NTT DATA

intra-mart The Power for Enterprise Web Solution

intra-mart WebPlatform/ AppFramework

Programming Guide

Script Development Model

Ver 6.1

株式会社 NTTデータ イントラマート

❖ Revision History

Revised Date	Revised Content
2007/07/31	First Edition
2007/08/31	Second Edition ”1.5 2 Web Application Models” has been amended. ”1.6 intra-mart Application Development Guideline” has been amended. ”2.3 Retrieving Data from Database” has been amended.

Contents

intra-mart WebPlatform/
AppFramework

Chapter 1 Introduction

1.1 Starting up and Exiting intra-mart

Since intra-mart is a Web-based development tool, the activation and termination methods are different from those of normal applications. With a Web-based application, the users use a web-browser to “log in” to use the system and “log out” to exit the system.

1.1.1 Login to intra-mart

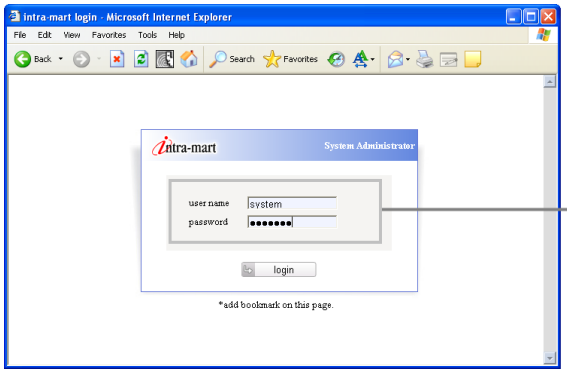
The login URL of intra-mart varies for the System Administrators, Login Group Administrators, and General Users. The following is the login procedure for Login Group Administrator. For the System Administrators and Users, please refer to the “Administrator Guide”.

1 Start up a browser and type in the intra-mart URL as shown below.
The intra-mart’s login screen will be displayed on the screen.

```
intra-mart WebPlatform
(Standalone)              :http://machineaddress/imart/(login group name).manager
(Distributed System)      :URL matched to the registered content of the web server connector
intra-mart AppFramework
        :URL matched to the registered content of intra-mart at the web application server
```

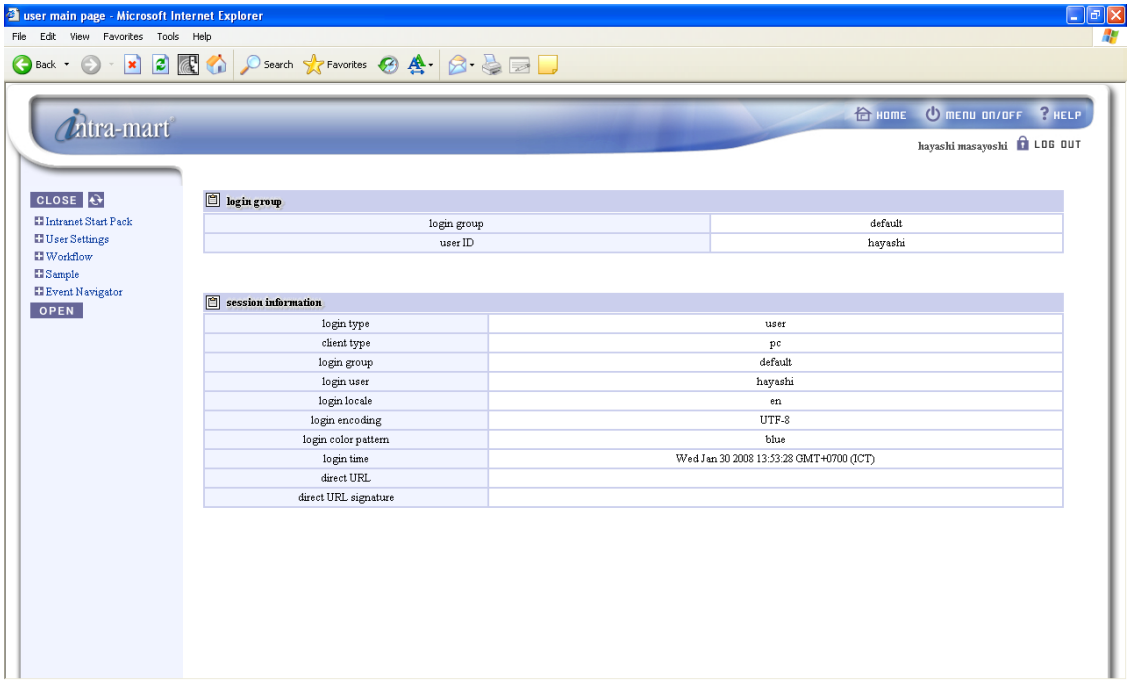
- If intra-mart WebPlatform is operated in stand-alone mode, the port number of URL (port of the web server) can be specified at the time of installation.
- It is recommended to bookmark the URL for convenience.
- There is also “Auto authentication without using Login Screen” (refer to the column on the P.3).

2 Enter user code and password on the login screen, and click on the [Login] button.



<intra-mart’s Login screen>

- The source file of the Login screen and Login default page can be found at the following location.
  %ResourceService%/pages/platform/src/system/security



<intra-mart's default page>

- The main page of the default page (the right frame) is a portal screen. By creating new portlets and registering to the portal function, various kinds of information can be displayed by default.

1.1.2 Logout from intra-mart

Go back to login screen to log out from the intra-mart to terminate the session. Click on the [LOGOUT] button at the top of the menu of the left side of the screen to go back to the login screen.

- If a browser was closed without clicking on the [LOGOUT] button, or the user left the intra-mart screen by going to another page, the user is considered logged onto the intra-mart server until the session time out. Please make sure to click on the menu's [LOGOUT] button.

Column

Auto Authentication without Login Screen

Include the user code and password in the login URL of intra-mart as shown below.
This way, the user can auto authentication without going through the login screen.

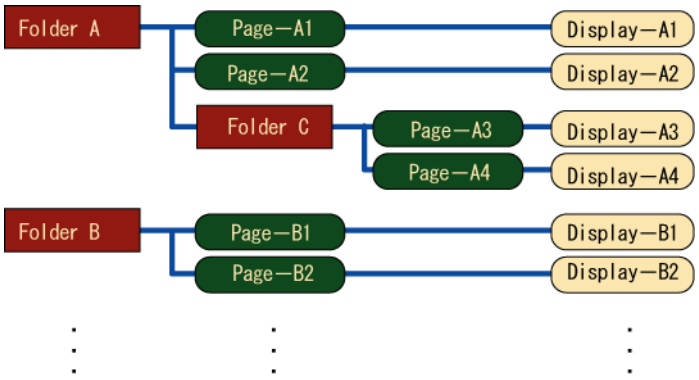
■System Administrator
http:/intra-mart/imart/system.admin?im_user=usercode&im_password=password

■Login Group Administrator
http:/intra-mart/imart/logingroupname.manager?im_user=usercode&im_password=password

■General Users
http://intramart/imart/logingroupname.portal?im_user=usercode&im_password=password

1.2 Menu Structure of intra-mart

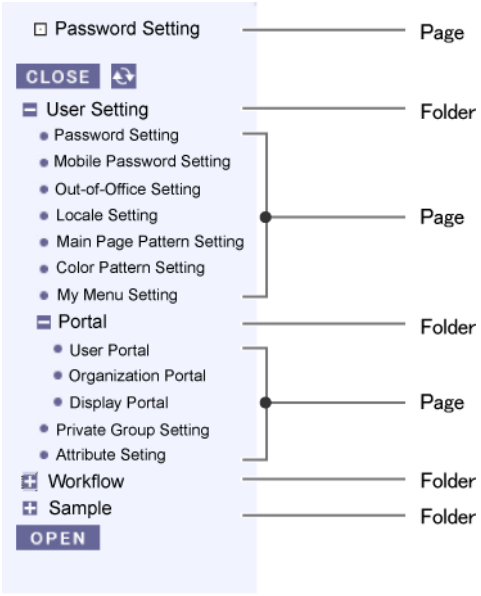
intra-mart's menu display format consists of "Folder" and "Page", and specific page will be displayed when it is selected. The folders can be registered to any tier, and used to consolidate all the pages that are under its scope.
Since the availability of folders and pages are dependent on the access rights of each login user, the actual page view may differ from those shown in this manual.



<Relationship between Folders and Pages>

1.2.1 Displaying Menu

intra-mart's menu is displayed in tree structure on the left frame of the screen. If a "Folder" was clicked, all the "Pages" under the folder will be displayed on the tree. Then, select a "Page" to display its page content.

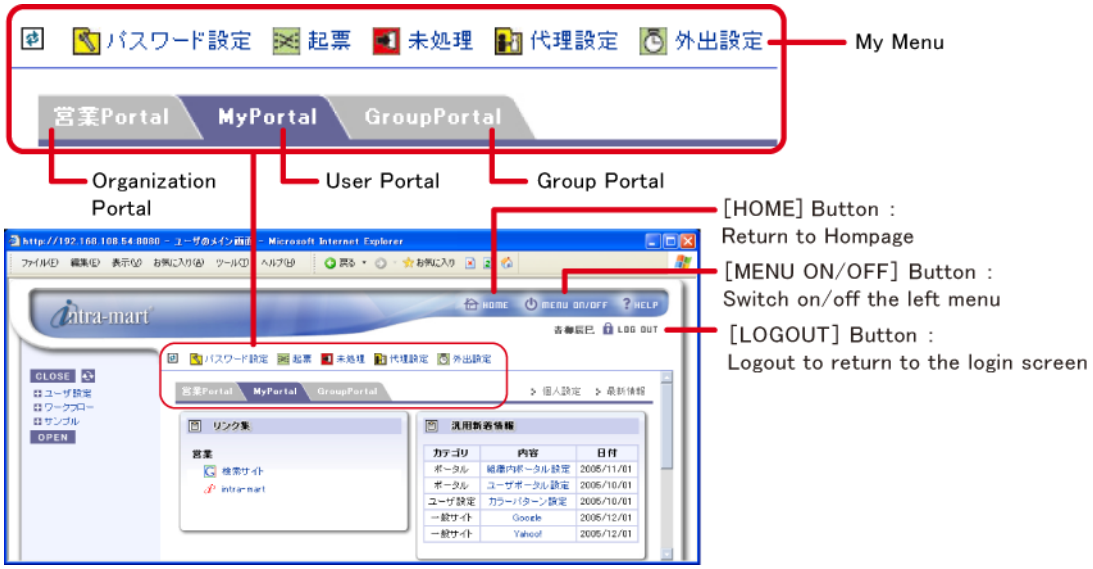


<Example of Menu Display Page>

1.2.2 intra-mart Home Page

Home page is the default page first displayed when you logged onto intra-mart. If the user had navigated to other screens through selection of the menu items, clicking on the [HOME] button in the upper left corner of the screen will bring the user back to the Home page.

The user can switch over to the intra-mart's portal screen from the home page, by using the tab located at the upper-left of the right frame screen.



[HOME] Button	To return to the Home page. The Home page is the default page that is displayed when the user login into intra-mart. He/she can also switch to a portal screen (if there are any) from the Home page.
[LOGOUT] Button	Terminate intra-mart session and return to the Login screen.
[MENU ON/OFF] Button	Turn ON/OFF the menu on the left-hand side. Once the menu is OFF, the working screen becomes more spacious and easier to work on. This function is available only if the user is using the Internet Explorer browser.
[Portal Switching] Tab	You can switch the portal screens using this tab. Please refer to "2.10 Portal Settings and Operations" and "3.4 How to Use Portal" of the "Administrator Guide" on the details of portal screens.
My Menu	You can register your frequently used menu as "My Menu". You can configure the menu in the top of the screen page or the top of the left menu.

1.3 Executing Sample Programs

Following chapters in this manual have been structured in a methodical way so that the reader can experience basic programming using sample programs installed in intra-mart WebPlatform/Framework. Users can display the source codes while reading this manual, and try executing them. There is also an environment available with which user can create the samples from scratch and execute.

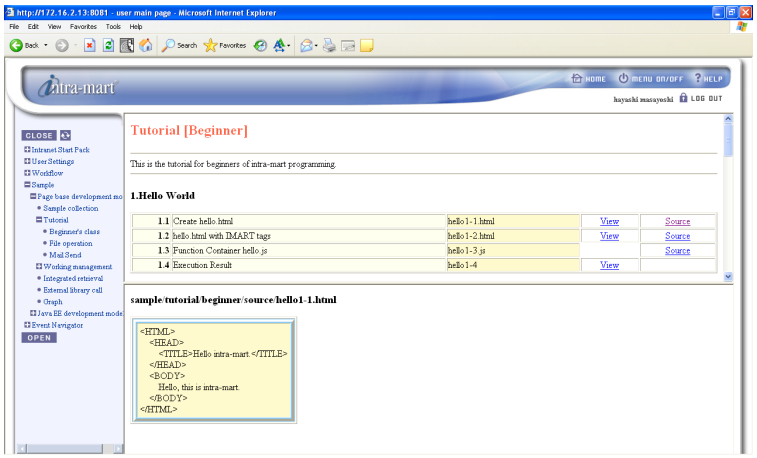
Script Development Model
Sample Program : [Sample]-[Script Development Model]-[Tutorial]-[Beginner]
Source File : %Storage Service%/sample/tutorial/beginner/source/

1.3.1 To View the Source List and Execution Screens

Select “Source” link in [Beginner] page to display the source list described in each section. Select “Execution View” link to see the page actually executed.

1.3.2 To create the Samples listed in this Manual

Once intra-mart is installed, a folder by the name of “sample” will be created automatically. There will be 2 folders in this folder, a folder for “Script development model” and another for “JavaEE Development Model”. Each folder contains template files by the same name as samples listed in the following chapters in this manual. When samples are created using those template files, you can actually execute it by selecting a link at the bottom of the [Beginner] page.
Additionally, all files can be copied into “tutorial/beginner/training” folder for execution since “tutorial/beginner/example” folder contains completed samples.



<Tutorial [Beginner] sample page>

- Other than the samples mentioned in this section, there are sample programs such as the Working management, etc, provided as reference for programming, in the [Sample] folder in the menu.

1.4 Viewing API List

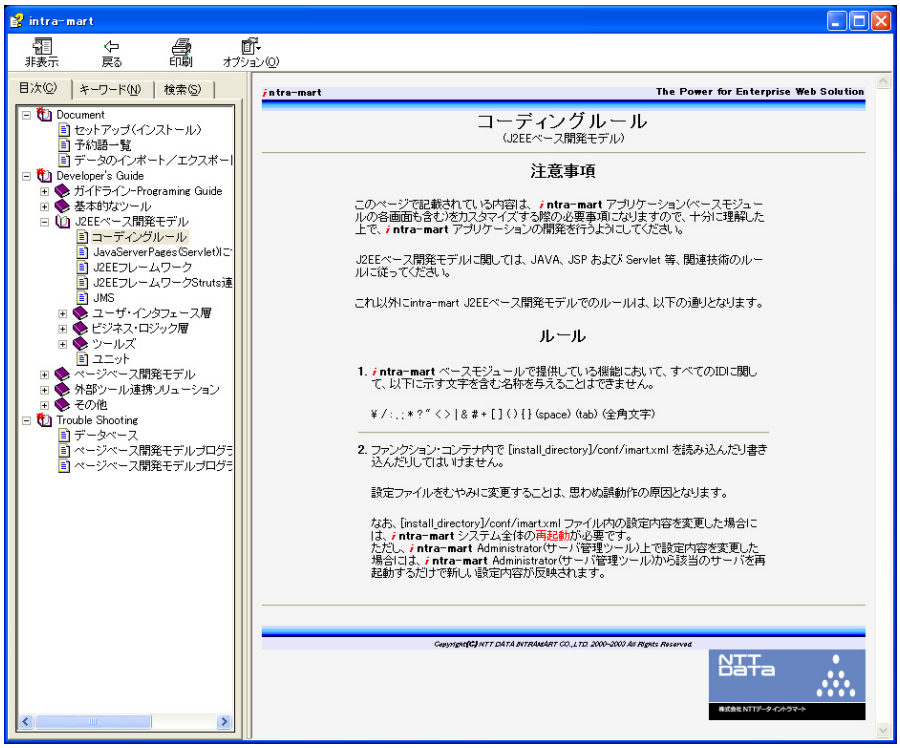
The API List contains the descriptions of the specifications of various functions and APIs available on the intra-mart WebPlatform/Framework. It can be used as reference material during development. The API List comes in both HTML file as well as Windows default Help file formats. You can make full-text search using the [Search] tab in Windows default Help file formats.

1.4.1 API List Storage Location

API List is stored at the following location in the Installation CD.

HTML format : iwp_afw/development/apilist_v61.zip
Windows default Help file format : iwp_afw/development/apilist_v61.chm

The following menu will appear when the user chooses to display the API list in Windows default Help file format.



<API List>

1.4.2 API List Operation

Click on a module name of any category from the [Content] tab on the left-hand side to display the contents of the module. The users can use the [Search] tab to run a search of any object. Currently, the [Keyword] tab is not in use.

1.5 2 Web Application Models

intra-mart comes with 2 development models, the “Script Development Model” and the “JavaEE Development Model”. They have unique characteristics and can be used for different purposes. Web applications developed using two different models can also be mixed together in a single system. Since both development models are able to utilize intra-mart’s default frameworks such as Access Security module and Workflow module, they are highly productive.

1.5.1 Script Development Model

This is a development model to develop Web system using HTML (server side) and JavaScrip for homepage use. As it is easy and simple even for a beginner, staff education/training cost can be considerably reduced by implementing this model. Even for those who do not have sufficient experience in creating web pages, the web operation pages creation process can be mastered in 2 weeks to 1 month time period. Another advantage is that because the development is as easy as creating web pages, it can flexibly cater to complex web system that needs to be updated frequently. Furthermore, depending on the Developer’s technical skill level, he/she can also call Java (Class/EJB), C++ or Stored Procedure without hassle from JavaScript.

- This guidebook has description on Script Development Model.

1.5.2 JavaEE Development Model

JavaEE (JavaEE EnterpriseEdition) is a Java platform advocated by Sun Microsystems Inc for enterprises. It consists of Servlet, JSP (JavaServerPages), and EJB (EnterpriseJavaBeans), and use MVC model (Model-View-Controller) for the system architecture. It is especially suitable for systems that have high transaction concentration.

- Please refer to “Programming Guide: JavaEE Development Model” for details.

1.5.3 Mixing of Applications Developed by 2 Models

intra-mart WebPlatform/AppFramework allows web applications developed by these 2 development models to be used together in one system. For instance, the development of web system with low-budget and short time line can use the Script Model primarily, and then part of the system where intensive transaction is expected can be separated out to utilize JavaEE Model.

.

- In addition to Programming Guides (for both Script and JavaEE), please refer also to the Administrator Guide for the details on program development. On top of these manuals, there is an “im-JavaEE Framework Specifications” in the installation CD.
  Installation CD: iwp_afw/specification/im-javaee_framework-spec_v61.pdf

1.5.4 File Storage Location

The files of intra-mart WebPlatform/AppFramework are stored at the following locations.

1.5.4.1 intra-mart WebPlatform(Resin)

■ Static Contents (e.g. HTML files and image files, etc.)

Under the Installation Directory of the Web Server Connector

(For standalone system, under the directory right under an installation directory of the Server Module)

■ Script Development Model Program

(Presentation Page (.html), Function Container (.js))

Under the Source Directory (Usual - %ResourceService%/pages/src/)

■ JavaEE Development Model Program(JSP)(JSP files(.jsp, .xtp))

Under “%Application Runtime%(For standalone system, server module)/doc/” directory

■ JavaEE Development Model Program(Servlet)(JAVA Class files(.class))

Under “WEB-INF/classes/” within the relevant directory under “%Application Runtime%(For standalone system, server module)/doc/” directory.(or, within the directory set as a class path)

■ Files to be managed centrally by Storage Service

Within “%Storage Service%/storage/” directory

1.5.4.2 intra-mart WebPlatform(JBoss) and intra-mart AppFramework

■ Static Contents(e.g. HTML files and image files, etc.)

Under the “doc/” directory immediately beneath the Installation Directory of the Framework Server

■ Script Development Model Program

(Presentation Page(.html), Function Container(.js))

Under the Source Directory (Usual - %ResourceService%/pages/src/)

■ JavaEE Development Model Program(JSP)(JSP files(.jsp))

Under “%Application Runtime%(For standalone system, server module)/doc/” directory

■ JavaEE Development Model Program(Servlet)(JAVA Class files(.class))

Under “WEB-INF/classes/” within the relevant directory under “%Application Runtime%(For standalone system, server module)/doc/” directory.(or, within the directory set as a class path)

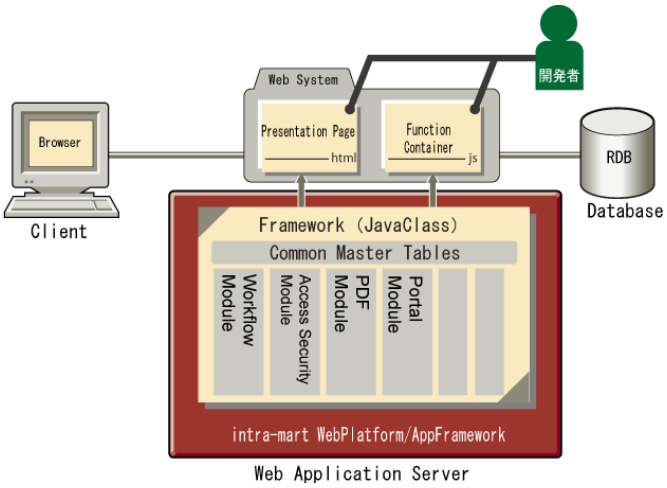
■ Files to be managed centrally by Storage Service

Under “%Storage Service%/storage/” directory

1.6 intra-mart Application Development Guideline

(Script Development Model)

In the process of application development using intra-mart WebPlatform/AppFramework, the Developer needs to create the user interface shown on the browser and the business logic executed on the web server. For Script Development Model, the Developer creates 2 files (Presentation Page in HTML and Function Container in Server Side JavaScript). Higher productivity is achieved by utilizing modules prepared in intra-mart WebPlatform/AppFramework during the development.

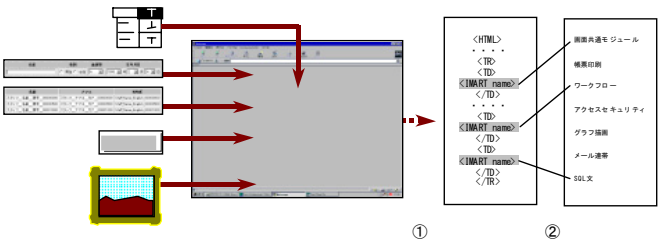


1.6.1 Presentation Page

Presentation Page comes with the ".html" extension and serves as the user interface. Web-based Presentation Pages are created by Developer or End-user using "intra-mart eBuilder".

Pages full of multi-media content such as animations and sounds can be created for browser-based user interface. Furthermore, as Presentation Pages are HTML files, in the course of system development these user interfaces can be extracted and contracted out to the web designers. JavaScript in Function Container can be linked and called by adding <IMART> tags into the HTML files created by web page development tool. <IMART> extension tags for invoking user defined functions can also be added.

Upon registration of the completed HTML page, it can immediately be linked with DB for high speed execution



① Create template for Presentation Pages using commercial web page development tool or "intra-mart "eBuilderW".

② Use intra-mart eBuilder or text editor to insert (program) <IMART> tag that links with JavaScript functions in Function Container or intra-mart WebPlatform objects into HTML source codes that were auto-generated by the web page development tools.

Sample view of Presentation Page is as shown below.

It calls the various modules by using unique extension tag <IMART>.

＜社員マスタからのデータ取得用HTML（一覧表示用）ー項目を拡張＞

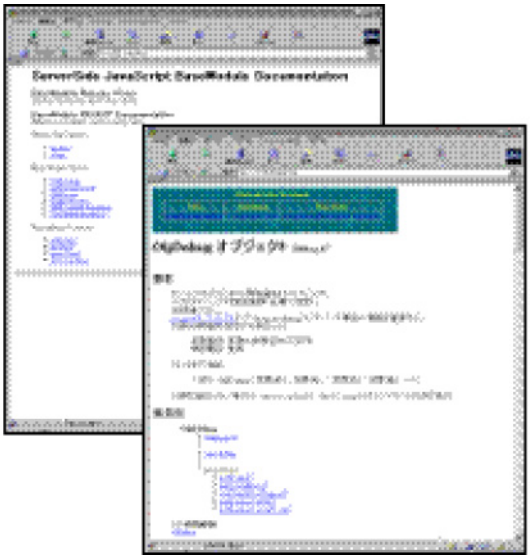
```
 1: <HTML>
 2: <BODY>
 3: <TABLE border="1">
 4: <TR>
 5:     <TD>社員コード</TD>
 6:     <TD>社員名</TD>
※ 7:     <TD>社員名（カナ）</TD>
 8: </TR>
 9: <IMART type="repeat" list=staffList  item="record">
10: <TR>
11:     <TD><IMART type="string" value=record.staff_cd></IMART></TD>
12:     <TD><IMART type="string" value=record.stf_name_kanji></IMART></TD>
※13:    <TD><IMART type="string" value=record.stf_name_kana></IMART></TD>
14: </TR>
15: </IMART>
16:  </TABLE>
17: </BODY>
18: </HTML>
```

Calls each module by using unique extension tag <IMART>

1.6.2 **Function Container**

Function Container comes with the ".js" extension and serves as the business logic that operates on APP server within multi-hierarchical architecture. Since Function Container and Presentation Page works as a set, the same name should be used for both file labels. Developer programs the JavaScript which is called from Presentation Page in the Function Container. Specifically, the Developer selects the necessary objects and functions from framework functions provided by intra-mart WebPlatform/AppFramework, and use the "intra-mart eBuilder" (programmed in JavaScript) to create business logic which operates with those selected objects and functions on the server side. (By using "intra-mart eBuilder", the productivity will be improved.) The SQL languages for the database should also be programmed into the Function Container. Since actual connections with RDB and issuance of SQL are executed from intra-mart WebPlatform/AppFramework, Developer neither needs to concern about the detailed session management nor transaction management. The created business logic is called and executed from the <IMART> tag in the Presentation Pages. Details of framework functions are described in "intra-mart API List".

As these programming can be done in HTML and JavaScript, developers are hence able to build a full web system that interfaces with DB as they create a homepage.



- Page Common Module
- Ledger Sheet Print
- Workflow
- Access Security
- Graph Drawing
- Mail Linkage
- SQL Sentence

1.7 Script Development

From chapter 2 onward in this manual the topics listed below are covered.
Please refer to the applicable pages for the development using Script Development Model.

Chapter 2 Basic of Script Programming

1 Creating “Hello World”
2 Data Sharing between Pages (Session Management)
3 Retrieving Data from Database
4 Listing of Retrieved Data
5 Registering/Updating/Deleting Data

Chapter 3 Using Various Component Group(im-BizAPI)

1 How to Use the Storage Service
2 Sending E-Mails
3 Example on Usage of Extension <IMART> tag Function
4 Registration and Usage of User Defined Function
5 Linkages with JavaClass
6 Linkages with EJB
7 Calling External Processes
8 Handling XML Data
9 How to Use E4X
10 Debugging Process
11 Unit Test Environment (im-JsUnit)
12 JavaScript Compiler Function
13 Compiler function of JavaScript
14 Linkages with im-JavaEE Framework
15 Sample Applications
16 Embedding & Operating the Modules
17 Embedding & Operating the Units
18 Embedding & Operating the Extension Modules
19 Other Functions

Appendix

1 Message Settings
2 Reserved Word List
3 Restrictions

intra-mart WebPlatform/
AppFramework

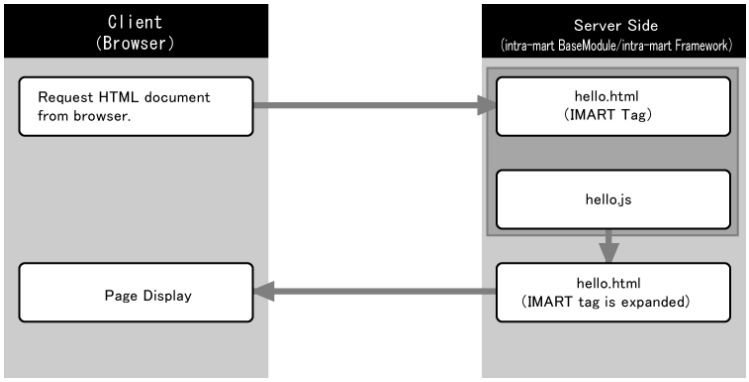
Chapter 2 Basic of Programming in Script Development Model

2.1 Creating “Hello World”

This section provides better understanding of actual works on creating presentation page and function container by creating simple applications using intra-mart Script Development Model.

- All samples introduced in this chapter are registered in [Tutorial] folder in [Samples] of intra-mart. Following sections in this manual corresponds to the tutorials indicated below.
  Chapter 2 “1 Creating Hello World” ～ “5 Registering/Updating/Deleting Data” → Beginner Pages
  Chapter 3 “1 How to Use the Storage Service” → File Operation Pages
  Chapter 3 “2 Sending Mails” → Sending Mails Pages

At the beginning, create a simple application to show a greeting message on the browser by linking up with intra-mart WebPlatform. The message “Hello, this is intra-mart” shall appear when “hello.html”, a Presentation Page created on the server side, is started from the browser.

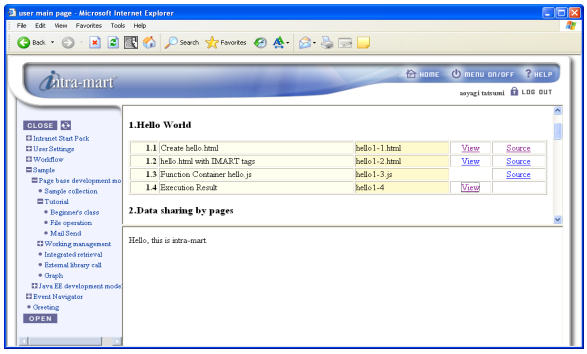


2.1.1 Creating the Base - Presentation Page (.html)

Create the Presentation Page to be displayed on the browser in HTML format. You can create the HTML file using intra-mart “eBuilder” or any web page creation tool/editor available in the market.
Here we start with creating a simple static HTML that displays only text as shown below. Name the file “hello.html”, and save in Sample folder under imart folder.

<Created “hello.html” file>

```
1:    <HTML>
2:    <BODY>
3:    Hello, this is intra-mart.
4:    </BODY>
5:    </HTML>
```



<Execution screen of “hello.html”>

On the HTML file, embed <IMART> tag in the location where it should be linked with function container and complete the presentation page. In this example, the word “intra-mart” is tagged with <IMART> and configured to be replaced by character string (nameValue) specified in the Function Container.

<Modified “hello.html” file with IMART tag embedded>

```
1:    <HTML>
2:    <BODY>
3:    Hello, this is <IMART type="string" value=nameValue></IMART>.
4:    </BODY>
5:    </HTML>
```

Column

<IMART type="string">

In this scenario, “string” is specified by the Type command for <IMART> tag.
“String” is an attribute for replacing the variable specified by the Value command to a value in the Function Container. There are many attributes in intra-mart WebPlatform that can be specified with the Type command for linking Presentation Page and Function Container, such as ”link”, “repeat”, “form”, “input”, “select”, etc. For the details of <IMART> tags in intra-mart, please refer to “API List” that comes with intra-mart WebPlatform.

2.1.2 Creating Function Container (.js)

To create the function container that works with the created presentation page.
In the function container, create the init function (initializing function). In this example, configure the character string “intra-mart” as a property with the name “nameValue”. Name the file “hello.js” to link up with hello.html, and place it in the same folder as the presentation page.

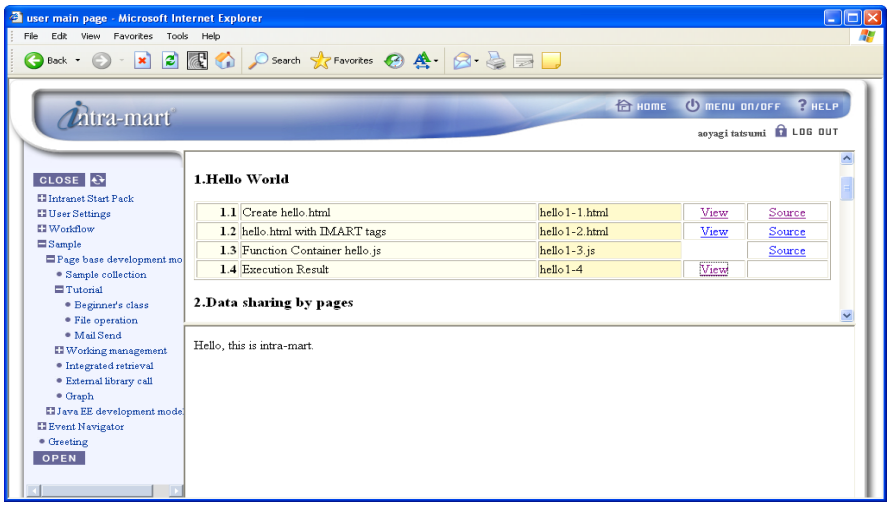
<Created Function Container(hello.js)>

```
1:    // DECLARE VALUE TO PASS TO HTML
2:    var nameValue;
3:
4:    // Define init Function
5:    function init()
6:    {
7:          nameValue = "intra-mart";                    // Configure Value to pass to HTML
8:    }
```

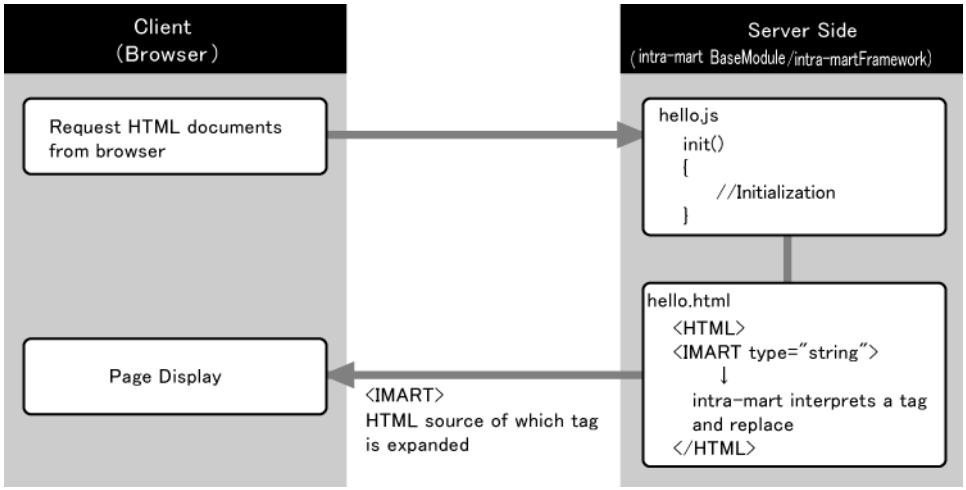
- The function “init()” in the Function Container is a fixed-function that intra-mart WebPlatform automatically interpret once initialized.

