all oeml protocol transports (aps,soap,rest,web,static-file) on local

*client.login.model string value [anonymous][basic][form][container][sso]*

- Specifies which Spring Security client security (login) process to select and execute at runtime.
- [anonymous]
What does the .csv file customize?

- Control URL access controls for web applications
  - In 11.6

```xml
<intercept-url
  pattern="/web/sales/**"
  method="GET"
  access="hasAnyRole('ROLE_PSCAdmin', 'ROLE_PSCUser')"/>
```

- In 11.7 separated from .properties because it does not store a standard property=value pair

```
"/web/sales/**", "GET", "hasAnyRole('ROLE_PSCAdmin', 'ROLE_PSCUser')"
```

- More detail a little later

Which file should I edit? Hierarchy of files...

- **Progress install** is the superset includes all properties set defaults for all server instance
  - `[Install]/servers/pasoe/conf/oeablSecurity.properties and .csv`

- **An Instance** effects all ABL applications deployed on that instance
  - `[Instance-name]/conf/oeablSecurity.properties and .csv`

- **ABL Application** level effects all web applications
  - `[Instance-name]/ablapps/oepas1APP/conf/oeablSecurity.properties and .csv`

- **Web application** controls on that branch of the business application
  - `[Instance-name]/webapps/inventory/WEB-INF/oeablSecurity.properties and .csv`
How do I apply changes?

- You will need to restart the Progress Application Server for OpenEdge
- When you start the Progress Application Server for OpenEdge the property changes will take effect

```
proenv/pasman pasoestart -I oepas1 -restart
Starting stopped PASOE instance oepas1

Start action: start
Initial state: stopped
Initial processes: 0
Exit state: started
Exit description: Starting stopped PASOE instance oepas1
Exit processes: 4332 4844
Exit status: 0
Exit errors:
proenv>
```

Making it easier to test!
Anonymous test application – Available in 11.7.2

Use a browser to call the ABL Application Ping service

Success looks like this json...

Why use a simple ping service?
To focus on Spring settings application

Exploring authentication
Let’s make a change requiring the user to login

* For every request!

What will the user experience differently?

Default value is anonymous

http://localhost:8810/rest/_oepingService/_oeping

To require a login use form

http://localhost:8810/rest/_oepingService/_oeping
Setting the `client.login.model` property to `form`

Default value is anonymous

- `oeeblSecurity.properties` with `client.login.model=anonymous`

Require a username and password

- `oeeblSecurity.properties` with `client.login.model=form`

- **Development**
  - Test cases not for Production use!

- **Users.properties** flat file located in `C:\Progress\OpenEdge\servers\pase\webapps\ROOT\WEB-INF`

Using `form` provides default login page

- The request is redirected to the `login.jsp` page
- The username and password are sent to Spring for authentication and authorization
What is the `http.all.authmanager`?

```
# Authentication Manager For All Transports
#
# Set which Authentication Provider [to user accounts] to use for all
# transports.
#
# The list of names includes:
# local
# extlocal
# ldap
# ad
# oirealm
#
# User.properties flat file located in
# C:\Progress\OpenEdge\servers\pasoe\webapps\ROOT\WEB-INF

http.all.authmanager=local
```

http.all.authmanager: String value {local|extlocal|ldap|ad|oirealm}

Specifies the source of user accounts used to validate user authentication credentials. Ignored when the client.login.model is {anonymous|sso|container}
Applies to all ceabl protocol transports (apsv, soap, rest, web, static-file) [local]
Checking roles

1. Check Request
   - User requests access
   - Check User
     - Anonymous
     - User info
     - Yes
     - Check Roles
       - No roles
       - Yes
     - 3. Authorize URL
       - Authentication Provider
       - Yes
       - Yes
     - 4. Generate the CLIENT-PRINCIPAL

How does a direct login handle the request

- The oeablSecurity.csv file contains the roles
- Intercept-url for each transport type
- Default for REST is ‘ROLE_PSCUser’
Dissecting the entry

```
## Intercept-url definitions for the REST transport URIs ##
''/rest/**','*','hasAnyRole('ROLE_PSCUser')''
```

1. Pattern - URL pattern (wildcards and regular expressions)
   - * Current directory
   - ** Current and subdirectories
2. Method – HTTP access method (optional in 11.6, but required in 11.7)
3. Access - Role[s] that are allowed access to the resource

Changing the role

- Use the 'ROLE_PSCAdmin'
Troubleshooting

Where to look when it goes wrong

**Bad user**

![Image of Progress Application Server with login field showing 'test']

**Immediate feedback**

![Image of Progress Application Server error message with 401 Unauthorized and Authentication failed]

