GUIDEWIRE

InsuranceSuite 10 Integration: Essentials

Student Workbook

Labs and Tutorials

Document Published: 2019-10-10

Table of Contents

Introduction

Welcome to the Guidewire InsuranceSuite 10.0 Integration - Essentials course.

The Student Workbook will lead you through the course labs. The lesson numbers correspond to the lesson numbers in your training. Complete the assigned labs to the best of your ability.

Lesson 1

Authentication

This lab requires that you use TrainingApp, ExternalApp, Guidewire Studio, and a supported web browser. Start Guidewire Studio for TrainingApp. Start the server as Debug 'Server'.

Start ExternalApp using the **Start ExternalApp** shortcut.

The default URL for TrainingApp is: http://localhost:8880/ab/ContactManager.do. Log in to TrainingApp as Super User whose login/password is su/gw.

1.1 Configure external authentication service

Succeed Insurance wants to replace the base application's user authentication integration with a custom plugin and associated class that validates authentication information from an external system. The external system in this lab is ExternalApp. ExternalApp will function as an external authentication system like an LDAP system.

1.1.1 Requirements

Spec 1 Create an authentication package named **authentication**.

- Employ Guidewire recommended naming conventions for your package.

Spec 2 Create a Gosu class in the authentication package, named SIAuthenticationServicePlugin that implements gw.plugin.security.AuthenticationServicePlugin.

- The **authenticate** method must:
  - Take a UserNamePasswordAuthenticationSource that contains the http request's username and password parameters (key value pair). If the source is not valid, throw a gw.pl.exception.GWConfigurationException.
  - Supply these values to ExternalApp's User Authentication web service. The web service contains one method — authenticate(username, password) — that returns a Boolean value of true when the values match.
  - The credentials for the web service are:
    - Username = externalappuser
    - Password = gw
  - The setCallback method simply sets the callback handler.
  - If external authentication is successful, returns the public ID for the user. If unsuccessful, throws a javax.security.auth.login.FailedLoginException.

Spec 3 Register new SIAuthenticationServicePlugin class in the AuthenticationServicePlugin registry.

Spec 4 ExternalApp's User web service WSDL URL is:

http://localhost:8890/ab/ws/externalapp/webservice/UserAuthenticationAPI?WSDL

Spec 5 ExternalApp's User Authentication web service authenticates the following key value pairs:

User name	Password
su	**guidewire**
aapplegate	**guidewire**
bbaker	**guidewire**

1.1.2 Tasks

1. **Create a new webservice.authentication package.**
2. **In the package, create a web service collection that connects to ExternalApp's UserAuthentication web service.**
3. **Create a new plugin.authentication package.**
4. **Create a new implementation of the AuthenticationServicePlugin class.**
5. **Edit the existing AuthenticationServicePlugin registry file.**
6. **Deploy code changes.**
7. **Perform verification steps.**

1.1.3 Verification steps

1. **Login into TrainingApp using the username/ password key value of su/guidewire.**
   - Confirm successful login.
   - Log out.
2. **Login into TrainingApp using the username/ password key value of aapplegate /guidewire.**
   - Confirm successful login.
   - Log out.
3. **Login into TrainingApp using the username/ password key value of bbaker/ guidewire.**
   - Confirm successful login.
   - Log out.
4. **Login into TrainingApp using the username/ password key value of su/gw**
   - Confirm failed login.
5. **Login into TrainingApp using the username/ password key value of afakename/guidewire.**
   - Confirm failed login.
6. **Restore TrainingApp to its original AuthenticationServicePlugin.**
   - Right-click on AuthenticationServicePlugin registry in Project view.
   - Select **Revert to Base**.
   - From the Studio menu, **Restart** the server.
   - Verify su/gw log in is successful.

1.1.4 Solution

1. **Create a new webservice.authentication package.**

- Right-click on **gsrc** package and select **New → Package**.
- Enter **si.ta.webservice.authentication** as the new package name.

2. **In the package, create a web service collection that connects to ExternalApp's UserAuthentication web service.**
   - Right-click on the authentication package and select New → Web Service Collection.
   - Enter userauthwsc as the new Web Service Collection name.
   - Click the Add New Resource button.
   - Enter the resource URL:

`http://localhost:8890/ab/ws/externalapp/webservice/UserAuthenticationAPI?WSDL`

   - Click the **Fetch** button.

