Mobile Security Jump Start

Wayne Henshaw & Mike Jacobs
Progress OpenEdge
October 8, 2013

PROGRESS EXCHANGE 2013
DISCOVER. DEVELOP. DELIVER.

Agenda

- Architectural basics
  - REST service
  - Mobile client
- Making required choices
  - Authentication model
  - User sessions
  - AppServer SSO
- Diving into the code
  - REST service
  - Mobile client
- What to do when things go sideways

Some Assembly Required...

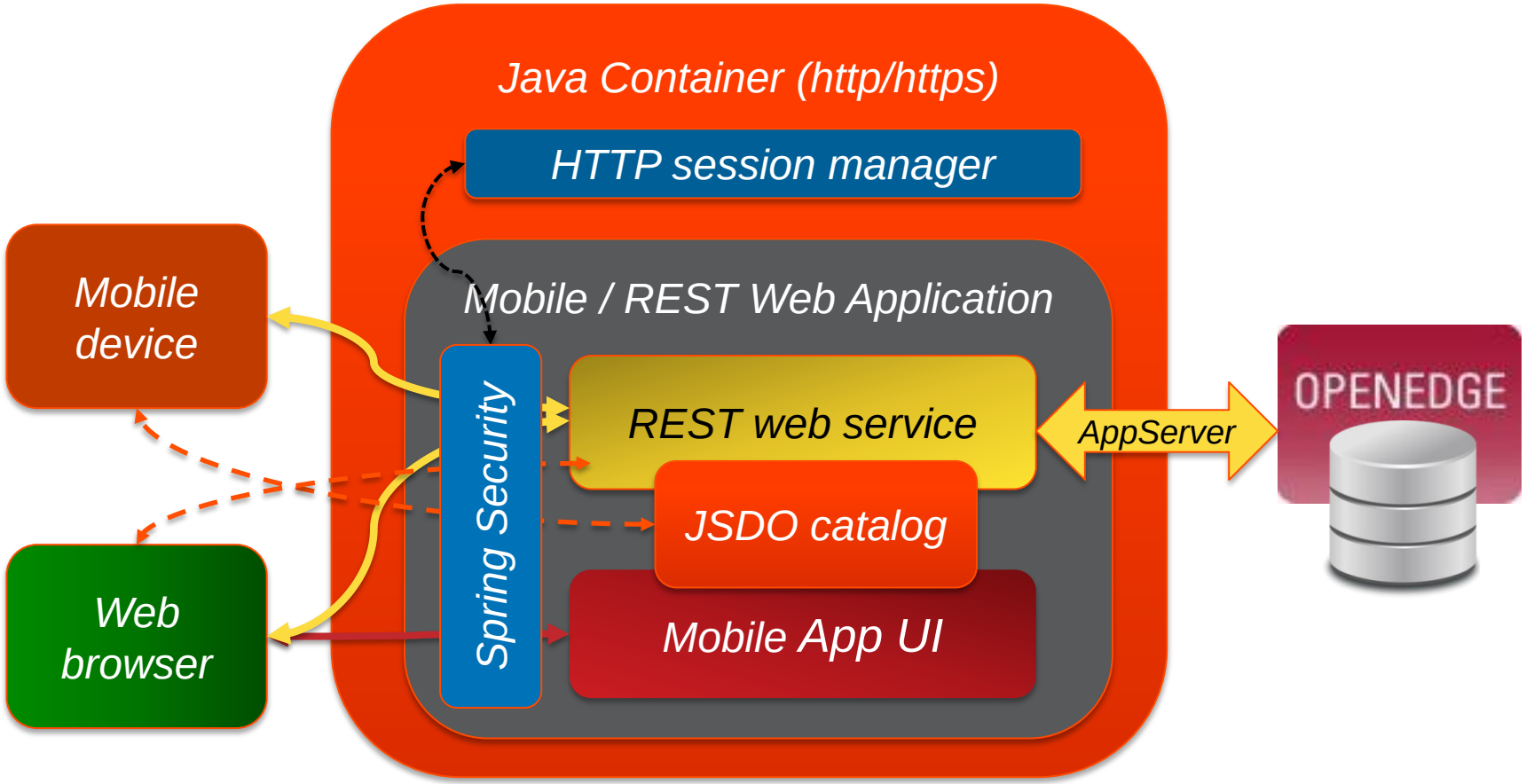
- OpenEdge (OE) Web applications provide the starting point for your application's security
- New in Mobile & REST services you will configure and use these security layers
  - Web server (i.e. Tomcat 7+)
    1. Everything will use **SSL/TLS** for web application's client to web server
    2. **Web server** [login] **session management**
    3. **Web server or Mobile/REST web application** user **authentication**
    4. Mobile/REST **web application** role-based **authorization** to HTTP resources
  - OpenEdge AppServer
    1. OpenEdge AppServer for **application** level **authorization**

Web Application Security Goals

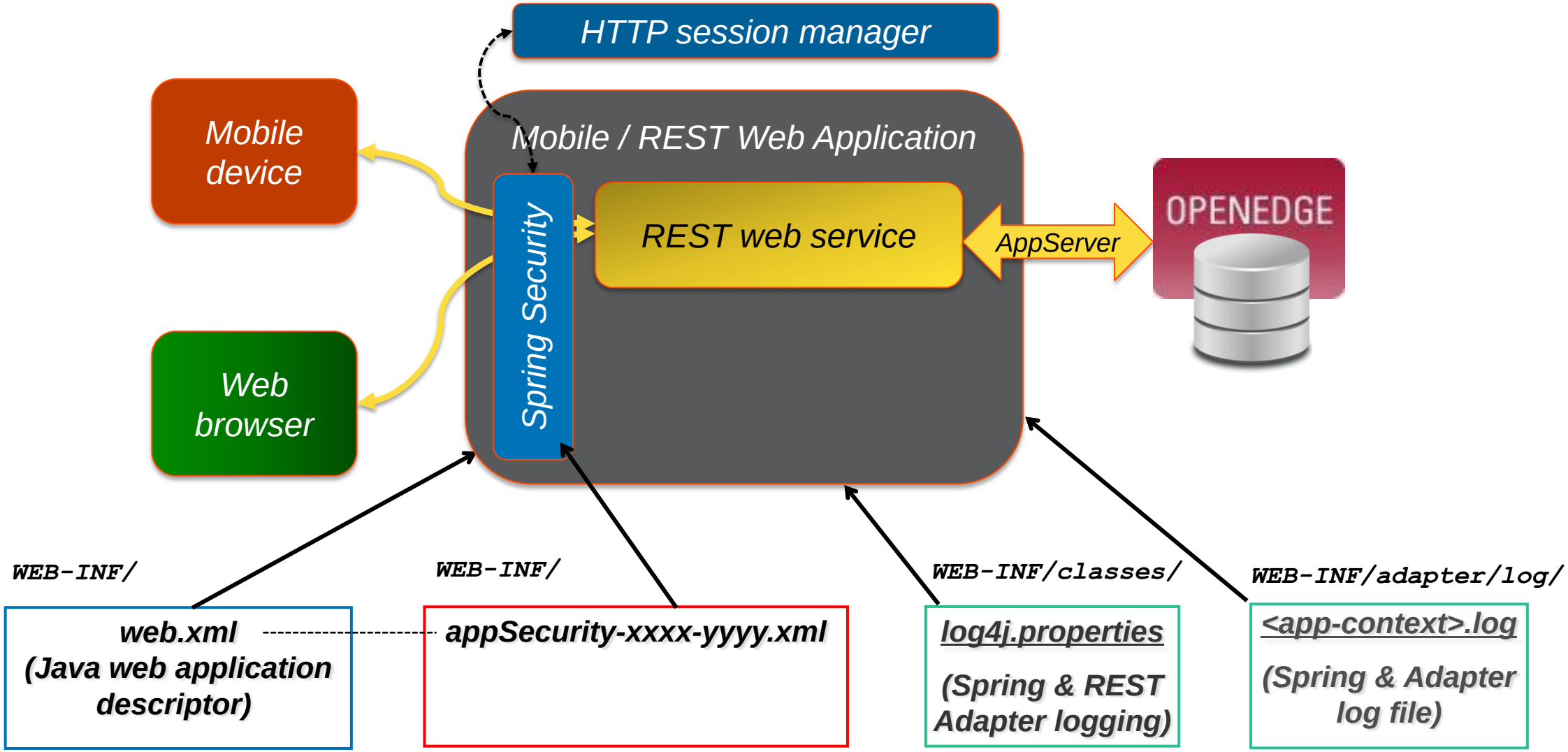
- Your web application will be probed by hackers & bots within 60 seconds
- Design and build security into my web application from day 1
- Use strong perimeter security before accessing business servers
  - Use OWASP web application security guidelines (*www.owasp.org*)
- Use strong, peer reviewed, industry security technologies
- Push identity from perimeter security to back-end servers for application authorization

