### Introduction

This exercise guide contains all of the training exercises for the PolicyCore Developer Bootcamp course from EIS University. To complete the exercises in this document, you will need access to your personal training environment. If you did not receive information on your training environment, please consult with the course instructor.

### Course Goals

The purpose of this course is to familiarize EIS Suite developers with the product development process. At the end of the course, you will have the basic knowledge necessary to begin participating in your first PolicyCore implementation project.

### Audience

PolicyCore Developer Bootcamp is intended for:

* EIS Suite Developers
* EIS Suite Architects

**Contents**

[Introduction 1](#_Toc511758646)

[Course Goals 1](#_Toc511758647)

[Audience 1](#_Toc511758648)

[Module 1: Introduction to Product Development 4](#_Toc511758649)

[Exercise 1:1 – Importing a Product 4](#_Toc511758650)

[Scenario 4](#_Toc511758651)

[Process 4](#_Toc511758652)

[Module 1: Introduction to Product Development 6](#_Toc511758653)

[Exercise 1:2 – Creating a Customer 6](#_Toc511758654)

[Scenario 6](#_Toc511758655)

[Process 6](#_Toc511758656)

[Module 1: Introduction to Product Development 8](#_Toc511758657)

[Exercise 1:3 – Rating a Quote 8](#_Toc511758658)

[Scenario 8](#_Toc511758659)

[Process 8](#_Toc511758660)

[Expected Result 8](#_Toc511758661)

[Module 2: Product Factory User Interface 10](#_Toc511758662)

[Exercise 2:1 – Deactivating a Product 10](#_Toc511758663)

[Scenario 10](#_Toc511758664)

[Process 10](#_Toc511758665)

[Module 3: Creating New Components 11](#_Toc511758666)

[Exercise 3:1 – Creating a Persistent Component 11](#_Toc511758667)

[Scenario 11](#_Toc511758668)

[Process 11](#_Toc511758669)

[Module 3: Creating New Components 18](#_Toc511758670)

[Exercise 3:2 – Extending an Existing Component 18](#_Toc511758671)

[Scenario 18](#_Toc511758672)

[Process 18](#_Toc511758673)

[Module 4: Enhancing the User Interface 25](#_Toc511758674)

[Exercise 4:1 – Creating a Non-Persistent Component With a Custom View 25](#_Toc511758675)

[Scenario 25](#_Toc511758676)

[Process 25](#_Toc511758677)

[Module 5: Using Product Factory Rules 32](#_Toc511758678)

[Exercise 5:1 – Adding an Assertion Rule 32](#_Toc511758679)

[Scenario 32](#_Toc511758680)

[Process 32](#_Toc511758681)

[Solutions 34](#_Toc511758682)

[Module 5: Using Product Factory Rules 35](#_Toc511758683)

[Exercise 5:2 – Adding a Default Value Rule 35](#_Toc511758684)

[Scenario 35](#_Toc511758685)

[Process 35](#_Toc511758686)

[Solutions 37](#_Toc511758687)

[Module 5: Using Product Factory Rules 38](#_Toc511758688)

[Exercise 5:3 – Adding a Field Visibility Rule 38](#_Toc511758689)

[Scenario 38](#_Toc511758690)

[Process 38](#_Toc511758691)

[Solution 40](#_Toc511758692)

[Module 5: Using Product Factory Rules 41](#_Toc511758693)

[Exercise 5:4 – Adding a Component Applicability Rule 41](#_Toc511758694)

[Scenario 41](#_Toc511758695)

[Process 41](#_Toc511758696)

[Solution 42](#_Toc511758697)

[Module 7: Developing Product-Specific Services 43](#_Toc511758698)

[Exercise 7:1 – Customizing the Issue Action 43](#_Toc511758699)

[Scenario 43](#_Toc511758700)

[Process 43](#_Toc511758701)

[Module 8: Adding Endorsement Forms 46](#_Toc511758702)

[Exercise 8:1 – Using the DataObjectManager Component to Manage Endorsement Forms 46](#_Toc511758703)

[Scenario 46](#_Toc511758704)

[Process 46](#_Toc511758705)

[Module 9: Using PolicyCore Platform Services 48](#_Toc511758706)

[Exercise 9:1 – Creating a New Lookup and Drop-Down Box 48](#_Toc511758707)

[Scenario 48](#_Toc511758708)

[Process 48](#_Toc511758709)

|  |
| --- |
| Module 1: Introduction to Product DevelopmentExercise 1:1 – Importing a Product |

|  |  |
| --- | --- |
| Scenario | In this exercise, you will import a product into Product Factory. This is the product that you will use throughout the rest of the exercises in this course. |

### Process

1. In the main application, navigate to the Solution Administration tool by clicking the **Admin** link in the top right corner of the screen.
2. Select the **Product** tab.
3. Click **Search** without entering anything in the search fields. There shouldn’t be any products returned from the search.
4. Click on the **Import** button.
5. Click **Choose File.** Navigate to the following directory:

D:\new-training-pwc\products\training\training-product-deploy\src\main\product

Select the file **PREC-HO** and select **Open**.

1. Click **Import**.
2. The product’s home screen will display. Click on the lock button in the lower right corner to activate the product.
3. Click on the menu button in the top left corner.
4. Click on the dropdown at the top of the menu and select **Product search** to return to the product search screen.

|  |
| --- |
| Module 1: Introduction to Product DevelopmentExercise 1:2 – Creating a Customer |

|  |  |
| --- | --- |
| Scenario | In this exercise, you will act as a broker for an insurance company. You have been asked to create a new customer in the system. |

### Process

1. Launch the EIS Suite and log in.
2. Click on the **Search+** button at the top of the screen, next to the search field. This will send you to the Search+ screen. The Customer tab of the Search+ screen is open by default.
3. On the Customer tab of the Search+ screen, click on the **Create Customer** button below the search form.
4. Select the **Individual New Customer Type** and click **OK**.
5. Fill in the following customer information:

|  |  |
| --- | --- |
| **First Name** | John |
| **Last Name** | Smith |
| **Address Type** | Mailing |
| **Country** | Canada |
| **Zip Code** | G1R3X2 |
| **City** | Quebec City |
| **State/Province** | Quebec |
| **Address Line 1** | 12 Rue Sainte-Anne |

1. Click **Next**. You will be navigated to the Relationship tab.

|  |
| --- |
| **Note:** If you are navigated to the Customer Party Search Result Screen, select an existing party and click **Select**. Do not select an address. |

1. Do not enter any relationship data and click **Done**. You will be navigated to a summary of your new customer’s information.

|  |
| --- |
| Module 1: Introduction to Product DevelopmentExercise 1:3 – Rating a Quote |

|  |  |
| --- | --- |
| Scenario | In this exercise, you will create and rate a quote. |

### Process

1. Locate the customer named John Smith that you created in Exercise 2.1, “Creating and Finding Customers”. Open the customer details.
2. Click on the **Add Quote** button in the **Quote** section. Alternatively, you can simply click on the **Quote** tab.
3. In the **Product** dropdown, select the **PREC-HO** product and click **Add New Quote**.
4. Go through all of the tabs and fill in the policy details, clicking **Next** to navigate to the next tab.
5. When you’re on the **Premium** tab, click the **Rate** button to obtain the policy premium.
6. Click **Next** to save the quote and go back to the Consolidated View. In the header of the Consolidated View you should see that the quote has the status “Premium Calculated”. If for some reason it is still in the “Data Gathering” status, repeat steps 4 and 5.

