[TopCoder]

LEADS 2.0 EJB Report Configuration and Quartz Services Version 1.0 Component Specification

1. Design

The Leadership Acceleration and Development System 2 (LEADS 2) applications provides managers with the means to assess and predict their employee's leadership potential. It provides the capability to assign employees to specific leadership pipelines and to predict the employee's ultimate potential.

The purpose of the architecture is to define the business service layer for the application to perform the following tasks:

- Allow a manager to assess the progress of employees by answering questions and then calculating their progress.
- Manage various entities, such as profiles, questions, pipelines and eligible populations.

The purpose of this component is to provide services for managing configuration values in the system, getting reports. This component will also provide the Quartz Job that will be responsible for automating process pipeline identification cycling. This will also require a daemon class to allow the jobs to be run.

1.1 Design Patterns

Strategy pattern – This component provides ReportService and SystemConfigurationPropertyService interfaces together with their implementations that can possibly be used in some external strategy context. ReportServiceBean uses pluggable EmployeeProfileService and LDAPUserService interfaces. ReportSortingComparator<T> implements the pluggable Comparator<T> interface, so that it can be used as parameter of Collections.sort() method to sort list. TogglePipelineIdentificationCycleJob implements the pluggable Job interface, so that it can be invoked by quartz scheduler.

DAO/DTO pattern –ReportService retrieves EmployeeDTO and Report DTO; SystemConfigurationPropertyService acts as a DAO for SystemConfigurationProperty DTO.

1.2 Industry Standards

J2EE 1.5, EJB 3.0, JPA 1.0

1.3 Required Algorithms

1.3.1 Searching for all employees of a manager in an organization

When searching for the employees in the organization, the search must be recursive. A manager may have several employees, who in turn may be managers of other employees, etc. We need all of these employees. The following algorithm should be used:

1. Do search for all employees under a specific manager with LDAPUserService#searchUsers() method.
2. Add the employees to a running list.
3. While running list is not empty, do the following:
    3.1. Remove the first employee from running list, and add it to the result list, this employee will be used in the following steps.
    3.2. Get all employees under this employee as if he was a manager with LDAPUserService.searchUsers() method.
    3.3. For each employee retrieved in step 3.2, do the following:
        3.3.1. Use EmployeeProfileService#getEmployeeProfilesByManager() to get all employees to whom the current employee is a matrix manager.
    3.4. Add all employees retrieved in step 3.2 and step 3.3 to the running list.

Note that there may be duplication for the retrieved employees, duplicated employees need to be remove so that each employee in the result list should have a unique cnum. By the end of this process we will have one list of all employees that are directly or indirectly under the given manager.

Please refer to method document of ReportServiceBean#searchEmployees() for detail implementation notes.

1.3.2 Running the TogglePipelineIdentificationCycleDaemon

TogglePipelineIdentificationCycleDaemon class is the class that bootstraps the quartz scheduler. It is expected to be invoked via a command line interface.

Refer to section “4.3.4 Invoking TogglePipelineIdentificationCycleDaemon via command line” to see the sample invocation command.

The command entered by user can provide one parameter which should be the path of the properties file. If this parameter is not provided, TogglePipelineIdentificationCycleDaemon will try to load a default properties file (“TogglePipelineIdentificationCycleDaemon.properties”) in class path. The properties file will contain configurable values used to create scheduler jobs which will toggle pipeline identification cycle.

Refer to section “3.2.1 Properties file used by TogglePipelineIdentificationCycleDaemon” for detail information as to what these configurable values are.

Refer to section “4.3.5 Sample properties file for TogglePipelineIdentificationCycleDaemon” to see a sample properties file.

1.3.3 Logging

The EJBs, TogglePipelineIdentificationCycleJob and TogglePipelineIdentificationCycleDaemon will log activities and exceptions using Log4j. They will log errors at Error level, and method entry/exit information at DEBUG level. Specifically, logging will be performed as follows, if logging is turned on.

- Method entrance and exit will be logged with DEBUG level.

- Entrance format: [Entering method {className.methodName}]
- Exit format: [Exiting method {className.methodName}]. Only do this if there are no exceptions.