2.1.3 Executing the Application Program

When the application program created with the Presentation Page and Function Container is executed, the greeting text “Hello, this is intra-mart.” will be displayed.



<Execution result screen>

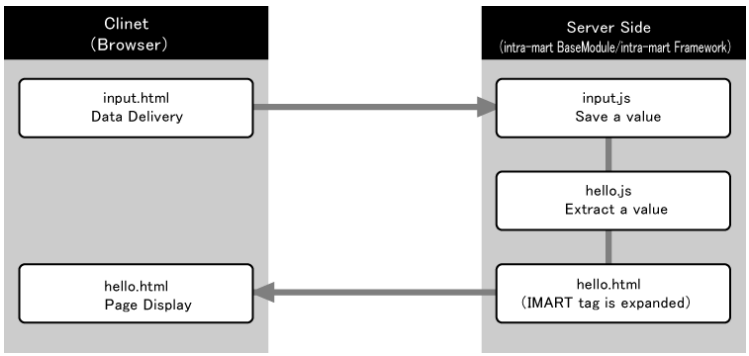


- To run the created application in any directory, user needs to login as the Login Group Administrator and register the page in [Menu Settings] in [Login Group Setting] – [Menu management] first. For the registration of application, please refer to “2.7 Application Registration” in Administrator Guide.

2.2 Data Sharing between Pages

(Session Managementl)

With intra-mart WebPlatform, data can be shared between multiple pages (session management) easily. An application will be created in this exercise, which will first enter data from presentation page, save the data on the server side via function container, and finally display the data on another page.
First, create a new page where name should be entered (input.html), and display the data retrieved here on the “hello.html” page created in the last section.



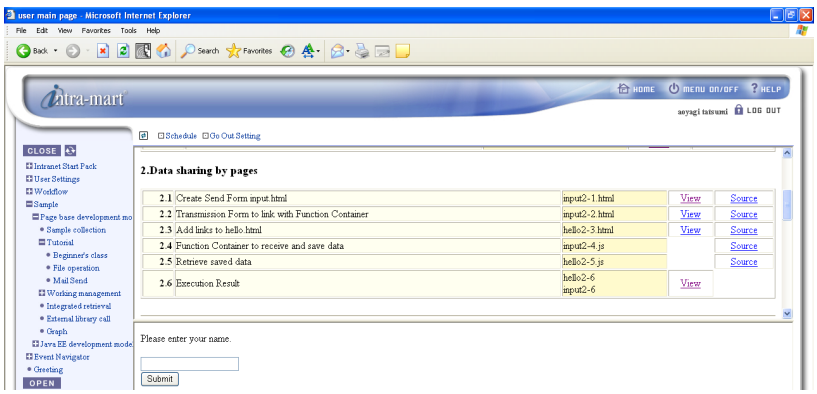
2.2.1 Creating the Base - Presentation Page (.html)

Create the base-presentation page. The extension is restricted to ".html” only.

2.2.1.1 Preparing Sender-side HTML File (input.html)

For the first step, create HTML page where data can be entered using <FORM> tag, the same way as creating normal web pages. Name the file as “input.html”.

```
<Send Data Form input.html>
1:    <HTML>
2:    <BODY>
3:    Enter name. <BR>
4:    <FORM method="POST">
5:          <INPUT type="text" name="yourname"><BR>
6:          <INPUT type="submit" value="Send">
7:    </FORM>
8:    </BODY>
9:    </HTML>
```



< “input.html” execution screen>

You need to enable the values that have been keyed in the form to be retrieved from the function container side, by invoking a function specified on the function container when the send button of the form is clicked.
For this purpose, replace the <FORM> tag and <INPUT> tag that were needed for linkage to the function container with <IMART> tag.
Save the completed “input.html” file.

```
<Sending Form HTML for linkage with Function Container>
1:    <HTML>
2:    <BODY>
3:    Enter name. <BR>
4:    <IMART type="form" method="POST" action="inputName" page="sample/user1/hello">
5:          <IMART type="input" style="text" name="yourname"></IMART><BR>
6:          <IMART type="submit" value="Send"></IMART>
7:    </IMART>
8:    </BODY>
9:    </HTML>
```

2.2.1.2 Preparing Display-side HTML file (hello.html)

Although the created HTML source can be directly displayed on the browser, this section explains how to display “input.html” by linking it from “hello.html” which was created earlier.
Add the text line shown below into “hello.html”.

```
<Add to hello.html>
      <IMART type="link" page="saved folder name/input">enter name</IMART>
```

Column

<IMART type="link">

Using this tag enables program to run while maintaining the intra-mart session. For details, please refer to “API List”.

Path to the Target HTML Source

It uses the relative path from the source directory under Resource Service in intra-mart WebPlatform ([%Resource Service%/pages/src/] by default). Note that its extension is not specified.

"./input" or "saved folder/input.html" statements will cause error.

2.2.2 Creating Function Container (.js)

Next, create Function Container file using JavaScript.

2.2.2.1 Creating Recipient-side JavaScript file (input.js)

<JavaScript to receive and save data>

```
1:    // Define inputName function
2:    function inputName(request)
3:    {
4:          // Save the retrieved Value in Client Object
5:          Client.set( "nameValue", request.yourname );
6:    }
```

2.2.2.2 Creating JavaScript file to Call Saved Data (Hello.js)

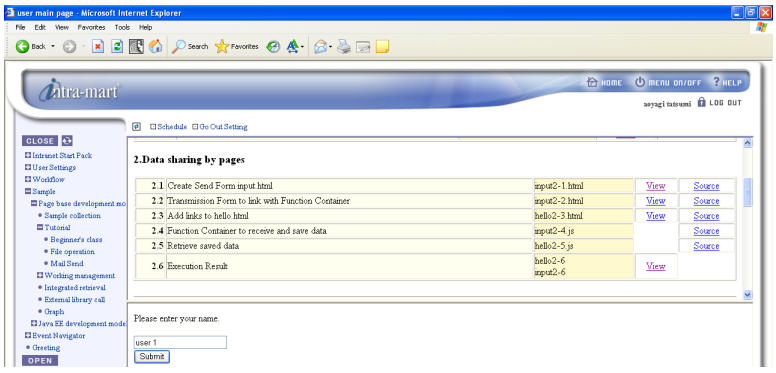
Rewrite the code on the 7th line in previously created “hello.js” as shown below, in order to have it retrieving the input value.

<Example to extract saved data>

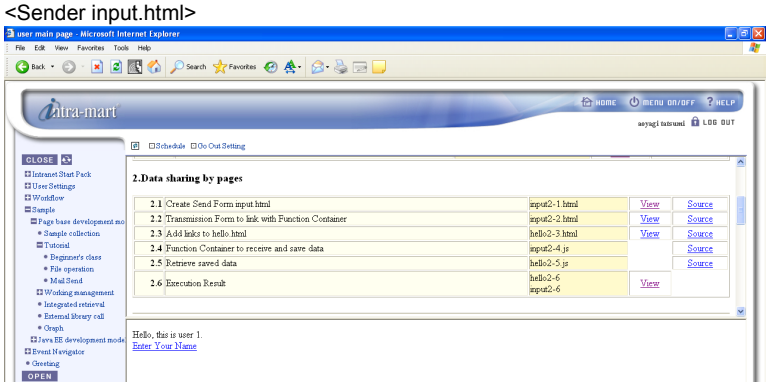
```
7:    nameValue = "intra-mart";                    // Set Value to Pass to HTML
                          ↓
7:    nameValue = Client.get( "nameValue" );       // Set Value to Pass to HTML
```

2.2.3 Executing Application Program

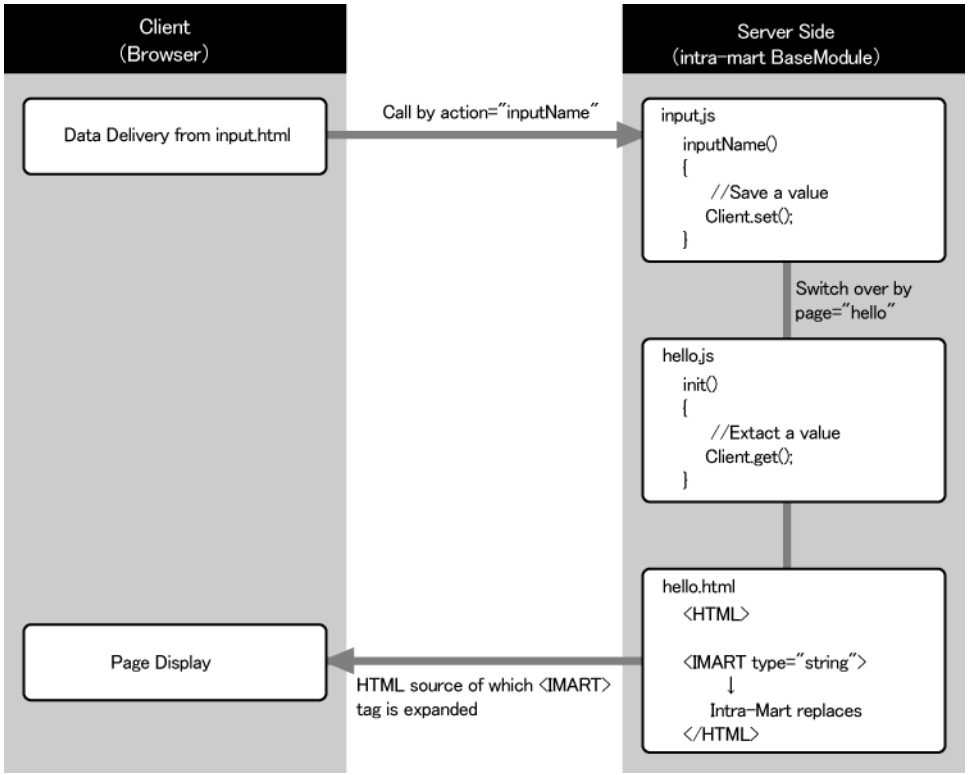
When the application made up of Presentation Page and Function Container is executed, the data “user 1” keyed-in and sent from sender side “input.html” will be displayed on “hello.html” at the recipient side.



<Sender input.html>



<Recipient hello.html>



Column

<IMART type="form">

Here are some further details on <IMART type="form">, which appeared on the 4th line in input.html.

<IMART type="form" method="POST" action="inputName"page="sample/user1/hello">

This <IMART type="form"> is a method specified to supply the <FORM> tag used to pass the data to intra-mart function container.

The action attribute specifies the name of the user defined function that is called when form data is sent to the server. Check in input.js and define the function by the same name on the server side function container.

The page attribute can specifies the page to be displayed after the completion of form data transmission and server side processing. When this is omitted, the current page will be refreshed.

Please keep in mind that here the specifying method for link destination will still be the relative path from the source directory managed by Resource Service (default % Resource Service%/pages/src).

Column

Request Object

Once Function Container receives the data, it can view the data of the Presentation Page using request object. Request Object can be received as an argument of function (init() and functions that are specified by <IMART> tag), which is automatically called from intra-mart WebPlatform/ Framework.

Client.set("nameValue", request.yourname);

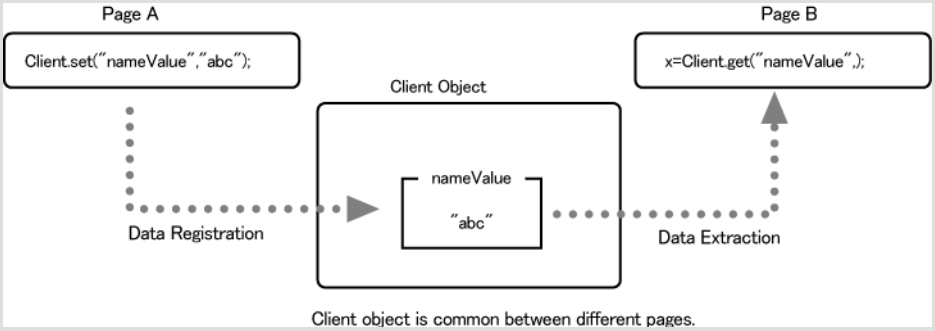
When the data is sent from presentation page, inputName function (in input.js) will be called.

At the Function Container side, the received data can be extracted easily by specifying “yourname” as the property name of request object (in “input.html”, the term ‘yourname’ is attached to the text input field).

Column

Session Management by Client Object

In order to save data on server side even after moving to another page, the “set” method of the Client Object is used.
Client.set method is a method to save data while Client’s web browser is connected to the server application created with intra-mart. Stored data can be named, and storage of multiple sets of data is possible.
In below example, data is named as “nameValue” for storage.



There are numerous other convenient methods and objects defined in the object of intra-mart WebPlatform/AppFramework. Application Developers can build high quality applications within a short period by using these methods and modules.

Column

Holding Time Limit of each Client Information

(Session Timeout Value)

intra-mart maintains the information of each client, such as the above-mentioned session management information and access security information, at the Http session for a fixed amount of time. The default time is set as 10 minutes, therefore re-login will be required if a client did not access for more than 10 minutes. You can change this default time setting using the “conf/http.xml” (basic configuration file). Please refer to “Setting Guide” on how to edit the “conf/http.xml” file.

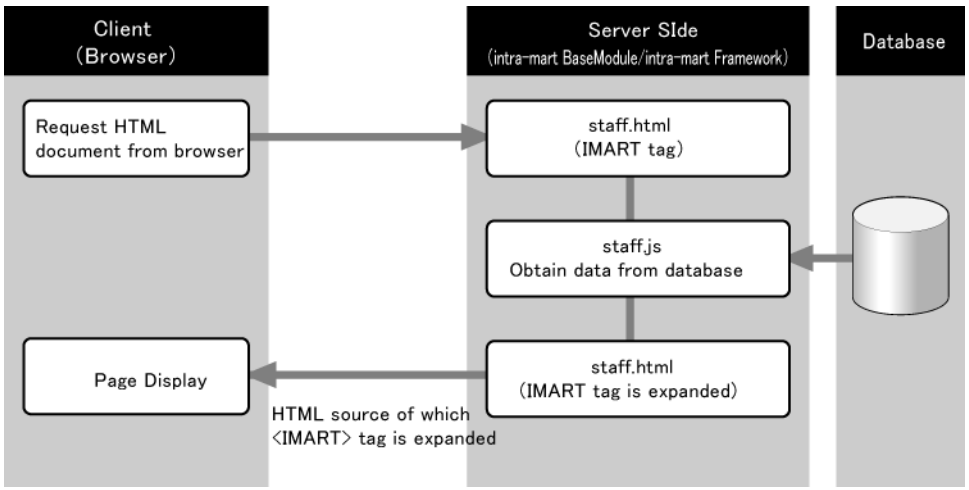
2.3 Retrieving Data from Database

In this section, we shall look at how to retrieve data from actual database using objects and methods made available by intra-mart and intended for database access.
Create dynamically-generated pages on server side, according to the instructions listed in the previous section.

Employee Master (m_sample_stf)				
Sequence Name	**Contents**	**Attribute**	**Data Type**	**Size (bytes)**
staff_cd	Employee Code	Main Key	Text Type	20
stf_name_kanji	Employee (Kanji)		Text Type	50
stf_name_kana	Employee (Kana)		Text Type	50
stf_name_eng	Employee (English)		Text Type	50

<Database to be used>

- You can create this Employee Master (m_sample_stf) used as the sample here from [License] in System Administrator screen after installing intra-mart WebPlatform.

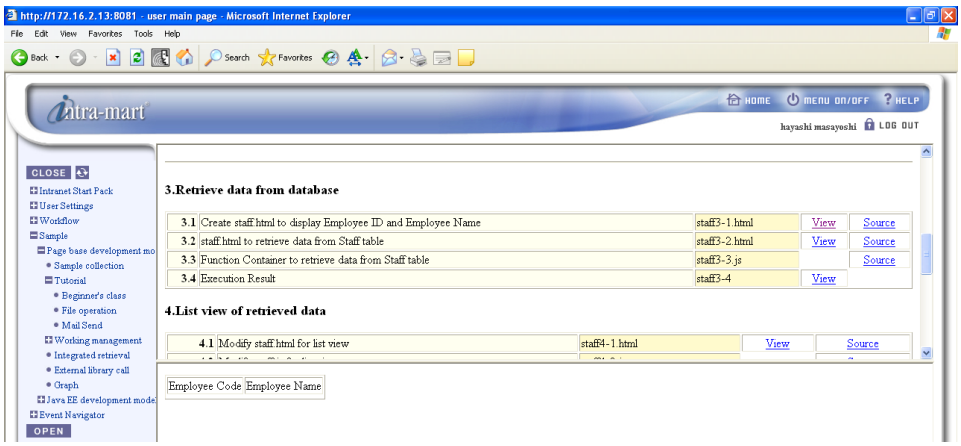


2.3.1 Creating the Base - Presentation Page (.html)

The following HTML file is an example of standard HTML source to display employee code and employee name. It is recommended to use visual tools for HTML design such as Netscape Composer, Microsoft FrontPage or intra-mart "eBuilder" to create the preliminary template for Presentation Pages due to the ease in creating chart/table with these tools.

<"staff.html" to Display Employee Code and Employee Name>

```
1:    <HTML>
2:    <BODY>
3:    <TABLE border="1">
4:    <TR>
5:          <TD> Employee Code </TD>
6:          <TD> Employee Name </TD>
7:    </TR>
8:    <TR>
9:          <TD></TD>
10:         <TD></TD>
11:   </TR>
12:   </TABLE>
13:   </BODY>
14:   </HTML>
```



<Execution result for staff.html file>

Below is an example of "staff.html" modified to reflect data from Function Container.

<"staff.html" to Retrieve Data from Employee Master>

```
1:    <HTML>
2:    <BODY>
3:    <TABLE border="1">
4:    <TR>
5:          <TD> Employee Code </TD>
6:          <TD> Employee Name </TD>
7:    </TR>
8:    <TR>
9:          <TD><IMART type="string" value=staff_code></IMART></TD>
10:         <TD><IMART type="string" value=staff_name></IMART></TD>
11:   </TR>
12:   </TABLE>
13:   </BODY>
14:   </HTML>
```

2.3.2 Creating Function Container (.js)

Create the init function for the function container. Name the file “staff.js”.
This staff.js carries out the following 2 actions.

①┌ Retrieve data from Employee Master File in the Database
②┌ Pass the retrieved data to HTML.

<”staff.js” to Retrieve Data from Employee Master>

```
1:      // Declare Value to pass to HTML
2:              var staff_code;
3:              var staff_name;
4:
5:      // Define init Function
6:      function init(request)
7:      {
8:              var objData    = false;       // For Storing Data Retrieved from Database
9:
10:             // Retrieve All Employee Data from Database
11:             objData = DatabaseManager.select("SELECT * FROM m_sample_stf");
12:
13:             // Configure Value to Pass to HTML from 1st Record
14:             staff_code = objData.data[0].staff_cd;
15:             staff_name = objData.data[0].stf_name_kanji;
16:
17:     }
```

- “staff_cd” and “stf_name_kanji” are column names of database.

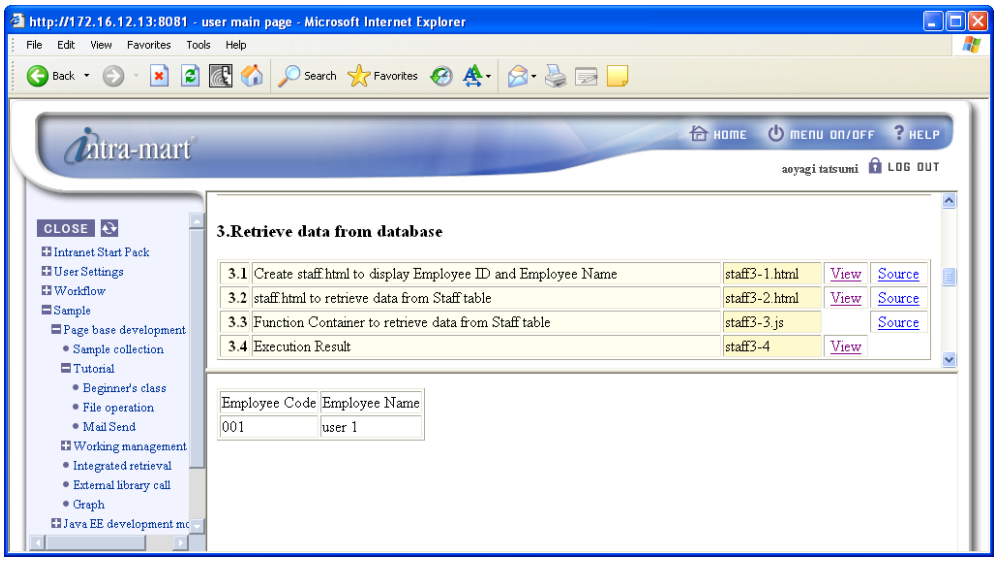
Column

DatabaseManager Object - 1

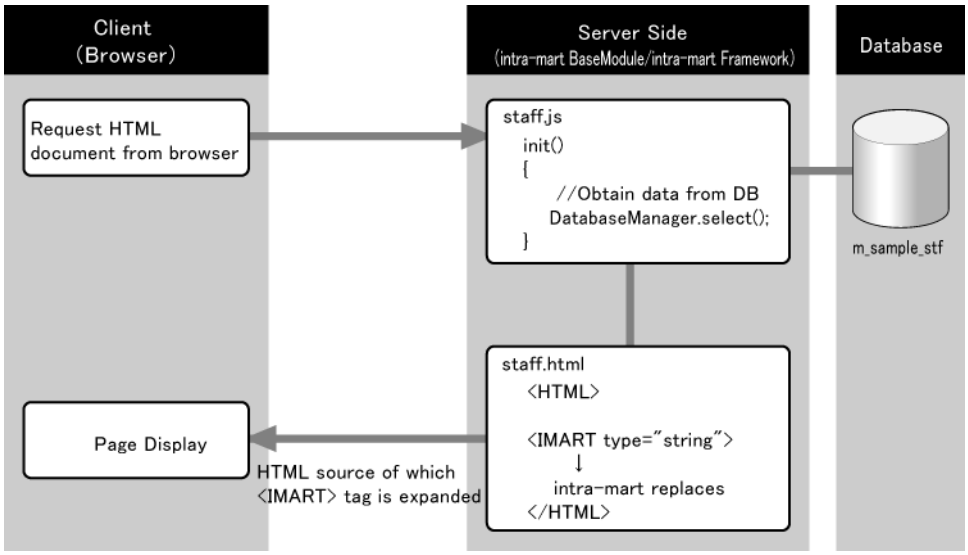
Accessing database is made easy with DatabaseManager Objects.
In this example, the “select” method in DatabaseManager is called. Data can be retrieved from database by coding SELECT statement in SQL format in the parameter as shown here.
Retrieved data will be returned as object. The record retrieved from database will be saved as a column property of object returned by the Select method, with the name “data”. Other than this object returned by the Select method, there are others like “countRow” in which the number of retrieved records will be stored.
As for the details on objects and methods available in intra-mart WebPlatform, please refer to “API List” that comes with intra-mart WebPlatform.

2.3.3 Executing Application Program

Below is the result of application execution.



<Execution Result>



- Change "data[0]" in the 14th and 15th lines of the created "staff.js" to "data[1]" and "data[2]" to display 2nd and 3rd records.

2.4 Listing of Retrieved Data

This section describes how to improve the page created in the previous section for displaying the employee name (staff), with modifying the source code to display a listing of employee names.

2.4.1 Additions to the Base - Presentation Page (.html)

Use <IMART type="repeat"> tag to display the listing. Repeat is used to expand HTML by executing the codes nested between the tags for the number of times specified. Furthermore, it can operate simultaneously with matrixes created in the JavaScript side to display the data in sequence.

<HTML to Retrieve Data from Employee Master (for listing display)>

```
1:    <HTML>
2:    <BODY>
3:    <TABLE border="1">
4:    <TR>
5:          <TD> Employee Code </TD>
6:          <TD> Employee Name </TD>
7:    </TR>
8:    <IMART type="repeat" list=staffList   item="record">
9:    <TR>
10:         <TD><IMART type="string" value=record.staff_cd></IMART></TD>
11:         <TD><IMART type="string" value=record.stf_name_kanji></IMART></TD>
12:   </TR>
13:   </IMART>
14:    </TABLE>
15:   </BODY>
16:   </HTML>
```

2.4.2 Creating Function Container (.js)

Below is to display the JavaScript file.

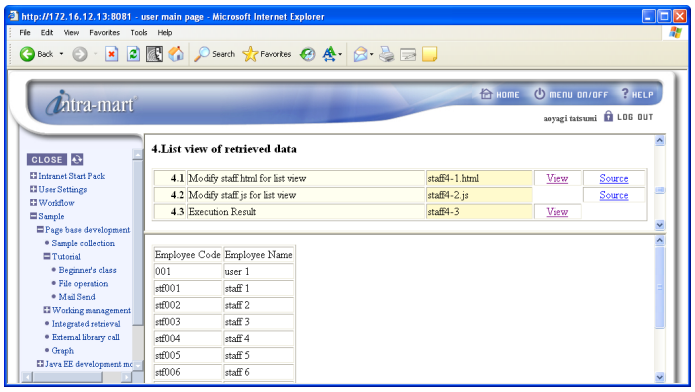
<JavaScript to Retrieve Data from Employee Master (for Display Listing)>

```
1:    // Declare Value to Pass to HTML
2:    var staffList;
3:
4:    // Define init Function
5:    function init()
6:    {
7:          var objData   = false;      // For Storing Data Retrieved from Database
8:
9:          // Retrieve All Employee Data from Database
10:         objData = DatabaseManager.select( "SELECT * FROM m_sample_stf" );
11:
12:         // Configure Data List to Pass to HTML
13:         staffList = objData.data;
14:
15:   }
```

"m_sample_stf" is the name of Employee Master File.

2.4.3 Executing Application

Below is a result view of executing this application.



<Execution Result>

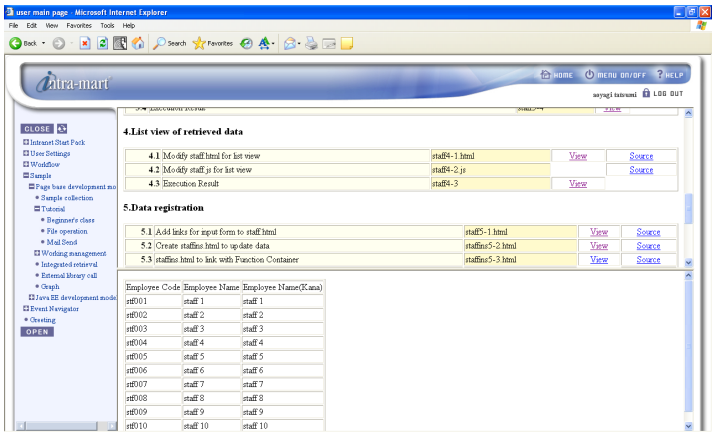
2.4.4 Item Expansion

Follow the instructions to add “Employee Name (Kana)” to the application created. Modification is required only for the presentation page (staff.html) but not for the function container.

In the list below, modification is made to those lines indicated with “←”.

<HTML to Retrieve Data from Employee Master (for Display Listing)— Item Expansion>

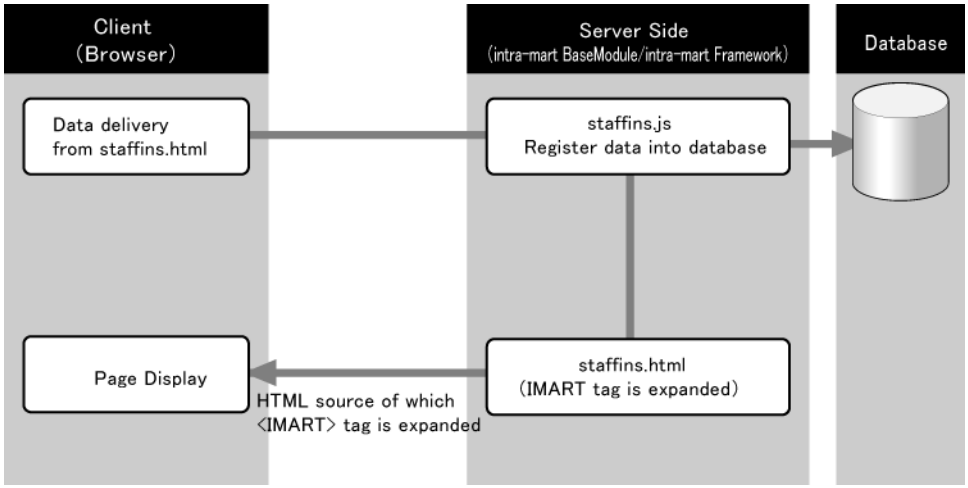
```
1:    <HTML>
2:    <BODY>
3:    <TABLE border="1">
4:    <TR>
5:          <TD>Employee Code</TD>
6:          <TD>Employee Name</TD>
7:          <TD>Employee Name(Kana)</TD>                          ← Modified line
8:    </TR>
9:    <IMART type="repeat" list=staffList  item="record">
10:   <TR>
11:       <TD><IMART type="string" value=record.staff_cd></IMART></TD>
12:       <TD><IMART type="string" value=record.stf_name_kanji></IMART></TD>
13:       <TD><IMART type="string" value=record.stf_name_kana></IMART></TD>   ← Modified line
14:   </TR>
15:   </IMART>
16:    </TABLE>
17:   </BODY>
18:   </HTML>
```



<Execution Result>

2.5 Registering/Updating/Deleting Data

Since we have created the procedures to browsed data, next create a page to do additional registration of data. Name the page “staffins”, and use the <IMART type="link"> to link from the staff.html file created in the previous section.



2.5.1 Creating the Base - Presentation Page (.html)

First, code a general HTML for registring the employee data. Use <FORM> tag to carry out the data transmission. Refer to the following list, and confirm that the information is displayed correctly on the browser.

<HTML to Display Data Update Form>

```
1:    <HTML>
2:    <BODY>
3:    Enter data, and click Register. <BR>
4:    <FORM method="POST">
5:          Employee Code
6:          <INPUT type="text" name="staff_code"><BR>
7:          Employee Name
8:          <INPUT type="text" name="staff_name"><BR>
9:          <BR>
10:         <INPUT type="submit" value="register">
11:   </FORM>
12:   <BR>
13:   <IMART type="link" page="sample/user1/staff">back</IMART>
14:   </BODY>
15:   </HTML>
```

Confirm that the information is correctly displayed on the browser, and then change some of the tags to <IMART> tags in order to send data to intra-mart Function Container.

<”staffins.html” for the linkage with Function Container>

```
1:    <HTML>
2:    <BODY>
3:    Enter data, and click Register. <BR>
```

```
4:    <IMART type="form" method="POST" action="insertStaffName">
5:          Employee Code
6:          <IMART type="input" style="text" name="staff_code"></IMART><BR>
7:          Employee Name
8:          <IMART type="input" style="text" name="staff_name"></IMART><BR>
9:          <BR>
10:         <IMART type="submit" value="Register"></IMART>
11:   </IMART>
12:   <BR>
13:   <IMART type="link" page="sample/user1/staff">back</IMART>
14:   </BODY>
15:   </HTML>
```

2.5.2 Creating Function Container (.js)

Coding the JavaScript to reflect the data received from Client's browser into the database in the Function Container on the server. File name is "staffins.js".

In this section, following two actions are carried out.

① Check the Request Data from Client
② Add into Database

Adding record to the database is achieved by invoking the following 3 methods defined in DatabaseManager Object.

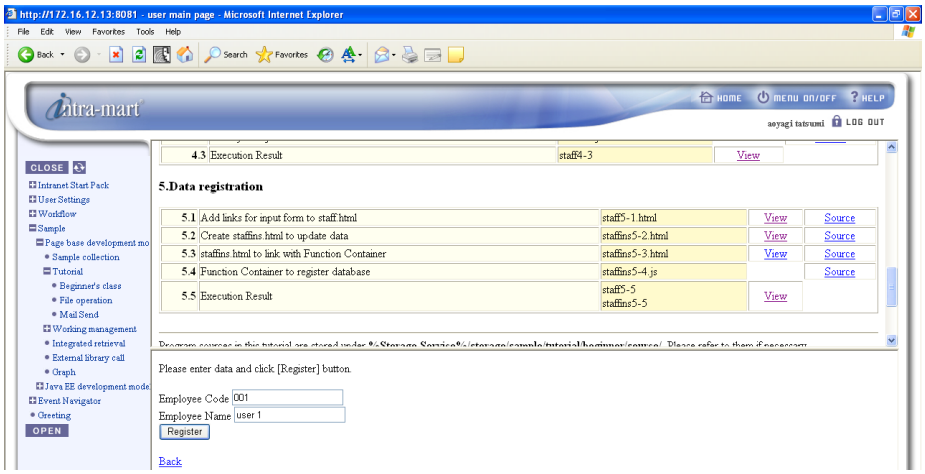
beginTransaction Method	Declare transaction start
Insert Method	Insert data
commit/rollback Method	Commit/roll back the inserted data

<JavaScript to Register Data to Database>

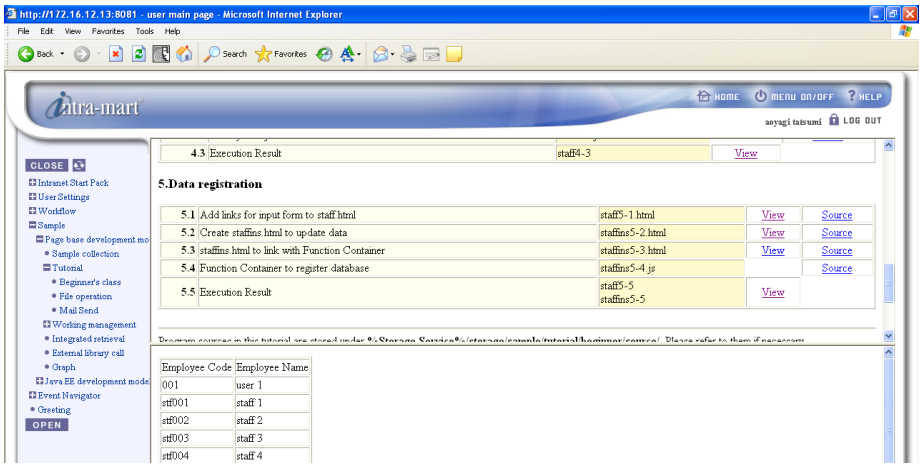
```
1:  function insertStaffName(request)
2:  {
3:      var objRecord = new Object();   // Create Record Registration Object
4:      var result;           // DB Access Result
5:
6:      // Create Record
7:      objRecord.staff_cd          = request.staff_code;
8:      objRecord.stf_name_kanji = request.staff_name;
9:      objRecord.stf_name_kana   = request.staff_name_kana;
10:     objRecord.stf_name_eng    = request.staff_name_eng;
11:
12:     // Start DB Processing
13:     DatabaseManager.beginTransaction();
14:
15:     // Insert Record
16:     result = DatabaseManager.insert("m_sample_stf", objRecord);
17:
18:     // Error Check
19:     if (! result.error)
20:     {
21:         DatabaseManager.commit();              // Commit since successful
22:     }
23:     else
24:     {
25:         DatabaseManager.rollback();            // Rollback since fail
26:     }
27: }
```

2.5.3 Executing Application

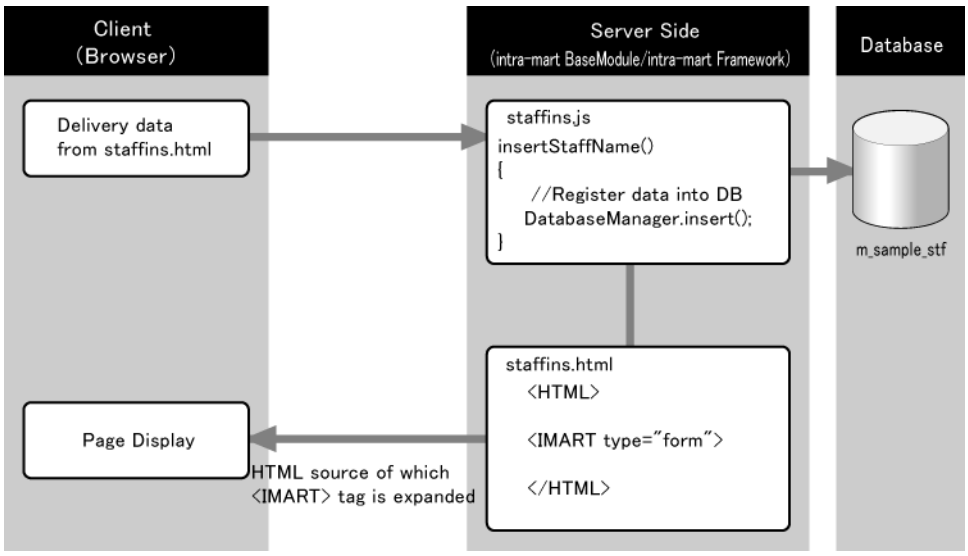
Execution result is as shown below.