3. **Create a new plugin.authentication package.**
   - Right-click on **si.ta** package and select **New → Package**.
   - Enter **plugin.authentication** as the new package name.

4. **Create a new implementation of the AuthenticationServicePlugin class.**
   - Right-click on **authentication** package and select **New → Gosu Class**.
   - Enter **SIAuthenticationServicePlugin** as the new class name.
   - Implement the AuthenticationServicePlugin interface.
     - Implement the **authenticate** method.
     - Implement the **Callback** property.

```
package si.ta.plugin.authentication

uses gw.api.system.PLLoggerCategory
uses gw.pl.exception.GWConfigurationException
uses gw.plugin.security.AuthenticationServicePlugin
uses gw.plugin.security.AuthenticationServicePluginCallbackHandler
uses gw.plugin.security.AuthenticationSource
uses gw.plugin.security.UserNamePasswordAuthenticationSource
uses si.ta.webservice.authentication.userauthwsc.userauthenticationapi.UserAuthenticationAPI
uses javax.security.auth.login.FailedLoginException
uses com.guidewire.pl.web.controller.UserDisplayableException

class SIAuthenticationServicePlugin implements AuthenticationServicePlugin {

  private var _handler : AuthenticationServicePluginCallbackHandler

  /**
   * Authenticate user to external authentication system.
   */
  @Param("source", "AuthenticationSource")
  @Returns("Public ID of the authenticated user")
  @Throws(FailedLoginException, "Thrown for a variety of user-failed login reasons")
  @Throws(UserDisplayableException, "Thrown for a variety of system-failed login reasons")
  override function authenticate(source : AuthenticationSource) : String {
    // retrieve username and password from AuthenticationSource
    if (!(source typeis UserNamePasswordAuthenticationSource)) {
      throw new GWConfigurationException("Invalid AuthenticationSource: " + source.getClass())
    } else if (!source.determineSourceComplete()) {
      // if the auth source does not contain all required info
      throw new FailedLoginException("Incomplete AuthenticationSource")
    }
    var castSource = source as UserNamePasswordAuthenticationSource
    var username = castSource.Username
    var password = castSource.Password

    try{
      // query external system for user authentication
```

```
      var api = new UserAuthenticationAPI()
      api.Config.Http.Authentication.Basic.Username = "externalappuser"
      api.Config.Http.Authentication.Basic.Password = "gw"
      api.Config.CallTimeout = 30000
      var authenticated = api.authenticate(username, password)

      // react to external system results
      if (authenticated) {
        if (this._handler == null ) {
          throw new GWConfigurationException("Invalid AuthenticationServicePluginCallbackHandler:
null")
        }
        // lookup the Public ID
        var publicID = _handler.findUser(username)
        if (publicID == null) {
          throw new FailedLoginException("Unauthorized User: " + username)
        } else {
          return publicID
        }
      } else {
        throw new FailedLoginException("Unauthorized User: " + username)
      }
    } catch (f: FailedLoginException) {
      // log (info level for training purposes) & rethrow FailedLoginException
      PLLoggerCategory.SECURITY.info(f.Message)
      throw f
    } catch (t: Throwable) {
      // catch, log as error and wrap all other exceptions
      PLLoggerCategory.SECURITY.error(t.Message)
      throw new UserDisplayableException("Authentication Plugin Failure!\n Please remain calm,
the authorities are on their way.")
    }
  }

  override property set Callback(callbackHandler : AuthenticationServicePluginCallbackHandler) {
    _handler = callbackHandler
  }
}
```

5. Edit the existing AuthenticationServicePlugin registry file.

- Navigate to configuration.config.Plugins.registry.
- Open AuthenticationServicePlugin registry.
- Remove the existing configured plugin by clicking the minus symbol..
- Add a Gosu Plugin by clicking the plus symbol.
- Define the Gosu plugin.

6. Deploy code changes.

- From the Studio menu, **Restart** the server.

7. Perform verification steps.

Lesson 2

Batch Processes

This lab requires that you use TrainingApp, Guidewire Studio, and a supported web browser. Start Guidewire Studio for TrainingApp. Start the server as Debug 'Server'.

The default URL for TrainingApp is: http://localhost:8880/ab/ContactManager.do. Log in to TrainingApp as Super User whose login/password is su/gw.

2.1 Create a custom batch process

Succeed Insurance needs a batch process that flags a contact if they are missing an assigned business user.