401 Unauthorized
Authentication failed
**Default logging**

- Log files provide general error messages but not all details

<table>
<thead>
<tr>
<th>Name</th>
<th>Date modified</th>
<th>Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>startupErrors</td>
<td>10/10/2017 12:43</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>pasoeStart</td>
<td>10/10/2017 12:43</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>cepas1.agent</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>349 KB</td>
</tr>
<tr>
<td>cepas1.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>cemanager.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>manager.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>localhost_access_log.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>localhost.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>host-manager.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>0 KB</td>
</tr>
<tr>
<td>cemanager.pid</td>
<td>10/10/2017 12:44</td>
<td>PID File</td>
<td>1 KB</td>
</tr>
<tr>
<td>cemanager.2017-10-10</td>
<td>10/10/2017 12:44</td>
<td>Test Document</td>
<td>3 KB</td>
</tr>
</tbody>
</table>

**Troubleshooting during Development**

- Restart the server for changes to take effect
- WARNING: Generates HUGE log files for development use only!!!

```xml
<logger name="org.springframework" level="DEBUG"/>
```
Catching authentication errors

- Same bad user

- More details with DEBUG level logging

- Don’t leave on in Production, debugging only during development

Catching authorization errors

- A user doesn’t have the right role

- More details
Catching configuration errors

- Manual edits are error prone, how let's look an example

```java
Client.login.model='form'
```

Context initialization failed

Extra quote

There it is. `form`

There it is again

Catching configuration errors (continued)
11.7 Log Viewer – View log files for PAS for OpenEdge

- The Log Viewer helps you to view and monitor the log files

11.7 Troubleshooting direct logins in Production

- Provides
  - More login failures than `localhost_access.<date>.txt`
  - Less than full Spring Security logging you would use in a development environment
  - Just right…Gathers additional information without crippling your file system with gigabyte logs

- User authentication:
  - INFO - User-id/password validation and a hint of why it failed
  - DEBUG – Authentication failures and why, including a fuller description of the user's credentials

- User authorization
  - INFO –URL path access check failures
  - DEBUG –URL path access check, including the user’s ROLE information and URL access rule
Example: Bad password & Bad userid

# Test of user with a bad password
15:04:17.485/126304 [catalina-exec-2]
INFO c.p.a.s.s.OEAuthenticationLogger - Authentication event:
 status=failure :
 user=restadmin :
 reason=bad password :
 details=Bad credentials

# Test of a user with a bad user-id
15:04:43.760/152579 [catalina-exec-3]
INFO c.p.a.s.s.OEAuthenticationLogger - Authentication event:
 status=failure :
 user=estadmin :
 reason=bad password :
 details=Bad credentials

Log: Authenticated with no access

# Test of an authenticated user, but with no access granted in oeablSecurity.properties
15:05:08.718/177537 [catalina-exec-5]
INFO c.p.a.s.s.OEAuthorizationLogger - Authorization event :
 status=denied :
 reason=Access is denied :
 user=org.springframework.security.authentication.UsernamePasswordAuthenticationToken@9fbbfefd:
Principal:
org.springframework.security.core.userdetails.User@ebf23d9f:
Username: restuser; Password: [PROTECTED]; Enabled: true;
AccountNonExpired: true; credentialsNonExpired: true; AccountNonLocked: true;
Granted Authorities: ROLE_PSCUser;
Credentials: [PROTECTED];
Authenticated: true;
Details: org.springframework.security.web.authentication.WebAuthenticationDetails@b364:
RemoteIpAddress: 0:0:0:0:0:0:0:1; SessionId: null;
Granted Authorities: ROLE_PSCUser :
object=FilterInvocation: URL: /rest/_oepingService/_oeping :
attributes=[hasAnyRole('ROLE_PSCAdmin')]

11.7.2 Enhancements

- ABL application PING service
- Authentication using the OpenEdge Authentication Gateway
- Extending OpenEdge SSO to Web Applications
- Authentication with OAuth2 and JWT

Authentication using the OpenEdge Authentication Gateway
Using the OpenEdge Authentication Gateway

User requests access

Authentication Provider

OpenEdge Authentication Gateway
Security Token Service (STS)

Configuring the Gateway (producer)

On the instance level `oeablSecurity.properties`:

```plaintext
## OpenEdge Authentication Gateway client configuration
## for direct user logins to a PA90E server

sts.AuthProvider.multiTenant=true
sts.AuthProvider.userDomain=
sts.UserDetails.stsURL=HTTPS://host:port
sts.UserDetails.stsKeystore=sts.UserDetails.clientHeaderName-x-oSTS-token
sts.UserDetails.noHostVerify=false
sts.UserDetails.certLocation=${psc.as.oe.dlc}/certs
sts.UserDetails.tlsCipherSuites=
sts.UserDetails.tlsProtocols=
sts.UserDetails.userAgent=PA90E (Spring)
```