<Execution screen of staffins.html>



<Execution Result>

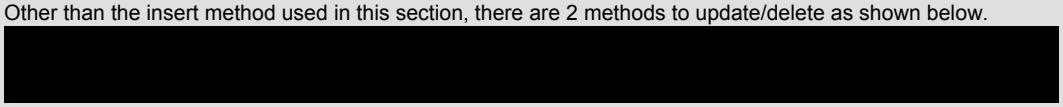


- The “staffins.html” displayed initially is a standard HTML file that has been tag-interpreted and sent by intra-mart WebPlatform.

Column

DatabaseManager Object - 2

Other than the insert method used in this section, there are 2 methods to update/delete as shown below.



The 'Where Statement Conditions' used for defining the parameter is coded with conditional statement specified by SQL language for the Where Statement.

For example, to "delete the Employee Code [stf0001]", specify as follows:

```
DatabaseManager.remove( "m_sample_stf", "staff_cd = 'stf0001'" );
```

In reality, as most of the time the employee code specified in the presentation page is referred to, it should be specified as follows.

```
var strWhere = "staff_cd = '" + request.staff_code + "'";
DatabaseManager.remove( "m_sample_stf", strWhere );
```

This object also has option to access multiple databases simultaneously. For details, please refer to the section regarding this object in "API List".

intra-mart WebPlatform/
AppFramework

Chapter 3 Using Various Component Group (im-BizAPI)

3.1 How to Use the Storage Service

With the use of Storage Service, intra-mart can share files easily even if clustered system architecture is developed. This chapter shows how to create pages that can save files uploaded from a browser in the Storage Service or to download files that has been saved by the Storage Service.

3.1.1 File Upload

First of all, this section creates a form that uploads and save files into Storage Service.

<Data Sending Form “filer.html”>

```
1: <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 //EN">
2:
3:   <HTML>
4:       <HEAD>
5:           <TITLE>File Center</TITLE>
6:       </HEAD>
7:       <BODY bgcolor="WhiteSmoke">
8:           <H2>Upload</H2>
9:           <IMART type="form" action="action_upload" method="POST" enctype="multipart/form-data">
10:              <INPUT type="file" name="local_file">
11:              <INPUT type="submit" value=" send ">
12:          </IMART>
13:      </BODY>
14: </HTML>
15:
16: <!-- End of File -->
```

Code the function container for storing the received file data in storage Service.
Since Function Container receives the file data in binary format, save the file as binary data for Storage Service.

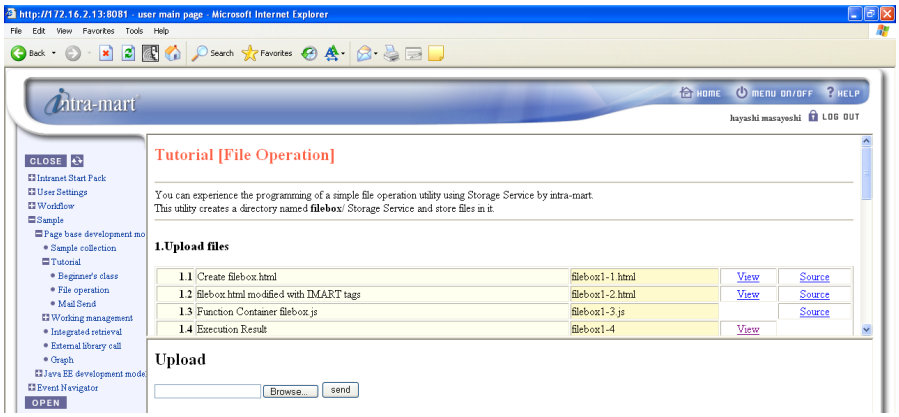
<Function Container to receive files “filer.js”>

```
1:   // Page Initialization
2:   function init(request){
3:       var root = new VirtualFile("filebox");
4:
5:       // Check if storage directory exist
6:       if(! root.isDirectory()){
7:           root.makeDirectories(); // Create directory
8:       }
9:
10: }
11:
12: // Receive Specified File
13: function action_upload(request){
14:
15:      // Retrieve Parameter Information(=RequestParameter Object)
16:      var parameter = request.getParameter("local_file");
17:
18:      // Retrieve File Contents (binary)
19:      var fileData = parameter.getValueAsStream();
20:
21:      // Retrieve File Name
22:      var fileName = parameter.getFileName()
23:
24:      // Write File
```

```
25:     var vf = new VirtualFile("filebox/" + fileName);
26:     var res = vf.save(fileData);
27:
28: }
```

This page will operate in the following manner.

1 : The selected file data will be sent to the server by file control.
2 : The data will be interpreted.
3 : The name of the received file will be retrieved.
4 : The file data with original file name will be output to Storage Service.



<Execution screen (Result)>

Column

<INPUT type="file">

Additional description of <INPUT type=”file”> used in a HTML form is as follows. With the use of form control <INPUT type=”file>, you can upload files directly from a browser to a server.
For the purpose, the form needs the following description.

```
<IMART type="form" method="POST" enctype="multipart/form-data">
```

This description encodes the file information into MIME format and sends request to the server in POST mode.

Column

RequestParameter Object

intra-mart offers RequestParameter Object which facilitates management of uploaded files. By using this object, files can be retrieved without needing any special process on the received data at the server-side. RequestParameter Object also has methods to retrieve uploaded file names. For details, please refer to “API List”.

3.1.2 Displaying File List

In this section, the page created in previous section will be improved by modifying the source to display a list of uploaded files.

<Presentation Page(filer.html) to display the listing>

```
1:  <HTML>
2:      <HEAD>
3:          <TITLE>File Center</TITLE>
4:      </HEAD>
5:      <BODY bgcolor="WhiteSmoke">
6:          <H2>Upload</H2>
7:          <IMART type="form" action="action_upload" method="POST" enctype="multipart/form-data">
8:              <INPUT type="file" name="local_file">
9:              <INPUT type="submit" value=" send ">
10:         </IMART>
11:         <HR>
12:         <H2>File List</H2>
13:         <TABLE>
14:             <TR><TH>File Name</TH></TR>
15:             <IMART type="repeat" list=fList item="rec">
16:                 <TR>
17:                     <TD><IMART type="string" value=rec></IMART></TD>
18:                 </TR>
19:             </IMART>
20:         </TABLE>
21:     </BODY>
22: </HTML>
```

To display a listing of uploaded files, the saved files list must first be retrieved. Codes are added for connection to the Storage Service and obtaining the file list.

<Function Container modified to retrieve the file list>

```
1:  var fList, rec;
2:
3:  // Page Initialization
4:  function init(request){
5:      var root = new VirtualFile("filebox");
6:
7:      // Check whether storage directory exist
8:      if(! root.isDirectory()){
9:          root.makeDirectories();
10:     }
11:
12:     fList = root.files(); // Retrieve File List
13: }
14:
15: // Receive Specified File
16: function action_upload(request){
17:
18:     // Retrieve Parameter Information(=RequestParameter Object)
19:     var parameter = request.getParameter("local_file");
20:
21:     // Retrieve File Contents (binary)
22:     var fileData = parameter.getValueAsStream();
23:
24:     // Retrieve File Name
25:     var fileName = parameter.getFileName()
26:
```

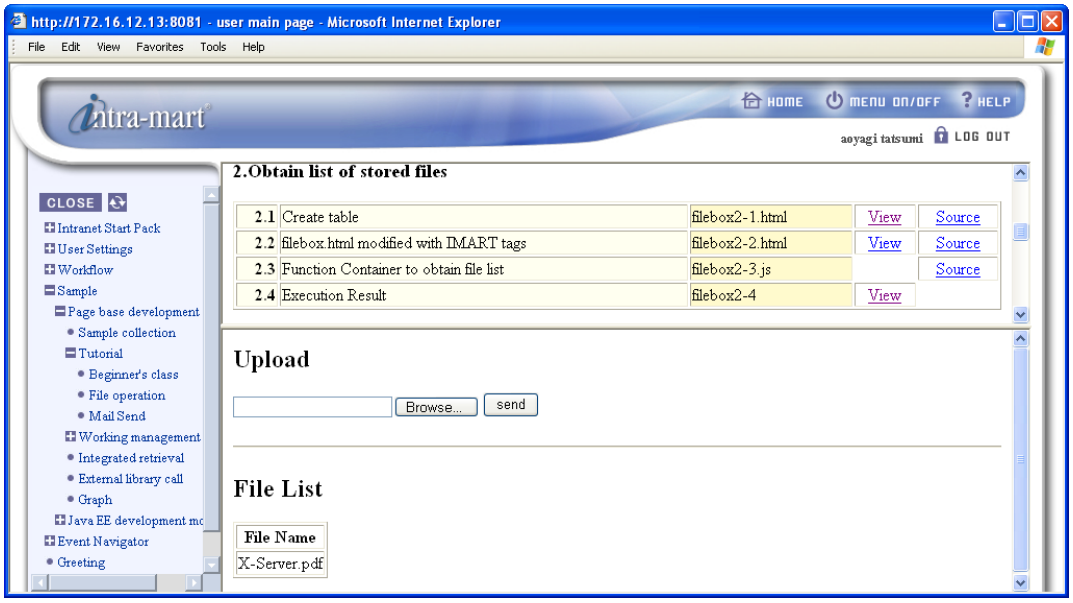
```
27:     // Write File
28:     var vf = new VirtualFile("filebox/" + fileName);
29:     var res = vf.save(fileData);
30:
31: }
```

Column

API VirtualFile to Manage Files in Storage Service

This API is used to retrieve, modify (create/delete) the files and directories in Storage Service ([%Storage Service%/storage/] in the Standard Version).

It enables creating and retrieving of various files, creating directories and retrieving of listings, as well as sharing of files independent of the environment between Application Runtimes in the case of clustered system.



<Execution screen (Result)>

On this execution screen, you can check whether the files have been uploaded.

3.1.3 File Download

You have completed making the functions to upload files and to check their presence in the Storage Service so far. Next step is to add functions to download the saved files from a browser. The source code of the page created above is modified to allow file downloading.

To download files, a download link is added to the Presentation Page. In addition, logic codes have to be added in the Function Container to transmit the files when the link is clicked.

<Presentation Page with added file download link(filer.html)>

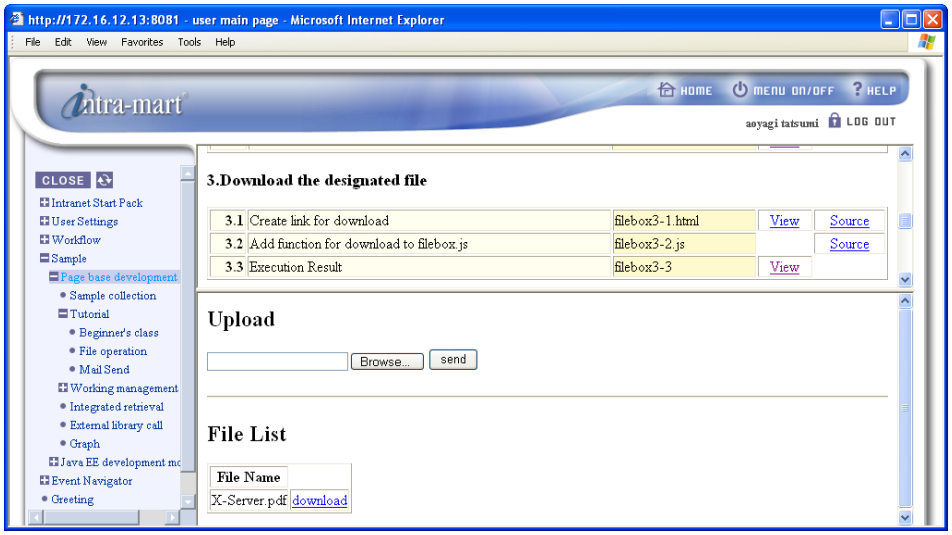
```
1:  <HTML>
2:      <HEAD>
3:          <TITLE>File Center</TITLE>
4:      </HEAD>
5:      <BODY bgcolor="WhiteSmoke">
6:          <H2>Upload</H2>
7:          <IMART type="form" action="action_upload" method="POST" enctype="multipart/form-data">
8:              <INPUT type="file" name="local_file">
9:              <INPUT type="submit" value=" send ">
10:         </IMART>
11:         <HR>
12:         <H2>File List</H2>
13:         <TABLE>
14:             <TR><TH>File Name</TH><TH></TH></TR>
15:             <IMART type="repeat" list=fList item="rec">
16:                 <TR>
17:                     <TD><IMART type="string" value=rec></IMART></TD>
18:                     <TD>
19:                         <IMART type="link" action="action_download" server_file=rec>download</IMART>
20:                     </TD>
21:                 </TR>
22:             </IMART>
23:         </TABLE>
24:     </BODY>
25: </HTML>
```

Create a link to show “Download”.
As this link is specified with the server function “action_download” to process downloading when clicked, it is also necessary to add the download function “action_download” in the Function Container.

<Function Container with added Function for Downloading>

```
1:  var fList, rec;
2:
3:  // Page Initialization
4:  function init(request){
5:      var root = new VirtualFile("filebox");
6:
7:      // Check if storage directory exist
8:      if(! root.isDirectory()){
9:          root.makeDirectories();
10:     }
11:
12:     fList = root.files(); // Retrieve File List
13: }
14:
15: // Receive Specified File
16: function action_upload(request){
17:
18:     // Retrieve Parameter Information(=RequestParameter Object)
```

```
19:     var parameter = request.getParameter("local_file");
20:
21:     // Retrieve File Contents (binary)
22:     var fileData = parameter.getValueAsStream();
23:
24:     // Retrieve File Name
25:     var fileName = parameter.getFileName()
26:
27:     // Write File
28:     var vf = new VirtualFile("filebox/" + fileName);
29:     var res = vf.save(fileData);
30:
31: }
32:
33: // Send Specified File
34: function action_download(request){
35:     var fpath = new VirtualFile("filebox/" + request.server_file);     // Retrieve
36:     Module.download.send(fpath.load(), request.server_file);          // Send
37: }
```



<Execution screen (Result)>

Column

Download “API Module.download.send()”

This API is used to send data from the server in formats other than HTML, and is used mainly for downloading files. When downloading data, browsers need to have additional information to specify which format the data is in. Whereas this API would have the function to auto-detect the data format and download in an appropriate way.

For instance, the extension “***.doc” is used for Word files in downloading. The Download API sees the file extension and set the downloading to MIME code. The downloaded document will appear on the browser if the computer is already installed with Microsoft Word, or a dialogue box to save the downloaded file will be shown if Microsoft Word is not installed.

3.1.4 Deleting Files

By now, the process to upload and download files had been described and demonstrated. However, if these files were left there, the Storage Service will be full after a while. This section adds a function that delete files saved on the Storage Service.

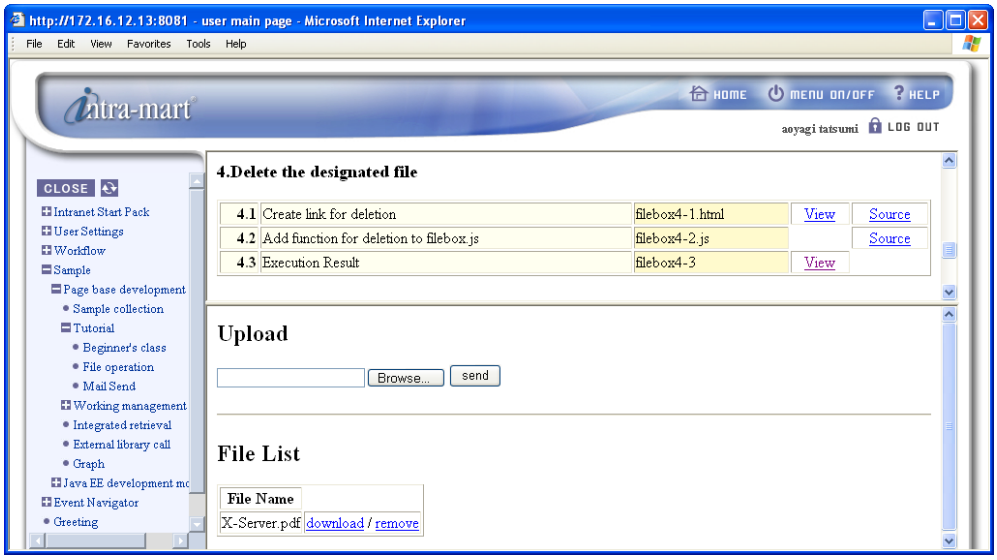
<Presentation Page with an added Delete Link(filer.html)>

```
1:  <HTML>
2:      <HEAD>
3:          <TITLE>File Center</TITLE>
4:      </HEAD>
5:      <BODY bgcolor="WhiteSmoke">
6:          <H2>Upload</H2>
7:          <IMART type="form" action="action_upload" method="POST" enctype="multipart/form-data">
8:              <INPUT type="file" name="local_file">
9:              <INPUT type="submit" value=" send ">
10:         </IMART>
11:         <HR>
12:         <H2>File List</H2>
13:         <TABLE>
14:             <TR><TH>File Name</TH><TH></TH></TR>
15:             <IMART type="repeat" list=fList item="rec">
16:                 <TR>
17:                     <TD><IMART type="string" value=rec></IMART></TD>
18:                     <TD>
19:                         <IMART type="link" action="action_download" server_file=rec>download</IMART>
20:                         / <IMART type="link" action="action_remove" server_file=rec>remove</IMART>
21:                     </TD>
22:                 </TR>
23:             </IMART>
24:         </TABLE>
25:     </BODY>
26:</HTML>
```

<Function Container(filer.js)with added Delete Function>

```
1:  var fList, rec;
2:
3:  // Page Initialization
4:  function init(request){
5:      var root = new VirtualFile("filebox");
6:
7:      // Check if storage directory exist
8:      if(! root.isDirectory()){
9:          root.makeDirectories();
10:     }
11:
12:     fList = root.files(); // Retrieve File List
13: }
14:
15: // Receive Specified File
16: function action_upload(request){
17:
18:     // Retrieve Parameter Information(=RequestParameter Object)
19:     var parameter = request.getParameter("local_file");
20:
21:     // Retrieve File Contents (binary)
22:     var fileData = parameter.getValueAsStream();
```

```
23:
24:         // Retrieve File Name
25:         var fileName = parameter.getFileName()
26:
27:         // Write File
28:         var vf = new VirtualFile("filebox/" + fileName);
29:         var res = vf.save(fileData);
30:
31: }
32:
33: // Send Specified File
34: function action_download(request){
35:         var fpath = new VirtualFile("filebox/" + request.server_file); // Retrieve
36:         Module.download.send(fpath.load(), request.server_file);            // Send
37: }
38:
39: // Delete Specified File
40: function action_remove(request){
41:         var fpath = new VirtualFile("filebox/" + request.server_file); // Retrieve
42:         fpath.remove();                                                                    // Delete
43: }
```



<Execution screen (Result)>

Click the delete link to delete the file selected.

Column

remove() Method of VirtualFile

This API is used to delete files and directories.
However, for deleting directories, please ensure that the directory should be empty and should have neither files nor directories before deleting.

Column

Screen Transitions

Screen transition is not demonstrated in this sample as all functions are available in 1 screen.

Therefore, in this case it is not necessary to specify ‘page’ for the links and forms after the various function requests or the display after requests, the page will refresh itself.

3.2 Sending E-Mails

This section describes how to create a page to send e-mails.

Please configure SMTP server of “conf/imart.xml” in order to send e-mails.

3.2.1 Creating Form for Sending E-Mail

In this section, a form to send mail will be created.

The form (in the Presentation Page) needs to have a control to register recipient’s address, sender’s address, e-mail title, and e-mail message. In addition, a function (in the Function Container) that sends an e-mail using Mail API based on the received information also needs to be defined.

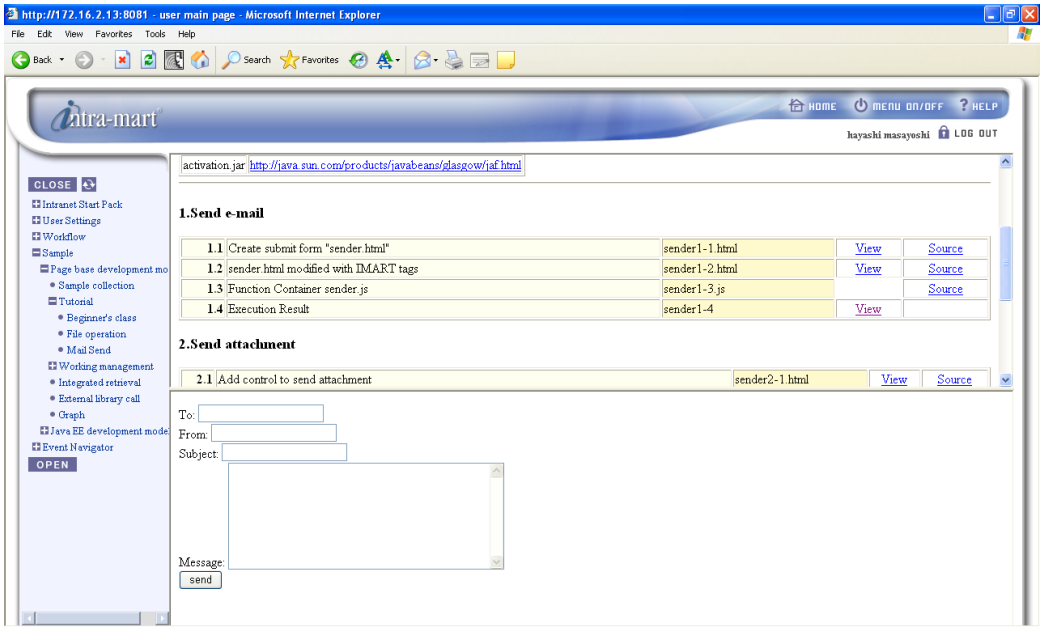
<Presentation Page (sender.html) with Mail Send Data Form coded>

```
1:    <HTML>
2:        <HEAD>
3:            <TITLE>Mail Sender</TITLE>
4:        </HEAD>
5:        <BODY bgcolor="WhiteSmoke">
6:            <IMART type="form" action="action_send" method="POST">
7:                To: <INPUT type="text" name="mail_to"><BR>
8:                From: <INPUT type="text" name="mail_from"><BR>
9:                Subject: <INPUT type="text" name="mail_subject"><BR>
10:               Message: <TEXTAREA name="mail_body" cols="40" rows="8"></TEXTAREA><BR>
11:               <INPUT type="submit" value=" send ">
12:           </IMART>
13:       </BODY>
14:   </HTML>
```

<Function Container coded with Mail sending logic(sender.js)>

```
1:  // Send Mail
2:  function action_send(request){
3:
4:      // Retrieve Locale
5:      var locale = AccessSecurityManager.getSessionInfo().locale;
6:
7:      // Create MailSender Object
8:      var mailSender = new MailSender(locale);
9:
10:     // Set Sending Information
11:     mailSender.addTo(request.mail_to);
12:     mailSender.setFrom(request.mail_from);
13:     mailSender.setSubject(request.mail_subject);
14:
15:     // Set Mail Text
16:     mailSender.setText(request.mail_body);
17:
18:     // Send Mail
19:     if( mailSender.send() ){
20:         // Transmission Successful
21:         Module.alert.reload("SYSTEM.SUCCESS",
22:                             " Mail has been sent");
23:     }
24:     else{
```

```
25:             // Transmission Failed
26:             Module.alert.reload("SYSTEM.ERR",
27:                                     mailSender.getErrorMessage());
28:         }
29:
30:     }
```



<Execution screen (Result)>

Click on the [send] button after entering all the necessary details in the form to send an e-mail.

Column

MailSender API

This is an API to send mails. In addition to recipient's address, sender's address, mail title, and mail message coded in the sample, CC and BCC can also be set.

Column

Setting To: and From: Fields

Using MailSender Object enables the setting of addressee, sender, CC and BCC to be carried out using either the mail addresses or the name of the person.
For details, please refer to API List.

Column

E-mail Delivery and Server Processing Speed

intra-mart Application Server and SMTP sever must be linked up to send an e-mail.
It may take a longer processing time to send an e-mail depending on the volume of information of the e-mail, as well as network environment or network traffic condition.

3.2.2 Sending E-Mail with Attachment File

In this section, the mail delivery form created earlier will be modified such that an attachment file can be sent together with the mail. First of all, a form control will be added in the Presentation Page to enable uploading of attachment file, and the form attribute will also be changed.

Function Container will be modified corresponding to the form modification so that the e-mail delivery function can accept attachment files.

<Presentation Page with added File uploading control(sender.html)>

```
1:    <HTML>
2:        <HEAD>
3:            <TITLE>Mail Sender</TITLE>
4:        </HEAD>
5:        <BODY bgcolor="WhiteSmoke">
6:            <IMART type="form" action="action_send" method="POST" enctype="multipart/form-data">
7:                To: <INPUT type="text" name="mail_to"><BR>
8:                From: <INPUT type="text" name="mail_from"><BR>
9:                Subject: <INPUT type="text" name="mail_subject"><BR>
10:               Attachment: <INPUT type="file" name="mail_file"><BR>
11:               Message: <TEXTAREA name="mail_body" cols="40" rows="8"></TEXTAREA><BR>
12:               <INPUT type="submit" value=" send ">
13:           </IMART>
14:       </BODY>
15:   </HTML>
```

Column

Form in order to Upload Files

The following form must be coded to upload a file.

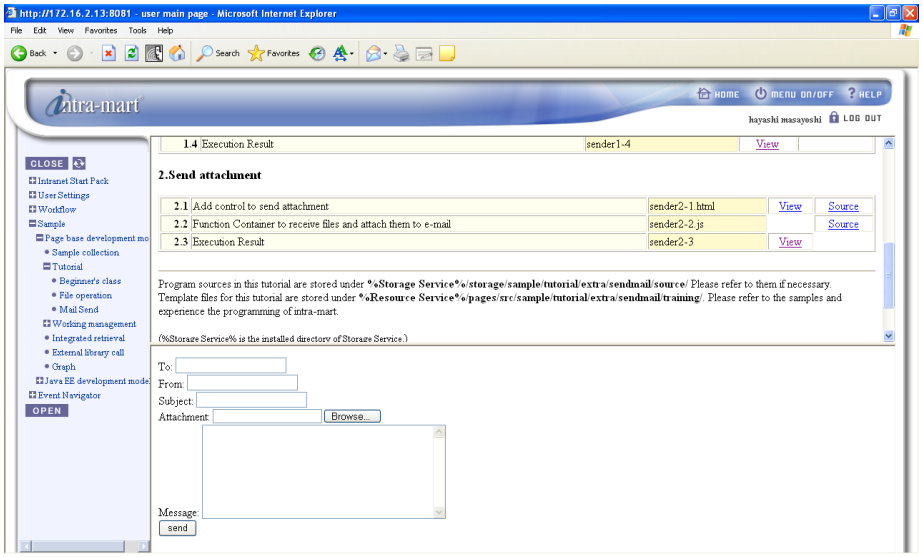
<FORM method=”POST” enctype=”multipart/form-data”>

With this form codes, the form control <INPUT type="file"> can accept a local file on the server.

< Function Container(sender.js)that can handle transmission of File Attachment>

```
1:  // Send Mail
2:  function action_send(request){
3:
4:      // Retrieve Locale
5:      var locale = AccessSecurityManager.getSessionInfo().locale;
6:
7:      // Create MailSender Object
8:      var mailSender = new MailSender(locale);
9:
10:     // Set Sending Information
11:     mailSender.addTo(request.mail_to);
12:     mailSender.setFrom(request.mail_from);
13:     mailSender.setSubject(request.mail_subject);
14:
15:     // Set File Attachment (Retrieve as RequestParameter Object)
16:     var parameter = request.getParameter("mail_file");
17:
18:     if( parameter != null && parameter.getLength() > 0 ) {
19:
20:         // Retrieve File Name
21:         var fileName = parameter.getFileName()
22:
```

```
23:          // Retrieve File Contents (binary)
24:          var fileData = parameter.getValueAsStream();
25:
26:          // Set File Attachment
27:          mailSender.addAttachment(fileName, fileData);
28:
29:      }
30:
31:      // Set Mail Text
32:      mailSender.setText(request.mail_body);
33:
34:      // Send Mail
35:      if( mailSender.send() ){
36:          // Transmission Successful
37:          Module.alert.reload("SYSTEM.SUCCESS",
38:                              "Mail has been sent");
39:      }
40:      else{
41:          // Transmission Failed
42:          Module.alert.reload("SYSTEM.ERR",
43:                              mailSender.getErrorMessage());
44:      }
45:
46:  }
```



<Execution screen (Result)>

Column

File Attachment and Mail Sending Speed

When an e-mail is sent out with file attachment, according to the regulations set by RFC, the file data must be first encoded, and then the entire e-mail information, including the e-mail's main body, has to be encoded before sending out to the SMTP server. Mail Server API automatically runs this encoding process, but it is inevitable to experience high-load on the application server during this process.

Column

Attachment File and Processing Speed

When an attachment file is send out, a browser will send the file data to the Web server, and then the application server that receives the file data will conduct e-mail delivery processing.

Since e-mail delivery must go through multiple numbers of networks, an e-mail with large-sized file attachment is likely to take much longer processing time.

3.3 Example on Usage of Extension <IMART> Tag Function

Extension <IMART> tag refers to the ability of the function to define any <IMART> tag with unique functionality in additional to the intra-mart default <IMART> tags.

To be more specific, it is a function that extends the API available in Presentation Page by defining new type-attribute to <IMART> tag, and registering the tag function that runs the process when this type-attribute is requested.

(Users can define their own definition for the “xxx” in <IMART type="xxx">)

3.3.1 Defining and Registering Tag

Code the process function for extension <IMART> tag in the function container. Register the invoking keyword by function and type attribute at the same time. Once this file is executed, information will be registered to Imart Object and it will always be available within Presentation Page. (Thus, this file is usually executed only once at the time of initial start-up.)

```
1:    //  Register function and invoking keyword
2:    //  Register the executing function “nowdate()” to
3:    //  the invoking keyword “NOW_DATE”
4:    Imart.defineType("JP_DATE", jpdate);
5:
6:    //  Define the executing function
7:    //【Argument】    oAttr : Tag-attribute Argument Group Object
8:    //                oInner: nested between <IMART>” and “</IMART>”
9:    //【Return】HTML Source
10:   //【Overview】Format conversion and displaying of the date data
11:   //  which is specified as <IMART type="JP_DATE"> process execution function
12:   //  with Date attribute.
13:   function jpdate(oAttr, oInner){
14:         var target = oInner.date;                    // Time to be displayed
15:         var format = "yyyyMMdd hhmmss";              // Display Format
16:         var src = Format.fromDate(format, target); // Character string to be displayed
17:
18:         //  Interpret and execute the source in the presentation page
19:         //  and obtain display source of area between
20:         //  <IMART type="NOW_DATE"> and </IMART>
21:         var sub = oInner.execute();
22:
23:         //  Return display source
24:         return src + sub;
25:   }
```

The character string returned from this extension <IMART> tag execution function will be sent to the browser as HTML source. In addition, this must be returned in the character string format.

3.3.2 Using Extension <IMART> tag

sample.html

To use this tag, you have to code the extension <IMART> tag in the presentation page.

```
<IMART type="NOW_DATE" date=now></IMART>
```

Once executed, the contents of date object (date and time of execution) bound from Function Container will be displayed.

sample.js

Create date data that is bound with the extension <IMART> tag.

```
var now;                    // Bind Variable
function init(request){
    now = new Date();       // Retrieve current time info
}
```

- For details, please refer to “Imart.defineType()” in “Application Common Module” in API List.

Column

Extension <IMART> Tag

User can create own definition for “XXX” in <IMART type="XXX">. For details, please refer to the section on Extension <IMART> Tag described in the Page Common Module in “API List”.

Column

Setting Constant Value to <IMART> Tag

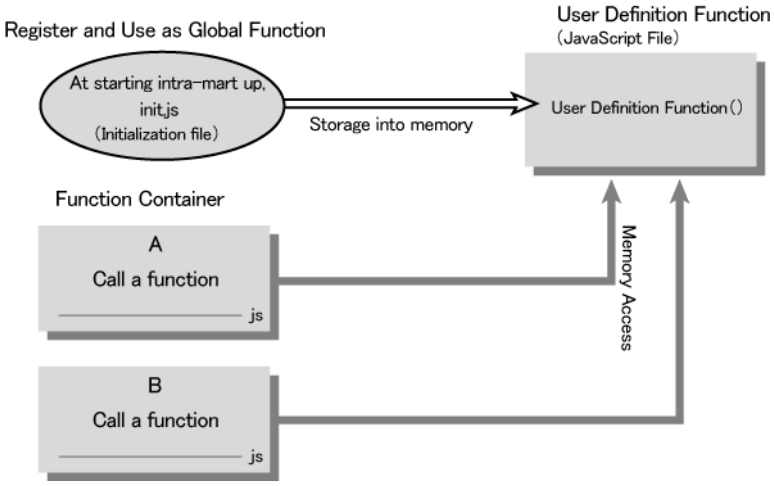
Constants or functions specified by user can be called with keyword specification of <IMART> tag-attribute. For details, please refer to the following methods of imart object described in “API List”

Imart.defineAttribute(sKeyWord, value)

3.4 Registration and Usage of User Defined Function

For development using intra-mart WebPlatform, the user defined functions coded with JavaScript can be registered as a “global function”.
Although registering function as “global function” using Procedure.define() method consume memory, performance will be much faster as the functions are saved in the memory.
Instructions to register/use the user defined function as global function are as follows.

