

Mobilizing Domino Applications onto Blackberry Devices

Part 1: Developing with MDS

Table of contents

Table of contents	1
1. Introduction.....	3
1.1. Why BlackBerry?.....	3
1.2. Architecture	4
1.3. Building a pilot environment.....	5
1.4. Which Development Environment to use?	5
1.5. About the author	5
2. Web Services	6
2.1. Introduction	6
2.2. Web Services and Domino.....	6
2.3. LotusScript web services in Domino 7.....	6
2.4. Testing your Web Service	8
3. Creating your first MDS Studio Application.....	8
3.1. Installing BlackBerry MDS Studio	9
3.2. Creating the MDS studio application	9
3.3. Using the Simulator	11
4. Improving the Application UI.....	13
4.1. Redirect the starting page and the "Done" button	13
4.2. Fix the screen titles.....	15
4.3. Insert our own graphics.....	15
4.4. Making the returned string read-only.	16
4.5. Getting rid of the "Message Sent" dialog.	16
5. Summary	17

1. Introduction

This is the first of three articles discussing mobilizing Domino applications onto BlackBerry handsets.

In this article, we shall discuss the fundamentals of publishing Domino data onto a BlackBerry handset, using one of the two development tools that BlackBerry provide – the BlackBerry MDS Studio.

In the next article, I shall show enhancing the application and the process of deploying the application to your corporate BlackBerries.

The last article will cover the more advanced BlackBerry development tool – the JDE.

During this article and the next article, we shall develop an example application together – a simple contact application to demonstrate these techniques. This application will allow the end user to view corporate contacts, and ultimately create new ones.

1.1. Why BlackBerry?

Why is this article focusing on BlackBerry handsets? Aren't there other mobile solutions in our corporate world, such as Windows Mobile devices, and Palm-based devices?

I firmly believe that the solution that BlackBerry provides is superior:

- Its far more secure. Data is transmitted in encrypted (and compressed) form, using user-generated keys, between your data centre and the handset. Should the handset be lost, a "Kill" command can be sent which will destroy all data on the device. In comparison, data held on other mobile platforms is easily retrieved.
- The handset itself is far more robust – just look at the second-user market on eBay for handsets, and you can see hundreds of older devices for sale. I have owned a windows handset in the past, and found it to be far more physically fragile. Something worth considering when choosing a handset that will live with you (and your users) 24x7.
- Ease of use. The people who use these devices are probably not computer experts, and wish a simple, clear end user interface that works all the time in a predictable manner. The BlackBerry experience provides that.
- Scalable. I know of customers who have rolled out tens of thousands of BlackBerry handsets to their users, who demand 24x7 service. Other mobile platforms require far more support in order to keep them going.
- Handsets. BlackBerry now have three physical handset formats, and seem to update these at a dizzying pace, adding camera, memory stick and now Wifi capability, whilst at the same time retaining the ease of use and battery life that our users demand. I do not for a second believe that this upgrade progress will slow down.

Many users firmly believe that BlackBerry can only provide eMail, contact and calendar solutions – this series of articles show how to mobilise business critical applications as well.

If you already have a BlackBerry Domino solution in place – these techniques will allow you to gain more value from your existing investment, and help deploy your applications to very influential end users.

1.2. Architecture

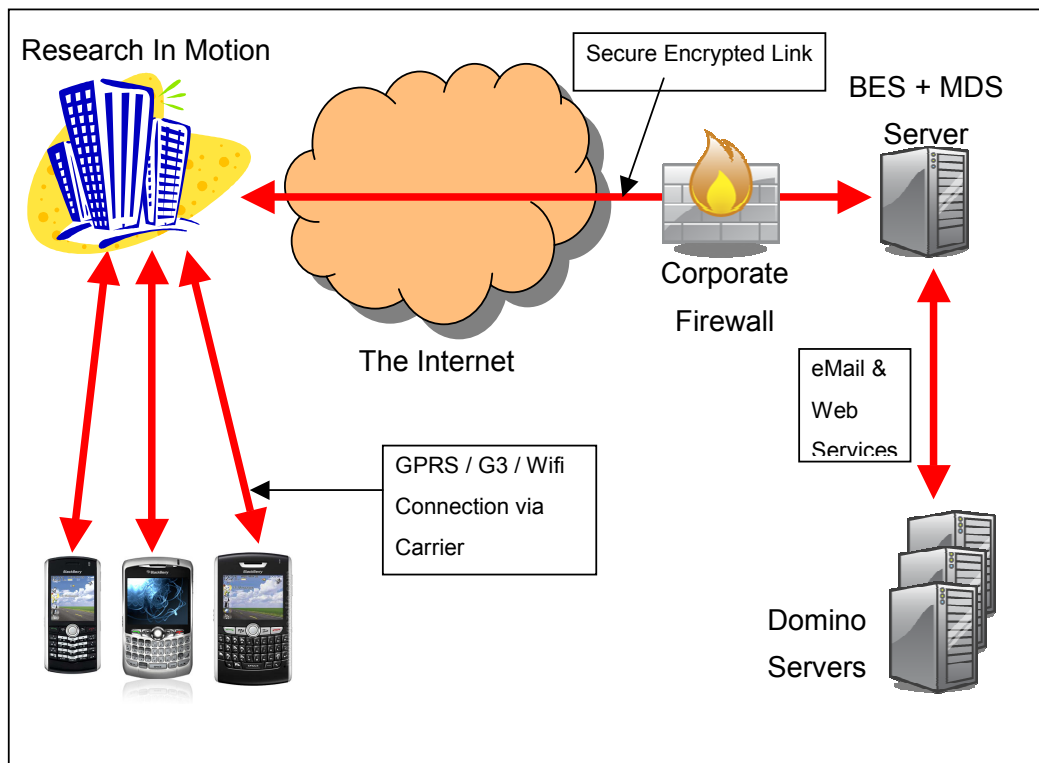
In order to appreciate fully the design decisions you must make, you must appreciate how the solution is architected.

