WebSphere®

IBM WebSphere Multichannel Bank Transformation Toolkit
Version 8.0

BTT Business Template Sample

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" section.

This edition applies to Version 8, Release 0, Modification 0, of *IBM WebSphere Multichannel Bank Transformation Toolkit* (5724-H82) and to all subsequent releases and modifications until otherwise indicated in new editions.

IBM welcomes your comments. You can send to the following address:

IBM China Software Development Lab
Bank Transformation Toolkit Product
Diamond Building, ZhongGuanCun Software Park, Dongbeiwang West Road No.8,
ShangDi, Haidian District, Beijing 100193 P. R. China

Include the title and order number of this book, and the page number or topic related to your comment.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

BTT Business Template Sample

THE IBM® WebSphere® Multichannel Bank Transformation Toolkit Business Template is a sample that demonstrates new features of this version, and demonstrates best practices for customer for how to use BTT more efficiently. At BTT version 8.0.0, this sample includes both internet banking sample applications and mobile sample applications.

1 Internet Banking

This part of documentation will cover business template sample web applications in following format:
a. Scenarios that included in this sample
b. Implementation
c. Running the sample

1.1 Scenarios

Internet Banking is a popular and important channel to provide high quality services to customers. Bank Transformation Toolkit can help to create Internet Banking application, meanwhile, it can help to increase and ensure the benefits of IT investments.

From business perspective, a typical Internet Banking solution composed of Authentication and authorization component, business operations such as account management, investment management etc.

The Internet Banking Sample Application takes fund management as an example to demonstrates the most important steps in developing an online banking application using the features of the IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT that support XUI editor and transaction builder tool. Further more, the latest Web2.0 technology is supported by THE IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT. This sample should be used as a guide for solution providers who wish to gain a better understanding of the facilities of the IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT in this area, and how to better use them to build an online banking application.

1.1.1 User log in.

This is user authentication and authorization part. A user is conducted to input his username, password and verification code to login. A BTT operation will be invoked to authenticate this user. If successfully, the main page will be displayed. Otherwise, an error record will be generated and the user will be redirected to login page. In this sample, please use user01 as username, and any six characters as password to login the Internet Banking Sample.

1.1.2 My Service

Shortcuts to online services. This service list is configurable. Users can configure it to list their favorite services. Please refer to the Sample Implementation part for how to configure My Service List.

1.1.3 My Reminder

Shortcuts to messages and reminders which are generated by system. This is configurable. Users can configure it to list reminders which are stored in local file , or connect it with core banking system to fetch dynamic news. Please refer to the Sample Implementation part for how to configure My Reminder List.

1.1.4 Special Offer To You

Unica Interact enable user to provide real-time marketing offers for inbound interactions.
BTT provides seamless integration with Unica Interact by Interact connector service and unified user session management. BTT Unica Interact Integration can accelerate the application development of Interact for BTT users. Please refer to BTT Unica Interact Integration part for details.

1.1.5 Fund

Fund sample is an implementation of fund management, including fund account summary, fund search and product list, user's interested fund list and position list, fund purchase process. From business perspective, fund management scenario is a very typical banking business operation. It covers most common operations such as query, fund product list, interaction with backend system, business process flow etc. All of these can be developed using IBM® WEBSPHERE®

MULTICHANNEL BANK TRANSFORMATION TOOLKIT. Please refer to Sample implementation part to learn how to develop this application using BTT.

1.2 Sample implementation

In this part, a fund management transaction will be introduced as an example to demonstrate how to implement an internet banking solution with IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT: technical design, Unica integration, how to define context, operation, flow and related data will be introduced in this part.

From technical perspective, below table lists all IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT technical components that need to be implemented in the Internet Banking Sample.

Component	Details	Comments
Flow	fundFlow	Fund management transaction flow. This is the main flow of this sample.
	fundBuySubFlow	Fund purchase flow that will be invoked as a sub-flow in fundFlow flow
	fundInterestedSubFlow	Adding selected fund product into interested list flow that will be invoked as a sub-flow in fundFlow flow.
Operation	startUpHtmlSessionOp	An operation that connects to the server to establish a session and create a new context for the user
	LoginVerificationOp	An operation that verifies user's authentication information. If successfully, a Unica service will be invoked meanwhile to read special offers from Unica server.
	UnicaOp	An operation that redirects user to Unica page.
	signOutOp	An operation that will sign current user out.
	ws* operation	All web service operations that are used to invoke back-end web services. All these web service operations are generated by BTT Web Service Wizard Tool.
Data	Please refer to the Data tab of fundFlow transaction editor tool.	Defining all data structure and fields that are used in this sample.
XUI	login	User login page.
	fund_main	Main page of fund management.
	fundbuy_main	Pages used in fund purchase sub-flow.
	fundbuy_confirm	

	fundbuy_complete	
	fundinterested_complete	Pages used in adding selected fund product into interested list sub-flow.
	fundinterested_confirm	
	signOutResult	Sign out result page.
JPA	BusinessTemplatesUtils	A java project that implements all JPA functions in this sample.

1.2.1 JPA (Java Persistence API)

JPA is a simpler programming model for Entity Persistence. IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT leverages the capability of JPA to facilitate access of database. For more information about JPA, please refer to JPA website.

First of all, you have to change persistence.xml under BusinessTemplatesUtils project. All database access information is configured here.

- Change *YourDBIPAddress* to the IP address of your database server
- Change *dbusername* and *dbpassword* to the username and password of your database

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="BusinessTemplatesUtils">
        <class>com.ibm.btt.sample.entities.Ejtable</class>
        <class>com.ibm.btt.sample.entities.EjtablePK</class>
        <class>com.ibm.btt.sample.entities.Logininfo</class>
        <properties>
            <property name="openjpa.ConnectionDriverName"
value="com.ibm.db2.jcc.DB2Driver"/>
            <property name="openjpa.ConnectionURL"
value="jdbc:db2://YourDBIPAddress:50000/SAMPLE:retrieveMessagesFromServerOnG
etMessage=true;"/>
            <property name="openjpa.ConnectionUserName" value="dbusername"/>
            <property name="openjpa.ConnectionPassword" value="dbpassword"/>
            <property name="openjpa.jdbc.Schema" value="db2inst1"/>
        </properties>
    </persistence-unit>
</persistence>
```

Two database tables have been used in this sample. The first one is Logininfo table used to store users records, another one is Ejtable used to store Electronic Journal records.

In this sample, use Script.sql to create a DB2 database and tables. The Internet Banking Sample will leverage JPA to connect to this database.

Below is the sql statements that used to create tables.

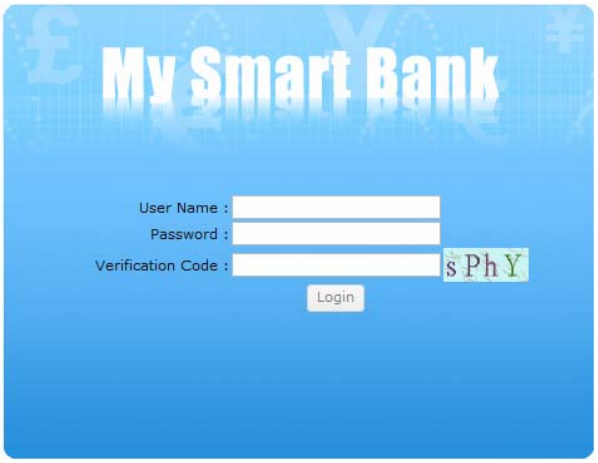
```
CREATE TABLE "DB2INST1"."EJTABLE" (
        "ACCOUNT" VARCHAR(40),
        "ACCOUNT2" VARCHAR(40),
        "AMOUNT" DECIMAL(5 , 0),
        "AMOUNT2" DECIMAL(5 , 0),
        "CURRENCYTYPE" VARCHAR(40),
        "ERRORCODE" VARCHAR(40),
        "REMARK" VARCHAR(100),
```

```
        "RETCODE" VARCHAR(40),
        "TXNID" VARCHAR(40) NOT NULL,
        "TXNTIME" VARCHAR(40),
        "RECORDSEQNUM" INTEGER NOT NULL,
        "USERID" VARCHAR(40) NOT NULL,
        "RECORDDATE" VARCHAR(40) NOT NULL
    )
    DATA CAPTURE NONE;