- For details of Procedure.define() method, please refer to “Procedure” in “Application Common Module” in API List.



<Usage of User Defined Function>

3.4.1 Registering and Invoking as Global Function

The first step is to create the user defined function in any js file and code the default setting file (init.js) to store it in memory. This will enable the user defined function to be stored in the memory when intra-mart is started, and called directly from the memory by the function container (application js file).
Invoking of user defined function is coded in the function container.

1 Store the user defined function in any js file.

At this point, common function “addVariables()” is created in “library/common.js file” as an example.
Code the js file where global user defined function is to be stored using Procedure.define() method as follows.

```
< Code to store the user defined function to js file>
        //Register as a common function
        Procedure.define("addVariables", addVariables);

        //Create common function
        function addVariables( value A, value B )
        {
                return value A + value B;
        }
```

2 Configure settings to store in memory at start-up.

Code in the “init.js file (Default Setting File)” to retrieve the js file (with the user defined function) to store it in memory after intra-mart start-up.

```
<Coding to retrieve common function storage file>
        /* init.js */

        //Retrieve common function storage file
        include("library/common");
```

- Coding is not always necessary to be in init() function, as functions can also be retrieved with include() from other JavaScript file and become available as global functions.

3 Codes to call the user defined function in the function container.

Code in the Function Containers requiring the User Defined Functions to call the function stored in the memory after intra-mart start-up.
Here, common function “addVariables()” shall be called by “applicationPath/app001.js file” in the function container, as an example.
To call the user defined function, code the application js file as follows.

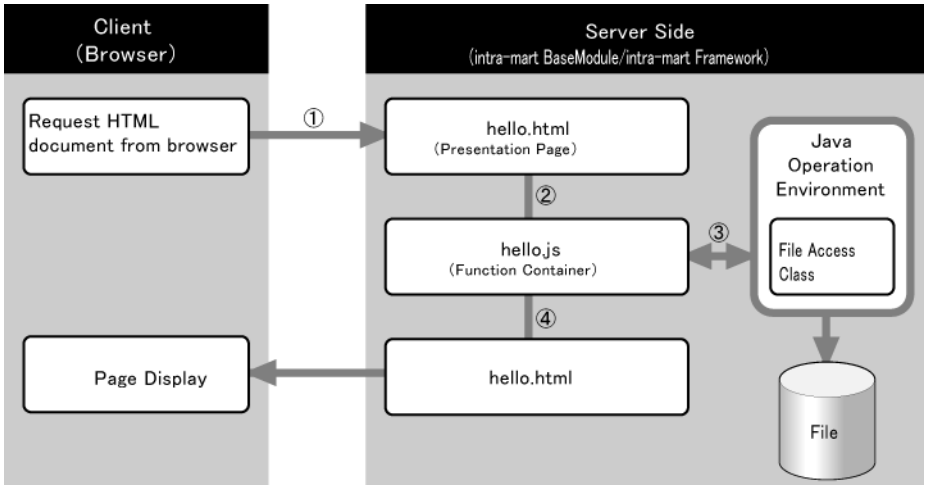
```
<Codes to call global user defined function in function container>

        /* applicationPath/app001.js */

        function add( )
        {
                //Call common function
                return Procedure.addVariables( 1,2 );
        }
```

3.5 Linkages with JavaClass

Server side JavaScript used in intra-mart has various excellent functions.
However, there could be some system architecture problems such as those related to communication functions or special file accesses due to its limitation as a script language.
To solve these problems by extending the problem area in the System Architecture as functions; this section describes on the linkage function between JavaScript and JavaClass.

3.5.1 How to Link with Standard JavaClass

intra-mart operates on JRE (Java Runtime Environment)-enabled environment and it can be linked with JDK's standard Class easily.
You can access to JavaClass method in a same way as with intra-mart object from intra-mart by defining the class using specified declaration method.
This section explains the method on usage of JavaClass use "Hello World" application described in the previous section as an example.
This "Hello World" application in Java is an application to display the contents of file on the server. Process flow of this application is as shown below.



<Processing image of "Hello World" Application Java Version>

1 Start the presentation page (HTML) file (hello.html) on the server from web browser.

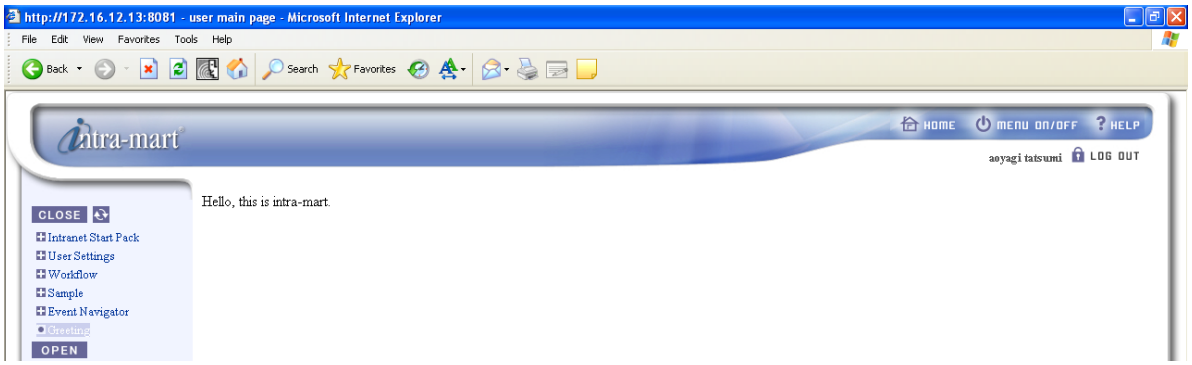
```
<hello.html (presentation page)>
<HTML>
<BODY>
Hello, this is <IMART type="string" value=nameValue></IMART>.</H1>
</BODY>
</HTML>
```

2 **The server side JavaScript processing in the function container (hello.js) coupled with presentation page is started.**

3 **Java file access class (1 – 4 of hello.js) called by server side JavaScript reads the external text files and returns the result to server side JavaScript.**

```
<hello.js (function container)>
  var nameValue = " " ;
        //Define init Function
  function init(request) {
        // Create JavaClass FileInputStream as JavaScript Object
        var javaObjFileIn = new java.io.FileInputStream( "c:¥¥hello.dat" );          (1)

        // Create JavaClass DataInputStream as JavaScript Object
        var javaObjDataIn = new java.io.DataInputStream( javaObjFileIn );           (2)

        // Read 1 record of file
        var javaObjString = javaObjDataIn.readLine( );                              (3)

        //File close
        javaObjDataIn.close( );                                                     (4)

        // Pass value to presentation page
        nameValue = Unicode.from(javaObjString);
  }
```

4 **Server side JavaScript replaces the <IMART> tag in the presentation page (HTML) with the result obtained from Java file access class and output.**



<hello.html Execution screen>

Column

Code Converting API

Unicode.*

Since the executing environment of intra-mart is Unicode, it cannot directly access the character string in local character system such as the contents saved in the files. For such occasion, it is necessary to use API to convert the character string in local character system supplied by intra-mart to Unicode. For details, please refer to “API List”.

3.5.1.1 Issue on Linkages with Standard JavaClass 標準JavaClasss 連携時の問題点

There is an issue on linking server side JavaScript with standard JavaClass; there is no established means to receive any exceptions occurred in standard JavaClass side.

Since server side JavaScript directly uses standard JavaClass, this issue is unavoidable indeed. However, the users can create their own JavaClass (refer to the following section) and providing instance variables/methods as a measure to check the occurrence of an exception.

3.5.1.2 How to Link with the Self-Created JavaClass

Linkage with self-created JavaClass requires special settings for intra-mart start-up and further coding on both the self-created JavaClass side and server side JavaScript.

3.5.1.3 How to Configure intra-mart Starting-up

It is necessary to either add the directory where self-created JavaClass is stored, or the class path of jar file using the “-cp” option of java command during intra-mart start-up. For the details on the usage of “-cp” option of java command, please refer to Java references.

- Class path is specified using “-cp” option. Setting is done with the conf/imart.xml file.

3.5.1.4 How to Code the Self-Created JavaClass side

Keep the following in mind when creating the self-created JavaClass.

Create as package.
Place the Class files where the class path passes through.

3.5.1.5 Coding Server Side JavaScript side

To create server side JavaScript, always add the “Packages” statement in front of Java package name when generating the self-created JavaClass as JavaScript objects.

```
var javaObjMail = new Packages.orgclass.myclass( );
```

As explained so far, just keep those points mentioned in mind and the server side JavaScript and self-created JavaClass can be linked easily. Server side JavaScript can also function as a container to call Java program applications.
As stated, development of more sophisticated applications is achievable using JavaScript on server side as base.
Below is a listing of “Hello World” as an example of coding using the self-created JavaClass.

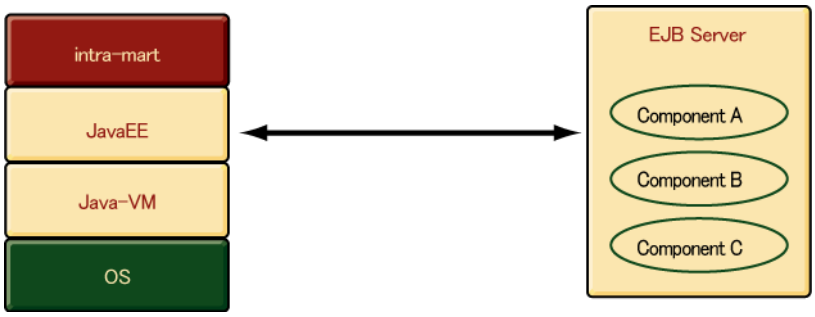
```
<hello.js (server side JavaScript source code)>
  var nameValue = " ";

  //Define init Function
  function init( requesst ) {

        //Create Java class hello as JavaScript object
        var javaObjHello = new Packages.intramart.imartjava.hello();

        // Read 1 record of file by getHellostr method of helloClass
        var javaObjString = javaObjHello.getHellostr( "c:¥¥hello.dat" );

        //Read instance variable for error check in JavaClass
        var javaObjError = javaObjHello.errstr;

        //Determin JavaClass error
        if(javaObjError.substring( 0,2 ) == "ER") {

              //If Error occurs
              //Pass error contents to the object which passes value to the presentation page
              nameValue = Unicode.form(javaObjError);
        } else {

              //Normal
              //Pass read contents to the object which passes value to the presentation page
              nameValue = Unicode.from(javaObjString);
        }
  }
```

<hello.java (self-created JavaClass source code)>

```
//Package definition
package intramart.imartjava;

//Class import
import java.lang.*;
import java.io.*;
import java.util.*;
//Class definition
class hello {

    //Instance variable to check error such as exception
    public String errstr;

    //Contractor
    public hello( ) {
    }

    //File read method
    public String getHellostr( String fnamestr ) {

        //Create instance of string variable to return
        String readstr = new String();

        //Initialize instance variable
        errstr = "OK";
        try {
            //Create instance of FileInputStream
            FileInputStream fs = new FileInputStream( fnamestr );

            //Create instance of FileInputStream
            DataInputStream ds = new DataInputStream( fs );

            //Read 1 record of file
            readstr = ds.readLine( );

            //Check file contents if null
            if(readstr == null) {
                //Set ERROR in instance variable
                errstr = "ERROR1";
            }
            //Close file
            ds.close( );
        }
        //Process exception
        catch(IOException e) {
            // Set ERROR in instance variable
            errstr = "ERROR2";
        }
        // Return read contents
        return readstr;
    }
}
```

3.6 Linkages with EJB

You can utilize various components of EJB (Enterprise JavaBeans)in intra-mart.



<Linkage with EJB server by embedding JavaEE>

3.6.1 Creating EJB Components

Create class in accordance with EJB Guidelines. For parts linked with JavaScript, it has to conform to the invoking by the self-created class. As for registration of created EJB components to EJB server and the setting for the names, please refer to the manuals supplied with each EJB server product.

3.6.2 Invoking from JavaScript

Configure class path appropriately and activate Application Runtime. Call the intended EJB component from inside the Function Container as shown below

```
<(Example) In case of invoking EJB component “XXX”>
        var initial = new Packages.javax.naming.InitialContext();
        var objref = initial.lookup("XXX");
        var home = Packages.javax.rmi.PortableRemoteObject.narrow
        (objref,java.long.Class.forName("XXXHome"));
        var interfaceXXX = home.create();
```

Since the called component is stored in JavaScript variable “interfaceXXX”, any API with EJB components can be executed in a same manner as invoking Java procedures.

- For details, please refer to “Using JavaClass” in API List.

3.7 Calling External Processes

To execute the program created by user from intra-mart application, global function, “execute()”, of the application common module is used. This function will execute a specified character string command as a new process, and idle till the executed process is completed.

Object-type Function	**(Object) execute ((String) command)**
Input Value	**(String) command: Executing Command**
Return Value	**Return value will be in Object-format and as shown below.**

When Executed Process Ends Successfully

```
return_object
    ├ output // Standard Output Stream from Process
        (String)
    ├ error // Error Output Stream from Process
        (String)
    └ exit // Process End Code
```

When Executed Process Does Not End Successfully

```
return_object
    ├ error //Error Contents (String)
    └ exit //Process End Code
```

- Process End Code gives a “0” when ended successfully.
- For details, please refer to Global Function “execute()”, “Application Common Module” in API List.

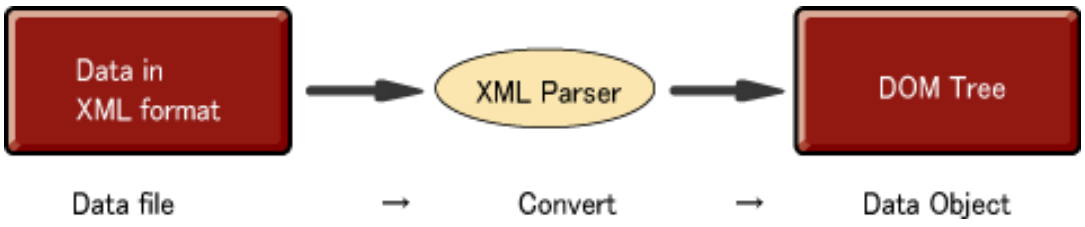
3.8 Handling XML Data

XML parser can help you to interpret XML data to extract desired data information.
XML (Extensible Markup Language) is a commonly used, easy to understand and highly flexible language for use with any type of environment. As a result, data can be exchanged easily and smoothly between intra-mart and other applications using a XML file.

Data can be transmitted between the systems using common format, XML.

3.8.1 XML Parser and Retrieving Data

The XML parser provided by intra-mart API can interpret and convert XML data into DOM (Document Object Model) tree format, and subsequently extract the XML tags and their contained data information from the DOM tree object.



- W3C regulates the standard for XML and DOM tree. Please refer to the W3C web site for the latest information.
- As for XML parser, the latest technical information is available on the W3C and SAX web sites.
- Please refer to “DOMXXX object” and “XML Parser object” under “Application Common Modules” in the API List for details.

3.8.2 How to Receive XML Data

Here is a simple example to create an application to receive the following XML format data.

```
<?xml version='1.0' encoding='UTF-8'?>
<account>
    <user-id>ueda</user-id>
    <name>Ueda</name>
    <role>
        <role-id sample-attr="Sample Attribute">level1</role-id>
    </role>
</account>
```

3.8.2.1 How to Receive XML Data by Using Request Object

You can refer to a value of the XML format data by using getParameter() and getParameterValue() method by using Request object (This process makes it easy to deal with XML format data sent from the rich clients such as Adobe Flash Player).

Specify the parameter name for the arguments of Request#getParameter() according to the following format.

- Specify each tag name of XML format data by separating "/".
- When obtaining the attribute values, you have to attach "@" in front of the attribute name.

Creating Function Container

```
1 : function  init(request){
2 :           var  userId       = request.getParameterValue("/account/user-id");
3 :           var  name           = request.getParameterValue("/account/name");
4 :           var  roleId       = request.getParameterValue("/account/role/role-id");
5 :           var  sampleAttr  =  request.getParameterValue("/account/role/role-id/@sample-attr");
6 :
7 :           Debug.browse(userId,  name,  roleId,  sampleAttr);
8 : }
```

In order to use this function, the following conditions have to be satisfied.

- Method of the request is "POST".
- Content-type entity header field of the request is "text/xml".
- The body part of the request message is XML data of which sentence structure can be analyzed.

3.8.2.2 How to Receive XML Data by Using XML Parser Object

Creating Function Container

```
 1 : function init(request){
 2 :
 3 :       //----------------------------
 4 :       // Obtain message body
 5 :       //----------------------------
 6 :       var messageBody = request.getMessageBody("UTF-8");
 7 :
 8 :       //----------------------------
 9 :       // Syntax analysis of XML data
10 :       //----------------------------
11 :       var xmlParser = new XMLParser();
12 :       var doc = xmlParser.parseString(messageBody);
13 :
14 :       if(xmlParser.isError()){
15 :             Debug.browse("Error occurs.", xmlParser.getErrorMessage());
16 :       }
17 :
18 :       //----------------------------
19 :       // <user-id>, <name>, <role> Obtai ning Node
20 :       //----------------------------
21 :       var childNodes    = null;
22 :       var accountNode = doc.getDocumentElement();
23 :       var userIdNode    = null;
24 :       var nameNode       = null;
```

```
25 :      var roleNode      = null;
26 :
27 :      childNodes = accountNode.getChildNodes();
28 :      for(var i = 0 ; i < childNodes.length ;i++) {
29 :           if(childNodes[i].getTagName() == "user-id") {
30 :                userIdNode = childNodes[i];
31 :           }
32 :           else if(childNodes[i].getTagName() == "name") {
33 :                nameNode = childNodes[i];
34 :           }
35 :           else if(childNodes[i].getTagName() == "role") {
36 :                roleNode = childNodes[i];
37 :           }
38 :      }
39 :
40 :      //-----------------------------
41 :      // <role-id>Obtain nodes
42 :      //-----------------------------
43 :      var roleIdNode    = null;
44 :      childNodes = roleNode.getChildNodes();
45 :      for(var i = 0 ; i < childNodes.length ;i++) {
46 :           if(childNodes[i].getTagName() == "role-id") {
47 :                roleIdNode = childNodes[i];
48 :           }
49 :      }
50 :
51 :      //-----------------------------
52 :      // <role-id>Obtain node attribution
53 :      //-----------------------------
54 :      var roleIdAttr = roleIdNode.getAttribute("sample-attr");
55 :
56 :      //-----------------------------
57 :      // Display the value of each node示
58 :      //-----------------------------
59 :      Debug.browse(userIdNode.getChildNodes()[0].getValue(),
60 :                   nameNode.getChildNodes()[0].getValue(),
61 :                   roleIdNode.getChildNodes()[0].getValue(),
62 :                   roleIdAttr);
63 :
64 : }
```

3.8.3 How to Send XML Data

General Web browsers use the Content-Type entity header field of the response in order to analyze the received data format. You have to specify “text/xml” for the Content-Type entity header field of the response in order to send XML format data created in the server. Here is a simple example to create an application to receive the following XML format data.

```
<?xml version='1.0' encoding='UTF-8'?>
<account>
    <user-id>ueda</user-id>
    <name>Ueda</name>
    <role>
        <role-id sample-attr="Sample Attributes">level1</role-id>
    </role>
</account>
```

3.8.3.1 How to Send XML Data by Using <IMART type="Content-Type"> Tag

Creating a presentation page (.html)

```
1 : <IMART type="Content-Type" value="text/xml; charset=UTF-8"></IMART>
2 : <?xml version='1.0' encoding='UTF-8'?>
3 : <IMART type="string" value=xmlString></IMART>
```

Creating a function container (.js)

```
1 : var xmlString = "";
2 :
3 : function init(request){
4 :
5 :     //-----------------------------
6 :     // Develop DOM tree
7 :     //-----------------------------
8 :     var doc = new XMLDocument("<account/>");
9 :     var accountNode = doc.getDocumentElement();
10 :
11 :    // Create elements
12 :    var userIdNode   = doc.createElement("user-id");
13 :    var nameNode     = doc.createElement("name");
14 :    var roleNode     = doc.createElement("role");
15 :    var roleIdNode   = doc.createElement("role-id");
16 :
17 :    // Create text nodes
18 :    var userIdText = doc.createTextNode("ueda");
19 :    var nameText   = doc.createTextNode("ueda");
20 :    var roleIdText = doc.createTextNode("level1");
21 :
22 :    // Configure attributes
23 :    roleIdNode.setAttribute("sample-attr", "sample attribute");
24 :
25 :    //Add child-nodes
26 :    userIdNode.appendChild(userIdText);
27 :    nameNode.appendChild(nameText);
28 :    roleNode.appendChild(roleIdNode);
29 :    roleIdNode.appendChild(roleIdText);
30 :
31 :    accountNode.appendChild(userIdNode);
32 :    accountNode.appendChild(nameNode);
33 :    accountNode.appendChild(roleNode);
34 :
35 :    //-----------------------------
36 :    //Bind XML character strings
37 :    //-----------------------------
38 :    xmlString = doc.getXmlString();
39 :
40 : }
```

3.8.3.2 How to Send XML Data by Using HTTPResponse Object

Creating function container (.js)

```
1 : function init(request){
2 :
3 :     //----------------------------
4 :     // Develop DOM tree
5 :     //----------------------------
6 :     var doc = new XMLDocument("<account/>");
7 :     var accountNode = doc.getDocumentElement();
8 :
9 :     // Create elements
10 :    var userIdNode   = doc.createElement("user-id");
11 :    var nameNode     = doc.createElement("name");
12 :    var roleNode     = doc.createElement("role");
13 :    var roleIdNode   = doc.createElement("role-id");
14 :
15 :    // Create text nodes
16 :    var userIdText = doc.createTextNode("ueda");
17 :    var nameText   = doc.createTextNode("ueda");
18 :    var roleIdText = doc.createTextNode("level1");
19 :
20 :    // Configure attributes
21 :    roleIdNode.setAttribute("sample-attr", "sample attributes");
22 :
23 :    // Add child-nodes
24 :    userIdNode.appendChild(userIdText);
25 :    nameNode.appendChild(nameText);
26 :    roleNode.appendChild(roleIdNode);
27 :    roleIdNode.appendChild(roleIdText);
28 :
29 :    accountNode.appendChild(userIdNode);
30 :    accountNode.appendChild(nameNode);
31 :    accountNode.appendChild(roleNode);
32 :
33 :    //----------------------------
34 :    // Bind XML character strings
35 :    //----------------------------
36 :    var encoding = "UTF-8";
37 :    var xmlString = "<?xml version='1.0' encoding='" + encoding + "'?>" + doc.getXmlString();
38 :
39 :    //----------------------------
40 :    // Configure Content-Type
41 :    //----------------------------
42 :    var response = Web.getHTTPResponse();
43 :    response.setContentType("text/xml; charset=" + encoding);
44 :
45 :    //----------------------------
46 :    // Send data
47 :    //----------------------------
48 :    response.sendMessageBodyString(xmlString);
49 :
50 : }
```

3.9 How to Use E4X

3.9.1 What is E4X?

ECMAScript for XML (E4X) is a programming language extension to add native XML support to JavaScript. E4X can help you to trace the class structure of XML by ". (dot)" and to deal with the attributes with "@ (at sign)" like JavaScript property.

E4X is standardized as ECMA-357 standard by ECMA International.

- http://www.ecma-international.org/publications/standards/Ecma-357.htm
- http://www.ne.jp/asahi/nanto/moon/specs/ecma-357.html(Japanese translation)

3.9.2 Creating XML Object

3.9.2.1 Creating XML Objects from XML Syntax

E4X can create XML object (delete this) XML objects by coding XML syntax in JavaScript Code. E4X can use <> for the initialization of XML as if JavaScript could use [] for initialization of the array or {} for initialization of the object.

```
1: var xml = <root>
2:                 <node attr="0">Sample</node>
3:             </root>;
```

3.9.2.2 Creating XML Objects from Character Strings

E4X can create XML objects from XML format character strings.

```
1: var src = "<root><node attr='0'>Sample</node></root>";
2: var xml = new XML( src );
```

3.9.3 Obtaining Values

E4X can help you to trace the class structure of XML by ". (dot)" and to deal with the attributes with "@ (at sign)" like JavaScript property.

```
1: var xml = <root>
2:                 <node attr="0">Sample0</node>
3:             </root>;
4:
5: Debug.print(xml.node);              // Value of the factor "Sample0"
6: Debug.print(xml.node.@attr);       // Value of the attribute "0"
7: Debug.print(xml.node["@attr"]); // The value of attribute, "attr",can be obtained in this way
```

3.9.4 Obtaining Child-Nodes

Child-nodes can be obtained by using “children()” or “.*”.

```
1: var xml = <root>
2:                  <node1>AAAA</node1>
3:                  <node1>BBBB</node1>
4:                  <node2 num="0">CCCC</node2>
5:                  <node2 num="1">
6:                      <node2Child>DDDD</node2Child>
7:                  </node2>
8:              </root>;
9:
10: var children = xml.children();    //This can be coded as “var children = xml.*;”
11:
12: for(var prop in children){
13:     Debug.print("children[" + prop + "] = " + children[prop]);
14: }
15:
16: // Obtain the value by using for each sentence.
17: for each (var value in children){
18:     Debug.print("value = " + value);
19: }
```

3.9.5 Adding nodes

You can add child-nodes and attributes as follows.

```
1: var xml = <root>
2:                  <node1>AAAA</node1>
3:              </root>;
4:
5: Debug.print("Before adding:" + xml.toString());
6:
7: xml.addedNode       = "BBBB";   // Adding child-nodes
8: xml.addedNode.@id = "CCCC";   // Adding attributes
9:
10: Debug.print("After adding:" + xml.toString());
```

3.9.6 Deleting Nodes

You have to use delete operator to delete nodes.

```
1: var xml = <root>
2:                  <node1>AAAA</node1>
3:                  <node1>BBBB</node1>
4:                  <node2 num="0">CCCC</node2>
5:                  <node2 num="1">
6:                      <node2Child>DDDD</node2Child>
7:                  </node2>
8:              </root>;
9:
10: Debug.print("Before deleting:" + xml.toString());
11:
12: delete xml.node1[1];
13: delete xml.node2[1].@num;
14:
```

```
15: Debug.print("After deleting:" + xml.toString());
```

3.9.7 Inserting Variables

A part of XML sentence can be variables in E4X. If inserting a variable with putting parentheses in XML sentence, the surrounded part can be replaced by the corresponding character strings to the default object.

```
 1: var attrName = "code";
 2: var tagName  = "name";
 3:
 4: var attrVal  = "0001";
 5: var content  = "product 1";
 6:
 7: var xml = <order>
 8:                 <item {attrName}={attrVal}>
 9:                     <{tagName}>{content}</{tagName}>
10:                 </item>
11:             </order>;
12:
13: Debug.print("============================");
14: Debug.print(xml.toXMLString());
15: Debug.print("============================");
16:
17:
18: //Add “product 2”
19: attrVal = "0002";
20: content = "poduct 2";
21: xml.appendChild(
22:         <item {attrName}={attrVal}>
23:             <{tagName}>{content}</{tagName}>
24:         </item>
25: );
26:
27: // Add “product 3”
28: attrVal = "0003";
29: content = "product 3";
30: xml.appendChild(
31:         <item {attrName}={attrVal}>
32:             <{tagName}>{content}</{tagName}>
33:         </item>
34: );
35: Debug.print("============================");
36: Debug.print(xml.toXMLString());
37: Debug.print("============================");
38:
39:
40: // Delete “product 2”
41: delete xml.item[1];
42:
43: Debug.print("============================");
44: Debug.print(xml.toXMLString());
45: Debug.print("============================");
```

3.10 JSSP-RPC

You call invoke server side logic coded in JavaScript seamlessly by using <IMART type="jsspRpc"> tag from Client Side JavaScript (CSJS as follows).

3.10.1 Operation Image

If "**sample/test1**.js"exists in the server side and "**testFunction**()"is defined, you can execute a function of server side from CSJS in the following procedures.

1 **Code <IMART type="jsspRpc"> tag in HTML file as follows.**

```
<IMART type="jsspRpc" name="serverLogic" page="sample/test1" >
```

2 **Execute server side logic by coding the following in CSJS.**

```
serverLogic.testFunction();
```

❖ If you would like to receive the processing result of the server side in asynchronous way, specify the callback attribute. The processing result of the server side is given to the argument of CSJS function specified as callback attribute. Please refer to API list for details.

3.10.2 JSSP-RPC Communication Error Object

When an error occurs in the communication with the server side using <IMART type="jsspRpc"> tag, the object which stores the error will be transmitted.

The followings are the cases when the communication error will occur.

- HTTP status code of the response is other than "200" (including the case that runtime error occurs in the server side).
- Session time-out occurs
- Debug.browse() is executed.

The way to transmit the error objects is different from JSSP-RPC communication method(synchronous communications or asynchronous communication). Please refer to the explanation of "jsspRpc" tag in API list about the details of error object structure and transmission method.

3.10.3 JSSP-RPC Sample Program (synchronous communications)

Make alert display about “Now = (current date)” after executing “**getNow**()” function defined in “**jssp_rpc_test/sample1**.js”.

3.10.3.1 HTML Source of Client Side

```
1: <html>
2:     <head>
3:         <IMART type="jsspRpc"
4:                 name="jsSample"
5:                 page="jssp_rpc_test/sample1">
6:         </IMART>
7:
8:         <script language="JavaScript">
9:             /**
10:             * Execute “getNow()”in “jssp_rpc_test/sample1.js” function.
11:             */
12:            function execute(){
13:                try{
14:                    var result = jsSample.getNow("Now = ");
15:                    alert(result);
16:                }
17:                catch(ex){
18:                    alert(ex.message);
19:                    return;
20:                }
21:            }
22:        </script>
23:    <head>
24:
25:    <body>
26:        <input type="button" value="Execute (synchronous)" onclick="execute();">
27:    </body>
28: </html>
```

3.10.3.2 JS Source of Server Side “jssp_rpc_test/sample1.js”

```
1: function getNow( args ){
2:     return args + (new Date()).toString();
3: }
```

3.10.4 JSSP-RPC Sample Program (Asynchronous communications)

Execute “**getObject**()”function defined in ”**jssp_rpc_test/sample2**.js”in the server side, and obtain the result object by call back function, “**callBackFunction**”.

3.10.4.1 HTML Source of Client Side

```
<html>
    <head>
        <IMART type     = "jsspRpc"
               name     = "jsSample"
               page     = "jssp_rpc_test/sample2"
               callback = "callBackFunction">
        </IMART>

        <script language="JavaScript">
            /**
             * Execute “getObject()” function in “jssp_rpc_test/sample2.js”.
             */
            function execute(){
                // Create arguments
                var obj = new Object();
                    obj.stringProp    = "value1";
                    obj.booleanProp   = true;
                    obj.numberProp    = -15;
                    obj.arrayProp     = new Array();
                    obj.arrayProp[0] = "ary0";
                    obj.arrayProp[1] = "ary1";
                    obj.arrayProp[2] = "ary2";
                    obj.dateProp      = new Date();

                // Check contents
                var str  = "";
                    str += "In order to check the callback function,"          + "¥n";
                    str += "the system sleeps for 5 seconds at the server side"    + "¥n";
                    str +=                                                          "¥n";
                    str += "Before executing"                                          + "¥n";
                    str += "-------------------------------------" + "¥n";
                    str += "obj.stringProp    = " + obj.stringProp    + "¥n";
                    str += "obj.booleanProp   = " + obj.booleanProp   + "¥n";
                    str += "obj.numberProp    = " + obj.numberProp    + "¥n";
                    str += "obj.arrayProp[0] = " + obj.arrayProp[0] + "¥n";
                    str += "obj.arrayProp[1] = " + obj.arrayProp[1] + "¥n";
                    str += "obj.arrayProp[2] = " + obj.arrayProp[2] + "¥n";
                    str += "obj.dateProp      = " + obj.dateProp      + "¥n";
                    str += "-------------------------------------" + "¥n";

                alert(str);

                // Execute server logic
                jsSample.getObject(obj);
            }


            /**
             * Callback function
```

```
50:                 */
51:                 function callBackFunction( result ){
52:
53:                     // Check contents
54:                     var str  = "";
55:                         str += "After executing"                                    + "¥n";
56:                         str += "-------------------------------------------"        + "¥n";
57:                         str += "result.stringProp    = " + result.stringProp       + "¥n";
58:                         str += "result.booleanProp   = " + result.booleanProp      + "¥n";
59:                         str += "result.numberProp    = " + result.numberProp       + "¥n";
60:                         str += "result.arrayProp[0] = " + result.arrayProp[0] + "¥n";
61:                         str += "result.arrayProp[1] = " + result.arrayProp[1] + "¥n";
62:                         str += "result.arrayProp[2] = " + result.arrayProp[2] + "¥n";
63:                         str += "result.dateProp       = " + result.dateProp       + "¥n";
64:                         str += "-------------------------------------------" + "¥n";
65:
66:                     alert(str);
67:                 }
68:
69:             </script>
60:         <head>
71:
72:         <body>
73:             <input type="button" value="Execute(synchronous)" onclick="execute();">
74:         </body>
75: </html>
```

3.10.4.2 JS Source of Server Side “jssp_rpc_test/sample2.js”

```
1: function getObject( args ){
2:
3:     // Display contents of the received object
4:     for(var prop in args){
5:         if (args.hasOwnProperty(prop)) {
6:             Debug.print(prop + " : " + args[prop] + " [" + typeof args[prop] + "]")
7:         }
8:     }
9:
10:    // Delayed treatmentis inserted in order to check the call back function.(for 5 seconds)
11:    Client.sleep(5 * 1000);
12:
13:    // Process the received object contents
14:    args.stringProp    = args.stringProp + " (modified !)";
15:    args.booleanProp   = false;
16:    args.numberProp    = args.numberProp + 10000;
17:    args.arrayProp[0] = args.arrayProp[0] + " (modified !)";
18:    args.arrayProp[1] = args.arrayProp[1] + " (modified !)";
19:    args.arrayProp[2] = args.arrayProp[2] + " (modified !)";
20:    args.dateProp.setFullYear(2100);
21:    args.dateProp       = args.dateProp;
22:
23:    // Return the result
24:    return args;
25: }
```

3.11 Debugging Process

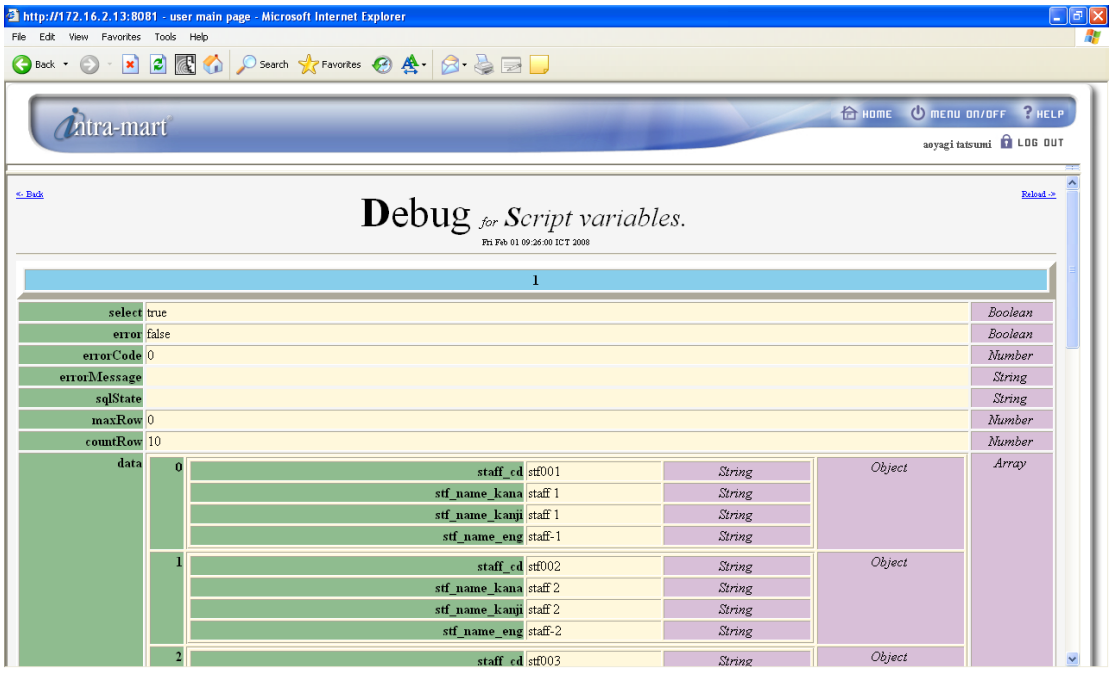
JavaScript created by the Developer can be debugged using Debug object. When Debug is executed, the name, type, value and dependency of user defined objects specified by the debug method can be checked on the debug result display page and console screen.