- Method request and response parameters will be logged with DEBUG level

- Format for request parameters: [Input parameters[{request_parameter_name_1}:{ request_parameter_value_1}, {request_parameter_name_2}:{ request_parameter_value_2}, etc.)]]
- Format for the response: [Output parameter {response_value}]. Only do this if there are no exceptions and the return value is not void.
- If a request or response parameter is complex, use its toString() method. If that is not implemented, then print its value using the same kind of name:value notation as above. If a child parameter is also complex, repeat this process recursively.

- All exceptions will be logged at ERROR level, and automatically log inner exceptions as well.

- Format: Simply log the text of exception: [Error in method {className.methodName}:Details {error details}]

In general, the order of the logging in a method should be as follows:

1. Method entry
2. Log method entry
3. Log method input parameters
4. If error occurs, log it and skip to step7
5. Log method exit
6. If not void, log method output value
7. Method exit

1.4 Component Class Overview

hr.leads.services

ReportService [interface]

This service provides methods to retrieve different kinds of reports, these reports are direct / organization nine box report, direct / organization summary report, direct / organization rollup report and high level summary report.

SystemConfigurationPropertyService [interface]
This service provides methods to manage SystemConfigurationProperty entity. It also provides the getPipelineCycleStatus() and updatePipelineCycleStatus() methods as convenience methods for managing the configuration value for pipeline cycle status.

hr.leads.services.ejb

ReportServiceLocal [interface]
This interface represents the local interface for ReportService session bean. It extends ReportService interface and provides no additional methods.

SystemConfigurationPropertyServiceLocal [interface]
This interface represents the local interface for SystemConfigurationPropertyService session bean. It extends SystemConfigurationPropertyService interface and provides no additional methods.

SystemConfigurationPropertyServiceRemote [interface]
This interface represents the remote interface for SystemConfigurationPropertyService session bean. It extends SystemConfigurationPropertyService interface and provides no additional methods.

BaseReportConfigurationServiceBean [abstract]
This is the base class for all EJBs defined in this component. It provides Logger instance created according to injected logger name.

ReportServiceBean
This class is an EJB that implements ReportService business interface to retrieve different kinds of reports. It extends BaseReportConfigurationServiceBean class. It uses instances of EmployeeProfileService and LDAPUserService to retrieve employee information. It uses ReportSortingComparator to sort employees / report records based on given sort columns.

SystemConfigurationPropertyServiceBean
This class is an EJB that implements SystemConfigurationPropertyService business interface to manage SystemConfigurationProperty entity. It extends BaseReportConfigurationServiceBean class. It accesses entities in persistence using JPA EntityManager.

ReportSortingComparator<T>
This comparator is used by ReportServiceBean to sort report records or employee profiles. It implements Comparator<T> interface. It uses given objectType and sortColumns to determine the order of the two objects.

hr.leads.services.jobs

TogglePipelineIdentificationCycleJob
This class will toggle pipeline identification cycle by updating the pipeline cycle status. It implements Job interface so that it can be invoked by quartz as a scheduler job. It will retrieve system configuration property service and pipeline cycle status from given execution context, and use them to update status.

TogglePipelineIdentificationCycleDaemon
This is the class that bootstraps the quartz scheduler from the command line. It reads properties file specified by user in command line, and use it to start scheduler jobs.

1.5 Component Exception Definitions

hr.leads.services.jobs

TogglePipelineIdentificationCycleDaemonException

This exception is thrown by TogglePipelineIdentificationCycleDaemon if any error occurs while starting scheduler. It extends Exception.

TogglePipelineIdentificationCycleJobExecutionException
This exception is thrown by TogglePipelineIdentificationCycleJob if any error occurs while updating pipeline cycle status. It extends JobExecutionException.

1.6 Thread Safety

This component is not technically thread-safe because the EJB implementation classes are mutable. For these EJB implementations, since they are managed by the container and their states will not change after configuration, they are effectively thread-safe.

Implementations of ReportService, SystemConfigurationPropertyService, ReportServiceLocal and SystemConfigurationPropertyServiceLocal and SystemConfigurationPropertyServiceRemote must be thread-safe when their configurable properties do not change after initialization.