### Expected Result

You should be taken to the policy’s Consolidated View. The Status field should now show “Premium Calculated”.

|  |
| --- |
| Module 2: Product Factory User InterfaceExercise 2:1 – Deactivating a Product |

|  |  |
| --- | --- |
| Scenario | This exercise will show you how to navigate to Product Factory from the main application, how to open a product and how to activate/deactivate a product. A product must be deactivated in order to configure it in Product Factory. However, it must be re-activated before it can be used for quote creation in PolicyCore. |

### Process

1. In the main application, navigate to the Solution Administration tool by clicking the **Admin** link in the top right corner of the screen.
2. Select the **Product** tab.
3. Click **Search** without entering anything in the search fields.
4. Choose **PREC-HO** from the list of products.
5. Deactivate the product by clicking on the lock icon on the bottom right of the screen.
6. Navigate back to the main application by clicking on the **Main** link in the top right corner of the screen.
7. Attempt to create a new quote for customer John Smith. Note that no products are available for selection.
8. Navigate back to the Solution Administration tool and re-activate the product by clicking on the lock icon again.

|  |
| --- |
| Module 3: Creating New ComponentsExercise 3:1 – Creating a Persistent Component |

|  |  |
| --- | --- |
| Scenario | In this exercise, you will create a component that allows an insurance broker to capture information about the pets residing in the insured Dwelling. |

### Process

1. Open Eclipse and create a domain entity for the component. To do so:
   1. Navigate to the training-product-domain module
   2. Navigate to the src/main/java directory
   3. Navigate to the com.exigen.ipb.training.domain package
   4. Create a new file named PetEntity.java, make sure to specify package to com.exigen.ipb.training.domain.add the code below and save the file.

|  |
| --- |
| package com.exigen.ipb.training.domain;  import com.exigen.ipb.base.datatypes.BaseEntity;  import javax.persistence.\*;  import com.exigen.ipb.components.domain.\*;  @Entity  public class PetEntity extends BaseEntity implements ComponentInfo{    @Embedded  private ComponentInstanceMetadata instanceMetadata;    private String petType;  private String breed;    public String getPetType() {  return petType;  }    public void setPetType(String petType) {  this.petType = petType;  }  public String getBreed() {  return breed;  }    public void setBreed(String breed) {  this.breed = breed;  }  public ComponentInstanceMetadata getInstanceMetadata() {  if (instanceMetadata == null) {  instanceMetadata = new ComponentInstanceMetadata();  }  return instanceMetadata;  }  public void setInstanceMetadata(ComponentInstanceMetadata instanceMetadata) {  this.instanceMetadata = instanceMetadata;  }    } |

1. Register the new entity in persistence.xml. To do so:
   1. Navigate to the training-product-domain module
   2. Navigate to the src/main/resources/META-INF directory
   3. Open persistence.xml
   4. Add the <class> tag line from the markup below and save the file.

|  |
| --- |
| <persistence-unit name="default" transaction-type="RESOURCE\_LOCAL">  <class>com.exigen.ipb.training.domain.PetEntity</class>  </persistence-unit> |

1. Update the Liquibase changelog with a changeset for the new entity. To do so:
   1. Navigate to the training-product-deploy module
   2. Navigate to the src/main/resources/db directory
   3. Open training-product-changelog.xml
   4. Add the changeset below to the changelog and save the file.

|  |
| --- |
| <changeSet id="7" author="eis-training">  <comment>Create PetEntity Training product</comment>  <!-- base entity -->  <createTable tableName="PetEntity">  <column name="id" type="NUMERIC(19,0)">  <constraints primaryKey="true" nullable="false"/>  </column>  <!-- component info metadata -->  <column name="componentInstanceName"  type="java.sql.Types.NVARCHAR(255)" />  <column name="connectedToInstanceName"  type="java.sql.Types.VARCHAR(255)" />  <column name="instanceName"  type="java.sql.Types.VARCHAR(255)" />  <column name="producerComponentName"  type="java.sql.Types.VARCHAR(255)" />  <column name="producerComponentVersion" type="FLOAT" />  <!-- domain columns -->  <column name="PetType" type="java.sql.Types.VARCHAR(10)" />  <column name="Breed" type="java.sql.Types.VARCHAR(15)" />  </createTable>  </changeSet> |

1. Create a Spring bean representing the component. You don’t need to create your own Component Service, as there is no custom logic—the SmartComponent can be used directly. To do so:
   1. Navigate to the training-product-components module.
   2. Navigate to the src/main/resources/META-INF/spring directory.
   3. Open the file thome-lifecycle-beans.xml.
   4. Within the <beans> element, add the markup below and save the file.

|  |
| --- |
| <bean id="PetBean" parent="pf.smartComponentTemplate"  class="com.exigen.ipb.components.SmartComponent">  <constructor-arg index="0">  <value>com.exigen.ipb.training.domain.PetEntity  </value>  </constructor-arg>  <property name="name" value="Pet"/>  <property name="version" value="1.0"/>  <property name="componentType" value="Other"/>  <property name="description"  value="Pets on the Property"/>  <property name="defaultTabLabel" value="Pets"/>  <property name="providedDataInterfaces">  <set>  <value>  com.exigen.ipb.training.domain.PetEntity  </value>  </set>  </property>  <property name="attributesMetadata">  <map>  <entry key="petType">  <bean  parent="pf.attributeMetaTemplate">  <property name="name"  value="petType"/>  <property name="label"  value="Pet Type"/>  <property name="dataType"  value="String"/>  <property name="orderNr"  value="1"/>  <property name="pathToEntityField"  value="petType"/>  <property name="displayable"  value="true"/>  <property name="ratingFactorInd"  value="true"/>  </bean>  </entry>  <entry key="breed">  <bean parent="pf.attributeMetaTemplate">  <property name="name" value="breed"/>  <property name="label" value="Breed"/>  <property name="dataType" value="String"/>  <property name="orderNr" value="2"/>  <property name="pathToEntityField"  value="breed"/>  <property name="displayable" value="true"/>  <property name="ratingFactorInd" value="true"/>  </bean>  </entry>  </map>  </property>  </bean> |

1. If you still have the EIS Suite running, close the window running Tomcat.
2. While still in the Terminal Emulator, navigate to the project folder using the following commands:

* D:
* cd \ new-training-pwc

1. Rebuild the application with the following command:

mvn clean install

1. After the application is finished building, navigate to the deployment folder with the following command:

cd / new-training-pwc/applications/training-deploy

1. Prepare the system to run Apache Ant with the following command:

mvn clean install -Ddeploy-db=dev -Ddbms=oracle

1. Navigate to the target directory with the following command:

cd / new-training-pwc/applications/training-deploy/target/training-deploy-dist

1. Run your Liquibase script with the following command:

ant db.prepare -Denvironment=..\..\training.env

1. In the Terminal Emulator, restart the web server by entering the following commands:

* c:
* cd \apache-tomcat-7.0.67-9.3\bin
* startup

1. Open the application by opening a web browser and navigating to the URL <http://localhost:8080/training>.
2. In the EIS Suite, navigate to Product Factory within the Solution Administration tool.
3. Open the PREC-HO product and deactivate it.
4. Navigate to the **Components** configuration section and switch to **Port** view.
5. Expand **policyDetail.riskItems**, then expand the PreconfigDwell component.
6. Within the **PreconfigDwell** component, select **universalPort**.
7. In the configuration panel, search for “pet.” The Pet component should be returned from the search.
8. Click **Connect** to connect the Pet component to the PreconfigDwell component via the Universal port, then click **Save**.

|  |
| --- |
| **Note:** Ignore the warning icon indicating that a persistent component has been connected through the Universal port. You will connect the component to a dedicated port in the next exercise. |

1. Select the **Pet** component and change the value of the **Instances Allowed** field to **99**.
2. In the **Instance table** section, select the **Show custom columns** checkbox, then click the **Customize** link.
3. Click the **+** button to add a new column.
4. In the **Label** field, enter “Breed”, and in the **Value expression** field, enter “@{breed}”.
5. Click **ADD** to add the column.
6. Repeat steps 23–25, entering “Type” and “@{petType}” in the **Label** and **Value expression** fields, respectively.
7. Click **DONE**, then click the **Save** button to save the component configuration.
8. Select the **Workspaces** configuration tab and expand the **Dwell** workspace.
9. Click on the **+** button to create a new tab on the workspace.
10. Give the new tab the name “Pets” and click **DONE**.
11. In the Pets tab configuration pane, search for the Pet component using the search bar in the **Available** section.
12. The Pet component should be returned from the search. Locate it and click **Assign**.
13. Click **Save** to save your changes.
14. Navigate back to the product’s Home screen and re-activate the product.
15. Navigate back to PolicyCore and create a new quote to see your changes reflected in the product.