At the core of the BlackBerry solution is the BlackBerry enterprise server ("BES"). This is a windows-based server which connects to your Domino environment – and indeed runs on a Domino server. This takes data from your corporate environment and pushes it out to the handsets.

Out of the box, the BES server provides you with email, calendar and contact (often referred to as "PIM") synchronisation.

An optional installation – Mobile Data Suite ("MDS") Server – allows you to leverage application connectivity as well as PIM information.

From a network point of view, the solution looks like:



So we can expose web services on our Domino servers, which are then consumed by the BES server and then passed to the handset via a secure encrypted link over the internet. The traffic is passed via RIM's servers which provide the capability to connect a corporate BES server to a handset regardless of carrier or country.

In effect, we now have a secure VPN link from the handset right back to behind our firewall, using encryption keys that are unique to our corporate.

In order for our Domino application to be exposed on our BlackBerry handset therefore we need to:

- Web service enable our Domino Application
- Write an application for the BlackBerry that consumes that web service.

Thankfully, BlackBerry MDS studio provides us with a development environment that mimics this connectivity, as well as providing us with a simulator which

allows us to write, deploy and test applications on a single development machine, without any of this infrastructure being in place.

1.3. Building a pilot environment

An important part of proposing this as a solution to management is to understand the costs of running this architecture. The initial costs for a pilot may preclude management buy-in. This section helps you understand the costs of each of the components, and how to minimise these for a pilot.

The BES server itself is an expensive software component, and needs to run either in a virtual machine image (such as Vmware) or on dedicated hardware. This will require a license for the operating system and a Domino server license. BES itself is sold as a software package, and you require a CAL for each BlackBerry handset you wish to connect to this.

BlackBerry are running a promotion at the moment for Lotus Domino users where you can download and run a free 10-user BES server in your environment – go to <http://www.blackberry.com/products/software/server/domino/quickstart.shtml> .

Each handset then costs money – you have to obtain handsets from your phone carrier, who will also sell you BES software and CAL's. And lastly, your phone needs to come with an air-time plan that includes BlackBerry connection as well as data. For example, here in the UK, T-Mobile offer BlackBerry service with your airtime package for only £9/month extra.

Should you require a large number of handsets you may find that your existing corporate mobile phone provider is very flexible in terms of pricing, and may even provide test hardware in order to demonstrate a pilot environment.

Second-user handsets are of course available via eBay, and this route forms an invaluable source of cheap handsets suitable for testing, etc.

1.4. Which Development Environment to use?

Both the BlackBerry MDS Studio and JDE development environments allow you to develop applications that will run on BlackBerry handsets. But which one is suitable for your application development requirements?

The MDS Studio relies on the MDS server hosted on your BES, and provides a virtual VPN between your handsets and your corporate infrastructure. These applications are then pushed out (over the air) to your users. MDS users have to therefore be connected to BES servers that you control.

This in turn means that MDS studio should only really be used for applications that will be used by corporate users in your environment.

JDE allows you far more granular control over your applications, and allows you to build far more complex applications – ones that can store data on the handset and connect to general services. These are more appropriate for developing applications that you might wish to provide for any user, regardless of how they are connected.

The first two articles in this series will focus on the BlackBerry MDS Studio development environment.

Both development environments are free and can be downloaded from <http://na.blackberry.com/eng/developers>.

