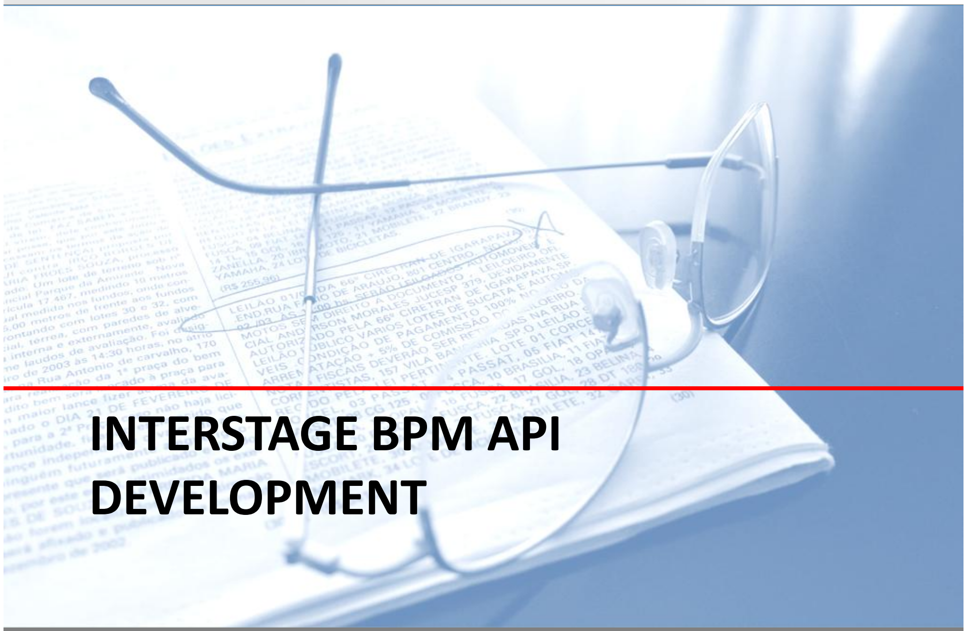


INTERSTAGE BPM API DEVELOPMENT

Objectives

- Overview of the Interstage BPM Model API
  - Managing Workflows
  - Process Modeling
  - Other Features

Introduction

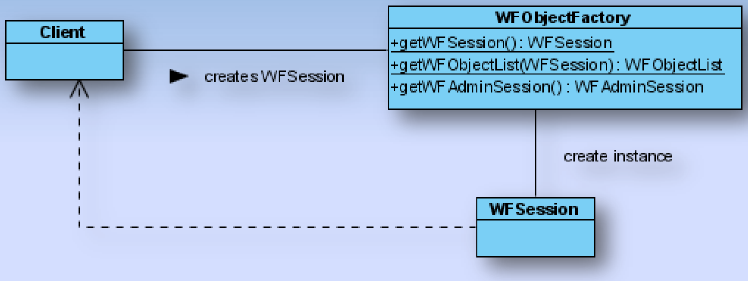
- The Interstage BPM Model API provides
  - Full access to Interstage BPM Workflow Applications
  - Administration
    - Create/Delete Users and Roles (local directory only)
    - Manage applications
    - Publish/archive
  - Developer
    - Create/edit process definitions
    - Create /edit and manage instances
    - Manage work items
    - Create actions and agents

Model API - Java Packages

Package and Classes	Description
com.fujitsu.iflow.model.event	Event related interfaces and classes.
com.fujitsu.iflow.model.util	Low level common utility classes.
com.fujitsu.iflow.model.wfadapter	Document Management System and Directory Service interfaces.
com.fujitsu.iflow.model.workflow	Contains interfaces that manage information required by the workflow process definitions and process instances
Arrow and ArrowInstance	
AttachmentRef	
DataItemRef	Wrapper object for UDAs
JavaActionSet	
Node	
Plan	Process Definition Object
ProcessInstance	
WFAdminSession	
WFObjectList	Factory for retrieving objects
WFSession	
WorkItem	
com.fujitsu.iflow.server.intf	Contains an interface that provides access to workflow data. This interface is called *ServerEnactmentContext*

WFObjectFactory

- Use this Factory class to create a WFSession to authenticate communication with the IBPM Server


Client
creates WFSession
WFObjectFactory
+getWFSession() : WFSession
+getWFObjectList(WFSession) : WFObjectList
+getWFAdminSession() : WFAdminSession
create instance
WFSession