*Configured by the Authentication Gateway Servers Administrator
Configuring the application (consumer)

User requests access

```
# OpenEdge Authentication Gateway client configuration
# for direct user logins to a BASIX server
sts.AuthProvider.multitenant=true
sts.AuthProvider.userDomain=
sts.UserDetails.stsURL=https://host:port
sts.UserDetails.stsKeyStore=
sts.UserDetails.noHostVerify=true
```

*Configured by the Application Servers Administrator

Example from workshop configure Application

- Authentication Provider is now **sts**
- Same **stsURL** you tested with the authenticate command
  
  ```
  C:\OpenEdge\WRK\oepas1\webapps\ROOT\WEB-INF\oeablSecurity.properties
  File Edit Search View Encoding Language Settings Tools Macro Run Plugins Win
  1+http/all.authmanager-std
  2+client.login.model-form
  3+sts.AuthProvider.multitenant-true
  5+sts.UserDetails.noHostVerify-true
  +
  ```

- WARNING: For the training we have noHostVerify=true only to avoid the extra steps for configuring certificates because that is not Spring specific
Authentication Gateway requires Client\Server keys

Install as client key

Requires a Server Key to create client
Warning: Remember to remove server key once client is installed

Require a Server Key

Installing the client key

- The server key will be given to you by the Authentication Gateway server administrator
- On the client machine enter the following:

```
proenv> stsWithutil
install
-file c:\stsClientkey\oests-key.ecp
```

- Enter the password as password

```
proenv> stsWithutil install -file C:\stsClientkey\oests-key.ecp -url https://52.14.215.117:8810
Enter server key file password: Created client key file
```

- Once created you will delete the server key so only the generated file remains
Try your key before you lock your application

- You can use the authenticate command to test authenticating users

```bash
proenv> stsclientutil
-cmd authenticate
-nohostverify
-user ldap_admin@stspug -password password
```

Authentication succeeded

Test the application
Difference between Development and Production

Development Server
- Includes a server certificate specifically crafted to allow TLS connections
- Designed to fail but allow for testing with `–nohostverify` option

Production Server
- Administrator must obtain, configure, and use a non self-signed server certificate to validate
- We provide `certutil` tools to install

Using SSO
Direct Logins vs. Single Sign-on

- **Direct logins** require users to supply a user name and password for authentication and authorization.

- **Single Sign-On (SSO)** is an authentication process that allows a user to access multiple applications with one set of login credentials.

- **SSO** is a common procedure in enterprises, where a client accesses multiple resources connected to a local area network (LAN).

---

SSO typically works across servers.

- **Credentials passed (no new login)**

  - Direct logins require users to supply a user name and password for authentication and authorization.

  - Single Sign-On (SSO) is an authentication process that allows a user to access multiple applications with one set of login credentials.

  - SSO is a common procedure in enterprises, where a client accesses multiple resources connected to a local area network (LAN).
Applications may also reside on a single machine

The user is authenticated someplace!

- **SSO – Single Sign-On**
  - **Producer**

- **User requests access**

- **LDAP**

- **APSV**

- **SOAP**

- **REST**

- **WEB**

- **Static**
Configure the SSOProd (Producer) application

- Deploy a SSOProd application
  
  `tcman deploy -a SSOProd %DLC%\servers\paseo\extras\oeabl.war`

- Customize with properties

Producer’s oeablSecurities.properties

Existing LDAP and CP details

Additional SSO Token details
Deploying the Payments application (SSO Consumer)

- Deploy the **Payments** application
  
  ```
  tcman deploy -a Payments %DLC%\servers\pasoe\extras\oeabl.war
  ```

- Customize with properties

---

Consumer’s `oeablSecurities.properties`

Client login model is irrelevant

```
1 http.all.authmanager=local
2 client.login.model=basic
```

```
OEClientPrincipalFilter.enabled=true
OEClientPrincipalFilter.registryFile=ldapreg.bin
OESSO.error.detail=false
OESSO.tokenManager.tokenPolicy=disabled
OESSO.tokenManager.ssoTokenURLOption=OECF
OESSO.tokenManager.ssoTokenExpires=3600
OESSO.tokenManager.ssoAllowScope=Payments
OESSO.tokenManager.ssoRefresh=true
OESSO.tokenManager.ssoRefreshDeltaTime=3600
OESSO.tokenManager.springRolePrefix=
OESSOFilter.authPolicy=required
OESSOFilter.authClientType=*1
OESSORefreshFilter.refreshURL=/static/auth/token
OESSORefreshFilter.refreshClientType=*
```

Consumer only cares about a valid CLIENT-PRINCIPAL

ROOT and Payments are consumers
SSO tokens have a default timeout

![Diagram showing SSO tokens with default timeout]