1.5. About the author

Bill Buchan is a veteran and long-standing member of the Lotus Domino community, and regularly presents on stage at Lotusphere, The View or the

Irish/Scottish/UK user group conferences. If you've ever seen a man in a kilt at a Domino event, chances are it's him. Come up and say "Hi" – he's not as scary as he looks.

Outside of work (those precious few minutes!) he can be found being bossed around by his wife ("She Who Must Be Obeyed") and daughter ("Squid"), and tinkering with his beloved motorbike.

He's a founder of [HADSL](http://www.hadsl.com) and his personal blog can be found at <http://www.billbuchan.com>, where all his presentations and articles can be found.

2. Web Services

Web services are used to expose our Domino data to the MDS studio. So what are web services, and how can we construct them?

2.1. Introduction

Web services are a platform and language independent application to application data connectivity methodology. In general, this means that one application – the web service server – will allow web service consumers to submit queries and receive answers. The data transport mechanism is usually (but not restricted to) http-style traffic, and the data is formatted using well formed and well defined XML constructs. SOAP – Simple Object Application Protocol – is a well known implementation of such a service.

Remember – web services are an industry wide standard and as such, lots of resources are available for you outside the Lotus Domino environment. Similarly, you may find that other applications within your environment can serve and consume web services – so this methodology may be more beneficial than just providing BlackBerry handsets with data.

2.2. Web Services and Domino

There are a number of different ways to provide a web service server in Domino. We shall focus on the simplest approach which is the ability of Lotus Domino 7 to provide a web service based on LotusScript code that you write.

Other methods include writing Java Servlets, or writing simple LotusScript agents. These approaches may be more appropriate if you have to host your application on Domino servers older than ND7. These other approaches and their strengths and weaknesses are discussed in a web article:

<http://www.billbuchan.com/web.nsf/d6plinks/BBUN-75HDA4>.

2.3. LotusScript web services in Domino 7

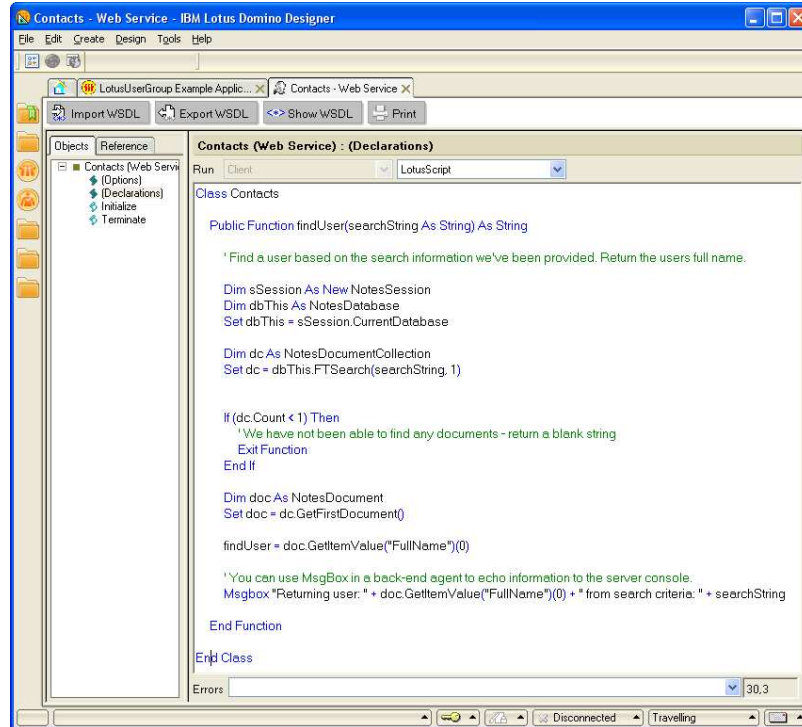
Writing a web service in Lotus Domino 7 is simple. You:

1. Create a new Lotus Notes Database
2. Open this database in domino designer and go to "Shared Code, Web services"
3. Click on the "New" button
4. Enter some LotusScript code in the "Declarations" Section. Note that this code has to be contained in a class, and has to have one or more "public" functions that return simple data types (string, date, number, etc). In the next article, we shall discuss returning more complex data types.

In our case, we shall create a class called "Contacts", and at this stage, just create a single function called "findUser".

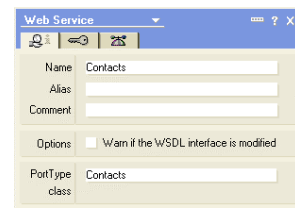
This function takes a single string as a parameter and uses this string to search the database for a record.