Anatomy of an OpenEdge Mobile/REST Web Application

- Standard Java web application architecture & functionality
- Spring Security replaces Java container authentication & authorization security
- Combinations of REST api & OpenEdge Mobile components



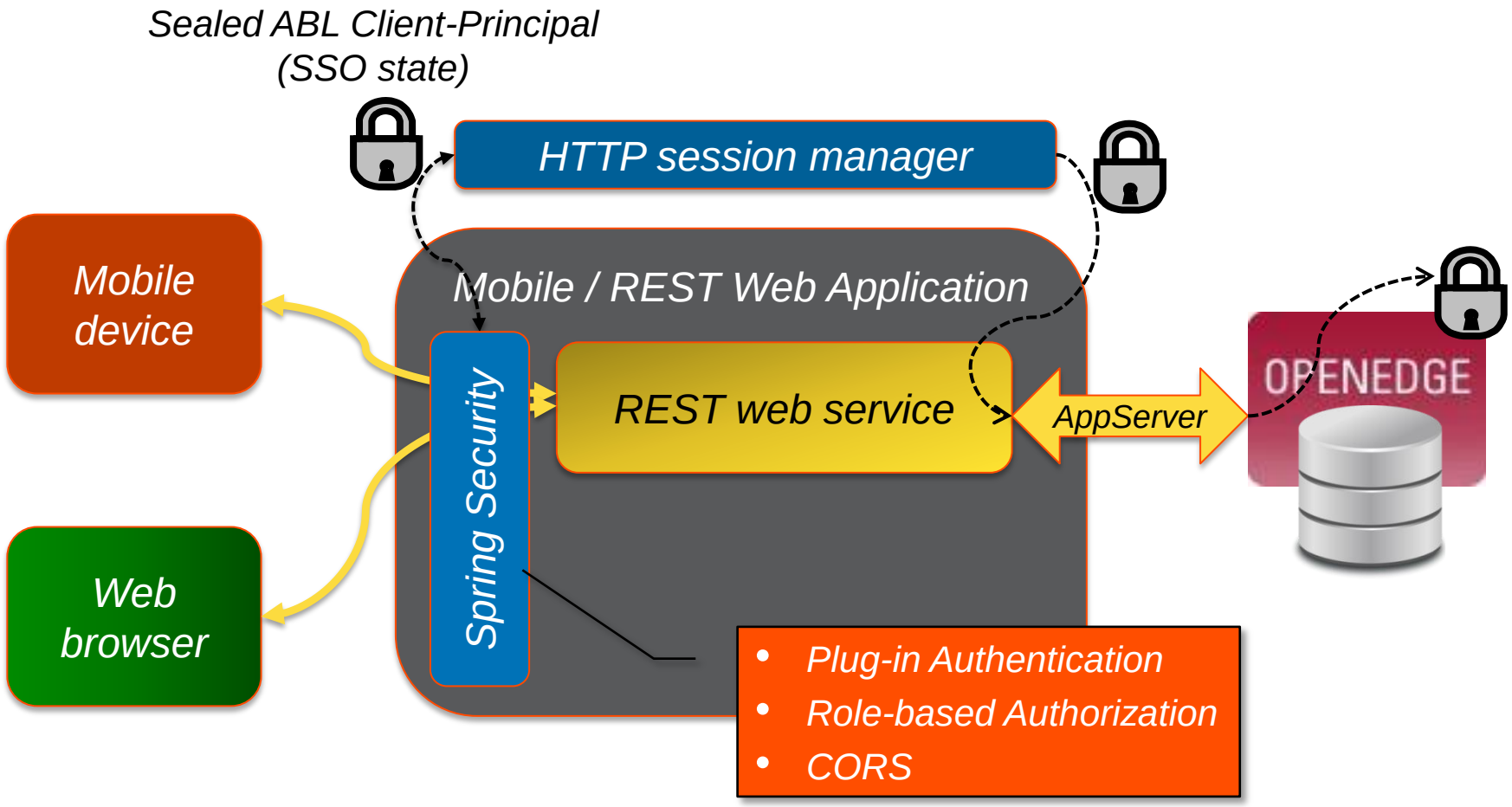
General REST Web Application Architecture



Spring Security Templates

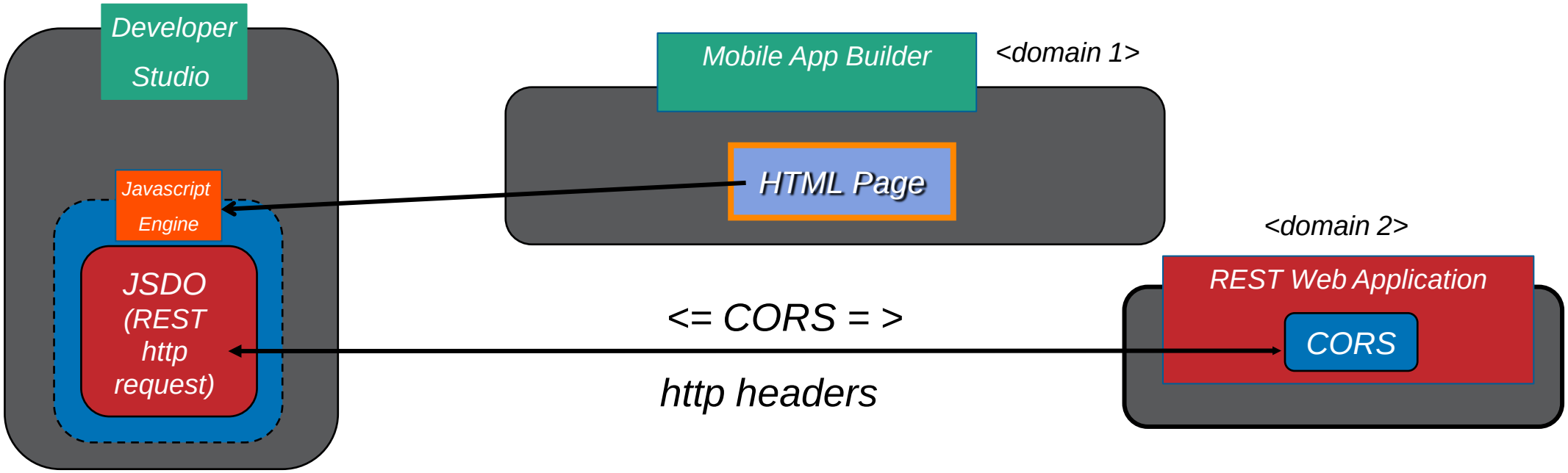
Using an AppServer for Application Level Authorization

- The Spring Security's **authentication** credentials are transformed into a sealed Client-Principal that is accessible via SESSION's REQUEST-INFO object

