Magic University

Going Mobile with Magic xpa 3.x

Self-Paced Tutorial

Book ID: UTGMMXA30

Edition: 1.00, February 2015

Course ID: UCGMXPA3x

Magic University Official Courseware

Going Mobile with Magic xpa 3.x

February 2015

Contents

Introduction 1

- About Magic xpa 1
- About the Seminar 2
- Seminar Prerequisites 2
- Magic Software University 2

Setting Up the Mobile Clients 3

- Mobile Design Mode 3
- My First Magic Program 4
- Mobile Preview Pane 4
- Execution Properties File 5
- Running the Program on a Mobile Device 6
- How Does It Work? 8
- Executing a Program Directly from the Studio (Android only) 8

Understanding the Client 11

- About Mobile Devices 12
- Magic xpa on a Mobile Device 13
- Developing in RIA 13
- Form Considerations 14
  - Window Types 14
  - Form Size 14
- Placement 15
- Supported Controls 20
  - Edit Controls 21
  - Image Controls 21
  - Two-State Images 21
  - Button Controls 21
  - Tab Controls 21
- Application Navigation 21
  - Navigation 21
  - Termination 22

Design Considerations 22
Status Bar 23
Colors and Fonts 23
Colors 23
Fonts 23
Adding a font manually 25
Summary 26
Advanced Controls 27
Tables 28
Landscape Mode 28
Mobile Events 29
Selection Lists 30
Selection Program 30
Selection Controls 30
Menus 30
Summary 31
Interacting with the Device 33
Mobile File System 34
Querying Device Characteristics 34
Finding the Device Location (GPS) 35
Camera Support 36
Gallery Support 36
Accessing the Mobile Devices' Capabilities 37
Summary 38
Offline Implementation 39
Concept 40
How Does It Work? 41
Summary 42
Customization and Installation 43
Execution Properties File – Additional Information 44
Customizing the Application 44
Keystore File 46

APK File 46
Installing the Client on the Android Device or Emulator 47
Summary 48
Preparing the Testing Environment 49
Defining a Simulator 50
Android 50
Android SDK Manager 50
Android Virtual Device Manager 50
Installation on the Device 51
Using the Keyboard on the Device Simulator 52
Troubleshooting 52
Mobile device fails to communicate with the Web server 52
Execution properties file defined in the URL dialog is wrong 53
Debugging 53
Android 53
iOS 54
Summary 55

Introduction

Welcome to Magic Software University's **Going Mobile with Magic xpa 3.x** self-paced tutorial. We, at Magic Software University, hope that you will find this guide informative and that it will assist you in getting started with this exciting product.

About Magic xpa

Magic xpa provides all aspects of the application development and deployment process within a single end-to-end platform. It features a ready-made business application engine that simplifies the code-writing process and enables you to deploy to market faster, using fewer resources.

Applications developed using Magic xpa typically have fewer coding mistakes, undergo more thorough prototyping, benefit from greater business-side input and optimization, and can be more easily adapted to changing business needs.

Magic xpa enables you to focus more on the business logic of the application and less on what is happening behind the scenes.

Install Magic xpa 3.x.

Please ensure that your computer meets the hardware requirements listed on the following page.

If you encounter any problems, contact msu@magicsoftware.com.

About the Seminar

The course is intended for people with a working knowledge of developing RIA applications who want to know how to successfully develop applications for use on mobile devices while using Magic xpa.

During the course you will learn about how Magic xpa works with mobile devices.

By the end of the course you will have created programs that demonstrate the ease of use of the Magic xpa and mobile integration.

Seminar Prerequisites

Before you start with the seminar there is basic knowledge that you need to have:

Development knowledge	Working knowledge of Magic xpa while developing RIA applications.

Your computer must also meet some basic requirements:

Hardware and Software	Windows 7 and later. • You will need IIS 7 installed on your computer. • .NET framework V4.0 (or above) installed on your computer. • Magic xpa 3.0 or later. • A mobile device. • The Magic xpa client installed on the mobile device. You can download it from the Google Play store or from the App Store.

Magic Software University

MSU offers various courses that may be of interest to you. To see the current offering, open the following link:
http://www.magicsoftware.com/en/services/?catID=59&pageID=45&subPageID=756

Lesson 1

Setting Up the Mobile Clients

Before we start discussing the development paradigm of mobile applications and the differences between developing a desktop and a mobile application, we will start by setting the mobile environment, so that you will be able to run a Magic application on your mobile device.

Mobile Design Mode

When designing a desktop application, the physical screen is normally a landscape-type screen, meaning that the width is greater than the height. However, on mobile devices this is not the initial mode. The initial mode is portrait-type in which the height is greater. You need to take this issue into consideration. As an example, a portrait form will not enable you to display a long row of text. You will need to consider using a multi-edit box with word-wrap. In addition, the device size is much smaller than desktop applications and the controls are larger.

Magic xpa allows you to easily develop mobile applications by using a Mobile Design mode.

When the Mobile Design mode is enabled and controls are added to the form, some of the properties will be set with different default values, which better support the mobile platforms.

You activate this feature by clicking the Mobile Design Mode button from the toolbar or by opening the **Options** menu and selecting the **Mobile Design Mode** option.

The following properties differ from the non-Mobile Design mode:

- The screen size is smaller. This is because it is recommended to design a screen for the smallest device and use placement to increase the size of the controls for larger devices.
- The size of controls is larger.
- The color of the table is defined according to the table color and not the column color (in the Set Table Color property).
- The Table control will not show scrollbars.

- The Group control will not have a default text.
- The Radio control will have a button appearance.
- The Line control placed in a container control will be stretched horizontally to fill the control.
- The width placement of the Edit, Group and Line control will be set to 100%. Therefore, by default, these controls will be resized.
- Tab controls will be placed on the entire form area.
- An Image control will have a different size when dropped on the form or in a Table control.

My First Magic Program

We'll start with a simple "Hello World" program:

1. Activate the Mobile Design mode as explained above.
2. Open the **demo** application.
3. Create a program named **Hello World** and give it a **Public name** of **start**.
4. Select the **External** box.
5. This is a RIA program, so in the **Task Properties** dialog box, set the **Task Type** to **Rich Client**.
6. Add a Virtual variable named **Sample text**, which will be **Alpha** with a length of **20**. Provide an initial text of **hello world**.
7. Zoom into the Form Designer.
8. Now drop the **Sample text** variable on the form.
9. If the Mobile Preview pane is not visible, it is advised to show it as detailed below.

Mobile Preview Pane

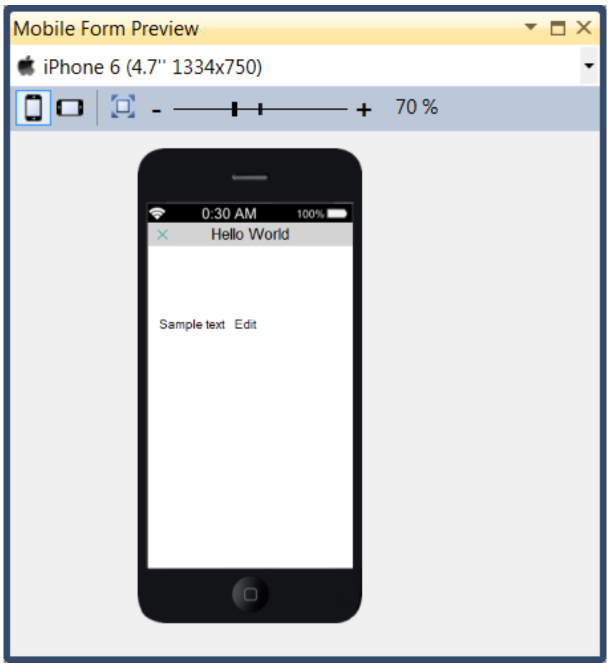
When developing for mobile platforms, you can preview the Display forms that you are developing by using the **Mobile Form Preview** pane. You access this by clicking the mobile button 📱 from the toolbar or by opening the **Form** menu and selecting the **Mobile Preview Pane** option.

This lets you play around with the placement and size of the controls and see how the controls will appear on various mobile devices. The form and control properties related to placement and size are the properties that are supported for this preview pane.

From the floating pane you can select which mobile device you want to preview, such as iPhone 6. Next to the name of the device, you will see the dimensions of the device.

The options that appear here are defined in the Magic.ini file's **[MAGIC_DEVICES]** section or in the Mobile Device repository, which you can access from **Options > Settings > Mobile Devices**.

You can also change the orientation of the device and change the zoom level.



Controls that are not supported for mobile devices, appear as a crossed out rectangle:
.

The program is now ready for execution.

This is a Rich Client program, so if you execute the program using the F7 key, it will execute in the RIA environment. So, to execute the program, you need to run the project.

10. Run the project (**CTRL+F7**).

The project is now ready to be executed on your mobile device.

Execution Properties File

To execute the application on your mobile device, you need to tell the client where the server is located and which application and program need to be executed.

This is done by defining an **execution.properties** file.

This file can be later be embedded in your client or placed on your server accessed you're your client using a URL.

Follow these steps to define the **execution properties** file:

1. Copy the **devprops.txt** file from the **RIAModules\Android** folder to a folder on your server that has an IIS alias so that it will be exposed for use. You can use for example the **Scripts** folder under the Magic folder, which is exposed by default.
2. Open the file in a text editor and update the properties as follows:

<properties>	
<property key="protocol" val="http"/>	The protocol used. Leave this as **http**.
<property key="server" val="**10.1.10.75**"/>	Your server. It is recommended to use the IP.
<property key="requester" val="**MagicScripts/MGrqispi.dll**"/>	The address of the requester. Make sure that the alias name refers to the **scripts** folder.
<property key="appname" val="**demo**"/>	The application name. Write the **demo** value or your application name if you used another application.
<property key="prgname" val="**start**"/>	The public name of the program that should run upon execution. Write the **start** value or your start program's public name if you used another name.
</properties>	

The properties file does not need to have the txt extension. You can use any extension you choose as long as you set the MIME type on the server so that the mobile devices will be able to open the file.

It also does not need to be called devprops.

Running the Program on a Mobile Device

The first step in running your application on the mobile device is to install the client application on the mobile device.

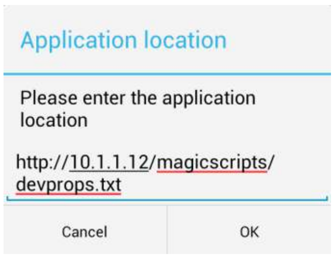
At this stage, we will use a generic Magic xpa client application available on the Google Play store and the App Store.