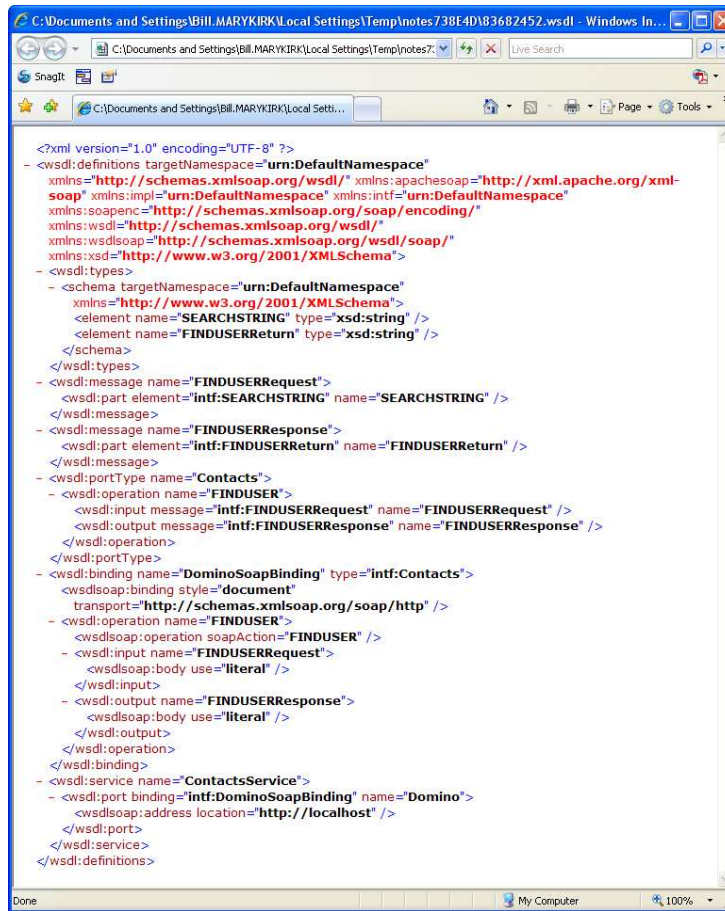
If it finds a Notes document, it will return the users name as a string.



We need to remember to save this Web service. During the save process, we will be prompted to fill in the class name that we wish to expose to the Web service in the "PortType class" field. In this case, we have put the class name "Contacts" in this field.