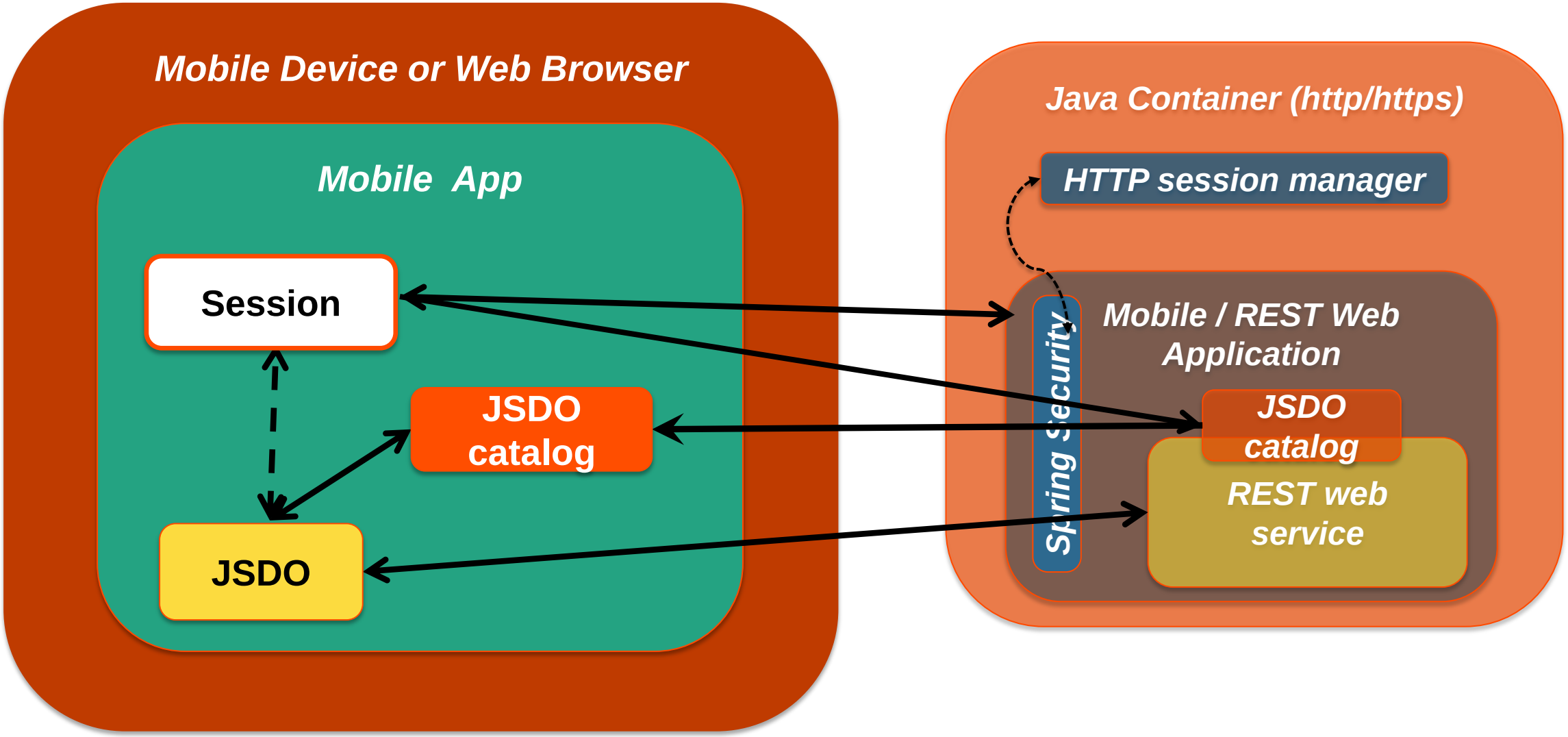


CORS - Cross Origin Resource Scripting

- Javascript engines **always** block resource access to a domain external from the page
- CORS is a W3C group standard that allows *Javascript to access Web application resources in a DNS domain different from the one the current HTTP page and JavaScript were loaded from*
  - CORS works by using HTTP headers that allow servers to grant/deny Javascript resource access to permitted client domains



OpenEdge Mobile Client Architecture

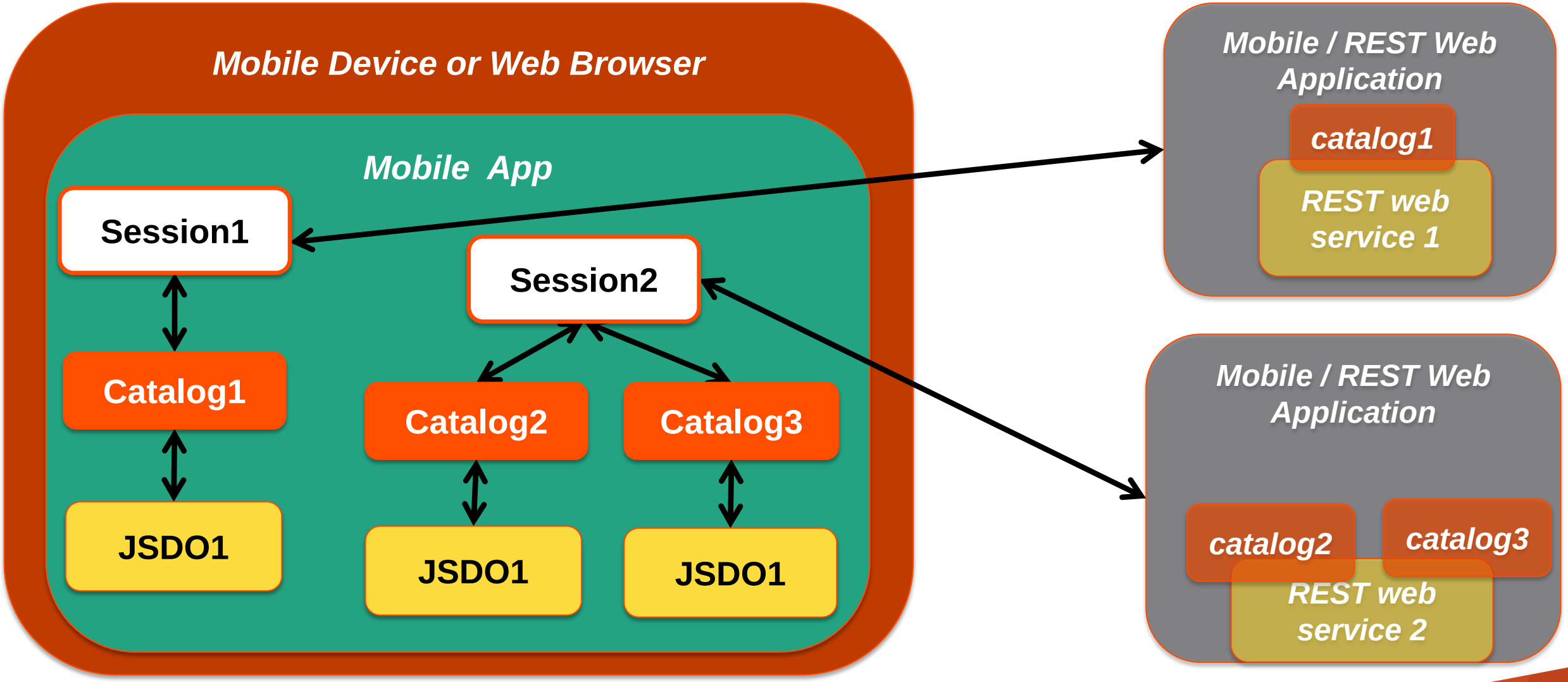


OpenEdge Mobile Client JavaScript Session Object

progress.data.Session :

- Log in to mobile service, sending necessary credentials
- Get and store the JSDO catalog
- Add session and credential information to the requests that a JSDO sends to mobile service
- Log out

OpenEdge Mobile Client Architecture



Agenda

- Architectural basics
  - REST service
  - Mobile client
- Making required choices
  - Authentication model
  - User sessions
  - AppServer SSO
- Diving into the code
  - REST service
  - Mobile client
- What to do when things go sideways

Major Security Technology Decisions

- **Required choices**
  - **The web application's user authentication model**
    (note: NOT where user accounts exist)
  - **The web application's user login session model**
    (provided by the Web Server)
  - ***The web application's role-based authorization***
    (application defined role names or production defined role names)
- ***Secondary choices***
  - ***The web application's CORS configuration***
    (restrict client domain access)
  - ***The web application's AppServer SSO***
    (use Client-Princpals to control access to application and OpenEdge)

** Other authentication models available - not certified*

Web Application Authentication Models

- **Anonymous** — The *no user authentication or login session [default]*
  (NOT recommended for production applications – used for test & debug)

- **HTTP BASIC authentication** — *Client sends base64 encoded user name/password to web application in each http request*
  - HTTP header: *Authorization*
  - *No user login session*
  - *No user logout*

- ***HTTP FORM authentication*** — *The client logs into and out of the web application once per session*
  - HTTP form passed to REST web application for **login** & HTTP session management
  - HTTP cookie returned to client – client echoes cookie for each HTTP request
  - HTTP request used to **logout** from REST web application & delete HTTP session

Choose Your User Login Session Model

- Three different concepts of *session*
  - Mobile client (server connection)
  - Web server user [login] session ← common to all web application clients
  - Application user session
- Only two web application *session* models:
  - **stateless** (BASIC default)
  - **stateful** (FORM default)
- Web servers control user sessions (**not** OpenEdge or your AppServer application)
- Web servers do not share user sessions across web applications
- Client & server web server user session models must ALWAYS agree

OpenEdge Web Application Security Templates

- appSecurity-anonymous.xml
  - Anonymous security - every internet/intranet user is allowed full access
- appSecurity-basic-local.xml
  - HTTP BASIC model using an unsecured user.properties text file
- appSecurity-form-local.xml
  - HTTP FORM model using an unsecured user.properties text file
- appSecurity-container.xml
  - Spring Security SSO from Java container's authentication token
- 11.2.1+
  - appSecurity-basic-ldap.xml
  - appSecurity-form-ldap.xml
- 11.3+
  - appSecurity-basic-oerealm.xml
  - appSecurity-form-oerelam.xml

Agenda

- Architectural basics
  - REST service
  - Mobile client
- Making required choices
  - Authentication model
  - User sessions
  - AppServer SSO
- Diving into the code
  - REST service
  - Mobile client
- What to do when things go sideways

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`**

web.xml — Edited

```
<!-- 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-basic-ldap.xml
            /WEB-INF/appSecurity-form-ldap.xml
            /WEB-INF/appSecurity-basic-oerealm.xml
            /WEB-INF/appSecurity-form-oerealm.xml
            -->
            /WEB-INF/appSecurity-basic-local.xml
        </param-value>
    </context-param>
```