Of course you can create your own client application with your own application name and logo. We will discuss this later on in the course.

1. After you install the client application on your device, start it by double clicking on the application icon.

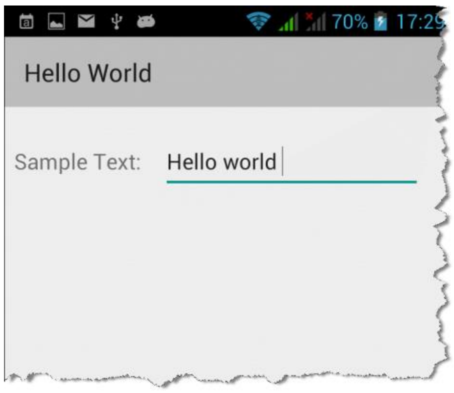
When the program is run initially, you will receive the following dialog box, requiring you to enter the URL of a details file. You will also receive this dialog box if the URL file is not found or accessible. (This dialog box will not appear for your users in your customized application.)

2. You can enter the following location (assuming you used the **magicscripts** alias): **http://<ip address>/magicscripts/devprops.txt**.


Application location
Please enter the application location
http://10.1.1.12/magicscripts/devprops.txt
Cancel
OK

Note: Make sure that the mobile device has access to this URL. In most cases, the development server is not exposed to the outside world. In these cases, the mobile device should be connected to the same network as the development server using WiFi. You can easily check it out by opening a browser and using the address above as the URL. You should see the file's content.

3. After writing the URL, click **OK** and the **Hello World** program will appear:



Congratulations. You have just created and executed your first program for a mobile device.

? If you want to run another program as the start program, how will you implement this?

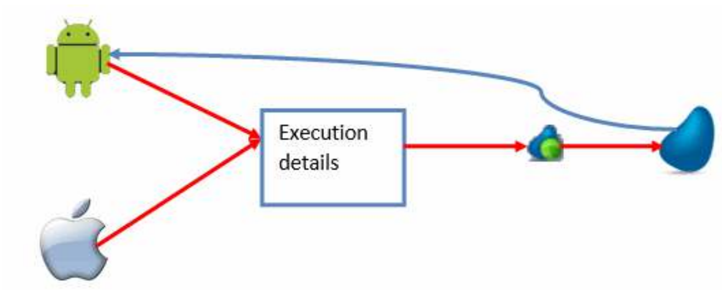
There are a few ways of implementing this, such as:

- In the Program repository, remove the **start** public name from the **Hello World** program and add it to the requested program. Do not forget to select the **Expose** option. This is a very simple method.
- Add a public name to the requested program, such as **orders** and in the **devprops** file, change the *<property key="prgname" val="**start**"/>* to *<property key="prgname" val="**orders**"/>*
  Note: It is not always feasible to use this option and you will understand why later in this course.

- Execute it directly from the Studio (on Android devices) as explained later on.

How Does It Work?

As this is a Rich Client application, it works in a similar manner as other RIA applications.



When the mobile client application loads it searches for the execution details. The execution details file is retrieved in one of the following two methods:

- Opening a dialog box to receive the URL of the file as you did in this exercise.
- Embedding the details with the client in an **execution.properties** file. You will learn more about this in the lesson about customization.

The client reads the content of the file to find the address of the server, the name of the application and the name of the initial program. This is the reason you were instructed to define the name of the application as **demo** and the program as **start**.

The process now behaves like any other Rich Client application.

Executing a Program Directly from the Studio (Android only)

You can run a program or project on an Android device or simulator directly from the Studio by first pressing in the **Execution on Android** toggle button. You can do this from the Magic xpa toolbar or the **Debug** menu.

Once the button appears pressed in, you can execute your program or project on an Android device directly from the Studio:

1. Press **F7** to execute a Rich Client program. This will switch the engine to Runtime mode and launch the application on the mobile device or simulator using the selected program as the start program.
2. Press **Ctrl+F7** to execute the project. This will switch the engine to Runtime mode and launch the application on the mobile device or simulator using the start program that is defined in the execution properties file.

For this to work, you should first:

1. Install the application on the mobile device or simulator.
   You can use the client from the Google Play store or install the generic **MagicDev.apk**, which is available in the **RIAModules\Android** folder. Refer to the **Installing the client on the Android Device or Emulator** section in the **Customizing the Application** lesson.
2. Connect the device to your network so that the application will be able to connect to the Magic xpa server to run the project.
3. From the **Android settings** menu, enable the **USB Debugging** option.
4. If you are using a device, connect the device to the PC using a USB cable.
5. In some cases, you will need to install the generic Google USB driver (from the Android SDK Manager, install the **Google USB Driver** package from the **Extras** folder) or the driver that came with the device.
6. If you used a custom client or the store client – Define the application package name in the **Run.bat** file located in the **RIAModules\Android** folder under the installation folder.

   To do it, open the file using a text editor and change the value of the **PackageName** variable to your package name, as it is defined in the **settings.properties** file.

   By default, the value is set to **com.magicsoftware.magicdev** in order to run the **MagicDev.APK** client.

   If you are using the store client, set the name to **com.magicsoftware.richclient**.

To check if your PC can access your device, open the command prompt, navigate to the **RIAModules\Utils\ADB** folder under the installation folder and execute the following command: **adb devices**.

You should see your device in the devices list.

Note: If the server on which Magic xpa is running is not defined in DNS, the Android mobile client will fail to access the Magic xpa server, since the device cannot translate the machine name to the IP address and you will get the following error: "Unable to resolve host name". If you encounter this error, you need to set the full URL with your machine's IP address in the **HTTP Requester** environment setting. For example: http://192.168.0.111/magicscripts/MGrqispi.dll.

Usually, when using the Genymotion simulator, you can define the following value as the IP address: http://192.168.56.1/magicscripts/MGrqispi.dll (192.168.56.1 is the default host IP address that the Genymotion VirtualBox supplies).

Lesson 2

Understanding the Client

Developing mobile RIA applications using Magic xpa requires the same skill set as developing desktop RIA applications. However, since the user interface and expected user experience are significantly different from a desktop computer, there are important differences that need to be taken into account when designing the application screens and planning the user interaction.

Devices differ in screen size, fonts, expected interaction device features (such as a camera and GPS), security related features and more. This lesson will give you a better understanding of the client devices.

This lesson covers various topics including:

- Form considerations
- Design considerations
- Application navigation
- Supported controls

About Mobile Devices

When you start developing for mobile devices, you will notice that there are a number of characteristics that you will need to take into consideration.

iOS and Android devices each have different characteristics when it comes to displaying information and enabling the user to enter information. You will learn more about this during the seminar.

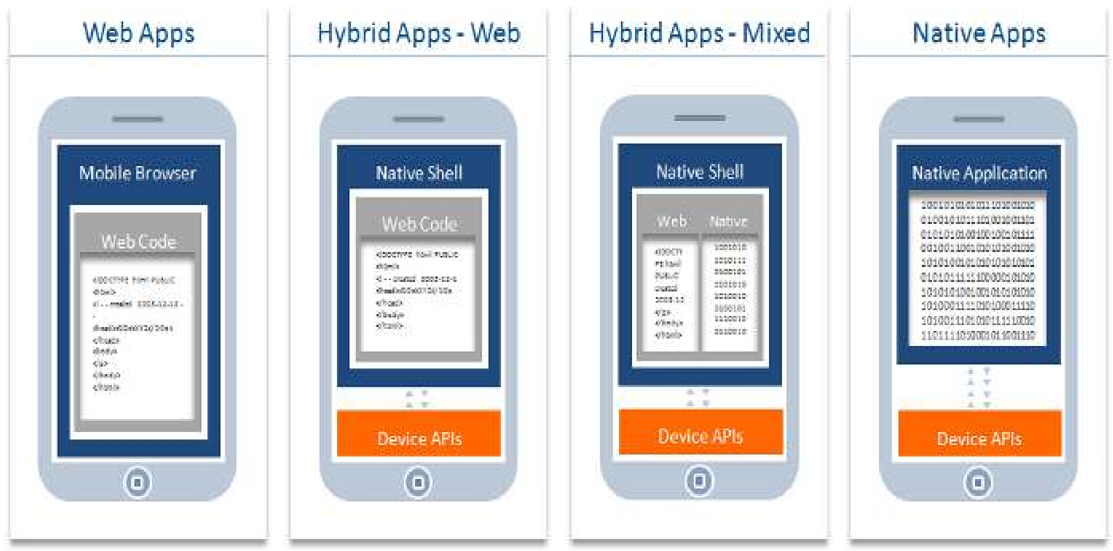
- Screen sizes and orientation – Mobile devices have various resolutions and screen sizes in both landscape and portrait orientations.
- Keyboard devices – Some mobile devices have a full QWERTY keyboard. In addition to a keyboard, some devices have a dedicated Menu key, an Esc key and a trackpad or trackball that are equivalent to the desktop keyboard arrow keys. The trackpad also provides a dedicated Fire action when pressed. Keyboard-only devices have a fixed screen orientation and cannot be rotated.
- Touch devices – Some mobile devices have a touch screen, some in addition to a full keyboard, and some without a keyboard. Touch devices support screen rotation and provide an on-screen virtual keyboard when a full keyboard is not available.
- Windowing model – Mobile devices support a simple stacked window model. Each application can open multiple windows, but each new window is stacked on top of the previous windows and is modal. There is no mouse pointer and therefore the end user has no control over the window, meaning that it cannot be resized or moved. When an application is run, its main window (and subsequent stacked windows) occupies the entire device screen.
- Form navigation using touch keyboard – Touch devices use an on-screen virtual keyboard. Some devices rely on tapping on the controls to navigate between the fields while others have Tab functionality in the virtual keyboard. The navigation inside an Edit control is made using a long press on the field content.
- Context menu – The context menu is an important and central user interaction tool. Since the screen size is relatively small, it is common to perform most tasks using the context menu, instead of wasting screen real-estate on buttons and on-screen menus.
- Input modes – The Edit control is always in Insert mode. There is no equivalent Overwrite mode on the mobile devices.
- Running in the background – The mobile devices' OS is a multi-tasking operating system meaning that each application can run either in the foreground or in the background. The end user can see the running applications and switch between them. An application running in the background is not suspended and continues to run, but does not have access to the screen.