WFObjectFactory

Method	Description
getWFSession()	Create a user session to connect to engine
getWFAdminSession()	Create admin user session
getWFObjectList(WFSession wfs)	Get object list, use filters to get definition, instance or task objects
getPlan()	Create a new definition
getPlan(WFSession wfs, long pdid)	Get definition identified by id
createProcessInstance(WFSession wfs, long pdid)	Create an instance from definition identified by id
getProcessInstance(WFSession wfs, long piid)	Get instance identified by id
getWorkItem(long workItemId, WFSession wfs)	Get workitem identified by id

WFSession

Method	Description
initForApplication()	Initializes session to use with this application
chooseApplication(String appid)	Choose application identified by id
logIn(String server, String user, String password)	Login using the credentials provides
logOut()	
validateSession()	Validates if this session is still connected with the server

WFObjectList

- Factory Object list contains one of the following types
  - Plan
  - ProcessInstance
  - WorkItem

Method	Description
openBatchedList(Filter.AllWorkItems);	Retrieves list of objects based on filter
addFilter(String udaName, String udaType, String sqlOperator, String value)	Adds a UDA based filtering criteria for the list of the objects to be retrieved
estimateCount()	Returns number of objects in the database matching the filter criteria

WFSuperSession

- Session for Tenant Manager or Super User
- Manage System Properties
  - getSystemProperties
  - setSystemProperties
- Manage Users
  - getSuperUsers, createSuperUser, deleteSuperUser
- Manage Tenants
  - getTenant, createTenant
  - activateTenant
  - getTenantProperties
  - setTenantProperties

Server Connection

```
WFSession wfSession = WFObjectFactory.getWFSession();
Properties sessionProps = getConnectionProps(IBPM_CLIENT_PROPERTIES);
wfSession.initForApplication(null, sessionProps);
wfSession.logIn("", userName, password);
wfSession.chooseApplication(applicationName);
```

Property Key	Value
TWFTransportType	EE
NamingProvider	Naming provides for the application server, e.g. weblogic.jndi.WLInitialContextFactory
NamingProviderURL	t3://localhost:49950
UserAgentServiceName	iflow.UserAgentService
TenantName	name of bpm tenant

Get Object List (Work Item)

```
Object[] workItemBatch = null;;
WFObjectList wfObjectList = WFObjectFactory.getWFObjectList(wfSession);
wfObjectList.openBatchedList(Filter.AllWorkItems);

while ((workItemBatch = wfObjectList.getNextBatch(50)) !=null) {

    for (int index = 0; index < workItemBatch.length; index++) {
         WorkItem workItem = (WorkItem) workItemBatch[index];
         System.out.print("ID " + workItem.getId());
         System.out.print(" Role " + workItem.getAssignee());
         System.out.print(" State " + workItem.getState());
         System.out.println();
    }
}
```

- Similarly Process Definitions (Plan) and Instances can be retrieved and processed

WorkItem Choice Example

```
wfObjectList.openBatchedList(Filter.MyWorkItems);
Object[] workItemBatch;

    while ((workItemBatch = wfObjectList.getNextBatch(50)) !=null) {
     for (int index = 0; index < workItemBatch.length; index++) {

        if (workItem.getName().equals("Verify Loan Request")) {

          String choices[] = workItem.getChoices();
          workItem.makeChoice("Accept");
        }

     }
 }
```

- Worklist UDAs can be also accessed and updated using workitem object
- Non-worklist UDAs are accessible from Instance object

Work Item Operations

FUJITSU

```
WorkItem workItem = wfObjectFactory.getWorkItem(workItemId, wfSession);

if (workItem.getId() == 6744) {
   workItem.accept();
}

if (workItem.getId() == 6745) {
   workItem.markAsRead();
}

if (workItem.getId() == 6746) {
   workItem.decline();
}

if (workItem.getId() == 6747) {
   String users[] = {"group1"};
   workItem.reassignTo(users);
}
```

Accessing UDA

FUJITSU

- DataItem class is a wrapper for UDAs
- UDAs can be accessed from a Process Instance object

```
DataItem udas[] = process.getDataItems();
//
DataItem uda = process.getDataItemById(String id);
//
DataItem udas[] = process.getDataItem(String name);
```

- WorkList UDAs can be accessed from WorkItem Object

```
DataItem udas[] = workItem.getWorkListDataItems()
```