Example: Choosing the Session Management Model

appSecurity-basic-local.xml — Edited

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

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

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

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

HTTP Basic Real name shown to users

Example: User Account Authentication Control

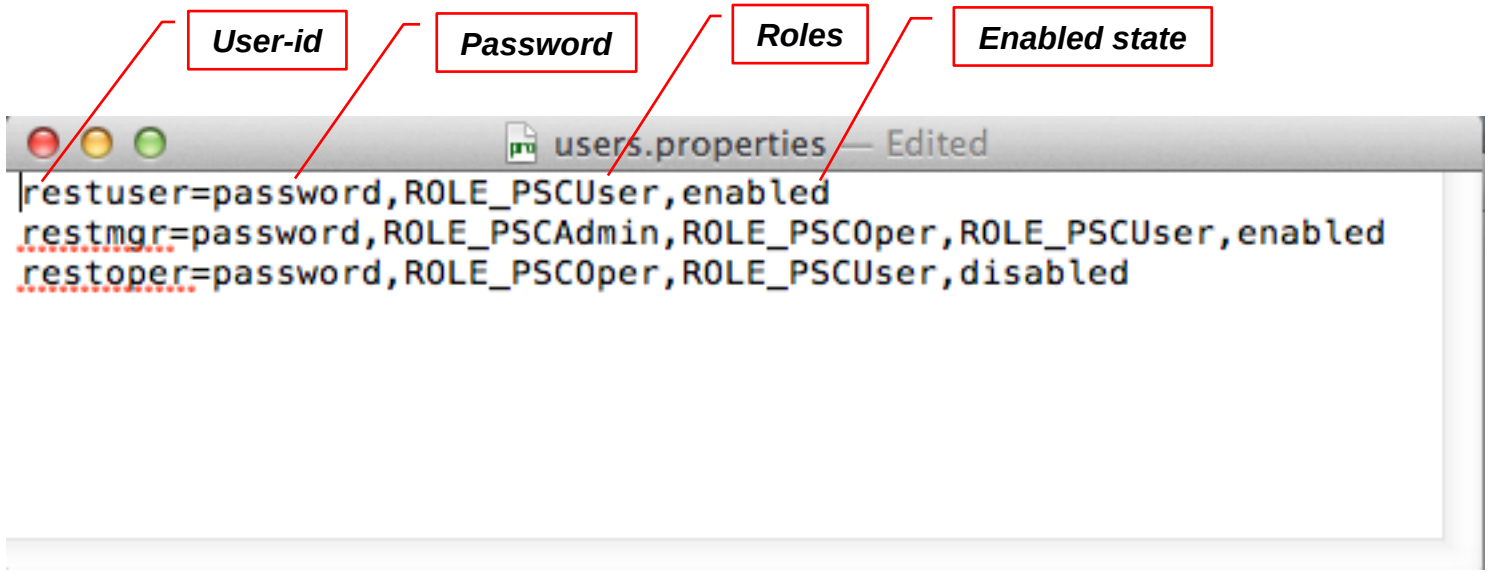
appSecurity-basic-local.xml — Edited

```
<!-- Authentication manager reserved for PUBLIC anonymous authentication
     to the static and dynamic application content.
     Note: If you enable password salting (recommended for production
           installations) you may use the following Java console
           utility to generate new passwords :
             com.progress.rest.security.EncodePassworc
 -->
<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>
```

Username and encoded password and authorities

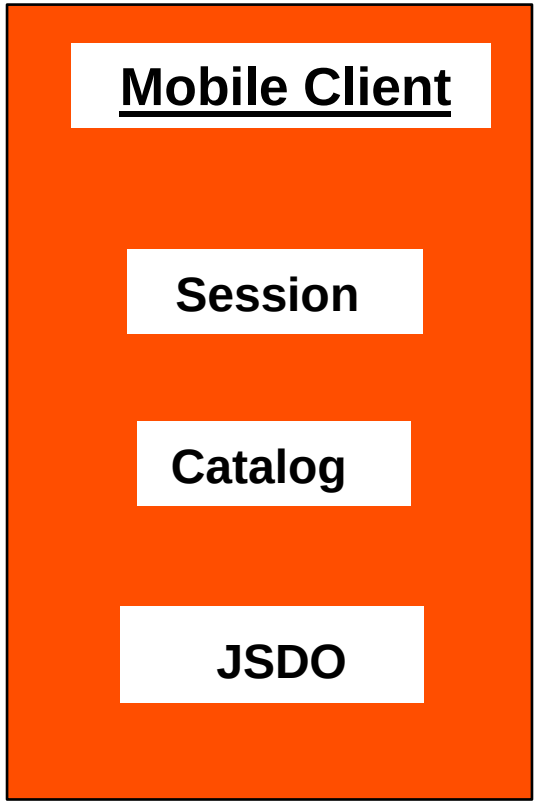
user.properties

- NOT A SECURE SOURCE OF PRODUCTION USER ACCOUNTS
- Simple to maintain source of user roles and roles for testing
- Format: <userid>=<password>,ROLE_<rolename>[,ROLE_...],{enable|disable}
- Clear-text password
- All role names have “ROLE_” prefix (so Spring can distinguish between userids & roles)
- Must restart web application for edits to take affect



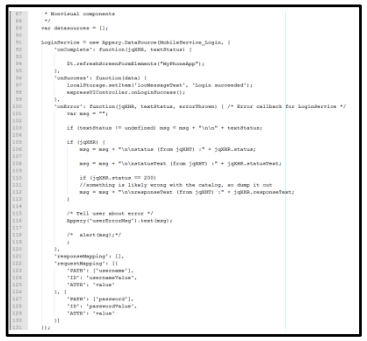
Session API at its simplest

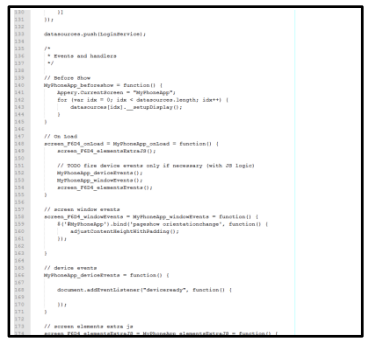
```
pdsession = new progress.data.Session();

pdsession.authenticationModel  =
          progress.data.Session.AUTH_TYPE_FORM;

var loginResult = pdsession.login(serviceURI
                                  [, uname, pw] );

pdsession.addCatalog( catalogURI );

     ( create and use JSDO(s) )

pdsession.logout( );
```

