PART I
Getting Started

➤ CHAPTER 1: An Introduction to Tizen
➤ CHAPTER 2: Tizen Application Packages
➤ CHAPTER 3: Tizen Development Tools
An Introduction to Tizen

WHAT’S IN THIS CHAPTER?

➤ Discovering Tizen
➤ Installing the SDK
➤ Creating your first Tizen web application
➤ Creating your first Tizen native application

WROX.COM CODE DOWNLOADS FOR THIS CHAPTER

The wrox.com code downloads for this chapter are found at www.wrox.com/go/professionaltizen on the Download Code tab. After decompressing the downloaded Zip file you will have HelloWorldNative and HelloWorldWeb directories with the finished application.

This chapter provides a brief introduction to Tizen, including the project’s history, the way it’s organized, and what makes it unique. Although it provides some good reasons to develop for Tizen, it’s likely you already know the potential of the platform, so the chapter quickly turns into a practical introduction.

You will learn how to set up the Tizen development environment, and get started with both web and native application development. You’ll then create the Hello World Tizen application using both the web and native application frameworks, and see how easy it is to use the UI Builder and get started with writing code.

DISCOVERING TIZEN

The Linux Foundation announced the Tizen project in September 2011 and the Tizen Association was formed in 2012 to drive industry engagement and support for the project. Tizen as a platform hasn’t emerged from nowhere. It’s an evolution of the previous Linux-based
platforms: MeeGo and LiMo. The Tizen platform embraces the latest standards, such as HTML5 and web-centric technologies, to provide a robust unified experience across multiple devices.

At the time of writing, the latest version of Tizen is 2.2.1. Compared to the initial release, it has a lot of new features and APIs, and the IDE and related tools have also been enhanced. In this chapter, you’ll discover how the platform has evolved, look at some of Tizen’s most important features, and see why you need to learn about Tizen programming.

We’ll start with the two governance bodies behind the Tizen Platform: the Tizen Association and the Tizen Project.

The Tizen Association and Tizen Project

Tizen is an open-source, standards-based, cross-architecture software platform designed for multiple device categories such as smartphones, tablets, in-vehicle infotainment systems, and smart TVs. Tizen platform development is led by two bodies: the Tizen Association and the Technical Steering Group (TSG). The Tizen Association and the TSG are complementary to each other in terms of the marketing and technical directions of Tizen.

The Tizen Association is a mobile operator-led, industry consortium chartered with actively developing the ecosystem around Tizen, which includes marketing presence, gathering of requirements, identification and facilitation of service models, and overall industry marketing and brand awareness. You can find out more at [www.tizenassociation.org](http://www.tizenassociation.org).

The Tizen Project resides within the Linux Foundation and is governed by the TSG, which is the primary decision-making body for the open-source project and whose focus is the development and delivery of the platform itself. Currently the TSG has two members: Intel and Samsung. The Tizen brand is actually owned by the Linux Foundation, not by any single company.

The Tizen Ecosystem

As an open-source software platform, Tizen provides many opportunities to application developers, platform developers, and original equipment manufacturers (OEMs). Together, these groups comprise the key components of the Tizen ecosystem.

The application developers are important to any mobile ecosystem because they are the ones who create the best applications and user experience for the end user. If you think of ecosystems as the battlefields of the platform war, application developers are in the front line. System integrators are those who squeeze every last ounce of performance from mobile chipsets and GPUs, and write the device drivers to work with sensors and other hardware. An open-source platform like Tizen offers huge opportunities to these developers.

The operators and OEMs might be described as the commanders and generals of the platform war. Operators choose to add Tizen-enabled devices to their networks and deploy their own services on those devices, for functions such as billing and promotions. OEMs are those who make the products that use Tizen, and decide what features will appear on these devices.

Another key element of the Tizen ecosystem is the Tizen Store, which was announced and demonstrated in February 2013 at the Tizen 2.0 release event. Currently the Tizen Store is still under development and not open to the public, but Tizen application developers may submit their applications for the certification process via the seller website.
The mobile platform war is actually one of the ecosystems. The platform that can create a bigger and healthier ecosystem is the one more likely to win the leading position in the market, and to gather more application developers around it. That brings us to the main question: Why should you choose to develop for the Tizen platform?