CREATE TABLE "DB2INST1"."LOGININFO" (
        "USER_ID" VARCHAR(40) NOT NULL,
        "CUST_PRIVATE_INFO" VARCHAR(100) NOT NULL,
        "LAST_LOGIN_IP_ADDRESS" VARCHAR(40) NOT NULL,
        "LAST_LOGIN_TIME" TIMESTAMP NOT NULL,
        "LOGIN_ALIAS" VARCHAR(40) NOT NULL,
        "PASSWORD_ERROR_COUNT" INTEGER NOT NULL,
        "CHANNEL" VARCHAR(40) NOT NULL
        )
    DATA CAPTURE NONE;
```

1.2.2 Log in

1.2.2.1 Functional requirements



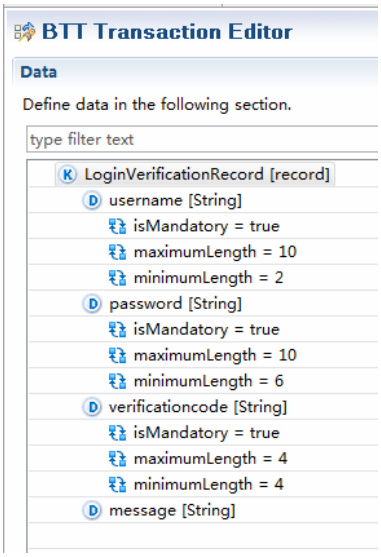
On login page, a user is conduct to input his username, password and verification code before clicking the login button. Then a verification operation will be invoked to authenticate this user. If successfully, the main page will be displayed. Otherwise, an error record will be generated and the user will be redirected to login page again.

1.2.2.2 Operations

LoginVerificationOp is a BTT operation which extends BTTServerOperation. Login verification logic is implemented in this operation. Please refer to src/*definitions/operations/LoginVerificationOp.transaction* for details.

1.2.2.3 Context hierarchy

LoginVerificationCtx context has a refRecord child which references to LoginVerificationRecord. Username, password, verification code are defined as *data* with several properties such as maximumLength and minimumLength. Another data named as *message* is used to record error message which will be promoted to users to indicate login error.

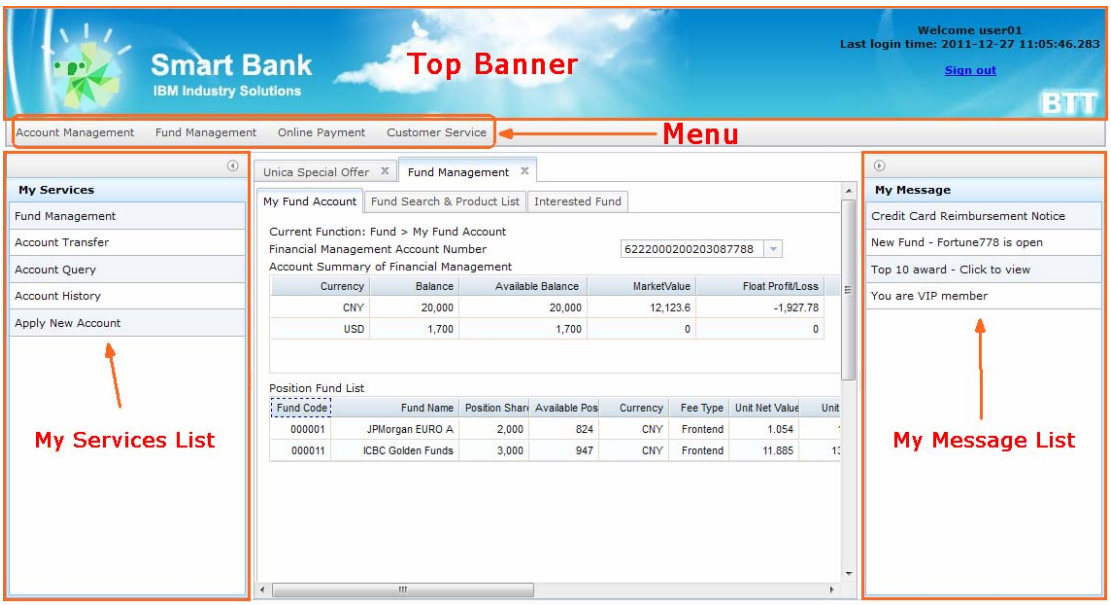


1.2.2.4 User interface pages

login.xui is WebSphere Multichannel Bank Transformation Toolkit XML-based transaction UI files created by the XUI editor. The login.xui file is in XML format at development time, and is then converted to Web 2.0 and SWT widgets at runtime by the WebSphere Multichannel Bank Transformation Toolkit XUI engine.
Please refer to *xui/login.xui* for details.

1.2.2.5 Main page

After login successfully, the main page will be displayed. Users can finish their business on main page.



As show in the figure above, there are Top Banner, Menu, My Services List, My Message List and main area. Please refer to *WebContent/jsp/mainPage.jsp* for details.

1.2.3 Fund

1.2.3.1 Functional requirements

Fund sample is an implementation of fund management, including fund account summary, fund search and product list, user's interested fund list and position list, fund purchase process. The following implementations are available:

- Query customer account list and details
- Query position fund list
- Query fund product list
- Process a fund product purchase sub-flow.
- Add a fund product to customer's interested list sub-flow.
- Query customer interested fund product list

1.2.3.2 Operations

The operations supplied for this application are sample operations to illustrate what would be present in a typical internet banking application.

The sample application has an operation that connects to the server to establish a session and create a new context for the user

(StartUpHtmlSessionOp).

The sample has the following operations to support the transactions:

- Web service operations that lists all account (wsAccount_GetAccountSummary) and related sub-account list (wsAccount_GetSubAccountList).
- A reusable operation for implementing risk assessment (checkRiskAssessOp) in fundBuySubFlow.
- A reusable operation for implementing customer account balance check (checkBalanceOp) in fundBuySubFlow.
- A web service operation that interacts with backend services to create a fund purchase record (wsProduct_BuyFund).
- An operation that interacts with backend database to create a fund purchase electronic journal record (fundBuyEJOp).

- Three web service operations that interacts with backend services to add one selected fund record to customer's interested list (wsProduct_AddToInterestedFund), get customer's fund position list (wsProduct_GetFundPosition) and get customer's interested fund list (wsProduct_GetInterestedFund).
- A web service operation that processes the customer's query condition and return query result (wsProduct_QueryFundProduct.transaction).

Please refer to related transaction file for details.

1.2.3.3 Context hierarchy

The contexts used in the application are arranged in a hierarchy starting at the server root context. The root context, in addition to maintaining session tables for all the clients that connect to the server, keeps all the resources (data and services) to be shared by all the processes and operations launched from any session established with the server.

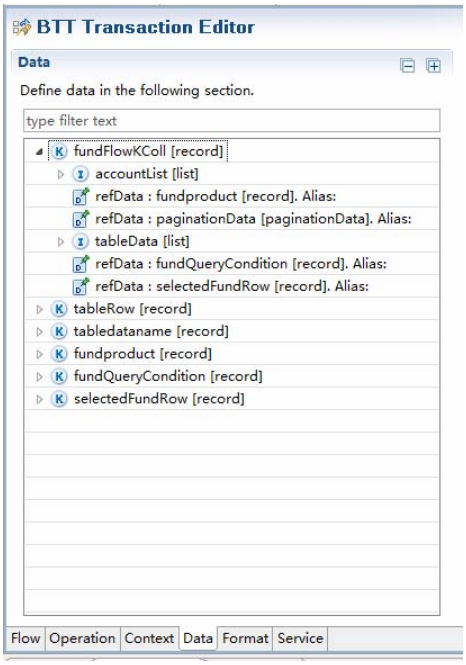
Whenever a session is established on the server, the startUpHtmlSessionOp operation creates an htmlSession context and chains it to the root context. This session context keeps all the resources that operations and processors used during the user session. Services must be properly attached to this context so that they are created only when the session is created instead of being created whenever the user requests an operation.

The toolkit chains all operations and processors used during the session to their own copy of the session context. These contexts store data for the operations and processors.

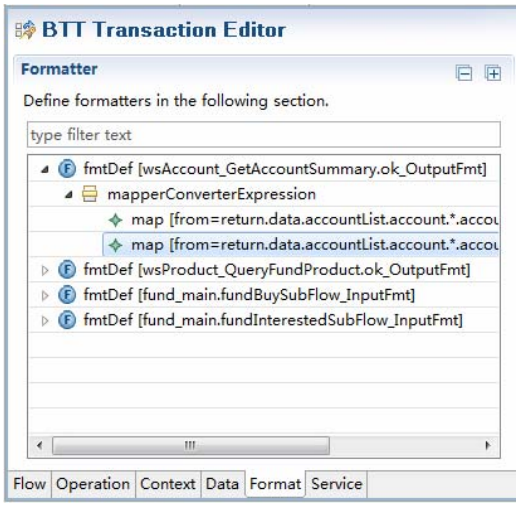
Please refer to *src/definitions/processors/*.transaction* files for details.

1.2.3.4 Data elements and formatter

Data elements are what the toolkit uses to hold data in a structure that enables toolkit entities, processes, and services to add, update, and delete the data during runtime.
In the Internet Banking sample, data elements are defined as below.



Formatters provide a way of transforming the data from one format to another so that various entities, processes, and services can use it. In the definition files for the Internet Banking Sample Application, all formats defined in Figure ** are defined by mapping. Please refer to Defining data mappings.



1.2.3.5 Navigation flows

This is the main component of the architecture being used by the Internet Banking Sample. It provides process management and transactions support. The application implements navigation processes as state machines (processors) and transactions as operations.

The fund purchase process is the only business operation in the Internet Banking Sample Application that requires a navigation flow. This process accepts and processes fund account query, enabling a customer of the internet banking application to query fund product. The process may optionally call a subflow to collect purchase information from the customer to finish fund purchase, or call a subflow to add selected fund product into customer's interested list. The application provides the subflow to demonstrate the main capabilities of the toolkit architecture and reflect the real business world in which some complex business flows are required. In summary, the following navigation processes are used for the application:

- A fund management process (fundFlow.transaction)
- A fund purchase process (fundBuySubFlow.transaction)
- Add selected fund into interested list process (fundInterestedSubFlow.transaction)

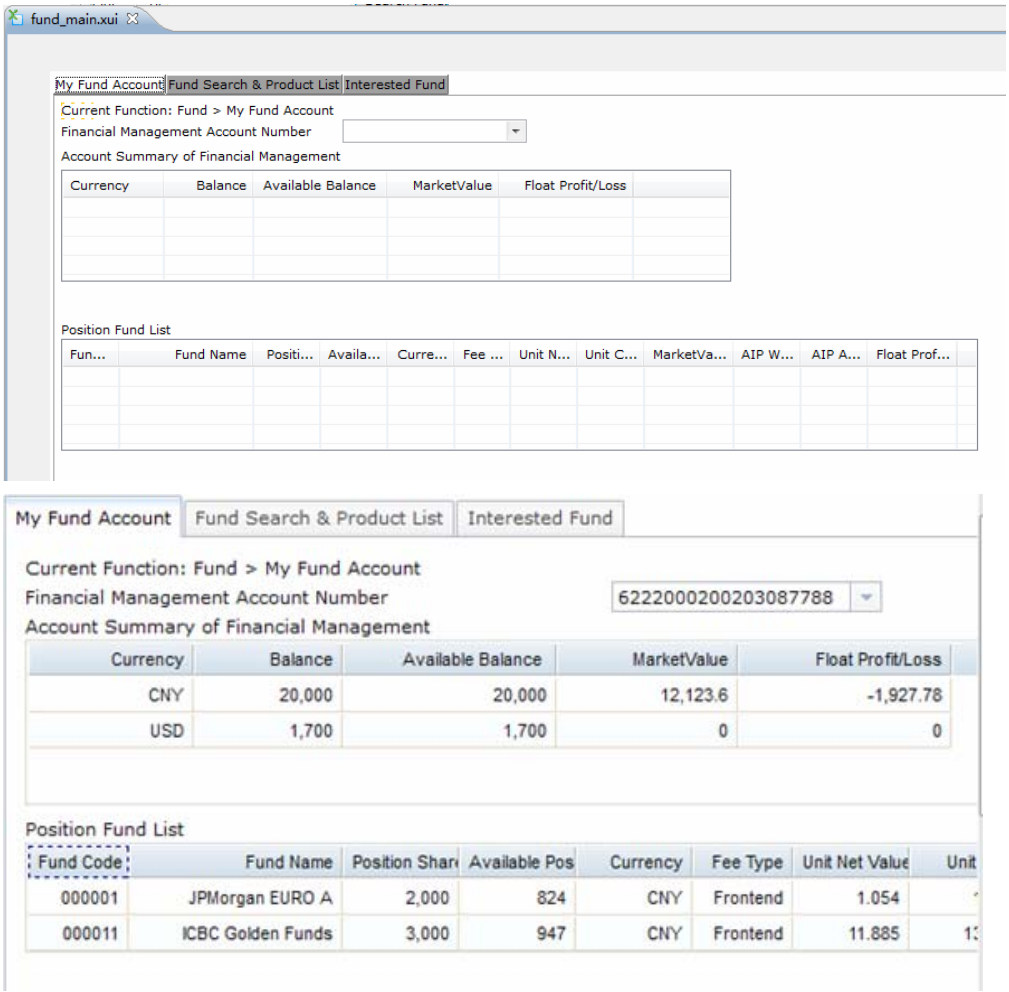
Please refer to *src/definitions/processors/*.transaction* files for details.

1.2.3.6 User interface pages

The IBM® WebSphere® Multichannel Bank Transformation Toolkit XUI editor is a WYSIWYG (what you see is what you get) tool that enables you to design and create user interfaces. XUI is an abbreviation for XML User interface.

Eight XUI pages have been implemented in this sample. Please refer to Table ** for details.

Figure ** is one user interface page named as fund_main.xui design with XUI editor. Figure ** is the fund_main.jsp file generated from fund_main.xui.



1.2.4 Unica integration

Unica Interact enable user to provide real-time marketing offers for inbound interactions.

BTT provides seamless integration with Unica Interact by Interact connector service and unified user session management. BTT Unica Interact Integration can accelerate the application development of Interact for BTT users.

1.2.4.1 BTT Unica Marketing Service

BTT Unica Marketing Service provides the connectivity to the Unica Interact server. BTT Unica Marketing Service is based on BTT service architecture. It provides XML based configuration for Unica Interact connectivity and wrapped Interact Java API for easier usage.

The following is an example of service XML definition of Unica Interact marketing service.