|  |
| --- |
| Module 3: Creating New ComponentsExercise 3:2 – Extending an Existing Component |

|  |  |
| --- | --- |
| Scenario | Now that you have completed Exercise 3.1, “Creating a Persistent Component,” you should have a functioning Pet component. We have connected the Pet component to its parent component, PreconfigDwell, using the universalPort. However, using the universalPort is less efficient than using dedicated ports. For that reason, your task in this exercise is to extend the base PreconfigDwell component to add a proper port for the Pet component. |

### Process

1. Open Eclipse and create a domain entity for the component. To do so:
   1. Navigate to the training-product-domain module
   2. Navigate to the src/main/java directory
   3. Navigate to the com.exigen.ipb.training.domain package
   4. Create a new file there named DwellPet.java, add the code below and save the file.

|  |
| --- |
| package com.exigen.ipb.training.domain;  import com.exigen.ipb.base.datatypes.BaseEntity;  import javax.persistence.\*;  import com.exigen.ipb.components.domain.\*;  import com.exigen.ipb.policy.domain.\*;  import com.exigen.ipb.training.domain.PetEntity;  import java.util.\*;  @Entity  public class DwellPet extends DwellEntity {  @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER)  @JoinColumn(name = "riskitem\_id")  private List<PetEntity> pets;  public List<PetEntity> getPets() {  return pets;  }  public DwellPet() {  pets = new ArrayList<PetEntity>();  }  public void setPets(List<PetEntity> pets) {  this.pets = pets;  }  } |

1. Register the new entity in persistence.xml. To do so:
   1. Navigate to the training-product-domain module
   2. Navigate to the src/main/resources/META-INF directory
   3. Open persistence.xml
   4. Add the markup highlighted below and save the file

|  |
| --- |
| <persistence-unit name="default" transaction-type="RESOURCE\_LOCAL">  <class>com.exigen.ipb.training.domain.PetEntity</class>  <class>com.exigen.ipb.training.domain.DwellPet</class>  </persistence-unit> |

1. Create a Liquibase changeset to add a foreign key to the PetEntity table that connects PetEntity to the RiskItem table. To do so:
   1. Navigate to the training-product-deploy module
   2. Navigate to the src/main/resources/db directory
   3. Locate training-product-changelog.xml and add the following changeset:

|  |
| --- |
| <changeSet id="8" author="eis-training">  <comment>Add Foreign Key to PetEntity table</comment>  <addColumn tableName="PetEntity">  <column name="RiskItem\_ID" type="NUMERIC(19,0)">  <constraints foreignKeyName="FK\_PetRiskItem"  references="RiskItem(ID)"/>  </column>  </addColumn>  </changeSet> |

1. Create a new Spring bean for a new component called TrainingDwellComponent and map the new collection to a port that can accept the Pet component. To do so:
   1. Navigate to the training-product-components module
   2. Navigate to the src/main/resources/META-INF/spring directory.
   3. Open the thome-lifecycle-beans.xml file and add the following markup:

|  |
| --- |
| <bean id="TrainingDwellComponent" parent="PreconfigDwellComponent\_1\_0"  class="com.exigen.ipb.policy.home.preconfig.components.PreconfigDwellComponent">  <constructor-arg index="0">  <value>  com.exigen.ipb.training.domain.DwellPet  </value>  </constructor-arg>  <property name="name" value="Training Dwell"/>  <property name="version" value="1.0"/>  <property name="componentType" value="RiskItem"/>  <property name="description" value="Pets on the  Property"/>  <property name="defaultTabLabel" value="Pets"/>  <property name="providedDataInterfaces">  <set merge="true">  <value>  com.exigen.ipb.training.domain.DwellPet  </value>  </set>  </property>  <property name="dependencies">  <map>  <entry  key="com.exigen.ipb.policy.domain.PolicySummary">  <null />  </entry>  </map>  </property>  <property name="providedConnectionPoints">  <map merge="true">  <entry key="pets">  <bean parent="pf.connectionInfoTemplate">  <property name="className"  value="com.exigen.ipb.training.domain.PetEntity"/>  <property name="connectionPointType"  value="Multiple"/>  </bean>  </entry>  </map>  </property>  </bean> |

1. If you still have the EIS Suite running, close the window running Tomcat.
2. While still in the Terminal Emulator, navigate to the project folder using the following commands:

* D:
* cd \ new-training-pwc

1. Rebuild the application with the following command:

mvn clean install

1. After the application is finished building, navigate to the deployment folder with the following command:

cd / new-training-pwc/applications/training-deploy

1. Prepare the system to run Apache Ant with the following command:

mvn clean install -Ddeploy-db=dev -Ddbms=oracle

1. Navigate to the target directory with the following command:

cd / new-training-pwc/applications/training-deploy/target/training-deploy-dist

1. Run your Liquibase script with the following command:

ant db.prepare -Denvironment=..\..\training.env

1. In the Terminal Emulator, restart the web server by entering the following commands:

* c:
* cd \apache-tomcat-7.0.67-9.3\bin
* startup

1. Open the application by opening a web browser and navigating to the URL <http://localhost:8080/training>.
2. In the EIS Suite, navigate to Product Factory within the Solution Administration tool.
3. Open the PREC-HO product and deactivate it.
4. Navigate to the **Components** configuration section and switch to **Relation** view.
5. Select the **PreconfigDwell** component. Select the **Replace** option from the dropdown on the upper right-hand side.
6. Select **Training Dwell** as the replacement from the dropdown.
7. When prompted, select **Merge**.
8. Make sure that the reference name of the Training Dwell component is set to “PreconfigDwell”.
9. Verify that the following components are still attached to PreconfigDwell:

* **PreconfigConstructionInfo 1.0** via the **constructionInfo** port
* **PreconfigPreCovA 1.0** via the **coverages** port
* **PreconfigPreCovB 1.0** via the **coverages** port
* **PreconfigPreCovC 1.0** via the **coverages** port
* **PreconfigPreCovD 1.0** via the **coverages** port
* **PreconfigPreCovE 1.0** via the **coverages** port
* **PreconfigPreCovF 1.0** via the **coverages** port
* **PreconfigLocation 1.0** via the **location** port

1. Switch to **Port** view. Disconnect the **Pet** component from the Universal Port of Preconfig Dwell. Select the **pets** port of the Preconfig**Dwell** component and connect the **Pet** component through this port. Select the **Pet** component and set the value of the **Instances allowed** field to **99**.
2. In the **Instance table** section, select the **Show custom columns** checkbox, then click the **Customize** link.
3. Click the **+** button to add a new column.
4. In the **Label** field, enter “Breed”, and in the **Value expression** field, enter “@{breed}”.
5. Click **ADD** to add the column.
6. Repeat steps 23–25, entering “Type” and “@{petType}” in the **Label** and **Value expression** fields, respectively.
7. Click **DONE**, then click the **Save** button to save the component configuration.
8. Manage the attributes of the Preconfig Dwell component and eliminate all of them except for seqno and territorycd. Make them disabled. Give territorycd a default value of 001, and remove its lookup connections.
9. Manage the attributes of the Policy component and make sure that the following attributes are visible:

* imported
* countryCd
* riskStateCd
* policyFormCd
* policyTitle
* effective
* contractTermTypeCd
* expiration

1. Navigate to the **Workspace** configuration section, add the Pets component to the Pet sub-tab of Dwell and click the **Save** button to save the workspace configuration.
2. Re-activate the product.
3. Navigate back to PolicyCore and create a new quote to see your changes reflected in the product.

|  |
| --- |
| Module 4: Enhancing the User InterfaceExercise 4:1 – Creating a Non-Persistent Component With a Custom View |

|  |  |
| --- | --- |
| Scenario | The customer has asked you to add a feature to the application that lets users see all of the coverages and their limits in a table on the last tab of the Data Gather workspace. |

