# isCOBOL Evolve: Mobile for Android

# Key Topics

- Overview
- Running the sample application
- Developing a hello world application from scratch
- Troubleshooting

# Table of Contents

# Chapter 1

# Overview

isCOBOL Mobile lets you run COBOL code on mobile devices. The goal is to reuse the existing backend COBOL logic as well as sequential, relative and indexed files (in JIsam format) on a mobile application while the UI is rewritten using a HTML5/CSS3 UI Framework such as jQuery Mobile or Dojo Mobile or Sencha Touch.

The code described in this book takes advantage of JQuery Mobile for the UI. You can find documentation about JQuery Mobile at http://view.jquerymobile.com/1.3.0/.

## Technical Notes About isCOBOL Mobile

The COBOL program has to be transformed to a stateful REST WebService. This is done by an internal class that allows communication with HTML pages using AJAX to retrieve data and print results.

The UI is rendered on mobile devices using the WebView component available on Android ADT while the COBOL logic runs locally on the device thanks to the new Framework library provided along with isCOBOL Mobile. This library is compatible with Android version 7 or greater.  It will interact with API functions on your mobile devices such as Phone Book, Memo, etc.

# Chapter 2

# Getting Started

This guide provides information for the installation on the Windows platform. If you wish to install the products on a Unix-like platform, contact Veryant.

The setup of a isCOBOL Mobile development environment requires the following steps:

1. Download and install the Java Development Kit (JDK)
2. Download and install isCOBOL Evolve (recommended) or alternatively Download and install isCOBOL Evolve SDK
3. Download and install the Android SDK
4. Activate the License

In order to activate your isCOBOL Evolve products, you will need the e-mail you received from Veryant containing your license key. Contact your Veryant representative for details.

## Download and install the Java Development Kit (JDK)

A JDK must be installed on your machine in order to use isCOBOL IDE and isCOBOL SDK. For best results and performance, install the latest JDK version available for your platform. isCOBOL IDE is certified to work correctly with both Oracle JDK and OpenJDK.

The development with isCOBOL Mobile requires a JDK version 8.

The isCOBOL IDE requires a JDK version 11 or 17.

If you plan to develop using the isCOBOL IDE (recommended), install both a JDK version 8 and a JDK version 11 or 17.

If you plan to develop using isCOBOL SDK on the command line, you can install just a JDK version 8.

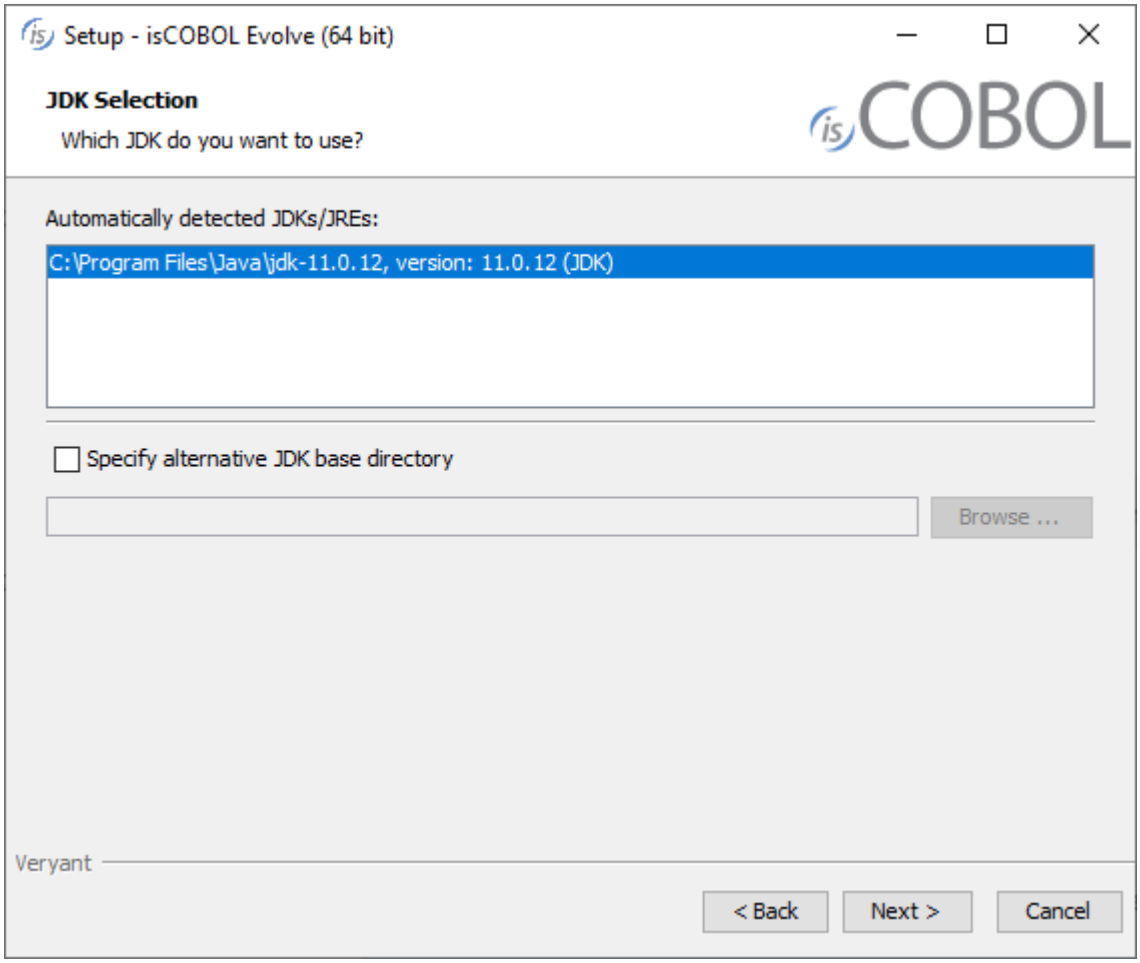
A self-extracting setup is provided for the Windows platform.

## Download and install isCOBOL Evolve

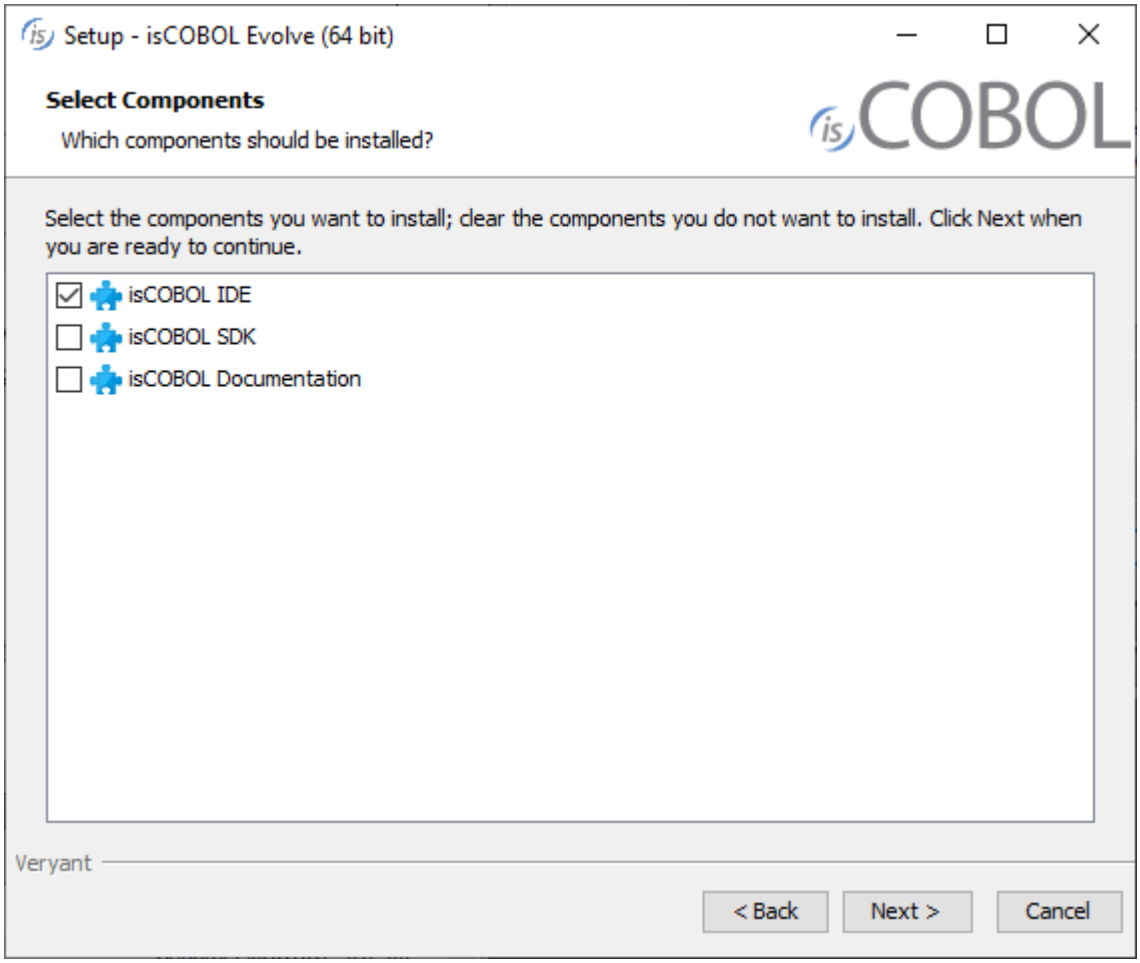
### Windows

1. If you haven't already done so, Download and install the Java Development Kit (JDK).
2. Go to "https://support.veryant.com".
3. Sign in with your User ID and Password.
4. Click on the "Download Software" link.

5. Scroll down to the list of files for Windows x64 64-bit. Select isCOBOL_Evolve2023_1_*n*_Windows_64.msi, where *n* is the build number.
6. Run the downloaded installer to install the files.
7. Select your JDK when prompted




8. Choose if you wish to install only the IDE or also the isCOBOL SDK. Installing the isCOBOL SDK will allow you to compile, run and debug from a command prompt, outside of the IDE.




9. Follow the wizard procedure to the end. In the process you will be asked to provide the installation path ("C:\Veryant" by default) and license keys. You can skip license activation and perform it later, as explained in Activate the License.

### Quiet mode

The isCOBOL Evolve setup supports the msi quiet mode. Settings can be driven with a response file.

A response file is a text file with name-value pairs that represent installer variables.

A response file is generated automatically after an installation is finished. The generated response file is found in the *.install4j* directory of the isCOBOL Evolve and is named *response.varfile*.

When an installer is executed, it checks whether a file with the same name and the *.varfile* extension can be found in the same directory and loads that file as the response file. For example, if an installer is named *foo_setup.msi* on Windows, the response file next to it has to be named *foo_setup.varfile*.

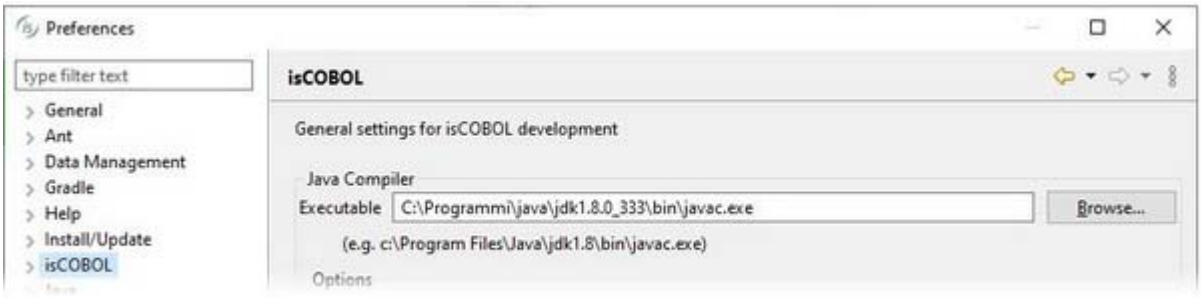
For more information about msi setups and their command line options, see Microsoft Standard Installer Command-Line Options.

## Bind Java 8 to the workspace

The isCOBOL IDE is bound to a JDK version 11 or 17, but the development of Android applications with isCOBOL Mobile requires a JDK version 8.

Although the IDE runs with a JDK version 11 or 17, you can associate a JDK version 8 to the workspace.