```
<com.ibm.btt.cs.marketing.UnicaMarketingService
id="unicaDemoService"  audienceLevel = "Individual"
```

```
interactiveChannel = "BTTtest"
serverURL="http://UnicaServerIP:7001/interact/servlet/InteractJSServi
ce" />
```

BTT provide the implementation of Unica Interact marketing service by class `com.ibm.btt.cs.marketing.UnicaMarketingService.` The following is the property of UnicaMarketingService.

Property ID	Description	Default Value
serverURL	URL of Unica Interact service	N/A
interactiveChannel	The name of the interactive channel this session refers to. This name must match the name of the interactive channel defined in Campaign exactly.	N/A
audienceLevel	An audience level is a collection of identifiers that can be targeted by a campaign. For example, a set of campaigns could use the audience levels “Household,” “Prospect,” “Customer,” and “Account.” Each of these levels represents a certain view of the marketing data available for a campaign.	Individual
initialDebugFlag	A boolean which enables or disables debug information. Valid values are true or false. If true, Interact logs debug information to the runtime server logs. The debug flag is set for each session individually. Therefore, you can trace debug data for an individual session	true

BTT provide a common implement interface MarketingService. The `UnicaMarketingService` implement interface MarketingService. The following is the interface definition:

```
public interface MarketingService {

    public boolean startSession(String sessionID, NameValuePair[]
initialAudienceId, NameValuePair[] initialParameters ) throws
Exception;

    public Offer[] getOffers(String sessionID, String
interactionPointName, int reqCount) throws Exception;

    public boolean endSession(String sessionID) throws Exception;

    public boolean postEvent(String sessionID, String eventName,
NameValuePair[] eventParameters) throws Exception;
```

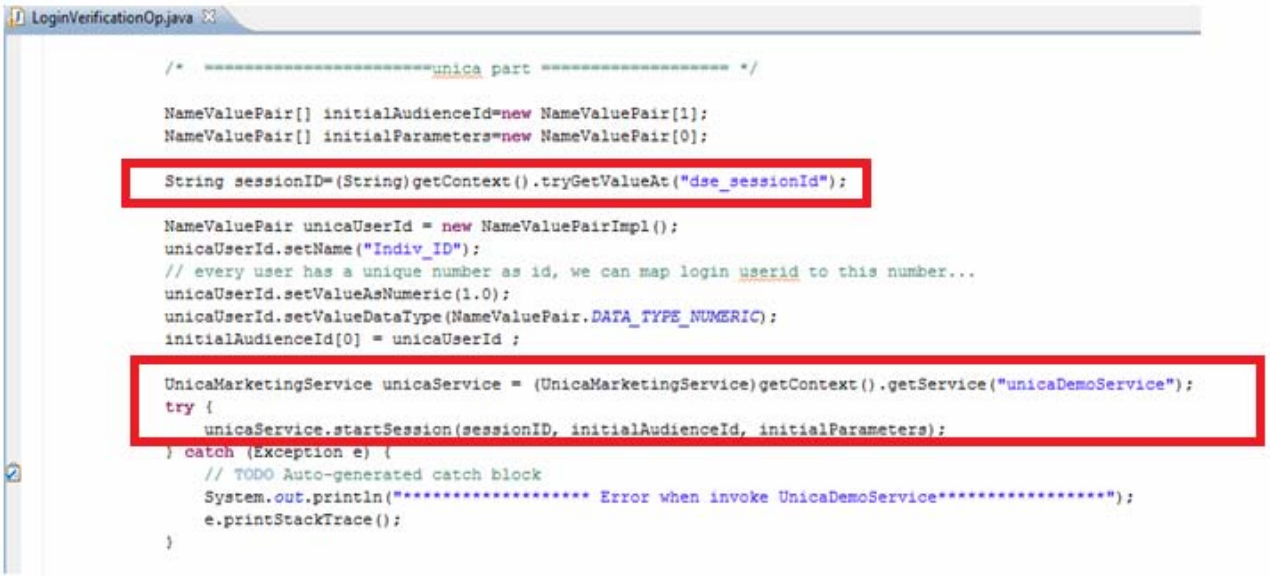
}

Note that in the Business Template, there is a dummy marketing service that also implements MarketingService. The dummy service can be used in a DEMO/sample environment. User can switch to use real Unica Interact environment by a bit change of XML, no need to change any code.

1.2.4.2 Session Management Integration

In runtime, Unica Interact uses session to keep the state of the user inbound interactions. Also BTT has powerful session management. So we can leverage BTT session management to provide seamless session management integration between Unica Interact and BTT.

After a user login BTT application, BTT creates BTT session and BTT session context. In the LoginVerification Operation of Business Template, the Unica service is created and BTT session ID and user ID is used as the input parameter of the startSession method of Unica Marketing Service.



BTT use session context to keep the data and referenced service for a login user. The following is the session context definition in Business Template.

