

2nd Derivative Information Portal

Technical Documentation



by

Randr, Inc.

www.RANDRINC.com

Contents

	Page
Overview	3
Hibernate and Database	4
Using More than One Database	6
Search Conditions	6
Tag Mappings (iSeries DB2)	7
Database	10
Adding fields to existing Reports	11
Creating new Reports	13
Security	Separate Document
Reports Samples	Separate Document

Overview:

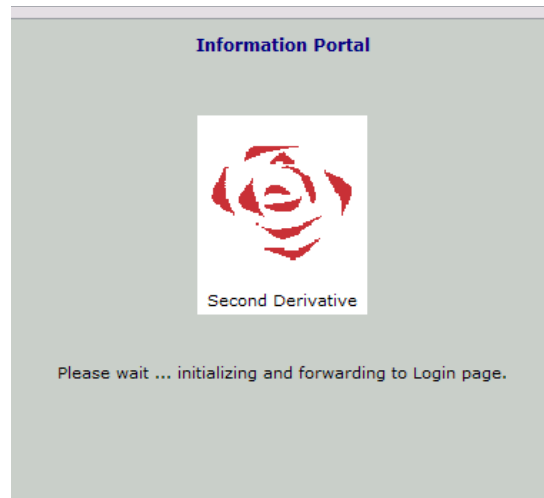
Infoportal is a web application used for real time key business indicators presented in a centralized web browser with drill down capabilities, Originally, written in 2000 in Net.Data on the iSeries for executives making regular trips internationally and around the country who needed real time information on the financials, inventory and sales.

Infoportal is NOT a report writer. It required mapping and data validation, but IS worth it. Ensures consistent dashboard and operational metrics across the organization, Easy downloads for further analysis. Easy to customize. Intuitive to use.

Infoportal_Salesportal is designed specifically to run against the Sales Portal database. The database scripts should be run against that DB and the new tables will be added to the Sales Portal database. The latest version of Sales Portal includes these DB scripts as well as a menu option that will open the Information Portal in a separate window.

For Security setups, see the separate security documentation. You need to set up a separate user id and password for everyone who will have access to the Infoportal. Infoportal allows you to setup your Salesmen so that they are restricted to one or more territories. Also by the security profiles you can restrict which reports/queries each salesman has access to.

The hibernate.cfg.xml contains the location, name of your database and password.
The globalAppConfig.xml jkmount has to be set to the Infoportal context.
Run the DB creation and population scripts.



Hibernate and Database access in the Randr 2D Infoportal (Information Portal)

Data base access is through Hibernate and through SQL queries in xml_config/infoportal/storedQueries. Here is more detail:

JavaSource/com/randr/infoportal/config_files - this contains the hibernate xml files for the key tables used in security and the drop down combo boxes on the main screen.

.1 Infoportal (IP) Tables:

- .1 ApplicationGroup.hbm.xml – part of IP security tables
- .2 Country.hbm.xml – here because of links into our 2nd Derivative system. Used in webuser table when you set up a user. Not used for anything else.
- .3 SecurityAction.hbm.xml – part of IP security tables
- .4 SecurityAuthorization.hbm.xml – part of IP security tables
- .5 SecurityProfile.hbm.xml – part of IP security tables
- .6 State.hbm.xml - here because of links into our 2nd Derivative system. Used in webuser table when you set up a user. Not used for anything else.
- .7 UserCompanyRelation.hbm.xml – part of IP security tables
- .8 UserType.hbm.xml – part of IP security tables
- .9 Webuser.hbm.xml – part of IP security tables, valid system users

.2 Your Application Tables- used in drop down combo boxes:

1. Company.hbm.xml – your companies

table="company" - the name of your company table
column name="company_id" - the column name for the unique company identifier
column="company" - the column name for the description or name of a company

2. Customer.hbm.xml – your customers

table="WEBUSER" - the name of the customer table
name="CUSTOMER_NO" - the column name for the unique customer identifier
column="company" - the column name for the customer description or name

3. Installer.hbm.xml – this is also used for CSR or customer service rep or a 2nd territory/salesman.

table="WEBUSER" - the name of the installer/csr/2nd territory table. Note that in the base 2nd Derivative Infoportal we use the same table for customer, installer and territory.
name="CUSTOMER_NO" - the column name for the unique installer identifier.
column="first_name" - the column name for the installer first name
column="last_name" - the column name for the installer last name

4. Territory.hbm.xml – this is also used for salesman

table="WEBUSER" - the name of the installer/csr/2nd territory table. Note that in the base 2nd Derivative Infoportal we use the same table for customer, installer and territory.
name="CUSTOMER_NO" - the column name for the unique installer identifier.
column="first_name" - the column name for the installer first name
column="last_name" - the column name for the installer last name

5. UserCompanyRelation.hbm.xml – part of IP security tables. Links your companies with security(users are authorized to all or specific companies).

column = "CUSTOMER_NO" - the column name for the unique company identifier. This should be the same as the Company.hbm.xml, above.