- For the details of debug method, please refer to “Debug.browse()”, “Application Common Module” in API List.
- Note that once Debug.browse() method is issued, the debug page will be displayed. Any script thereafter will not be executed.

3.11.1 Debugging Example

Example of codes for debugging in the function container and the display of debugging result:

```
<Example of Debugging Codes>
        // Declare Value to Pass to HTML
        var nameVale;
        var test;

        // Define init Function
        function init(){
                nameValue = Client.get( "nameValue" );       // Configure Value to pass to HTML
                var newDate = new Date();
                var returnOfGetAge = procedure.getAge( newDate );
                test = returnOfGetAge;
                Debug.browse(newDate, returnOfGetAge);   // Set variable
        }
```



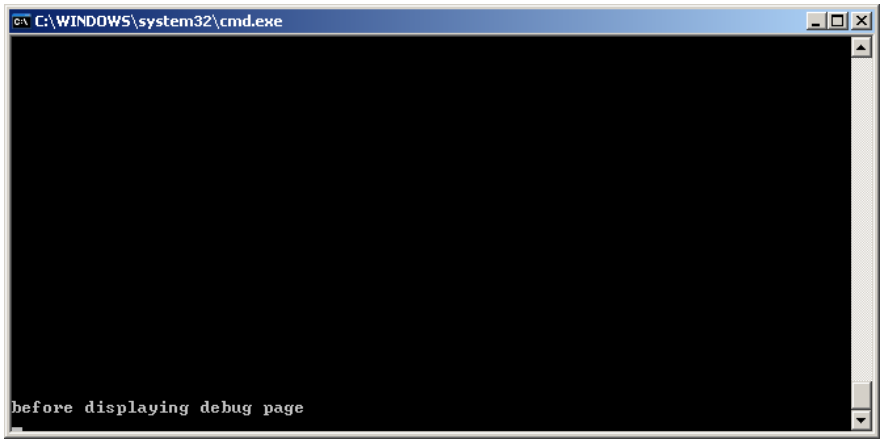
<Debugging Result Display Page>

3.11.2 How to Use Debug API

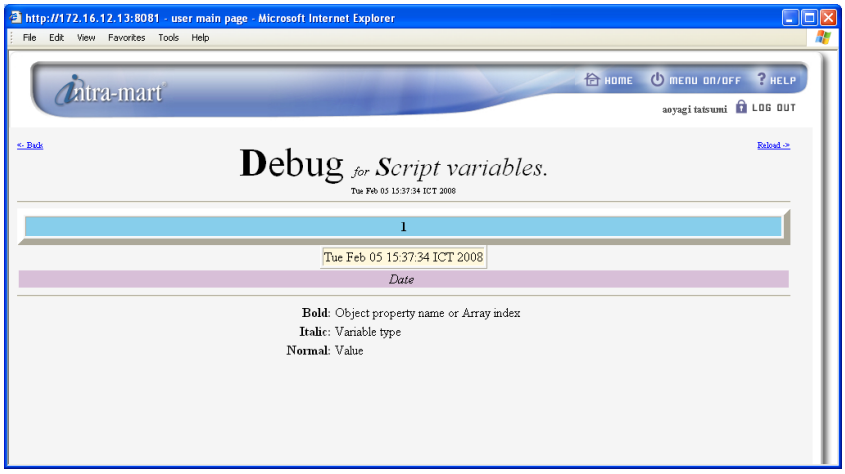
Use Debug API to check the contents of variables while writing codes for the Function Container.
The functions are available in the DebugClass in intra-mart WebPlatform.
To use Debug API while coding, use the following format.

```
<sample.js>
1:    //=========================================================
2:    //          【Enter】request: URL Argument Retrieving Object
3:    //          【Return】NIL
4:    //          【Overview】
5:    //=========================================================
6:    function init(request){
7:          var now = new Date();
8:          Debug.print("before displaying debug page");          // Output to Console
9:          Debug.browse(now);                                    // Output to Display
10:         Debug.print("after displaying debug page ");         // Output to Console
11:   }
```

In this sample source, a message “before displaying debug page” is displayed on DOS console screen, followed by the contents of the now variable (executed date and time) of the browser.
When “browse()API” is executed at the 9th line, execution of script will be stopped and debug screen will be displayed on the browser, and thus “print()API” on the 10th line will not be executed.



<DOS Console View (Result)-1>



<Browser Execution View (Result)-2>

- For details, please refer to “Debug” in API List.

Column

Debug.print()

This is a method which is able to output the debug code to the console window when debug mode is running. For details, please refer to “API List”.

Column

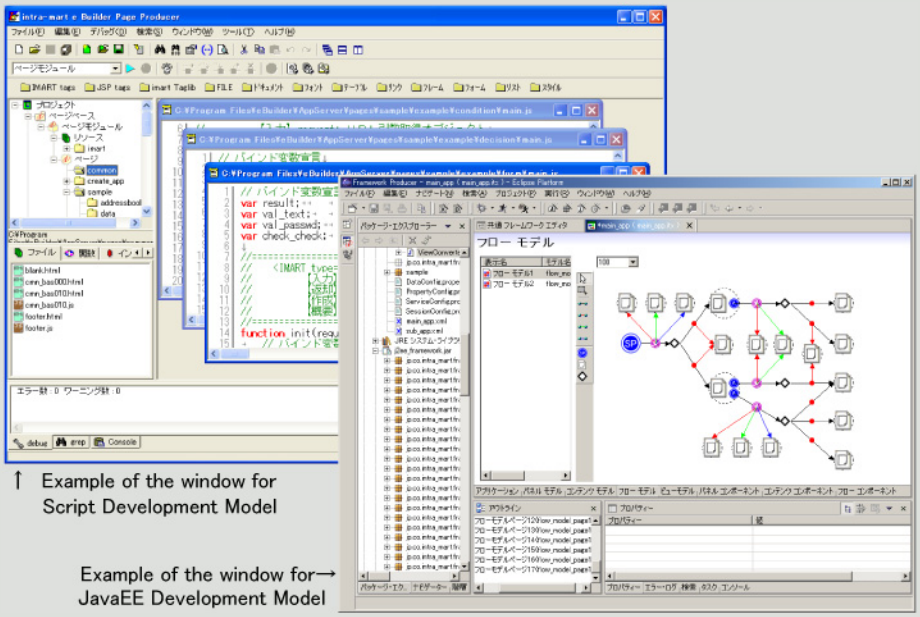
Debug.console()

This is a method which is able to output the object contents to the console window. The output contents are character strings in JSON format. For details, please refer to “API List”.

Column

“eBuilder 6.1” Supporting Program Development Environment (will be released on 2007/09/07)

With the use of intra-mart “eBuilder 6.0” (optional), it is possible to develop user applications more efficiently. intra-mart “eBuilder Ver6.0” comes with two form: “intra-mart eBuilder Page Producer” for script development model, consisting of presentation page and function container; and “eBuilder Framework Producer” consisting consist of JSP and Servlet that can be used as a plug-in for “Eclipse”.
Please refer to the Tutorial Guide “1.8 intra-mart eBuilder Ver6.0” for further details.



↑ Example of the window for Script Development Model

Example of the window for→ JavaEE Development Model

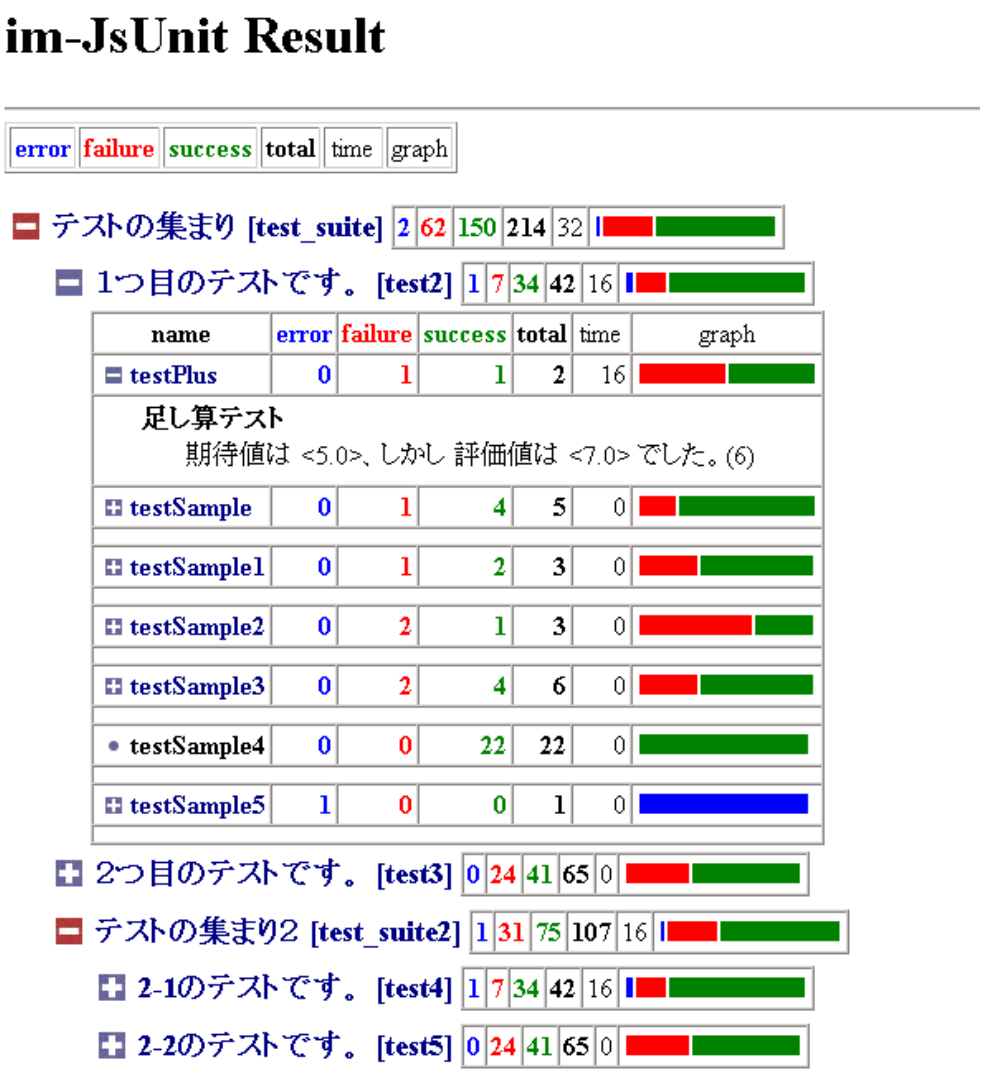
3.12 Environment of Unit Test (im-JsUnit)

im-JsUnit provides unit test environment in script development model.

Create a test case in script and execute a unit test of function test on the server.

The following figure is a sample of the result page of the unit test on the server.

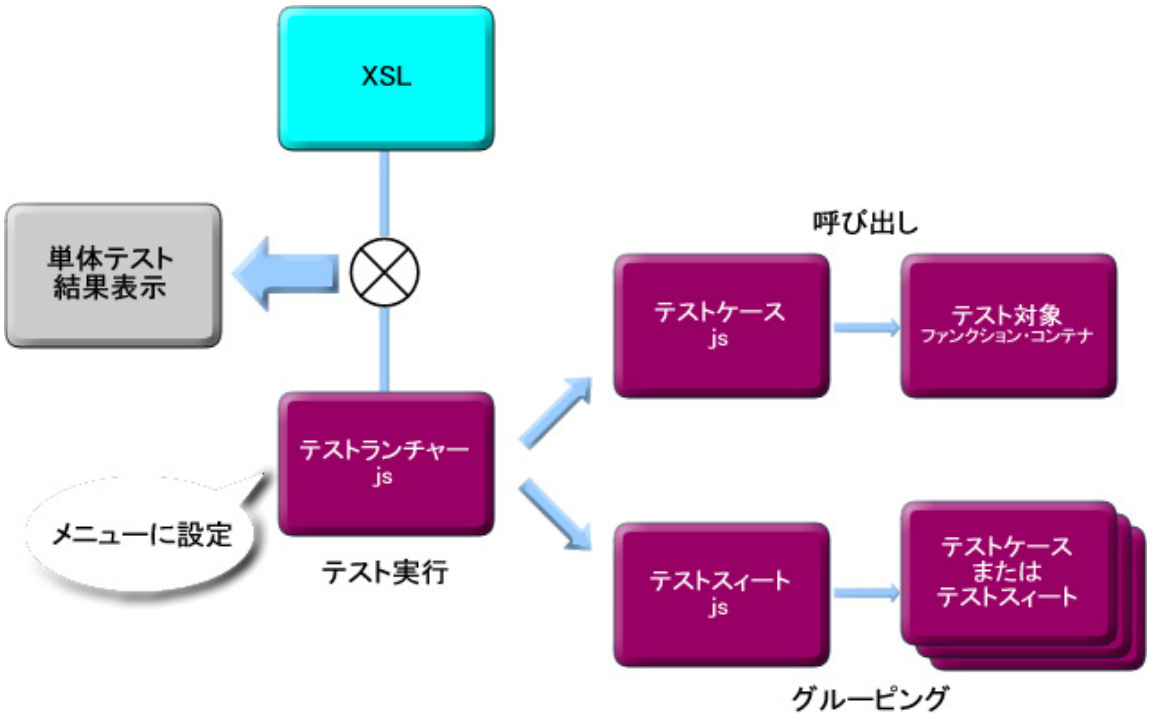
The test result status and error status are expressed in visually-apparent way.



3.12.1 im-JsUnit Guideline

Execute test case and test suite via test launcher.

After executing, form layout by XSL and display the test result.



Test Target	Test target is a function container targeted to the test. The functions described in the test target are tested.
Test Case	This is described as the functions of test target or test cases toward API. This is coded in JavaScript. Its extension is “ js”.
Test Suite	This makes multiple test cases or test suites grouped in one test group. You can execute multiple test cases or test suites by defining test suite. This is coded in JavaScript. Its extension is “ js”.
Test Launcher	This is a file to execute one specified test case or test suite. This is composed of function containers (js). (Presentation page is not necessary) Create this in each execution unit and register it to the menu. The unit test is executed from this menu.
XSL	The result by executing a test launcher is output in XML format. XSL creates a display layout based on the output XML and display it in the browser. Standard XSL is defined in “xsl/jsunit/im_jsunit.xsl”. You have to specify the file path of the XSL used in the function containers of test launcher.

3.12.2 Execution Order of Test Case

This section explains the order in which the functions described in test case are executed.

Types of the functions in test case

Types of the functions executed in test case are as follows.

testXXXXXX()	Search a function begun by “test” and execute it as needed. There is no rule about the order.
setUp()	This is executed before executing each testXXXXXX(). If this does not exist, this is not executed.
tearDown()	This is executed after executing each testXXXXXX(). If this does not exist, this is not executed.
oneTimeSetUp()	This is executed only once directly after loading a test case. If this does not exist, this is not executed.
oneTimeTearDown()	This is executed only once after executing all the testXXXXXX(). If this does not exist, this is not executed.
defineTestSuite()	If this is defined, this file is dealt as test suite. The other functions are neglected. If you create a file of test suite, you have to define only this function.

Execution order of the functions

The execution order of the functions in the test case is as follows.

If **testSample1()** and **testSample2()** are described as test functions in test case file, the order will be as follows.

① **oneTimeSetUp()**
② **setUp()**
③ **testSample1()**
④ **tearDown()**
⑤ **setUp()**
⑥ **testSample2()**
⑦ **tearDown()**
⑧ **oneTimeTearDown()**

3.12.3 Creating Test Case

Creating a test function

Test content is defined in the function begun by “test”.

```
function testSample1() {
    Describe test content.
}
```

Load test target file

In order to load a file of test target, you have to use the following API.

The path of the test target is specified with removing the extension.

If “user/test/source.js” is a test target, you have to specify “user/test/source”.

```
// Load a test target (user/test/source.js) as an object.
var module = JsUnit.loadScriptModule("user/test/source");
```

In order to execute a function in the test target, you have to code as follows.

```
// Load a test target (user/test/source.js) as an object.
var module = JsUnit.loadScriptModule("user/test/source");

function testSample1() {
    // Call the functions of test target.
    var result = module.calcPlus(1,2);
}
```

Creating test suite

Define **defineTestSuite** function for creating test suite.

Define **defineTestSuite** function for test suite file.

Create a test suite object in **defineTestSuite** function and add test case or test suite file to be grouped.

```
function defineTestSuite() {

    // Creating test suite object.
    var suite = new JsTestSuite("Test group");

    // Add a test case(Test Suite).
    suite.addTest("This is the first test.","test2");
    suite.addTest("This is the second test.","test3");
    suite.addTest("This is the third test.","test_suite2");

    // Return the test suite object.
    return suite;
}
```

3.12.3.1 How to Use Evaluation Function

Evaluation function is used to evaluate test result.

By using this function, information as test result is collected.

A list of Evaluation Function

assert([comment] ,value)	Check whether the evaluation value is “True”.
assertEquals([comment] ,value1 ,value2)	Check whether the evaluation value is same as expectation value.
assertFalse([comment] ,value)	Check whether the evaluation value is “False”.
assertNaN([comment] ,value)	Check whether the evaluation value is “NaN”.
assertNotEquals([comment] ,value1, value2)	Check whether the evaluation value is different from expectation value.
assertNotNaN([comment] , value)	Check whether the evaluation value is not “NaN”.
assertNotNull([comment] , value)	Check whether the evaluation value is not “Null”.
assertNotUndefined([comment] , value)	Check whether the evaluation value is not “Undefined”.
assertNull([comment] , value)	Check whether the evaluation value is “Null”.
assertTrue([comment] , value)	Check whether the evaluation value is “True”.
assertUndefined([comment] , value)	Check whether the evaluation value is “Undefined”.

Coding example

```
// Load a test target (user/test/source.js) as an object.
var module = JsUnit.loadScriptModule("user/test/source");

function testSample1() {
    // Call the function of test target.
    var result = module.calcPlus(1,2);

    // Test whether the function result is correct.
    JsUnit.assertEquals(3,result);

    // Test whether the function result is correct (with comment).
    JsUnit.assertEquals("Test of addition(1 + 2)",3,result);
}
```

3.12.3.2 Notes in Creating Test Case

When creating a test case, please pay attention to the following point.

You cannot use API associated with page transition.

If API occurring with page transition(**Debug.browse(), redirect(), forward(), Module.alert.*** or etc.)is described, the page will transit to the specified page. This causes an abnormal operation.

Please make a test with dividing the API.

3.12.4 Creating Test Launcher

Test launcher is a launcher file to execute test case or test suite.

Create a function container (js).

Code a test launcher by using a function which executes test case.

```
JsUnit.execute(Test case path,XSL path);
```

Specify the test case or test suite to be executed as a test case path.

If “user/test.js” is test target, specify “user/test”.

Specify a file which layouts the test result as XSL path.

Specify “**xsl/jsunit/im_jsunit.xsl**” normally.

The return value of this “**JsUnit.execute**” method is a character string of XML format.

In order to execute test case file(test.js), code as follows in test launcher file.

```
function init(request) {

    // Execute a test. (The result will be returned in XML character string)
    var result = JsUnit.execute("user/test","xsl/jsunit/im_jsunit.xsl");

    // Definition of content type
    // The result is XML format.  The encoding is UTF-8.
        var response = Web.getHTTPResponse();
        Web.getHTTPResponse().setContentType("text/xml; charset=UTF-8");

        // Data delivery
    response.sendMessageBodyString(result);
}
```

3.12.5 Executing Unit Test

The following explains the execution order of the unit test.

1. Create test case file or test suite file (if needed).
2. Create a test launcher in which the path of test case file or test suite file is described.
3. Register a path of test launcher created in□(delete) in menu setting page of system administrator or group administrator. (Specify it as launcher file name +.jssp)
4. Execute a launcher from the menu.
5. The test result is displayed in the screen.

3.13 JavaScript Compiler

Function

JavaScript compiler function is a function to convert (compile) the function container coded in JavaScript into JavaClass.

There are 2 types of JavaScript compiler functions as shown below.

Auto Compilation	**Application server (Application Runtime) will compile automatically when program (Function Container) is executed. Subsequently, execution will be with the compiled JavaClass (while server is operating, modification of the source will not be reflected instantaneously).** **This function will be activated when you switch the enable-attribute of "resource-file/javascript/compiler" tag in "%Resource Service%/pages/src/source-config.xml" to TRUE. Setting to FALSE will set the system in interpreter mode without compiling the Function Container (interpreter mode). (Any changes in the source code when the server is running will be reflected from the subsequent program run).**
Manual Compilation	**Create function container and compile it to JavaClass using JavaScript compile command. (For the details on JavaScript compile command, please refer to API List.)** **The advantage is that the performance will be better than using auto compilation. It is recommended to create JavaClass file by this method prior to the execution.**

Column

source-config.xml File

source-config.xml file is a setting file for programs in the directory (including subdirectory) where source-config.xml file is allocated.

Example of setting source-config.xml file

```
<resource-file>
    <charset>Windows-31J</charset>
    <javascript>
        <compiler enable="true" />
        <!-- enable:true = Auto compiler to Java class -->
        <!-- enable:false = Interpreter -->

        <optimize level="0" />
        <!-- level:0 to 9 = Optimize level of Compile -->
    </javascript>
    <view>
        <compiler enable="true" />
```

```
            <!-- enable:true = Auto compiler -->
            <!-- enable:false = Interpreter -->
        </view>
</resource-file>
```

Following settings can be done with source-config.xml file.

● resource-file/charset

Specifies the character encoding name for the source program.

● resource-file/javascript/compiler

Configures auto-compilation TRUE/FALSE of the function container.
When the setting is enabled (TRUE), the function container will be compiled into JavaClass for execution. (The class file will be created in %Application Runtime%/work/jssp/_functioncontainer directory.)
On the other hand, if this setting is disabled (FALSE), the function container will be executed by JavaScript interpreter.

● resource-file/javascript/optimize

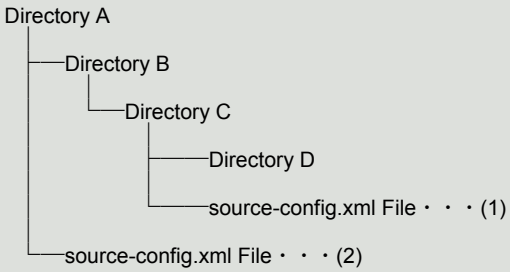
Configures the optimal level to compile the function container into JavaClass.

● resource-file/view/compiler

Configures auto-compilation TRUE/FALSE of the Presentation Page.
When the setting is enabled (TRUE), the presentation page will be compiled for execution. (The class file will be created in %Application Runtime%/work/jssp/_presentationpage directory.)
To disable this setting, select FALSE.

When “source-config.xml” file is allocated as in the chart shown below, the “source-config.xml” file referred by each program will be as follows.

* Program directly under A Directory： Configured contents in (2) will be enabled.
* Program directly under B Directory： Configured contents in (2) will be enabled.
* Program directly under C Directory： Configured contents in (1) will be enabled.
* Program directly under D Directory： Configured contents in (1) will be enabled.



< Example of source-config.xml>

Column

Configuring Auto Compilation per File Unit

For the programs of Script Development Model, the settings for the character code and auto-compilation status are done in pairs for the presentation page and the function container.
First create the “Subject File Label Name.properties” file, and configure as shown below.

charset = Character Encoding Name of Program
javascript.compile.enable = Auto Compilation Settings of Function Container
javascript.optimize.level = Optimization Level for Compiling Function Container to JavaClass
view.compile.enable = Auto Compilation Settings of Presentation Page

For instance, to enable the auto-compilation of function container and disable the auto-compilation of presentation page in “sample.html” and “sample.js” created using character code “Windows-31J”, create a sample.properties in the same directory, and code the following content:

sample.properties file

```
charset=Windows-31J
javascript.compile.enable=true
javascript.optimize.level=0
view.compile.enable=false
```

3.13.1 Runtime Search Procedure for Function Container

Function Container can be searched / executed in the following procedure.

1. **Searches manually-compiled Function Container (Java Class) from Class Path.**
2. **Searches Java Class that was updated later than source file (Function Container yet to be complied) from [%Application Runtime%/work/jssp/_functioncontainer].**
3. **Searches source file with [%Resource Service%/pages/src] as the root directory.**
4. **Searches source file with [%Resource Service%/pages/product/src] as the root directory.**
5. **Searches source file with [%Resource Service%/pages/platform/src] as the root directory.**

Once the applicable function container is found in the above process, it will be executed (If auto-compilation is enabled, the JavaClass file will be created under “%Application Runtime%/work/jssp/_functioncontaine” directory when source file is found under step 3 – 5.)
Executing a combination of JavaClass file and source file is also possible (for example, compiling part of the Function Container and executing the rest in Interpreter mode.)

Column

Directory Structure under Pages in “%Resource Service%”

Structure under [%Resource Service%/pages] has been upgraded in intra-mart Ver5.0. Developers usually store the programs in “%Resource Service%/pages/src”.

```
%Resource Service%/pages
  |
  ├─platform
  |   |
  |   └─src   ← Storing Script Programs of intra-mart WebPlatform/AppFramework
  |
  ├─product
  |   |
  |   └─src   ← Storing Script Programs of intra-mart applications like intranet StartPack,etc.
  |
  |
  |
  └─src       ←Storing Script Programs Created by Developer
                  (intra-mart WebPlatform samples will be stored in this directory too.)
```

3.13.2 Specification Details

The name of JavaClass created by the Compiler will be based on the file in the Function Container. If the name of the file in the Function Container contains any characters that cannot be used in JavaClass name (see notes below), the character will be automatically replaced with the “_” (underscore). However, error may occur when the original file name with the replaced “_” ended up overlapping with another file.

- For characters that cannot be used for class name, please refer to documents on Java specifications.

- If auto-compilation is enabled, JavaClass file will be created in [%Application Runtime%/work/jssp/_functioncontainer] when function container is executed. Please restart the server if program has been modified. If modification is still not reflected after restart, initialize the execution environment as follows.

  1 : Server Stop
  2 : Delete work/jssp/
  3 : Server Start

- JavaScript function name must be unique within the file. This rule applies to the functions declared within the function as well.
- Byte code structure of JavaClass file created with optimization function at the time of JavaScript compilation differs from the structure built without the optimization function. Therefore, there may be some differences in error occurrence or error contents between the two compilations.
- Load error may occur depending on the program contents.
- Load error may occur if code size (post-compilation) is too large. For such error, reduce the codes in each JavaScript function. (The functions in Function Container are compiled into JavaClass individually)
- If the program path at the invoking side of the function container is ambiguously defined (e.g. having incomplete matching of upper/lower-cases), load error or runtime error may occur (such as page attribute for include() function, link tag of (<IMART> tag, etc.). For such errors, correct the specified path.
- Similar errors will occur when path is incorrectly set for the batch program executions, etc. For such cases, please correct the path.
- Load error may occur when variables not used in the program are declared. Do not declare any unnecessary variables.

3.13.3 Specifications of Areas Not Directly Related to Compiler

Character string exceeding 64KB in the external memory area cannot be saved. This applies to API below.

Client.set()

3.13.4 Restrictions

3.13.4.1 Restrictions by File Size

For the static script area in the presentation page (html) restricted by file size (except <IMART> tags), runtime error will occur if the size of continuous codes exceeds 64 [KB].

3.13.4.2 Restrictions by Program Writing

Runtime error will occur when the following 2 conditions are met.

When “source-config.xml” is configured as follows:
Enabling attribute of resource-file/javascript/compiler tag is set to TRUE
Level attribute of resource-file/javascript/optimize tag is set to above 1
When program is written as follows:
There is a common function called from both functions executed by init() function and action-attribute.
Invoking function executed by action attribute from init() function.

Code causing Errors when level attribute of resource-file/javascript/optimize tag is set to more than 1

test_page.html

```
<HTML>
<HEAD>
<TITLE>Test Page</TITLE>
</HEAD>
<BODY bgcolor="WhiteSmoke">
<CENTER>
	<HR>
		<!—Call actionFunction -->
		<IMART type="form" action="actionFunction">
			<INPUT type="submit">
		</IMART>
	<HR>
</CENTER>
</BODY>
</HTML>
```

test_page.js

```
/**
* Initializing function
* @param request Web request argument
*/
function init(request){
	//Here, call actionFunction
	//actionFunction is also called form test_page.html
	actionFunction(null);
}
/**
```

```
* The Function Called by action-attribute of the Form
* @param request Web request argument
*/
function actionFunction(request){
	Debug.print(viewTime().toString());
}
/**
* Common function
* @return date type value which displays the current time
*/
function viewTime(){
	return new Date();
}
```

Code that does not cause errors when level attribute of resource-file/javascript/optimize tag is set to more than 1

```
test_page.html
<HTML>
<HEAD>
<TITLE>Test Page</TITLE>
</HEAD>
<BODY bgcolor="WhiteSmoke">
<CENTER>
	<HR>
	<!--Call actionFunction -->
	<IMART type="form" action="actionFunction">
		<INPUT type="submit">
	</IMART>
		<HR>
</CENTER>
</BODY>
</HTML>
```

```
test_page.js
/**
* Initializing function
* @param request Web request argument
*/
function init(request){
	//Here, actionFunction is not called
	//actionFunction is called only from test_page.html
	Debug.print(viewTime().toString());
}
/**
* Functions called by action attribute of the form
* @param request Web request argument
*/
function actionFunction(request){
	Debug.print(viewTime().toString());
}
/**
* Common function
* @return date type value which displays the current time
*/
function viewTime(){
	return new Date();;
}
```

3.14 Linkages with im-JavaEE Framework

Below API is used for the screen transition from pages of Servlet, JSP, etc to Script Development Model pages, under the im-JavaEE Framework.

```
jp.co.intra_mart.jssp.net.URLBuilder
```

This class is to create the URL to call Script Development Model pages. Use the following utility class to create URL showing request context.

```
jp.co.intra_mart.common.aid.jsdk.utility.URLUtil
```

Example

```
// Create URLBuilder
URLBuilder urlBuilder = new URLBuilder(request, response);

// Obtain URL to link to the specified Script Development Model page
// while maintaining the HTTP session.
java.net.URL url = urlBuilder.createURLonSession (Page path of Script Development Model) ;

// Construct the character string expression of this URL
String nextPageURL = url.toExternalForm();
```

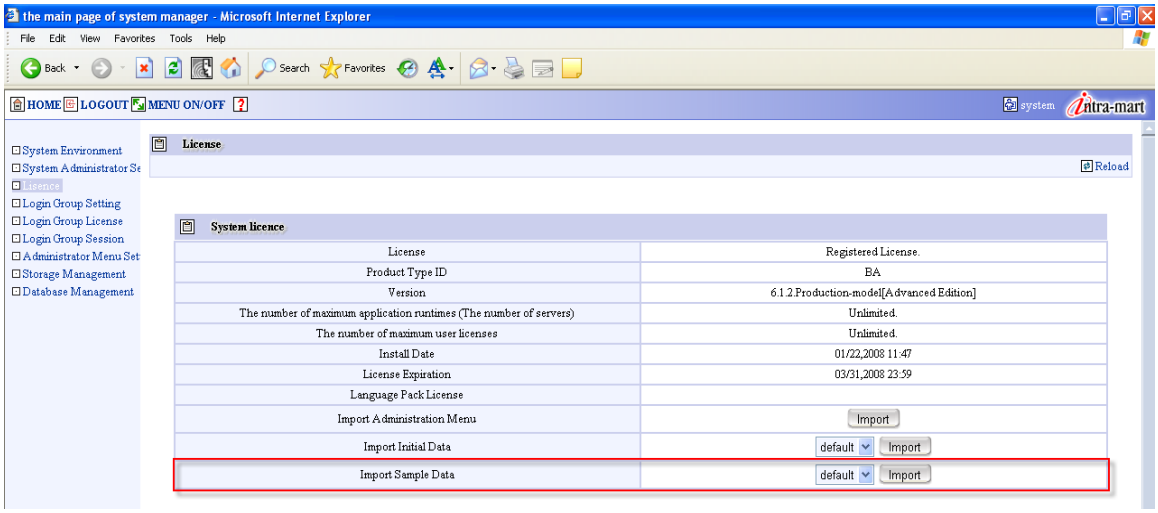
For the page path of Script Development Model, specify the same path (relative from %Resource Service%/pages/src) as the path usually specified for the implementations of Script Development Model.

3.15 Sample Applications

Intra-mart intra-mart WebPlatform/AppFramework offers the “Working management” as a sample application for Script Development Model.

3.15.1 Registering Sample Data in Database

Register sample data to the database. Sample data can be imported in [LICENSE] menu while logged in as the System Administrator. Select login group from the pull-down menu to import sample data, and click on the [IMPORT] button.