```
<context id="branchServer" type="branch" parent="nil">
    <refKColl refId="branchData"/>
    <refService refId="realCSServer" alias="realCSServer" type="cs"/>
    <refService refId="dummy_unicaDemoService" alias="unicaDemoService" type="unicaDemoService"/>
</context>

<context id="sessionCtx" type="session" addToDynamicKColl="true">
    <refKColl refId="sessionData"/>
</context>
```

The sessionCtx will chain to branchServer root context in runtime. So the session context can refer the Unica Marketing Service and get the singleton instance of service for current session by following code: `getContext().getService("unicaDemoService").`

The unified session ID is stored in the BTT session context, so when BTT application want to retrieve an offer in some contact point of a specific page, user can use following JSP code to get offer (Please refer unica.jsp of Business Template).

```
String contactPoint = "BTT_MainBanner";

String sessionID = utb.getSessionId();
UnicaMarketingService unicaService = (UnicaMarketingService) utb
        .getContext().getService("unicaDemoService");
if (unicaService != null){
    Offer[] offers = unicaService.getOffers(sessionID, contactPoint, 2);
```

1.3 Running the sample

1.3.1 Setting up the sample

1.3.1.1 About this task

The following procedure describes how to install the Internet Banking Sample Application in RAD. The procedure assumes that you will run the sample in one of the Test Environment configurations but you can also run it on a WebSphere Application Server instead. If you are not using one of the Test Environment configurations, make the appropriate adjustments to the procedure.
The following procedure applies whether you are running RAD on Windows or Linux.

To set up the application in RAD:

1.3.1.2 Procedure

1. Create a DB2 database using Script.sql. Start the database service. For more information about DB2 database administration, please visit DB2 Information Center.
2. Start RAD and open JavaEE perspective.
3. Create a server. For more information, see Create a server.
4. Add the sample project to the server. For more information, see Adding projects to the server.
5. Starting the server. For more information, see Starting the server.

1.3.2 Running the sample

1.3.2.1 Before you begin

Once you have set up the Internet Banking Sample Application as described in Setting up the application in RAD, you are ready to run the application in the WebSphere 7.0 Test Environment on RAD. If you set up the application on a different configuration, make the appropriate adjustments to the procedure.
The following procedure describes how to start the server and then run the Internet Banking Sample Application.

1.3.2.2 Procedure

1. Start the application server by selecting the Internet Banking Sample Application Server Instance available in the Servers view of the JavaEE or Web perspective. Right-click and select Start.
2. Start the client by entering the following in the location or address field of a Web browser: http://<hostname>:<port>/BusinessTemplateWeb where <hostname> is the name of the server running the sample; <port> is the port number on which the application server is listening the HTTP requests (usually on port 9080).

1.3.3 Deploying the application on WebSphere Application Server

1.3.3.1 About this task

The Internet Banking Sample Application is a demonstration of how to implement HTML Channel and Web2 Channel for IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT. You can deploy the sample on any of the server platforms supported by IBM® WEBSPHERE® MULTICHANNEL BANK TRANSFORMATION TOOLKIT.

The following procedure describes how to deploy and run the sample on IBM WebSphere Application Server:

1.3.3.2 Procedure

1. Start the WebSphere Application Server.
2. Start the WebSphere Administrator's Console. For more information, see Starting and logging off the administrative console
3. In the console, install the BusinessTemplate Application. For more information, see Installing enterprise application files with the console.
4. Open a Web browser.
5. To start the client, enter the following in the location or address field of a Web browser: http://<hostname>:<port>/BusinessTemplateWeb where <hostname> is the name of the server running the BTT application; <port> is the port number on which the application server is listening the HTTP requests (usually on port 9080).

1.3.4 Results

During the session establishment request, the startup operation is executed and the sign in page is created for the browser. Your browser displays the sign in page.
Sign in using user01 as username and any six characters as password. If successfully, you will be redirect to the main page of Internet Banking Sample.

2 Mobile Banking

2.1 Background