6.UserCustomerRelation.hbm.xml - part of IP security tables. Links your customers with security(users are authorized to all or specific customers).

column = "CUSTOMER_NO" - the column name for the unique customer identifier. This should be the same as the Customer.hbm.xml, above.

7.UserInstallerRelation.hbm.xml - part of IP security tables. Links your installers/CSR's with security(users are authorized to all or specific installers/CSR's).

column = "CUSTOMER_NO" - the column name for the unique installer identifier. This should be the same as the Installer.hbm.xml, above.

8.UserTerritoryRelation.hbm.xml - part of IP security tables. Links your territories/salesmen with security(users are authorized to all or specific territories).

column = "CUSTOMER_NO" - the column name for the unique territory identifier. This should be the same as the Territory.hbm.xml, above.

3. hibernate.cfg.xml - this defines the link between the application and the database.

Postgres - the base 2nd Derivative Infoportal uses **postgres** as the DB, so this contains the location of the DB and the user name and password to connect to the DB:

4. backend.hibernate.cfg.xml - allows for a 2nd link to a different database. (see Below)

POSTGRES:

```
<property
name="dialect">org.hibernate.dialect.PostgreSQLDialect</property>
<property
name="hibernate.connection.driver_class">org.postgresql.Driver</property
>
<property
name="hibernate.connection.url">jdbc:postgresql://test.randrinc.com:
5432/sd_base</property>
<property
name="hibernate.connection.username">postgres</property>
<property
name="hibernate.connection.password">postgres</property>
```

AS/400/iSeries DB2-

```
<property
name="dialect">net.sf.hibernate.dialect.DB2400Dialect</property>
<property name="connection.username">user</property> (setup
this user with Initial Menu = *signoff)
<property name="connection.password">password</property>
<property name="connection.libraries">Infoportal, MYFILES,
MOREFILES</property> (these are the libraries that will be accessed)
<property
name="connection.url">jdbc:as400://10.1.1.1/</property>
<property
name="connection.driver_class">com.ibm.as400.access.AS400JDBCdriver</pr
operty>
```

Using more than one database with Infoportal:

in the class businessObjects/BaseSecuredAction:

```
    protected SessionFactory getSessionFactory() {
        ServletContext servletContext =
getServlet().getServletContext();
        SessionFactory sessionFactory
            = (SessionFactory)servletContext.getAttribute(
                HibernatePlugIn.SESSION_FACTORY_KEY);
        return sessionFactory;
    }

    protected SessionFactory getBackendSessionFactory() {
        ServletContext servletContext =
getServlet().getServletContext();
        SessionFactory sessionFactory
            = (SessionFactory)servletContext.getAttribute(
                HibernatePlugIn.SESSION_FACTORY_BACKEND_KEY);
        return sessionFactory;
    }
```

in the class utilities/HibernatePlugIn:

```
    public static final String SESSION_FACTORY_KEY =
SessionFactory.class.getName() + "_1"; - this will use the
hibernate.cfg.xml
    public static final String SESSION_FACTORY_BACKEND_KEY =
SessionFactory.class.getName() + "_2"; - this will use the
backend.hibernate.cfg.xml

    private String _configFilePath = "/infoportal/hibernate.cfg.xml";
    private String _backendConfigFilePath =
"/infoportal/config_files/backend.hibernate.cfg.xml";
```

Using the correct sessionFactory in the Action class:

```
//Instantiate ActivitySummary object and store in request
        ArSummaryDetail arSummaryDetail = new ArSummaryDetail(
            ((ActivitySummaryForm)form),

            (Webuser)request.getSession().getAttribute("webuser"),
            sessionFactory());
```

either use:

getBackendSessionFactory() or getSessionFactory() depending on which DB this query will connect to.