### Process

1. Open Eclipse and create a non-persistent data model entity. To do so:
   1. Navigate to the **training-product-components** module
   2. Navigate to the **src/main/java** directory.
   3. Right-click on the **com.exigen.ipb.training.home.services** package and select **New** > **Class**.
   4. In the **Name** field, enter “CoverageLimitViewModel” and click **Finish**.
   5. Enter the code below and save the file.

|  |
| --- |
| package com.exigen.ipb.training.home.services;  import com.exigen.ipb.components.domain.UniversalUnpersistentEntity;  import com.exigen.ipb.policy.domain.PolicyEntity;  import com.exigen.ipb.policy.domain.Coverage;  import java.util.Map;  import java.util.List;  public class CoverageLimitViewModel extends UniversalUnpersistentEntity {  PolicyEntity policy;  List<Coverage> coverageList;  public PolicyEntity getPolicy() {  return policy;  }  public void setCoverageList(List<Coverage> coverageList)  {  this.coverageList = coverageList;  }  public List<Coverage> getcoverageList() {  return coverageList;  }  public void setPolicy(PolicyEntity policy) {  this.policy = policy;  }  public void init() {  coverageList=policy.getPolicyDetail().getRiskItems().get(0).getCoverages();  }  } |

1. Create a custom component service. The service should override the SmartComponent#getViewModel() method, adding custom logic to gather data from all the coverage components. To do so:
   1. Navigate to the **training-product-components** module
   2. Navigate to the **src/main/java** directory.
   3. Right-click on the **com.exigen.ipb.training.home.services** package and select **New** > **Class**.
   4. In the **Name** field, enter “CoverageLimitComponent” and click **Finish**.
   5. Enter the code below and save the file.

|  |
| --- |
| package com.exigen.ipb.training.home.services;  import com.exigen.ipb.components.SmartComponent;  import com.exigen.ipb.components.domain.ComponentInfo;  import com.exigen.ipb.components.domain.ComponentLoadInfo;  import com.exigen.ipb.components.domain.ViewType;  import com.exigen.ipb.policy.domain.PolicyEntity;  import java.util.List;  /\*\*  \* <p>  \* Represents a view-only component that renders coverage's limits.  \* <p>  \* </p>  \* <b>Note</b>: to work properly it must be connected to a <code>Policy</code>  \* component. </p>  \*  \* @author eis-training  \*  \*/  public class CoverageLimitComponent extends SmartComponent {  private static final long serialVersionUID = -1;  public CoverageLimitComponent() {  super();  }  public CoverageLimitComponent(Class<?> classToInstantiate) {  super(classToInstantiate);  }  @Override  public Object getViewModel(ComponentInfo runtimeInstance, List<ComponentInfo> dependencies, ViewType viewType) {  CoverageLimitViewModel viewModel = (CoverageLimitViewModel) runtimeInstance;  for (ComponentInfo dependency : dependencies) {  if (dependency instanceof PolicyEntity) {  viewModel.setPolicy((PolicyEntity) dependency);  break;  }  }  viewModel.init();  return viewModel;  }  } |

1. Create a Spring bean for a new component called CoverageLimitComponent\_1\_0. Specify com.exigen.ipb.policy.domain.Coverage as a data interface. To do so:
   1. Navigate to the **training-product-components** module.
   2. Navigate to the **src/main/resources/META-INF/spring** directory.
   3. Open the file **thome-lifecycle-beans.xml**.
   4. Within the <beans> element, add the markup below and save the file.

|  |
| --- |
| <bean id="CoverageLimitComponent\_1\_0" parent="pf.smartComponentTemplate"  class="com.exigen.ipb.training.home.services.CoverageLimitComponent">  <constructor-arg index="0">  <value>com.exigen.ipb.training.home.services.CoverageLimitViewModel  </value>  </constructor-arg>  <property name="name" value="CoverageLimit" />  <property name="version" value="1.0" />  <property name="componentType" value="Coverage" />  <property name="description" value="Policy Coverage information" />  <property name="defaultTabLabel" value="Coverage Info" />  <property name="dependencies">  <map merge="true">  <entry key="com.exigen.ipb.policy.domain.PolicySummary">  <null />  </entry>  </map>  </property>  <property name="providedDataInterfaces">  <set merge="true"> <value>com.exigen.ipb.components.domain.UniversalUnpersistentEntity  </value>  <value>com.exigen.ipb.policy.domain.Coverage  </value>  </set>  </property>  <property name="views">  <map merge="true"  key-type="com.exigen.ipb.components.domain.ViewType"  value-type="java.lang.String">  <entry key="dataGather"  value="/components/PolicyCoverageLimit.xhtml" />  </map>  </property>  </bean> |

1. Create a custom view to display the list of coverages. To do so:
   1. Navigate to the **training-product-components** module.
   2. Navigate to the **src/main/resources/components** directory.
   3. Open the file “PolicyCoverageLimit.xhtml,” or create it if it doesn’t exist.
   4. Replace the existing content with the markup below, or simply add the markup if you had to create the file. When finished, save the file.

|  |
| --- |
| **Note:** You may need to create a new file association for XHTML files so that they open in Eclipse, rather than in a web browser. To do so:   1. Go to **Window > Preferences** 2. In the Preferences pane, go to **General > Editors > File Associations** 3. In the **File types** section, click **Add** 4. In the **File type** field, enter **\*.xhtml** 5. In the **Associated Editors** section, click **Add** 6. Select **Text Editor** and click **OK** 7. Click **OK** |

|  |
| --- |
| <ui:composition  template="/components/datagather/componentViewPanelTemplate.xhtml"  xmlns:ui="http://java.sun.com/jsf/facelets"  xmlns:h="http://java.sun.com/jsf/html"  xmlns:f="http://java.sun.com/jsf/core"  xmlns:ef="http://exigengroup.com/ipb/jsf/facelets"  xmlns:eft="http://exigengroup.com/ipb/tags/facelets"  xmlns:epf="http://exigengroup.com/ipb/components/jsf/facelets"  xmlns:rich="http://richfaces.org/rich">  <ui:define name="componentContextViewForm">    <rich:dataTable id="CoverageLimitTable"  value="#{model.coverageList}"  headerClass="header"  style="border-left-width:0px"  rowClasses="evenRow lightGreyHover"  footerClass="noBorderRight noBackgroundColor lightGreyHover"  var="row">  <rich:column>  <f:facet name="header">Coverage Code</f:facet>  <h:outputText value="#{row.coverageCd}" />  </rich:column>  <rich:column>  <f:facet name="header">Coverage Limit</f:facet>  <h:outputText value="#{row.limitAmount}" />  </rich:column>    </rich:dataTable>  </ui:define>  </ui:composition> |

1. If you still have the EIS Suite running, close the window running Tomcat.
2. While still in the Terminal Emulator, navigate to the project folder using the following commands:

* D:
* cd \ new-training-pwc

1. Rebuild the application with the following command:

mvn clean install

1. In the Terminal Emulator, restart the web server by entering the following commands:

* c:
* cd \apache-tomcat-7.0.67-9.3\bin
* startup

1. Open the application by opening a web browser and navigating to the URL <http://localhost:8080/training>.
2. In the EIS Suite, navigate to Product Factory within the Solution Administration tool.
3. Open the PREC-HO product and deactivate it.
4. Navigate to the **Components** configuration section and switch to **Port** view.
5. Connect CoverageLimit to the UniversalPort of the root component. Save the configuration.
6. Navigate to the **Workspaces** configuration section and add a new tab called “Limits” to the default workspace.
7. Assign the CoverageLimit component to the Limits tab and save the configuration.
8. Re-activate the product.
9. Navigate back to PolicyCore and create a new quote to see your changes reflected in the product.