Magic xpa on a Mobile Device

When you decide to deploy on a mobile device, there are a few methods of developing your application so that it may be deployed on the device:

- Web application – A Browser application using the mobile internet browser. There are advantages to this method in that if you already have a browser application, it can already be deployed. Your application will not look like a native mobile application. You will also have limited, if any, access to the mobile's functionality, such as the camera.
- Hybrid – The interface is a mixture of a native application, JS and HTML. As an example, you may be using a Browser control to display a PDF. In this case, you have local access to the device.
- Native – This is a native mobile interface with full access to the device.

Magic xpa enables you to develop whichever method you select. This seminar will deal with the native application.



Developing in RIA

Developing a RIA application for a mobile device, you use the same skills that you use for a regular RIA application. The difference being that you have to take into account a smaller sized screen and different operating systems.

Each device has its own native look and feel. The preferred way to develop an application for a mobile device is to design your application with a native look and feel. You can define your program so that it has a context menu, but that is of little use if your device is an iPhone in which there is no context menu. Your challenge is not the technology, but the display. You do not need to worry about Java for Android or Objective C for iOS. You will see this later on in this course.

Form Considerations

Window Types

All mobile forms are modal by definition. There are no non-modal windows and focus cannot move between open windows. When defining Window Types in the Form Designer, all form types are ignored and the Modal form type is used.

- Full-screen forms – By default, the form will automatically be full screen, taking up the entire screen area of the device. All position and size properties of the form are ignored. This means that when you design for a device such as Android or iOS, which have both a mobile device and a tablet, the screens will be expanded or decreased accordingly and will have a different look.
- Pop-up forms – The form will be opened as a popup window (not full screen) if the **Pop Up** form property is set to **True**. The location of the pop-up window is defined in the **Startup Position** form property as follows:
    - The **Customized** value will open the popup window in the location defined by the **X** and **Y** form properties.
    - The **Centered to...** and **OS Default** ... values will open the popup window as centered on the device.

Each application must start with a full-screen form that remains open for the duration of the session. When the initial program is closed, the application terminates. You can open additional forms, but if the initial program closes, all programs will terminate.

Form Size

The RIA client supports both form scrolling and placement. However, when developing an application that needs to run on devices with significantly different screen sizes, such as the iPhone and iPad, it is best practice to develop two separate forms, since the number of controls will be different on each device.

Placement should be used for each form so that it can be displayed on different devices.

The **Width** and **Height** Dialog unit values of a full screen form with a title bar are:

- For iPhone: 18.75 x 37.5
- For iPad: 53.5 x 111.5
- For Android devices, each client is different.

You may decide that you want different forms for different devices or a specific form for a mobile phone and a different form for a tablet. The **ClientOSEnvGet** function enables you to query for specific device capabilities or features and use it for conditional execution of logic. You will learn more about this in a later lesson. Here are some properties that you can use:

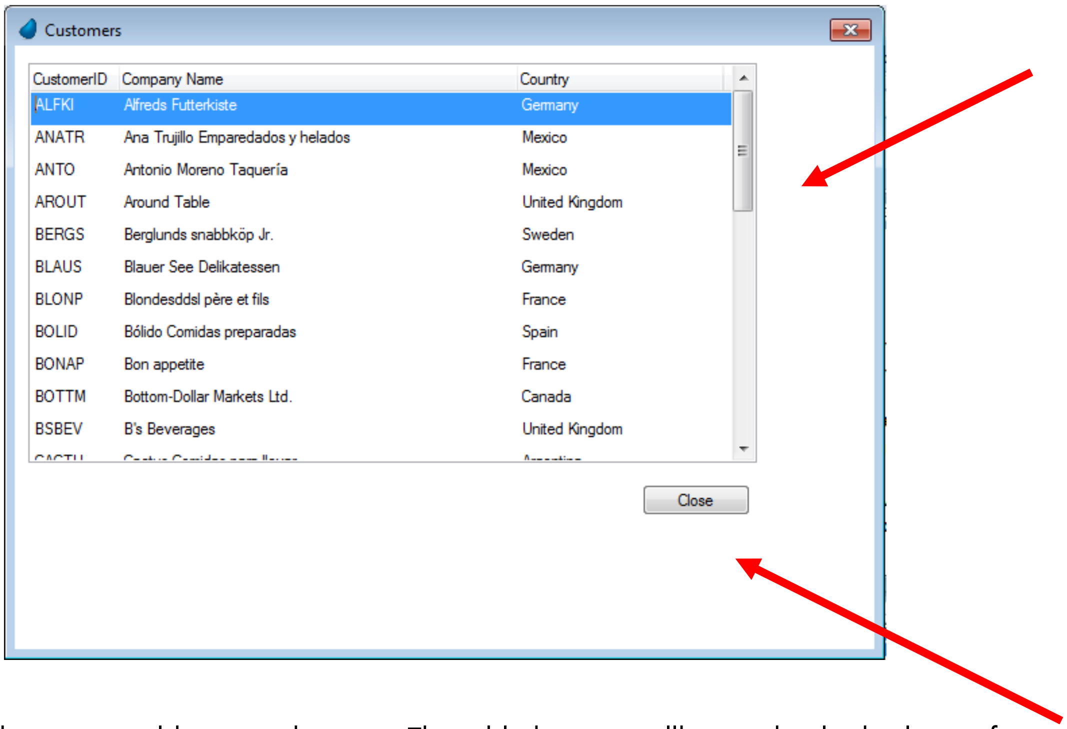
- **ClientOSEnvGet ('device_os')** – returns the device's operating system.
- **ClientOSEnvGet ('device_orientation')** – returns the device's screen orientation.
- **ClientOSEnvGet ('device_screen-width')** – returns the device's screen width in portrait mode in pixels.
- **ClientOSEnvGet ('device_screen-height')** – returns the device's screen height in portrait mode in pixels.

Please note that the values are returned in pixels.

Placement

Often you design your form and the controls on the form according to a fixed size. You place all the controls on the form relative to one another in the Form Designer and then you run the program. In a Windows environment that is often satisfactory.

However what happens when the user increases the size of the form? This is a built-in feature of the Windows operating system. This can happen when the user drags one of the sides of the form or even the bottom right corner. The form simply gets increased. The image below shows an example of what the end user will see when they use the bottom-right corner to increase the size of the form.



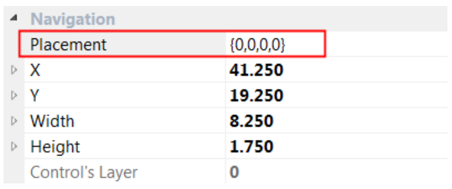
CustomerID	Company Name	Country
ALFKI	Alfreds Futterkiste	Germany
ANATR	Ana Trujillo Emparedados y helados	Mexico
ANTO	Antonio Moreno Taquería	Mexico
AROUT	Around Table	United Kingdom
BERGS	Berglunds snabbköp Jr.	Sweden
BLAUS	Blauer See Delikatessen	Germany
BLONP	Blondesddsl père et fils	France
BOLID	Bólido Comidas preparadas	Spain
BONAP	Bon appetite	France
BOTTM	Bottom-Dollar Markets Ltd.	Canada
BSBEV	B's Beverages	United Kingdom

This is not what you would want to happen. The table has a scrollbar and only displays a few rows. You would expect that the table increase in size and the push button move accordingly. This feature is handled by a property named **Placement**.

Increasing and decreasing a form is a typical windows movement, but how does that relate to the mobile world? When you move from Portrait mode to Landscape mode you are in fact invoking a change in size of the dimensions. You would want the table size to increase or decrease accordingly. You would want control sizes to be changed. What about when you move from a hand-held to a tablet or an Android device in which every device has different dimensions. Placement is very important in these cases.

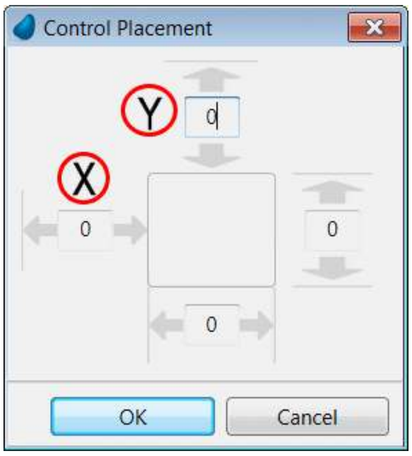
The examples here will be shown using regular Rich Client programs.

The **Placement** property determines whether or not controls are resized when a parent form or parent container control is resized. When a control's Placement property equals zero, the relative size of the control does not change when the size of the parent container is changed in runtime.
When the **Placement** property is larger than zero, the relative size changes proportionally when the size of the parent container changes in runtime.



Navigation	
Placement	{0,0,0,0}
X	41.250
Y	19.250
Width	8.250
Height	1.750
Control's Layer	0

The placement of a control is governed by four values.

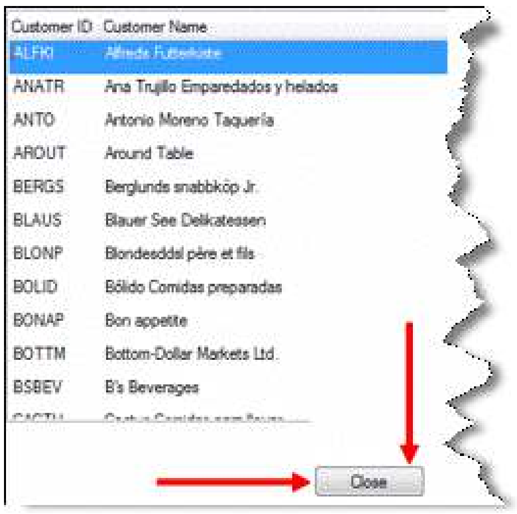


The X and Y values determine how the control moves itself when the form is resized. The value is in percentages. When you set the value as zero, you are defining that the control stays in place. When you set a value of 100, you are saying that the control moves with the form for its full placement.

You can see this with a simple exercise using a regular Rich Client program in which you will define that the form's push button remains relative to the form when it is resized. To do this:

1. Define a data source named **Customers** with the following columns:
   - Customer ID
   - Customer Name
   - Address

2. Add a Rich Client program named **Customers**.
3. Zoom into the **Customers** program.
4. Define the Main Source as the **Customers** data source and add the **Customer ID** and **Customer Name** columns from step 1.
5. Zoom into the Form Designer and add a Table control and drop **Customer ID** and **Customer Name** onto the table.
6. Add a Button control. Open the control property sheet.
7. From the **Event Type** property, select **Internal.**
8. Set the **Event** property to the internal event, **Exit**.
9. Click on the **Placement** property and then click the Zoom button.
10. Set the value of **100** in the **X** property and the value of **100** in the **Y** property.
11. Click **OK**. The **Placement** property will show: **100,0,100,0**.
12. Execute the program.