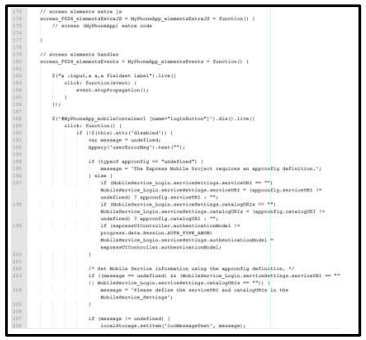


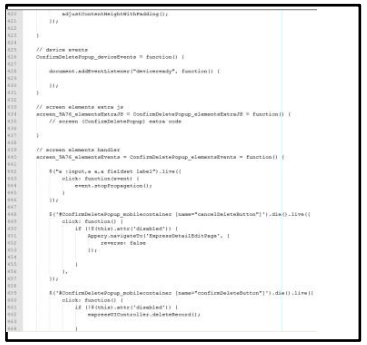
Session API

Using the Session API: The Reality



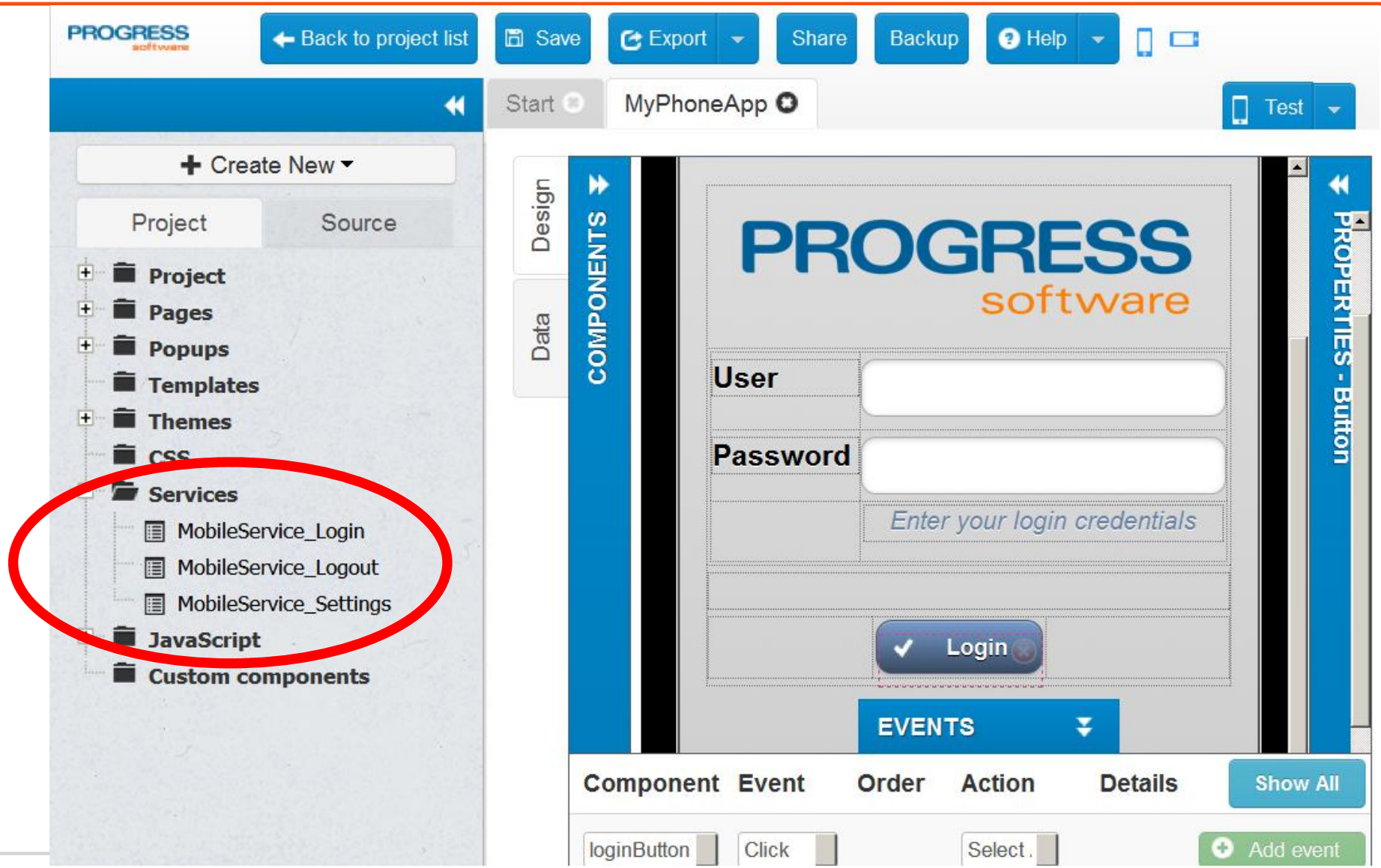






+ 11 more …

Session Services in the Mobile App Builder



Session Services in the Mobile App Builder

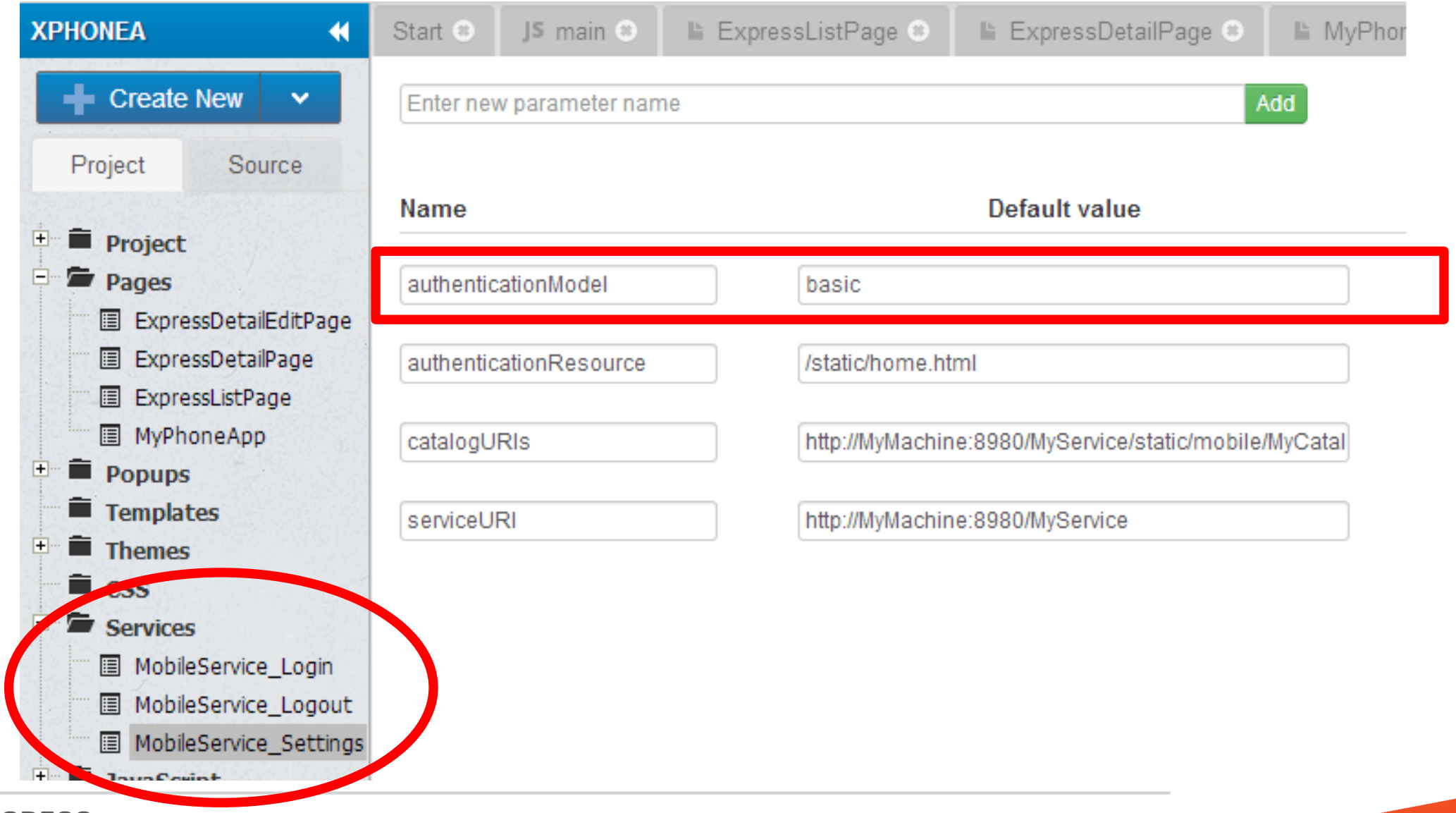
WHAT THE MOBILE APP BUILDER + TEMPLATES DO FOR YOU

- UI Login and Logout buttons
- UI fields for user to enter credentials
- UI fields and Settings values mapped to the calls made to the server
- Event handlers for the Login and Logout buttons
- Error and Success handlers for the Login service