|  |
| --- |
| Module 5: Using Product Factory RulesExercise 5:1 – Adding an Assertion Rule |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scenario | In this scenario, the customer has asked you to set coverage limits for the various coverages included in the product.   |  |  |  | | --- | --- | --- | | **Coverage** | **Min** | **Max** | | Coverage A | 200,000 | 5,000,000 | | Coverage B | 0 | 500,000, but not more than Coverage A | | Coverage C | 0 | 3,000,000 | | Coverage D | 25,000 | 500,000 | | Coverage E | 100,000 | 500,000, but not more than Coverage A | | Coverage F | 0 | 25,000 | |

### Process

1. Open the **PREC-HO** product in the Solution Admin.
2. Deactivate the product.
3. Go to the **Rules** tab. The Search tab will open by default.
4. In the **Applied to component** field, enter the reference name of the coverage component, e.g. “PreconfigPreCovA”. In the **Applied to attribute** field, enter “limitAmount”. Click **Search** at the bottom of the screen.
5. Select all of the existing rules and delete them.
6. Select the appropriate coverage, expand it and select the **limitAmount** attribute.
7. Click the **+** button in the top right corner to create a new rule.
8. In the **Applied to** field, click the down arrow.
9. Search for the attribute “limitAmount”.
10. Select the appropriate coverage component, select **limitAmount** and click **Select**.
11. Enter the appropriate assertion expression in the **Assertion Expression** field.
12. Enter an error message in the **Error** field. This message will display if the user enters a value outside of the allowed range.
13. Select the checkboxes corresponding to the **Quote Rate** event and **Data Gather** action.
14. Save the rule.
15. Save again on the **Rules** screen.
16. Repeat steps 4-15 for each rule that you need to add.
17. Activate the product.
18. Test your new rules.

### Solutions

**PreConfigPreCovA**

PreconfigPreCovA.limitAmount>=200000 && PreconfigPreCovA.limitAmount<=5000000

**PreConfigPreCovB**

PreconfigPreCovB.limitAmount>=0 && PreconfigPreCovB.limitAmount<=500000 && PreconfigPreCovB.limitAmount<=PreconfigPreCovA.limitAmount

**PreConfigPreCovC**

PreconfigPreCovC.limitAmount>=0 && PreconfigPreCovC.limitAmount<=3000000

**PreConfigPreCovD**

PreconfigPreCovD.limitAmount>=25000 && PreconfigPreCovD.limitAmount<=500000

**PreConfigPreCovE**

PreconfigPreCovE.limitAmount>=100000 && PreconfigPreCovE.limitAmount<=500000 && PreconfigPreCovE.limitAmount<=PreconfigPreCovA.limitAmount

**PreConfigPreCovF**

PreconfigPreCovF.limitAmount>=0 && PreconfigPreCovF.limitAmount<=25000

|  |
| --- |
| Module 5: Using Product Factory RulesExercise 5:2 – Adding a Default Value Rule |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Scenario | Now that you’ve set coverage limits for the customer, they are requesting that the default coverage limits are pre-filled in the corresponding field when starting the quotation process.  The default values are:   |  |  | | --- | --- | | **Coverage** | **Default** | | Coverage A | 300,000 | | Coverage B | 0 | | Coverage C | 50% of Coverage A | | Coverage D | 15% of Coverage A | |

### Process

The procedure here is the same as in Exercise 5.1, “Adding an Assertion Rule,” but you’re entering a default value expression rather than an assertion expression. No error message is needed for rules that have no assertions.

1. Open the **PREC-HO** product in the Solution Admin.
2. Deactivate the product.
3. Go to the **Rules** tab.
4. Expand the node for the appropriate coverage component and select the **limitAmount** attribute.
5. Click the **+** icon to create a new rule.
6. Enter the appropriate default value expression in the **Default Value Expression** field.

|  |
| --- |
| **Note:** If instructed to do so, turn the **Override** widget on—see your instructor for more information. |

1. Select the checkboxes corresponding to the **Data Gather** action and the **Quote Rate** event.
2. Save the rule.
3. Save again on the **Rules** screen.
4. Activate the product.
5. Test your new rules.

### Solutions

**PreconfigPreCovA:** 300,000

**PreconfigPreCovB:** 0

**PreconfigPreCovC:** PreconfigPreCovA.limitAmount\*.5

**PreconfigPreCovD:** PreconfigPreCovA.limitAmount\*.15

|  |
| --- |
| Module 5: Using Product Factory RulesExercise 5:3 – Adding a Field Visibility Rule |

|  |  |
| --- | --- |
| Scenario | The customer wants the Year Updated field in the Preconfig Dwell Data Gather workspace to only be shown if the Year Built is not the current year. |

### Process

The procedure here is the same as in Exercises 6.1 and 6.2, except that here you’re entering a condition expression. Additionally, you should select the **Hidden** indicator for these rules. No error message is needed for rules that have no assertions.

**Hint:** This exercise utilizes a built-in business rule function. Refer to the “Built-In Business Rule Functions” page in this course’s training material for a list of the built-in functions available to business rules.

1. Open the **PREC-HO** product in the Solution Admin.
2. Deactivate the product.
3. Go to the **Rules** tab.
4. Select the appropriate PreconfigPreCovF, expand it and select the **limitAmount** attribute
5. Click the **+** icon to create a new rule.
6. Enter the appropriate condition expression.
7. Check the **Hidden** checkbox. This will cause the attribute to be hidden if the condition expression resolves to True.
8. Select the **Data Gather** action.
9. Select the **Quote Rate** event.
10. Save the rule.
11. Save again on the Rules screen.
12. Activate the product.
13. Test your new rule.

### Solution

PreconfigConstructionInfo.yearBuilt==Year(Today())

|  |
| --- |
| Module 5: Using Product Factory RulesExercise 5:4 – Adding a Component Applicability Rule |

|  |  |
| --- | --- |
| Scenario | In this scenario, the customer is requesting for Medical Payments Coverage to be only be available on policies where the Personal Liability Coverage Limit is higher than $100,000.  **Hint:** In the PREC-HO product, the PreconfigPreCovE component defines Personal Liability Coverage and the PreconfigPreCovF component defines Medical Payments Coverage. |

### Process

1. Open the **PREC-HO** product.
2. Deactivate the product.
3. Go to the **Components** tab.
4. Expand the **Preconfig** **Dwell** component and select the **PreconfigPreCovF** component.
5. In the **Applicability** section, click on the **SET CONDITIONS** button. Observe the existing applicability rule that is attached to this component. Add an additional constraint to this rule based on the requirements of this exercise.
6. Enter your applicability expression and click SAVE.

|  |
| --- |
| **Note:** The component will display if the applicability expression evaluates to True. |

1. Activate the product.
2. Test your new rule.

### Solution

(Policy.policyFormCd == 'HO2' || Policy.policyFormCd == 'HO3' || Policy.policyFormCd == 'HO5' || Policy.policyFormCd == 'HO4' || Policy.policyFormCd == 'HO6') && PreconfigPreCovE.limitAmount > 100000

|  |
| --- |
| Module 7: Developing Product-Specific ServicesExercise 7:1 – Customizing the Issue Action |

|  |  |
| --- | --- |
| Scenario | In this scenario, the customer is an insurance company that specializes in insuring historic properties. As a result, they have a special requirement that no house that is built before 1890 can be declined for coverage. |

### Process

1. Create a new class called QuoteCancelByCompanyActionYear that extends QuoteDeclineActionImpl. To do so:
   1. Navigate to the training-product-components module.
   2. Navigate to the src/main/java directory.
   3. Within the com.exigen.ipb.training.policy.services package, create a class called QuoteCancelByCompanyActionYear.
   4. Add the code below and save the file.

|  |
| --- |
| package com.exigen.ipb.training.policy.services;  import com.exigen.ipb.policy.core.services.lifecycle.impl.\*;  import com.exigen.ipb.base.datatypes.ProcessingException;  import com.exigen.ipb.policy.dto.PolicyTxInfo;  import com.exigen.ipb.policy.domain.\*;  import com.exigen.ipb.training.domain.\*;  /\*\*  \* <p>  \* Represents an override of QuoteDeclineActionImpl action  \* <p>  \*  \* @author eis-training  \*  \*/  public class QuoteCancelByCompanyActionYear extends QuoteDeclineActionImpl<PolicySummary>  {  @Override  public PolicySummary companyDecline(PolicySummary policy, PolicyTxInfo details) throws ProcessingException {  DwellPet myDwellPet = (DwellPet)(policy.getPolicyDetail().getRiskItems().get(0));  DwellDetail myDwellDetail = myDwellPet.getDwellDetail();  ConstructionInfo myConstruct = myDwellPet.getConstructionInfo();  int yearBuilt = myConstruct.getYearBuilt();  if(yearBuilt < 1890)  {  throw new ProcessingException("Policy can't be declined for a dwelling built earlier than 1890", null);  }  else  return (super.companyDecline(policy, details));  }  } |

1. Create a Spring bean for the action. To do so:
   1. Navigate to the training-product-components module.
   2. Navigate to the src/main/resources/META-INF/spring directory.
   3. Open the file thome-lifecycle-beans.xml.
   4. Verify that the markup highlighted below has already been added to the <beans> element of the file. If not, add it.