REASONS FOR PROGRAMMING IN TIZEN

Tizen is still a new platform with a lot of potential. Compared to other mobile platforms, it is the only platform that has all these characteristics:

➤ Tizen is based on standards.
➤ Tizen is open.
➤ Tizen is heterogeneous.
➤ Tizen has industry support.

Tizen Is Based on Standards

Tizen’s primary focus for application development is HTML5, a set of standards that is well supported by the industry. It is rapidly merging as the preferred development environment for mobile developers. According to the results of HTML5 compatibility test published on the HTML5 Test website (http://www.html5test.com), the Tizen browser tops the HTML5 benchmarking test amongst all other mobile platforms.

Not only is the Tizen browser compatible with more HTML5 features, but it also performs better. Tizen web apps achieve a frame rate of more than 60 frames per second (fps) when displaying graphics using WebGL technologies. This makes Tizen a capable environment for developing 3D and animations applications even in HTML5.

Tizen Is Open

Another characteristic of Tizen is its openness. This refers not only to the platform’s source code, but also the governance model.

During the Tizen 3.0 announcement at the Tizen Developer Summit in South Korea in 2013, it was stated that beginning with version 3.0, Tizen will embrace a governance model that is even more open than the current model. The Tizen source code is currently uploaded to the tizen.org git repository only at particular milestones, which means that developers can't access the nightly build of Tizen source code for features as they are being developed. Beginning with Tizen 3.0, all source code will be available on the tizen.org git repository as it is being developed. Developers will be able to download the latest source code from tizen.org, build it, and try out the latest features. Any developer will be able to contribute to the project, and recommendations for and contributions of new Tizen features will be publicly available on the JIRA server, the issue-tracking system used by Tizen Project. Developers will also be able to vote and influence the Tizen roadmap.

All this makes Tizen a truly open platform, especially compared to the closed governance model of Android.
Tizen Is Heterogeneous

Tizen was designed with multiple device profiles in mind. Currently two profiles are under active development: for mobile devices and in vehicle infotainment (IVI) systems. In the future you’re likely to find Tizen in PCs, printers, TV set top boxes, and even cameras. In fact, the first official Tizen-enabled device is a camera, the Samsung NX300M. At Mobile World Congress in 2014, Samsung also announced Gear 2, a Tizen-powered smart watch.

From Tizen 3.0 onwards, there will be one codebase that can be configured to support multiple profiles and architectures. The potential of the Tizen platform is considerably bigger than other mobile-only platforms. For application developers, this offers the enticing prospect of writing one application not only for many devices of the same type, but also for a whole range of different types of devices.

NOTE When we use the word “Tizen” in this book, we are referring to the mobile profile of the Tizen platform.

Tizen Has Industry Support

Tizen is backed by a large group of industry leaders that form the Tizen Association board. The board includes many operators and device manufacturers and at the time of writing consists of Samsung, Intel, Fujitsu, Huawei, KT Corporation, LG, NTT DOCOMO, Orange, SK Telecom, and Vodafone.

The Tizen Association recently launched the Tizen Association Partner Program, which consists of 36 companies in different areas across a diverse set of connected device manufacturers, operators, application developers, and software vendors. In February 2014, the Tizen Association announced that an additional 15 partners had joined the Tizen Association Partner Program, including Baidu, ZTE, and SoftBank Mobile.

INSTALLING THE TIZEN SDK

Now that you’ve heard the reasons why you should learn Tizen programming, it’s time to get practical and start your development by downloading and installing the Tizen SDK.

The examples in this book make use of version 2.2.1 of the Tizen SDK, which was released in November 2013. You can download the latest version of the SDK from the Tizen developer website: https://developer.tizen.org/downloads/tizen-sdk.