BaseReportConfigurationServiceBean, ReportServiceBean and SystemConfigurationPropertyServiceBean are mutable and not thread-safe. But they are always used in thread-safe manner in EJB container because their states don't change after initialization.

ReportSortingComparator<T> is immutable and thread safe.

TogglePipelineIdentificationCycleJob is immutable, it's thread safe assuming that the execute() method's context parameter do not change during the execution of job.

TogglePipelineIdentificationCycleDaemon is immutable and thread safe.
All the exception classes defined in this component are mutable and not thread-safe, and they must be used in a thread safe manner.

2. Environment Requirements

2.1 Environment

- Java 1.6/J2EE 1.5
- WebSphere Application Server ND 7.0
- DB2 for z/OS version 9, New Function Mode
- LDAP
- JPA 1.0 (with Apache OpenJPA 1.2.1: http://openjpa.apache.org/)
- Spring 2.5.6
- Quartz 1.8.3
- Log4j 1.2.15

2.2 TopCoder Software Components

LEADS 2.0 EJB Model and Exceptions 1.0 - Provides entities and exceptions used by this component.

LEADS 2.0 EJB User Services 1.0 - Provides the EmployeeProfileService and LDAPUserService used by this component.

2.3 Third Party Components

Log4j 1.2.15 (http://logging.apache.org/log4j/1.2/)
OpenJPA 1.2.1 (http://openjpa.apache.org/)
Quartz 1.8.3 (http://www.quartz-scheduler.org/)

3. Installation and Configuration

3.1 Package Name

hr.leads.services
hr.leads.services.ejb
hr.leads.services.jobs

3.2 Configuration Parameters

3.2.1 Properties file used by TogglePipelineIdentificationCycleDaemon

The following table shows all the configurable values in properties file:

Parameter	Description	Values
Provider URL	Used to create InitialContext instance and retrieve SystemConfigurationPropertyService using JNDI. The key name should be “provider_url”	String. Required. Cannot be empty. Should be the URL for JNDI service.
Initial Context Factory	Used to create InitialContext instance and retrieve SystemConfigurationPropertyService using JNDI. The key name should be “initial_context_factory”	String. Required. Cannot be empty. Should be full class name for initial context factory.
System Configuration Property Service JNDI Name	The name of SystemConfigurationPropertyService stored in JNDI directory. The key name should be “system_configuration_property_service_jndi_name”	String. Required. Cannot be empty.
Execution Time and Status (Can appear for many times)	The execution time and pipeline cycle status for TogglePipelineIdentificationCycleJob. Every execution time and status pair should use “execution_time_and_status_” as the prefix for key, the value should contain execution time and pipeline cycle status separated using “@@” as separator. Execution time should use valid quartz cron expression. Pipeline cycle status should be the status defined in PipelineCycleStatus enum (case sensitive).	String. Required. Cannot be empty. The value should be: “execution time” + “@@” + “status”. Should at least have one execution time and status pair.

3.2.2 BaseReportConfigurationServiceBean

The following table shows all the configurable values that can be injected to BaseReportConfigurationServiceBean:

Parameter	Description	Values
loggerName	The logger name used to retrieve logger. @Resource	String. Optional. If not set, the logger will be retrieved using current class name.

3.2.3 ReportServiceBean

The following table shows all the configurable values that can be injected to ReportServiceBean:

Parameter	Description	Values
employeeProfileService	Represents the EmployeeProfileService instance used to retrieve employee profiles. @EJB(name="employeeProfileService")	EmployeeProfileService. Required. Cannot be null.
ldapUserService	Represents the LDAPUserService instance used to retrieve user detail. @EJB(name=" ldapUserService ")	LDAPUserService. Required. Cannot be null.
submittedStatusName	The name that should match EmployeeProfileStatus#name if the status is submitted.	String. Optional. Default to “Assessed”

	@Resource	Cannot be null / empty.
notSubmittedStatusName	The name that should match EmployeeProfileStatus#name if the status is not submitted. @Resource	String. Optional. Default to “Not Assessed” Cannot be null / empty.