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <beans xmlns="http://www.springframework.org/schema/beans"  xmlns:eis="http://www.exigenservices.com/schema/exigen-beans"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://www.exigenservices.com/schema/exigen-beans  http://www.exigenservices.com/schema/exigen-beans.xsd  http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd"> |

* 1. Within the <beans /> element, add the bean definition and service substitution definition below, then save the file.

|  |
| --- |
| <bean id="quoteCancelByCompanyActionYear" parent="core\_quoteDeclineAction" class="com.exigen.ipb.training.policy.services.QuoteCancelByCompanyActionYear">  </bean>  <eis:serviceSubstitute contextKey="PREC-HO" resolverCd="ProductCodeResolver">  <entry key="core\_quoteDeclineAction" value="quoteCancelByCompanyActionYear" />  </eis:serviceSubstitute> |

1. If you still have the EIS Suite running, close the window running Tomcat.
2. While still in the Terminal Emulator, navigate to the project folder using the following commands:

* D:
* cd \ new-training-pwc

1. Rebuild the application with the following command:

mvn clean install

1. In the Terminal Emulator, restart the web server by entering the following commands:

* c:
* cd \apache-tomcat-7.0.67-9.3\bin
* startup

1. Open the application by opening a web browser and navigating to the URL <http://localhost:8080/training>.
2. In the EIS Suite, navigate to PolicyCore and begin quoting a policy. On the **Dwell** tab, in the **Year Built** field, enter a year before 1890.
3. Rate and issue the policy. You should receive an error message stating that the policy can’t be issued for a dwelling built before 1890.

|  |
| --- |
| Module 8: Adding Endorsement FormsExercise 8:1 – Using the DataObjectManager Component to Manage Endorsement Forms |

|  |  |
| --- | --- |
| Scenario | The customer wants to add the following endorsement forms to the product:   * Inflation Guard * Watercraft Endorsement |

### Process

#### Adding the DataObjectManager Component to Manage Forms

1. Open the **PREC-HO** product in Product Factory and deactivate it.
2. Connect the **DataObjectManager** component to the universalPort of the Dwell component.
3. Select the DataObjectManager component and click the **SET PROPERTIES** button.
4. Ensure that the **com.exigen.ipb.policy.domain.Form** is specified in the **manageableInterfaces** property. This instructs DataObjectManager to start managing all components that implement this interface.
5. Click the **Save** button to save the component configuration.
6. Go to the **Workspaces** tab and create a new tab called “Forms”.
7. Assign the DataObjectManager component to the Forms tab.
8. Click the **Save** button to save the workspace configuration.
9. Activate the product and check to see that the Forms tab has been added, along with the DataObjectManager component. In the next exercise, you’ll add some actual forms to the component.

#### Adding Form Components

1. Navigate back to Product Factory, open the PREC-HO product and deactivate it.
2. Select the **Components** tab.
3. Select **Port** view.
4. Expand the **policyDetail.riskItems** component, then expand the **Dwell** component.
5. Select the **forms** port.
6. Some forms have already been added. Observe that the WatercraftEndorsement and InflationGuard forms have already been added.
7. Click the **Save** button to save the component configuration.
8. Activate and test your product. You should see the two forms you connected—and others—in the **Available** section.

|  |
| --- |
| Module 9: Using PolicyCore Platform ServicesExercise 9:1 – Creating a New Lookup and Drop-Down Box |

|  |  |
| --- | --- |
| Scenario | In this scenario, the customer wants to add another restriction to the product: They want to restrict the types of pets that can be listed under the policy. The pet attribute type should be displayed as a drop-down with the following options:   * Cat * Dog * Raccoon   The breeds available depending on the pet type should be:   * Cat breeds   + Siamese   + Maine Coon   + Russian Blue   + Other * Dog Breeds   + German Shepherd   + Great Dane   + Labrador Retriever   + Other * Raccoon should have no breed options—the breed field should be hidden if Raccoon is selected |

### Process

1. Define a class for the lookup’s domain by extending CodeValueLookup. To do so:
   1. Navigate to the training-product-domain module.
   2. Navigate to the src/main/java directory.
   3. Navigate to the com.exigen.ipb.training.domain package.
   4. Create a new file called PetBreedLookup.java and add the following code:

|  |
| --- |
| package com.exigen.ipb.training.domain;  import javax.persistence.Entity;  import com.exigen.ipb.base.lookups.CodeValueLookup;  /\*\*  \* Lookup definition for Pets and Breeds  \* @author  \*  \*/  @Entity  public class PetBreedLookup extends CodeValueLookup {  private static final long serialVersionUID = -1;    public static final String NAME="PetBreedLookup";    private String petType;  public String getPetType() {  return petType;  }  public void setPetType(String petType) {  this.petType = petType;  }  @Override  public String toString() {  return super.getValue();  }  } |

1. Create a Lookup Provider for the PetBreed lookup by creating a new file called PetBreedLookupTemplate.java in the same location as the previous step. Add the following code to the file:

|  |
| --- |
| package com.exigen.ipb.training.domain;  import java.util.Map;  import javax.persistence.EntityManager;  import com.exigen.ipb.base.lookups.AbstractCodeValueTemplateProvider;  import org.springframework.cache.annotation.Cacheable;  import com.exigen.ipb.base.BaseCache;  import com.exigen.ipb.base.annotations.CanInvoke;  import com.exigen.ipb.base.cache.BaseCacheKeyStrategies;  import com.exigen.ipb.base.cache.CacheKeyStrategy;  @CanInvoke  public class PetBreedLookupTemplate extends AbstractCodeValueTemplateProvider {    public static final String PARAM\_PET = "petType";  public static final String QUERY\_ALL = "select lookupValue\n "  + "from LookupList as lookup\n " + "inner join lookup.lookupValues as lookupValue\n"  + "where lookup.lookupName=:lookupName\n";  public static final String QUERY\_VALUE = "select lookupValue.value\n " + "from LookupList as lookup\n "  + "inner join lookup.codeValueLookups as lookupValue\n" + "where lookup.lookupName=:lookupName\n "  + "and lookupValue.code=:code ";  @Override  @Cacheable(BaseCache.LOOKUP\_CACHE\_MODULE\_ID)  @CacheKeyStrategy(BaseCacheKeyStrategies.HASH\_CODE\_WITH\_I18N)  public Map<String, Object> getAvailableValues(String lookupName, Map<String, Object> params) {  return super.getAvailableValues(lookupName, params);  }  @Override  @Cacheable(BaseCache.LOOKUP\_CACHE\_MODULE\_ID)  @CacheKeyStrategy(BaseCacheKeyStrategies.HASH\_CODE\_WITH\_I18N)  public String getPresentationFor(String lookupName, Object currentValue, Map<String, Object> params) {  return super.getPresentationFor(lookupName, currentValue, params);  }  @Override  public void appendQuery(Map<String, Object> params, StringBuffer queryStr, Map<String, Object> qparam) {  String petType = (String) params.get(PARAM\_PET);  if (petType != null){  queryStr.append(" and (lookupValue.petType is null or :petType = lookupValue.petType) ");  qparam.put("petType", petType);  }  }  @Override  protected void appendOrderBy(StringBuffer queryStr) {  queryStr.append(" order by lookupValue.code");  }  @Override  protected String getQueryAllSql() {  return QUERY\_ALL;  }  @Override  protected String getQuerySingleSql() {  return QUERY\_VALUE;  }  } |