IBM WebSphere Multichannel Bank Transformation Toolkit (BTT) has the built-in ability to support multichannel application development for banking and financial industry. With BTT, you can easily build a single set of application to support users from various of channels and devices. In this set of sample mobile applications, we are demonstrating the mechanism to develop iOS and Android rapidly basing on existing web application.

The usage of mobile devices, especially smart phones and pads, are increasing fast in people's daily life in recent years. People are willing to deal with many kinds of business on their mobile devices at their convenience, such as some banking operations like account query, financial product purchase etc. In these two sample mobile applications, we are going to show you some basic fund purchase views, to help you understand the basic methodology to implement a mobile application with BTT.

The sample assumes the following scenario. A bank customer get mobile banking app installed onto her mobile device, she can log in to bank in the app, query funds and purchase desired funds online.

2.2 Prerequisite

To run the sample app, you need to follow the steps of previous internet banking part to setup your development environment, and make the internet banking web application run properly.

You also need to set up your mobile application development environment for iOS and/or Android, you need a MAC OS X system for iOS development. The sample will run on top of iOS 4.3+ version and Android 2.2+ version. You need to find and follow their own guide of iOS/Android development, this is not covered in this document.

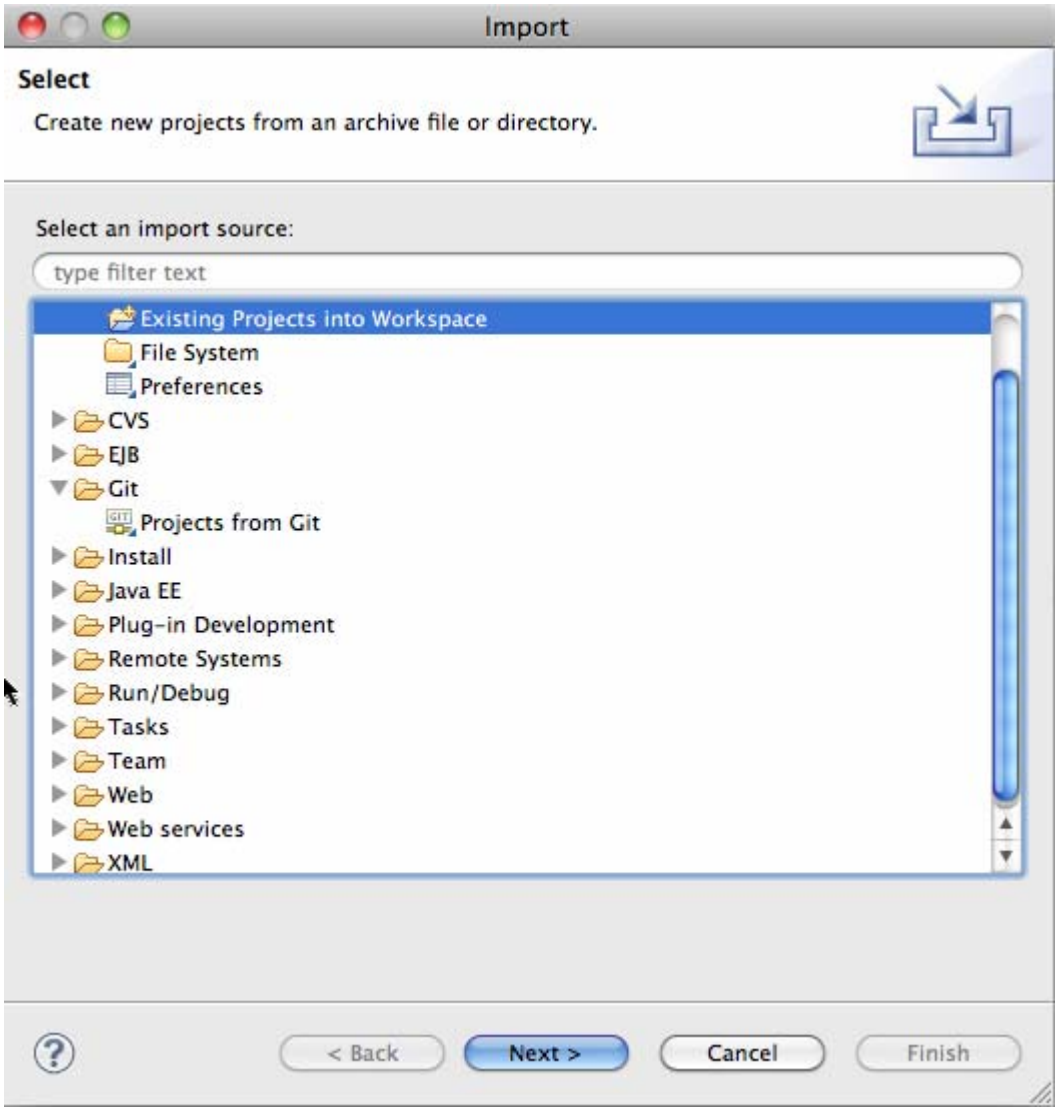
Once you set the environment up, you need to make sure the iOS/Android development system can connect to the internet banking web application through local network, otherwise the mobile app would not be able to run properly.

Due to the limited document coverage, we presume you have the basic knowledge, skill and experience of iOS/Android platform application development, so we are not going to cover specific details of each platform. If you do not understand some terminology for any platform, please try to search and learn by Internet.

2.3 Importing mobile app projects

You need to get the sample mobile app projects in zip file, BTTMobileBankIOS.zip and/or BTTMobileBankAndroid.zip, from BTT release package. Then extract them to your desired location on your mobile development system.
For iOS app, you need to open the project at your extracted location from XCode 4.

For Android app, you need to import the project into eclipse 3.6 or above version through the project import wizard.

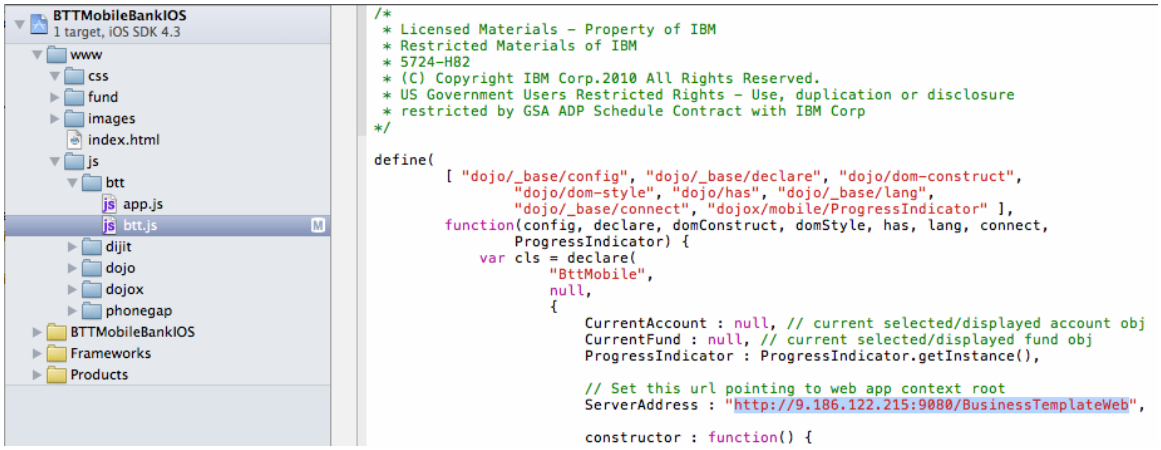


2.4 Setting web application url

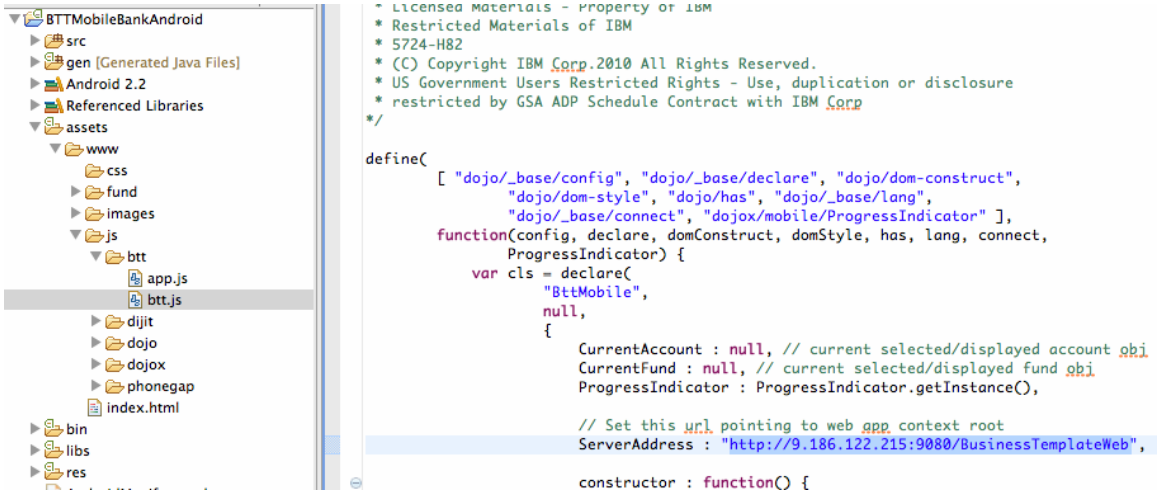
Once you opened the project, you need to edit a variable value to point the app to the right url of the web application.

For iOS app, please browse to www/js/btt/btt.js in the project , find the variable ServerAddress, then set its value to the right url of the web application context root, then

save the modification.



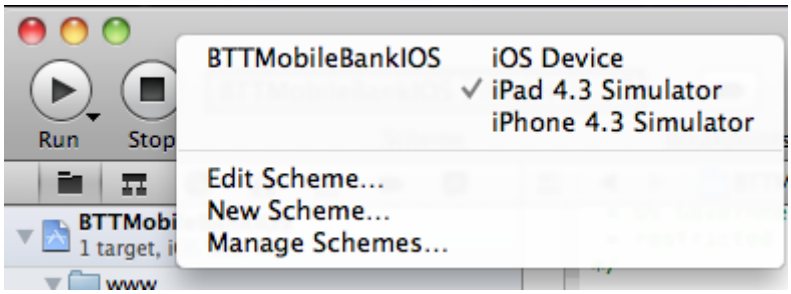
For Android app, please browse to the “assets/www/js/btt/btt.js” in the project, find the same variable ServerAddress and set its value to the right web application context root in your environment.



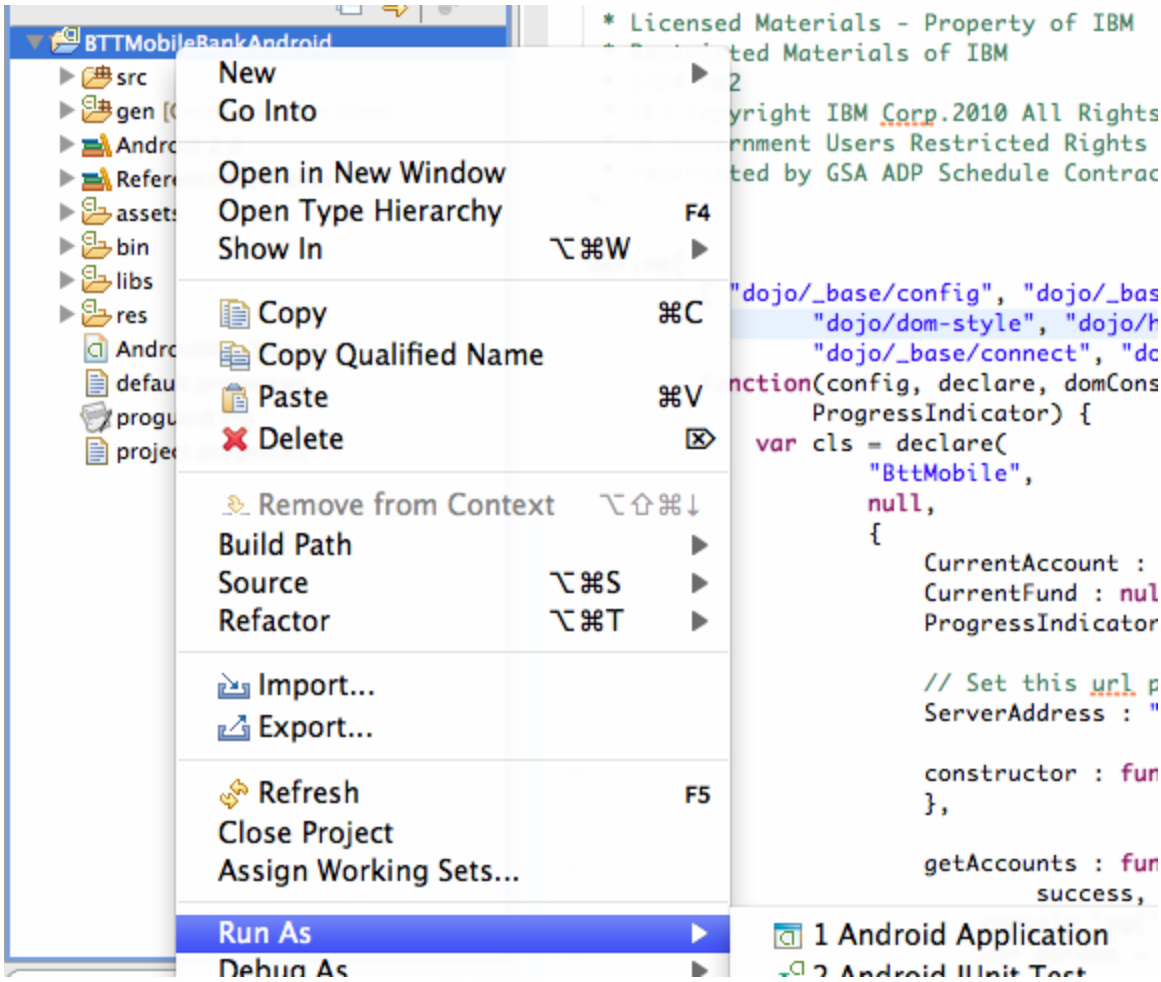
2.5 Starting up the mobile apps

Note, please make sure the internet banking sample web application is running properly before launching the mobile app.

For iOS, please choose your target on the top left of XCode. The app supports both iPad and iPone, and will automatically show different UI layout for them to adapt to the different screen size. Then please click the Run button to start the selected simulator/device, the app will be built and pushed to the simulator/device to run.



For Andoird app, please make sure you have created a Android Virtual Device or connected a physical Android phone, with Android 2.2 or above version Android OS. Then right click on the project, then choose Run As -> Android Application on the popup context menu.



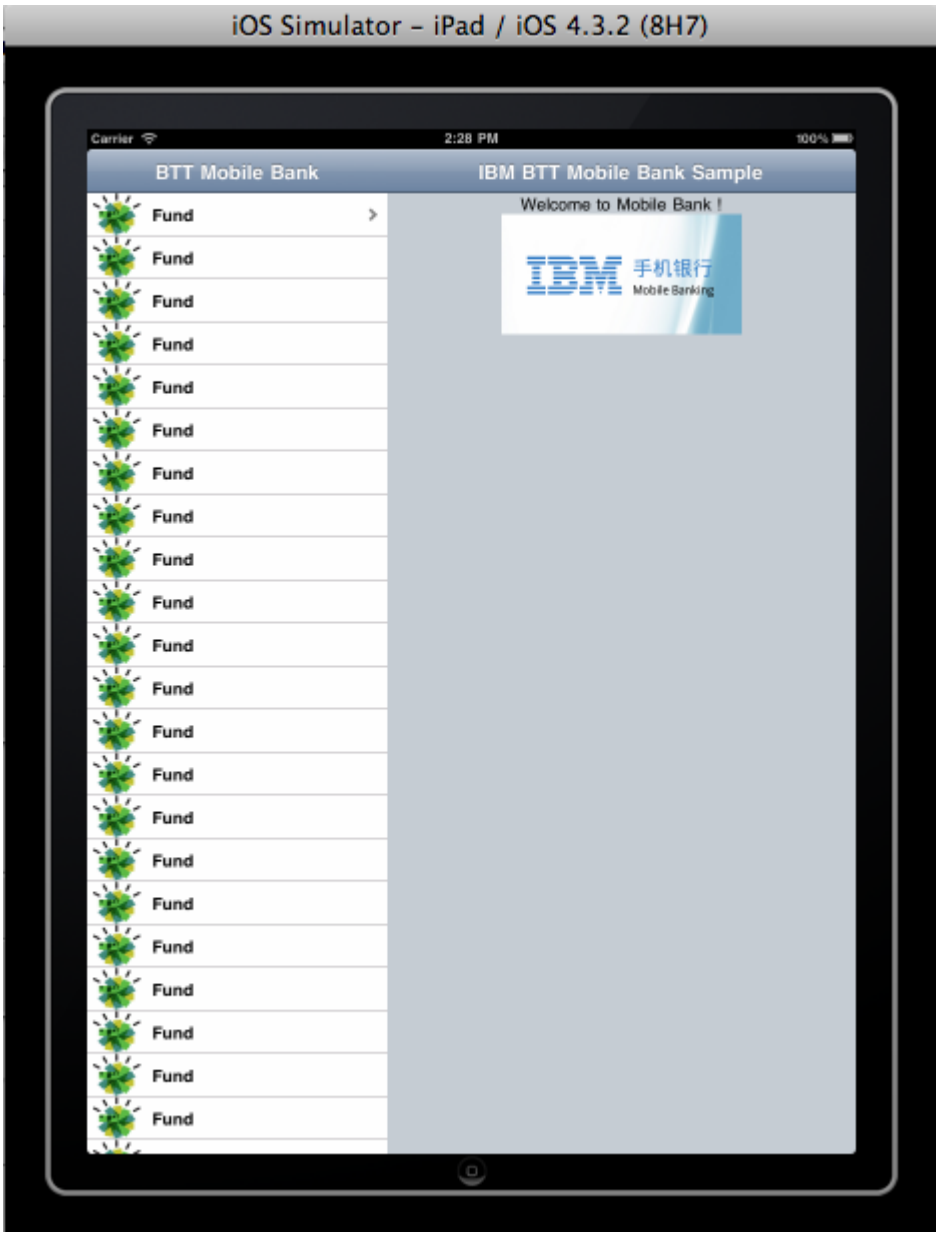
2.6 Paying with the mobile app

The iOS app and Android app have same scenario and same UI and even same core code, so you should be able to get similar experience. To save the duplicate words, we are going to use iOS as sample in this section, to lead you playing with the app. The app might perform a little worse on the Android simulator, because of the bad

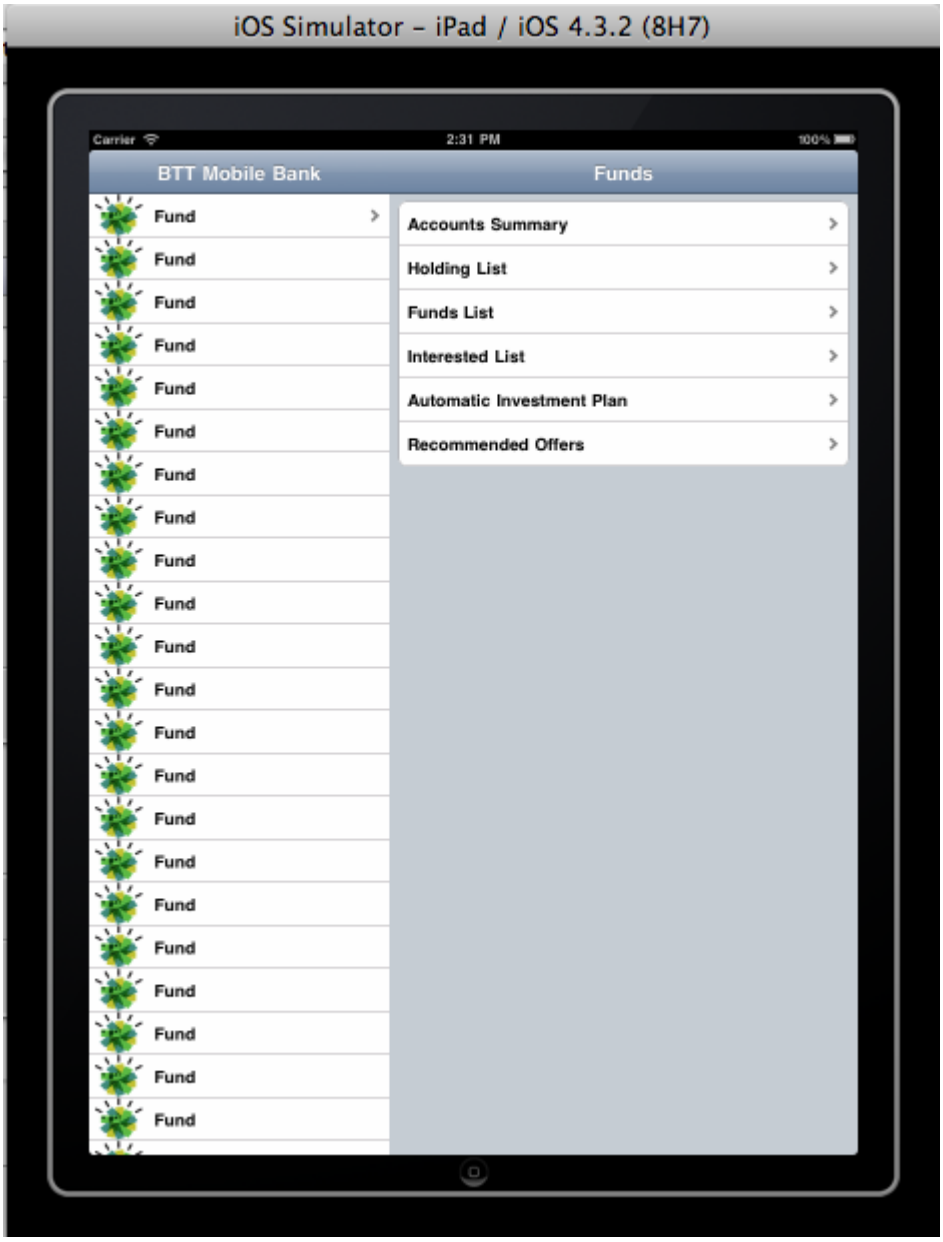
performance of Android simulator, but the business scenario is exactly same.