Prerequisites

The Tizen developer website (https://developer.tizen.org/downloads/sdk/installing-sdk/prerequisites-tizen-sdk) lists all the prerequisites for running the Tizen SDK on different operating systems. Here are the system requirements for the most common platforms — Ubuntu 12.04 or 12.10 (32- or 64-bit), Microsoft Windows XP (32-bit) Service Pack 2 or later, Microsoft Windows 7 (32- or 64-bit), Apple Mac OS X 10.7 Lion (64-bit), or Apple Mac OS X 10.8 Mountain Lion (64-bit):
➤ At least a dual-core 2 GHz CPU
➤ At least 2GB of RAM
➤ At least 3GB of free disk space
➤ A local admin account

The Java Runtime Environment (JRE) version 6 or later is required for all operating systems.

**NOTE** Please make sure you install the JRE instead of the Java Development Kit (JDK). The links provided from the developer site do not include a direct JRE download link, so you will need to choose the correct version.

### Installing the SDK

The Tizen SDK consists of two parts: the Tizen IDE and the Tizen SDK image. The Tizen IDE is based on the Eclipse IDE but integrates additional Tizen-specific development tools. The Tizen SDK image can be downloaded from the network during the installation or downloaded separately as a Zip package.

The Tizen SDK supports Windows, Ubuntu, and Mac OS X operating systems. The installation process is fairly simple and is well documented on the Tizen developer website (https://developer.tizen.org).

**NOTE** If you install the SDK on Mac OS X 10.7.5 or later and an error dialog appears (see Figure 1-1), when you launch the Install Manager, you may need to change your security settings temporarily to prevent the OS X Gatekeeper feature from blocking the installation.

Choose Apple Menu ➪ System Preferences, open the Security preference pane and click the General tab. Under the heading “Allow applications downloaded from,” choose the Anywhere radio button. You should now be able to launch the Install Manager.

Once you’ve finished installing the Tizen SDK, restore your settings to the previous state.

![Figure 1-1](image-url)
To install the Tizen SDK, you need to download the latest Install Manager from https://developer.tizen.org/downloads/tizen-sdk. Once you have downloaded the Install Manager, you can choose to install the Tizen SDK from the network or local file.

Once the Install Manager is running, you can choose to install the SDK from the network by clicking the Next button, agree to the terms and conditions, and select the components you want to install. This is the default option when installing the SDK.

You can also install the SDK from the local SDK image or select the previous version of the SDK by clicking the Advanced button (see Figure 1-2).

To install an SDK version other than the latest version from the network, click the Package server radio button, shown in Figure 1-3. This gives you the option to choose which package server you want to use, and which version of the SDK image to install.

To install the SDK from the local SDK image, click the SDK image radio button and choose the image file that you downloaded as a Zip file.

**NOTE** If you use Safari on Mac OS X, ensure that the Preferences ➤ General ➤ Open “safe” files after downloading option is unchecked. Otherwise Safari will extract the SDK image file automatically and the Install Manager won’t be able to find the Zip file.

Once you have successfully installed the Tizen SDK, you are ready to launch the Tizen IDE.
Creating the Tizen Hello World Web Application

It’s traditional for a getting-started tutorial to begin with the Hello World example application, and this chapter is no exception. The Tizen Hello World application demonstrates quite a few features of the SDK and gives you a good start in Tizen programming.

The Hello World example you will create in this section does three simple things:

➤ Presents an OK button on the screen when it is launched
➤ Prints “Hello World” to the screen when the OK button is clicked
➤ Exits when the user clicks the hardware Back key

The simplest way to create a Hello World project is to use the New Project wizard. The New Project wizard provides several templates to use. There are four predefined template types:

➤ Basic — A blank application with minimum files and resources to run a Tizen application
➤ Tizen Web UI Framework — A set of templates with all the necessary files for creating a web application using the Tizen UI framework
➤ jQuery Mobile — A set of templates containing all the necessary files and libraries for creating web applications using the jQuery Mobile framework
➤ Tizen Web UI Builder — A set of templates containing all files to create a web application using the integrated Tizen Web UI Builder

To launch the New Project wizard and choose a template, follow these steps:
1. In the IDE, choose File ➪ New ➪ Tizen Web Project. The New Tizen Web Project dialog will be displayed as shown in Figure 1-4.