1. Add Spring bean configuration for the lookup. To do so:
   1. Navigate to the training-product-components module.
   2. Navigate to the src/main/resources/META-INF/spring directory.
   3. Open the file thome-lifecycle-beans.xml.
   4. Within the <beans> element, add the markup below and save the file.

|  |
| --- |
| <bean id="petBreedLookupTemplate"  class="com.exigen.ipb.training.domain.PetBreedLookupTemplate">  <property name="templateDescriptor">  <bean class="com.exigen.ipb.base.lookups.domain.LookupTemplateDescriptor">  <property name="templateName" value="PetBreed Lookup" />  <property name="templateImplClass" value="com.exigen.ipb.training.domain.PetBreedLookup" />  <property name="paramsDefinitions">  <map>  <entry key="petType">  <bean class="com.exigen.ipb.base.lookups.domain.LookupParamDefintion"  p:paramName="petType" p:paramLabel="Pet Type"  p:paramImplClass="java.lang.String" />  </entry>  </map>  </property>  <property name="resultDefinition">  <null />  </property>  </bean>  </property>  </bean> |

1. Register the lookup in persistence.xml. To do so:
   1. Navigate to the training-product-domain module
   2. Navigate to the src/main/resources/META-INF directory
   3. Open persistence.xml
   4. Add the markup below.

|  |
| --- |
| <class>com.exigen.ipb.training.domain.PetBreedLookup</class> |

1. Update the Liquibase changelog with 2 changesets that will accomplish the following, respectively:

* Add the PETTYPE column to the LOOKUPVALUE table
* Add a row for PetBreedLookup to the LOOKUPLIST table and load the values from csv/PetBreedLookup.csv into the LOOKUPVALUE table

To do so:

* 1. Navigate to the training-product-deploy module
  2. Navigate to the src/main/resources/db/csv directory
  3. Open training-product-changelog.xml
  4. Add the changesets below to the changelog and save the file.

|  |
| --- |
| <changeSet id="9" author="eis-training">  <comment>Add a column to lookupvalue table</comment>  <addColumn tableName="LOOKUPVALUE">  <column name="PETTYPE" type="java.sql.Types.VARCHAR(10)"/>  </addColumn>  </changeSet>    <changeSet id="10" author="eis-training">  <preConditions onFail="MARK\_RAN">  <sqlCheck expectedResult="0">select count(\*) from LookupList where lookupName = 'PetBreedLookup'</sqlCheck>  </preConditions>  <comment>change lookups</comment>  <delete tableName="LookupValue">  <where>LOOKUPLIST\_ID = (SELECT id FROM LookupList WHERE lookupName = 'PetBreedLookup')</where>  </delete>  <insert tableName="LookupList">  <column name="defClass" value="com.exigen.ipb.training.domain.PetBreedLookup"/>  <column name="lookupName" value="PetBreedLookup"/>  </insert>  <sql>ALTER TABLE LookupValue DISABLE CONSTRAINT FK\_LookupValueLookupListID;</sql>  <loadData tableName="LookupValue" file="db/csv/PetBreedLookup.csv" />  <sql>UPDATE LookupValue SET LOOKUPLIST\_ID = (SELECT id FROM LookupList WHERE lookupName = 'PetBreedLookup')  WHERE LOOKUPLIST\_ID IS NULL  </sql>  <sql>ALTER TABLE LookupValue ENABLE CONSTRAINT FK\_LookupValueLookupListID;</sql>  </changeSet> |

1. Create a CSV file to hold the breed lookup values. To do so:
   1. Navigate to the src/main/resources/db/csv directory.
   2. Create a new file called PetBreedLookup.csv.
   3. Add the values below.

|  |
| --- |
| DTYPE,code,displayValue,petType  PetBreedLookup,Siamese,Siamese,Cat  PetBreedLookup,Main Coon,Main Coon,Cat  PetBreedLookup,Russian Blue,Russian Blue,Cat  PetBreedLookup,Other,Other,Cat  PetBreedLookup,German Sheperd,German Sheperd,Dog  PetBreedLookup,Great Dane,Great Dane,Dog  PetBreedLookup,Labrador,Labrador,Dog  PetBreedLookup,Other,Other,Dog  PetBreedLookup,,,Raccoon |

1. Create another changeset to add a row for PetLookup to the LOOKUPLIST table and load the values from PetLookup.csv. To do so:
   1. Navigate to the training-product-deploy module
   2. Navigate to the src/main/resources/db/csv directory
   3. Open training-product-changelog.xml
   4. Add the changeset below to the changelog and save the file.

|  |
| --- |
| <changeSet id="11" author="eis-training">  <preConditions onFail="MARK\_RAN">  <sqlCheck expectedResult="0">select count(\*) from LookupList where lookupName = 'PetLookup'</sqlCheck>  </preConditions>  <comment>change lookups</comment>  <delete tableName="LookupValue">  <where>LOOKUPLIST\_ID = (SELECT id FROM LookupList WHERE lookupName = 'PetLookup')</where>  </delete>  <insert tableName="LookupList">  <column name="defClass" value="com.exigen.ipb.base.lookups.CodeValueLookup"/>  <column name="lookupName" value="PetLookup"/>  </insert>  <sql>ALTER TABLE LookupValue DISABLE CONSTRAINT FK\_LookupValueLookupListID;</sql>  <loadData tableName="LookupValue" file="db/csv/PetLookup.csv" />  <sql>  UPDATE LookupValue SET LOOKUPLIST\_ID = (SELECT id FROM LookupList WHERE lookupName = 'PetLookup')  WHERE LOOKUPLIST\_ID IS NULL  </sql>  <sql>ALTER TABLE LookupValue ENABLE CONSTRAINT FK\_LookupValueLookupListID;</sql>  </changeSet> |

1. Create a CSV file to hold the pet type lookup values. To do so:
2. Navigate to the src/main/resources/db/csv directory.
3. Create a new file called PetLookup.csv.
4. Add the values below.

|  |
| --- |
| DTYPE,code,displayValue  CodeValueLookup,Cat,Cat  CodeValueLookup,Dog,Dog  CodeValueLookup,Raccoon,Raccoon |

1. If you still have the EIS Suite running, close the window running Tomcat.
2. While still in the Terminal Emulator, navigate to the project folder using the following commands:

* D:
* cd \ new-training-pwc

1. Rebuild the application with the following command:

mvn clean install

1. After the application is finished building, navigate to the deployment folder with the following command:

cd / new-training-pwc/applications/training-deploy

1. Prepare the system to run Apache Ant with the following command:

mvn clean install -Ddeploy-db=dev -Ddbms=oracle

1. Navigate to the target directory with the following command:

cd / new-training-pwc/applications/training-deploy/target/training-deploy-dist

1. Run your Liquibase script with the following command:

ant db.prepare -Denvironment=..\..\training.env

1. In the Terminal Emulator, restart the web server by entering the following commands:

* c:
* cd \apache-tomcat-7.0.67-9.3\bin
* startup

1. Open the application by opening a web browser and navigating to the URL http://localhost:8080/training.
2. In the EIS Suite, navigate to Product Factory within the Solution Administration tool.
3. Open the PREC-HO product and deactivate it.
4. Navigate to the **Components** configuration.
5. Select the **Pet** component.
6. Configure a lookup for the attribute **petType** using PetLookup.
7. Configure a lookup for the attribute **breed** using PetBreedLookup. When configuring parameters, use **Pet.petType**.
8. Navigate back to PolicyCore, create a new quote and verify that the values available in the **Breed** field change depending on the value selected in the **Pet Type** field.