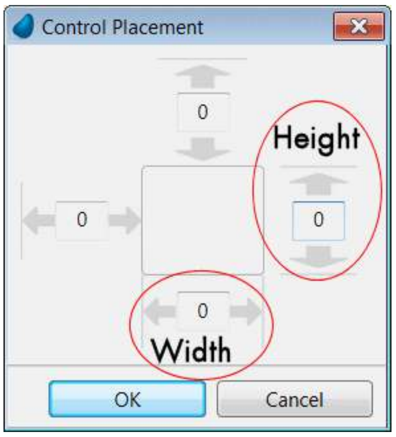
Customer ID	Customer Name
ALFKI	Alfreds Futterkiste
ANATR	Ana Trujillo Emparedados y helados
ANTO	Antonio Moreno Taquería
AROUT	Around Table
BERGS	Berglunds snabbköp Jr.
BLAUS	Blauer See Delikatessen
BLONP	Blondesddsl père et fils
BOLID	Bólido Comidas preparadas
BONAP	Bon appetite
BOTTM	Bottom-Dollar Markets Ltd.
BSBEV	B's Beverages

When you increase the form, it will look similar to the image on the right.

As you can see, the push button remains fixed in position, in the bottom right corner.

Notice that the table is still fixed.

In the **Placement** property, the **Width** and **Height** values determine how the control resizes itself when the form is resized. The value is in percentages. When you set the value as zero, you are defining that the control will not be resized. When you set a value of 100, you are saying that the control resizes itself with the form.



As an example, you will increase the size of the table when the form is increased. To do this:

1. Zoom into the **Customers** program and zoom into the Form Designer.
2. Park on the table.
3. Open the control property sheet.
4. Click on the **Placement** property and then click the Zoom ... button.
5. Set the value of **100** in the **Width** property and the value of **100** in the **Height** property.
6. Click **OK**. The **Placement** property will show: **0,100,0,100**.
7. Execute the program.

If you increase or decrease the form you will see the changes. When you increased the form, the table also increased. You saw more records in the table. This is in essence what will happen when you move from Landscape mode to Portrait mode on your mobile device.

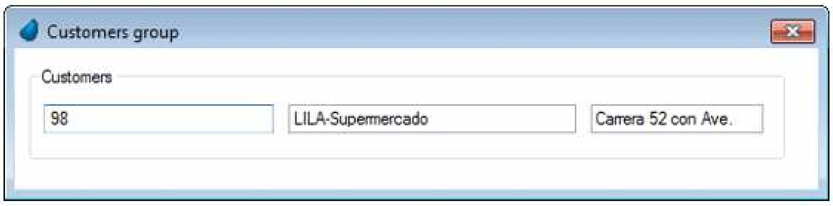
What you saw here was:

- Increasing one control, the Table control, when the form itself increases.
- Have a control, the push button, remain in the same place relative to the bottom of the form when the form is increased.

The controls will never get smaller than their original size. So if you are expecting a form to be resized, create each control at the smallest required size.

The issue of placement gets more complicated when you have more than one control on the form and you want the controls to increase in width, when the form is increased. This is important when you have a large control, such as an Edit control, that you cannot initially display all the information, but when you increase the form you find that you have enough space to display more information.

The image below shows a very simple example:



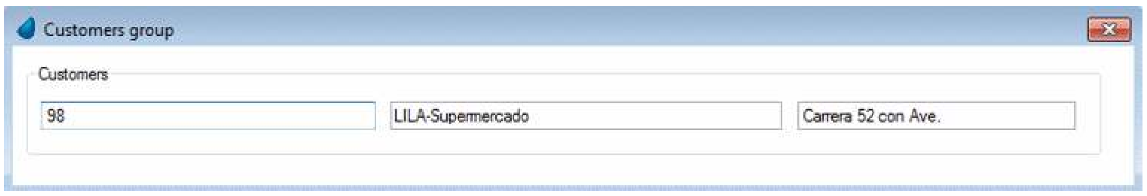
When you increase the group container, you would expect that each control would increase in width. The controls would move and resize in the following way:

- The Group container will increase in width and height.
- The ID control will increase in size.
- The Name control will move to the right to make way for the increase in size of the ID control and will also increase in size.
- The Address control increases in a similar way to the Name control, but you need to take into account that the Name control both moved to the right and increased in size.

To do this:

1. Add a Rich Client program named **Customers Group**.
2. Zoom into the **Customers Group** program.
3. Define the Main Source as the **Customers** data source and add:
    - Customer ID
    - Customer Name
    - Address
4. Zoom into the Form Designer and add a Group control onto the form.
5. Park on the Group control and zoom into the **Placement** property. Set the **Width** property to **100** and the **Height** property to **100**.
6. You can name the Group control as **Customers**.
7. Drop the **Customer ID** Edit control onto the Group control. Set the **Width** property to **32**. This control remains in place, but it must increase in size. Remember that we have three controls that need to increase in size, so each control will increase in size by a third. Zoom into the **Placement** property. Set the **Width** property to **33**. The setting will be: **0,33,0,0**.
8. Drop the **Name** Edit control onto the Group control. Set the **Width** property to **40**. Now comes the fun part. The **ID** control increased in size by 33%, so the **Left** or **X** offset of this control must move by 33%. It must also increase in size by 33%. Zoom into the **Placement** property. Set the **X** property to **33**. Set the **Width** property to **33**. The setting will be: **33,33,0,0**.
9. Drop the **Address** Edit control onto the Group control. Set the **Width** property to **24**. The **Name** control moved 33% and increased in size by 33%, so the **Left** or **X** offset of the **Address** control must take both into consideration. Therefore it must move by 66%. It must also increase in size by 33%. Zoom from the **Placement** property. Set the **X** property to **33**. Set the **Width** property to **33**. The setting will be: **66,33,0,0**.

When you increase the size of the form, the form will now look similar to the image below.



As a summary, when resizing the form:

- Defining values of X=100 and Y=100 will keep the control in the bottom right position. This is mostly used for buttons that should remain in that position, regardless of the size of the form.
- Defining values of Width=100 will resize the control horizontally. This is mostly used when the control size is smaller than its content.
- Defining values of X=100 will move the control horizontally. This is mostly used when this control is displayed after a control that has a Width placement of 100%.
- When you resize a control, you need to take into account the starting offset of the adjacent controls.

In the mobile world you have many different sizes and so you cannot develop, one size fits all. Therefore placement is very important. Bear in mind when designing your form, that when you design for a tablet and want the same layout on the mobile, such as developing for an iPad and viewing on an iPhone, the form may be too small and user-unfriendly.

On mobile devices, the Magic xpa placement mechanism is implemented in a specific way to accommodate different device resolutions and to handle device rotation. The placement mechanism is not relevant for pop-up forms. It works as follows:

- **Resize on open** – Before opening the form, the form size as defined by the developer is compared with the current screen size. The difference between the sizes on each axis is considered as a window resize, and will activate the placement mechanism on all controls. This enables forms designed for smaller screen resolutions to "expand" in higher resolutions.
- **Resize on rotate** – On touch devices, it is possible to rotate the device, effectively changing the resolution on the fly, such as from 480x360 to 360x480. In such a case, the currently displayed form will consider the new resolution as a window resize event, and will activate the placement mechanism on all controls.

Supported Controls

Controls on mobile devices are different than the Windows controls in their appearance, minimum size and space padding. They are also different on each device. The size of the control will change according to the device's DPI (dots per inch). As a result, the size of the controls defined using the Magic xpa **Fit to Size** option may not be large enough to display the entire control text due to the larger padding on the mobile device controls.

The Rich Text, Rich Edit, Tree, List Box and .NET controls are not supported on mobile devices.

Edit Controls

You can control the layout of the keyboard by using the following properties in the Edit control:

- Keyboard Type – Defines the keyboard content.
- Keyboard Return Key – Defines the return key value and action.
- Allow Suggestions – Defines whether the suggestion text will be used.

Image Controls

Images with the **Copied** style do not change and use the same amount of pixels as in the image, so the images will appear in different sizes when used on devices with different DPIs.

It is best practice to use **Distorted Scaling** or **Scaled to Fit** styles to attain the desired appearance.

Two-State Images

You can create two-state images. This means that by clicking on the image on the mobile device, the image will switch to an alternate image. This is done by setting the **Image List File Name** property in the **Check Box** control's **Mobile** section.

You can also create multiple two-state images. This is done by setting the **Image List File Name** property in the **Radio Button** control's **Mobile** section.

Button Controls

The push button is not parkable. The current control will remain the parked control.

Tab Controls

The Tab control appears as a tab bar on Android and iOS devices. The tab bar will appear in full screen; therefore, when developing an app, it is recommended to stretch the Tab control across the entire form. In general, the tab bar's functionality is based on the device's default behavior. You can find more information about this in the *Magic xpa Help*.

Application Navigation

Navigation

Mobile devices can support a simple stacked window model. Each application can open multiple windows, but each new window is stacked on top of previous windows and is inherently modal. Closing the current window and returning to the previous window can be done either by raising the internal Exit event or using the built-in capabilities of the mobile devices:

- Android – using the 'Back' button
- iOS – using the 'Back' button on the title bar. If you decide not to display a system menu, then you need to raise the **Exit** event yourself.

Termination

The application can be closed either by raising the internal **Exit System** event or using the built-in capabilities of the mobile devices:

- Android – using the 'Back' button from the first screen of the application
- iOS – using the close button ('X') on the window title bar from the first screen of the application (if the window was defined to open a System menu)

When pressing the mobile device's home button, the application will remain running in the background.

Note:	When you terminate the application in other ways (such as from the Task Manager), the context on the server will not be released automatically. It will be released only after the context timeout period.

Design Considerations

Magic enables you to develop for mobile devices in the same manner that you developed for Windows with the same look and feel and functionality. However, when developing for mobile devices you want your application to look like other mobile applications, meaning native applications.

As an example of designing your screen for a mobile device, consider the virtual keyboard that was opened when you started editing and ask yourself:

- How do you add a new record? The Magic xpa shortcut is F4.
- How do you delete a record? The Magic xpa shortcut is F3.
- How do you move to the next record, or the previous record (Page Up / Page Down)?
- What about query / sort / range, etc.?

When you look at your mobile contacts or mobile emails or other programs, you normally have a list and when you want to add something you have a button for creating a record. The same for editing; you generally have a button that opens a different form for editing. You will practice this in the example in this lesson and the next lesson when you learn about tables, combo boxes and other controls.