![Figure 1-4](image)

2. Choose the template that suits your needs. For this helloworld project, select Single Page Application from the Tizen Web UI Framework template options.

3. In the Project name field, give your project a name. The example application uses “helloworld.”

4. Click the Finish button.

After you finish the preceding steps, the New Project wizard creates the basic web application structure for you, and this will be the foundation of your Tizen web application project.

The New Project wizard generates folders and files for your Hello World project, as shown in Figure 1-5.

Don’t worry too much about the details of each of the files and folders for now; they are explained in more detail in Chapters 2–4.

![Figure 1-5](image)
To display the “Hello World” text and an OK button, make the following changes to the index.html file as highlighted in the following code listing:

1. Change the auto-generated text between the <div data-role="content"> tags to <p id="labelOk"></p>.

2. Add a new <div> with "data-role=button".

INDEX.HTML

```html
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <meta name="description" content="A Tizen Web UI FW single-page template generated by Tizen Web IDE"/>
  <meta name="viewport" content="width=device-width, user-scalable=no"/>

  <link rel="stylesheet" href="tizen-web-ui-fw/latest/themes/tizen-white/tizen-web-ui-fw-theme.css" name="tizen-theme"/>
  <title>Tizen Web IDE - Template - Tizen - Tizen Web UI Framework - Single-Page</title>

  <!--NOTE:
  jquery.js and web-ui-fw.js must be included.
  DO NOT REMOVE below code!-->
  <script src="tizen-web-ui-fw/latest/js/jquery.js"></script>
  <script src="tizen-web-ui-fw/latest/js/tizen-web-ui-fw-libs.js"></script>
  <script src="tizen-web-ui-fw/latest/js/tizen-web-ui-fw.js" data-framework-theme="tizen-white"></script>

  <!--NOTE:
  Additional scripts and css files are to be placed here.
  You can use jQuery namespace($) and all functionalities in jQuery in your script. For example:

  <script src="main.js"></script>
  <link rel="stylesheet" href="my.css">

  -->
  <script type="text/javascript" src="./js/main.js"></script>
  <link rel="stylesheet" type="text/css" href="./css/style.css"/>
</head>

<body>
  <div data-role="page">
    <div data-role="header" data-position="fixed">
      <h1>Single-page application</h1>
    </div><!-- /header -->

    <div data-role="content">
      <p id="labelOk"></p>                            //(1)
      <div data-role="button" id="buttonOk">OK</div>         //(2)
    </div><!-- /content -->

    <div data-role="footer" data-position="fixed">
</div>
```

```
The main.js JavaScript file in the /js folder is generated by the New Project wizard and contains the code for interacting with the user. To implement the button click and hardware Back key event, use the jQuery bind() method to add the event callback to the OK button with the ID buttonOk that you added in the preceding index.html file. The hardware Back key event callback backEvent() is implemented automatically by the New Project wizard.

You’ve now completed your first Hello World Tizen web application. The completed main.js JavaScript file is shown below. Run the application by choosing Run ▶ Run As ▶ Tizen Web Simulator Application in the Tizen IDE.

**MAIN.JS**

```javascript
var backEventListener = null;

var unregister = function() {
    if ( backEventListener !== null ) {
        document.removeEventListener( 'tizenhwkey', backEventListener );
        backEventListener = null;
        window.tizen.application.getCurrentApplication().exit();
    }
}

//Initialise function
var init = function () {
    // register once
    if ( backEventListener !== null ) {
        return;
    }