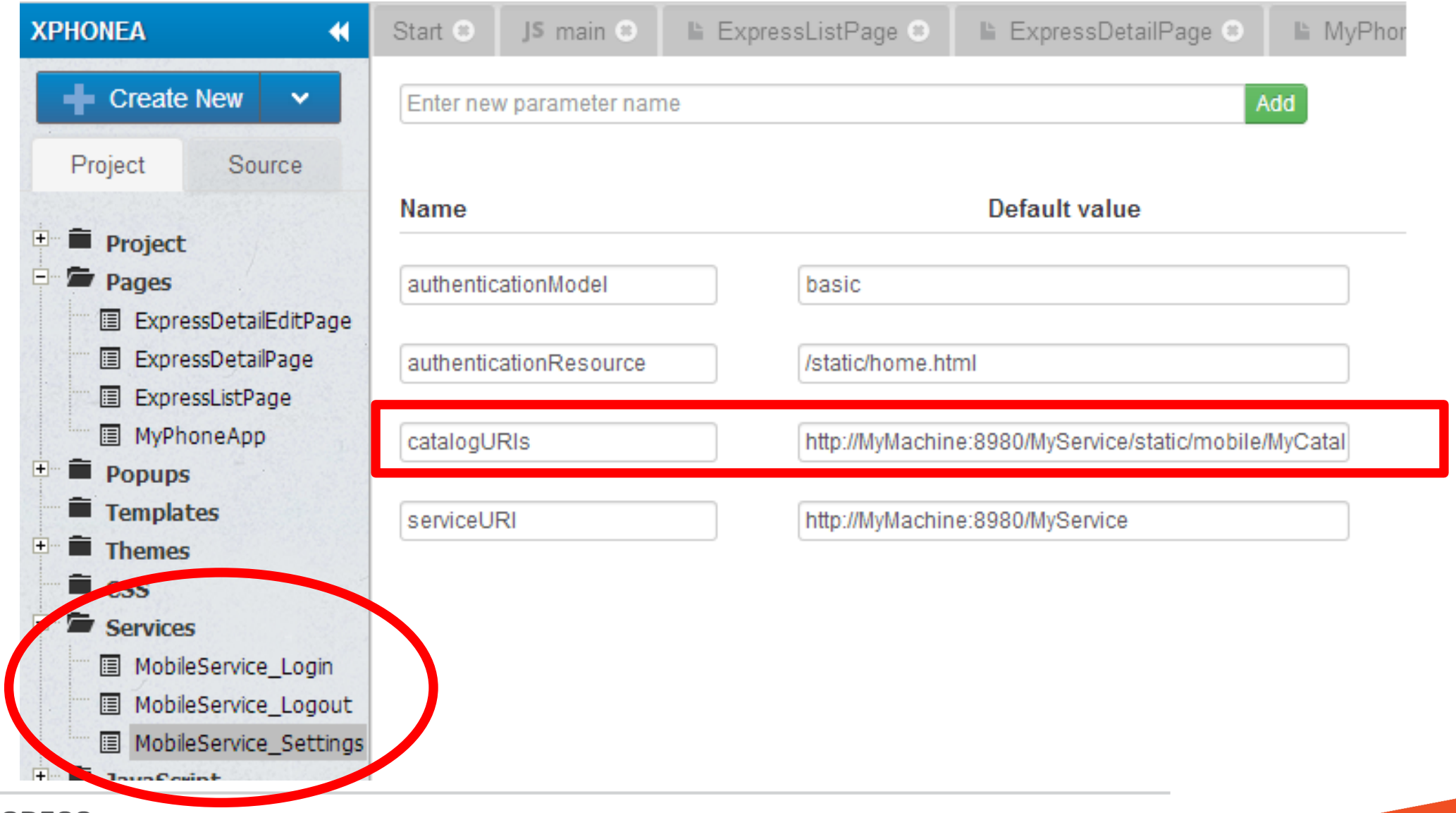
WHAT YOU DO FOR THE MOBILE APP BUILDER

- Define three settings

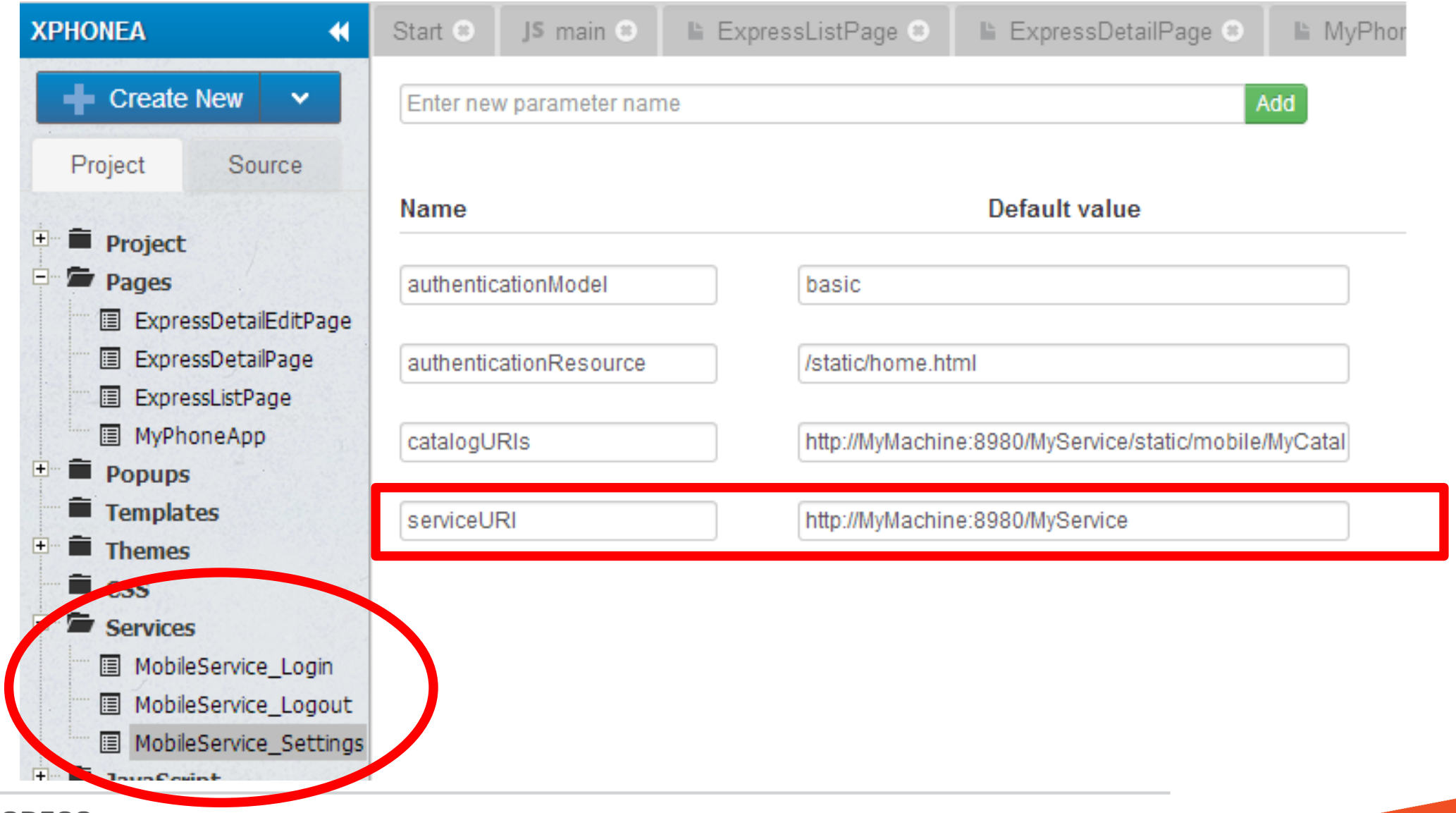
Session Service Settings



Session Service Settings



Session Service Settings



Express template appconfig.js file

```
var appconfig = {
    "catalogURI":
        "http://MyMachine:8980/XPhoneService/static/mobile/MyService.json",
    "serviceURI": "http://MyMachine:8980/MyService",
    "tableName": "Customer",
    "resourceName": "Customer",
    "tableRef": "ttCustomer",
    "listFields": "Name"
};
```

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*

appSecurity-basic-local.xml — Edited

```
    <!-- The security filter that implements the CORS standard for controlling
         cross site resource access by http clients. -->
    <b:bean id="OECORSFilter"
            class="com.progress.rest.security.OECORSFilter" >
            <!-- Examples:
            <b:property name="allowAll" value="false" />
            <b:property name="allowDomains" value="http://
studio.progress.com,http://mobile.progress.com" />
            <b:property name="allowDomains" value="*" />
            <b:property name="allowSubdomains" value="false" />
            <b:property name="allowMethods" value="" />
            <b:property name="messageHeaders" value="" />
            <b:property name="responseHeaders" value="" />
            <b:property name="supportCredentials" value="true" />
            <b:property name="maxAge" value="-1" />
            -->
    </b:bean>
```

OECORSFilter

*allowAll**

*allowDomains**

MessageHeaders

repsonseHeaders

AppServer Single Sign-On

- ClientPrincipal authentication token created from Spring authentication token
- ClientPrincipal passed with each request to Agent
- AppServer client 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 is using Anonymous security model
- ABL Client-Principal SESSION-ID attribute can be zero (0)
  - BASIC authentication with default stateless session model
- Same Client-Principal validation using *domain-name* and *domain-access-code*
- Cannot use with OpenEdge AppServer before 11.2

Example: OpenEdge Client-Principal Single Sign-On

AppServer Single Sign-On

appSecurity-basic-local.xml — Edited

```
<!-- 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="domain" value="sample" />
        <b:property name="roles" value="sample" />
        <b:property name="authz" value="false" />
        <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="true" />
        <b:property name="anonymous" value="true" />
        -->
</b:bean>
```

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

Agenda

- Architectural basics
  - REST service
  - Mobile client
- Making required choices
  - Authentication model
  - User sessions
  - AppServer SSO
- Diving into the code
  - REST service
  - Mobile client
- What to do when things go sideways

Diagnostic Steps: Browser Development Tools

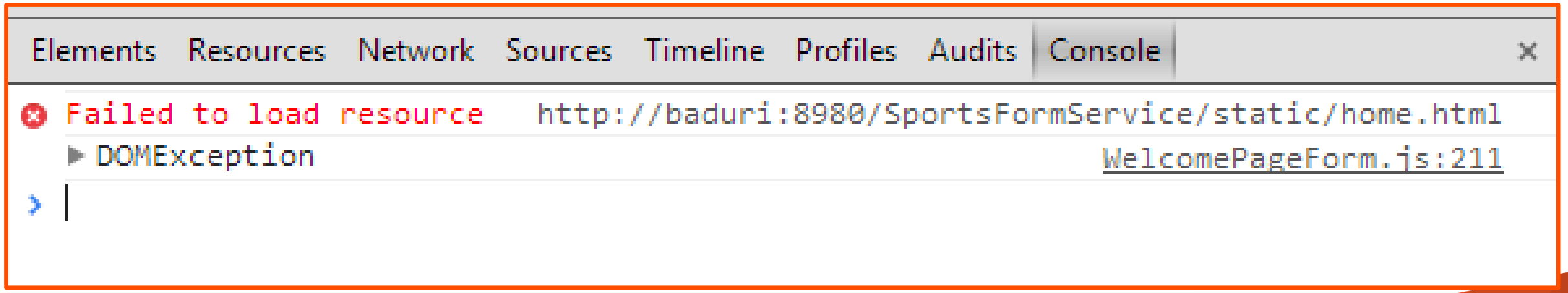
Use browser from Windows or OS-X

- Browser development tools:
  - Console
  - Network traffic
  - Debugger (“Sources”)
  - etc.