The end-user functionality component is usually not needed for mobile device applications. Therefore, it is recommended to remove the default entry, **UserFunctionality**, from the CRR when developing a mobile application.

Status Bar

You may have noticed that there is no status bar. Take this into account when displaying messages.

Colors and Fonts

Whenever you design your form, you always take into account colors and fonts.

Colors

Mobile devices use the same Magic xpa color table as other RIA clients. Each color must specify the exact RGB value for both background and foreground. You do this in the Magic xpa color table. However, take into account that the mobile devices do not support the Windows "System" colors.

Mobile devices also support transparent and opaque colors.

If a system color is selected, the mobile device will display its own default color for the form or control. If you want to use a specific color, you need to implicitly define the foreground and background colors.

For some controls, to see the border, you must use a color with a non-system color.

For Combo Box controls, to see the arrow you must use a color with a system color.

Fonts

The Magic xpa mobile RIA client uses the same font table as other interfaces. Each device has a set of available fonts that are usually different from the fonts found on a Windows desktop and different from one another. If the font defined in the font table is not found on the mobile device, the default font will be used with the size defined in the font table.

When you develop, you need to take into consideration whether you want to use specific fonts for each device or enable the device to use the defaults.

If you use different fonts for different devices, you will need to use an expression for the font of a control. You can use the ClientOSEnvGet function in the expression:
IF (ClientOSEnvGet ('device_os')='android', '1','4')

You may decide that working with the CASE function will better suit your application.

On Android devices you need to send the font family name and the style of the font you want. The system will find a matching font for you. For example, to use a Droid Serif font use: "Android Serif bold,Serif,12,0,0". You can see the font list in the **System/Fonts** folder in the device file system.

You cannot select mobile fonts using the Windows fonts dialog box. If you decide to use separate font entries for each device, you need to edit the fonts table manually using an external text editor such as Notepad.

For example, Android uses a font known as **Sans**:

1. Open the **fnt_rnt.eng** file in a text editor. By default, this file is in the **support** folder of the installation.
2. Add the following line: Android Edit Labels,sans,8,0,0,Bold
3. Save the file.
4. Close the **demo** application and reopen it so that the new font list will be reloaded.
5. Zoom into the **Hello World** program.
6. Zoom into the Form Designer and for the Label control, select the new font. Because the font is now bold, you need to increase the width.
7. Execute the project.

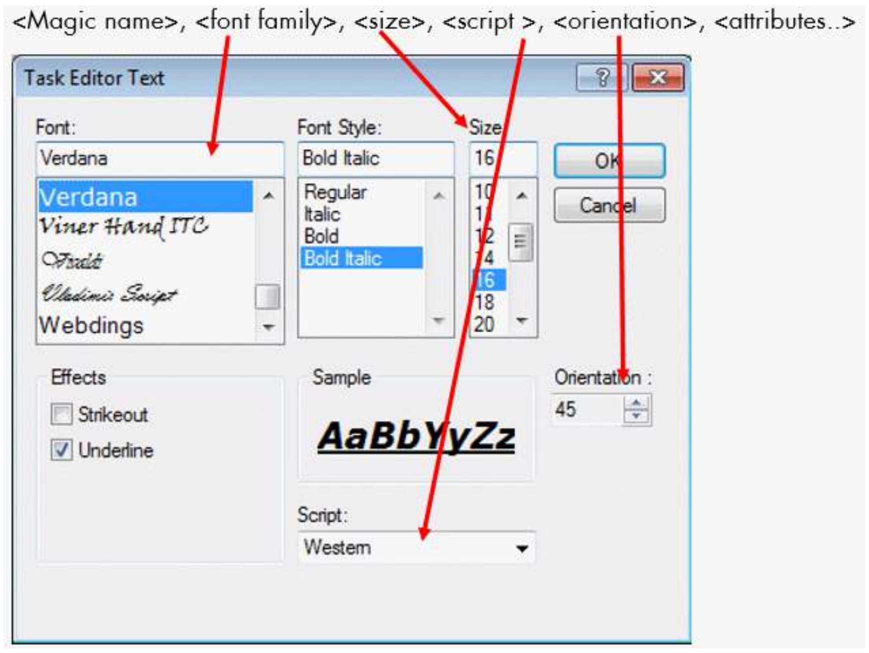
You will see that the label is now bold.

Adding a font manually

When you added the font manually, you added the line:
Android Edit Labels,sans,8,0,0,Bold to the Font list.

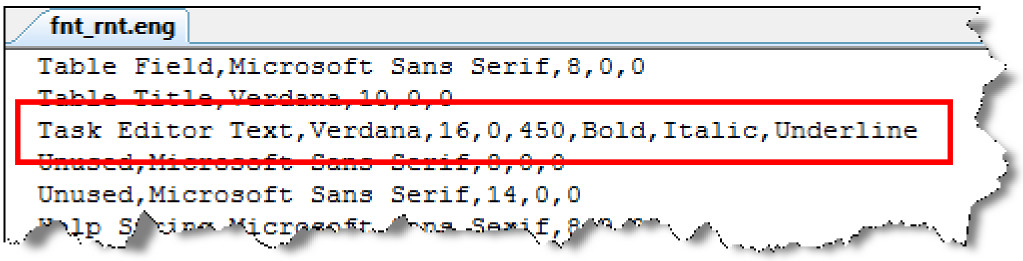
Each entry in the list is defined in the following way:

<Magic name>, <font family>, <size>, <script >, <orientation>, <attributes..>



Font: Verdana
Font Style: Bold Italic
Size: 16
Effects: Strikeout [ ] Underline [✓]
Orientation : 45
Script: Western

The attributes are the **Font Style** and the **Effects**. The dialog box above will be translated to the following entry in the Font list:


fnt_rnt.eng

```
Table Field,Microsoft Sans Serif,8,0,0
Table Title,Verdana,10,0,0
Task Editor Text,Verdana,16,0,450,Bold,Italic,Underline
Unused,Microsoft Sans Serif,8,0,0
Unused,Microsoft Sans Serif,14,0,0
Help String,Microsoft Sans Serif,8
```

Summary

You were introduced to the new world of developing mobile applications.

You learned that you can continue developing your application in the same way that you developed for desktop applications, but that it is good practice to take into account the limitations of the mobile device, which are smaller devices and are also not Windows devices. You learned to take into account the sizes, the colors and the fonts on the mobile device and you learned how to get information from the device, such as its operating system. You can use this knowledge to define that certain programs will not run on certain devices.

This lesson can be summarized by saying that when developing for a mobile device, think native!

Lesson 3

Advanced Controls

In the previous lesson you learned about the mobile client and some of the limitations when working with different clients. You also defined a simple program. One of the more important messages in the previous lesson was that you need to consider the native environment when you develop for the mobile. An Edit control with a long line of text will not be visible when it is displayed on a small mobile screen in portrait mode.

As you know when working with Magic xpa, when you design a program, you are not only using simple controls such as a Button, an Edit, or a Text control, but you also display data in a table, select data from a list and others. You will learn about this in this lesson.

This lesson covers various topics including:

- Tables
- Events
- Selection Lists
- Menus

Tables

As you learned, developing for a mobile device has limitations but it also provides you with some advantages. When you use a Table control in Magic xpa, you normally add as many columns of data as you need. If there is not enough room to display the table, you have a horizontal scrollbar on the form.

On mobile devices, horizontal scrolling is allowed on the form only. Horizontal scrolling on the content of the table is not possible.

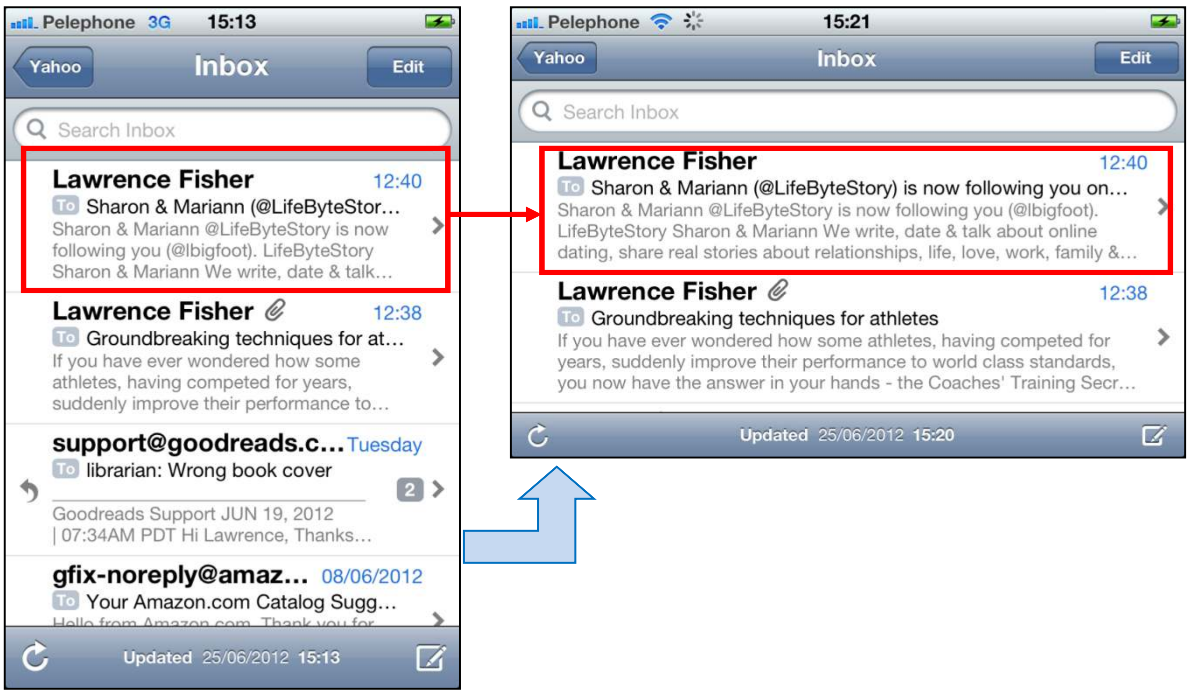
How many of you have developed a one-to-many form such as an order form in which the order lines are displayed in a table? To create a new order-line, you instructed the user to click F4 and you were able to enter a new line. The Create Line event is supported by Magic xpa, but there is no F4 key on a mobile phone. This is also not the accepted method for the user display when working with a mobile device. Remember also that when working with an Android in the one-to-many scenario, you will not have a vertical scrollbar.