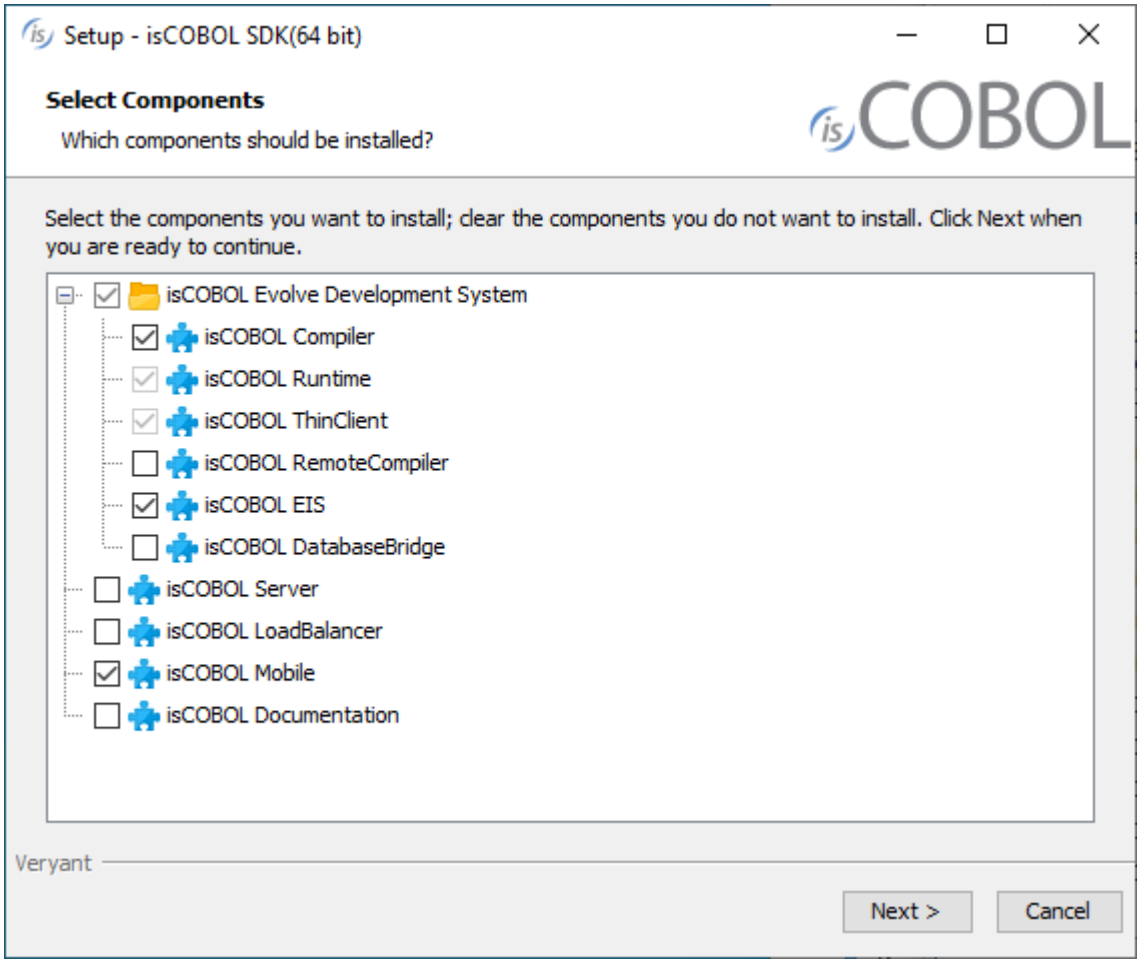
1. click *Window* in the menu bar and choose *Preferences*
2. select *isCOBOL* in the tree on the left
3. update the *Executable* field to point to a Java 8 compiler



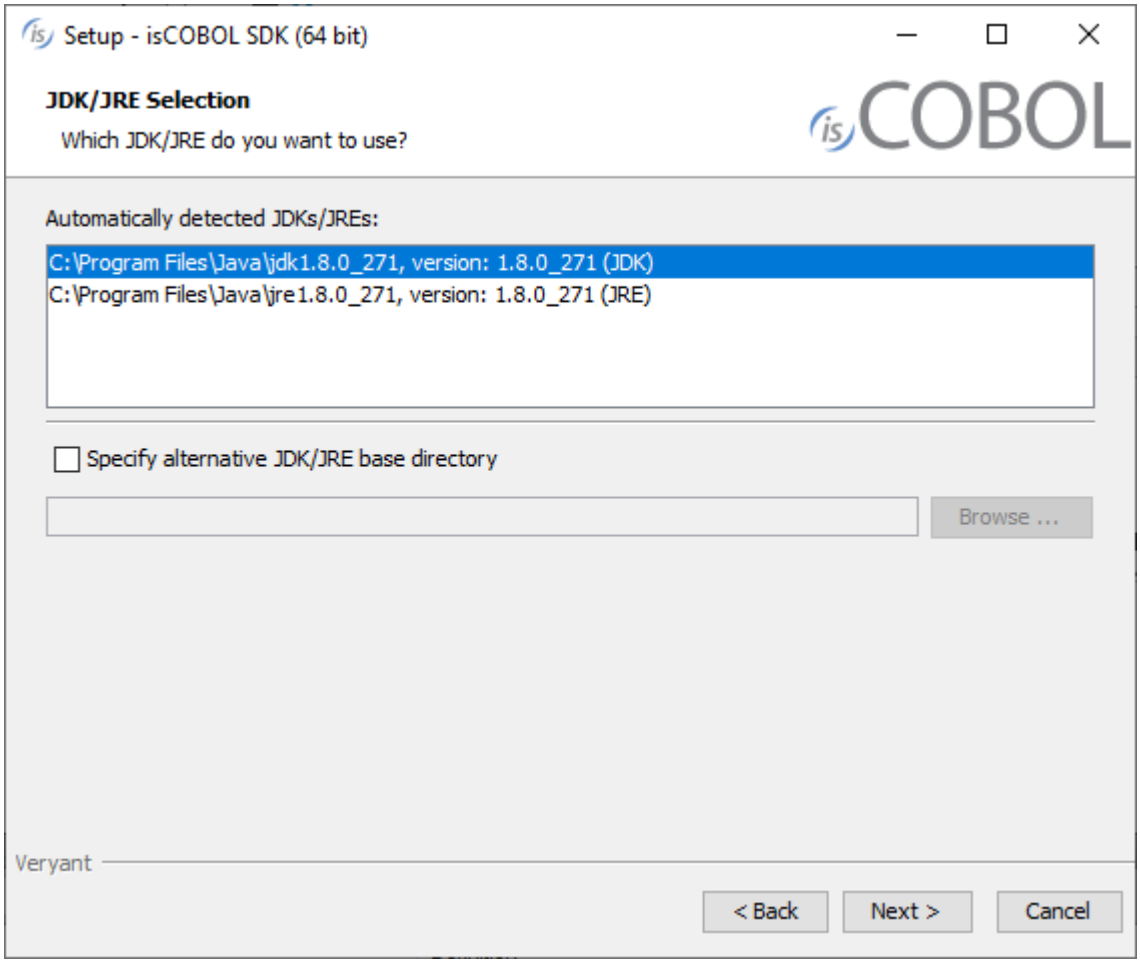

# Download and install isCOBOL Evolve SDK

## Windows

1. If you haven't already done so, Download and install the Java Development Kit (JDK) .
2. Go to "https://support.veryant.com".
3. Sign in with your User ID and Password.
4. Click on the "Download Software" link.
5. Scroll down to the list of files for Windows x64 64-bit or Windows x86 32-bit. Select isCOBOL_2023_R1_*n*_Windows.*arc*.msi, where *n* is the build number and *arc* is the system architecture.
6. Run the downloaded installer to install the files.
7. Select "isCOBOL EIS" and "isCOBOL Mobile" from the list of products when prompted.




8. Select your JDK when prompted




9. Follow the wizard procedure to the end. In the process you will be asked to provide the installation path ("C:\Veryant" by default) and license keys. You can skip license activation and perform it later, as explained in Activate the License.

## Quiet mode

The isCOBOL SDK setup supports the msi quiet mode. Settings can be driven with a response file.

A response file is a text file with name-value pairs that represent installer variables.

A response file is generated automatically after an installation is finished. The generated response file is found in the *.install4j* directory of the isCOBOL SDK and is named *response.varfile*.

When an installer is executed, it checks whether a file with the same name and the *.varfile* extension can be found in the same directory and loads that file as the response file. For example, if an installer is named *foo_setup.msi* on Windows, the response file next to it has to be named *foo_setup.varfile*.

For more information about msi setups and their command line options, see Microsoft Standard Installer Command-Line Options.

# Download and install the Android SDK

The Android SDK provides you with the API libraries and developer tools necessary to build, test, and debug apps for Android.

In order to download the latest Android SDK supported by isCOBOL, use the following URL: https://dl.google.com/android/installer_r24.4.1-windows.exe .

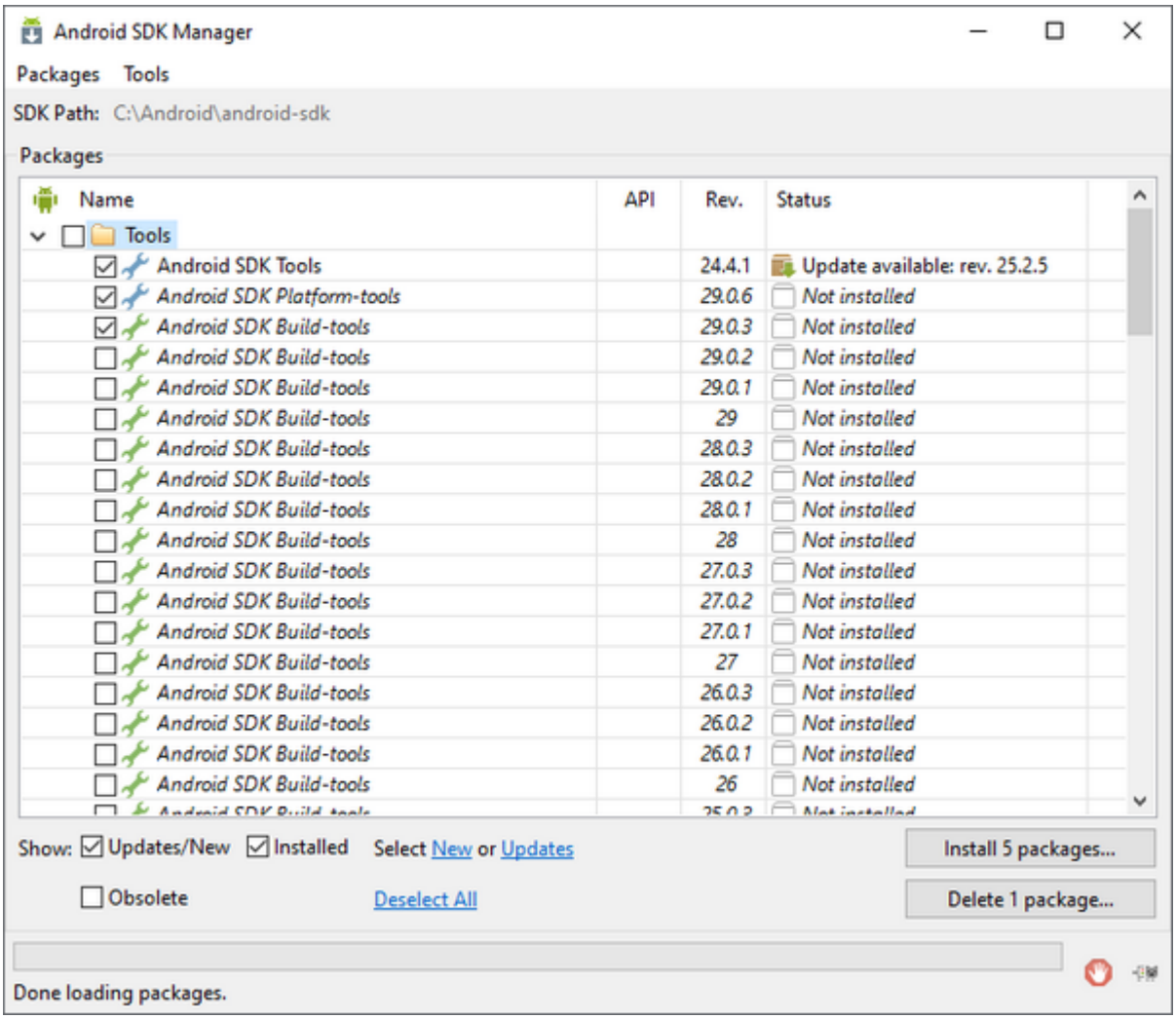
Once the download is complete, run the installer and follow the wizard procedure. You will be asked to provide the destination directory.

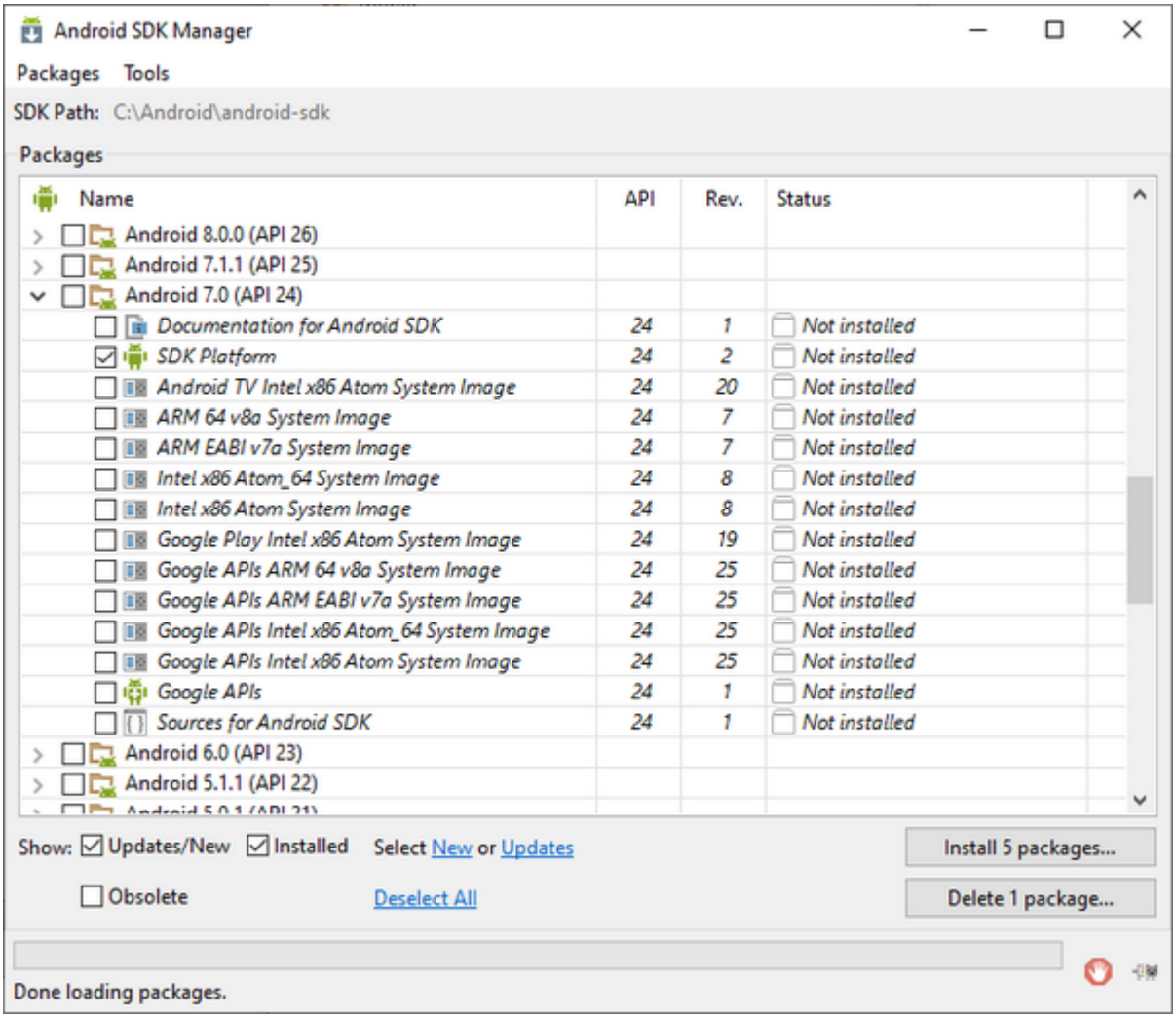
Once the installation is completed, start the SDK manager (*SDK Manager.exe*).