Updating WorkItem UDA

- Update the value of the UDA "Name"

```
wfObjectList.openBatchedList(Filter.AllWorkItems);
Object[] workItemBatch;
  while ((workItemBatch = wfObjectList.getNextBatch(50)) !=null) {

    for (int index = 0; index < workItemBatch.length; index++) {
      WorkItem workItem = (WorkItem)workItemBatch[index];

      if (workItem.getId() == 7259) {
        workItem.startEdit();
        DataItem items[] = workItem.getWorklistDataItems();
        Properties props = new Properties();
        props.put("Name","Mr. Anderson");
        workItem.setDataItemValues(props);
        workItem.commitEdit();
      }
    }
  }
```

Creating Instance

- Create a process instance using process definition id
- Update UDA and start the instance

```
     Object[] workItemBatch = null;;
     ProcessInstantiator instantiator = WFObjectFactory.getProcessInstantiator(planId, wfs);

     instantiator.startEdit();
// update UDAs
    instantiator.commitEdit();
    ProcessInstance instance = instantiator.startNewProcess();
```

Attachment and Comment

FUJITSU

- Add attachment to process

```
process.startEdit();
process.addAttachment("filename","path");
process.commitEdit();
```

- Add Comments to process

```
process.startEdit();
process.addComment(<String comment>);
process.commitEdit();
```

Filtering and Sorting

FUJITSU

- Workflow Object Lists can be Filtered and Sorted
  - Reduces Server and Network Overhead
  - Simplifies GUI development
- Predefined Filter types provide necessary filtering options
- WFObjectList fetches object using filter defined as argument
  - Basic filter filters object type from Plan, Instance and WorkItem
  - *wfobjectList.openBatchedList(Filter.AllWorkItems)*
- Additional filters can be based on UDA value etc.
  - addFilter(..)
  - addSortOrder(..)

Predefined Filters

Process Definitions	Process Instances	Work Items
AllPlans	AllProcesses	AllWorkItems
AllArchivedPlans	AllActiveProcesses	MyWorkItems
MyPlans	AllInactiveProcesses	MyAcceptedWorkItems
MyInactivePlans	AllArchivedProcesses	MyActiveWorkItems
	AllProcessesInErrorState	MyDeclinedWorkItems
	MyProcesses	MyCompletedWorkItems
	MyActiveProcesses	AllInactiveWorkItems
	MyInactiveProcesses	AllCompletedWorkItems

Filter and Sorting Example

- Filter WorkItem
  - for a specific process plan
  - Sort by assignee

```
wfObjectList.addFilter(wfObjectList.LISTFIELD_PLAN_NAME,
                              wfObjectList.SQLOP_EQUALTO,
                              "My Plan");

wfObjectList.addSortOrder(wfObjectList.LISTFIELD_WORKITEM_ASSIGNEE,false);

 wfObjectList.openBatchedList(Filter.AllWorkItems);

Object[] workItemBatch;
  while ((workItemBatch = wfObjectList.getNextBatch(50)) !=null) {
    for(int index = 0; index < workItemBatch.length; index++) {
      WorkItem workItem = (WorkItem)workItemBatch[index];
      System.out.print("ID " + workItem.getId());
      System.out.print(" Role " + workItem.getAssignee());
    }
  }
```

Structural Edits

- Model APIs can be used to create/edit a process definition or instance, called *Structural Edit*
- Like data edit, structural edit is also done in a transaction boundary.

```
startStructuralEdit()
commitStructuralEdit()
cancelStructuralEdit()
```

- Modify the structure of a Process Instance
  - Node
  - Arrows
  - UDA
- Process Instance goes to suspended state while in structural edit
  - *isInStructuralEditMode()*

Copyright © 2010 FUJITSU LIMITED

Change Process Priority

- Process Instances can be prioritized for resource utilization
- Priorities
  - Low
  - Medium (default)
  - High
- Changes are made in *StructuralEdit* mode

```
customerSurvey.startStructuralEdit();
customerSurvey.setPriority(ProcessInstance.Priority_HIGH);
customerSurvey.commitStructuralEdit();
```

Copyright © 2010 FUJITSU LIMITED

Process Modeling

- Create Process Definitions
  - Add Node
  - Add Arrows
  - Add Conditions
  - Add UDA
  - Add Actions, Timers, Triggers etc.