2.1.1 Requirements

Spec 1 Create a batch process that creates a flag entry when a contact does not have a user assigned to them.

Spec 2 If an identified contact already has an open flag entry for this reason, the batch process does not create an additional flag entry.

Spec 3 Create a custom FlagEntryReason typecode to support this use case:

- code = contactnoassigneduser
- desc = Contact missing assigned user.
- name = Contact missing assigned user.

Spec 4 Create a custom BatchProcessType typecode to support this use case:

- code = contactnoassigneduser
- desc = Contact missing assigned user.
- name = Contact missing assigned user.

Spec 5 Must be schedulable and executable from the UI batch process screen.

Spec 6 Must be able to stop the batch process during processing.

Spec 7 The solution should consider that thousands of contacts could conceivably be added during a single day. It is critical that this process executes successfully without consuming system resources unnecessarily. Re-runs in this environment are particularly costly and inconvenient.

Spec 8 Batch process package called batch.flagcontactnoassigneduser.

Spec 9 Batch process class called FlagContactNoAssignedUserBatch.

2.1.2 Tasks

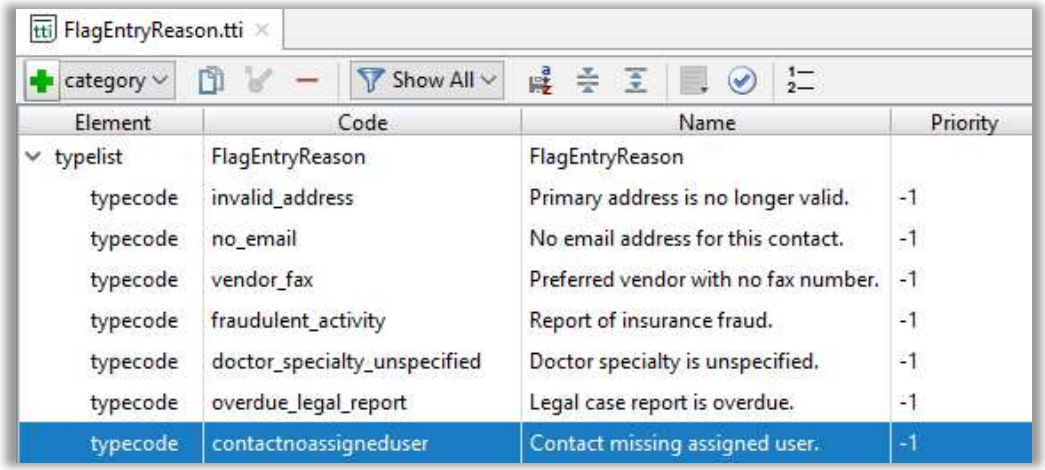
1. **Create new typecode in FlagEntryReason typelist.**
2. **Create new typecode in BatchProcessType typelist and add categories.**
3. **Create new batch process package.**
4. **Create batch process class that implements the specifications.**
5. **Modify the ProcessesPlugin plugin found under trn.ta.batch.ProcessPlugin.**
6. **Deploy code changes.**
7. **Perform verification steps.**

2.1.3 Verification steps

1. **Login into TrainingApp.**
   - Setup test contact cases by:
     - Setting the assigned user field for one contact
     - Not setting the assigned user field for the other contact
2. **Run the batch process manually.**
   - ALT + SHIFT + T
   - Click the **Run** button.
3. **Verify expected results.**
   - Find a contact where the assigned user is not set.
   - A flag entry should exist.
4. **Run the batch process again.**
5. **Verify expected results.**
   - No duplicate flags should be set.

2.1.4 Solution

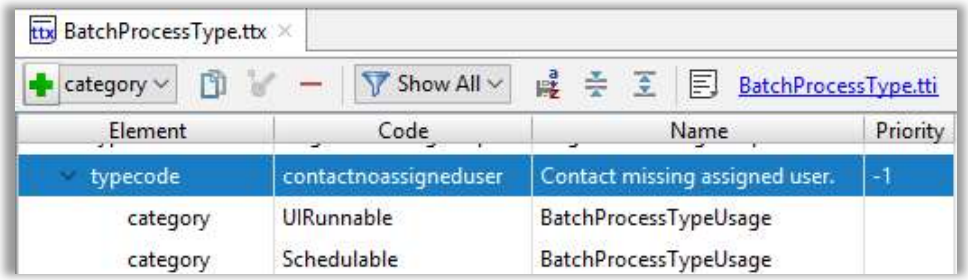
1. **Create new typecode in FlagEntryReason typelist.**
   - Open **FlagEntryReason** typelist.
   - Add new typecode.