3.2.4 SystemConfigurationPropertyServiceBean

The following table shows all the configurable values that can be injected to SystemConfigurationPropertyServiceBean:

Parameter	Description	Values
entityManager	Represents the EntityManager instance to access entities in persistence. @PersistenceContext(name="persistenceUnit", type=PersistenceContextType.TRANSACTION)	EntityManager. Required. Cannot be null.
pipelineCycleStatusPropertyName	Represents the system configuration property name for pipeline cycle status. @Resource	String. Required. Cannot be null / empty.

3.3 Dependencies Configuration

None

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute ‘ant test’ within the directory that the distribution was extracted to.

4.2 Required steps to use the component

The entities, interfaces and beans should be deployed in an EJB 3.0 container.

4.3 Demo

4.3.1 API usage

```
// create initial context
Properties prop = new Properties();
... // set properties like Context.PROVIDER_URL and Context.INITIAL_CONTEXT_FACTORY
InitialContext ctx = new InitialContext(prop);

// retrieve report service
ReportService reportService = (ReportService) ctx.lookup("ejb/ ReportService/local");

// retrieve direct nine box report
List<EmployeeDTO> dtos = reportService.getDirectNineBoxReport(managerCNUM, null);
// the direct nine box report for given manager should be returned

// retrieve organization nine box report
dtos = reportService.getOrganizationNineBoxReport(managerCNUM, null, filterManagerCNUM);
// the organization nine box report for given manager should be returned, the return list
should be filtered using filterManagerCNUM

// retrieve direct summary report
dtos = reportService.getSummaryReport(managerCNUM, ReportType.DIRECT_REPORT, new String[]
{ “cnum” });
// the direct summary report for given manager should be returned and sorted using
EmployeeDTO#employee#cnum field in ascending order

// retrieve organization summary report
dtos = reportService.getSummaryReport(managerCNUM, ReportType.ORGANIZATION, new String[]
{ “cnum” });
```

```
// the organization summary report for given manager should be returned and sorted using
EmployeeDTO#employee#cnum field in ascending order

// retrieve direct rollup report
Report report = reportService.getDirectRollupReport(managerCNUM, new String[]
{ “totalEmployees” });
// the direct rollup report for given manager should be returned and Report#records should be
sorted using ReportRecord#totalEmployees

// retrieve organization rollup report
report = reportService.getOrganizationRollupReport(managerCNUM, filterManagerCNUM, new
String[] { “totalEmployees” });
// the organization rollup report for given manager should be returned, the data for report
should be filtered by filterManagerCNUM and Report#records should be sorted using
ReportRecord#totalEmployees

// retrieve high level summary report
report = reportService.getHighLevelSummaryReport(filter, new String[] { “totalEmployees” });
// the high level summary report should be retrieved based on given high level summary report
filter, Report#records should be sorted using ReportRecord#totalEmployees

// retrieve system configuration property service
SystemConfigurationPropertyService scpService = (SystemConfigurationPropertyService)
ctx.lookup("ejb/SystemConfigurationPropertyService/local");


// set system configuration property value
scpService.setSystemConfigurationPropertyValue(“property1”, “value1”);
scpService.setSystemConfigurationPropertyValue(“property2”, “value2”);
// assume that these properties are not set before, 2 new SystemConfigurationProperty
entities should be created and saved to database:
```

id	name	value
1	property1	value1
2	property2	value2

```
// set system configuration property value
String value = scpService.getSystemConfigurationPropertyValue(“property1”);
// the retrieved value should be “value1”

// get all system configuration properties
List<SystemConfigurationProperty> properties =
scpService.getAllSystemConfigurationPropertyValues();
// the returned list should contain all the properties set in previous steps

// update pipeline cycle status
scpService.updatePipelineCycleStatus(PipelineCycleStatus.OPEN);
// assume that this is the first call to set pipeline cycle status and the
SystemConfigurationPropertyServiceBean#pipelineCycleStatusPropertyName is set to
“pipelineCycleStatus”, a new SystemConfigurationProperty should be added to database:
```

id	name	value
1	property1	value1
2	property2	value2
3	pipelineCycleStatus	OPEN

```
// update pipeline cycle status again
scpService.updatePipelineCycleStatus(PipelineCycleStatus.CLOSED);
// the SystemConfigurationProperty entity created in previous step should be updated to new
status:
```

id	name	value
1	property1	value1
2	property2	value2
3	pipelineCycleStatus	CLOSED

```
// get pipeline cycle status
PipelineCycleStatus status = scpService.getPipelineCycleStatus();
// the status should be PipelineCycleStatus.CLOSED
```