Once the iOS simulator/device is started up, the IBM BTT Mobile Bank Sample App will start up as well, it will connect to the internet web app, establish a session, and show the login view. You need to input the username, password and captcha verification code to login. As this is a sample, we are not verifying the password, you can simply input the default username “user01” with any password more than 6 characters, and the correct captcha code to proceed.

Once you login successfully, you will see the welcome view, in which there is a Special Offers section showing some offering images which can be flicked horizontally, a Messages section showing some fake messages, a Shortcuts section showing fake shortcut, and a button to proceed to main menu.

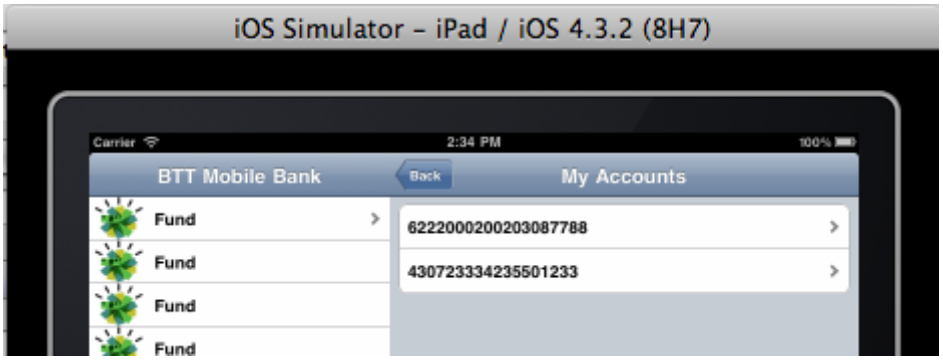
Clicking the Main Menu button, you will enter the main menu, this menu will display different layout basing on your device screen size, for iPAD, it will split the screen into two parts, left for quick navigation, right to display the function area.



Clicking on the first Fund menu, the right frame will show all fund related functions, such as Accounts Summary, Holding List, Funds List, Interested List, Automatic Investment Plan and Recommended Offers. Among them, the Automatic Investment Plan and Recommended Offers are not implemented in this sample.



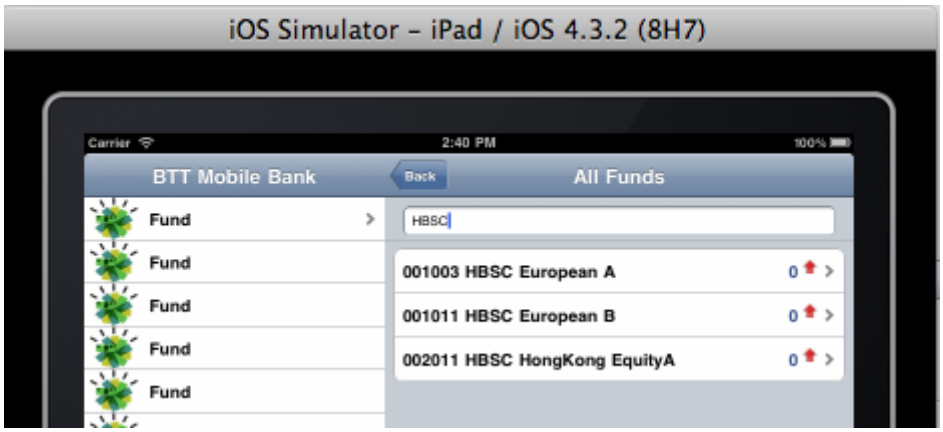
In account summary, it will list your existing accounts, and it will show the holding funds under your each account. The fund entry can be clicked to see more details. Please note, redeem function is not implemented in this sample.



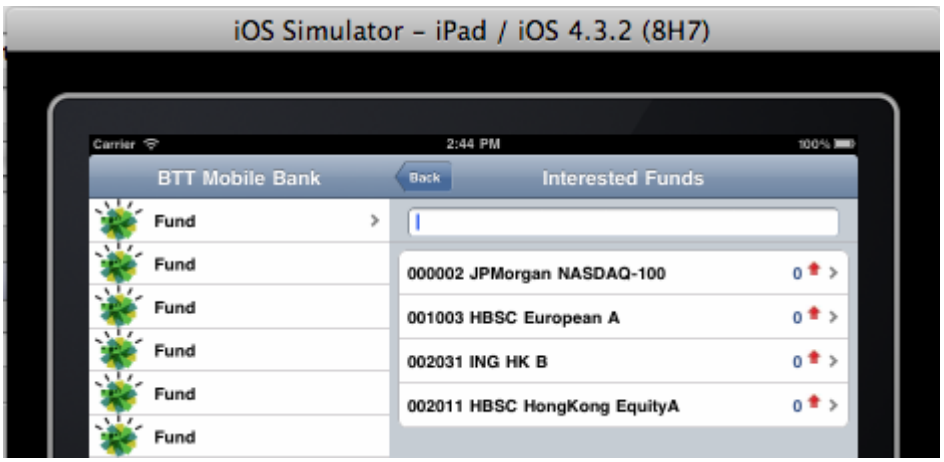
In Holding List view, you can check all your holding funds instantly.

In Funds List view, you can see all available funds, and you can type into the query box at the top of the view to quickly find the Fund by its code or name. Please note,

the query is case sensitive. Once you find a desired fund, you can click the Follow button to add it to your interested list or Purchase button to buy it.



In the Interested Funds view, you can check all funds in which you are interested, you can also quickly filter the funds by top search box, case sensitive as well. You can not remove a fund out of interested list because it is not implemented in this sample. You can proceed to purchase for selected fund.



2.7 Development mode

You may have noticed when you play with the sample apps, that they are implemented using purse html and JavaScript technology. There is almost no native code, neither Objective C code nor Java code, for the UI and business logics. This is made by hybrid development mode. The basic concept is that developer need not to learn and write a lot of native code for different mobile device platforms, they just need to write one set of html and JavaScript code, and then can achieve almost same effect on all major mobile platforms, such as iOS and Android. This extremely saves us effort and made supporting a new mobile platform easy, and the maintenance effort is reduced significantly as well. For more details, please refer to phonegap.com, which is an open source hybrid bridge for major mobile platforms in the market.

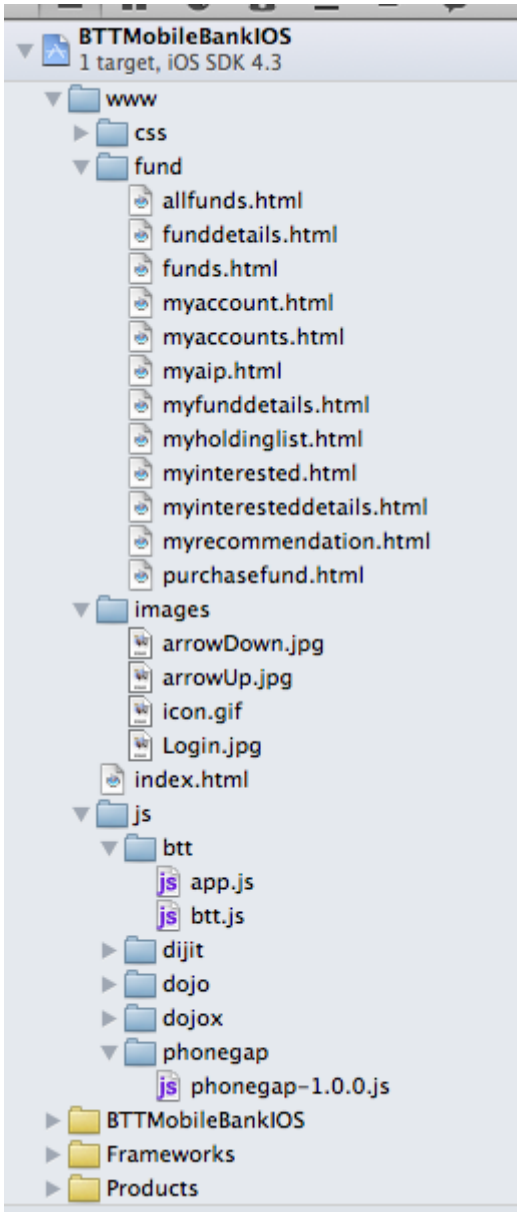
Another important factor is about the flexibility of BTT framework. In this sample, we are reusing the web application built in first part. We even need not to rewrite any code for mobile apps, we just call existing operations by Ajax channel from mobile app, then everything is working. This is exactly what the “Multichannel” mean for BTT product.

2.8 Inside into mobile apps

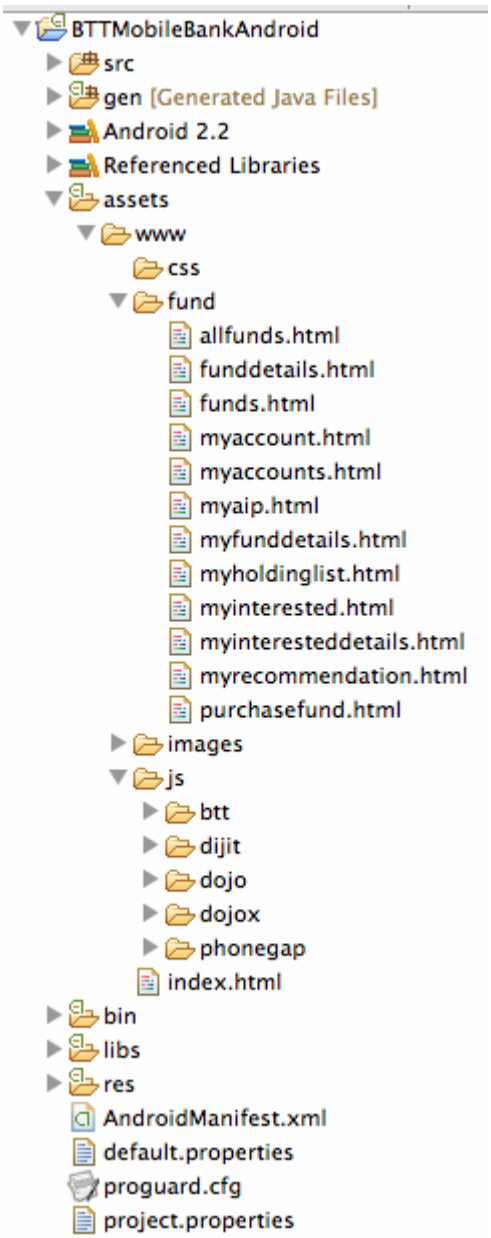
In the sample mobile apps, for both iOS and Android, we are using PhoneGap as native bridge, and Dojo as UI implementation. PhoneGap is just a wrapper to native platform functions, such as camera, accelerator … that are not used in this sample,

you can add them for your own app if you are interested in. Dojo is a opensource JavaScript framework, with many built-in functions and UI widgets, we are using v1.7 to implement the whole app UI and business process, please find more details at dojotoolkit.com.

For iOS app, its project structure is as below chart. “www” contains all UI and business logics. The “index.html” is the startup point of the app, contains some base views. The “btt.js” defined a global object with several global functions to speed up the integration with BTT web application. The “app.js” contains whole navigation logic, and is responsible to create some UI widgets dynamically. The “fund” folder contains several html files to separate UI views so that they can be loaded on demand dynamically to save memory and time.



For Android app, its project structure is as below chart. While, the core code under “www” is exactly same as the one in iOS project.



Index.html is the entrance of the app, it starts with standard HTML5 head, and added reference to dojo scripts.