|  |
| --- |
| Module 9: Using PolicyCore Platform ServicesExercise 9:1 – Scheduling a Job Batch |

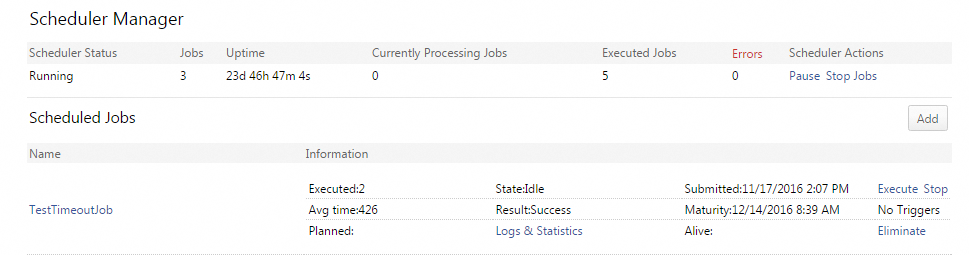
|  |  |
| --- | --- |
| Scenario | In this exercise, you will use the Scheduler in the Solution Administration application to schedule a simple job batch for execution.  Your job batch will contain a single job, activityTimeoutJob, which goes through the activities in the system, changing any that are still active to mark them as finished. |

### Process

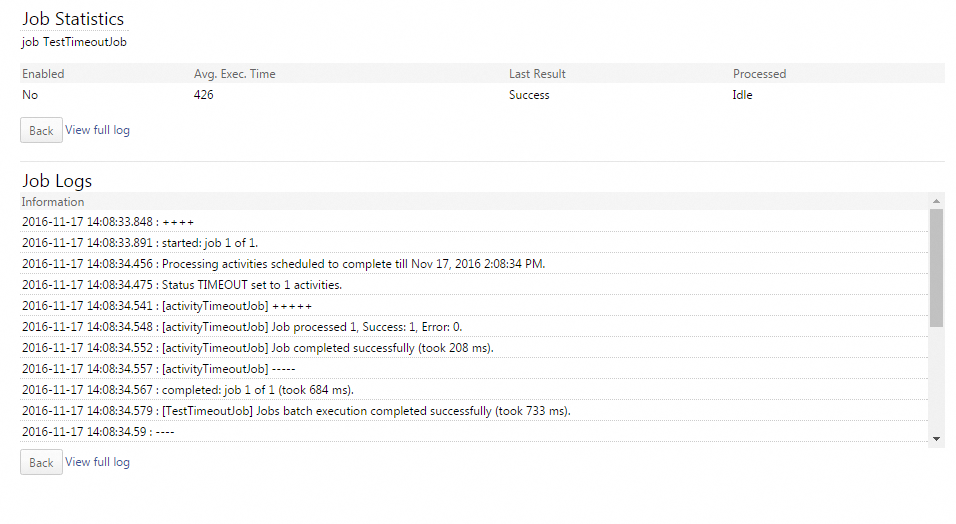
1. Log into the EIS Suite
2. Click on the **Admin** link in the top right corner of the screen to open the Solution Administration application.
3. The **General** tab should open. Click on the **Scheduler** sub-tab on the left side of the screen to open the Scheduler.
4. In the **Scheduled Jobs** section, click on the **Add** button to add a new job batch.
5. In the **Name** field, enter “TestTimeoutJob”.
6. Click on the **Add Job** button to add a new job to the batch.
7. In the dropdown field, select **activityTimeoutJob**.
8. In the **Scheduling** section, click the **Enabled** checkbox.
9. In the **Start** field, select the current date and enter a time 2 minutes later than the current time.
10. In the **End** field, enter a time 5 minutes after the Start time, to allow the job some time to execute. Leave all of the fields in the **Triggers** section with their default values.
11. Click on the **Save** button at the bottom of the screen to save the job.
12. Wait 7 minutes to allow the job time to start and finish executing.
13. Click on another tab in the Solution Admin then navigate back to the Scheduler, so that the content of the Scheduler tab refreshes.

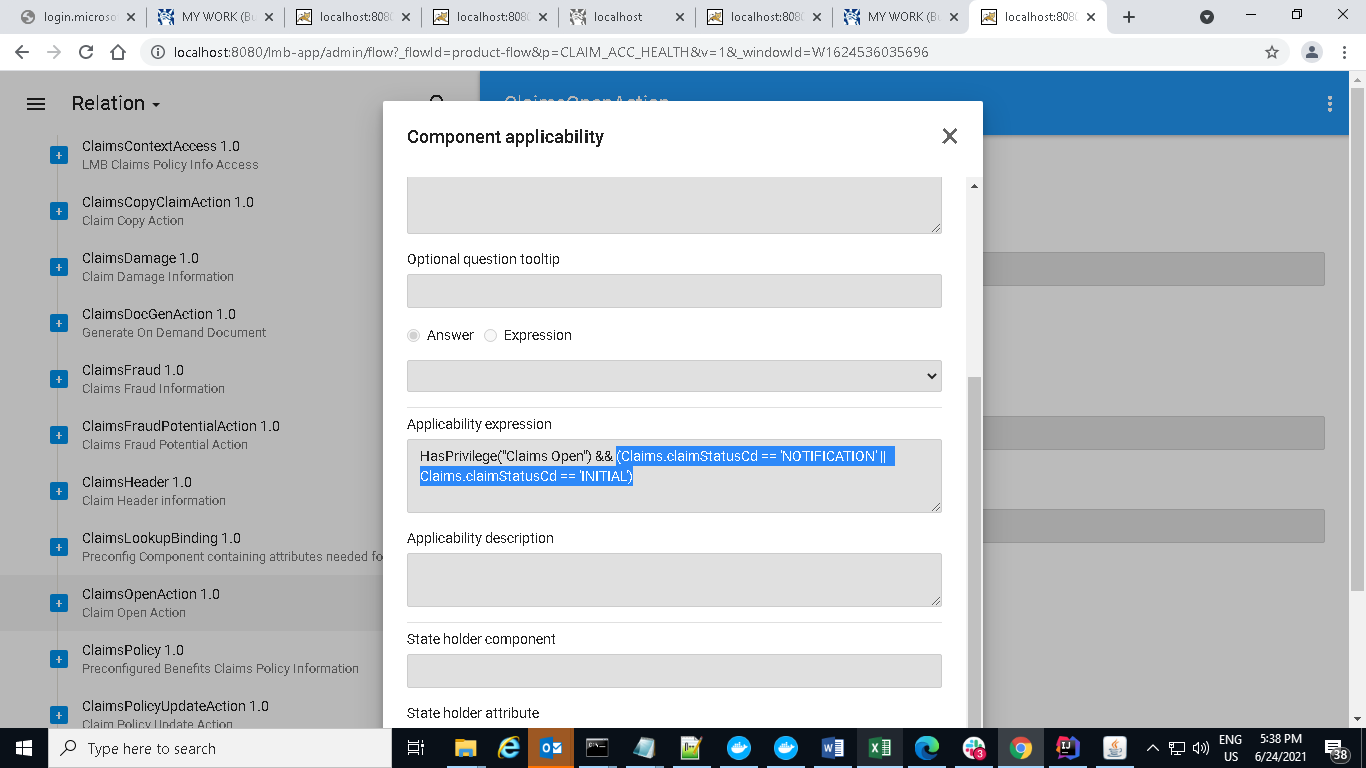
### Expected Result

Once your scheduled job has executed, in the **Scheduled Jobs** section of the Scheduler, you should see TestTimeoutJob listed with a **Result** of “Success.” Click on the **Logs & Statistics** link to review the log for the job.



To view the log output, click on the **Logs & Statistics** link. You should see log output resembling the following:





Claims.productCd==’VB\_RAC’

*VB\_RAC*

Claims.productCd == ' VB\_RAC '