POST   http://localhost:8811/SSOProd/static/auth/j_spring_security_check
Accept: application/json

j_username=ldap_admin
j_password=password

```json
{
  "access_token": "AKAAAQAAAAZBRE1JTgAaAAACAAAAAAACAAAAFtFq4MAAAAMAAAAAADgAAoAANAAAAADlZJTEVUQNDQXRSU4AAKAAAACAAAAAAAE2S6S5+AMAFAAAAAACFAAAAAGAAAAAGXNjaWVzZmlsaW5lcyAvcmVxdWVzdC9wYXNzd29yZS9uYXZl
  "refresh_token": "faab8d9bf-5861-4124-b9fe-7767e6eb256e.oepeas1",
  "token_type": "oecp",
  "expires_in": 3600
}
```

Token used to generate a new access_token for the same client-principal generated for the user
base64 encoded client-principal token
Performing SSO using OECP access_token

GET http://localhost:8811/web/pdo/DOHService/SportsCE
Authorization: OECP <access_token>

AKAAQAAAZBRE1J7gA0AACAAMABmzWVJAcOCcAaAAAAIAAAAIwAFn1Fo84AAAMAAAAAAAgAMAACAAGyAAAIUJ9bY2Nc9Ncb2bpg
AAaAAAADADrJrTEVUIFNDQURNSUAAAAAowAAAAdk3OEU5mJrDRqQbR1VrEDZ2FrMzhCnZ7Y4MkVznZ9j5Q0jTjDRQg3MkU2R9VDRDeYOC
Sw7XNRYu1Fq8BVrWAlA0AAQAAAACAAAB29jSS+hYMAAFAAAAACEAoAAuAAAAGXnjp3B1GzZpmbfYu5UsWS52W5o53J5AAAAGAAAXA
AAACP//////////8QANAAAGwAAA///////EADQABwAAAIAAAAAzicA8AsAAVAAAEG58KQdb5MLx6F6UPWzY=

Refresh button demonstrates refreshing
Generating a new access_token by using a refresh_token

**POST** http://localhost:8811/SSOProd/static/auth/token?op=refresh
Accept: application/json

```
{"token_type":"oecp","refresh_token":"faab8dba-5861-4124-b9fe-7767e6eb256e.oepas1"}
```

```
{"access_token":"AKAAAQAAAAAZBRE1JTgAAoAACAAAABmIzeWVkaAcQAAAAsAAAAIAAAAAFn1K8UAMAAMAAAAAgAMAKAAAGgAAAA9TU09BY2NiC3NUb2tgAAcAAAAAAADJfPEVUfFhDQURNSAAAAAAAdwAAAAAdkzRUREN0FFQ1QZ5zBCMTBDODc4MzI5NzhiFmzdDNDIyODQ5OTZGNkQvMEuNi5wYXNfRU1FOQVVRwAAAQAAAAAAAABZ9TrVAMAAFAAAACEAcAAUAAGXN1Gj3BlAGZpbmFuY2UsSW52ZTV5b3J5AAAAAAAAAAA=","refresh_token":"9301bf02-7a52-485b-b24d-6182d28307a9.oepas1","token_type":"oecp","expires_in":3600}
```

**Configuring logging**

- In the current release you will need to copy `logging.xml` to each application so that they match
- This is a known issue and scheduled for an upcoming release
Authentication with OAuth2 and JWT

User requests access

JWT

APSV SOAP REST WEB Static

Login to Progress Application - OAuth2

Username:
Password:

AWS Sign-in Google Sign-in

PASOE Support for JWT and OAuth 2 - Samples

Progress

Progress
What you learned:

- Spring security concept
- 11.7.x makes it easier
  - Customize moved from .xml to .properties and .csv files
  - Test with new ping service
  - Troubleshoot – Production debugging
  - Change authentication providers
    - Extending SSO to http client
    - Using Authentication Gateway (STS)
    - Adding OAuth2 with JSON Web Tokens (JWT) sample

Where can I learn more...

General resources

- Installation
  - oeablSecurity.properties.REAME
- Documentation
  - Spring Security Models and templates
- Education
  - Progress Application Server for OpenEdge Administration
  - Introduction to Progress Application Server OpenEdge for Developers
- Communities
  - Whitepaper
    - PAS for OpenEdge WebSpeed with OERealm Security (OpenEdge 11.6+)
  - Sample coming soon...
    - PASOE Support for JWT and OAuth2 - Samples

What’s new in 11.7.2

- OpenEdge-Service Pack 11.7.2: New Information Guide
  - ABL application PING service
  - Using the OpenEdge Authentication Gateway for Authentication
  - Authentication with OAuth2 and JWT
  - Extending OpenEdge SSO to Web Applications (Documentation)