Element	Code	Name	Priority
typelist	FlagEntryReason	FlagEntryReason	
typecode	invalid_address	Primary address is no longer valid.	-1
typecode	no_email	No email address for this contact.	-1
typecode	vendor_fax	Preferred vendor with no fax number.	-1
typecode	fraudulent_activity	Report of insurance fraud.	-1
typecode	doctor_specialty_unspecified	Doctor specialty is unspecified.	-1
typecode	overdue_legal_report	Legal case report is overdue.	-1
typecode	contactnoassigneduser	Contact missing assigned user.	-1

2. Create new typecode in BatchProcessType typelist and add categories.

- Open BatchProcessType typelist.
- Add new typecode.



Element	Code	Name	Priority
typecode	contactnoassigneduser	Contact missing assigned user.	-1
category	UIRunnable	BatchProcessTypeUsage	
category	Schedulable	BatchProcessTypeUsage	

3. Create new batch process package.

- Right-click on **si.ta** package and select **New → Package**.
- Enter **batch.contactnoassigneduserflag** as the new package name.

4. Create batch process class that implements the specifications.

- Right-click on **contactnoassigneduserflag** package and select **New → Gosu Class**.
- Enter **FlagContactNoAssignedUserBatch** as the new class name.
  - Extend the **BatchProcessBase** class.
  - Add constructor.
  - Configure the doWork method.

```
package si.ta.batch.flagcontactnoassigneduser

uses com.google.common.collect.Iterables
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.api.util.DateUtil
uses gw.processes.BatchProcessBase
uses gw.transaction.Transaction

class FlagContactNoAssignedUserBatch extends BatchProcessBase{

  construct() {
    super(BatchProcessType.TC_CONTACTNOASSIGNEDUSER);
  }
```

```
  override function requestTermination() : boolean {
    super.requestTermination()
    // updates the built-in TerminateRequested flag to true
    // to indicate a termination request will be evaluated in doWork()
    return true
  }

  override function doWork() : void {
    // construct the main query selecting all contacts with no assigned user
    var mainQry = Query.make(ABContact)
    mainQry.compare(ABContact#AssignedUser, Relop.Equals, null)

    // construct a subquery finding all open FlagEntries for a given Reason
    var subQry = Query.make(FlagEntry)
    subQry.compare(FlagEntry#IsOpen, Equals, true)
    subQry.compare(FlagEntry#Reason, Equals, FlagEntryReason.TC_CONTACTNOASSIGNEDUSER)

    // join the main and sub queries via subselect to only retrieve eligible persons
    var results = mainQry.subselect(ABContact#ID, CompareNotIn, subQry,
FlagEntry#ABContact).select()
    this.OperationsExpected = results.Count

    //feed the result set in manageable chunks
    this.setChunkingById(results, 1000)

    //split each chunk into smaller bundle-sized partitions
    var partitions = Iterables.partition(results, 100)

    for (eachPartition in partitions) {
      // check the built-in TerminateRequested flag at the start of each partition
      if (this.TerminateRequested) {
        break
      } else {
        Transaction.runWithNewBundle(\trn -> {
          for (eachContact in eachPartition) {
            // check the built-in TerminateRequested flag for each person iteration
            if (this.TerminateRequested) {
              break
            } else {
              // -- the app server side FlagEntries array filtering would also work
              // -- but very inefficient compared to the database subselect used above
              //if (!eachPerson.FlagEntries.hasMatch(\currentEntry ->
              //    currentEntry.IsOpen and currentEntry.Reason ==
FlagEntryReason.TC_NOEMAILORPHONE)) {
              var newFlagEntry = new FlagEntry()
              newFlagEntry.Reason = FlagEntryReason.TC_CONTACTNOASSIGNEDUSER
              newFlagEntry.FlagDate = DateUtil.currentDate()
              eachContact.addToFlagEntries(newFlagEntry)
              this.incrementOperationsCompleted()
              //}
            }
          }
        }, "su")
        // identifies the user to use when committing changes, required
        // for batch processes that are scheduled and have no inherent user
      }
    }
  }
}
```