Now save the web service.





```

<?xml version="1.0" encoding="UTF-8" ?>
- <wsdl:definitions targetNamespace="urn:DefaultNamespace"
  xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:apacheSOAP="http://xml.apache.org/xml-soap"
  xmlns:impl="urn:DefaultNamespace" xmlns:intf="urn:DefaultNamespace"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
- <wsdl:types>
  - <schema targetNamespace="urn:DefaultNamespace"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <element name="SEARCHSTRING" type="xsd:string" />
    <element name="FINDUSERReturn" type="xsd:string" />
  </schema>
</wsdl:types>
- <wsdl:message name="FINDUSERRequest">
  <wsdl:part element="intf:SEARCHSTRING" name="SEARCHSTRING" />
</wsdl:message>
- <wsdl:message name="FINDUSERResponse">
  <wsdl:part element="intf:FINDUSERReturn" name="FINDUSERReturn" />
</wsdl:message>
- <wsdl:portType name="Contacts">
  - <wsdl:operation name="FINDUSER">
    <wsdl:input message="intf:FINDUSERRequest" name="FINDUSERRequest" />
    <wsdl:output message="intf:FINDUSERResponse" name="FINDUSERResponse" />
  </wsdl:operation>
</wsdl:portType>
- <wsdl:binding name="DominoSoapBinding" type="intf:Contacts">
  <wsdlsoap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  - <wsdl:operation name="FINDUSER">
    <wsdlsoap:operation soapAction="FINDUSER" />
    <wsdl:input name="FINDUSERRequest">
      <wsdlsoap:body use="literal" />
    </wsdl:input>
    <wsdl:output name="FINDUSERResponse">
      <wsdlsoap:body use="literal" />
    </wsdl:output>
  </wsdl:operation>
</wsdl:binding>
- <wsdl:service name="ContactsService">
  - <wsdl:port binding="intf:DominoSoapBinding" name="Domino">
    <wsdlsoap:address location="http://localhost" />
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

Web service consumer applications can ask the Web service to describe itself – it does so by passing back a well-formed piece of XML using a standard called the Web Services Description Language.

We shouldn't worry too much about this, as the Domino 7 Web Services design element constructs this for us. In order to view the WSDL, click on the "Show WSDL" button. You should see something like this.

Congratulations, you have just created your first Lotus Domino web service!

2.4. Testing your Web Service

We can now test this web service.

A good initial test is to try and open the WSDL using a web browser. Do this by opening a web browser, and typing in the URL for the web service. For instance, if our test server is called "Server", our database is called "contacts.nsf" and we have named our web service "Contacts", then the URL will be:

<http://server/contacts.nsf/Contacts?WSDL>.

In terms of more comprehensive testing, I prefer using a Web Service testing package called "SoapUI", which can be downloaded from: <http://www.soapui.org/>. This package allows you to examine all web service calls provided and push test data into the web service quickly and easily.

3. Creating your first MDS Studio Application

We can now start constructing our first BlackBerry MDS application.

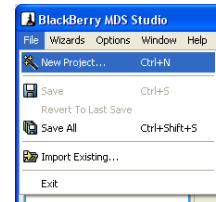