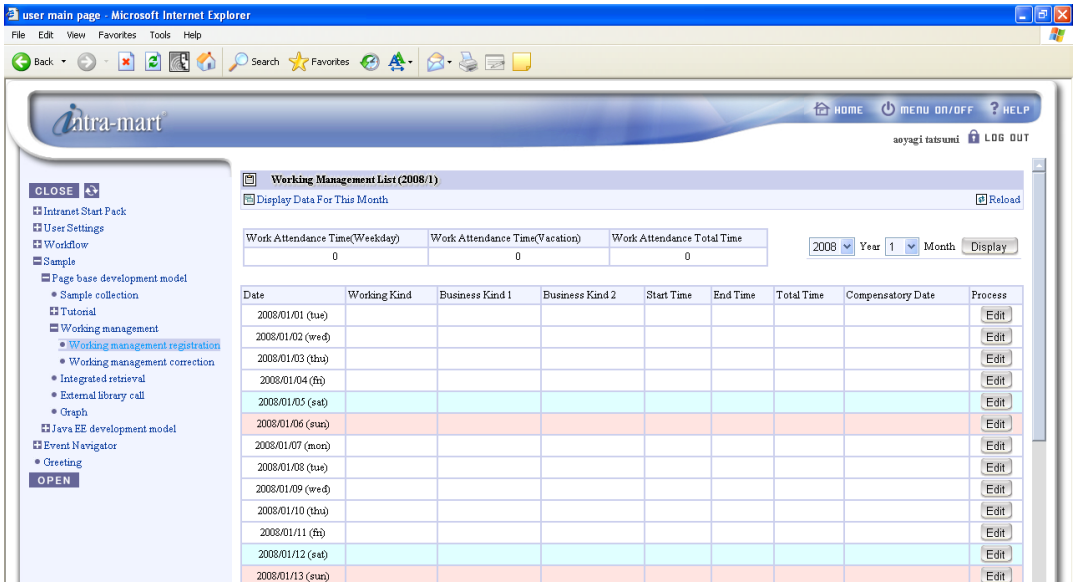
<Importing Data for Sample Applications>

3.15.2 Using "Working management"

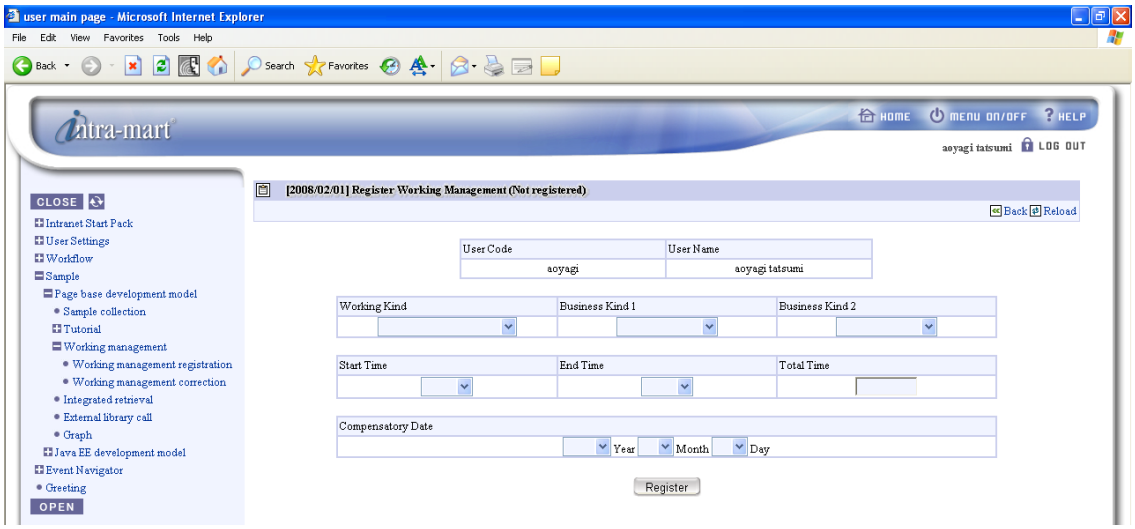
There are 2 menus: “Working management registration” and “Working management correction”, under the [Sample]-[Page base development model]-[Working management] in this application.

3.15.2.1 Working management registration

Select [Working management registration] from [Sample]-[Page base development model]-[Working management] to display the work attendance data for the current month. To view other months, select the desired year/month in the combo box on the right top of page and click the display button. To register the work attendance, click the edit button of the date to register. ”Working Kind”, “Business Kind”, “Start/End Time”, etc for the selected date can be edited.



<Work Attendance List>

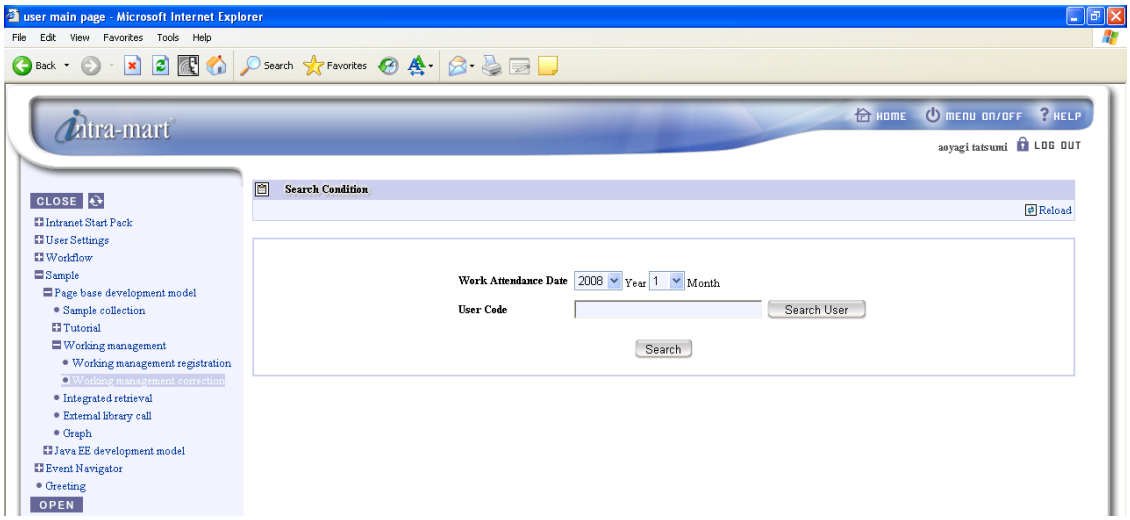


<Work Attendance Registration>

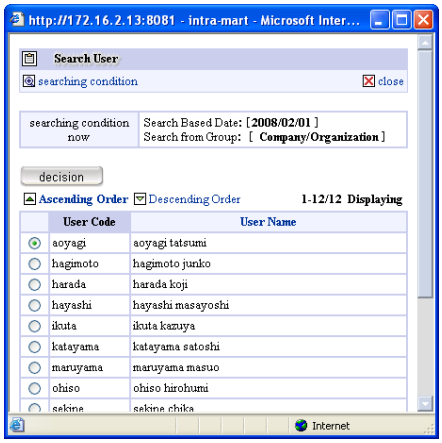
3.15.2.2 Working management correction

You can search the work attendance record by keyword of the registered record using “year/month” or applicant’s user code, and the found record can be opened for editing.

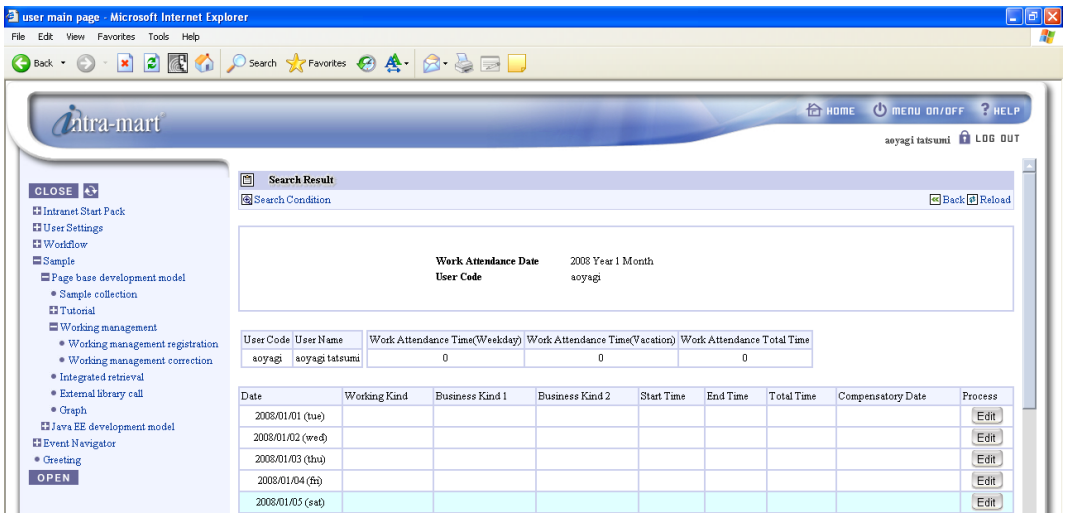
In the search criteria page, enter the work attendance year/month or the applicant’s user code. Click on the search button to display the match result. To amend, click on the edit button of the date to amend. Details like “Working Kind”, “Business Kind”, “Start/End Time”, etc can me edited.



<Search Criteria Input Page>



<User Search Page>

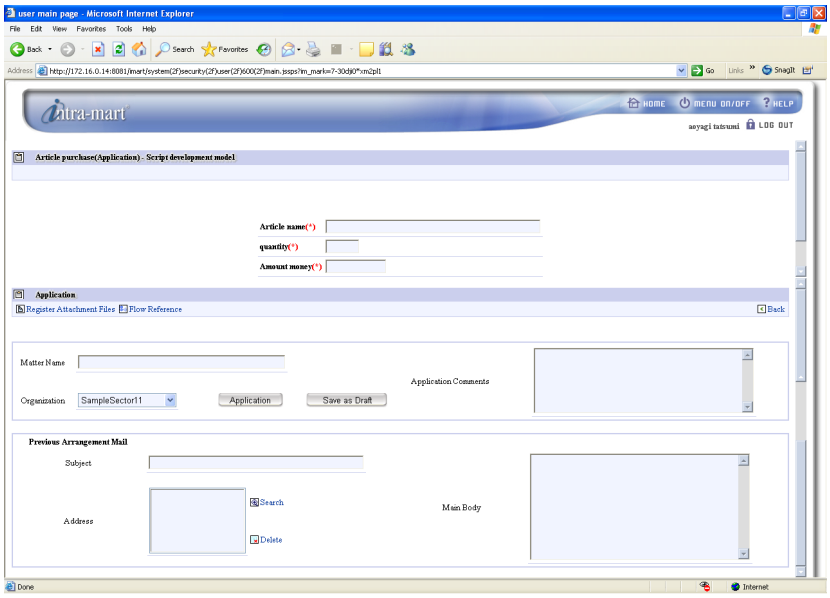


<Search Result Page>

3.15.3 Linkages with Workflow Module

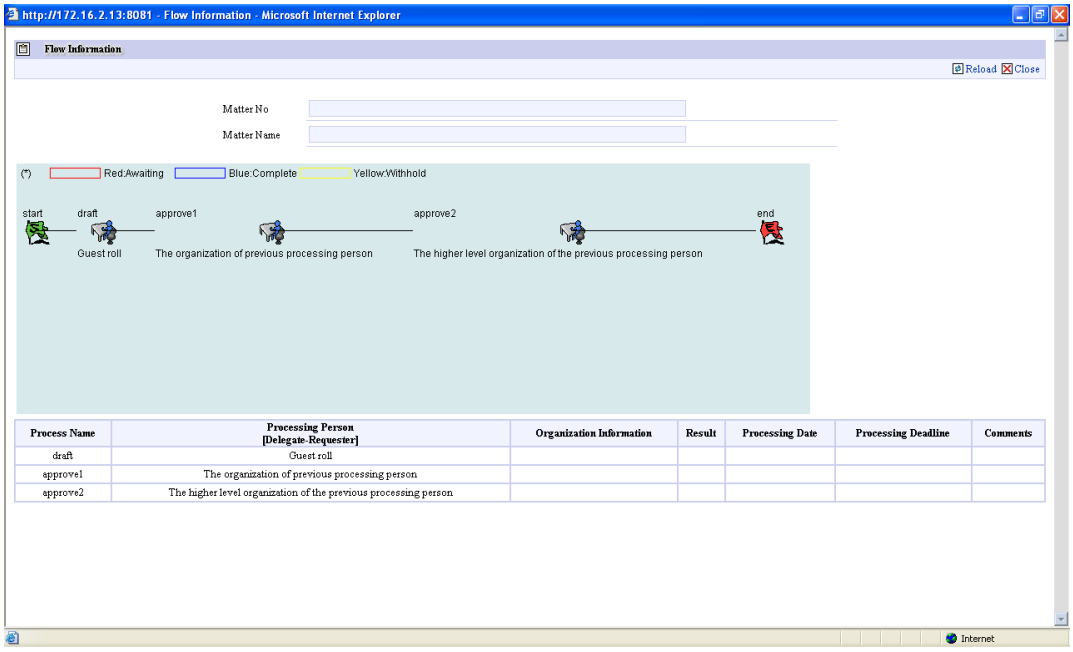
By using Workflow Module, you can create an application corresponding to the workflow easily. Please refer to sample applications linked with Workflow Module in “%Resource Service%/pages/src/sample/bpw/purchase/standard”.

3.15.3.1 Application Screen



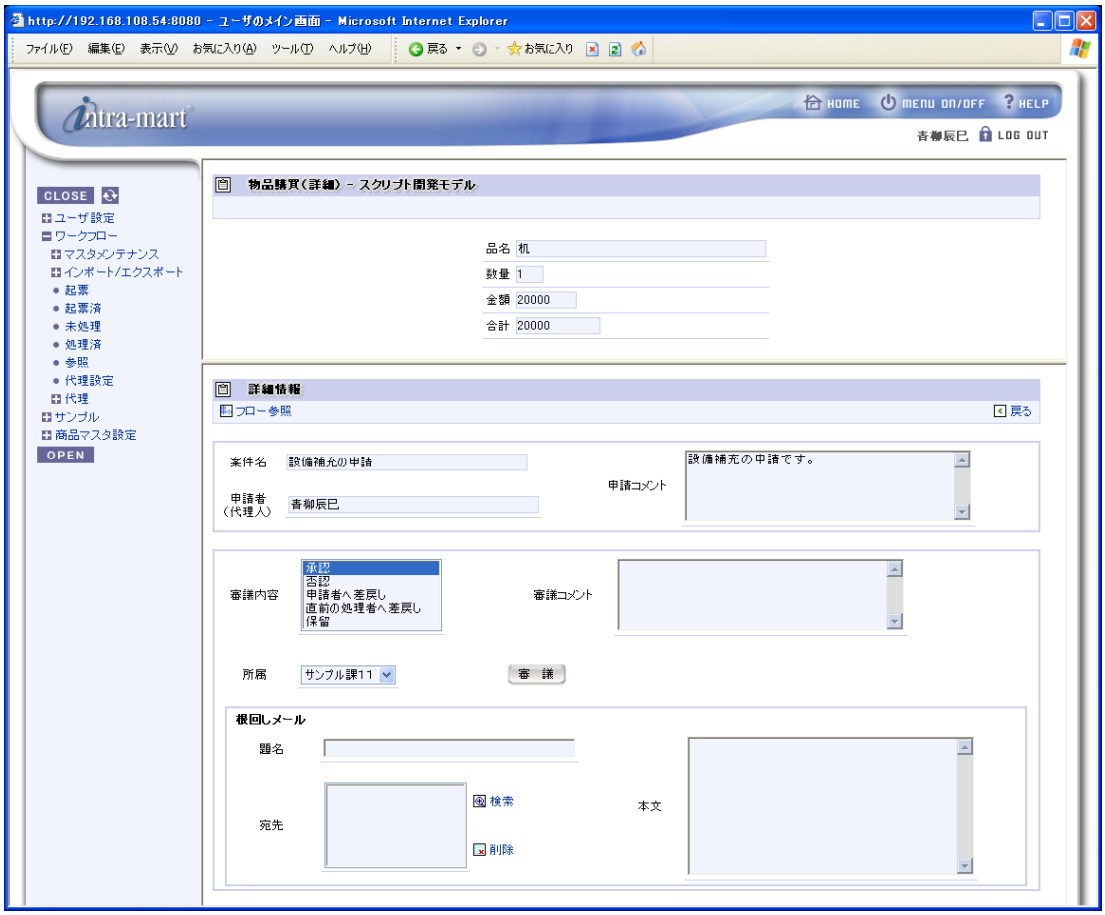
<Workflow Sample Slip Issue>

3.15.3.2 Flow Information Screen



<Workflow Sample Flow Information>

3.15.3.3 **Approval Screen**



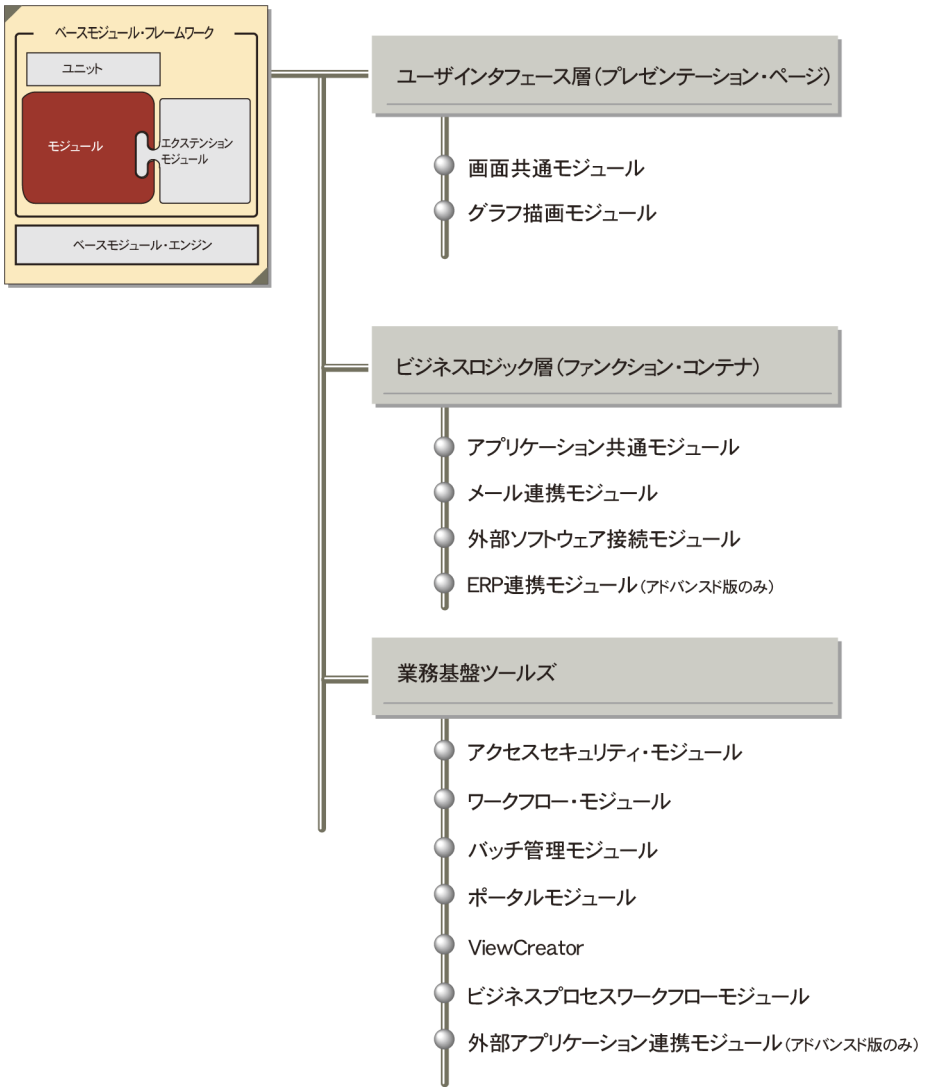
<Workflow Sample Approval

- Sample source is prepared in the folder below.
  %Resource Service%/pages/src/sample/bpw/purchase/standard
- By using Workflow Module, you can create an application corresponding to the workflow easily in intra-mart. Please refer to “Workflow Guide” and “Workflow Specification” about the details of linkage to the Workflow Module and workflow function when creating process (approval root).

3.16 Embedding & Operating the Modules

intra-mart can be divided into “User Interface Tier (Presentation Page)", “Business Logic Tier (Function Container)”, and “Business Fundamental Tool”.

The following are the descriptions of the modules.



3.16.1 User Interface Tier

This section describes the 3 modules that belong to the User Interface Tier.

3.16.1.1 Screen Common Module

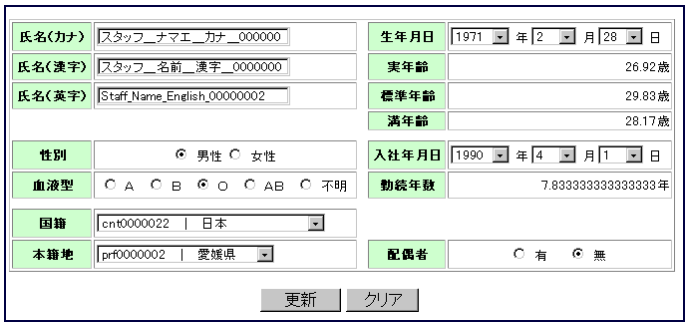
This is a page component module commonly used for a Web-based GUI development. A user interface that is linked up with the database can be easily created by setting the appropriate properties and of the module and invoking the module.

Examples of Provided Page Common Modules

General Input Control Modules	Include general input controls that are necessary for user interface development (such as text field, password box, radio button, check box, and text area, etc). These control groups can interact with server-side scripts and data.
Layout Control Modules	These modules can change the display values or display contents based on specified conditions, i.e. provide a programming element (which usually cannot be represented in HTML) within a presentation page.

Examples of Developed Web User Interfaces

By editing the above-mentioned objects/function groups on HTML, it is possible to develop user interfaces with much finer details, creating a Web system with a user interface equivalent to those created with the conventional VisualBasic. The following is an example of page creation.



<Example of Page Creation>

- Please refer to “Screen Common Module” under “Script Development Model” of API List for the details of Screen Common Modules.
- Please also refer to “3.18 Embedding and Operating Extension Modules” in the chapter 3 “Using Various Component Group (im-BizAPI)” in this book.

3.16.1.2 Graph Drawing Module (Presentation Page)

This module creates a graph image file on the server-side, and display the graph on the browser.
The following 5 types of graphs can be used.

```
Line graph
Bar graph
Pie graph
Radar chart
Portfolio
```

Graph Plotting Settings

Coding the <IMART> tag for drawing graphs in Presentation Page (HTML file). There are 5 <IMART> tags used as shown below.

Type Attribute	Graph
lineGraph	Line graph
barGraph	Bar graph
circleGraph	Pie graph
radarChart	Radar chart
portFolio	Portfolio

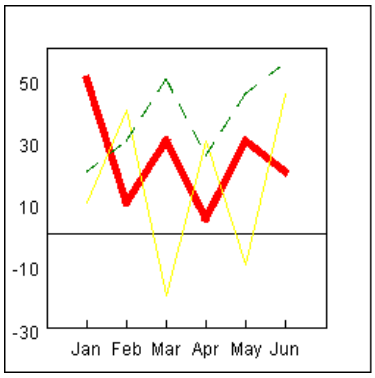
The coding example of <IMART> tag for line graph drawing is as follows.

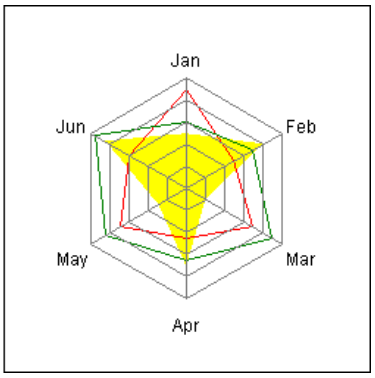
```
<HTML>
<HEAD>
	<TITLE>Line_Graph Sample</TITLE>
</HEAD>
<BODY>
	<IMART type="lineGraph"
		data=oData
		imageWidth="300"
		imageHeight="300"
		dataMin="-30"
		dataMax="60"
		scaleCount="20"
		alt="IM_LineGraph">
	</IMART>
</BODY>
</HTML>
```

Creating Graph Value Object

The Bind variable of attribute Data <IMART type="lineGraph"> tag shown above forms the data value of the graph. The setting of the object value is as shown below.

```
//Declare bind variable
var oData; // Line Graph Drawing Data

// Page Initializing Function
function init() {

        // Create Line Graph Drawing Data Object
        oData.aCaption = new Array ("January", "February", "March", "April", "May", "June");
        oData.aData = new Array();
        oData.aData[0] = new Object();
        oData.aData[0].aData = new Array(50, 10, 30, 5, 30, 20);
        oData.aData[0].sColor = "red";
        oData.aData[0].nWidth = 5;

        oData.aData[1] = new Object();
        oData.aData[1].aData = new Array(10, 40, -20, 30, -10, 45);
        oData.aData[1].sColor = "yellow";

        oData.aData[2] = new Object();
        oData.aData[2].aData = new Array(20, 30, 50, 25, 45, 55);
        oData.aData[2].sColor = "green";
        oData.aData[2].nStyle = new Array(10,10);
}
```





<Examples>

3.16.1.3 Access Controller Module

You can display or hide the content between access controller tags.

- The area surrounded by access controller tags is controlled by access controller.
- Access right can be controlled by roles, organizations, appointments and public groups.
- Users who do not have access rights cannot see the contents of access controller area.

Access Controller Tag

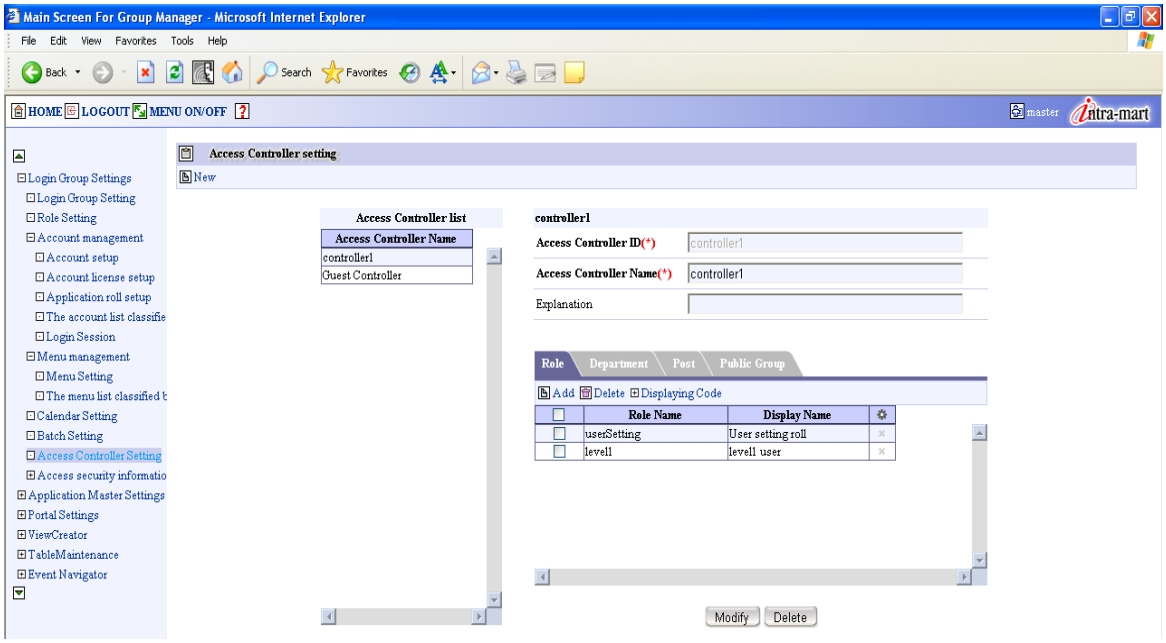
You can control displaying or hiding by coding access controller tags in the presentation page.

```
// Control the display by the access rights configured in access controller ID(controller1).
// If access right information of controller1 does not exist, the contents will not be displayed.
<IMART type="accessCtrl" controller="controller1">
    Contents between access controller tags:controller1.
</IMART>

// Control the display by the access rights configured in access controller ID(controller2).
// If access right information of controller2 does not exist, the contents will not be displayed.
<IMART type="accessCtrl" controller="controller2" defaultShow="true">
    Contents between access controller tags:controller2.
</IMART>

// Control the display by the access rights configured in access controller ID(controller3).
// If access right information of controller3 does not exist, the contents will not be displayed.
<IMART type="accessCtrl" controller="controller3" defaultShow="false">
    Contents between access controller tags:controller3.
</IMART>
```

Access Right Setting of Access Controller

Access Right Setting of Access Controller is implemented in “Access Controller Setting” page in Login Group Administrator.

Configure the authorization for display (role, organization, position, public group) toward each access controller (access controller ID).



3.16.2 Business Logic Tier

This section describes the modules that belong to the Business Logic Tier.

3.16.2.1 Application Common Module

Logic processing modules which are necessary for the development of the various applications are converted into objects for convenience. By embedding and editing these objects into business logics, it is possible to develop a Web system with multi-tier architecture in much shorter period of time, without needing to create the logics again.

Summary of Application Common Module Objects

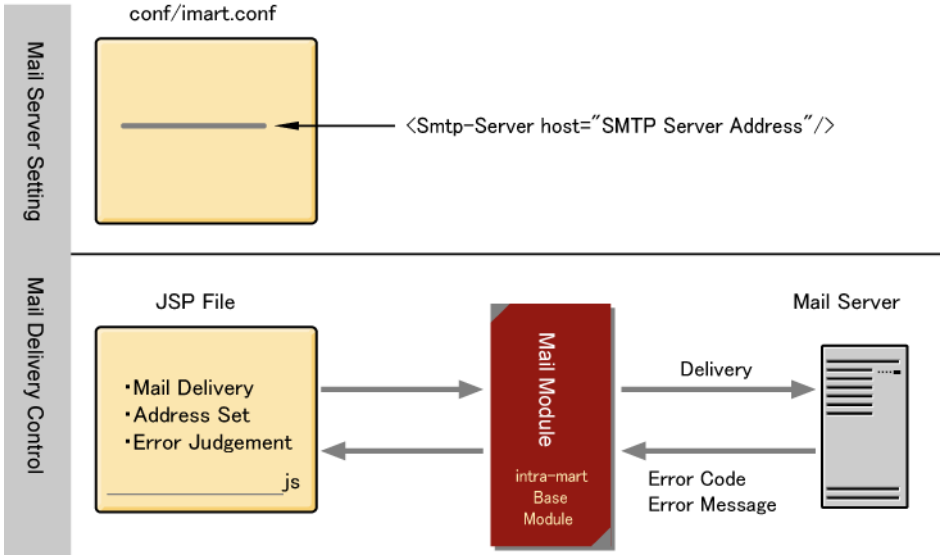
Logic processing modules that are vital for application development (session control or DB access) are converted into objects and provided for convenience. With these objects, it is possible to realize inter-page session control. In addition, footer information such as company name, or login user names to various databases can also be accessed from these objects. By embedding these objects into the application logics and editing them, it is possible to develop a complex Web system in much shorter period of time, without needing to create additional logics. In addition, there are also objects to gain access to various setting values including the application's environment variable, object to access section codes, database-related general object, date-related object, debugging-related object, and URL management objects, etc. By using all these objects, it is possible to access to various types of information, such as the employee codes of those currently logged onto the system, or the name of HTML page they viewed immediately before.

There are also much more advanced functions, such as methods that gain simultaneous access to multiple numbers of DBs, or search streaming that displays a specific number of search results each time when there is a large volume of search data, application lock function, XML-compatible module, etc.

- Please refer to the “Application Common Module” under “Script Development Model” of the API List for the further details of application common modules.

3.16.2.2 E-Mail Related Module (Function Container)

This module facilitates executing mail send process with mail server with SMTP/POP3 compatible mail server.



Mail Server Setting

Mail server setting is done using the “conf/imart.xml” file. The following is the coding example.

```
<Mail Server Setting>
 <smtp-server host="localhost" port="25" mailbox-check="false" />
```

Mail Sending Administration Setting

Use MailSender Object for the mail sending setting. MailSender object consists of the following 11 methods. Mail sending setting is set in the function container using these methods.

```
(1)  setFrom(String address ,String personal)          : Method to Set Mail Sender (From)
(2)  addTo(String address ,String personal)            : Method to Add Addressee (To)
(3)  addCc(String address ,String personal)            : Method to Add Addressee (Cc)
(4)  addBcc(String address ,String personal)           : Method to Add Addressee (Bcc)
(5)  addReplyTo(String replyto)                        : Method to Add Reply Address
(6)  addHeader(String name ,String value)              : Method to Add Mail Header
(7)  setSubject(String subject)                        : Method to Set Mail Subject
(8)  setText(String text)                              : Method to Set Mail Text
(9)  addAttachment(String filename ,String file)       : Method to Add File Attachment
(10) send()                                            : Method to Send Mail
(11) getErrorMessage()                                 : Method to Obtain Mail Sending Error Message
```

Coding example of the above-mentioned method is as follows.

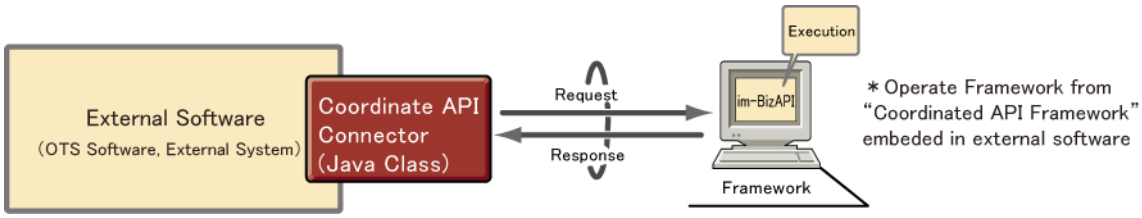
```
<Mail Sending Processing (Function Container)>
        var ret;
        var errorMessage;

        var locale = AccessSecurityManager.getSessionInfo().locale;   // Obtain locale
        var mailSender = new MailSender(locale);     // Create MailSender object

        //----------------------------
        // Configure Send Information
        //----------------------------

        // Destination Mail Address
        mailSender.addTo("mail000@nttdata.co.jp");
        mailSender.addTo("mail001@nttdata.co.jp");
        mailSender.addTo("mail002@nttdata.co.jp");
        mailSender.addTo("mail003@nttdata.co.jp");
        mailSender.addTo("mail004@nttdata.co.jp");

        // Set CC Mail Address
        mailSender.addCc("mail005@nttdata.co.jp");

        // Sender Mail Address
        mailSender.setFrom(request.mail_from);

        //----------------------------
        //Set Mail Subject and Contents
        //----------------------------

        // Set Subject
        mailSender.setSubject("Mail Send Sample");

        // Set Text
        mailSender.setText("This is the mail transmission testing." + "¥n" + "Was the transmission successful?");
```

```
//Sending mail
ret = mailSender.send();

//Error determination
if( ret ) {
        errorMessage = "Error message：" + mailSender.getErrorMessage();
        //Sending mail error
        Module.alert.back( "SYSTEM.ERR", errorMessage);
}
```

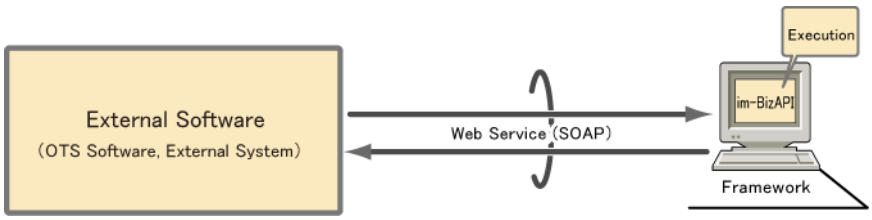
3.16.2.3 External Software Connection Module

This is a module that enables the linkage and connection between intra-mart and external software by directly invoking various API of im-BizAPI from off-the-shelf application packages. There are the following 2 methods to establish the linkage and connection.