5. Modify the ProcessesPlugin plugin found under trn.ta.batch.ProcessPlugin.

```
package trn.ta.batch

uses gw.plugin.processing.IProcessesPlugin
uses gw.processes.BatchProcess
uses si.ta.batch.flagcontactnoassigneduser.FlagContactNoAssignedUserBatch
uses trn.ta.batch.legalreport.FlagOverdueLegalReportsBatch

class ProcessesPlugin implements IProcessesPlugin {
```

```
  override function createBatchProcess(type : BatchProcessType, arguments : Object[]) :
BatchProcess {
    switch(type) {
      case BatchProcessType.TC_FLAGOVERDUELEGALREPORTS:
          return new FlagOverdueLegalReportsBatch()
      case BatchProcessType.TC_CONTACTNOASSIGNEDUSER:
        return new FlagContactNoAssignedUserBatch()
      default:
        return null
    }
  }
}
```

6. **Deploy code changes.**
   - From the Studio menu, **Restart** the server.
7. **Perform verification steps.**

Lesson 3

Document Management

This lab requires that you use ClaimCenter, Guidewire Studio, and a supported web browser. Start Guidewire Studio for ClaimCenter. Start the server as Debug 'Server'.

The default URL for ClaimCenter is: http://localhost:8080/cc/ClaimCenter.do. Log in to ClaimCenter as Super User whose login/password is su/gw.

3.1 Analyze Alternate OOTB DMS implementation

Succeed Insurance needs to understand the alternate DMS implementation to provide business user a proof-of-concept demonstration.

3.1.1 Requirements

Spec 1 Implement and enable alternate DMS implementation.

Spec 2 Set breakpoints to analyze document process flow.

3.1.2 Tasks

1. **Implement and enable IDocumentMetadataSource plugin using LocalDocumentMetadataSource Gosu class.**
2. **Implement and enable IDocumentContentSource plugin using LocalDocumentContentSource Gosu class.**
3. **Deploy code changes.**
4. **Set breakpoints to analyze document process flow:**
   - BaseLocalDocumentMetadataSource class
     - Line 105 – removeDocument method
     - Line 123 – saveDocument method
   - LocalDocumentContentSource class
     - Line 38 – addDocument method
     - Line 65 – updateDocument method
     - Line 70 – removeDocument method

3.1.3 Verification steps

1. **Log in to ClaimCenter.**
2. **Search for and select Ray Newton claim.**
   - Click Search tab.
   - Enter **Newton** in Last name search field.
   - Click **Search** button.
   - Click on claim link 235-53-365870.

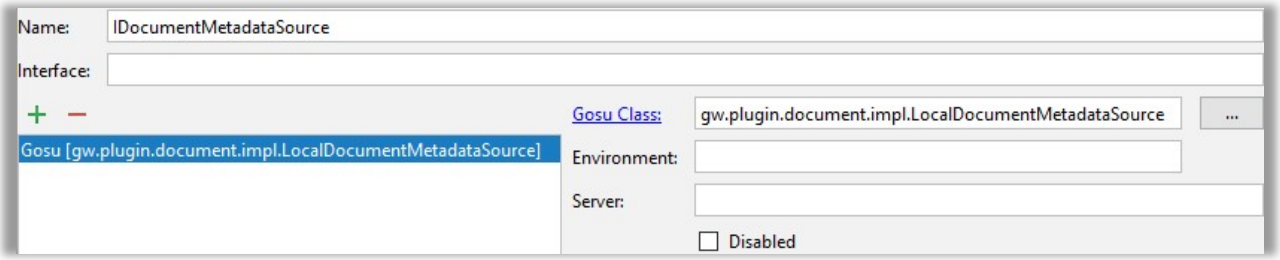
3. **Navigate to Documents page.**
   - Click on Documents link.
   - No documents are found.
4. **Add a new document.**
   - Click **NewDocument → Create from a template**.
   - In Document Template field, click Search… icon.
   - Select Excel Sample template.
   - Click Create Document button.
   - Click Update button.
     - The system breaks at line 38, addDocument method, in the LocalDocumentContentSource class. The method returns **false**, which means the **IDocumentMetadataSource** plugin is responsible for saving the metadata.
     - Click **Resume Program (F9)** in Studio.
     - The system breaks at line 123, saveDocument method, in the BaseLocalDocumentMetadatSource class. The method calls the storeDocument method to complete the metadata save process.
     - Click **Resume Program (F9)** in Studio.
5. **Locate and open metadata XML file in local file system.**
   - Open **File Explorer** in Windows.
   - Navigate to Windows User Temp folder.
     - C:\Users\<user>\AppData\Local\Temp
   - The metadata XML file is in the jetty…dir\metadata folder.
   - Open the metadata XML file and verify it corresponds to the document just added.
6. **Locate the contents file in local file system.**
   - Navigate to Windows User Temp folder.
     - C:\Users\<user>\AppData\Local\Temp
   - The content file is in the Guidewire_Server folder structure.
   - Navigate to the Excel Sample file.
7. **Update the Excel Sample document.**
   - Click the **Download document** icon.
   - Open the Excel file.
   - Make a change to the document and **Save As** to the Desktop.
   - Close the Excel file.
   - Click the **Upload document** icon.
   - Browse to the updated file on the Desktop.
   - Click the **Update** button.
     - The system breaks at line 65, updateDocument method, in the LocalDocumentContentSource class. The method will update the content file. The method returns **false**, which means the **IDocumentMetadataSource** plugin is responsible for saving the metadata.
     - Click **Resume Program (F9)** in Studio.
     - The system breaks at line 123, saveDocument method, in the BaseLocalDocumentMetadataSource class. The method calls the storeDocument method to complete the metadata save process.