Apple wrote a document known as Human Interface Guidelines (HIG) in which they defined a table as a view that presents data in a single-column list of multiple rows. Look at the Contacts or Emails. These are guidelines and so you define the interface as you want. As an example, eBay displays three-columns: image, description and current bid.

The bottom line is that Magic xpa enables you to develop the display that you want, but it is advisable to work with the guidelines of the device or at least develop a display that fits all guidelines.

Landscape Mode

When you rotate your device to Landscape mode, you are able to view more information on each line. When you use placement on the table you are able to view more of the text. For example, have a look at the email list below:



Pelephone 3G 15:13
Yahoo Inbox Edit
Search Inbox

Lawrence Fisher 12:40
To Sharon & Mariann (@LifeByteStor…
Sharon & Mariann @LifeByteStory is now following you (@lbigfoot). LifeByteStory Sharon & Mariann We write, date & talk…

Lawrence Fisher 12:38
To Groundbreaking techniques for at…
If you have ever wondered how some athletes, having competed for years, suddenly improve their performance to…

support@goodreads.c… Tuesday
To librarian: Wrong book cover 2
Goodreads Support JUN 19, 2012 | 07:34AM PDT Hi Lawrence, Thanks…

gfix-noreply@amaz… 08/06/2012
To Your Amazon.com Catalog Sugg…
Hello from Amazon.com. Thank you for…

Updated 25/06/2012 15:13

Pelephone 15:21
Yahoo Inbox Edit
Search Inbox

Lawrence Fisher 12:40
To Sharon & Mariann (@LifeByteStory) is now following you on…
Sharon & Mariann @LifeByteStory is now following you (@lbigfoot). LifeByteStory Sharon & Mariann We write, date & talk about online dating, share real stories about relationships, life, love, work, family &…

Lawrence Fisher 12:38
To Groundbreaking techniques for athletes
If you have ever wondered how some athletes, having competed for years, suddenly improve their performance to world class standards, you now have the answer in your hands - the Coaches' Training Secr…

Updated 25/06/2012 15:20

You also have the option of displaying more information on the table. Remember that you can use the **ClientOSEnvGet ('device_orientation')** to know whether you are currently in Portrait or Landscape mode.

Mobile Events

When working with a mobile device, you use your finger as the mouse; your touch is my command. The touch events are mapped to Magic xpa events as follows:

- The mobile “Touch”, “Click” or “Tap” events are equivalent to the Magic xpa mouse-click event. These events move the focus between fields on a form. Touching on selection controls changes the controls’ selection. Touching on buttons triggers the buttons’ event.
- The “Press” or “Hover” events trigger a Magic xpa **OK** internal event. This event can then be handled by the developer.
- “Swipe” events scroll within the form or control.

Selection Lists

A selection list enables you to select a value from a list. This can be a regular control such as the combo box, check box and the radio button, or a selection program that is defined in Magic xpa as a Selection program. Remember that the radio button is not supported on all platforms.

Selection Program

In Magic xpa Online programs and RIA programs, pressing Enter or the double-clicking action raises the Select event. On your mobile device, you do not have an Enter button and you are unable to double-click. You need to find other methods to raise the Select event. Two methods you can implement are:

- Use a push button that raises the Select event.
- Trap the OK event. This is raised by the mobile "Press" or "Hover" events. Within the handler you can raise the Select event.

Remember that the selection program is a modal full screen program, unless you define it as a Floating program.

Selection Controls

Selection controls, such as the Combo Box, appear differently on each device. This is not something you can control as this is defined by the operating system of the device itself. It does, however, provide a native look and feel.

Give some thought as to how you are going to implement the modification of a field.

Menus

When developing any system, defining the Menu system is an integral part of the screen design as the menus allow quick access to functionality. A menu system takes up valuable screen real estate and is impractical in small devices. The Windows pulldown menu is unsupported in mobile operating systems since there is no MDI On mobile devices. Therefore, it is recommended to remove the **Default pulldown menu** entry when developing a mobile application.

However, the context menu is an important and central user interaction tool. As the screen size is relatively small, it is common to perform most tasks using the context menu, instead of "wasting" screen space on buttons and on-screen menus.

In Magic xpa, in Android devices, context menus are available at both the form and the control level. On iOS devices, context menus are only available at the form level.

Summary

In this lesson you learned more about how to interact with the mobile device.

You learned about the table and some guidelines provided by mobile manufacturers.

You learned about some events that are unique to mobile devices and how they are defined in Magic xpa.

You learned about selection lists and you were given some guidelines about menus.

In the next lesson you will learn more about interacting with the mobile device.

Lesson 4

Interacting with the Device

The advantage of working with a mobile device is that you are able to use the device's special features such as the GPS and camera.

This lesson covers various topics including:

- Interacting with the file system
- Fetching information from the device
- Using the GPS and camera

Mobile File System

You can access the mobile device's file system using Magic xpa functions. The naming conventions for local folders and file names are according to the device's OS:

- Names are not case-sensitive.
- Folder separators are defined using a slash mark (/).

Access to the different device folders depends on the device. For example:

- On iOS, the access is allowed only to a temp folder provided by the OS. When using the ClientFileXXX functions with a relative client filename, the file folder will be the temp folder. When using these functions with a full path, it is recommended to use **ClientOSEnvGet('temp')** to retrieve the temp folder.
- On Android, you can access different folders.

Querying Device Characteristics

The ClientOSEnvGet function enables you to query for specific device capabilities or features and use it for conditional execution of logic. You already saw examples of this in previous lessons.

The syntax is: **ClientOSEnvGet ('keyword')**
Here are the keywords that may be used within the ClientOSEnvGet function:

Keyword	Description
device_os	Returns the device's operating system.
device_screen-width	Returns the device's screen width in Portrait mode in pixels.
device_screen-height	Returns the device's screen height in Portrait mode in pixels.
device_physical-width	Returns the device's physical screen width in portrait mode in inches.
device_physical-height	Returns the device's physical screen height in portrait mode in inches.
device_orientation	Returns the device's screen orientation.
device_os-version	Returns the device's OS version number.
device_model	Returns the device's model number or name.
device_touch	Returns "1" if the device has a touch screen.
temp	Returns the temp folder on the device.
device_location	Returns the current device's location.

Keyword	Description
device_magic_version	Returns the RIA client version number.
device_udf|getargs	Returns the query parameters when the application was launched from another application.
device_udf|getpushid	Returns the device ID, which can be used for sending push notifications to the device.
device_udf|my_string	Calls a user defined function.
device_resource-folder	Returns the value of the resource folder.

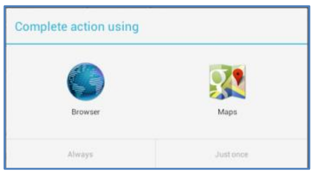
On Android devices, you can also use Java predefined keys to get additional information, for example: ClientOSEnvGet ('java.io.tmpdir'). For more information, refer to: http://developer.android.com/reference/java/lang/System.html#getProperty(java.lang.String)

Finding the Device Location (GPS)

The ClientOSEnvGet function can also be used to query the current device location using the internal or connected GPS device.

ClientOSEnvGet ('device_location') – returns the current device location, using any of the available location options (GPS, Network, etc.). The result is a string in the following format: **OK|Latitude|Longitude**, where **OK** is a fixed part for testing if a result was returned, and **Latitude** and **Longitude** are the coordinates of the current location. If a location could not be obtained, for any reason, an error message will be returned.

There are other methods of displaying the location in the device by using the **Invoke OS Cmd** operation with **Execute On=Client**. You would therefore not need the subtask:



- You can directly enter: **https://maps.google.com/?q='&Trim (GPS Location)**. If the Google Maps application is installed on the device, you will be asked to decide in which application to display the location.
- You can use '**geo:'&Trim (GPS Location).** This will display each application that can display a geographical address.
- You can load a specific application directly by using its name, such as: **'waze://?q=London'** to open the Waze GPS application and navigate to London.

Location queries can sometimes take time to respond, because the GPS device is searching for satellites. During this time, the client is blocked, waiting for a response. You should make proper indications for the user that this is the situation. As an example, before invoking the GPS service, you can display a form such as "Searching for GPS location…"

Location queries have a built-in timeout of 20 seconds.

This operation opens the connection to the GPS and closes it once the data is received. If you need to frequently get the GPS data, you should consider keeping the GPS connection open by writing native code to do so.

Camera Support

It is possible to initiate a camera from the device using the **ClientFileOpenDlg** function.

- When this function is used with the following word: **camera** as the second parameter, the camera will be opened.

This will enable the user to take a picture and select it. The full path name of the picture will be returned from the function. It is then possible to upload this picture to the server using the **ClientFileToServer** function.

Note that the **camera** value cannot be used to capture videos.

You need to provide all parameters for the ClientFileOpenDlg function. Here is an example for use on Android devices:

ClientFileOpenDlg('','camera','','FALSE'LOG,'FALSE'LOG)

Gallery Support

You can also get an image file from an image gallery using the **ClientFileOpenDlg** function.

- When this function is used with the following word: **images** as the second parameter, the images gallery will be opened.

The full path name of the image will be returned from the function. It is then possible to upload this image to the server using the **ClientFileToServer** function.

Accessing the Mobile Devices' Capabilities

It is possible to use the device capabilities such as calling a phone number, sending an SMS, and opening a browser, by using the Invoke OS command with the URL of the required command. For example:

- tel:1-408-555-5555
- sms:1-408-555-1212
- mailto:support@magicsoftware.com
- mailto:to=support@magicsoftware.com&subject=this is a test&body= test mail
- http://magicsoftware.com

Note:

- Do not forget to set the **Execute On** property to **Client**.
- When sending mail, there are some characters that are not allowed in the subject and body. On iOS devices, for example, the ampersand (&) character should be replaced with %26 and on Android devices the following characters should be removed: #,%,&, :, and =.

Refer also to:

http://developer.apple.com/library/safari/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007891-SW1

Summary

In this lesson you learned how to interact with the device using Magic xpa functions. The **ClientOSEnvGet** function has predefined keywords that enable you to use some of the device's options, such as the GPS and camera.

The built-in keywords enable you to fetch details about the device and display different information. For example, you may decide that a certain program is only available if you are using an iPhone.

Lesson 5

Offline Implementation

Magic xpa mobile applications work in a client server mode in which the client, the mobile device, is the interface that the user sees and contains part of the code. However, the server is at a remote location, has the database and handles any heavy workload. There is constant communication between the client and the server depending on the type of work involved. As you already know, when you call a new task, this process involves accessing the server to fetch the task details and then to display the task on the client.

