

TIZEN™

Tizen Web 应用程序开发 - 初学者

版本 1.0 (2014/09/XX)

目录

1. Tizen 简介	6
了解 Tizen	6
Tizen 作为开放源软件平台	6
Tizen 作为 Web 软件平台	6
Tizen 作为行业支持的软件平台	6
Tizen 社区网站	7
2. Tizen 体系结构.....	8
3. Tizen Web API 简介	9
Tizen 设备 API.....	9
4. Web 应用程序开发入门.....	1 2
计划和设计应用程序	1 2
安装 Tizen SDK	1 2
先决条件	1 2
安装 Tizen SDK（在线）	1 3
安装 Tizen SDK（脱机）	1 3
创建应用程序项目	1 3
创建 Web 应用程序项目	1 4
支持的模板和示例	1 6
创建用户模板.....	1 6
设置项目属性.....	1 8
配置应用程序.....	1 9
使用 UI Builder 创建应用程序 UI	2 6
创建 UI Builder 项目	2 7
添加页面	2 7
设计页	2 7
处理针对页面的事件.....	3 0
测试 UI Builder 项目	3 2
生成应用程序.....	3 2
运行和调试应用程序	3 2

在模拟器中运行应用程序	3 3
在仿真器中运行应用程序	3 3
在目标设备上运行应用程序	3 4
调试应用程序	3 4
快速开发支持	3 6
打包应用程序	3 7
查看应用程序包	3 7
本地化应用程序	3 8
5. 使用 Tizen 高级 UI	4 0
Tizen 高级 UI 简介	4 0
开始使用简单的应用程序	4 0
添加页面	4 1
处理针对页面的事件	4 2
可用的 UI 小程序	4 2
处理多个页面、小程序和事件	4 3
6. 使用 Tizen Web API	4 5
日历	4 5
日历 API 的主要功能	4 5
向日历添加事件	4 6
在批处理模式下向日历添加事件	4 7
管理事件	4 8
在批处理模式下管理多个日历事件	4 9
更新重复性日历事件	4 9
接收有关日历更改的通知	5 0
转换日历项	5 1
联系人	5 2
联系人 API 的主要功能	5 2
检索地址簿	5 3
添加联系人	5 4
在批处理模式下添加多个联系人	5 4
管理联系人	5 5
在批处理模式下管理多个联系人	5 6

接收有关联系人更改的通知.....	5 7
导入联系人	5 7
导出联系人	5 8
管理人士	5 9
消息传递	5 9
消息传递 API 的主要功能	6 0
发送消息	6 1
管理消息	6 2
同步电子邮件.....	6 3
接收有关消息存储更改的通知	6 5
多媒体.....	6 6
发现内容	6 6
捕获图像和视频	6 7
播放音频和视频	6 8
流式传输多媒体	7 1
NFC	7 2
NFC API 的主要功能	7 2
管理 NFC 连接性	7 5
检测 NFC 标记和对等设备	7 5
处理 NDEF 消息	7 6
与 NFC 标记交换 NDEF 数据	7 6
与对等设备交换 NDEF 数据	7 7

关于本文档

本文档面向 Tizen 应用程序开发人员，旨在帮助他们理解 Tizen 平台以及开发 Tizen 应用程序。本文档中的信息基于撰写之际的 Tizen 2.3 alpha 版本。

有关更详细信息以及最新参考信息，请访问 [Tizen 开发人员网站](#) 以便获得 Tizen SDK 帮助文档。

本文档中的信息仍会更改。有关最终版本，请参阅正式的 Tizen SDK 文档。

1. Tizen 简介

了解 Tizen

Tizen 是针对多种设备类别的开放源操作系统，例如智能电话、车载信息 (IVI) 设备、可穿戴设备、智能电视、计算机、相机和打印机等。

Tizen 具有以下关键特性：

Tizen 作为开放源软件平台

Tizen 是一种灵活的开放源操作系统，完全是为满足移动和互连设备生态系统的所有各方的需要而构建的，包括设备制造商、移动网络运营商、应用程序开发人员和独立软件供应商。作为开放源软件平台，Tizen 对个体和公司都提供了非常多的机遇。

Tizen 是由社区基于开放源的管控方式开发的，对于想要参与的所有人都开放。开发人员可以在不同的层级对 Tizen 项目作出贡献，例如，作为平台贡献者或应用程序开发人员。

Tizen 作为 Web 软件平台

Tizen 针对应用程序开发的重心是 HTML5，HTML5 目前被认为是针对移动应用程序的首选开发环境。HTML5 允许开发人员构建跨平台应用程序：以便编写一次代码，然后跨多个平台运行该代码。

Tizen 支持大多数正式 HTML5 标准以及某些补充标准和旧标准，并且支持大多数主流 W3C API。

Tizen Web 引擎包含一个呈现引擎和一个 JavaScript 引擎；呈现引擎将标记内容（例如 HTML 和图像文件）与格式信息（例如 CSS）结合在一起并且在屏幕上显示带格式的内容，而 JavaScript 引擎解释并且执行 JavaScript。

Tizen 作为行业支持的软件平台

Tizen 联盟对 Tizen 项目提供强有力的行业支持。成立 Tizen 联盟是为了指引 Tizen 的行业角色，包括收集需求、标识和促进服务模型以及整个行业市场推广和教育。有关详细信息，请参阅 www.tizenassociation.org。

Tizen 项目处于 Linux 基金会下并且由技术督导小组管辖。该技术督导小组是针对 Tizen 项目的主要决策实体，其工作重心放在平台开发和交付以及形成用于垂直支持设备的工作组上。