Inspect Element
from context menu

Developer Menu
(different places on different browsers)

Shortcut keys (e.g., F12 on Chrome, Firefox)



Diagnostic Steps: Check the Traffic

- Check the HTTP(s) traffic between your application and the server
  - 200
  - 4xx
  - 5xx
  - No request being sent!
  - Server returns 200 but you're getting a NETWORK ERROR on client
- From browser developer tools Network tab
- Standalone HTTP monitor

HTTP Monitors : Fiddler 2 and Other Standalones

Fiddler Web Debugger

File Edit Rules Tools View Help GET /book

Replay Resume Stream Decode Keep: All sessions Any Process Find Save Browse Clear Cache TextWizard Tearoff

#	Result	Protocol	Host	URL
1	401	HTTP	nbbe...	/SportsBasicService/static/home.html
8	304	HTTP	gdx....	/components/game/mlb/year_2013/mor

Statistics Inspectors AutoResponder Composer Filters Log Timeline

Headers TextView WebForms HexView Auth Cookies Raw JSON XML

```
GET http://nbbedwhenshaw.bedford.progress.com:8980/SportsBasicService/static/home.htm
Host: nbbedwhenshaw.bedford.progress.com:8980
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Authorization: Basic ejpx
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko
Accept: */*
DNT: 1
Referer: http://nbbedwhenshaw.bedford.progress.com:8980/SportsApp/static/mobile/Sports
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8
```

Find... (press Ctrl+Enter to highlight all) View in Notepad

Get SyntaxView Transformer Headers TextView ImageView HexView WebView Auth Caching

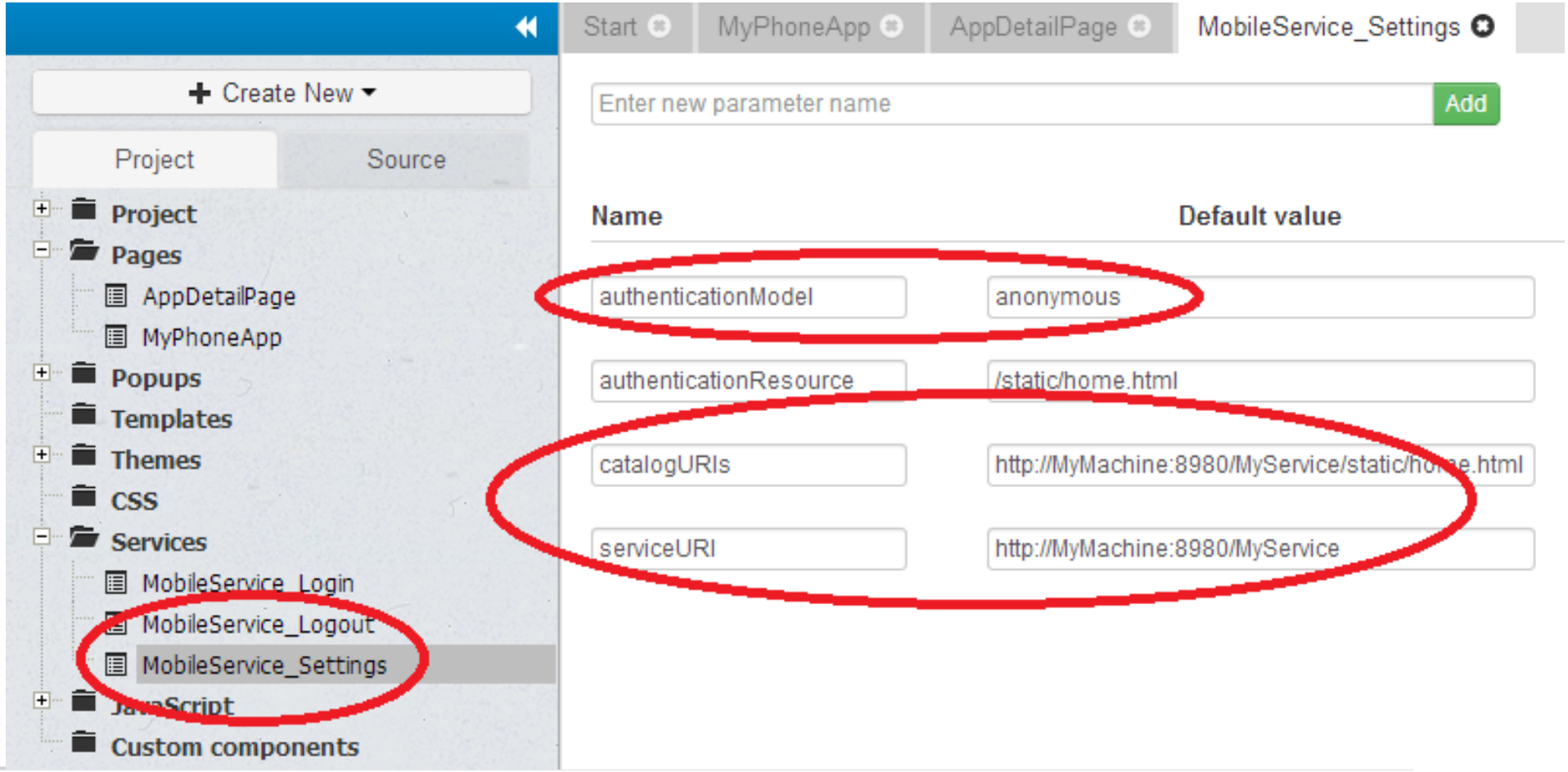
Cookies Raw JSON XML

```
HTTP/1.1 401 Unauthorized
Server: Apache-Coyote/1.1
WWW-Authenticate: Basic realm="REST Application"
Content-Type: text/html
Content-Length: 43
Date: Mon, 16 Sep 2013 17:38:48 GMT

<html> <body>
Unauthorized
</body> </html>
```


PROGRESS

Check Your Settings!

- Especially if you're getting strange errors
  Ex: login failure, getting an internal server error on a GET of /static/home.html



Diagnostic Steps: Where Does It Fail?

Access the service directly from a browser address bar

- Login ?

  http://hostname:port/<webApplicationName>

- Is the catalog accessible?

  http://hostname:port/<webApplicationName>/static/mobile/<catalogFileName>

- Is the REST adapter available?

  http://hostname:port/<webApplicationName>/rest

- Can you get data?

  http://hostname:port/<webApplicationName>/rest/<serviceName>/<resourceName>

  Ex: http://localhost:8980/MobileTestApplication/rest/MobileTestService/Customer

Diagnostic steps: What's Happening on the Back End?

Internal Server Error (HTTP status 5xx), or simply no data

- Is the AppServer running?
- Is the Database running?
- Check logs
- Debug! (see Developer Tools)