By default the SDK Manager prompts you to install the latest platform tools, system image and USB driver, but you can also install previous versions as well as additional resources.

Being bound to Java version 8, the minimum Android API for isCOBOL Mobile is "Android 7.0 (API level 24)". You can install it alone or along with more recent Android APIs, or you can install just a more recent Android API.

The following screenshots depict the suggested installation.






After you marked the items that you wish to install, then click on the *Install packages...* button.

At the end of the installation, you will find a new folder named *platform-tools* at the same level of *tools*.

Once the Android SDK has been installed, add the *tools* and *platform-tools* directories of Android's SDK to the PATH environment variable in order to be able to run android commands from the command line.

## Download and install Apache ANT and Tomcat (or another servlet container)

Apache ANT and Tomcat (or another servlet container) must be installed if you plan to work without isCOBOL IDE. If you plan to develop using the isCOBOL IDE, you can skip this chapter.

Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The Android SDK uses ANT in order to generate the apk of your mobile app.

The Apache ANT main page is https://ant.apache.org

1. Download the latest binary files from https://ant.apache.org/bindownload.cgi
2. Unzip the binary distribution in a folder of your choice (i.e. C:\ANT)
3. Add the *bin* directory of ANT to the PATH environment variable

The servlet containers like Tomcat allow you to run your app as a web application making it usable by every device equipped with a web-browser including Apple iOS devices. Running the app as web application is a good preliminary test before moving to the Android device.

The Apache Tomcat main page is https://tomcat.apache.org/

Here are some example steps to download and install Tomcat 8.5 on Windows:

1. Visit https://tomcat.apache.org/
2. Click on the *Tomcat 8* Download link (on the left side)
3. Find the *Binary Distributions* section and click on the *Windows Service Installer* link
4. Run the downloaded executable file and follow the prompts accepting the defaults

## Activate the License

If you provided license keys during the installation you should skip reading this chapter.

isCOBOL Mobile looks for the following configuration properties for the license keys at compile-time:

```
iscobol.compiler.license.2023=<license_key>
iscobol.eis.license.2023=<license_key>
iscobol.mobile.license.2023=<license_key>
```

These keys should be stored in one of the following files (if they exist):

1. \etc\iscobol.properties in the drive where the working directory is
2. C:\Users\<username>\iscobol.properties (the setup wizard saves licenses here, if you don't skip activation)
3. iscobol.properties found in the Java Classpath
4. a custom configuration file passed on the command line
5. %ISCOBOL%\iscobol.properties

**NOTE** - Files are listed in the order they're processed. If the license keys appears in more than one of the above files, then the last occurrence is considered.

# Chapter 3

# Running the sample application

isCOBOL comes with a sample application named Password Reminder.

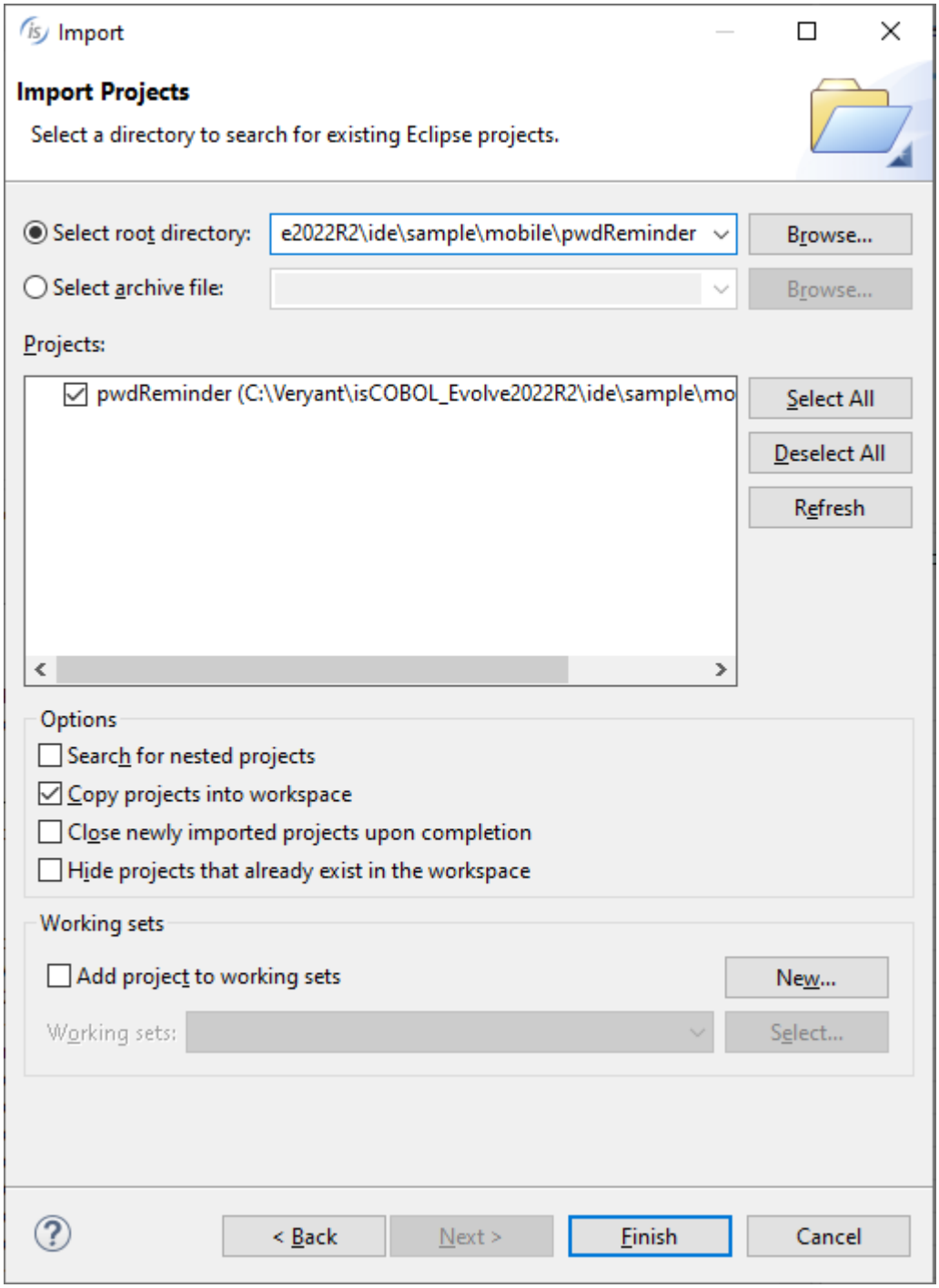
This chapter explains how to deploy and run the sample application.

## Running the sample application in isCOBOL IDE and exporting it to APK

isCOBOL IDE includes a sample project that shows how an isCOBOL Mobile application is created and installed.

Follow these steps to import the projects in the IDE and test them:

1. Create a new workspace or open an existing one
2. Click on the *File* menu
3. Choose *Import*
4. Choose *General* > *Existing Project into Workspace*
5. Having *Select root directory* checked, browse for the folder "C:\Veryant\isCOBOL_Evolve2023R1\ide\sample\mobile"

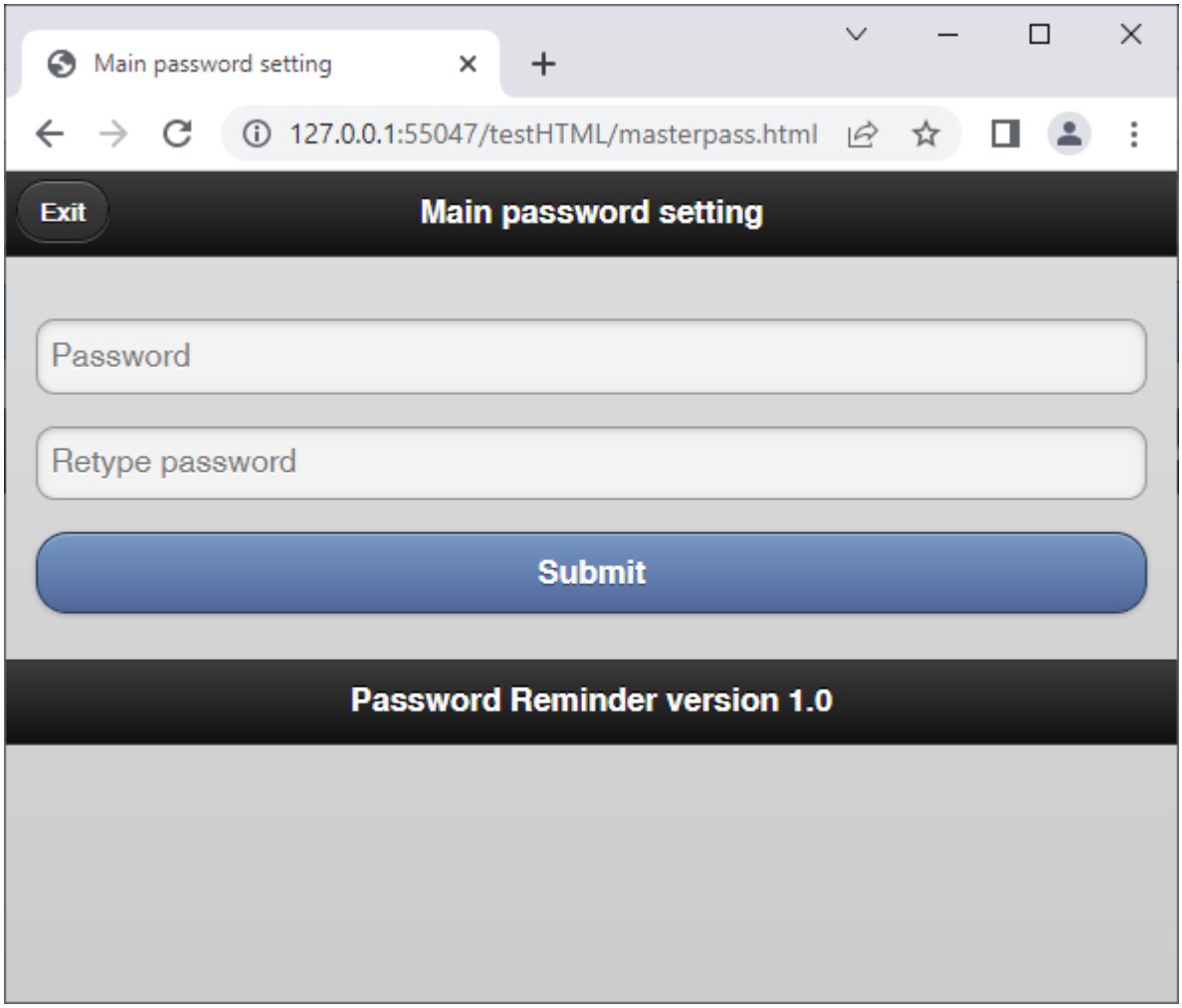



6. Optionally check the option *Copy project into workspace* and click *Finish*

At this point you can run the sample application as EIS Servlet.

To run as EIS Servlet:

1. Right click on *pwdReminder_COBOL_HTML* in the Explorer tree
2. Choose *Run As > isCOBOL EIS Servlet*



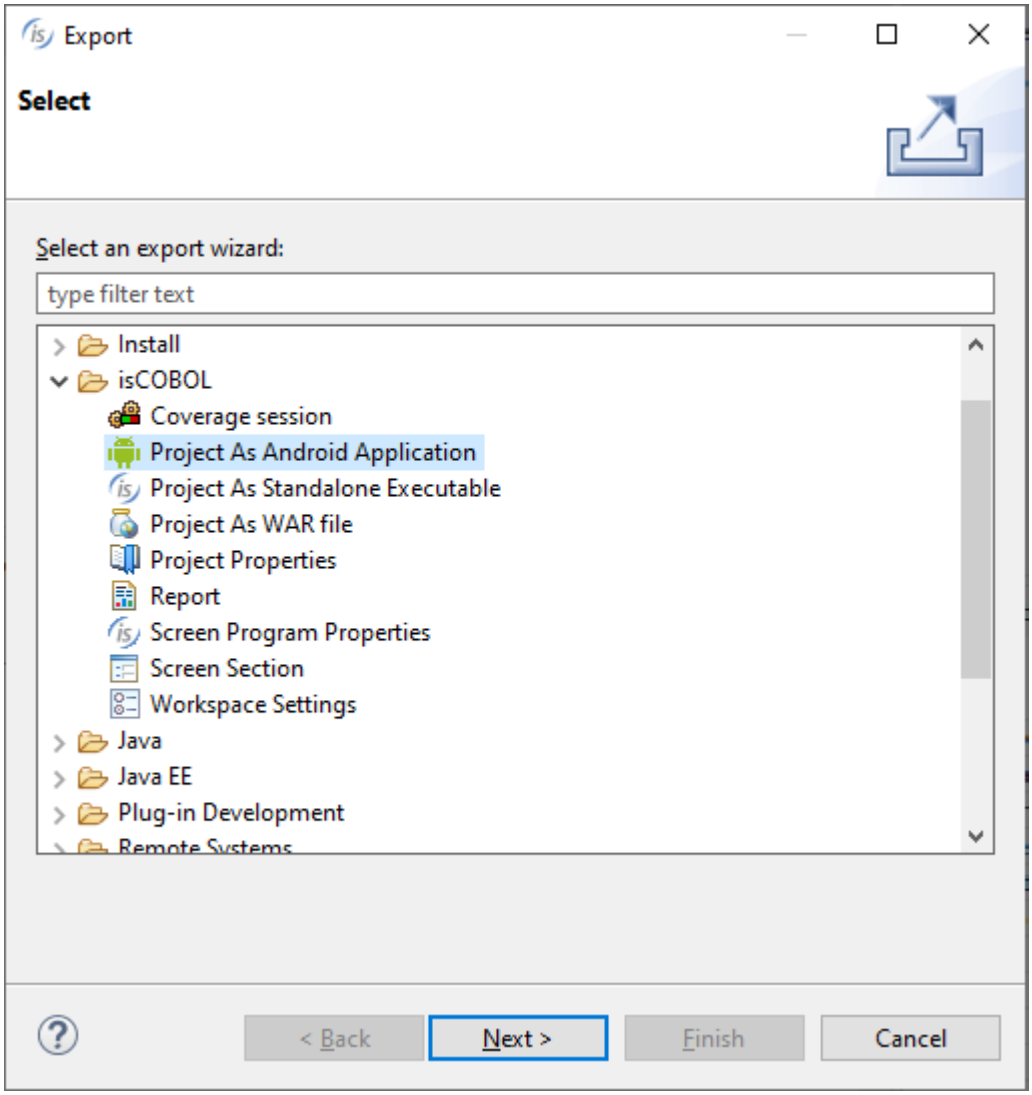

**Note -** for best rendering of the CSS styles we suggest you use an external browser rather than the internal Eclipse browser. To use an external browser, do this before performing the steps above:

1. Click on *Window* in the menu bar
2. Choose *Preferences*
3. Choose *General* > *Web Browser* in the tree
4. Switch from "Use internal web browser" to "Use external web browser"
5. Click on the *Apply* button

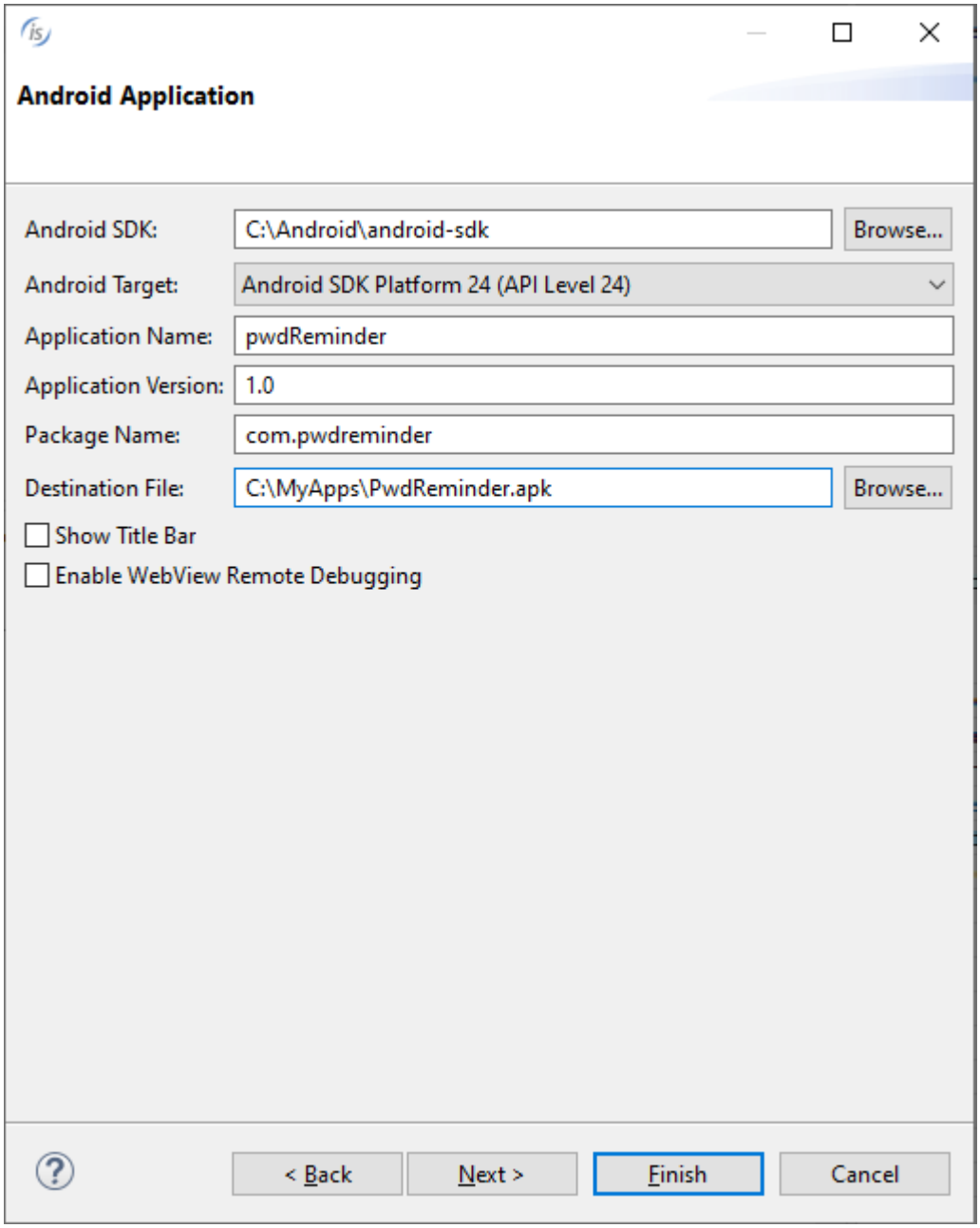
Once you've tested the application through the web-browser, it's time to make it and Android app.

1. Right click on *pwdReminder_COBOL_HTML* in the Explorer tree
2. Choose *Export*

3. Choose *Project as Android Application* from the *isCOBOL* tree




4. Fill the fields by providing the Android SDK location, the desired target version and the folder where the apk should be generated. Example:




5. click on the *Finish* button

See How to install and use the sample application for further information.

## Creating the sample application APK outside of isCOBOL IDE

The isCOBOL *samples* directory includes a folder named *mobile* that contains a folder named *pwd_Reminder* with all the necessary files to build and test the sample from command line..

1. Change to the directory *C:\Veryant\isCOBOL_SDK2023R1\sample\mobile\pwd_Reminder*
2. Edit the file *local.properties* to specify the location of the Android SDK

3. Edit the file *project.properties* to specify the Android target version.
   The target must be installed in your SDK. You can run the command

```
android list targets
```

   to obtain the list of available targets.

4. Copy *ismobile.jar* from *C:\Veryant\isCOBOL_SDK2023R1\mobile\lib* to *libs*
5. Compile the PASSWD program and put it in a jar file named cobol.jar under the "libs" folder:

```
cd src
iscc PASSWD.cbl
jar -cf ..\libs\cobol.jar PASSWD.class
cd ..
```

6. Run this command:

```
ant debug -Dout.final.file=PwdReminder.apk
```

7. Find *PwdReminder.apk* under the *bin* folder.

See How to install and use the sample application for further information

# How to install and use the sample application

## Installation

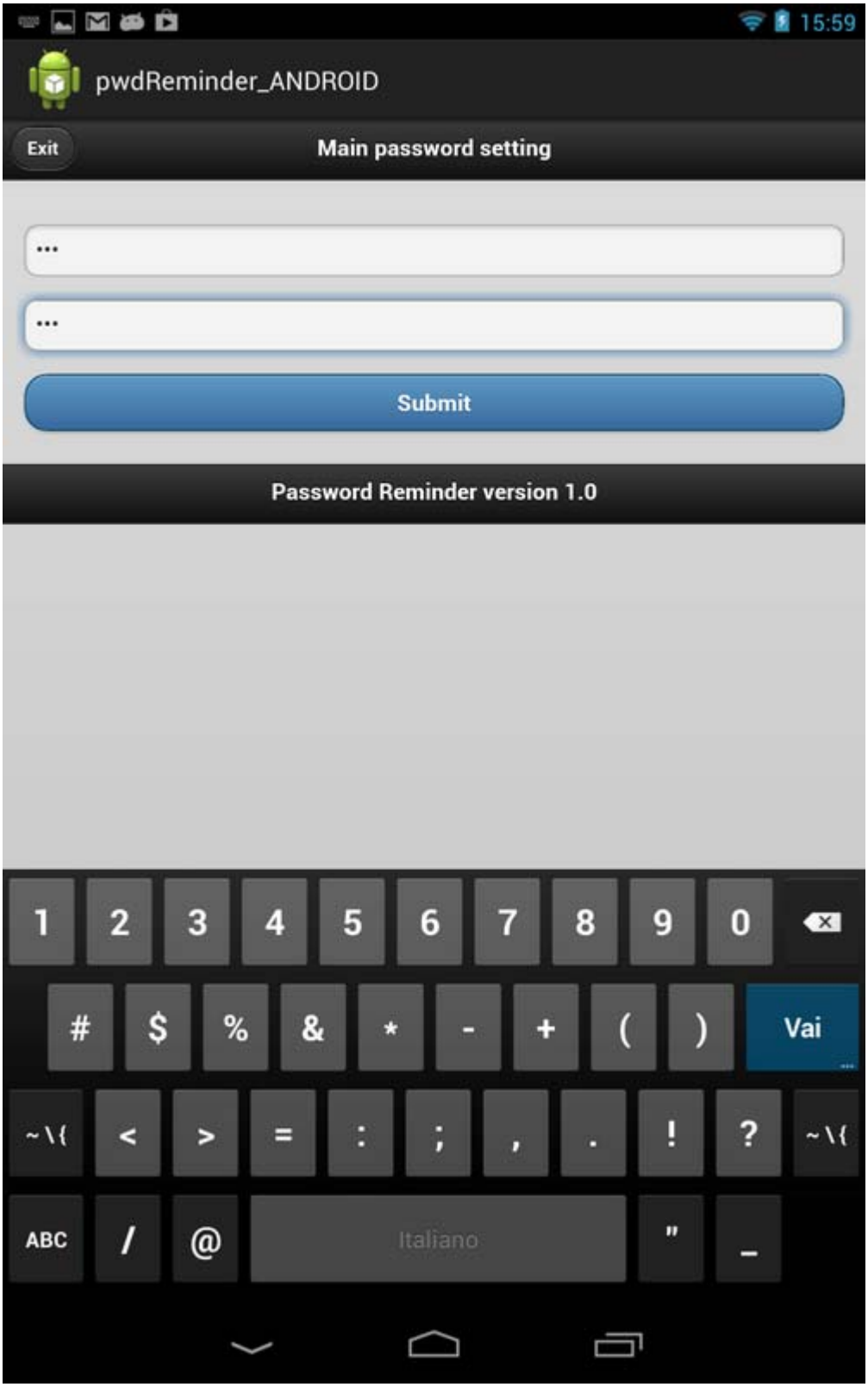
The apk that you created can be tested with an Android emulator as well as on a real physical Android device. For better performance and a more accurate outcome, the physical device is suggested.

In order to install the apk on a physical device, copy the apk to the device through USB or network file transfer features, then install it.

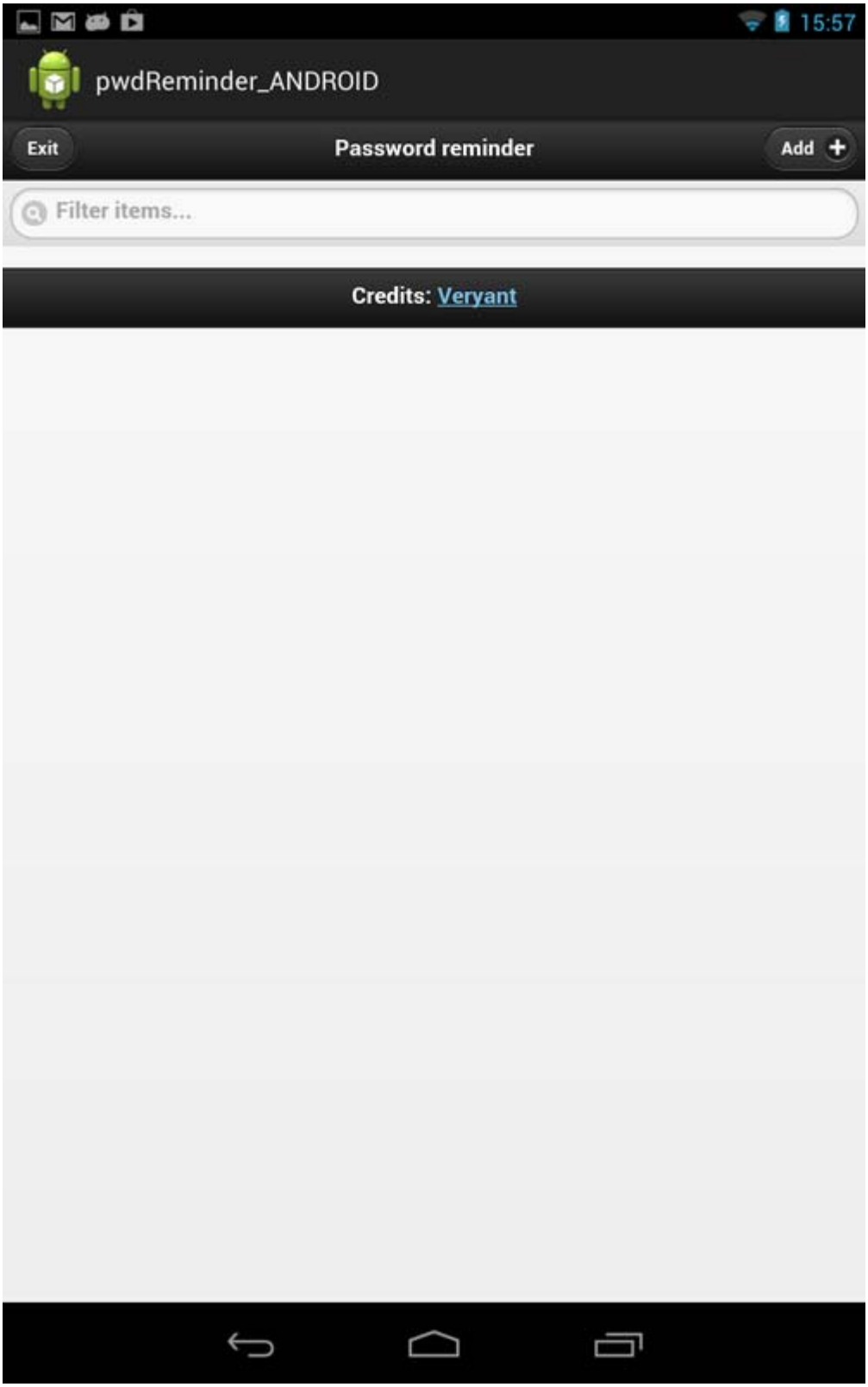
Find Password Reminder among the installed apps and touch it to start.