3.1. Installing BlackBerry MDS Studio

Download and install BlackBerry MDS studio from <http://na.blackberry.com/eng/developers>. If in doubt, select defaults for all parts of the installation process.

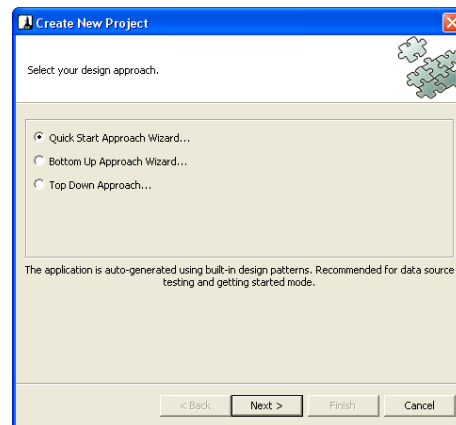
3.2. Creating the MDS studio application

We can now create the MDS application:

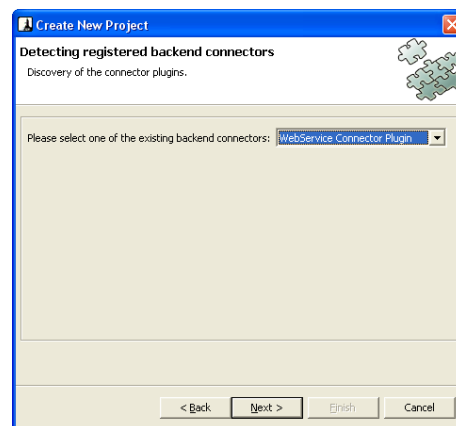
1. Within MDS Studio, click "File, New"



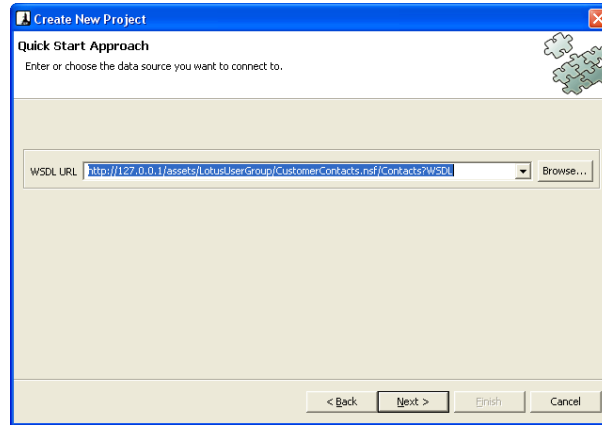
2. Select "Quick Start Approach"



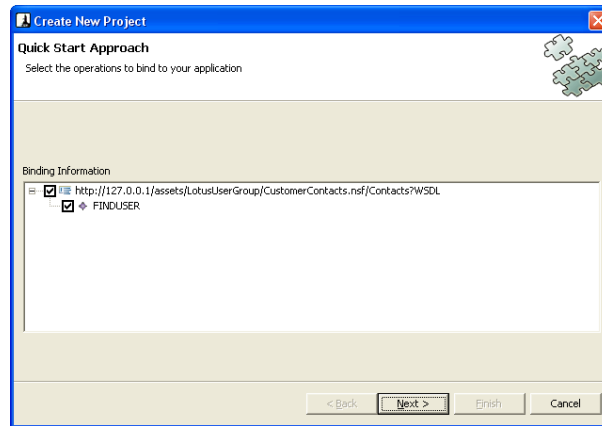
3. Select "WebService Connector Plugin":



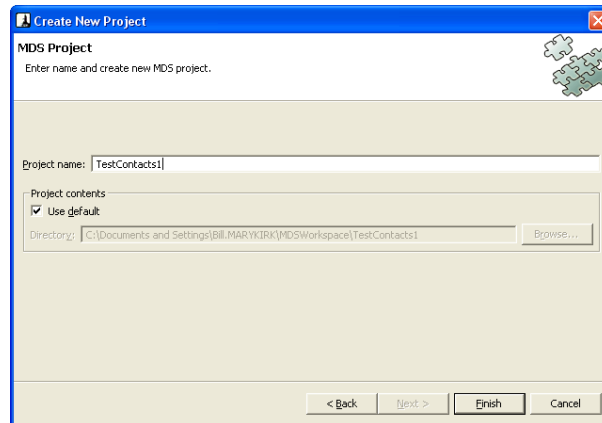
4. Enter the URL for the Web Service WSDL:



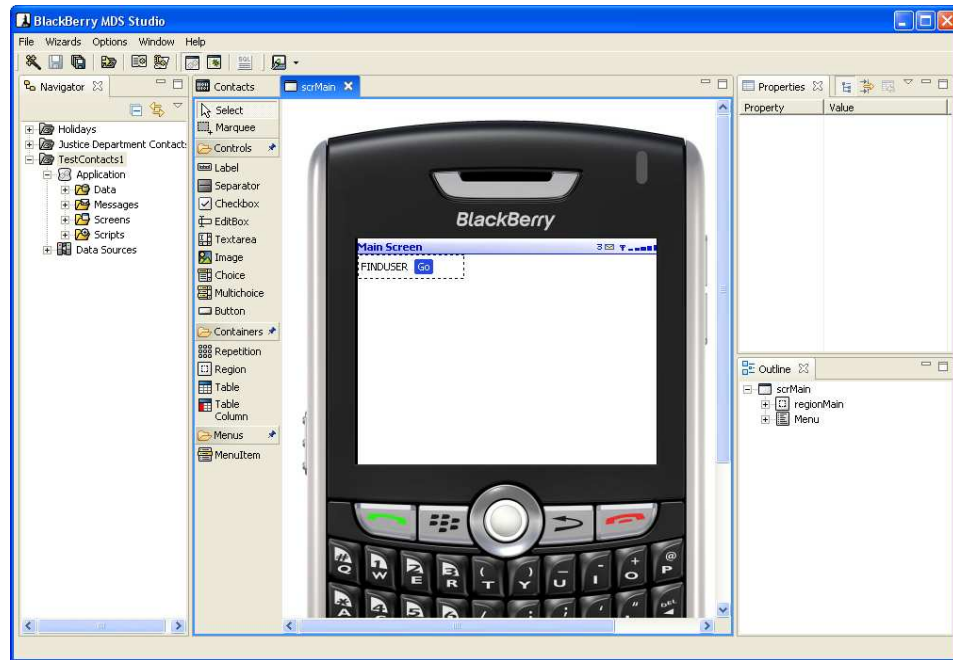
5. Select all the web services you wish this application to consume. In this case, there is only one service:



6. Lastly, give the Application a relevant name. In this case, we shall call this application "TestContacts1".



7. The MDS Application has now been built, and looks like this:

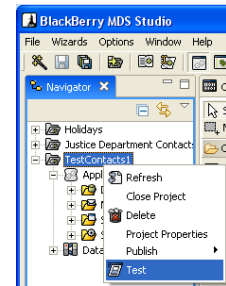


Without writing a line of code, we now have a fully featured (if slightly graphically challenging) MDS application which consumes our web service.

3.3. Using the Simulator

We can then test this application using the inbuilt BlackBerry simulator.

Do so by right-clicking on the application name on the left hand object browser, and selecting "Test".



This will then load the test environment and BlackBerry simulator, as well as compiling the application and loading it into the simulator.



The simulator looks like this: Note that I've selected the BlackBerry Pearl (8100) as the target device. The Pearl has a slimmer screen than its larger Curve (8300) or 8800 handsets, and therefore its useful to map your application onto this screen size.

In terms of using the simulator, use the scroll wheel on your mouse to emulate the old "scroll wheel" on the BlackBerry handset itself, and click on the thumbwheel for "Enter".

As you can see, the application "TestContacts1" has been added to the handset workspace, so it's just a case of selecting it, and pressing Enter:

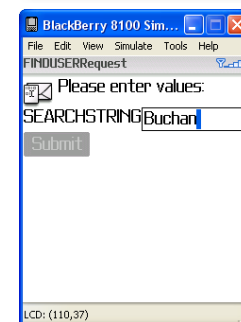


We can then view the application:

The opening screen looks like this. Select the "Go" button and press "Enter".

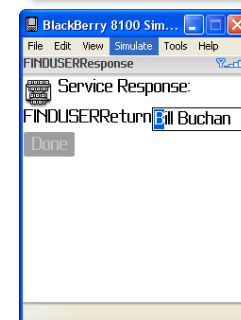


Selecting "Go" produces this screen. As I have a single test record in my database with my name, I choose to search on the string "Buchan".



When I click Submit, an annoying "This message has been sent" dialog pops up for a second, and then I'm presented with the results of my web service operation:

So as you can see, we have now created a Domino Web Service, and created an MDS Application which consumes



that web service. All without writing a single line of code in MDS Studio, and only 30 lines of code in LotusScript!

4. Improving the Application UI

As previously noted, the application UI, whilst functional, is somewhat challenging.

- We seem to have an unnecessary screen at the start. This might make more sense later on when we have more functions available to us, but at the moment, this is just one more screen to click through.