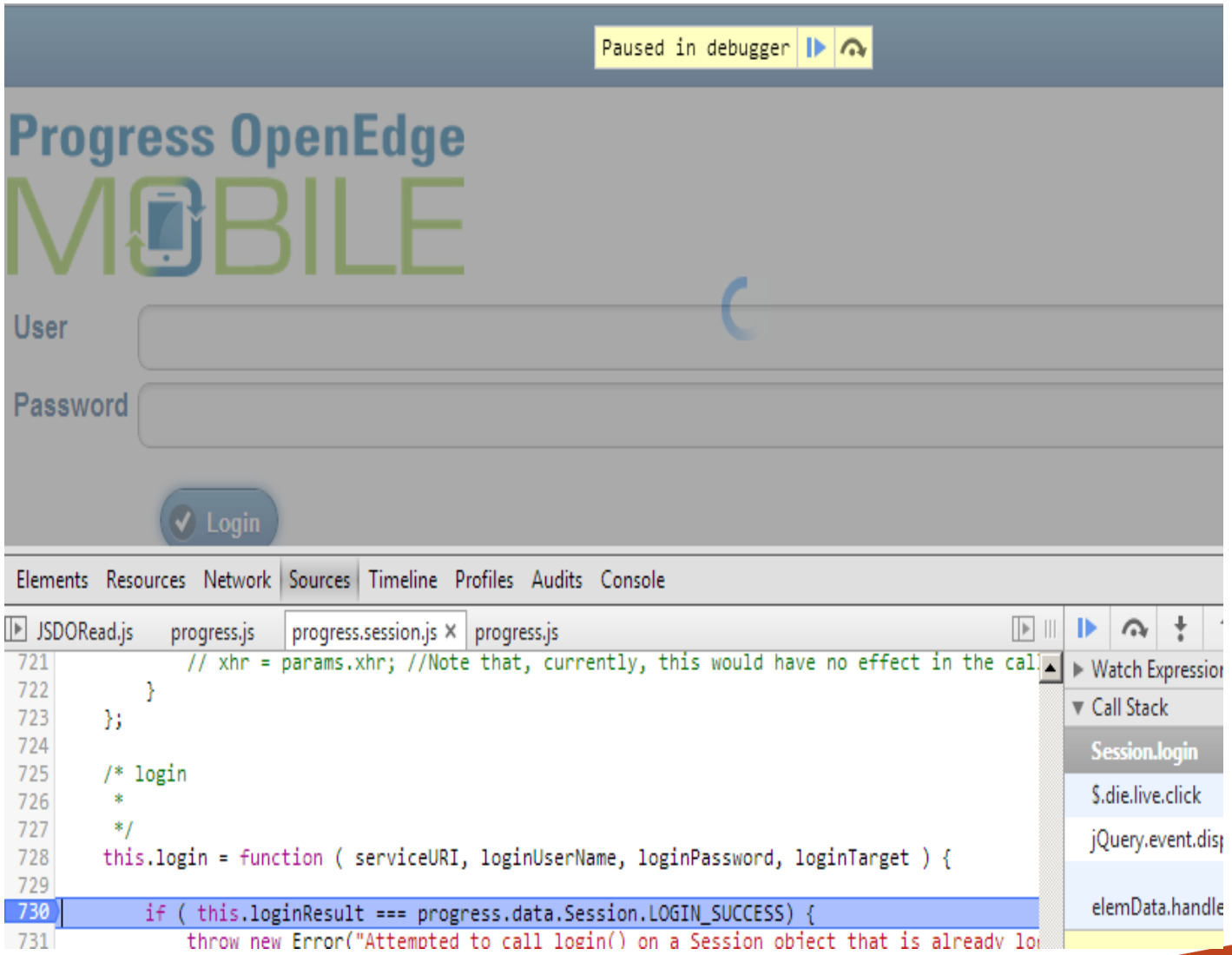
Browser Debugger

Useful breakpoints

progress.session.js

- this.login = function
- this.addCatalog = function
- this._openRequest = function

  (JSDO uses this to prepare requests)



Debugging Apps Running on Devices

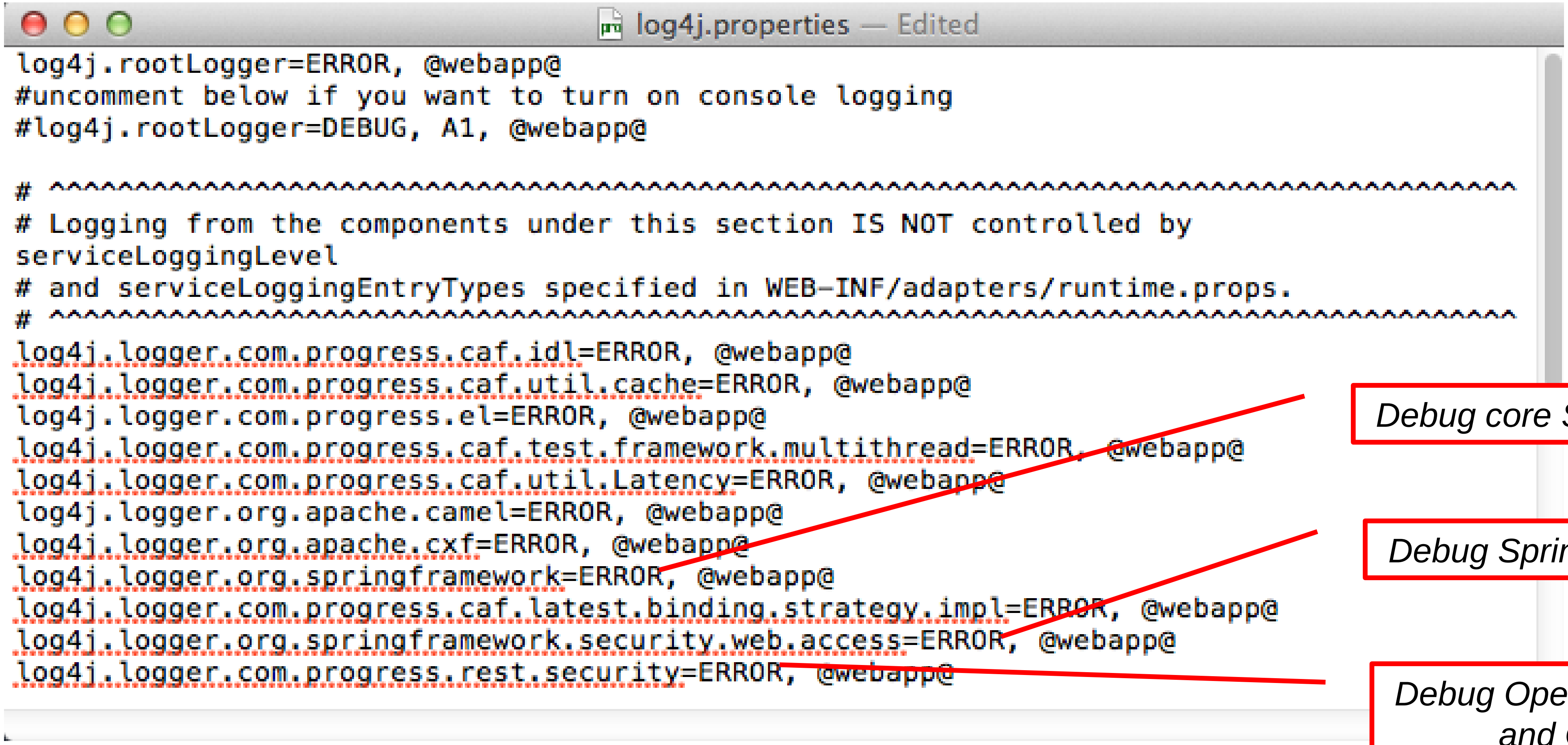
- Try running it in emulator in browser
- Run an HTTP monitor on your computer and set it as a proxy on the device
- Remote debuggers
  - iOS: Web inspector from Safari on OS-X
  - Android: Android Debug Bridge (ADB) through USB connection to computer
  - Weinre (**we**b **in**spector **re**mote)

Logs

- WRKDIR/ (development) or CATALINA_BASE/logs (production)
  - catalina.<date>.log or catalina.out
  - localhost.<date>.log
  - localhost_access_log.<date>.txt
- …/webapps/<web service application>/WEB-INF/adapters/logs
  - <service-name>.log
- AppServer broker logs

Debugging in the REST Adapter

- Edit WEB-INF/classes/log4j.properties
- Change ERROR to DEBUG for these packages:



```
log4j.rootLogger=ERROR, @webapp@
#uncomment below if you want to turn on console logging
#log4j.rootLogger=DEBUG, A1, @webapp@

# ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
# Logging from the components under this section IS NOT controlled by
serviceLoggingLevel
# and serviceLoggingEntryTypes specified in WEB-INF/adapters/runtime.props.
# ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
log4j.logger.com.progress.caf.idl=ERROR, @webapp@
log4j.logger.com.progress.caf.util.cache=ERROR, @webapp@
log4j.logger.com.progress.el=ERROR, @webapp@
log4j.logger.com.progress.caf.test.framework.multithread=ERROR, @webapp@
log4j.logger.com.progress.caf.util.Latency=ERROR, @webapp@
log4j.logger.org.apache.camel=ERROR, @webapp@
log4j.logger.org.apache.cxf=ERROR, @webapp@
log4j.logger.org.springframework=ERROR, @webapp@
log4j.logger.com.progress.caf.latest.binding.strategy.impl=ERROR, @webapp@
log4j.logger.org.springframework.security.web.access=ERROR, @webapp@
log4j.logger.com.progress.rest.security=ERROR, @webapp@
```

Debug core Spring security process

Debug Spring Security authn/authz

Debug OpenEdge Client-Principal and CORS handling

Troubleshooting Document

- For more information see:

  http://communities.progress.com/pcom/people/mcmann?view=overview

Summary

- Authenticate and authorize at the perimeter
- Client and server have to agree on authentication & session model
- Change the code on both client & server before testing
- Integrate OE Realm after local authentication works
- Beware of CORS configurations during initial testing
- Integrate AppServer SSO after the other things are done

PROGRESS