## Usage

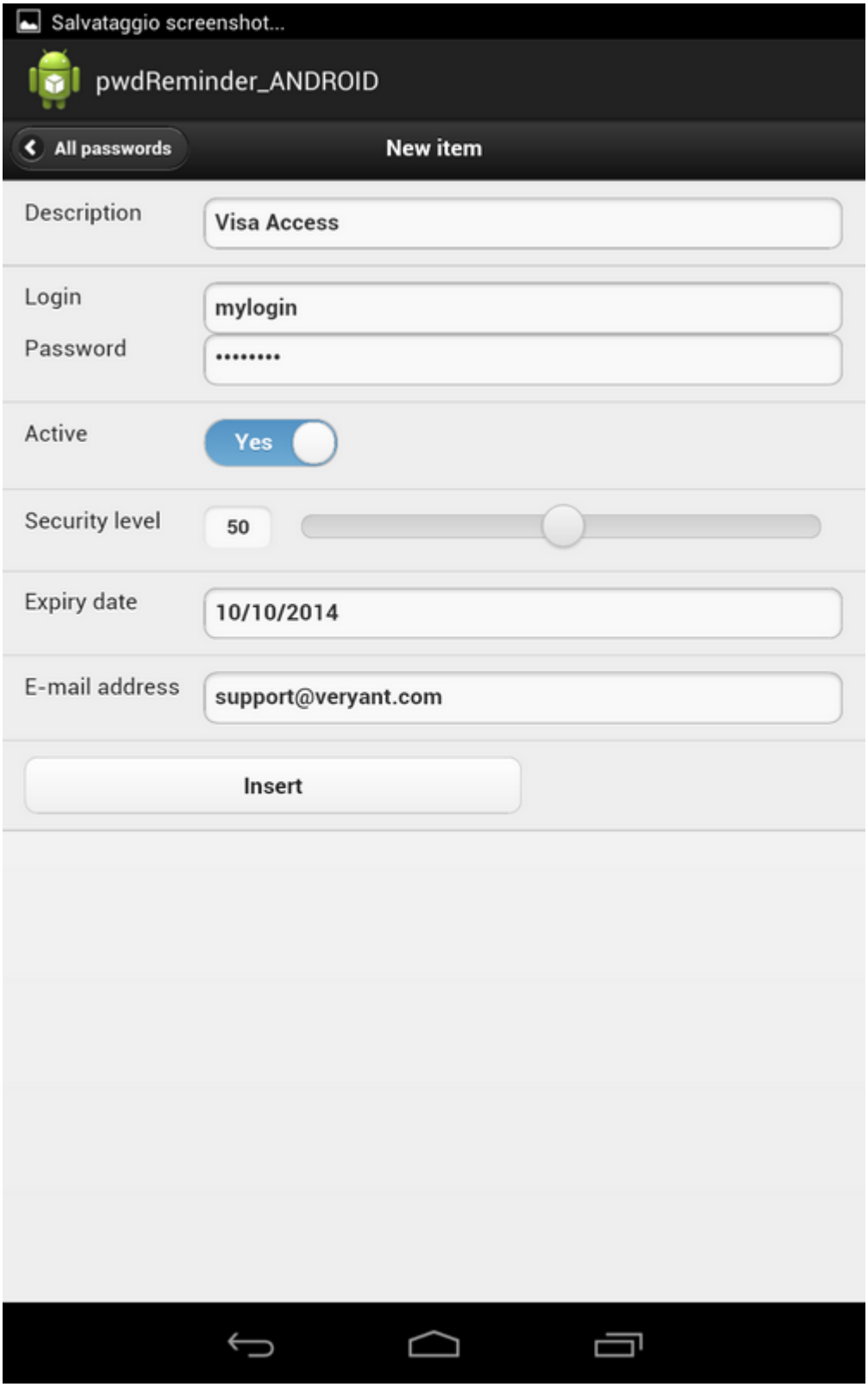
The following screen will appear in your emulator.


15:59
pwdReminder_ANDROID
Exit
Main password setting
Submit
Password Reminder version 1.0
Vai
ABC
Italiano

1. You will need to create a main password to proceed. Type the same text in both fields and touch *Submit* to reach the following screen:

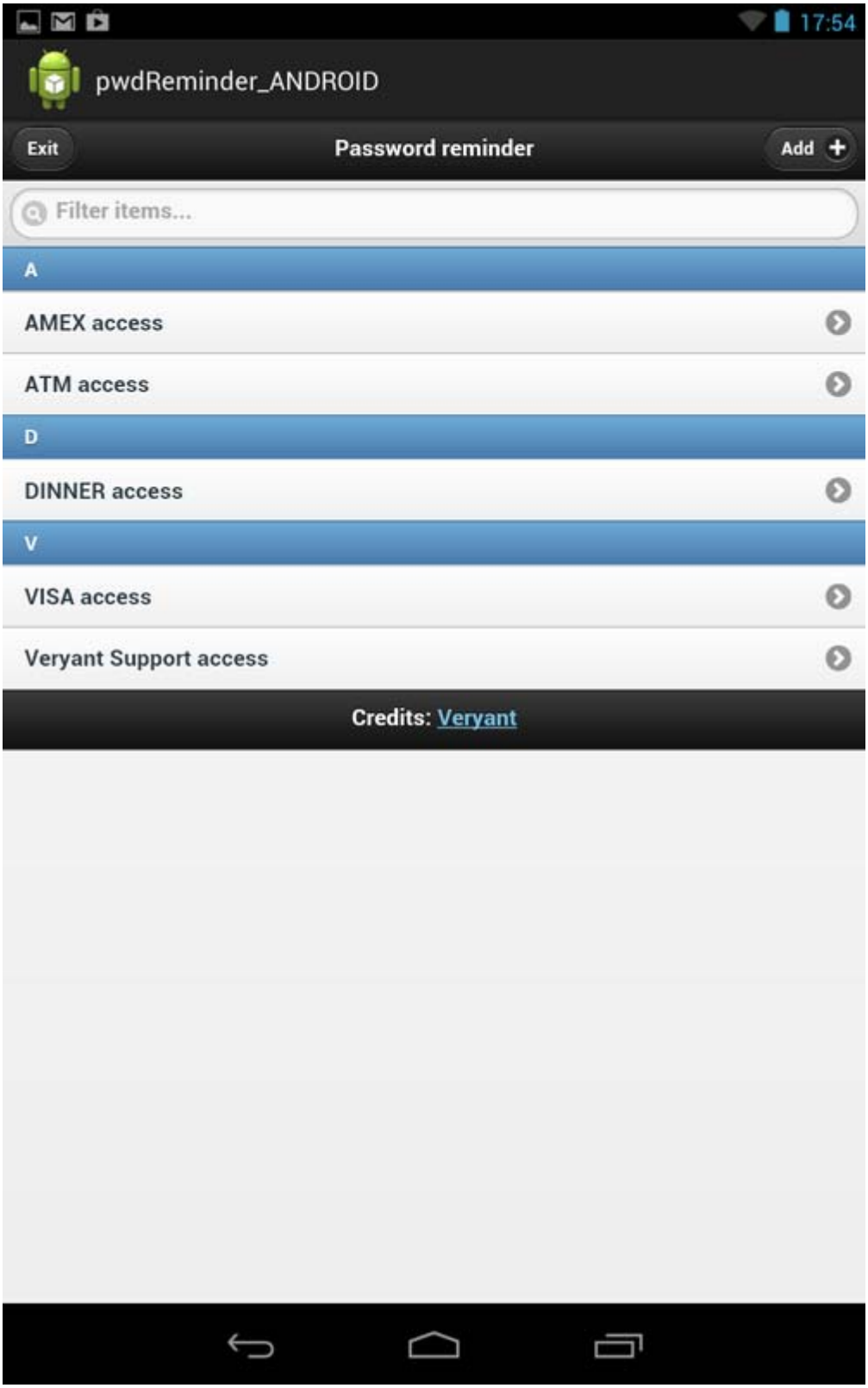



2. Touch the *Add* button to reach the "New item" screen where you can put free data, for example:




3. Touch the *All passwords* button to return to the list of passwords shown at step 1.

4. Repeat steps 2 and 3 to insert more items. At the end, the list of passwords should look like this:



# Chapter 4

# Development

## Developing a hello world application from scratch

The next chapter illustrates the steps to create an hello world application from scratch.

You will need to produce and maintain two kind of source codes:

1. an HTML/JavaScript user interface
2. a COBOL program that receives requests from the HTML interface and returns results to the interface

And you will create:

- a web application that can be used by any web-browser enabled device including Apple iOS devices
- an Android App that can run on Android devices natively

### Building the COBOL program

#### Using isCOBOL IDE

1. Click on the *File* menu
2. Choose *New*
3. Choose *isCOBOL Project*
4. Give it the name "simple_COBOL_HTML"
5. Click on the *Next* button until you reach the page where you can set *Compiler/Runtime options*
6. It's good practice to compile with the -whttp option in order to be advised if our program contains statements that are not supported in the Mobile environment. Switch to the "W" page and check the option "-whttp".
7. Click on the *Finish* button to confirm settings and complete the project creation.
8. Right click on the *source* folder and choose *New*
9. Choose *Source File*
10. Give it the name "hello.cbl" and click on the *Finish* button
11. Put the COBOL code into the program and compile it.

#### Without isCOBOL IDE

1. Create an empty text file and give it the name "hello.cbl"

2. Put the COBOL code into the file
3. Open a command prompt and change to the directory where the file is
4. It's good practice to compile with the -whttp option in order to be advised if our program contains statements that are not supported in the Mobile environment.
   Compile the program with the command:

```
iscc -whttp hello.cbl
```

5. HELLO.class will be created created; you need to include it in a jar to use it in the future steps. Use the following command:

```
jar -cvf cobol.jar HELLO.class
```

COBOL code

```
       PROGRAM-ID. hello.
       CONFIGURATION SECTION.
       REPOSITORY.
           class web-area as "com.iscobol.rts.HTTPHandler"
           .

       WORKING-STORAGE SECTION.
       01 hello-buffer identified by "_comm_buffer".
          03  filler identified by "_status".
              05 response-status pic x(2).
          03  filler identified by "_message".
              05 response-message pic x any length.
          03  filler identified by "hellotext".
              05  xml-hellotext pic x any length.

       LINKAGE SECTION.
       01  lnk-area object reference web-area.

       PROCEDURE DIVISION using lnk-area.
       main-logic.
           move "Operation successful" to response-message.
           move "OK" to response-status.
           move "Hello World from isCOBOL!" to xml-hellotext.
           lnk-area:>displayXML (hello-buffer).
           goback.
```

**Note -** The above code takes advantage of the HTTPHandler class (com.iscobol.rts.HTTPHandler) for communicating with the HTML user interface (explained later).

## Building the HTML interface and the web application

### Using isCOBOL IDE

1. Right click on the *html* folder
2. Choose *New* > *Other...*
3. Choose *Web* > *HTML File...*
4. Click on the *Next* button

5. Give it the name "index.html" and click *Finish*
6. Put the Content of Index.html in the file
7. Download JQuery scripts and css files (see Where to find JQuery files) and copy them to the *html* folder of the project

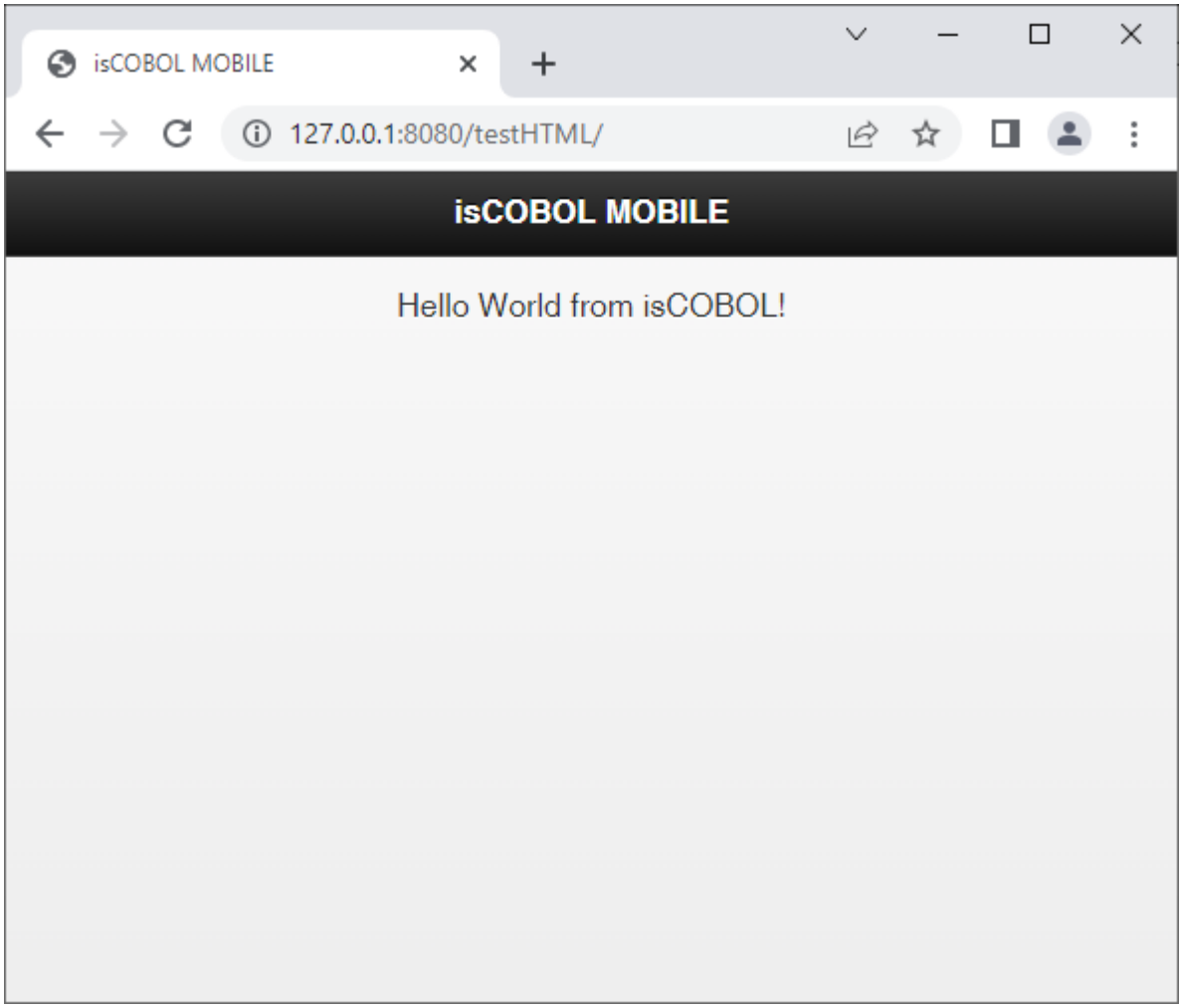
At this point you can test the HTML application in the Eclipse to see how it will look on Android. So far you've produced a web application that can run on a web server and be used by any browser enabled device.