- The titles of the screens are pretty ugly.
- The layout of each page could be improved.
- The result from the web service is editable, and the labels to the left of the return text mean that the effect is somewhat squashed.
- We'd like to put our own graphics in there.
- There's an annoying "Message sent" dialog that pops up.

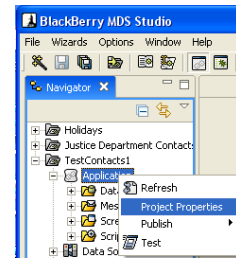
Let us therefore fix all these issues. This exercise will give you a pretty good journey around the MDS studio IDE.

Like all modern IDEs, you have an object browser on the left, some properties boxes down the right, and clicking on an item lets you view or update these properties.

4.1. Redirect the starting page and the "Done" button

We can choose which screen the application opens with by changing the "Entry Point" value on the Project Properties dialog. To do this:

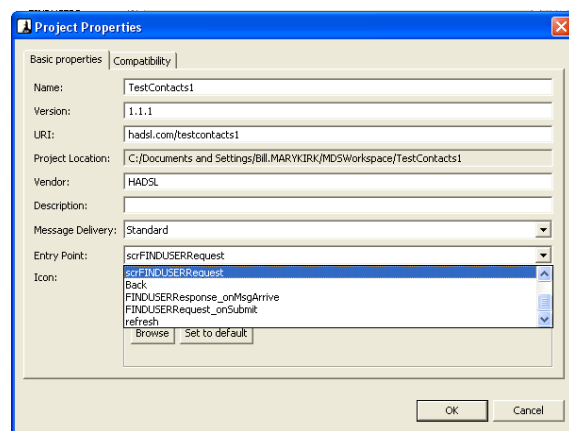
1. Right click on the application and choose "Project Properties"



2. Now select the new entry point screen on the "Entry Point" drop-down box.

In this case, we shall choose the screen that the user enters his search text – `scrFINDUSERRequest`.

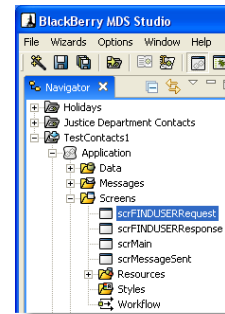
It should be noted that you can select both screens and JavaScript routines from this box, so take care to enter the correct screen name.



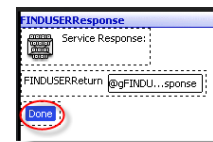
3. And click on OK to save and close this dialog.

The second part of this exercise is to change the button on the results page so that when the user clicks on the button, it now goes to the same screen. To do this:

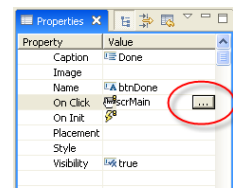
1. Open the screen scrFINDUSERResponse



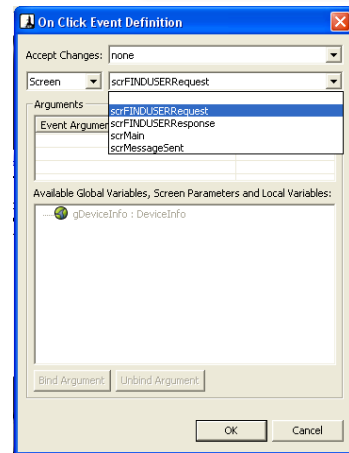
2. Select the "Done" button



3. On the properties pane on the right hand side, click on the button beside the OnClick event:



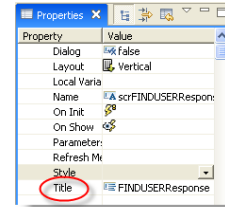
4. On the dialog, keep the first selection as "Screen", and then choose the scrFINDUSERRequest screen from the drop-down list.
5. Press OK to confirm these changes.
6. Now press the "Save" icon on the toolbar.



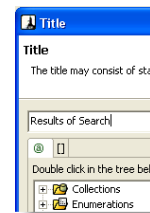
4.2. Fix the screen titles.

In both of our user facing screens, the screen titles are automatically generated, and therefore look fairly unfriendly. Lets go change these by:

1. Open each screen
2. Click on an area on the screen that doesn't contain any screen controls. This will ensure that you have selected the screen itself
3. On the properties box on the right, click on the Title property.



4. On the dialog screen, change the "Title" text to something more relevant.
5. Click on OK to save the new title.
6. Save each screen by pressing the "Save" icon on the toolbar.



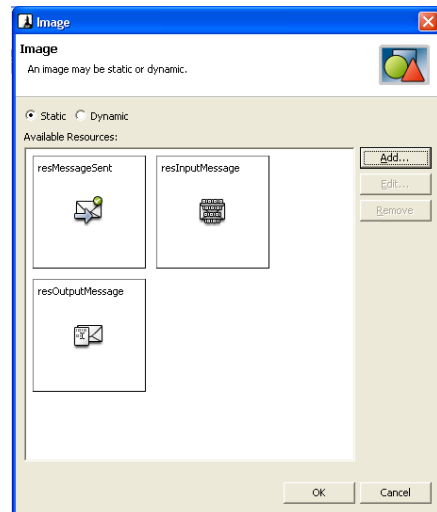
4.3. Insert our own graphics.

Thankfully, we can insert our own graphics to give the application a little more of a corporate feel. In this case, we shall put a header graphic on each page.

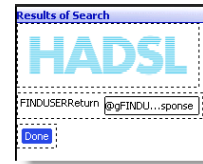
1. Open each screen
2. Click on the "Service Response" text and press delete
3. Click on the icon and press delete.