- Click **Resume Program (F9)** in Studio.

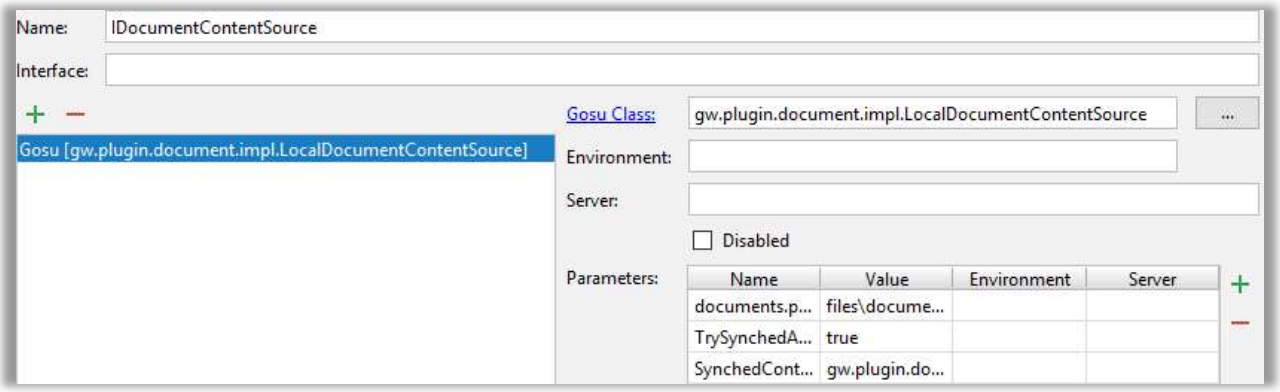
8. **View the updated document.**
   - Click the **Excel Sample** link to view the document.
   - Open the Excel Sample file.
     - Note the updated document is displayed.
   - Close the Excel file.
9. **Remove the Excel Sample document.**
   - Click the **Delete document from claim** icon.
   - Click **OK** to prompt.
     - The system breaks at line 70, removeDocument method, in the LocalDocumentContentSource class. The method deletes the content file from the system. The method returns **false**, which means the **IDocumentMetadataSource** plugin is responsible for deleting the metadata.
     - Click **Resume Program (F9)** in Studio.
     - The system breaks at line 105, removeDocument method, in the BaseLocalDocumentMetadataSource class. The method deletes the metadata file.
     - Click ***Resume Program (F9)*** in Studio.
   - The document is removed from the UI and local file system.

3.1.4 Solution

1. **Implement and enable IDocumentMetadataSource plugin using LocalDocumentMetadataSource Gosu class.**



2. **Implement and enable IDocumentContentSource plugin using LocalDocumentContentSource Gosu class.**



3. **Deploy code changes.**
   - From the Studio menu, **Restart** the server.
4. **Set breakpoints to analyze document process flow:**
   - BaseLocalDocumentMetadataSource class
     - Line 105 – removeDocument method
     - Line 123 – saveDocument method
   - LocalDocumentContentSource class
     - Line 38 – addDocument method
     - Line 65 – updateDocument method
     - Line 70 – removeDocument method