Example access a DB inside the class: (we used this for 1 query access another DB)

```
    public void init() {
        Connection con = null;
        try{
            Class.forName("org.postgresql.Driver");
```

```

        con = DriverManager.getConnection("jdbc:postgresql://
10.1.1.114:5432/bugs",
                                        "postgres", "postgres");
//      Make sure conditions and params are cleared
      queryManager = new QueryManager(con);

      performQuery();
      //con.commit();
      //con.close();
    } catch (Exception e) {
      e.printStackTrace();
      try{
        if(con!=null && !con.isClosed()){
          con.close();
        }
      } catch (Exception e2){
        e2.printStackTrace();
      }
    }
  }
}

```

Search Conditions:

The main search conditions on the custSelect.jsp:

Customer

- .1config_files/infoportal/customer.hbm.xml
- .2businessObjects/Customer
- .3viewObjects/CustomerList
- .4StoredQuery/Customer combobox population or Customer combobox population no display if you set:
- .5<DISPLAY_CUSTOMERS>FALSE</DISPLAY_CUSTOMERS> (e.g. If DB is too large)

Company

- .1config_files/infoportal/ompany.hbm.xml
- .2businessObjects/Company
- .3viewObjects/CompanyList
- .4StoredQuery/Company combobox population

Territory

- .1config_files/infoportal/territory.hbm.xml
- .2businessObjects/Territory
- .3viewObjects/TerritoryList
- .4StoredQuery/Territory combobox population

Status Id (also CSR, Installer and Group)

- .1config_files/infoportal/installer.hbm.xml
- .2businessObjects/Installer
- .3viewObjects/InstallerList
- .4StoredQuery/Installer combobox population

TagMappings –This only applies to AS/400/iSeries DB2.

This is where you define the libraries that are accessed. Note, that these libraries all have to be defined in the hibernate.cfg.xml in the connection.libraries statement

```
<property name="connection.libraries">Infoportal, MYFILES, MOREFILES</property>
```

WebContent/xml_config/infoportal:

For each iSeries library you need the following:

```
<library>
  <tag-name>LIB1</tag-name>
  <library-name>RDRFILES</library-name>
  <description>Sample Library</description>
  <datasource>foo:hibernate/SessionFactory</datasource>
</library>

<library>
  <tag-name>LIB4</tag-name>
  <library-name>RDRFILES4</library-name>
  <description>Sample Library</description>
  <datasource>foo:hibernate/SessionFactory</datasource>
</library>
```

The tag-name is what you use in the StoredQueries.

The library-name is the iSeries library name where the files are located.

You are NOT limited as to how many libraries you can define.

This is an example of how the tag-name is used in the Stored Queries

```
<query>
  <name>Customer Name</name>
  <description>no desc yet</description>
  <set>custSelection.jsp</set>
  <form>
    select
      cname as customer_name
    from
      [LIB4].RDRCUST

  </form>
</query>
```

When the query executes it will replace the [LIB4] with the library associated with tag-name LIB4 in the tag mappings. In this example this is the query that will run:

```
select
  cname as customer_name
from
  rdrfiles4.rdrcust
```

Where rdrfiles4 is the library and rdrcust is the file.
It is OK to have the same library defined more than once.

```
from [LIB3].ITMHST
  inner join [LIB1].vinitem on IHITEM=ICITEM
  left outer join [LIB1].varcust on IHCMP=RCMP and IHCUST=RCUST
```

This is an example where you can see it is ok to have files in different libraries within the same query.

Database:

webuser_ip	
customer_no	numeric(10)
user_type	numeric(3)
user_name	varchar(30)
password	varchar(20)
active	numeric(1)
first_name	varchar(30)
last_name	varchar(30)
address	varchar(50)
city	varchar(30)
zip	varchar(15)
state_id	numeric(6)
country_id	numeric(6)
phone	varchar(15)
company	varchar(30)
email	varchar(80)
exported	bpchar(1)
no_of_stores	numeric(6)
resale_no	varchar(30)
ref_cust_no	numeric(10)
fax	varchar(15)
system_type_id	numeric(5)
territory_id	numeric(5)
lob_id	numeric(5)
software_type_id	numeric(5)
verified_id	numeric(5)
status_id	numeric(5)
creation_date	timestamp(8)
last_logged	timestamp(8)
logon_counter	numeric(9)
sec_profile_fk	numeric(10)

user_customer_relation	
customer_rel_pk	numeric(10)
customer_no	numeric(10)
customer_fk	varchar(10)
company_fk	numeric(10)
relationship	bpchar(1)

user_company_relation	
company_rel_pk	numeric(10)
customer_no	numeric(10)
company_fk	numeric(10)
relationship	bpchar(1)

user_territory_relation	
territory_rel_pk	numeric(10)
customer_no	numeric(10)
territory_fk	numeric(10)
relationship	bpchar(1)