```
<!DOCTYPE HTML>
<html>
<head>
<meta name="viewport"
    content="width=device-width,initial-scale=1,maximum-scale=1,minimum-scale=1,user-scalable=no" />
<meta name="apple-mobile-web-app-capable" content="yes" />
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta http-equiv="cache-control" content="no-cache" />
<meta http-equiv="pragma" content="no-cache" />
<title>IBM BTT Mobile Bank</title>
<!-- stylesheet will go here  -->
<link href="js/dojox/mobile/themes/iphone/iphone.css" rel="stylesheet">
<script type="text/javascript">
            var dojoConfig = {
                parseOnLoad: true
            };
        </script>
<script type="text/javascript" src="js/dojo/dojo.js.uncompressed.js"></script>
<script type="text/javascript">
            require([
                "dojo/ready",
                "dojox/mobile",
                "dojox/mobile/TextBox",
                "dojo/data/ItemFileReadStore",
                "dojox/mobile/Carousel",
                "dojox/mobile/IconItem",
                "dojox/mobile/FixedSplitter",
                "dojox/mobile/FixedSplitterPane",
                "dojox/mobile/ScrollableView",
                "dojox/mobile/Button"],
            function(ready) {
                ready(function() {
```

“btt.js” is the core functions collection to integrate with BTT web application. For example, the “getAccounts” function will post a xhr to the wsAccount_GetAccountSummary operation defined in web application, once the operation return the accounts, this function will call back to “success” function, or call “failure” in case of any errors.