  - Change States
    - Draft
    - Published
    - Private
    - Obsolete
    - Deleted

Create Process Definition

```
Plan plan1 = WFObjectFactory.getPlan();
plan1.setWFSession(wfSession);
try {
  plan1.startEdit();
  plan1.setName("Test 1");
  plan1.setTitle("Minimal Plan");
  plan1.setDesc("Build using Model API");

  Node startNode = plan1.addNode("Start",Node.TYPE_START);
  startNode.setPosition(new Point(100,150));

  Node activityNode = plan1.addNode("Activity",Node.TYPE_ACTIVITY);
  activityNode.setPosition(new Point(200,250));
  activityNode.setRole("Role");

  Node exitNode = plan1.addNode("Exit",Node.TYPE_EXIT);
  exitNode.setPosition(new Point(300,350));

  Arrow goArrow = plan1.addArrow("go",startNode,activityNode);
  Arrow stopArrow = plan1.addArrow("save",activityNode,exitNode);
  plan1.validateProcessDef();
  plan1.createProcessDef();
} catch (..) {}
```

Event Types

- Event handling using observable pattern for responding to changes
- Events
  - Plan
    - Plan changed
  - Process Instance
    - Process Instance changed
    - State changed (draft, etc.)
  - WFObjectList
    - Object added to list
    - Object deleted from list
    - Object modified in list

Copyright © 2010 FUJITSU LIMITED

Event Interfaces and Classes

- Observable Interfaces
  - PlanListener
  - PlanInstanceListener
  - WFObjectListListener
- Event Classes
  - PlanEvent
  - ProcessInstanceEvent
- Event Classes Attributes
  - Action
  - Source
  - Type

Copyright © 2010 FUJITSU LIMITED

Event Example

- Add and Remove PlanListener

```
PlanListener planHandler = new PlanListener() {
  public void planChanged(PlanEvent event) {
    System.out.println("Plan changed " + event.getSource() );
  }
};

Plan plan1 = WFObjectFactory.getPlan();
plan1.setWFSession(wfSession);
try {
  plan1.addPlanListener(planHandler);
  plan1.startEdit();
  plan1.setName("Test 1");
  Node startNode = plan1.addNode("Start",Node.TYPE_START);
  :
  plan1.removePlanListener(planHandler);

} catch (Exception ..) {
}
```

Copyright © 2010 FUJITSU LIMITED

Commit Transaction (Node)

- A Node defaults to “commit” transaction
- Node API has two methods
  - Get Transaction Status
    - boolean getNodeTxnStatus()
      - false = commit transaction (default)
      - true = do not commit
  - Change Transaction Status
    - void setNodeInTxn(boolean)
      - false = commit transaction
      - true = do not commit transaction
- Changing transaction boundary requires structural edit

Copyright © 2010 FUJITSU LIMITED

Node Transaction Example

- Create Transaction on 3 Nodes
- Change Step 1 and Step 2 to not commit
- Change Step 3 to commit transition

```
Node activityNode1 = plan1.addNode("Activity 1",Node.TYPE_ACTIVITY);
activityNode1.setNodeInTxn(true); // don't commit

Node activityNode2 = plan1.addNode("Activity 2",Node.TYPE_ACTIVITY);
activityNode2.setNodeInTxn(true); // don't commit

Node activityNode3 = plan1.addNode("Activity 3",Node.TYPE_ACTIVITY);
activityNode3.setNodeInTxn(false); // commit trans

if (activityNode3.getNodeTxnStatus() == false) {
  System.out.println("commit transaction");
}
```

Copyright © 2010 FUJITSU LIMITED

JavaActionSet

- Model APIs can be used for adding actions to Process and Node
- ActionSet is a group of actions
- can be added as
  - Init Action
  - Role Action
  - Prologue Action
  - Epilogue Action
  - Commit Action
  - Compensation Action
  - Error Action
  - Timer Action
  - OnSuspend Action
  - OnResume Action
  - OnAbort Action

Copyright © 2010 FUJITSU LIMITED

Create JavaActionSet

- Create and add init action to process definition

```
JavaActionSet myActionSet = WFObjectFactory.getJavaActionSet();

JavaAction[] myAction = MyActionSet.createJavaActions(1);
myAction[0].setActionDescription("Decide on purchase requisition");
myAction[0].setActionName("RoutePurchaseRequisition");
myAction[0].setClassName("MyPackage.MyClass");
myAction[0].setMethodName("reassignIfTooExpensive(ServerEnactmentContext, int)");

String[] args = new String[2];
args[0] = "sec";
args[1] = "uda.amount";

String params = Utils.combineParametersToXML(args);
myAction[0].setArgumentsUDANames(params);

myActionSet.setJavaActions(myAction);
plan1.setJavaActionSet(myActionSet,JavaActionSet.PLAN_INIT);
```