user_installer_relation	
installer_rel_pk	numeric(10)
customer_no	numeric(10)
installer_fk	numeric(10)
relationship	bpchar(1)

security_actions	
sec_act_pk	numeric(10)
action_desc	varchar(50)
action_category	numeric(5)
entry_point	numeric(1)
comments	varchar(500)
app_group_fk	numeric(10)

application_group	
app_group_pk	numeric(10)
app_desc	varchar(50)

security_profile	
sec_profile_pk	numeric(10)
sec_desc	varchar(500)
sec_short_desc	varchar(50)
app_group_fk	numeric(10)

security_auth	
auth_pk	numeric(10)
sec_profile_fk	numeric(10)
sec_act_fk	numeric(10)
sec_code	varchar(1)

user_type	
user_type_pk	numeric(10)
description	varchar(500)
short_desc	varchar(30)

Adding fields to existing Reports:

To add additional fields to an existing reporting here are the key areas for changes:

1. Query – add the fields into the query in the storedQuery.xml
2. jsp – add the fields into the jsp
3. Decorator class – on the totals line add the additional columns

Here is an example, changes are in red:

1. Query

```
<query>
  <name>SalesActionsByTerritoryNextDetail</name>
  <description>IP_Salesportal</description>
  <set></set>
  <form>select
    t.territory,
    p.customer_company,
    p.contact_name,
    a.action,
    s.status,
    date(sa.action_date) as action_date,
    sa.action_date::date - current_date::date as days_left,
    sa.action_note
  from prospect p
  left join prospect_sales_action sa on p.prospect_id =
    sa.prospect_id
  inner join action a on sa.sales_action_id=a.action_id
  left join territory t on p.territory_id = t.territory_id
  left join status s on p.status_id=s.status.id
  where
    [CUSTOMER_CONDITION]
    [COMPANY_CONDITION]
    [TERRITORY_CONDITION]
    --[STATUSID_CONDITION]

    sa.sales_action_id is not null and sa.action_date is
not null and
    sa.action_date::date - current_date::date >= 0 and
sa.action_status = 0
    order by
      territory,action_date

  </form>
</query>
```

JSP

```
<display:table class="display"
name="salesActionsByTerritoryNextDetail.results"
    defaultsort="1"
    export="true"

decorator="com.randr.infoportal.activitySummary.salesActions.SalesActio
nsByTerritoryNextDetailDecorator"
    requestURI= "<%= appConfig.getMountPath() +
"salesActionsByTerritoryNextDetail.do" %>">
<display:column class="display:text" title="Territory"
property="territory"></display:column>
<display:column title="Company Name"
property="customer_company"></display:column>
<display:column title="Status Id" property="status"></display:column>
<display:column title="Contact Name" property="contact_name"
align="left"></display:column>
<display:column title="Sales Action" property="action"
align="left"></display:column>
<display:column title="Action Date" property="action_date" align="right"
decorator =
"com.randr.infoportal.viewObjects.columnDecorators.SqlDateToDisplayDate
ColumnDecorator"></display:column>
<display:column title="Days Left" property="days_left"
align="left"></display:column>
<display:column title="Action Note" property="action_note"
align="left"></display:column>

</display:table>
```

Decorator Class

```
public String finishRow()
{
    BasicDynaBean results = (BasicDynaBean) this.getObject();
    totalActions = totalActions.add(new BigDecimal(1)) ;

    StringBuffer sb = new StringBuffer( 1000 );

    if( count == this.getList().size() - 1 ) {
        sb.append( "<tr class=\"footer\" >" );
        sb.append( "<td colspan=6>Total Sales Actions:</td>" );

        sb.append( "<td colspan=2>" +
DecoratorUtilities.formatNoDecimal(totalActions) + "</td></tr>" );

    }

    count++;

    return sb.toString();
}
```

Creating New Reports

When you create a new query, here are the key steps:

1. Query – add the query into the StoreQueries.xml. Add any CONDITION statements.
2. Struts config - add the report
3. tiles/tiles-defs.xml if you are using tiles – add the report
4. jsp (under Content if using tiles) – create the jsp
5. Java Classes
 1. Business logic – remember to add the conditions
 2. Action class
 3. Decorator (for totals)
 4. Element – if we have additional calculations to do and are not displaying the query results directly.
6. Add the link onto a menu or other jsp
7. Security -
 1. add the security into the baseClasses/names/SecurityNames
 2. add the security into the Action Class
 3. Add the security into the DB scripts and into the security_actions table
 4. Authorize users to the new options