4.3.2 *Sample EJB configuration (ejb-jar.xml)*

This file shows configuration for SystemConfigurationPropertyServiceBean, the configuration is similar to other service beans.

```
<?xml version="1.0" encoding="utf-8"?>
<ejb-jar xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation=
    "http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd"
         version="3.0">
  <description>System Configuration Property service EJB</description>
  <display-name> System Configuration Property service EJB</display-name>
  <enterprise-beans>
    <session>
      <ejb-name>SystemConfigurationPropertyService</ejb-name>
      <local>
        hr.leads.services.ejb.SystemConfigurationPropertyServiceLocal
      </local>
      <ejb-class>
        hr.leads.services.ejb.SystemConfigurationPropertyServiceBean
      </ejb-class>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <env-entry>
        <env-entry-name>pipelineCycleStatusPropertyName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>pipelineCycleStatus</env-entry-value>
      </env-entry>
    </session>
    <session>
                <ejb-name>reportService</ejb-name>
                <local>hr.leads.services.ejb.ReportServiceLocal</local>
                <ejb-class>hr.leads.services.ejb.ReportServiceBean</ejb-class>
                <session-type>Stateless</session-type>
                <transaction-type>Container</transaction-type>
                <env-entry>
                        <env-entry-name>submittedStatusName</env-entry-name>
                        <env-entry-type>java.lang.String</env-entry-type>
                        <env-entry-value>SubmittedStatus</env-entry-value>
                </env-entry>
                <env-entry>
                        <env-entry-name>notSubmittedStatusName</env-entry-name>
                        <env-entry-type>java.lang.String</env-entry-type>
                        <env-entry-value>NotSubmitttedStatus</env-entry-value>
                </env-entry>
                <ejb-ref>
            <ejb-ref-name>employeeProfileService</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>hr.leads.services.EmployeeProfileServiceLocal</home>
        </ejb-ref>
            <ejb-ref>
            <ejb-ref-name>ldapUserService</ejb-ref-name>
            <ejb-ref-type>Session</ejb-ref-type>
            <home>hr.leads.services.LDAPUserServiceLocal</home>
        </ejb-ref>
    </session>
  </enterprise-beans>
</ejb-jar>
```

4.3.3 *Sample mapping for SystemConfigurationProperty*

```
<entity class="hr.leads.services.model.jpa.SystemConfigurationProperty">
        <table name="system_configuration" />
        <attributes>
                <id name="id">
                        <column name="id" nullable="false" />
                </id>
                <basic name="name">
```

```
                        <column name="name" length="45" unique="true" nullable="false" />
                </basic>
                <basic name="value">
                        <column name="value" length="45" nullable="false" />
                </basic>
        </attributes>
</entity>
```

4.3.4 *Invoking TogglePipelineIdentificationCycleDaemon via command line*

Example invocation is shown below:

```
"C:\Program Files\IBM\Java60\jre\bin\java" -cp ...
hr.leads.services.jobs.TogglePipelineIdentificationCycleDaemon c:\scheduler.properties
```

Or

```
"C:\Program Files\IBM\Java60\jre\bin\java" -cp ...
hr.leads.services.jobs.TogglePipelineIdentificationCycleDaemon
```

4.3.5 *Sample properties file for TogglePipelineIdentificationCycleDaemon*

```
provider_url=127.0.0.1:1099
initial_context_factory=org.jnp.interfaces.NamingContextFactory
system_configuration_property_service_jndi_name=systemConfigurationPropertyService
# set the status to open at 9AM every date
execution_time_and_status_1=0 0 9 ? * * @@ OPEN
# set the status to closed at 5PM every date
execution_time_and_status_2=0 0 17 ? * * @@ CLOSED
```

5. Future Enhancements

None