Java Action with Error Handling

```
JavaActionSet planInitSet = WFObjectFactory.getJavaActionSet();
JavaAction[] initAction = planInitSet.createJavaActions(1);
initAction[0].setActionName("InitPlan");
:
planInitSet.setJavaActions(initAction);

JavaActionSet planCompSet = WFObjectFactory.getJavaActionSet();
JavaAction[] compAction = planInitSet.createJavaActions(1);
compAction[0].setActionName("CompAction");
:
planCompSet.setJavaActions(compAction);

String exceptionTypes[] = {
  "java.io.FileNotFoundException", "java.lang.NumberFormatException“ };
JavaActionSet planErrSet = WFObjectFactory.getJavaActionSet();
JavaAction[] errAction = planInitSet.createJavaActions(1);
errAction[0].setActionName("ErrorAction");
errAction[0].setExceptionQualifiers(exceptionTypes);
:
planErrSet.setJavaActions(errAction);
initAction[0].setJavaActionSet(planCompSet,JavaActionSet.ACTION_COMPENSATE);
initAction[0].setJavaActionSet(planErrSet,JavaActionSet.ACTION_ERROR);
```

Server Enactment Context

- *ServerEnactmentContext* interface provides access to workflow data at runtime
- Provides access to a process instance data
- Can be used to access data in Custom Actions and Agents, and manage processes
- Access process information
- Access task information including assignees

Server Enactment Context

- Some of the methods found in the *ServerEnactmentContext* that demonstrate the capability available

```
Long      getCurrentProcessId();
long      getCurrentActivityId();
String    getProcessDefinitionName();
String    getProcessDefinitionId();
String[]  getProcessOwners();
void      setOwners(String[] users);
String    getProcessInitiator() ;
String[]  getGroupMembers(String groupName);
void      setActivityAssignees(String[] assignees);
String    evaluateScript(String script, String name);
```

Create Custom Java Actions

FUJITSU

- Steps for creating a Custom Java Action
  - Create Java Class with a default constructor
  - Define one or more business methods
  - Copy class file to the Workflow application project workspace within Interstage BPM Studio

Example: Basic Java Action

FUJITSU

- Implementation of previous example BasicJavaAction UML

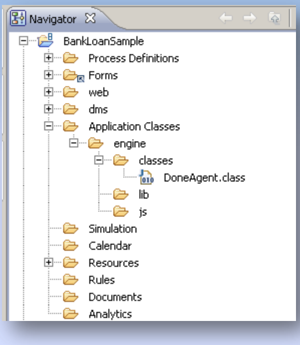
```
package com.fujitsu.fast;
/**
 * @author Administrator
 */
public class BasicAction {
 /** Creates a new instance of BasicAction */
 public BasicAction() {
 }

 public void businessMethod(ServerEnactmentContext ctx, String s)
    {
  // do stuff
 }
}
```

Bundle classes with Interstage BPM Application

FUJITSU

- The compiled java action classes needs to be copied in "engine/classes" folder in the Workflow Application Project

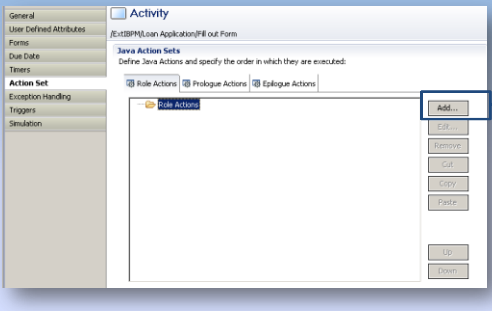

Navigator

```
BankLoanSample
  Process Definitions
  Forms
  web
  dms
  Application Classes
    engine
      classes
        DoneAgent.class
      lib
      js
  Simulation
  Calendar
  Resources
  Rules
  Documents
  Analytics
```

Assign Action to Process Definition

FUJITSU

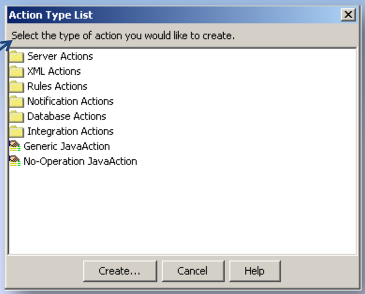
- Adding a Generic (User Created) Java Action to a Process Definition