What happens when you do not have internet communication? That is where offline programming is necessary. Magic xpa enables you to develop Offline tasks. Offline programs allow users to continue to be productive in areas with intermittent, limited or unavailable internet connectivity. While working offline, data is stored locally on a local database, and periodically, when internet connectivity resumes, you can synchronize it back to the server.

This lesson covers various topics including:

- The Offline concept
- How the Offline concept works

Concept

When planning an offline implementation, it is important to understand the challenges and constraints that you need to overcome to enable applications to work completely offline without a server connection. Unlike a connected application, server connectivity is either non-existent or intermittent, and applications need to be adjusted to handle this scenario properly, without compromising usability and data integrity. As a programmer, you need to take into account the technical aspect of being disconnected, and the data consistency aspect, as follows:

- You can store a subset of relevant server data or client-only data on the client. You need to understand what data is constant data, such as a list of countries and which is not updated often, such as suppliers and items that are updated often, such as stock. However, even when dealing with stock, you may not need all of the data on the client. For example, a salesperson who deals with orders involving computer hardware only needs the products dealing with computers, while a salesperson who handles cleaning products only needs that subset. You can also have a scenario where the salespeople have access to the entire catalog but only current stock quantities is kept on the server and needs to be synchronized.
- On systems that require user authentication, consider storing user credentials securely on the client.
- Allow data entry on the client and update client data with server data for data consistency.
- Working under intermittent network connectivity (network disconnects, slow connections) while allowing uninterrupted operation and data consistency.
- Keeping application resources such as application metadata, image resources and so on, locally on the client, while allowing updates during connectivity periods.

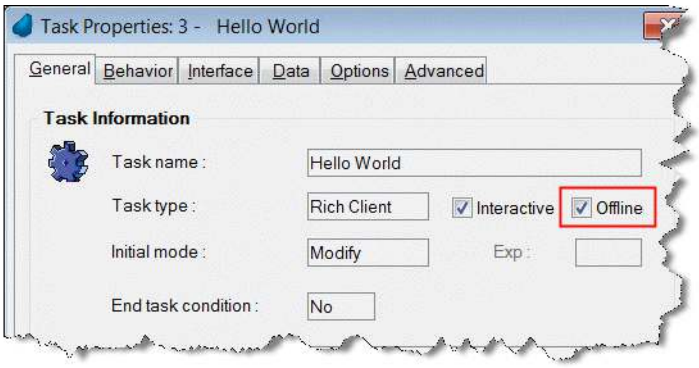
The above challenges define an offline pattern that is different from patterns used when network connectivity is guaranteed, and require the developer to handle additional usage scenarios. Magic xpa provides tools and features that allow developers to tackle these challenges and provide a complete offline experience.

Some programs are totally client programs and some are server-based and others are both. An example of a client program can be something like a university student's lesson timetable. This timetable is only updated every semester and, therefore, the data can be kept on the client. An auction-based client depends entirely on server data and, therefore, you need an internet connection for this. A customer order can have elements of both server and client data.

How Does It Work?

Unlike connected applications, offline applications are designed to work without (or with intermittent) network connectivity. This limitation defines a different execution flow for offline applications.

To define an Offline program, you simply define a Rich Client program and select the **Offline** check box.



Task Information

Task name : Hello World

Task type : Rich Client ✓ Interactive ✓ Offline

Initial mode : Modify Exp :

End task condition : No

Typical offline applications will work as follows:

- On first invocation, offline applications must download and synchronize all necessary resources required for the offline operation. These include application metadata, images, client-side data and so on. This requires an offline application to be connected at least once before it can work offline. If there was no initial connection, the programs will not work.
- For applications that require user authentication, user credentials should be securely stored on the client to allow for operation without server authentication. To ensure validity, such credentials should be re-checked when connected.
- Following initial invocation, all user interactions must be done using local resources only (local data, local images and so on). By using local resources exclusively, the application is guaranteed to work, regardless of the internet connectivity state and without requiring server access. All data updates should be stored locally on the local database.
- Periodically, at an application-dependent timing, the application must synchronize modified local data back to the server and download server data that was modified since the last synchronization. Remember that data objects can be updated by multiple clients and by the server simultaneously and therefore two clients may update the same data at the same time. You need to take this scenario into account.
- Since Magic xpa automatically synchronizes metadata objects while connected, an application that runs offline must be allowed to periodically synchronize changes to its metadata objects. Typically, if an internet connection is available on startup, metadata objects will be synchronized automatically. The developer should plan for allowing metadata updates to happen when the application changes.

For more information about how to implement each of the above, and the supporting Magic xpa features that enable each capability, see the **Developing Offline Rich Client Applications.pdf** that can be found in Magic xpa's **Support** folder or in the *Magic xpa Help*.

Summary

If you decide to give offline capabilities to your application, you need to give thought as to how to implement the programs and of course which programs are offline and which are not.

You need to think about the following issues and take into account:

- Which programs are fully offline and which need to be online.
- An Offline program cannot call a server program and you need to use the Main Program to call a server program.
- Your implementation of the synchronization operations and how to have the same data on the server and the client. Remember that when you have multiple clients you may encounter a situation where the same record is updated by two different clients.

Lesson 6

Customization and Installation

In previous lessons you used the Android emulator and you used the environment that the installation process provided. In this lesson you will learn how to provide your own environment.

This lesson covers various topics including:

- Customizing the client application
- Signing the keystore file
- Creating your own Android package file
- Installing the client on the Android device or emulator

Note: There are similar methods for creating an iPhone file. Check the Magic xpa Help for assistance.

Execution Properties File – Additional Information

In the first lesson, you learned about the execution properties file. In that lesson, you used an external file and defined its location in a dialog box opened from the generic Magic xpa client.

When a mobile device client is prepared, it is compiled with a certain execution properties file. This file tells the client where to find the initial program to load. There are three ways of defining an execution properties file in the compiled client application:

- An empty file – In this scenario, the end user will initially get the dialog box requesting the address of the properties file. This is the same behavior you saw in the first lesson when using the generic Magic xpa client. This is very useful during the development process because it enables you to separate your development and production environments. You do not want your end user to be prompted to enter your server details.
- Entering the server, application and programs – The format will be similar to the **devprops** file that you learned about. The end user will not be prompted for the path to the **devprops** file. This means that all information will be compiled with the client. The disadvantage here is that if you make changes to the application name or the program name, such as the change from start to orders, you will need to recompile the client application and upload it to the mobile device.
- Entering a URL – You enter a URL that is a full URL directing the application to the **devprops** file found on your server. The **devprops** file then directs the client to the application and program. The advantage here is that if you need to change the application name or the program name, there is no need to prepare a new client. In the **devprops** file, you saw the **<property key="server" val="10.1.10.75"/>** property. In the execution properties file, you can use either. To enter a URL you would use the following property: **<property key="url" val="http://example.com/MobileScripts/devprops.txt"/>**

You will learn more about compiling a new client in the next section.

Customizing the Application

Throughout this course, you used the icon provided by the installation, which includes the Magic icon and wallpaper. However, the client may already have an installation of another application or you may want to provide your icon. Creating a custom application requires compilation using tools provided by the mobile devices.

You can change items such as:

- Icon
- Startup splash screen
- Execution properties – You learned about this in the previous section.
- Client title
- Application version
- Package name

Here is an explanation on how to do this for an Android-based mobile application. For iOS-based devices, please see the *Magic xpa Help*. You will need the JDK and the Android SDK. The SDK is a part of the Android emulator. The source code for the Android application is provided in **%EngineDir%\RIAModules\Android\Source**. You can modify the mobile application's settings by updating the **settings.properties** file located in the directory mentioned above. You can change the following properties:

Property	Description
sdk.dir	The directory in which the Android SDK is installed. When you use the \ character, you need to double it: \\
target	The Android version that the compilation is done for. It is advised to use the most updated version. The minimum version that you should select it the version that Magic xpa is compatible with. Note that the version should be installed on your PC. You can see the installed versions using the Android SDK manager or by browsing to the **android-sdk\platforms** folder.
client.title	The application title as visible to the user.
client.version.code	An integer value that represents the version of the application.
client.version.name	A string value that represents the release version of the application code, as it should be shown to users. Your version code and name can be different, since only the name is displayed to the user.
package.name	The identifier of the project. The package name must be unique across all packages installed on the Android system.
output.dir	The folder that will contain the built installers.
build.dir	A temporary folder location used during the build process. This folder is removed after a successful build. It can be an absolute or relative path.
key.store, key.alias, key.store.password, key.alias.password	The properties of the keystore. A sample keystore is provided in the **Android** folder.

- Icon – To change the icon, replace the icons' files in the **res\drawable-XXX** subfolders. You will need to provide icon files in all of the following sizes: 36x36, 48x48 and 72x72 pixels.

- Startup screen logo – To change the startup screens, replace the **logo.png** files located in the **res\drawable-XXX** subfolders. You need to provide logo files in different sizes corresponding to the device's size.
- Execution properties – Open the **execution.properties** file and change the execution values.
    - You need to define either the execution properties' values or a URL referring to a file containing the execution properties' values as defined above. Leaving the URL property empty means that a dialog box will be opened and the end user will need to write the URL.

Now that you have all the information, you can start preparing the APK file (for Android devices).

There are a few stages (which are explain later on) to preparing an APK file:

1. Prepare a keystore file. The keystore file is a file that enables you to "sign" your APK file. To do this, you use your own keys and certificates. A sample file is provided with this course.
2. Create the signed APK file.
3. Upload the new APK file to the Android device or emulator.

Keystore File

A sample keystore file is provided in the installation. Remember that when you deploy your application, you need to provide your own keystore.

To create the keystore, you will use the **keytool.exe** application installed with the Java SDK, found in your **%JAVA_HOME%** folder.

4. Run the following command and follow the instructions:
   Keytool –genkey –keystory KeystoreName –alias AliasName

KeyStoreName and **AliasName** will be replaced with your own details. During the process, you will be asked for the password of the keystore and the alias. You will be asked for identifying details such as your name, company and address. At the end of this process the keystore file will be updated with the new information.

APK File

The APK file is based on both the **execution.properties** file and the **settings.properties** file. Make sure that both are defined as you require. Compilation of the Android project is performed using the **Ant** tool. The **Ant** tool is provided with the installation and is located in the **%EngineDir%\RIAModules\Utils\apache-ant-1.8.3** folder. To continue:

5. Navigate to the application source folder:
   %EngineDir%\RIAModules\Android\Source.
6. Run the following file: **build.cmd**.