    // TODO:: Do your initialisation job
    console.log("init() called");

    var backEvent = function(e) {
        if ( e.keyName == "back" ) {
            try {
                if ( $.mobile.urlHistory.activeIndex <= 0 ) {
                    // if first page, terminate app
                    unregister();
                } else {
                    // move previous page
                    $.mobile.urlHistory.activeIndex -= 1;
                    $.mobile.urlHistory.clearForward();
                    window.history.back();
                }
            } catch( ex ) {
                unregister();
            }
        }
    }
```
// add eventListener for tizenhwkey (Back Button)
document.addEventListener( 'tizenhwkey', backEvent);
backEventListener = backEvent;
$('#buttonOk').bind('click', function (){/
    document.getElementById("labelOk").innerHTML="Hello World";
});

$(document).bind( 'pageinit', init);
$(document).unload( unregister );

Figure 1-6 shows the result when the Hello World project runs in the Simulator.

NOTE You can find much more information about the Web Simulator in Chapter 3, “Tizen Development Tools.”

CREATING THE HELLO WORLD TIZEN NATIVE APPLICATION

Beginning with the Tizen SDK 2.0, it’s possible to develop native applications using the Tizen native application framework. In this section, you’ll learn how to create the same Hello World example you built in the previous section using the native application framework.

The easiest way is still to use the New Project wizard. This time you’ll use the Tizen Native Project wizard, shown in Figure 1-7.

The Tizen Native Project wizard provides different types of templates to choose from to create native applications:

➤ Empty Application — An empty project with minimum files.
➤ Form-based Application — A set of templates to create a simple project based on a form.
➤ IME Application — A template is suitable for creating an IME application project based on a form. This requires a partner-level privilege to run the application.
> **Library Application** — A set of templates to create different types of libraries that can be used by other applications.

> **OpenGL Application** — A set of templates to create simple projects based on GlPlayer or GlRender. The view of this project can be based on the Tizen native Form or Frame.

> **Service Application** — A set of templates to create simple service applications that don’t have a UI and always run in the background (e.g., DynamicBox provider). See Chapter 2, “Tizen Application Packages,” for more details about the DynamicBox.

> **Tab-based Application** — A set of templates to create tab-based applications with tab-switching functions.

> **Theme Application** — A set of templates to create themes that can be used for the home screen or lock screen.

To create a simple Hello World native application, follow these steps:

1. In the IDE, choose File ➪ New ➪ Tizen Native Project. The New Project window will be displayed (refer to Figure 1-7).

2. Choose the template that suits your needs. For this helloworld project example, select Form-based Application template ➪ Without SceneManager, as shown in Figure 1-8.

3. Add a name for your project. In this example it is named helloworld2.

4. Click the Finish button.

---

**FIGURE 1-8**

Create a Native Application Project
Create a Tizen native project of the selected type.

Form-based Application 1.0

This template is suitable for creating a simple project based on a

![New Tizen Native Project](image)

Description

Tizen Native
Empty Application
Library Application
OpenGL Application
Service Application
Tab-based Application
Theme Application

Project name: helloworld2

Location: C:\Users\[User]\workspace\helloworld2

Finish ➤ Next ➤ Cancel ➤ Back
Once the preceding steps are finished, the New Project wizard will generate necessary folders and files you will need to start your native application.

**NOTE** You can find more details about the contents of these files and folders in Chapter 2, while native applications are covered in detail from Chapter 10 onwards.

You need to pay attention to two files within the Hello World application: the IDL_FORM.xml file in the /res/screen-size-normal folder and the HelloWorld2Form.cpp file in the /src folder.

First you need to make the changes to the application UI, which requires editing the IDL_FORM.xml file. Double-click the IDL_FORM.xml file to open the native UI Builder which is integrated into the Tizen IDE.

To add the “Hello World” text and the OK button as you did in the web application, follow these steps:

1. Select the Text Box from the Toolbox panel, and drag it to the form. Highlight the Text field from the Properties panel of the Text Box, and type **Hello World** as shown as Figure 1-9.

2. The OK button is automatically generated by the template you selected, so you don’t need to do anything with the button.

![Figure 1-9](image-url)
The next step is to make corresponding changes in the source code of the form, which is the HelloWorld2Form.cpp file. In fact, the New Project wizard does most of the work here. It creates the hardware Back key event handler called OnFormBackRequested(), and it implements the callback for the button click event OnActionPerformed() in the following code listing:

```cpp
HELLOWORLD2FORM.CPP

#include "HelloWorld2Form.h"
#include "AppResourceId.h"

using namespace Tizen::Base;
using namespace Tizen::App;
using namespace Tizen::Ui;
using namespace Tizen::Ui::Controls;

HelloWorld2Form::HelloWorld2Form(void)
{
}

HelloWorld2Form::~HelloWorld2Form(void)
{
}

bool
HelloWorld2Form::Initialize(void)
{
    result r = Construct(IDL_FORM);
    TryReturn(r == E_SUCCESS, false, "Failed to construct form");

    return true;
}

result
HelloWorld2Form::OnInitializing(void)
{
    result r = E_SUCCESS;

    // TODO: Add your initialisation code here

    // Setup back event listener
    SetFormBackEventListener(this);

    // Get a button via resource ID
    Tizen::Ui::Controls::Button* pButtonOk =
        static_cast< Button* >(GetControl(IDC_BUTTON_OK));
    if (pButtonOk != null)
    {
        pButtonOk->SetActionId(IDA_BUTTON_OK);
        pButtonOk->AddActionEventListener(*this);
    }

    return r;
}
```
result
HelloWorld2Form::OnTerminating(void)
{
    result r = E_SUCCESS;

    // TODO: Add your termination code here

    return r;
}

void
HelloWorld2Form::OnActionPerformed(const Tizen::Ui::Control& source, int actionId)//(1)
{
    switch(actionId)
    {
        case IDA_BUTTON_OK:
            AppLog("OK Button is clicked!
            break;
        default:
            break;
    }
}

void
HelloWorld2Form::OnFormBackRequested(Tizen::Ui::Controls::Form& source)//(2)
{
    UiApp* pApp = UiApp::GetInstance();
    AppAssert(pApp);
    pApp->Terminate();
}

(1) is the callback method for handling actions triggered by Controls, such as the OK button.
(2) is the callback method for handling the hardware Back key.

Again, don’t worry too much about how to use the UI Builder, as it’s discussed in more detail in Chapter 3.

SIGNING THE HELLO WORLD APPLICATION

There is one more step before you can run your Hello World sample application on the Emulator or device: signing and verifying your application. This step is required for both web and native applications.

Register the certificate in your security profiles by opening the Security Profiles window in the Tizen IDE: Select Windows ➤ Preferences ➤ Tizen SDK ➤ Security Profiles. Figure 1-10 shows the Security Profiles window.
To add a signed profile, follow these steps:

1. Click the Add button in the Profiles panel, and provide a name for your profile. In the example it’s called “test”.

2. Click the Generate button in the Author Certificate panel, where you are asked to fill in some information about your author certificate. If you already have an author certificate, you can click the Edit button and choose your certificate from there.

   **NOTE** You can also use the tools provided by the Tizen SDK to generate your author certificate. To run the certificate generator tool, go to the `<TIZEN_SDK_HOME>/tools/certificate-generator` directory and run `certificate-generator.sh` in the shell if you are using Ubuntu or Mac OS X, or run the `certificate-generator.bat` file from the command line if you are using Windows. Alternatively, use the Generate button from the Security Profiles window.

3. At least one distributor certificate is required, and by default this is filled in automatically with the distributor certificate provided by the Tizen SDK. The second distributor certificate is optional and is used only for a specific purpose. More information on signing can be found in Chapter 2.
Once you have the security profile set up, you are ready to launch your Hello World application on the Emulator or the device. For information about how to launch your application on the Emulator, see Chapter 3.

**SUMMARY**

This chapter covered a lot of ground — a Tizen history lesson, the details of how a project is organized, what makes Tizen different, and why you should start creating Tizen applications.

After a brief look at the potential of Tizen development, you moved on to creating a Hello World application — both a web version and one using the native APIs. In this chapter, you launched the web application on the Web Simulator, and you may have taken a sneak peek ahead to Chapter 3 to learn how to launch applications in the Emulator.

The foundation provided in this chapter should enable you to now look at Tizen development in more detail. Chapter 2 delves into the contents of Tizen application packages, for both web and native applications, while Chapter 3 looks at the tools provided with the Tizen IDE.