```
    CurrentAccount : null, // current selected/displayed account obj
    CurrentFund : null, // current selected/displayed fund obj
    ProgressIndicator : ProgressIndicator.getInstance(),

    // Set this url pointing to web app context root
    ServerAddress : "http://9.186.122.215:9080/BusinessTemplateWeb",

    constructor : function() {
    },

    getAccounts : function(/*string*/username, /*func*/
            success, /*func*/failure) {
        console.log("entered global function");
        var params = {
            'dse_sessionId' : BttMobile.SessionId,
            'dse_pageId' : '1',
            'dse_operationName' : 'wsAccount_GetAccountSummary'
        };

        var xhrArgs = {
            url : BttMobile.ServerAddress + "/Ajax",
            handleAs : "json",
            postData : dojo.toJson(params),
            load : function(data, ioargs) {
                console.log("account summary ok", data,
                        ioargs);
                var accounts = data["return"].data.accountList.account;
                if (success)
                    success(accounts);
            },
            error : function(error, ioargs) {
                console.log(
                        "account summary query failed.",
                        error, ioargs);
                if (failure)
                    failure(error);
            }
        };
        var deferred = dojo.xhrPost(xhrArgs);
    },
```

“app.js” is the core navigation engine. It will listen to each “/dojox/mobile/afterTransitionIn” event, and execute business logics and UI changes according to the view id. For example, for “myAccountsView”, there is code to call the global function “getAccounts” mentioned above, if the account data is returned successfully, the call back function will iterate through each account object, and create ListItem for them, then add listening code to set the “CurrentAccount” when each account ListItem is clicked.

```
case 'myAccountsView':
    // initialize the accounts list
    var listItem;
    var list = dijit.byId("myAccountsList");
    if (list && list.hasChildren()) {
        list.destroyDescendants(false);
    }
    BttMobile
            .getAccounts(
                    null,
                    function(accounts) {
                        dojo
                                .forEach(
                                        accounts,
                                        function(
                                                account,
                                                i) {
                                            console
                                                    .log(
                                                            account,
                                                            "at index",
                                                            i);
                                            listItem = new dojox.mobile.ListItem(
                                                    {
                                                        url : "fund/myaccount.html",
                                                        sync : "false",
                                                        lazy : "true",
                                                        transition : "slide",
                                                        label : account.accountNo
                                                    });
                                            list
                                                    .addChild(listItem);
                                            dojo
                                                    .connect(
                                                            listItem.domNode,
                                                            "click",
                                                            listItem,
                                                            function(
                                                                    event) {
                                                                console
                                                                        .log("clicked list item");
                                                                console
                                                                        .log(event);
                                                                console
                                                                        .log(this.label);
                                                                BttMobile.CurrentAccount = account;
                                                            });
                                        });
                    },
                    function(error) {
                        alert("Query accounts failed, please retry.");
                    });
```

2.9 Conclusion

This part of document introduced a basic mobile app development approach, hybrid mode, to you, with which we can easily create mobile apps to interact with BTT multichannel web application, the work can be done and maintained at a minimum effort due to its great reusability and flexibility. This concept demonstrated the power of IBM WebSphere Multichannel Bank Toolkit and proved the value IBM is always trying to contribute to your banking business.