4. You should now see a large square empty box in the GUI – this is a "layout region". Click on the "image" icon on the left and drag it onto the layout region.



5. You should now see the Image Add dialog box. This dialog allows us to add new images to this application as well as selecting pre-installed images. In this case, click on the "Add" button and select an image from your hard drive. Once you have added a new image, click on it, and press OK.



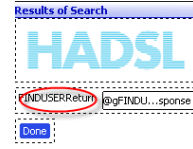
- The image will then appear on the screen for the application.
- Now click on the "File Save" icon to save each screen.



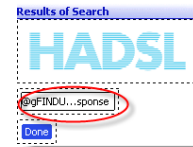
4.4. Making the returned string read-only.

When we return the users full name within our application, its returned as an editable string in a hard box, and has a label to the left hand side. This results in the string being wrapped, and looking pretty ugly. Lets go fix this:

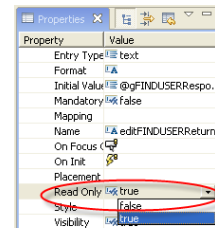
- Open the screen scrFINDUSERResponse
- Click on the label text to the left of the return value.
- Press delete. The label should now disappear.



- Click on the returned string field



- On the properties box on the right hand side, change the "Read-only" property to "true"

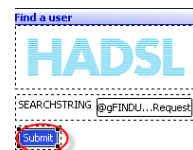


- Click on the "file save" icon to save this screen.

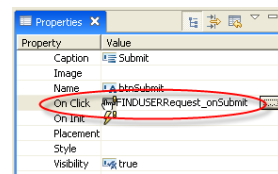
4.5. Getting rid of the "Message Sent" dialog.

When a user submits a new search, a dialog box pops up and says "Message Sent". This is quite annoying after a while. So lets go remove it:

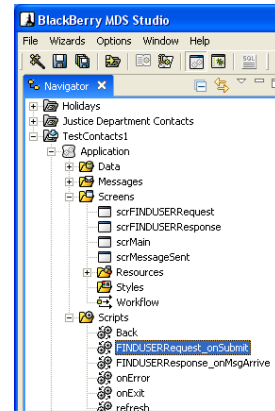
- Open up the screen scrFINDUSERRequest
- Click on the submit button



- On the properties box on the right, you can see the name of the script that this button will run. In this case, it'll run the FINDUSERRequest_onSubmit script.



4. Open the script:



5. As you can see, the second line of the script runs an event – `scrMessageSent.display()`. This javascript command literally runs the display event in the screen `scrMessageSent` – so as you can see, you can call other screens from script events within the MDS Application. Highlight this line, and press delete.
6. Click on the "File Save" icon on the toolbar to save this script.

As you have seen from this section, BlackBerry MDS studio is a fairly standard IDE and shouldn't take long to become familiar with.

5. Summary

We've been on a whistle-stop tour around the whole BlackBerry enterprise development environment in this article, and got to the stage where:

- We can expose some very simple application constructs using BlackBerry MDS studio.
- We can compile, deploy and test these applications
- We can edit screen and script objects within MDS studio.

So we can quickly build and test some very basic applications.

In the next article, we'll move on to passing complex class structures from your Domino web services into the MDS studio, so you can start passing lists of user names, or a complete person contact record.

We'll also approach the whole issue of actually deploying these applications onto handsets.