**Note -** Veryant suggests you run in an external browser instead of the IDE's internal browser. Set your preferences for an external browser with these steps:

1. Click on the *Window* menu
2. Choose *Preferences*
3. Expand *General* and select *Web Browser* in the tree
4. Check the "Use external web browser" option
5. Click *Ok*

To test the application, proceed as follow:

1. Right click on the project name in the tree
2. Choose *Run As > isCOBOL EIS Servlet*




If you wish to debug the COBOL program, choose *Debug As* > *isCOBOL EIS Servlet*, instead

**Note -** The debug of an 'EIS Servlet' runs in a remote debug session of the Jetty application server included in the IDE. The debugger suspends itself at first to wait for a connection from the client. When the debugged program ends, the Debugger session doesn't terminate as it would happen in a stand alone Debugger session. The connection with the client part is still active. So the Debugger server goes in a 'continue' state, it means that it will suspend only when a breakpoint is reached. Therefore, in order to debug the program another time, it is necessary to set some breakpoints.

Before going to the next step, running the simple hello world application on Android, you'll need to provide a valid isCOBOL Mobile license key in the file *iscobol.properties* under the *resources* folder.

Edit the file *iscobol.properties* under the *html* folder and insert your license key. When you're done the file should look like this:

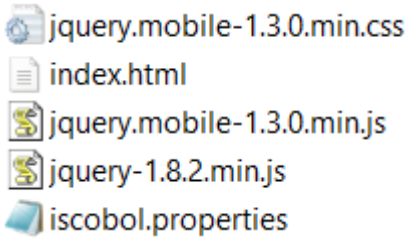
```
iscobol.mobile.license.2023=<your license code>
iscobol.exception.message=2
```

### Without isCOBOL IDE

1. Create a new file named “Index.html” and put the Content of Index.html in it.
2. Download JQuery scripts and css files (see Where to find JQuery files) .
3. Create a file named “iscobol.properties” and add the following entries to it:

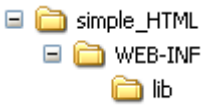
```
iscobol.mobile.license.2023=<your license code>
iscobol.exception.message=2
```

4. Using external software create a zip archive named "html.zip" and include the above items into it. The zip should contain:



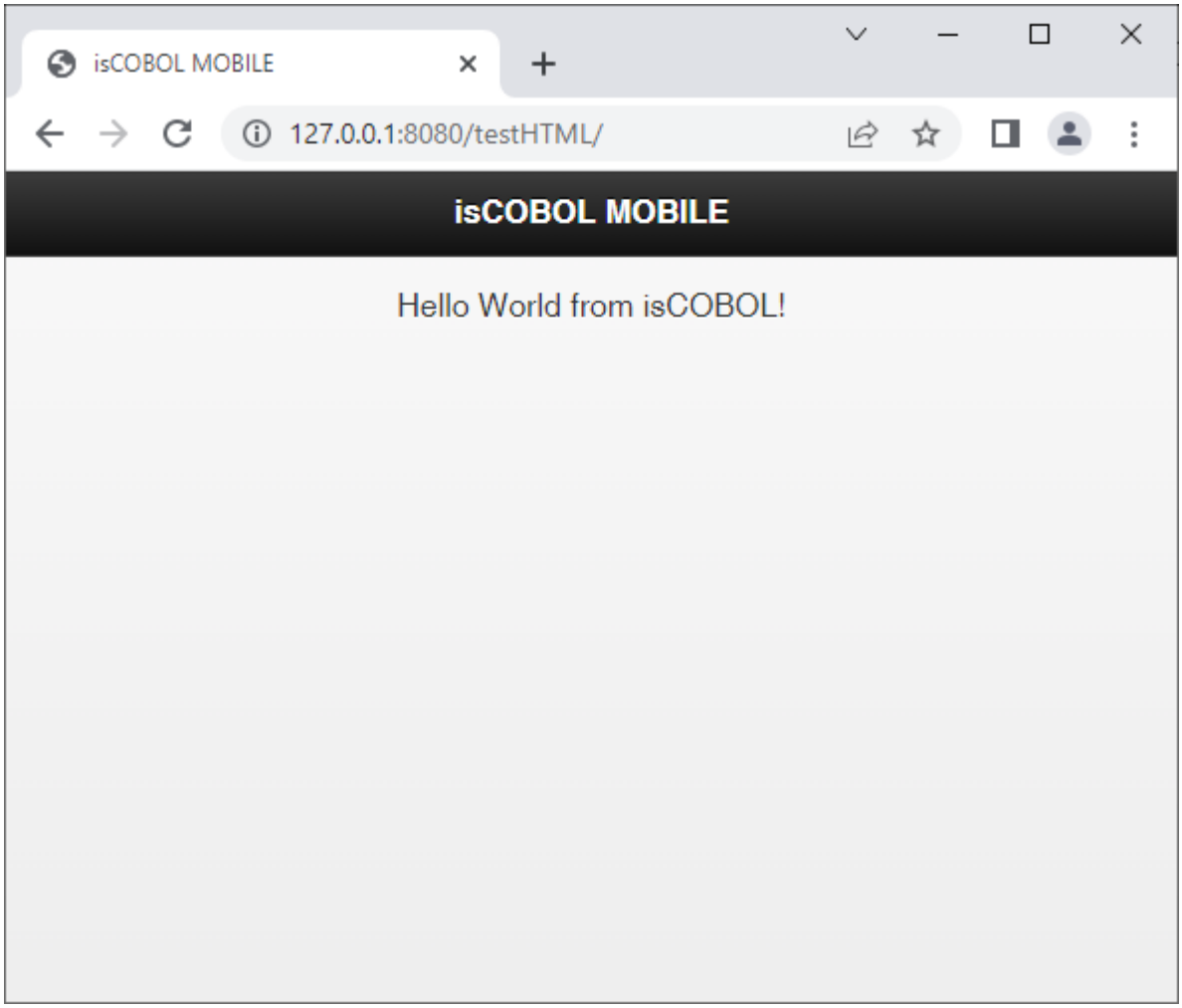

At this point you can test the application in a servelet container. The following steps show how to test in Tomcat.

1. Create the following folder structure under Tomcat’s *webapps* directory:




2. Put the following files under *simple_HTML*:
   - index.html
   - jquery-1.8.2.min.js
   - jquery.mobile-1.3.0.min.css
   - jquery.mobile-1.3.0.min.js
3. Create a file named “web.xml” under *WEB-INF* and put the Content of Web.xml into it.
4. Put the following files under *lib*:
   - cobol.jar (previously created)
   - iscobol.jar (found in C:\Veryant\isCOBOL_SDK2023R1\lib)
5. Start the Tomcat service
6. Assuming that you’re testing on localhost and Tomcat is started on the default port, navigate to “http://

127.0.0.1:8080/simple_HTML”



- Content of Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance" xmlns="https://
java.sun.com/xml/ns/javaee" xmlns:web="https://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" xsi:schemaLocation="https://java.sun.com/xml/ns/javaee https://
java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>testHTML</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
  <filter>
        <filter-name>isCOBOL filter</filter-name>
        <filter-class>com.iscobol.web.IscobolFilter</filter-class>
  </filter>
  <filter-mapping>
        <filter-name>isCOBOL filter</filter-name>
        <url-pattern>/servlet/*</url-pattern>
  </filter-mapping>
  <servlet>
        <servlet-name>isCobol</servlet-name>
        <servlet-class>com.iscobol.web.IscobolServletCall</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>isCobol</servlet-name>
        <url-pattern>/servlet/*</url-pattern>
  </servlet-mapping>
  <listener>
   <listener-class>com.iscobol.web.IscobolSessionListener</listener-class>
  </listener>
</web-app>
```

Content of Index.html

```
<html>
    <head>
    <title>Test Mobile </title>
   <link href="jquery.mobile-1.3.0.min.css" rel="stylesheet" type="text/css" />
   <script src="jquery-1.8.2.min.js"></script>
   <script src="jquery.mobile-1.3.0.min.js"></script>


   <script>
   function handleError (jqXHR, textStatus, errorThrown) {
      alert (textStatus +" "+ jqXHR.status + " "+jqXHR.statusText +
                  "\n" + jqXHR.responseText);
   }
   function handleSuccess (data, textStatus, jqXHR) {
      response = jqXHR.responseText;
      try {
         xmlDoc = jQuery.parseXML (response);
      } catch (err) {
         alert (response);
         return false;
      }
      xml = jQuery (xmlDoc);
      _status = xml.find ("_status");
      _message = xml.find( "_message" );
      _hello = xml.find( "hellotext" );
      jQuery("#hello_div").html(_hello.text());
      return true;
   }

   function callServer (cobolProg) {
      var url = "servlet/isCobol(" + cobolProg + ")";
      jQuery.ajax(url, {
                  success: handleSuccess,
                  error: handleError
               });
      return false;
   }

   window.onload = callServer("HELLO");
   </script>
```

```
</head>
<body>

<div data-role="page">

    <div data-role="header" data-theme="a">
        <h1>isCOBOL MOBILE</h1>
    </div><!-- /header -->

    <div data-role="content" data-fullscreen="true">
        <div id="hello_div" align="center"></div>
    </div><!-- /content -->