Firstly, since the “Interacting API connector” is provided as a Java-based API, user can link any process and im-BizAPI as long as the external software supports Java Runtime Environment. For instance, displaying intra-mart view within a portal by incorporating off-the-shelf portal server products or by interacting with other applications and operating user account information from Java processes running batch operations is enabled.



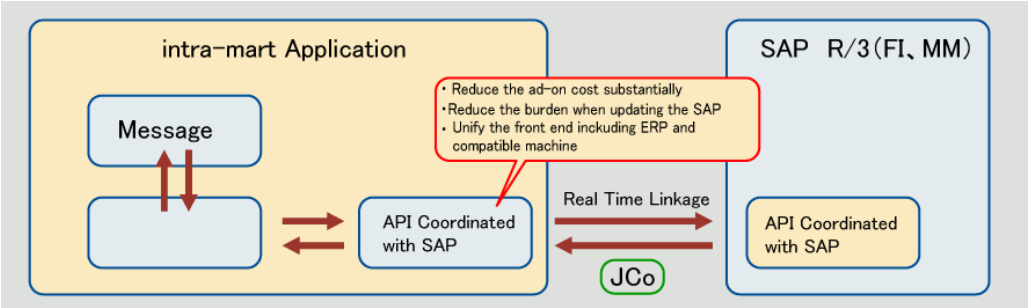
The other way is to call up various API of im-BizAPI from external software via Web service.



- There is also data linkage by XML.　As for the details, please refer to “3.8 Handling XML Data”.
- As for the details on the external software connection modules, please refer to “External Software Connection Modules” under “Developer’s Guide” of API List.

3.16.2.4 ERP Access Module

SAP API has been converted into library using SAP JCo technology. By using standard Java technology, add-ons can be developed at a smaller cost. *This is only for intra-mart WebPlatform/AppFramework Enterprise Version.



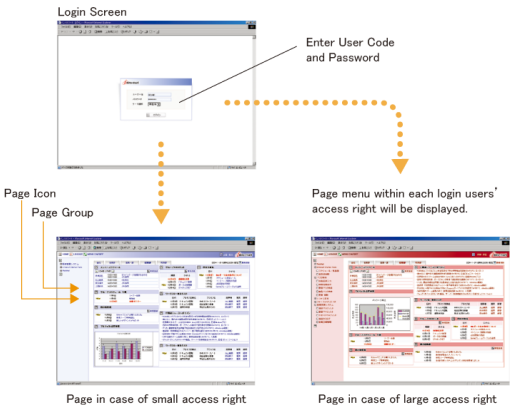
- The EPR Access Module currently supplied comes with various API that can be linked up with SAP. Other ERP Access Modules are scheduled to be added in future.
- Please refer to the following references for detailed information.
  - Tutorial Guide for real-time link up with SAP R3 (attached with the product)
    "ERP Linkage Module Tutorial Guide" (im_sap_api_tutorial_v61.pdf)
  - API List's "Script Development Model" – "Business Logic Tier" – "ERP Access Module"
- SAP, SAP R/3, SAP JCO, and other SAP product/service names written within products are trademarks or registered trademarks of SAP AG in Germany and other countries.

3.16.3 Business Fundamental Tool

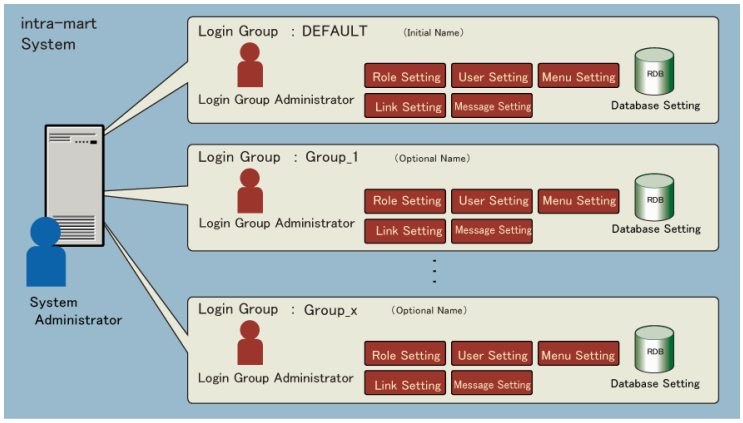
This section describes about the modules related to Business Fundamental Tool.

3.16.3.1 Access Security Module

This module manages the user/role login security information. It has functions to check the login user as well as to display appropriate web page according to user's access right. For example, if a regular employee accesses the system, the menu will not display the pages to which he/she does not have access rights. As a result, the user is not even aware of the existence of such pages. However, when an administrator accesses the system, the same menu page will show all the pages that the manager have access rights. By utilizing the Access Security Module, it is possible to customize different page contents for different login user. As for the setting-up, please refer to "2.5 Access Security Control" in the Administrator Guide.
Other than Access Security Module that comes with Standard Package, there is an "IM-SecureSignOn (optional)" Extension Module which enables single sign-on.



For intra-mart Ver5.0 and newer version, it is possible for each user to switch default language and select the screen theme color. Previously all user information setting has had to be done by the System Administrator as the specification, but Login Group Administrators is assigned under the System Administrator, so that the Login Group Administrators can manage the user information.



In addition, all the Access Security functions are now converted into API, making it possible to create a unique and original menu page. intra-mart can also support a complete Secure Sockets Layer (SSL). With SSL, it is possible to ensure security by encrypting all the contents to be sent.

- In addition to the standard intra-mart Access Security Module introduced above, there is also an optional Extension Module called “IM-SecureSignOn”. “IM-SecureSignOn” uses an independent agent-type reverse proxy method that allows easy to implement single sign-on over a wide scope of application. Please refer to the “3.18. Embedding and Operating the Extension Modules” under the Chapter 3 “Using Various Component Group (im-BizAPI)” for details.
- As for the details of access security, please refer to “Access Security Specification Document” about the details of access security specification.

3.16.3.2 Workflow Module

This module allows the efficient development of a Web browser-based workflow. By simply registering a created application to the document workflow as “Task”, it can be used as an application compatible with the workflow. Please refer to the separate volume “Workflow Guide” for the details of workflow function.

3.16.3.3 Business Process Workflow Module

Unlike document workflow, which centers on applications and approvals (the workflow module of standard package and IM-Workflow Designer, etc.), business process workflow module can automate business processes by registering the business processes in advance. As a result, it is possible to verify the correctness of office processes and improve work efficiency drastically. Please refer to the separate volume “Workflow Guide” on the details of business process workflow function.

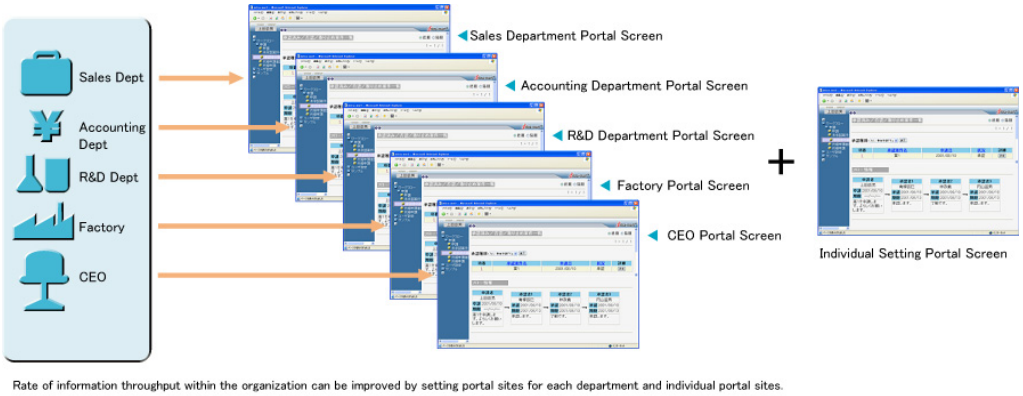
3.16.3.4 Batch Control Module

intra-mart comes with a scheduling function allowing for program execution scheduling by the schedule server. At first, a batch program that contains logic to be batch-executed is created, and then set the execution date on the batch setting page.

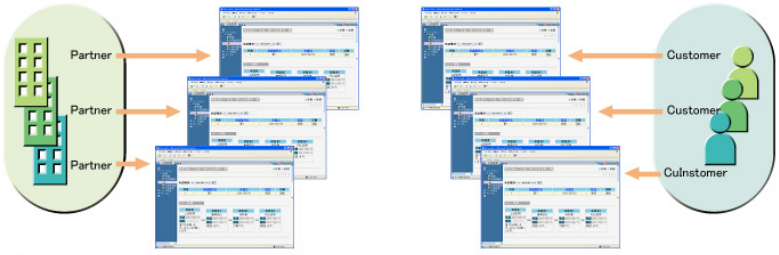
For the details on batch control, please refer to “2.11 Batch Control Operations” in Administrator Guide.

3.16.3.5 Portal Module

This module displays a preset page (called “portlet” in portal applications) as the login default page. By displaying a list of commonly used application and prompter pages for user at the initial default page would improve the work efficiency. Various pages created with the applications can be registered as portlets, and users can freely design and layout these portlets at the portal screen. There is no restriction on the number of portal screen combinations; user can use the tabs to switch the portal screens. By allowing the switching of portal screens for different organization, role, or user, the system can provide the most suitable page to support the business operations.

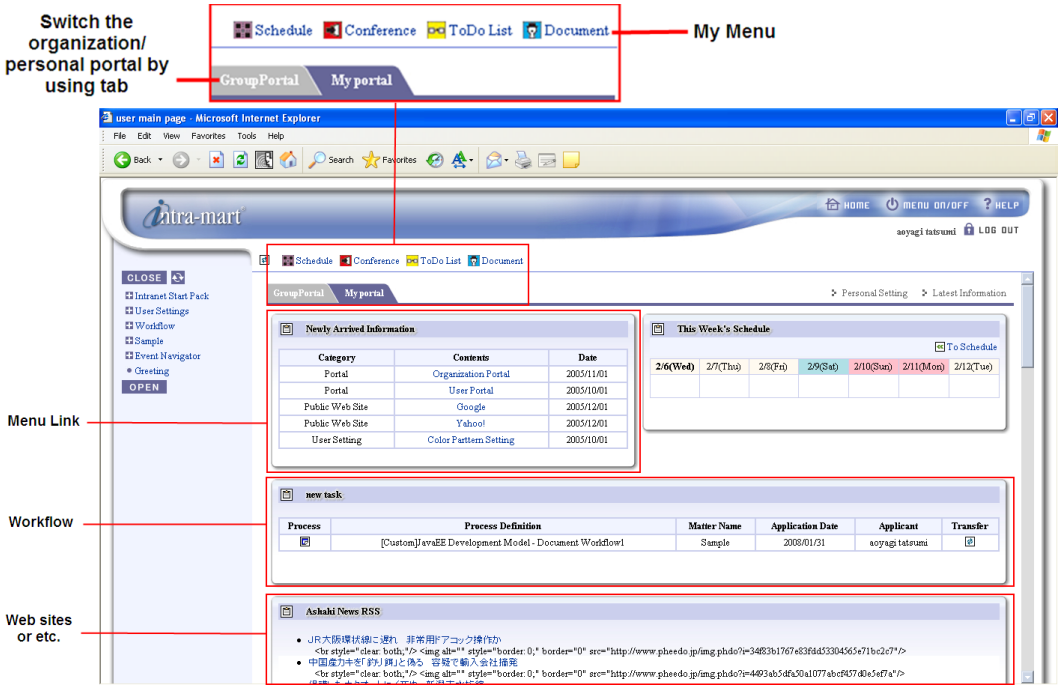


Rate of information throughput within the organization can be improved by setting portal sites for each department and individual portal sites.



Service level in EC site and market place can be greatly improved through setting portal site for each company (customer/partner).

<Portals for Company/Organization, User>



Creating Portlet Page

Create a portlet page to be displayed on the portal screen of Script Development Model, JavaEE Development Model and External URL.　intra-mart is pre-installed with sample portlet pages in advance.

3.16.3.6 ViewCreator

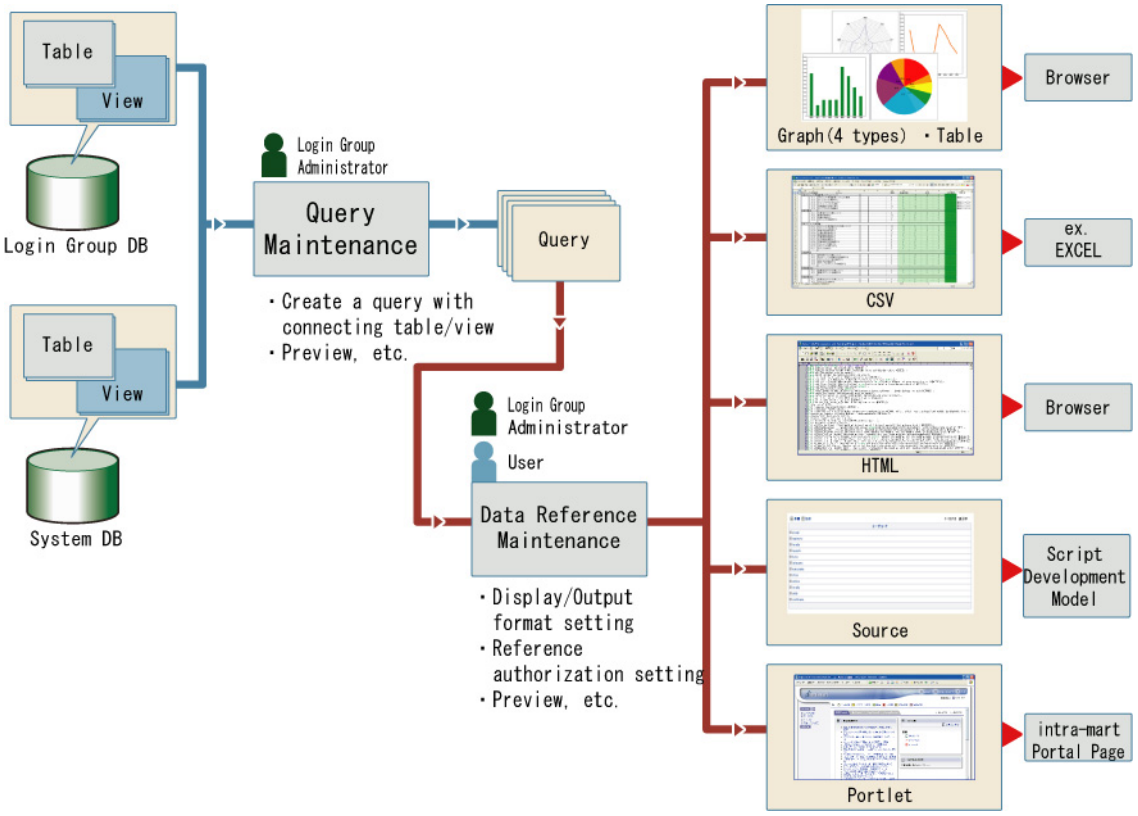
ViewCreator is a tool to create various tables or charts by using data in database from the pages of the intra-mart. Usable database are login group database and system database.　ViewCreator has two main functions; query maintenance and data reference maintenance.

Query Maintenance	This combines the table and view on the database and creates the table which is referred by the data reference maintenance.　In addition, this can display the created query and preview of the table, and create a view.
Data Reference Maintenance	This configures the way to display the data created in the query maintenance (chart or table) and to refine data.　In addition, this can also configure the reference authorization by each data reference.

In terms of created data reference, you can search data, refine the displayed items and change the listing order in the screen display.　In addition, you can add data reference as a portlet, output it as a CSV/HTML file or a program file of script development model.　Output program source can be freely customized and reused.

Thus, it is an important feature of ViewCreator to easily operate creation/display of the various tables and chart based on the database data on the Web browser.

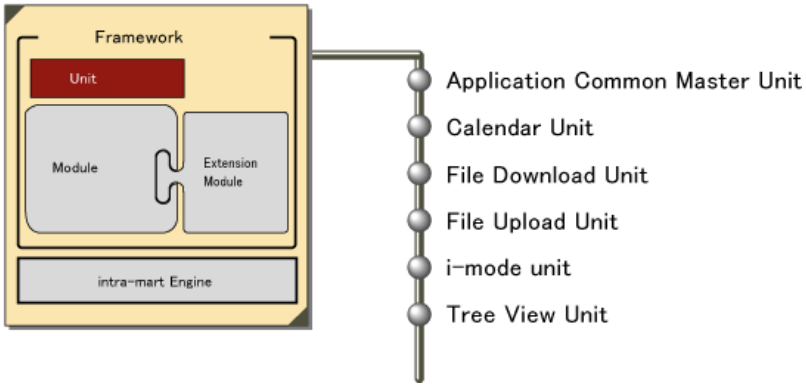
For the details of the operation of ViewCreator, please refer to the Administrator “2.13 ViewCreator”.



<ViewCreator>

3.17 Embedding & Operating the Units

intra-mart WebPlatform/AppFramework comes with units as reusable software parts. You can use them by simply embedding them into user applications. In addition, they can be customized freely as the source codes are open and public.



3.17.1 Application Common Master Unit

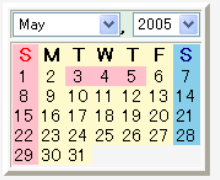
Commonly used masters for system development, such as company data, organization data, group data, customer data, client data, and product data comes together with the standard package. It is possible to design and develop a system in a short period of time by utilizing these masters. There are default APIs that helps to develop systems that can be linked with different intra-mart application series, and also the accessing of the masters.

For details, please refer to “2.8 Application Common Maste Unit” in Administrator Guide and the supplementary volume “Reference for Application Common Master”. For the details of APIs to access, please refer to [Unit]-[Application Common MAP Specifications] page in the API List.

- intra-mart Ver5.0 onwards the archive management of the master can be carried out by the date stamp.

3.17.2 Calendar Unit

By embedding a calendar unit, it is possible to display a calendar page that is linked with the data set on the Calendar Maintenance page. Date input with due consideration of the company’s non-working and working days can be done on the Calendar page.



<Calendar Unit>

3.17.2.1 Calling the Calendar Unit

Calendar unit will be called by linking the following keywords.

- @IM_CALENDAR_VIEW : Display format is auto-selected by the module
- @IM_CALENDAR_VIEW_COMPACT : Calendar in compact display format

```
<Coding Sample>
    <IMART type="link" page="@IM_CALENDAR_VIEW"
    year="1999"
    month="5">
    </IMART>
```

Other than specifying the links as shown in above coding example, src-attribute of <IMART type="frame"> or page-attribute of <IMART type="form"> / <IMART type="submit"> can also be specified in the same manner. Additionally, specifying the following option attribute can also define the calendar view operations.

Option Attributes

year (Required)	Year to display
month (Required)	Month to display
past (Optional)	Display amount of select-year-combo box (past) (years from current to the past can be selected in the combo)
future (Optional)	Display amount of select-year-combo box (future) (years from current to the future can be selected in the combo)
display (Optional)	Window name or frame name which calls the page when a day is clicked.
link (Optional)	Page path called when a day is clicked. (Relative path from program directory in the Resource Service – %Resource Service%/pages/src by default)

3.17.2.2 Receiving Calendar Data

When a day on the calendar is clicked, the page specified with the link option will be automatically called. The Function Container of this page can retrieve information which user clicked (selected). Day information which user clicked can be retrieved via request object.

```
<Coding Examples>
    var sGroup = request.group;;    // Calendar ID
    var sYear = request.dtyear;     // Year required
    var sMonth = request.dtmon;     // Month required
    var sDate = request.dtday;      // Day required
```

3.17.2.3 Calendar Extension Tag and Calendar Module

Calendar Extension <IMART> tag

Using extension tag <IMART type="calendar"> enables flexible creation of calendar view. For details, please refer to “API List”.

Calendar Module

```
CalendarManager.*
```

Using methods contained in above object enables invoking and utilizing calendar settings in the program. For details, please refer to “API List”.

- For details on calendar master maintenance, please refer to the Administrator Guide.

3.17.3 File Download Unit

Embedding the File Download Unit enables downloading files from server via the web browser to client’s PC. The downloading uses HTTP protocol.

Files can be managed in an integrated fashion by using the Storage Service.

3.17.3.1 How to Download

Data will be sent to client by using File Download API (Module.download.*) in the Function Container.

3.17.3.2 File Extension and MIME Type

MIME type is automatically determined based on the file extension during downloads.

The MIME type can also be specified by the program side using Download API using variables passed on to the Download API.

- File Download Unit is a function to send data to the browser. To download file from server, the file needs to be loaded first then call this unit.
- Download API does not encode characters when sending data to the client. Program side using the Download API must encode the data suitably before passing it to the Download API.
- For details on the coding method using File Download Unit, please refer to ”3.1 How to Use the Storage Service” in this book.

3.17.4 File Upload Unit

Embedding the File Upload Unit would enable the uploading of file from client’s PC to the server via web browser. The HTTP protocol is used for uploading.

Files can be centrally administered using the Storage Service.

3.17.4.1 Linkages with Presentation Page

In the presentation page, forms are created in the following manner.

```
<IMART type="form" method="POST" enctype="multipart/form-data">
        <INPUT type="file">
        <INPUT type="submit">
</IMART>
```

3.17.4.2 Retrieving Information

Form contents can be retrieved using the request object in Function Container, in a same way as links and forms.

- File Upload Unit is a function to retrieve uploaded data from the browser. To save file on the server, a different API is required.
- For the coding method to use the File Upload Unit, please refer to “3.1 How to Use the Storage Service” in this book.

3.17.5 Tree View Unit

By embedding Tree View Unit, the hierarchized data can be displayed in tree view, making it easier to understand the tiered structure and select from the menu. It is used to display organization tree view of [Page Setting] page under the [Page] menu. For the detailed information on Tree View Unit, please refer to [Script Development Model]-[User Interface Tier]-[Tree View Module] in API List.

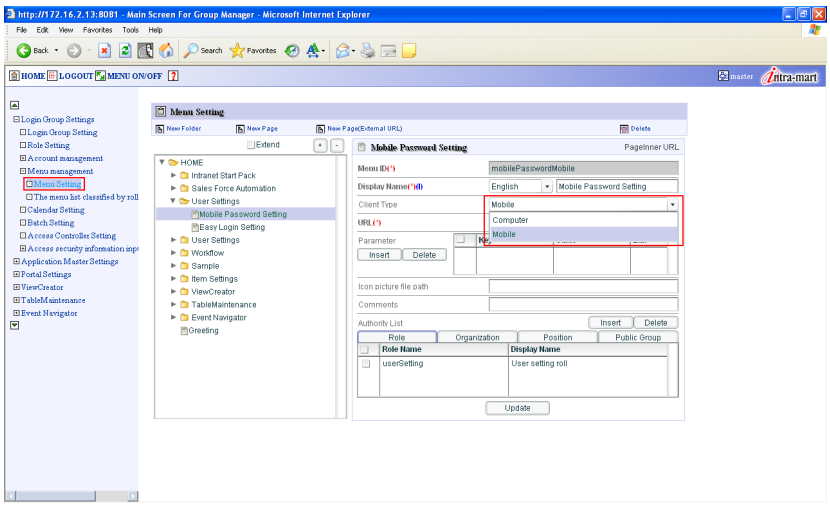
<Tree View Unit>

3.17.6 i-mode Unit

Embedding i-mode unit will enable pages created by user and registered in the Login Group Administrator's [Login Group Setting]-[Menu management]-[Menu Settings] to be viewed with i-mode mobile phones.
To create pages for i-mode, create the pages according to i-mode display size. When registering in [Login Group Setting]-[Menu management]- [Menu Settings], select [Computer] or [Mobile] for the client type. Also, several items must be preset in [Login Settings] – [Mobile Password].
For the details on settings, please refer to "i-mode Settings in Page Management Master Maintenance". For the APIs to use, please refer to "Module.mobile" in API List.

3.17.6.1 i-mode Settings in Page Management Master

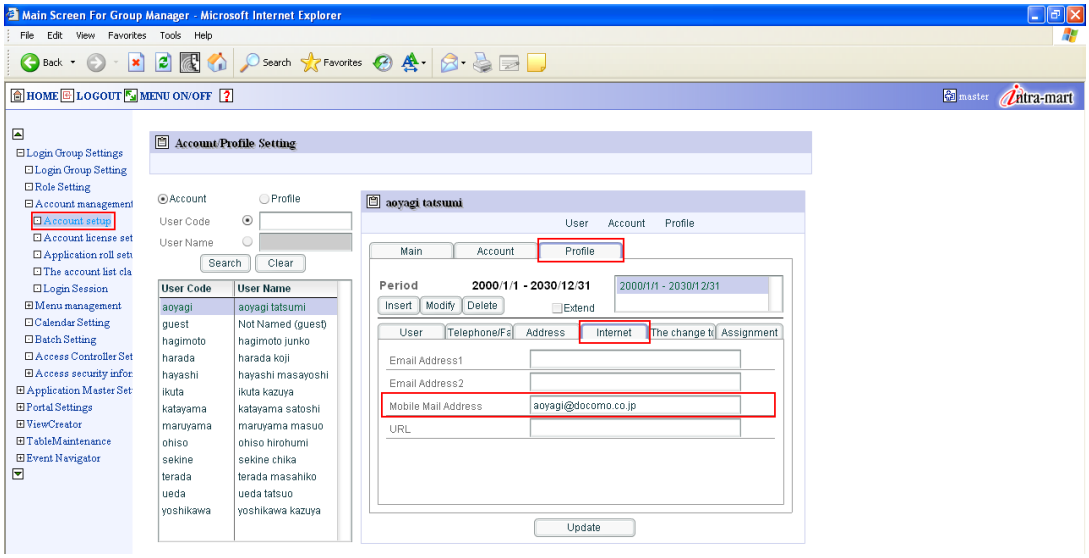
When a Login Group Administrator registers i-mode compatible page under [Login group Setting]-[Menu management]-[Menu Settings], selects "Mobile" in the "Client Type" option. As for the operation of [Menu Settings], please refer to"2.7 Application Registration" in the Administrator Guide.



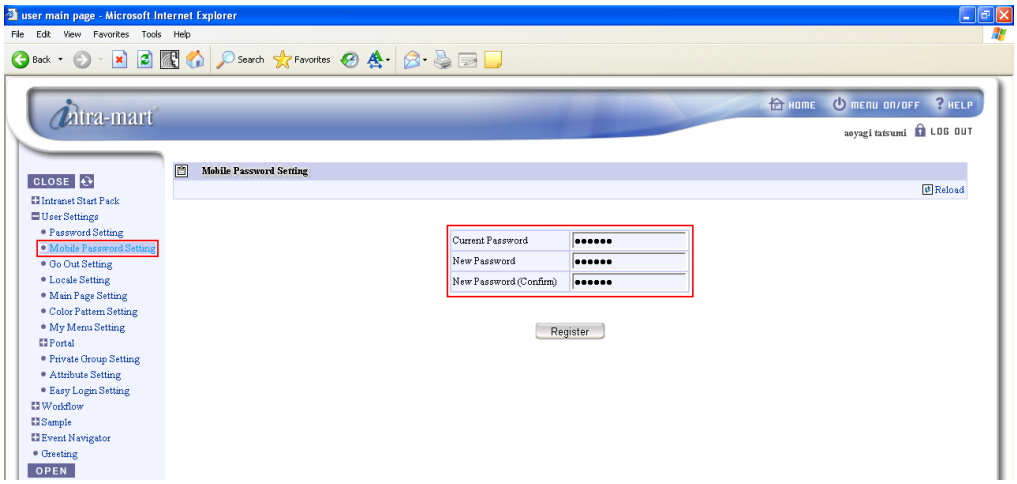
<[Login Group Settings] –[Menu management] – [Menu Settings]>

3.17.6.2 Setting i-mode Address and Password

In order to use the i-mode-compatible pages on an i-mode mobile phone, the Login Group Administrator must set a mobile e-mail address and mobile password for each user under the [Login Group Settings] – [Account Settings]. The individual user can then log in to change the default mobile password under [User Settings] – [Mobile Password Setting].



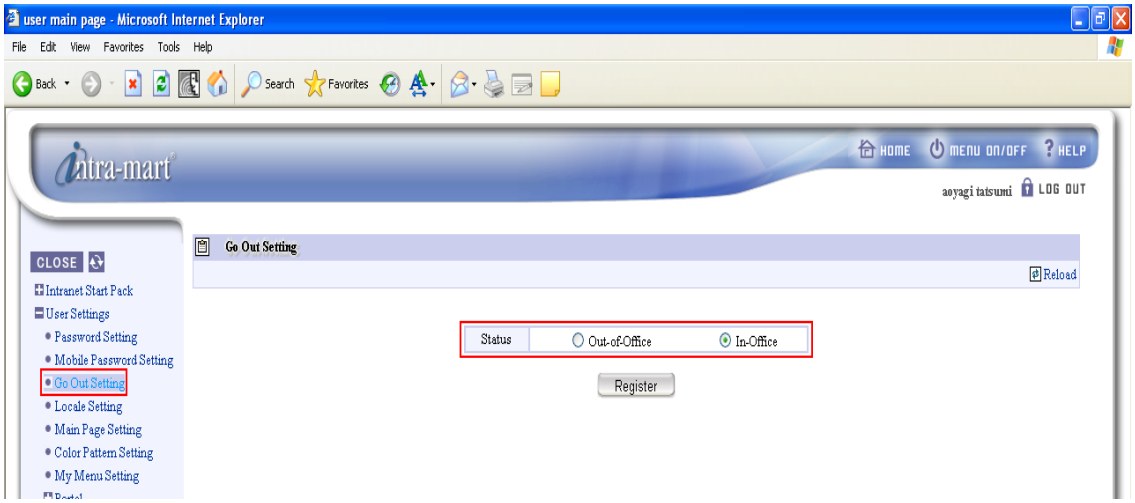
< [Login Group Settings]-[Account Management] – [Account Setup] of the Login Group Administrator>



<[Mobile Password Settings] under [User Settings] folder>

3.17.6.3 Go Out Settings for i-mode

All the e-mails sent from intra-mart are usually sent to the e-mail addresses registered under [Login Group Settings] – [Account Management]-[Account Setup]. If a user sets an [Out-of-Office Flag] setting in [User Settings] – [Go Out Setting], all the e-mails will be sent to the i-mode e-mail address of the user in addition to his/her regular registered e-mail address. The setting can be done under the user's [User Settings] – [Go Out Setting]. However, in order to utilize this function, the Login Group Administrator must have first setup the [Mobile Address] under [Login Group Settings] – [Account Management]-[Account Setup].



<Go Out setting>

- When a warning page needs to be used for i-mode, use "Module.mobile.alert()". Please refer to the "API List" for details.

3.18 Embedding & Operating the Extension Modules

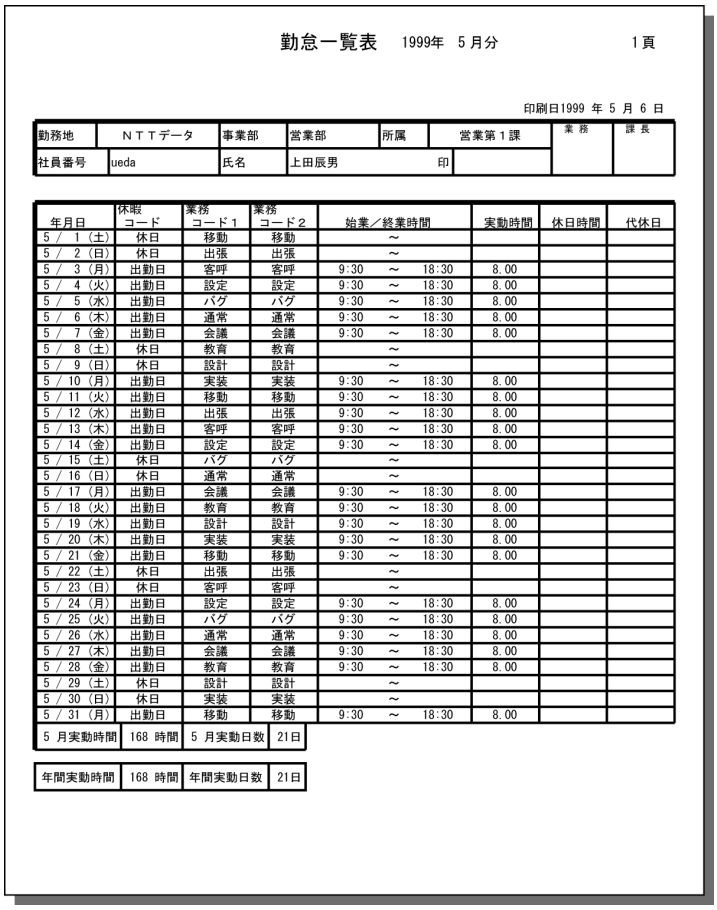
Extension modules are a set of module groups that are available separately from the standard modules of intra-mart WebPlatform/AppFramework. These extension modules are meant for the users who need much more advanced modules, which can be embedded as required just like the standard ones. This section describes the extension modules currently available.

3.18.1 Printing Form Module Extension

There are optional extension modules that can carry out more detailed printing or mass printings.

3.18.1.1 IM-PDF Designer

This is a module that can handle more complex form format by using PDF, and can also handle single form as well as multiple forms. Visual form designing tool “IOWebDoc” is used to create the form formats. It creates a PDF file based on the data and layout definition file from a user application, and then activates Acrobat to print it out.