Activity

/ExtIBPM/Loan Application/Fill out Form

Java Action Sets

Define Java Actions and specify the order in which they are executed:

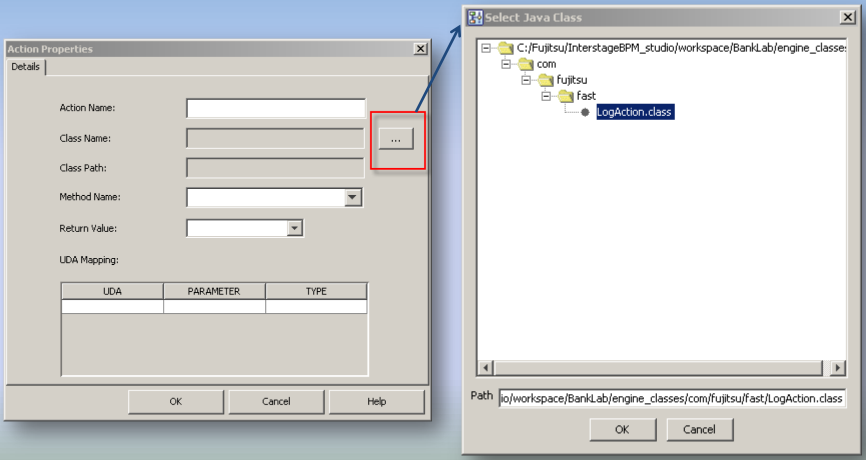

Action Type List

Select the type of action you would like to create.

- Server Actions
- XML Actions
- Rules Actions
- Notification Actions
- Database Actions
- Integration Actions
- Generic JavaAction
- No-Operation JavaAction

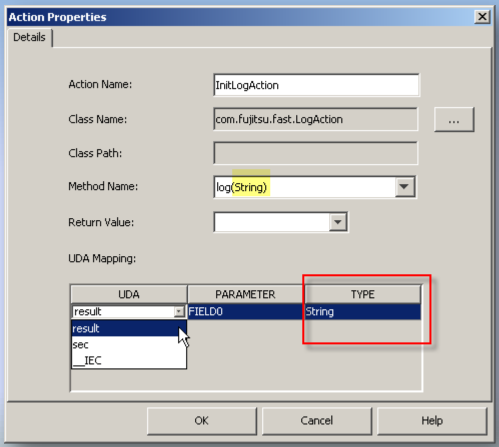
Assign Action to Process Definition

- Add a Generic Java Action to Process Definition
- Browse the engine_classes directory inside the Interstage BPM Studio project
- The “Browse…” button only browses inside the project directory



Assign Action to Process Definition

- Select the Java Action method
- Associate the UDA with the method parameters


Action Properties

Details

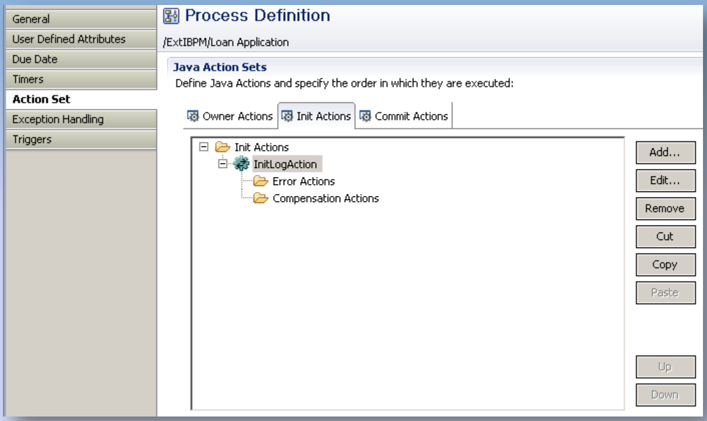
Action Name:	InitLogAction
Class Name:	com.fujitsu.fast.LogAction
Class Path:	
Method Name:	log(String)
Return Value:	

UDA Mapping:

UDA	PARAMETER	TYPE
result	FIELD0	String

Assign Action to Process Definition

- Java Action has been assigned



General
User Defined Attributes
Due Date
Timers
Action Set
Exception Handling
Triggers

Process Definition

/ExtIBPM/Loan Application

Java Action Sets

Define Java Actions and specify the order in which they are executed:

shaping tomorrow with you