</div><!-- /page -->

</body>
</html>
```

### Where to find JQuery files

JQuery script and css files used by Index.html can be downloaded from the following sites:

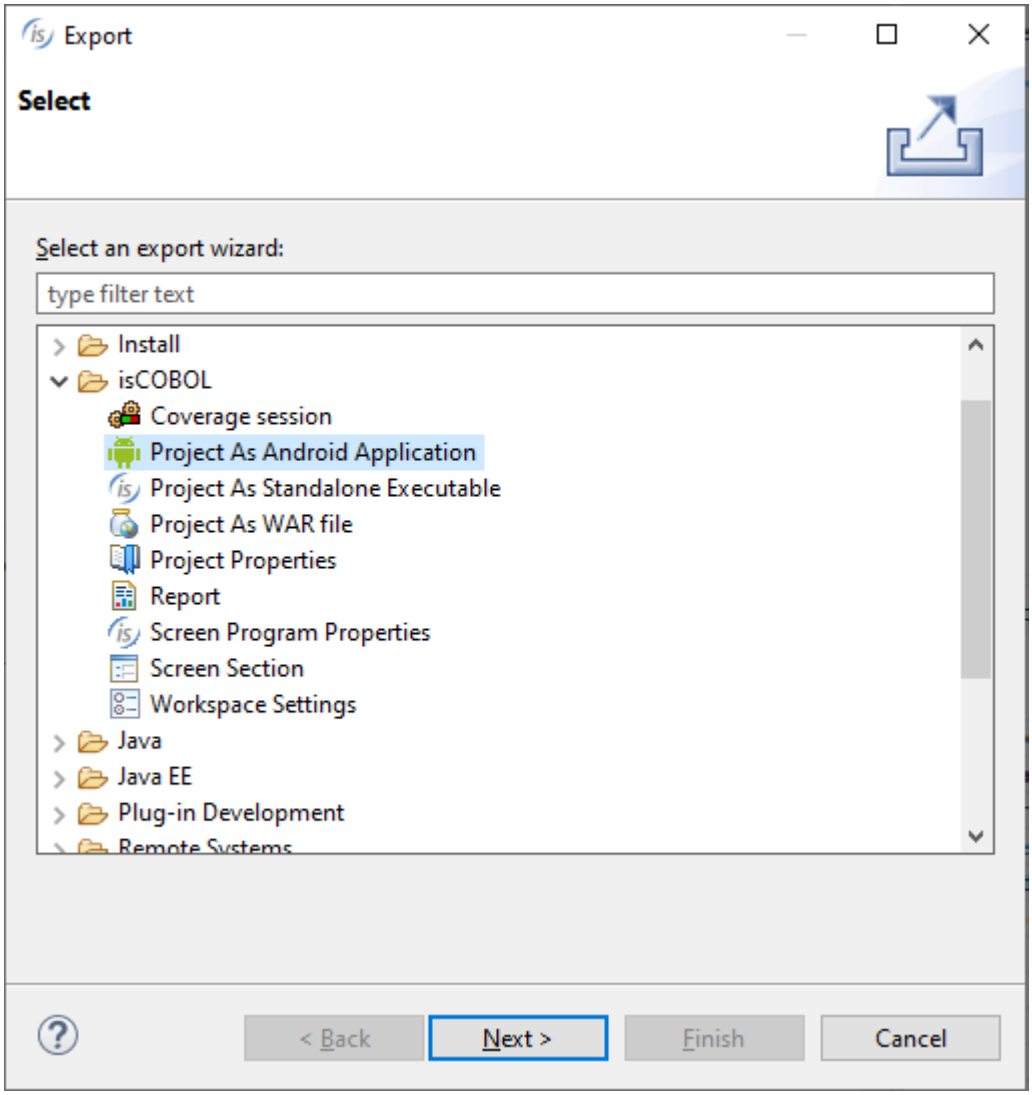
| File | Web Site | Direct URL |
|---|---|---|
| jquery-1.8.2.min.js | https://jquery.com | https://code.jquery.com/jquery-1.8.2.min.js |
| jquery.mobile-1.3.0.min.css | https://jquerymobile.com/ | https://code.jquery.com/mobile/1.3.0/jquery.mobile-1.3.0.min.css |
| jquery.mobile-1.3.0.min.js | https://jquerymobile.com/ | https://code.jquery.com/mobile/1.3.0/jquery.mobile-1.3.0.min.js |

## Creating the Android App

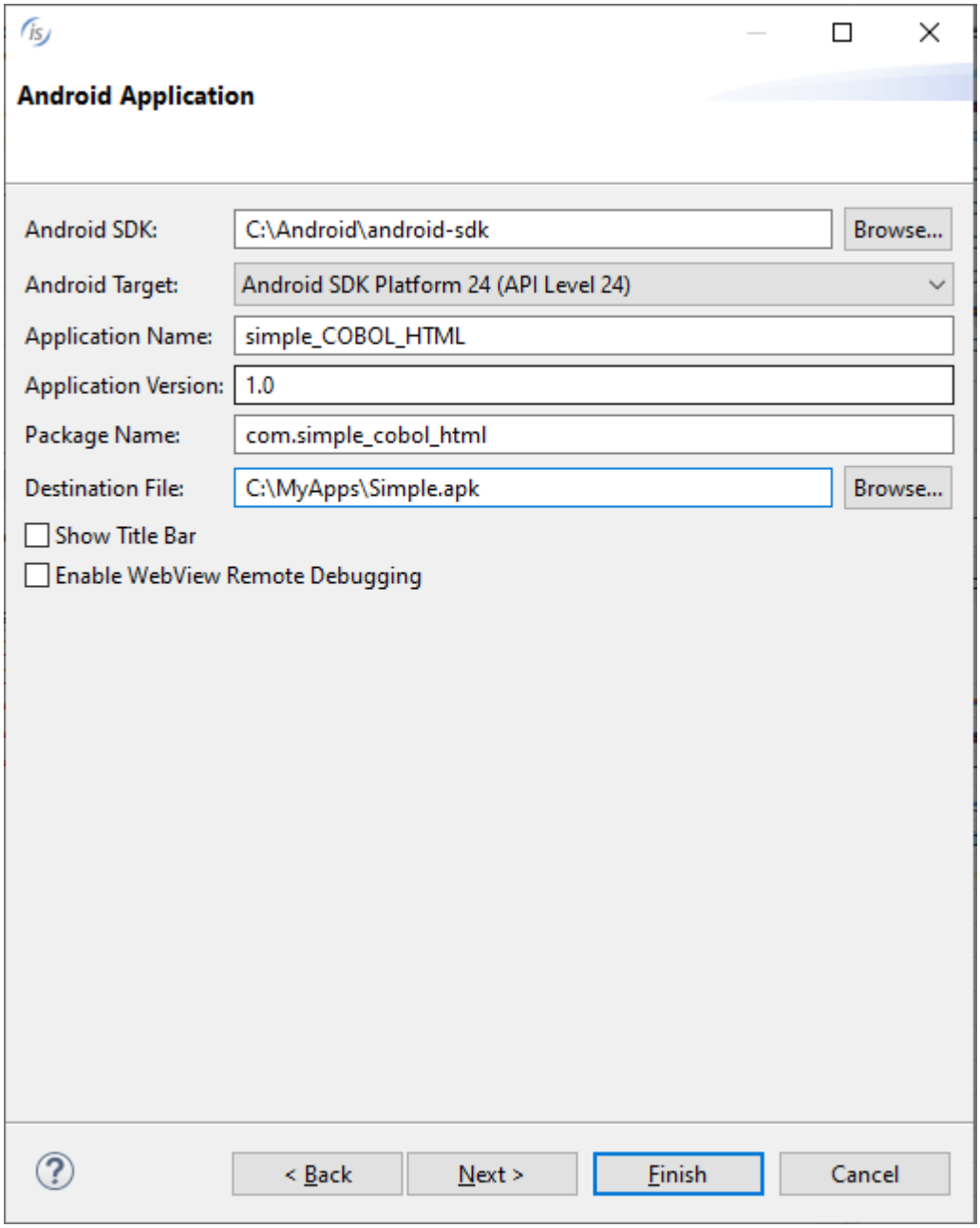
### Using isCOBOL IDE

1. Right click on *simple_COBOL_HTML* in the Explorer tree
2. Choose *Export*

3. Choose *Project as Android Application* from the *isCOBOL* tree




4. Select the desired project from the list, then click *Next*
5. Provide the required information: Android SDK location, target version, package and the folder where the generated apk should be saved. Example:

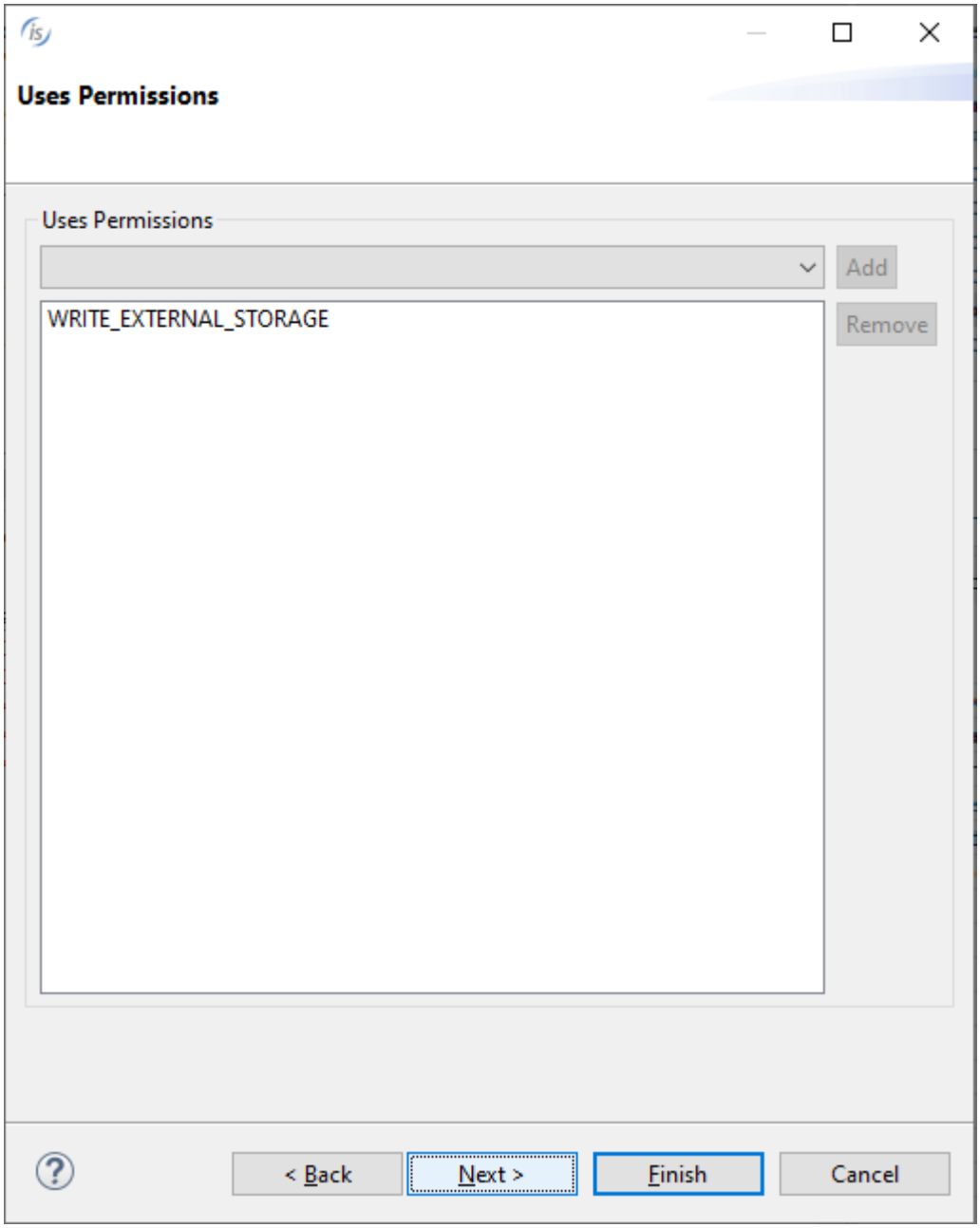



**Note** - Do not use the same package for more applications. The package is a sort of unique ID in the Android system.

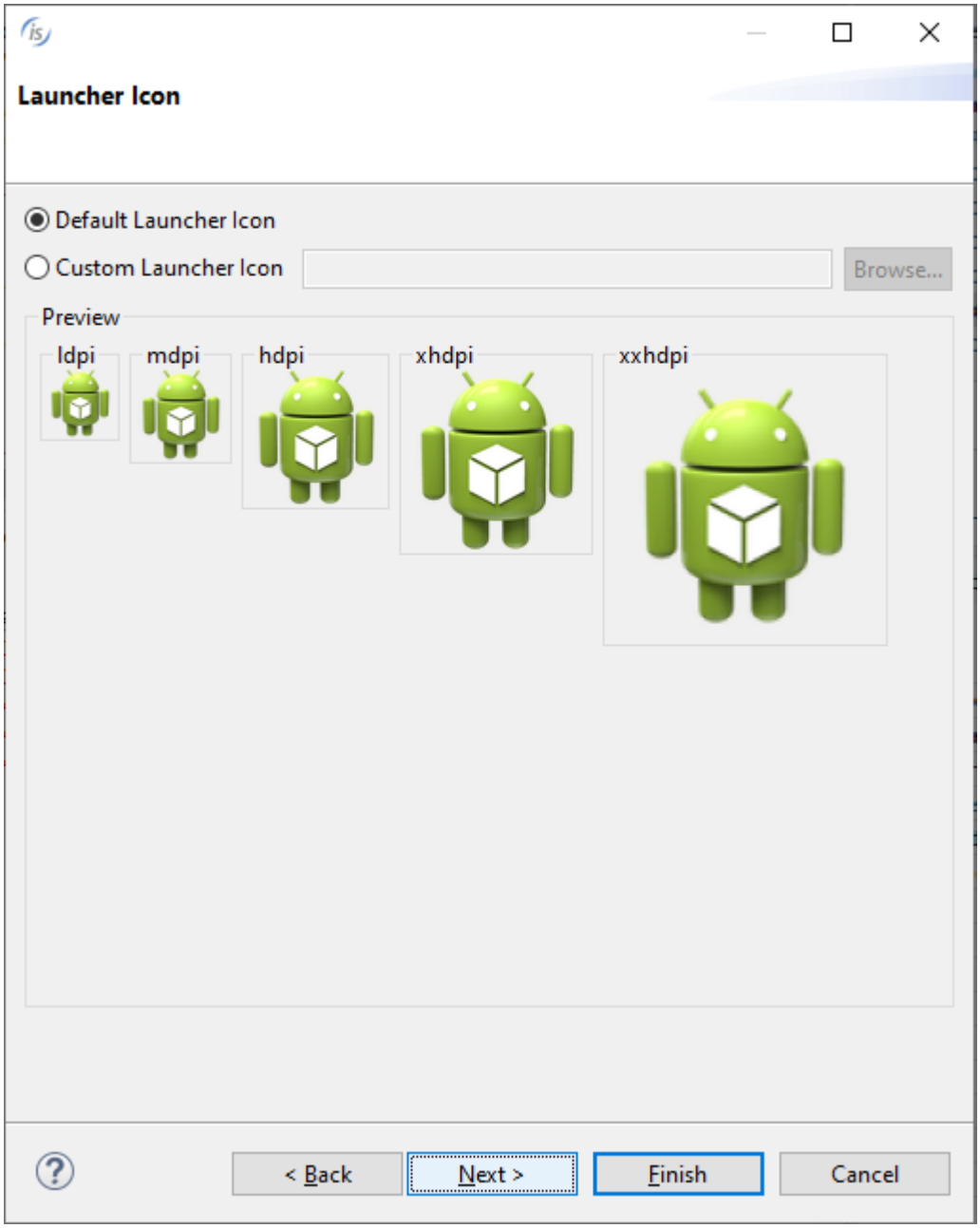
The "Enable WebView Remote Debugging" option enables the ability to attach the app with a Chrome browser in order to debug the HTML/JS frontend as described here.

6. You can click *Finish* or go ahead with the following optional steps.

7. Click *Next* to customize permissions. By default, only WRITE_EXTERNAL_STORAGE is active.

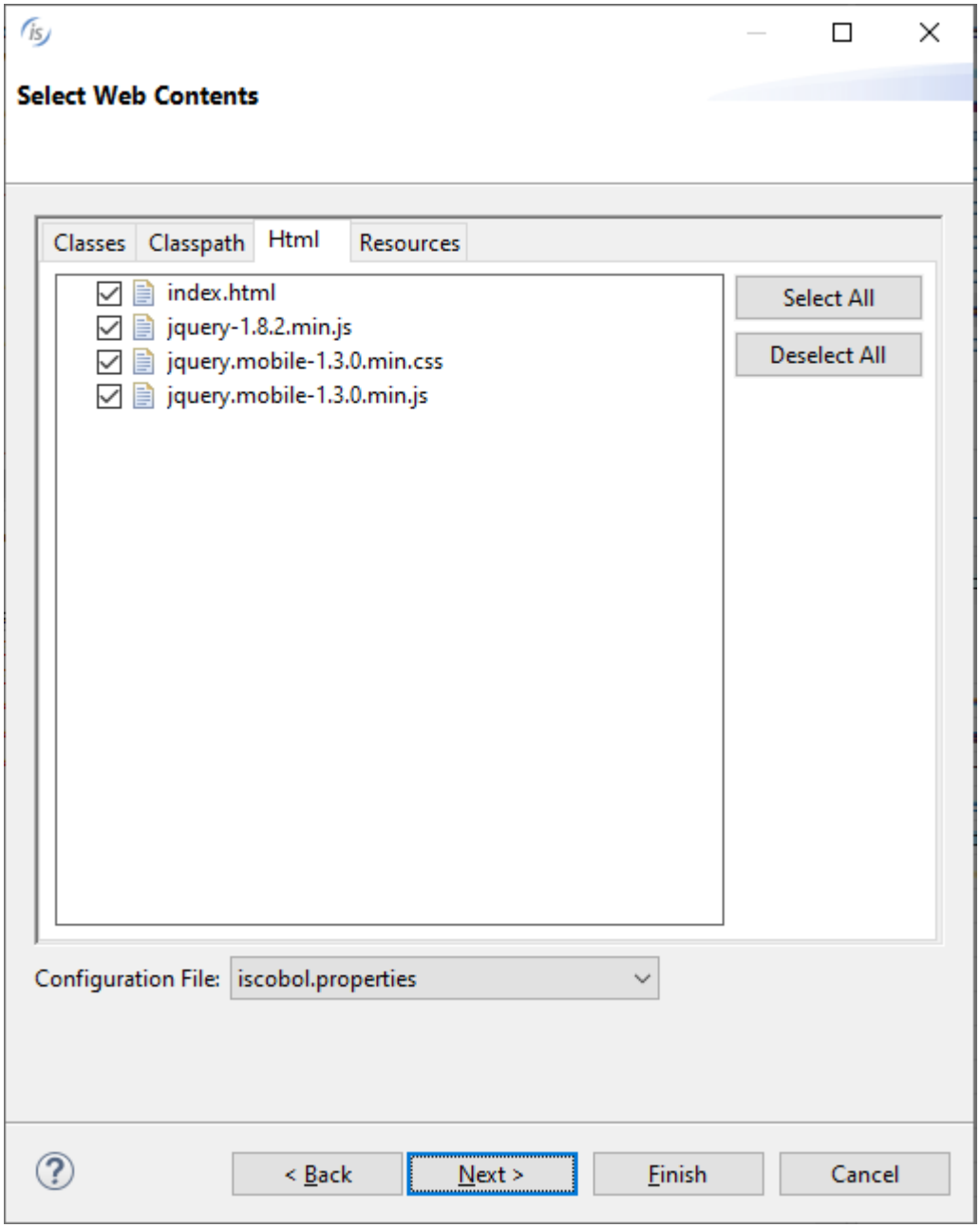


8. Click *Next* to configure the application icon



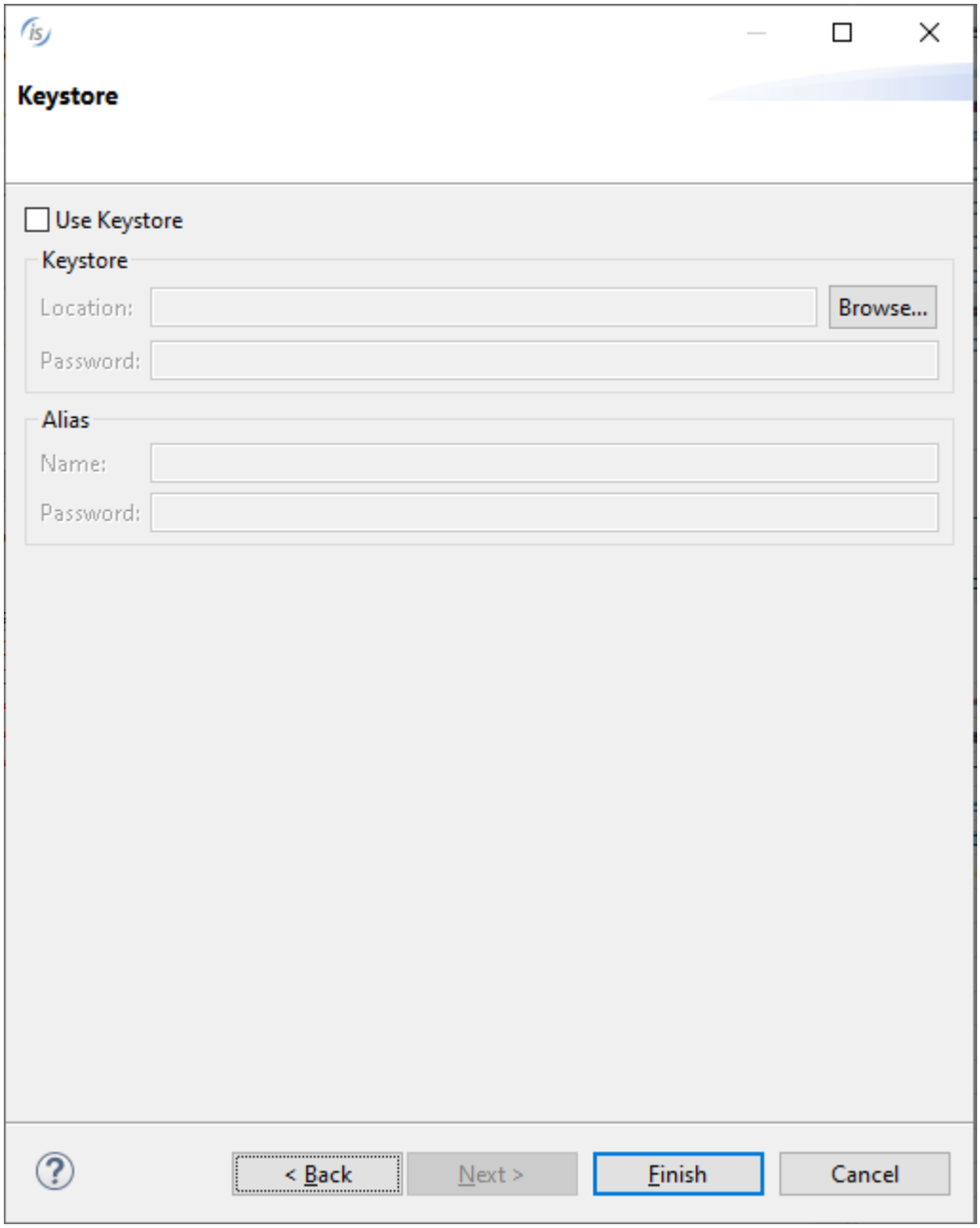

9. Click *Next* to to exclude some of the project items (e.g. a program or a css file). Ensure that these items are

really unused before excluding them, otherwise your app may not work correctly.




10. Click *Next* if you have a valid keystore to provide or you wish to exclude some project files from the app, click

*Next*, otherwise click *Finish*.




**Note -** If no keystore is provided, the IDE will include a default one. The default keystore is suitable for testing purposes. Before publishing your app in the Marketplace, you should apply a valid keystore to it.

### Without isCOBOL IDE

1. Create an empty folder to host project files (e.g. “C:\android_test”)

2. Run the command:

```
android create project --target 1 --name simple_ANDROID --path c:\android_test --
activity MainActivity --package  com.example.simple_android
```

You should obtain the following output

```
Created directory C:\android_test\src\com\example\simple_android
Added file c:\android_test\src\com\example\simple_android\MainActivity.java
Created directory C:\android_test\res
Created directory C:\android_test\bin
Created directory C:\android_test\libs
Created directory C:\android_test\res\values
Added file c:\android_test\res\values\strings.xml
Created directory C:\android_test\res\layout
Added file c:\android_test\res\layout\main.xml
Created directory C:\android_test\res\drawable-xhdpi
Created directory C:\android_test\res\drawable-hdpi
Created directory C:\android_test\res\drawable-mdpi
Created directory C:\android_test\res\drawable-ldpi
Added file c:\android_test\AndroidManifest.xml
Added file c:\android_test\build.xml
Added file c:\android_test\proguard-project.txt
```

3. In the directory you created at step 1, edit the file *AndoirdManifest.xml* and replace the current content with the following:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="https://schemas.android.com/apk/res/android"
    package="com.example.simple_android"
    android:versionCode="1"
    android:versionName="1.0"
    android:installLocation="auto" >
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name="com.example.simple_android.MainActivity"
            android:label="@string/app_name"
            android:configChanges="keyboardHidden|orientation|screenSize" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The manifest file describes the fundamental characteristics of the app and defines each of its components. Our main goal is to obtain write permissions on the device in order to store the html files for the UI and let the COBOL program create files if necessary.

4. In the same directory, edit the file *project.properties* to specify the Android target version. The target must be installed in your SDK. You can run the command

```
android list targets
```

to obtain the list of available targets.

5. Switch to the *res\layout* sub folder, rename the file *main.xml* to *activity_main.xml* and replace the current content with the Content of activity_main.xml
6. Switch to the *src\com\example\simple_android* sub folder and edit the file *MainActivity.java* replacing the current content with the Content of MainActivity.java.
7. Add *ismobile.jar* (taken from C:\Veryant\isCOBOL_SDK2023R1\mobile\lib) and *cobol.jar* (previously produced) to the *libs* sub folder.
8. Create a folder named "raw" under the *res\layout* sub folder and put *html.zip* (previously created) into that folder.
9. (optional) Edit the file *strings.xml* under the *res\values* subfolder and provide a custom name for your app. The current name is "MainActivity", you might call it "simple_ANDROID" to make it match with the project name or you can use whatever name you prefer.
10. From the root directory of your project (e.g. C:\android_test) run the command:

```
ant debug
```

The above command builds the apk file of your app under the project's *bin* directory. The apk is named "simple_ANDROID-debug.apk". You can specify a custom name by adding the option -*Dout.final.file=YourCustomName.apk* at the end of the command line. The apk file can also be renamed later using operating system commands.

### Content of activity_main.xml

```
<WebView  xmlns:android="https://schemas.android.com/apk/res/android"
    android:id="@+id/webView1"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

The WebView control allows to render the HTML user interface of our application. This is the only control we need.

Content of MainActivity.java

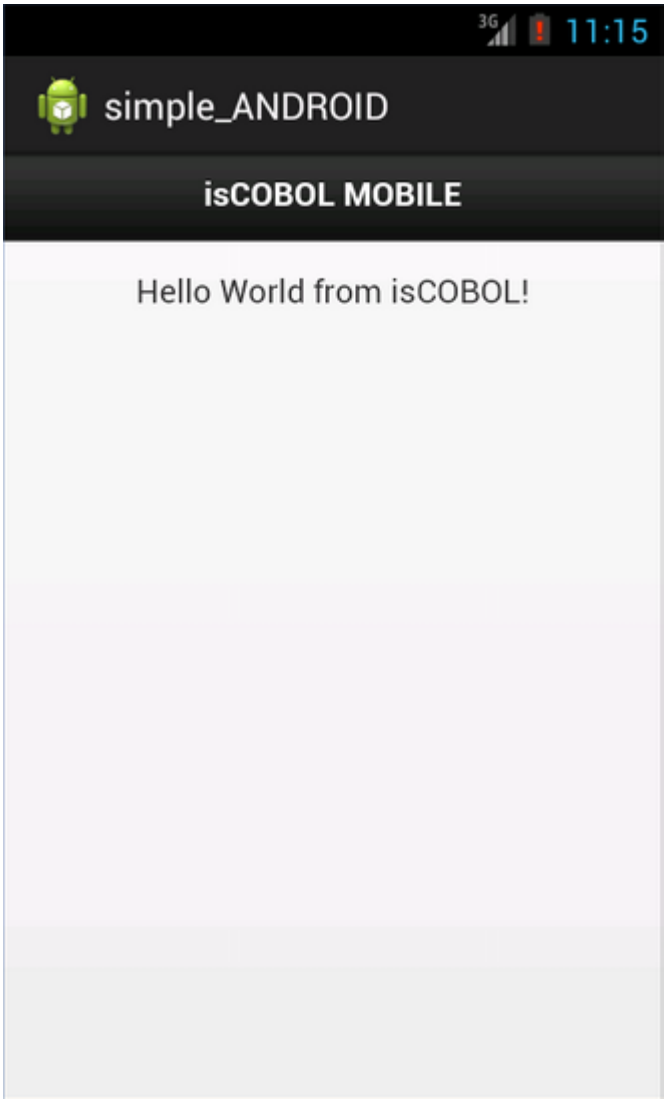
```
package com.example.simple_android;

import java.io.PrintWriter;
import java.io.StringWriter;

import javax.crypto.Cipher;
import javax.crypto.spec.SecretKeySpec;

import android.os.Bundle;
import android.webkit.WebView;

import com.iscobol.iscobol4android.IsCobolMainActivity;

public class MainActivity extends IsCobolMainActivity {

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);

      final WebView webView = (WebView) findViewById(R.id.webView1);
      init (webView, R.raw.html);
   }
}
```

## Testing the Android App

Once the apk has been generated, you can test it either in an Android emulator or in a real physical Android device. For better performance and a more accurate outcome, testing on a physical device is suggested.

In order to test it in a physical device, copy the apk to the device through USB or network file transfer features and install it.

Find your app on the Android screen and touch it to start it:



# Chapter 5

# Troubleshooting

This chapter explain how to diagnose errors that may appear while working with isCOBOL Mobile.

## How to debug the HTML interface

Problems in the HTML and Javascript code can be monitored and debugged through web-browsers' advanced developers' features. Consult your web-browser documentation for details about the availability and the usage of such features.

## Issues during the Export to an Android application

When the IDE exports a project to an Android application it prints the outcome in the *Console* view. Some errors may be available in the *Error log* view instead.