The build process creates an APK file named according to the **client.title** setting of the **setting.properties** folder and is located in the **output.dir** folder.

You can edit the **build.cmd** file in order to troubleshoot the build process. For example:

- To see the full actions, remove the **-q** from the command in the **build.cmd** file.
- To save the log to a file, add **-l build.log** to the command in the **build.cmd** file.

7. Copy the APK file to the **scripts** folder.

Installing the Client on the Android Device or Emulator

There are a few methods to install the APK client on the Android client:

- Run the APK directly on the Android, after receiving it as an email attachment or similar.
- Download the APK via the mobile device's browser by navigating to the APK file. For example: **http://server_name/magicscripts/myapp.apk**.
- Download your application from the Android market.

After successfully installing the APK you will see a new icon under the Applications in the device.

Refer to the **Customizing Your Application** and **Installing an Application on a Mobile Device or Simulator** topics in the *Magic xpa Help* for more information on how to customize and install the application on an Android device.

Summary

In this lesson you learned how to prepare your own client, which you can use to call your own Magic xpa application. Remember that the steps to prepare your application are:

1. Prepare the icons and splash images.
2. Define the **settings.properties** file.
3. Define the **execution.properties** file.
4. Get a keystore file. Check this link for more details: http://developer.android.com/tools/publishing/app-signing.html
5. Create the APK file.
6. Install the APK on your device.

Lesson 8

Preparing the Testing Environment

When you develop you need to have a testing environment. This seminar installed the Android Emulator as a part of the seminar but you will need these tools for your own testing environment.

This lesson covers various topics including:

- Installing the Android emulator
- Defining the MIME type
- Troubleshooting
- Debugging

Note: There are similar methods for creating the testing environment for iOS devices. You can find information about this in the *Magic xpa Help*.

Defining a Simulator

To test your environment, you need to set up a simulator.

Android

To run the Android emulator, you need to download and install the Java SE Development Kit (JDK) 6 and the Android SDK. The JDK is installed automatically during the Magic xpa installation if you installed the Web Services component during the Magic xpa installation.

You can download them from:

- JDK: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- SDK: http://developer.android.com/sdk/index.html

An alternative emulator can be downloaded from: https://cloud.genymotion.com/page/launchpad/download/. This emulator is actually a virtual machine running virtual Android devices. After downloading and installing the virtual machine, follow the wizard to download and install Android devices. Note that this emulator is faster than the native Android emulator.

Android SDK Manager

During the installation of the Android SDK, the Android SDK manager will be launched.

You need to select the following components (for each component you need to select the latest Android versions or at least the versions that Magic xpa is compatible with):

- **Tools\Android SDK Tools** – For Magic xpa 3.0, for example, you will need version 24 or above.
- **Tools\Android SDK Platform-tools** – For Magic xpa 3.0, for example, you will need version 21 or above.
- **SDK Platform** – For Magic xpa 3.0, for example, you will need version 21 (the package is available under the **Android 5.0.1 (API 21)** section) or above.

The installation package will then start downloading the package to your system. This process may take a while depending on your download speed.

The Android SDK Manager is installed on your system enabling you to add or remove packages in the future.

Android Virtual Device Manager

After installing the SDK platform:

1. Click the **Tools** menu and start the **Manage AVDs** (Android Virtual Device) entry in order to configure the Android emulator and virtual environment.

2. In the Android Virtual Device Manager screen, add a new device and specify:
    - **Name** – The name for the virtual device.
    - **Target** – Choose the Android version, for example: Android 4.0.
    - **SD Card Size** – Define a size for the SD card, for example: 20. The SD Card is the Android storage card. Since this is an emulator, you simply define the size of the storage file.
3. Click the **Create AVD** button. After the device creation is complete, the device will be added to the list.
4. Select the device and click **Start** to run the emulator.

Note: Additional information about the emulator can be found at: http://developer.android.com/guide/developing/devices/emulator.html.

Installation on the Device

As you learned in the previous lesson, there are a few methods to install the client on the Android device:

- Run the APK directly on the Android device, after receiving it as an attachment as an example.
- Download the APK via the browser as explained.
- Download the APK from the Android market.

To download from the browser or to run the APK directly from the browser, you need to configure the Web server to support the download of the APK file. This means defining a MIME type. According to the Microsoft MSDN, Internet Information Services (IIS) serves only static files with extensions registered in the **Multipurpose Internet Mail Exchange (MIME)** types list. IIS is preconfigured to recognize a default set of global MIME types, and also allows you to configure additional MIME types and change or remove MIME types. These MIME types are recognized by all Web sites you create in IIS.

To add a new MIME type, you:

5. Open the Internet Information Services (IIS) manager.
6. Navigate to the **Default Web Sites**.
7. In IIS 7, open the **Mime Types** and add a new one as follows:
    - Extension=.apk
    - MIME type=application/vnd.android.package-archive

For IIS 6, you define a MIME type as follows:

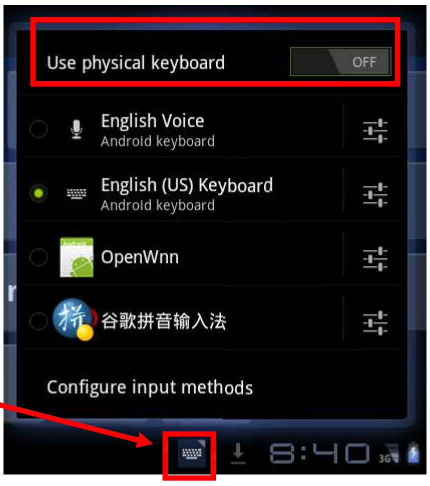
1. Open the Internet Information Services (IIS) manager.
2. Navigate to the **Default Web Sites**.
3. Right-click and click **Properties**.
4. Click the **HTTP Headers** tab.
5. Click the **File Types** button.

6. Click the **New Type** button and add a new one as follows:

   - Extension=.apk
   - MIME type=application/vnd.android.package-archive

Using the Keyboard on the Device Simulator

- When you start typing in an Edit control on the device simulator, you will notice that a virtual keyboard pops up on the emulator. It is important to use this keyboard. You will notice that, as an example there is no **Delete** key, since there is no functionality for the delete hotkey in some devices. The backspace key, which in many cases behaves like the delete key, is displayed as an X ✕. For regular typing you can use your own keyboard. If you use your keyboard Tab key, you may get unexpected results, so it is good practice to use the emulator keyboard.
- If the keyboard does not pop up, then click the keyboard icon at the bottom of the screen and you will get the popup dialog box as shown on the right. Set the **Use physical keyboard** property to **OFF**.
- On regular mobile devices with a virtual keyboard, the virtual keyboard is automatically displayed when parking on controls that require keyboard input, namely modifiable Edit controls. On other controls, the virtual keyboard is automatically hidden.



Use physical keyboard OFF

English Voice
Android keyboard

English (US) Keyboard
Android keyboard

OpenWnn

谷歌拼音输入法

Configure input methods

Troubleshooting

The most common problem regarding the mobile RIA client is that the client fails to connect to the Web server that is serving the application. There are several scenarios where this problem can occur:

Mobile device fails to communicate with the Web server

You will not see any requests in the Broker or GigaSpaces Monitor or any connection indication in the Web server log.

- If the communication is performed using WiFi, verify that the device is connected to the correct WiFi router.
- Try connecting from the device browser to the Web server to see if there is a connection. For example: http://192.168.137.1/MagicScripts

If you do not get a response, then the Web server may not be configured correctly.

Your firewall may be blocking the communication. Try to enable inbound http/https communication in the firewall rules. If the problem remains, try to disable the firewall completely. If you can connect from a browser to the Web server or if your application is now working, then you can enable it again and make sure that the firewall enables inbound connections on port 80/443. To enable port 80:

1. In the Control Panel go to:
   Windows Firewall > Advanced settings > Inbound Rules.
2. Open **World Wide Web Services (HTTP Traffic-In)** and select **Enabled**.

Execution properties file defined in the URL dialog is wrong

You will not see any requests in the Broker Monitor, but you will see the access to the server in the Web server log.

- Check that the **execution.properties** file was saved in ANSI format.
- Check that all the attributes in the **execution.properties** file are spelled correctly and have the correct values. Remember that the values are case sensitive.
- If the **settings.properties** file contains a URL attribute with reference to an external execution file, check that:

  * The external file can be accessed from the mobile device. You can check it by entering the same URL value on a browser on the device.

  * The external file is in ANSI format and all the attributes in this file have the correct value (case sensitive).

- Verify that the Magic xpa RIA server is up and is running the project defined in the settings properties.

To check your file on the Windows desktop, you can rename the file to **execution.properties** (remember that you gave it a txt file extension), place it in the **%EngineDir%\RIAModules\Desktop** folder and run the **MgxpaRIA.exe** application in that folder. If everything is properly defined, the RIA application should start on the Windows client.

Debugging

As in the desktop RIA, you can activate debug logs on your mobile client by defining a proper value for the **InternalLogLevel** property in the **[MAGIC_RIA]** section of the **Magic.ini** file.

Android

The application log is written in the internal device log.

You can see the log by using the Android Debug Bridge (ADB) utility, which is located in the **platform-tools folder** in your **SDK** folder, usually at: **C:\Program Files\Android\android-**

sdk\platform-tools.

To use the ADB utility on a mobile device, make sure that the mobile device is defined to:

- Run in debug mode. To enable it, go to **Settings/Applications/Development** and check the **USB Debugging** check box.
- Allow installation of a non-market application. To enable it, go to **Settings/Applications/Application Settings** and check the **Unknown sources** check box.

To view the device log, run the following command: **adb logcat**.

To view only the RIA client related entries in the device log, run the following command: **adb logcat MAGIC_DEBUG:D *:S**. Note that this filtering will not show crashes or illegal actions.

To clear the log, run the following command: **adb logcat -c**.

To save the log to a file, run the following command: **adb logcat > file.log**.

If you need to provide a log of a specific scenario or crash, you need to clear the log first, run the action that caused the crash and then save the full log to a file.

iOS

The application log can be downloaded to a PC using iTunes.

Simply connect your device to iTunes. In the device apps, select your application and then you can retrieve the log file.

Summary

You now have the tools to create your own testing environment. You can set up a testing environment before you try it out on your own tablet or mobile device. Bear in mind that you are working with an emulator and an emulator is slower than your device and is limited; therefore, there are some items that you may not be able to test satisfactorily, such as a camera or the GPS. You will however be able to test the general look and feel of the native application.