勤怠一覧表　1999年　5月分　1頁

印刷日1999 年 5 月 6 日

勤務地	ＮＴＴデータ	事業部	営業部	所属	営業第１課	業務	課長
社員番号	ueda	氏名	上田辰男	印			

年月日	休暇コード	業務コード１	業務コード２	始業／終業時間	実動時間	休日時間	代休日
5 / 1 (土)	休日	移動	移動	～			
5 / 2 (日)	休日	出張	出張	～			
5 / 3 (月)	出勤日	客呼	客呼	9:30 ～ 18:30	8.00		
5 / 4 (火)	出勤日	設定	設定	9:30 ～ 18:30	8.00		
5 / 5 (水)	出勤日	バグ	バグ	9:30 ～ 18:30	8.00		
5 / 6 (木)	出勤日	通常	通常	9:30 ～ 18:30	8.00		
5 / 7 (金)	出勤日	会議	会議	9:30 ～ 18:30	8.00		
5 / 8 (土)	休日	教育	教育	～			
5 / 9 (日)	休日	設計	設計	～			
5 / 10 (月)	出勤日	実装	実装	9:30 ～ 18:30	8.00		
5 / 11 (火)	出勤日	移動	移動	9:30 ～ 18:30	8.00		
5 / 12 (水)	出勤日	出張	出張	9:30 ～ 18:30	8.00		
5 / 13 (木)	出勤日	客呼	客呼	9:30 ～ 18:30	8.00		
5 / 14 (金)	出勤日	設定	設定	9:30 ～ 18:30	8.00		
5 / 15 (土)	休日	バグ	バグ	～			
5 / 16 (日)	休日	通常	通常	～			
5 / 17 (月)	出勤日	会議	会議	9:30 ～ 18:30	8.00		
5 / 18 (火)	出勤日	教育	教育	9:30 ～ 18:30	8.00		
5 / 19 (水)	出勤日	設計	設計	9:30 ～ 18:30	8.00		
5 / 20 (木)	出勤日	実装	実装	9:30 ～ 18:30	8.00		
5 / 21 (金)	出勤日	移動	移動	9:30 ～ 18:30	8.00		
5 / 22 (土)	休日	出張	出張	～			
5 / 23 (日)	休日	客呼	客呼	～			
5 / 24 (月)	出勤日	設定	設定	9:30 ～ 18:30	8.00		
5 / 25 (火)	出勤日	バグ	バグ	9:30 ～ 18:30	8.00		
5 / 26 (水)	出勤日	通常	通常	9:30 ～ 18:30	8.00		
5 / 27 (木)	出勤日	会議	会議	9:30 ～ 18:30	8.00		
5 / 28 (金)	出勤日	教育	教育	9:30 ～ 18:30	8.00		
5 / 29 (土)	休日	設計	設計	～			
5 / 30 (日)	休日	実装	実装	～			
5 / 31 (月)	出勤日	移動	移動	9:30 ～ 18:30	8.00		

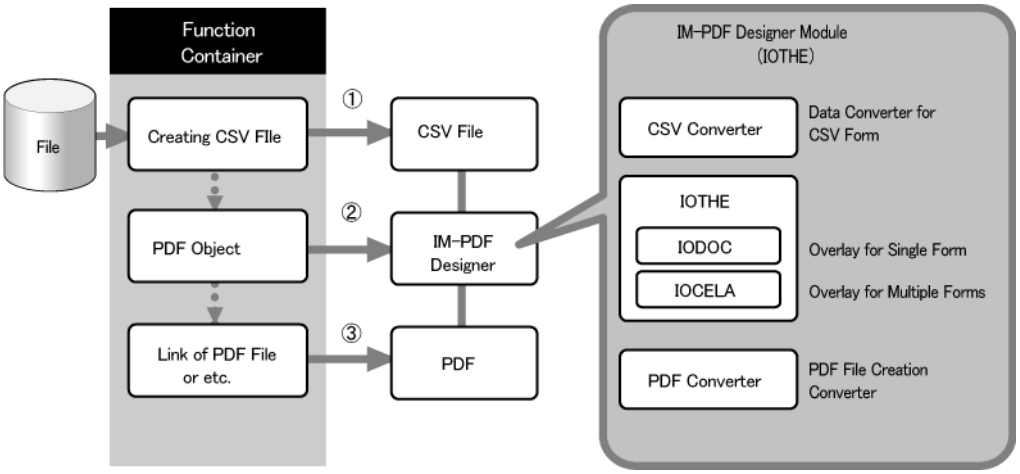
5 月実動時間	168 時間	5 月実動日数	21日

年間実動時間	168 時間	年間実動日数	21日

<Sample Page of IM-PDF Designer>

Operation Overview of IM-PDF Designer

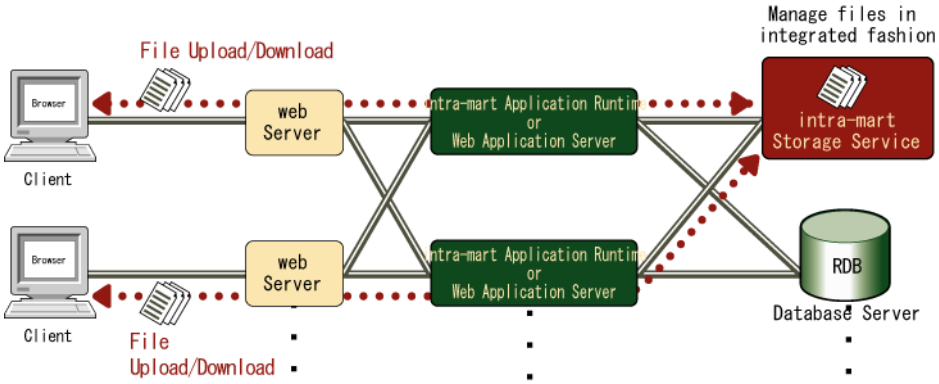
Embed the IM-PDF Designer and operate it as one of the intra-mart modules. Detailed, complicated forms can be created into PDF by first creating the form data in CSV file format from the intra-mart application function container and using the PDF objects for conversion.



- Create form design with IM-PDF Designer (IOWebDoc).
- For the details on the Control File and PDF Object of IM-PDF Designer, please refer to “API List” and supplementary online manual of IM-PDF Designer.

Using IM-PDF Designer in Multiple Application Server Environment

To use IM-PDF Designer in an environment where multiple numbers of applications servers are installed, IM-PDF Designer must be activated on Storage Service. This way, you can manage all the created PDF files by the Storage Service in the integrated fashion.



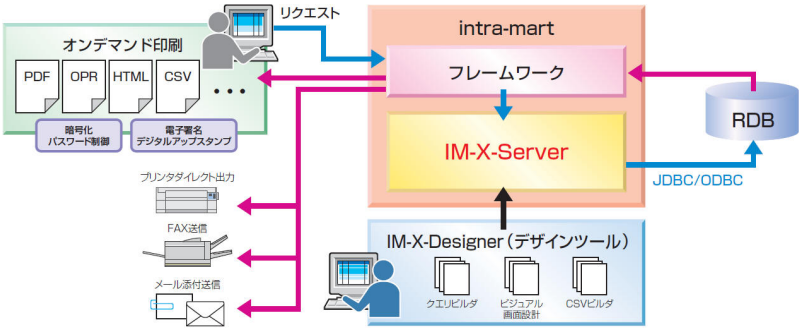
<Using IM-PDF Designer in a multiple application server environment>

3.18.1.2 IM-X Server

This is a printing module to output a lot of ledger sheets, directly output to the printer, and support electronic signature and time stamp. This provides diverse functions such as on-demand printing or direct output with advanced business ledger sheet solution corresponding to XML.

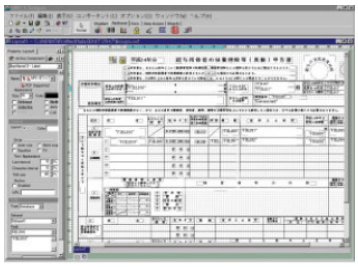
Creating, outputting and delivering various types of ledger sheets

Since creation of ledger sheets by IM-X-Server is defined as XML, you can reduce the development processes with making computerization (PDF, HTML, CSV, CPR), printing (direct printing, FAX delivering), searching and input form in one source multi-format. This provides a total solution to create not only main ledger sheets but also business reports required to daily work.

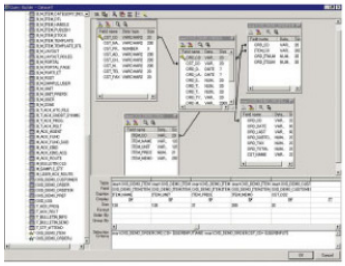


Also providing a design tool enabling to create a ledger sheet in detail

This corresponds to Japan-specific complicated design such as ruled line, report per page, consolidate or single ledger sheets, sub-report, label, customer size or etc. This also supports various business ledger sheets in flexible expression such as creating dynamic graph, electric signature, table calculation, column setting by grouping multiple objects or etc.



[Page Design]
Intuitively operate the deployment of the GUI parts and property input.
Need not to make setting by using each wizard.



[Query Builder]
Make configuration without considering database type.
Obtain DB, CSV and XML in GUI.

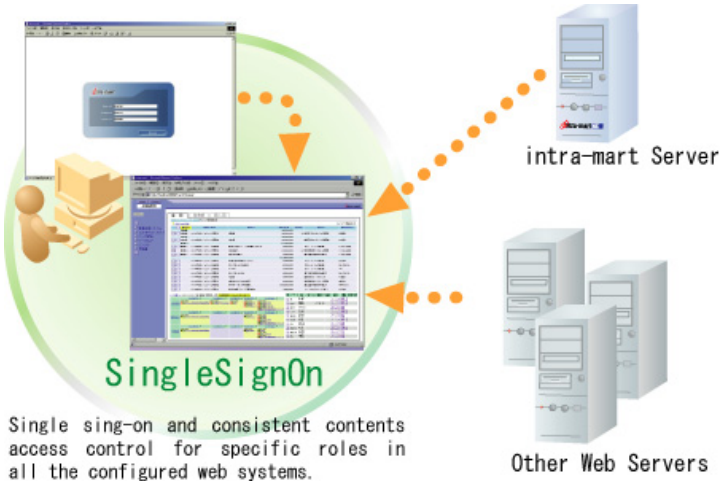
[Support Access Control]
Support PDF sample/Document Security and electronic signaturePDF.
On-demand password.

3.18.2 Access Security Module Extension

In addition to the standard Access Security Module, there is an optional Extension Modules to enable single sign-on.

3.18.2.1 IM-SecureSignOn (Secure Sign On)

IM-SecureSignOn is a tool that enables single sign-on. All the logins to various types of Web systems in the company can be achieved with a single login to this SecureSignOn. This system is based on a original agent-type reverse proxy method and is easy to introduce and implement with a wide scope of application.

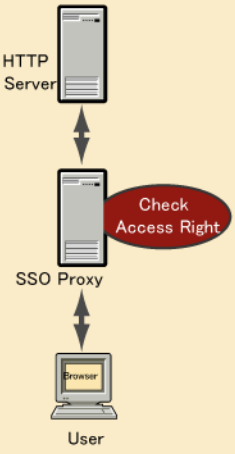
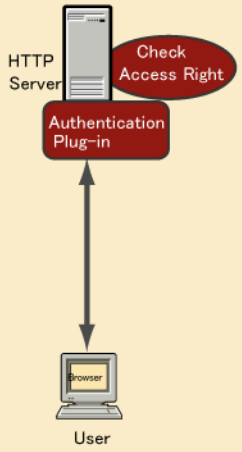
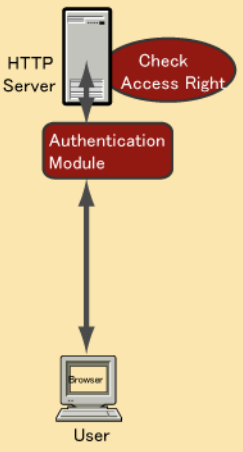


Characteristics of IM-SecureSignOn

- It is possible to use in a large-scale, multi-environment condition.
- The agent installation-type reverse proxy method has both characteristics of agent plug-in and reverse proxy method, such as “not requiring to have client setting”, “possible to implement by stages in a large-scale environment”, and “compatible with any Web servers or OS”.
- It is possible to implement in incremental stages when multiple number of Web servers are already in operation. (Not necessary to have a proxy server where all accesses consolidate at)
- ACL carries out distributed management over the servers (central control unction is currently under construction).
- It is compatible with almost all Web servers.
- The operation of the authentication module has been verified on WindowsNT, Solaris and Linux.
- The name, e-mail address, and department information can be retrieved using CGI.
- Backend User Database supports LDAP and NT domains. Possible to support proprietary data using plug-ins.
- It is not possible to tamper access ticket due to the use of electronic signature.

3 Methods to Achieve Single Sign On

Usually the reverse proxy method and the agent module method are used to achieve single sign-on. IM-SecureSignOn uses a proprietary agent installation-type reverse proxy method that has the advantages of both methods mentioned above.

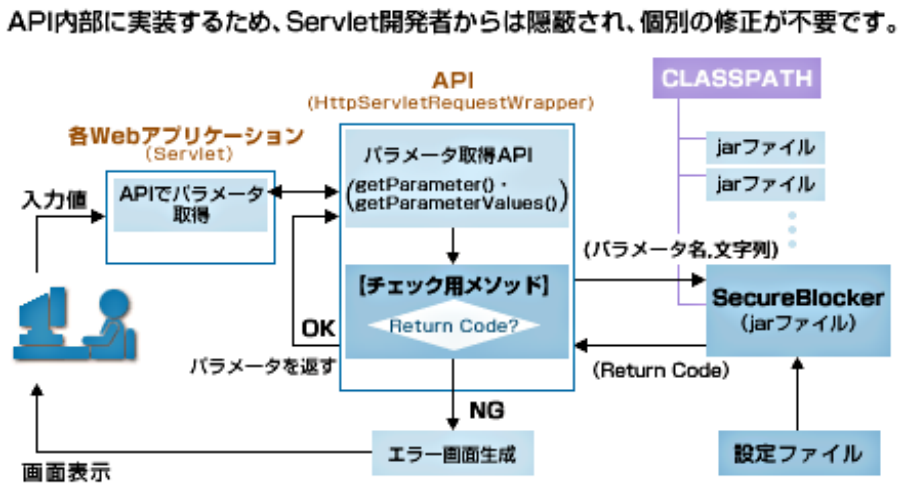
Method	Principle Use	Advantage	Disadvantage	System Conceptual Diagram
Reverse proxy method	Method of single sign-on suitable for internet and ASP, etc.	·The security level is high. ·No agent is needed. ·No restrictions on the Web server ·It is easy to apply to the ASP service.	·The load concentrates on the authentication sever. ·All the accesses pass the authentication server. ·Web browser has to be configured. ·Unstable to a large-scale user. ·Unstable to the distributed environment of Web server.	
Agent module method	Method of single sign-on suitable for intranet of management concentration type	·The load does not concentrate. ·Each agent receives a direct access. ·It is possible to deal with large-sclae users ·Multiple domains are easily corresponded.	·It is necessary to install the authentication module on each web server. ·There are some restrictions on the eeb server.	
Agent type reverse method	Method of single sign-on suitable for intranet managing distributed	·The load does not concentrate. ·Each agent receives a direct access. ·It is possible to deal with large-sclae users ·It is possible to correspond to the distributed environment of the Web server. ·The type of web server and host OS do not matter.	··It is necessary to install the authentication module on each web server.	

- For details please refer to the manual that comes with IM-SecureSignOn.

3.18.2.2 IM-SecureBlocker

This can realize security measures of Web applications opened in internet with low cost. You can apply this to vulnerable points to minimize the influence by the application to the whole Web application.

SecureBlocker is a Java class library to inspect input parameters and automatically render it harmless.



The followings are the advantages by introducing IM-SecureBlocker.

- Reduces verification and implementation costs for checking function of the parameters' input values.
- Implements vulnerability countermeasures of Web applications regardless of the developers' skill.
- Controls checking each parameter used in Web site.
- Check cross-site scripting, OS command injection, directory traversal and SQL injection.

3.18.3 Workflow Module Extension

In addition to the standard workflow module, there are 3 optional extension modules, the “IM-Workflow Designer (optional)”, “IM-FormatCreator (optional)”, and “IM-ΣServ (optional)”.

- Business Process Workflow is bundled with Advanced Version.

3.18.3.1 IM-Workflow Designer

This is a workflow function that can accomodate complicated conditions, such as parallel processes, fusions, or with conditional branches. For fusing or branching processes, they can be configured and executed as such.

- Please refer to the supplementary volume of “Workflow Guide” for the details on IM-Workflow Designer.

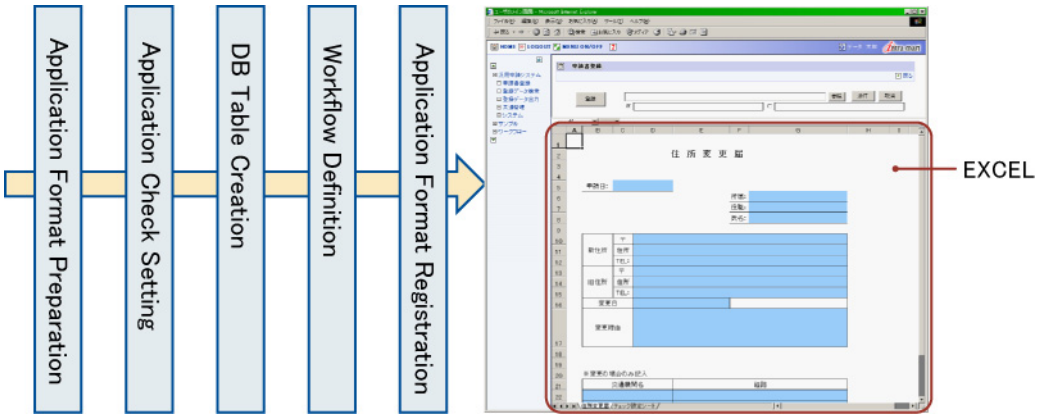
3.18.3.2 IM-FormatCreator

IM-Format Creator is a solution that automatically generates application forms for electronic application. The applications can be easily generated by following the built-in generation wizard, so you need not to have knowledge about Web page creation language such as HTML, JavaScript or XML and database. The created application documents can be easily associated with intra-mart workflow in the special screen page. In addition, it can efficiently issue a ledger sheet by re-using the previous ledger sheets.

- Please refer to “IM-FormatCreator” on details.

3.18.3.3 IM-EX Application System

This is a solution to use Excel sheets into application format page for electric application. By using this together with intra-mart workflow, you can create application workflow with only configuration but without programming. You can use document format or macro of Excel functions to check the input. It is possible to save the input information in the specified table in the database.



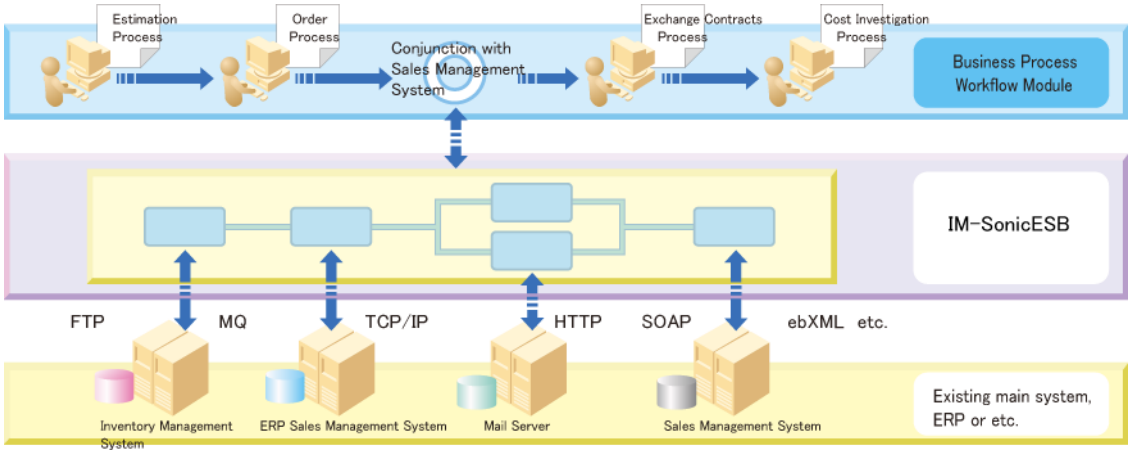
- You can make the application document into web resource only by preparing Excel format.
- You can coordinate this with intra-mart workflow by utilizing the used Excel format as input page.
- You can configure check the input from the client side by setting Excel format or provided macros.
- Information input to Excel can be saved in the specified table in the database.
- You can output the information saved in the database in Excel format.

You do not need knowledge about programming language for registering Excel format or input setting.

3.18.3.4 IM-SonicESB

IM-SonicESB is a back-end system integrated platform (ESB) linking up multiple numbers of existing systems. It also comes with functions that guarantee the consistency of data between systems, making the development of high-reliability application linked up between the Web services easily.

It can realize dynamic and mission-critical system integration with inclusive of the backend by linking up with the Business Process Workflow Module.



3.18.4 Solution to Link Up with External Software

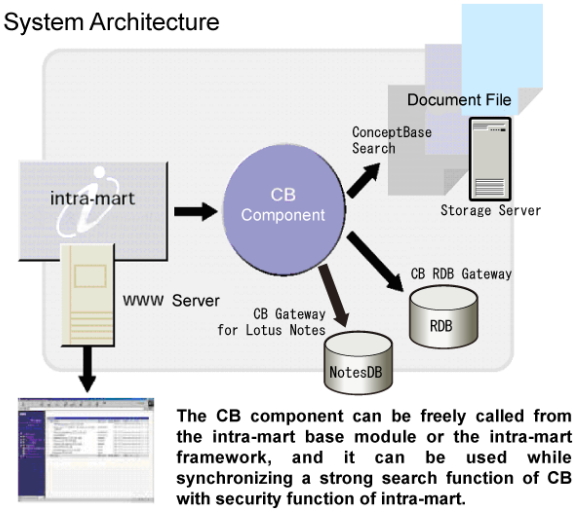
intra-mart comes with a solution to link up with external software applied by other companies.

3.18.4.1 Integrated Search Solution

IM-Integrated Search Solution is a document search system that can do knowledge search using natural language on various file types such as MS-Word, Ichitaro, MS-Excel, and PDF, a result of cooperation with “ConceptBase” by Justsystem Corporation.

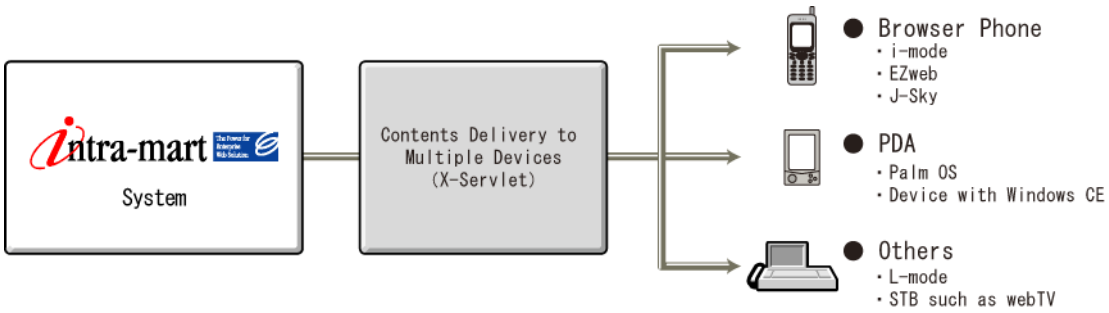
By providing a linkage module between ConceptBase server and Storage Service, it is possible to develop a system that utilizes the powerful document search function of ConceptBase. In addition, it is also possible to retrieved document summary and content of the document search in text format.

Concept Base Search can handle document file search, while CB RDB Gateway can handle RDB-related information search. Finally, CB Gateway for Lotus Notes can handle Notes DB search.



3.18.4.2 Multi Device Solutions

Using X-Servlet (Flexfirm) enables contents distribution to multi devices. Please refer to the attached manual of X-Servlet for details.



Main Features

Single URL	When a URL is accessed, automatically recognize the accessed system model.
Single Source	One code accords to access methods of all domestic mobile communication companies.
Auto Layout	According to the accessed model, automatically convert images or divide pages.

3.19 Other Functions

intra-mart WebPlatform/AppFramework comes with the following functions.

3.19.1 Library

There are some public functions for users that are outside the intra-mart WebPlatform/AppFramework. User can gain access to the public source codes and customize these functions. These public functions can be called with the [Procedure.method-name] format from the Function Container. For details, please refer to “Library” in the API List. Please note that these functions are available only in Script Development Model only.

3.19.2 Search Streaming Function

This function (database fetch method) is used for searching over a large database. It obtains and displays records from the database at once of a number specified. There are some sample pages with this method embedded provided, on approval progress search page and application progress search page of Workflow. For details, please refer to the API List.

```
DatabaseManager.fetch(sql, stratRow, maxRow)
```

3.19.3 Source Security Function

Source Security Function is a function that prevents any illegal file downloading via the Web, by placing only the bare minimum number of files on the Web server that must be directly referred by HTML, such as image files (e.g.if files), CSJS, and applets, while keeping major program data on Service Platform where Application Runtime or Resource Service runs. Since intra-mart’s can run on multiple number of hardware independently via the servers’ network connection, the program files on Application Runtime or Resource Service that are operating on other independent machine, and the data files stored on Storage Service, all cannot be illegally downloaded via the Web.

3.19.4 Application Lock Function

Application Lock Function (process transaction) can be realized. By using the Lock API, serial processing of programs can be achieved. In addition, this API can also lock all the servers in a distributed application server environment. (This function uses “Serialization Service”.) Please refer to “API List” for details.

3.19.5 Unique Information Retrieval Function

This API allows the retrieval of unique information from all the Application Runtime, even in a distributed Application Runtime environment. (This function utilizes the Serialization Service). Please refer to “API List” for details.

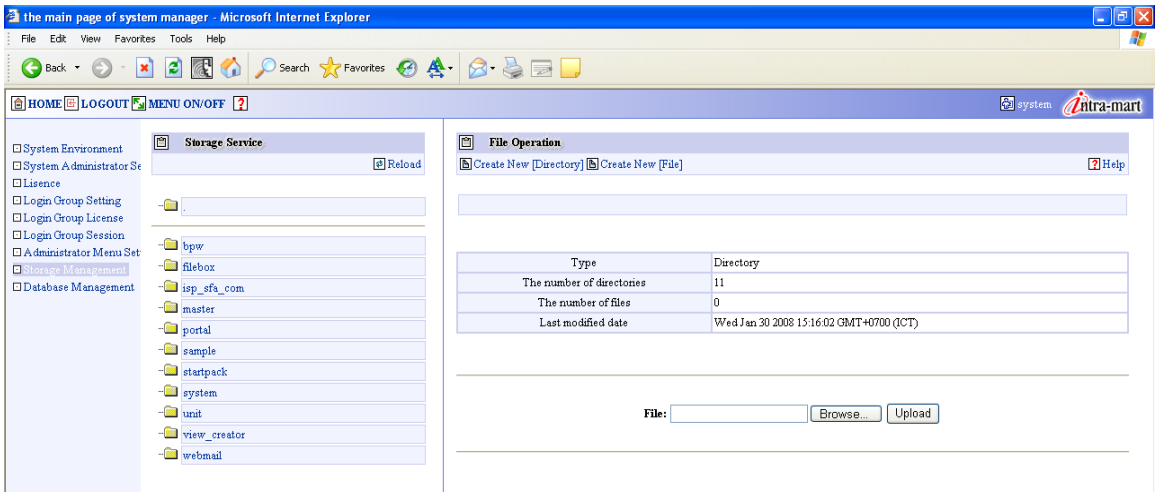
```
Identifier.get()
```

3.19.6 Invoking Stored Procedure in Database

Stored Procedure in the database can be called from intra-mart system. For details, please refer to [Developer's Guide]-[Script Development Model]-[Business Logic Tier]-[Application Common Module]-[DatabaseManager] in API List.

3.19.7 File Operation

This application is a utility that operates files and directories under the system root of Storage Service. It can create new directory or file, delete or upload files, change their names, or edit texts. All the uploaded files will be stored in Storage Service.

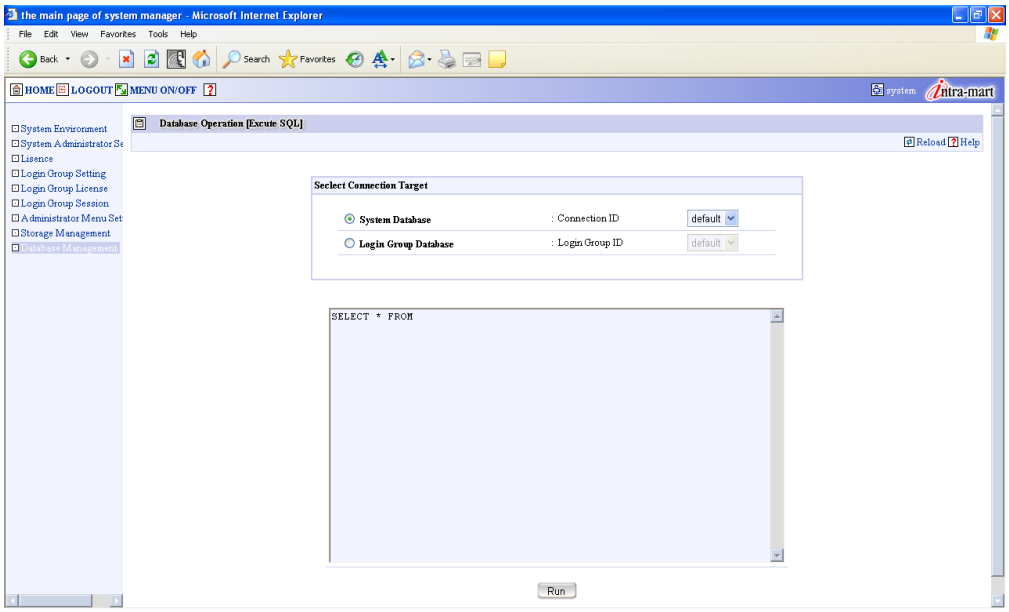


<File Operation Page>

3.19.8 Database Operation

Database operation is a simple tool that directly executes a SQL statement on a database.

Select a connection destination, key-in the SQL statement in the text area and click [Run] button.



<Database Operation Page>

3.19.9 Linkage with LDAP

Information on LDAP can be searched, updated, or deleted. For details, please refer to [Script Development Model]-[Application Common Module]-[Module.ldap] of API List. Login users can also be linked between intra-mart and LDAP. Please refer to “1.11 Usage with LDAP” in the Administrator Guide.

3.19.10 Internationalization

The following functions are designed and implemented in consideration of internationalization (i18n).

- Script Development Model
- im-JavaEE Framework(only Service Framework)
- Application Common Master
- Access Security Module

You can develop internationalized applications or localize each application by using these.

- Please determine the type of the locales to use before the operation.
- If adding a locale when the system runs, it might give some impacts on the application operation with occurring data disorder especially in application common master.
- Please refer to the following documents about the details of the functions.
  - ・API list
  - ・im-JavaEE Framework specification document
  - ・Application Common Master Materials
  - ・Access Security specification document

3.19.11 Creating Standard Screen (Common Screen Design)

The following documents are prepared.

- Page design guideline
- Style sheet specification document

APIs according to the page design guideline are also prepared. API specifications are included in the documents of Page design guideline.

By creating a screen page by using common API based on this, you can develop applications in the same screen page design as the product standard page. If the designs are standardized, the appearance and operability will be unified. This is an advantage for users to handle an application without uncomfortable feelings when they are called from the menu. Thus, please consider using this guideline as a method to unify the design at the application development.

- Each page source of intra-mart WebPlatform/AppFramework is installed in the plain status. Please use these as application methods of Page Design Guideline or an example to use API.

3.19.12 Using Portal Page

You can create and display a specific portlet in intra-mart portal page when creating intra-mart application.

- You have to configure the page displayed as a portal according to the rules of portal modules but not of the normal screen page. Please refer to [JavaEE Development Model]-[Business Fundamental Tools]-[Portal Module]-[Portal]-[Development Guide] and [Script Development Model]-[Business Fundamental Tools]-[Portal Module]-[Portal]-[Development Guide] in API list about creating a page for a portal.
- Please also refer to “2.10 Portal Setting and Operation” in Chapter 2, “Login Group Administrator” in Administrator Guide.

3.19.13 Shortcut Access Function

This is a function to switch the screen page after login to the arbitrary page by specifying URL parameters for short cut access as initial access URL.

In addition, you can configure the security about the display.

Please refer to “Access Security Specification Documents” for details.

Security functions which can be used when using short cut access functions are as follows.

- Specifying users to display a specific page (you can select multiple users)
- Specifying an expiration date to display a specific page.
- Selecting the following functions about the login control.
  - Accessing directly to the specified page after entering user ID and password in login screen.
  - Accessing directly to the page without specifying user ID and password
    (This can be used only if specifying one user allowed to display.)

Shortcut access URL

This is a URL added as a parameter to the normal initial access URL.

http://<server>/<context-path>/<login-group>.portal?im_shortcut=**<Shortcut ID>**

Example :
http://localhost/imart/default.portal?im_shortcut=xazh03nbe43wd

Creating a shortcut ID

Shortcut ID is tied with displayed page and security information.

This is created with using API by specifying displayed page and security information.

In the main areas, there is an explanation of the creation steps of ID shortcut in case of specifying “pages/src/sample/shortcut.js” and “shortcut.html”.

```
// Creating a shortcut manager
ShortCutManager manager = new ShortCutManager(“default”);

// Creating shortcut information
ShortCutInfo shortCutInfo = new ShotCutInfo();

// Displayed URL
shortCutInfo.setU("sample/shortcut.jssp");

//Parameter setting given to the displayed URL(arbitrary specification)
shortCutInfo.setUrlParam("arg1","value1");
shortCutInfo.setUrlParam("arg2","value2");

// User who can display
shortCutInfo.setAllowsUsers(new Sring[]{"guest","ueda"});

// Whether login authentication is necessary or not (authentication is required)
shortCutInfo.setAuth(true);

// Expiration date of this information (valid for 10 days after creating)
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.DATE,10);
shortCutInfo.setValidEndDate(calendar.getTime());

// Creating shortcut ID
var shortCutId = manager.createShortCut(shortCutInfo);
```

intra-mart WebPlatform/
AppFramework

Chapter 4 Appendix

4.1 Message Settings

You can execute the message setting in Java property file format under the “%Server Manager%/conf/message” directory.

- Please refer to “Administrator Guide” for further details on message setting.

Use “MessageManager” class to retrieve the configured message. MessageManager is equipped with the “getMessage()” method to obtain message character strings from a message ID. This method will retrieve the message based on the locale at the time of login.

- For details, please refer to the description on MessageManager Object in “API List”.

4.2 Reserved Word List

The following terms are intra-mart's reserved words and cannot be used for other purposes.

- IMXXX (prefix is “IM(im)”)
- *XXX(prefix is “* (underscore)”)
- Classes used by intra-mart API, and global function names
- (intra-mart API using capital letters as prefix)
- All the reserved words in JavaScript and Java

4.3 Restrictions

The followings are restrictions on the application file names and JavaScript function names.

4.3.1 File Name

The following characters cannot be used for a file name.

```
¥ / : ; * ? ” < > | & # [ ] ( ) { } (space) (tab)
(Two-byte Japanese characters cannot be used)
```

- File name refers to the presentation pages (.html files) and file containers (.js files). Data files are not included.

4.3.2 ID, Code

The following characters cannot be used for any of the ID or code (e.g. User code) in intra-mart functions.

```
¥ / : , ; * ? ' " < > | & # + [ ] ( ) { } (space) (tab)
(Two-byte Japanese characters cannot be used)
```

4.3.3 JS Function

The following characters cannot be used for a function name

```
* < > [ ]
(Two-byte Japanese characters cannot be used)
(Other restrictions include those based on JavaScript specifications)
```

- These are constraints for the functions operating on servers. They are not applicable to the functions that operate on client side (within HTML). In addition, these constraints are applicable not only to the functions, but also to the registered names and method names of the functions.

intra-mart The Power for Enterprise Web Solution

intra-mart WebPlatform/AppFramework Ver6.1

Programming Guide

Script Development Model

The Second Edition, August, 2007

株式会社 NTTデータ イントラマート
〒107-0052 東京都港区2-17-22 赤坂ツインタワー本館
TEL(03)5549-2821 FAX(03)5549-2816
E-mail:info@intra-mart.jp
ホームページ:http://www.intra-mart.jp