Tizen 社区网站

主要 Tizen 社区网站是：

- **Tizen 主网站** (www.tizen.org)

Tizen 项目的这个主要网站提供文章和博客。您可为其他 Tizen 网站创建 Tizen 帐户并且获取与项目和活动有关的最新新闻。

- **Tizen 开发人员** (developer.tizen.org)

针对 Tizen 开发人员的这个官方网站提供 Tizen SDK 以及用于开发 Tizen 应用程序的不同资源。您可与论坛中的其他开发人员讨论技术问题，并且查找用于开发您的应用程序的有帮助的示例代码。

- **Tizen 源** (source.tizen.org)

这个 Tizen 开放源项目的官方网站提供 Tizen 源代码以及用于生成源代码的说明。您可以使用 Git 跟踪源代码发布以及检索最新修订。

- **Tizen 项目 JIRA** (bugs.tizen.org/jira)

这个针对 Tizen 项目的官方 JIRA 网站允许您注册、跟踪 bug 以及针对新功能提供建议。

- **Tizen Wiki** (wiki.tizen.org)

该 Tizen wiki 允许您针对 Tizen 项目协调文档。

2. Tizen 体系结构

Tizen 体系结构包括用于开发 Web 应用程序的 Web API 以及用于开发本机应用程序的本机 API。因为 Web 应用程序构成了用于开发 Tizen 的主要方法，所以本文档主要针对 Web API。

下图展示了 Tizen 体系结构。

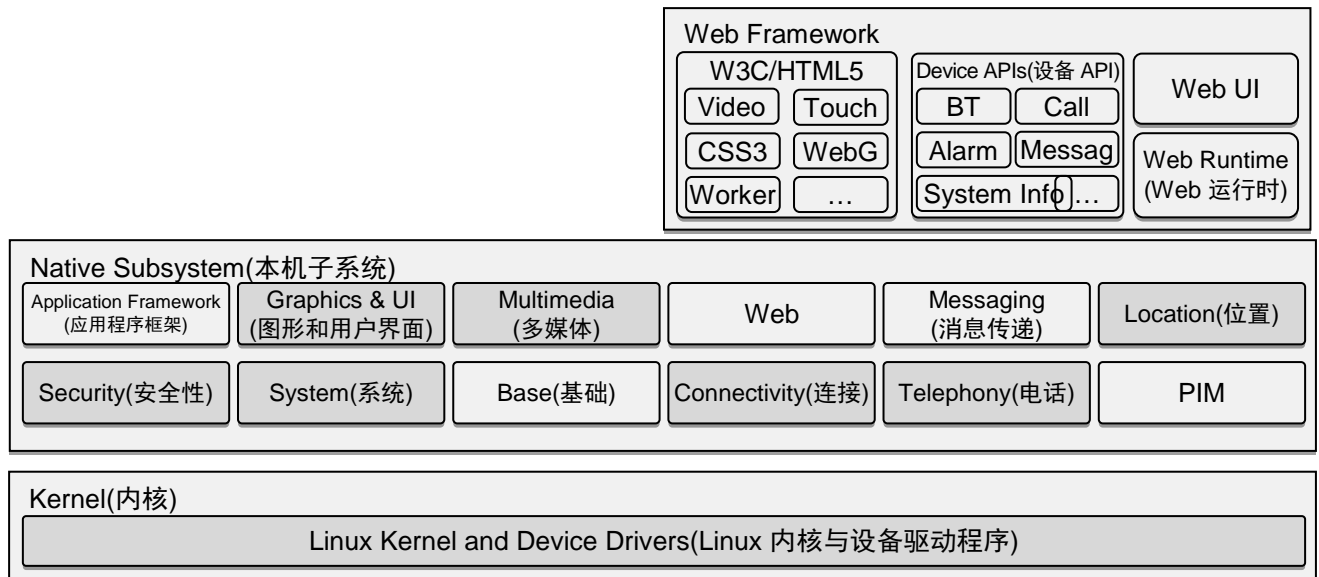


图 1：Tizen 体系结构

Tizen 体系结构由以下子系统构成：

- **Web Framework**

Web Framework 实现最新的 Web 技术，可通过 W3C/HTML5 API 访问。该框架支持 W3C 和其他标准化组定义的大量 HTML5 功能，例如音频、视频、窗体、2D 画布元素、振动、Web 图形库 (WebGL)、CSS3、WebSocket 和 Web 辅助线程。

该框架还包括设备 API，这允许您访问设备功能，例如警报、无线数据交换和消息传递。该框架通过严格的基于规则的安全控制系统（防止对 API 的恶意使用）提供设备功能。

核心 Web 模块提供为低功率设备优化的 Web 框架的完整实现。除了 Web API 之外，该实现还包括 WebKit（这是用于在浏览器中呈现网页的布局引擎）以及用于运行 Web 应用程序的 Web 运行时。

- **本机子系统**

本机子系统提供 Web 框架所要求的功能。该子系统由开放源库以及 Web 框架使用一组附加的本机 API 构成。有关本机 API 的详细信息，请参阅 Tizen SDK 帮助。

- **内核**

内核子系统包含为 Tizen 定制的 Linux 内核以及设备驱动程序。

3. Tizen Web API 简介

您使用 Tizen Web API 开发 Tizen Web 应用程序，Tizen Web API 由 W3C/HTML5 API、补充 Web API 和 Tizen 设备 API 构成。

下图展示了不同类型的 Web API。

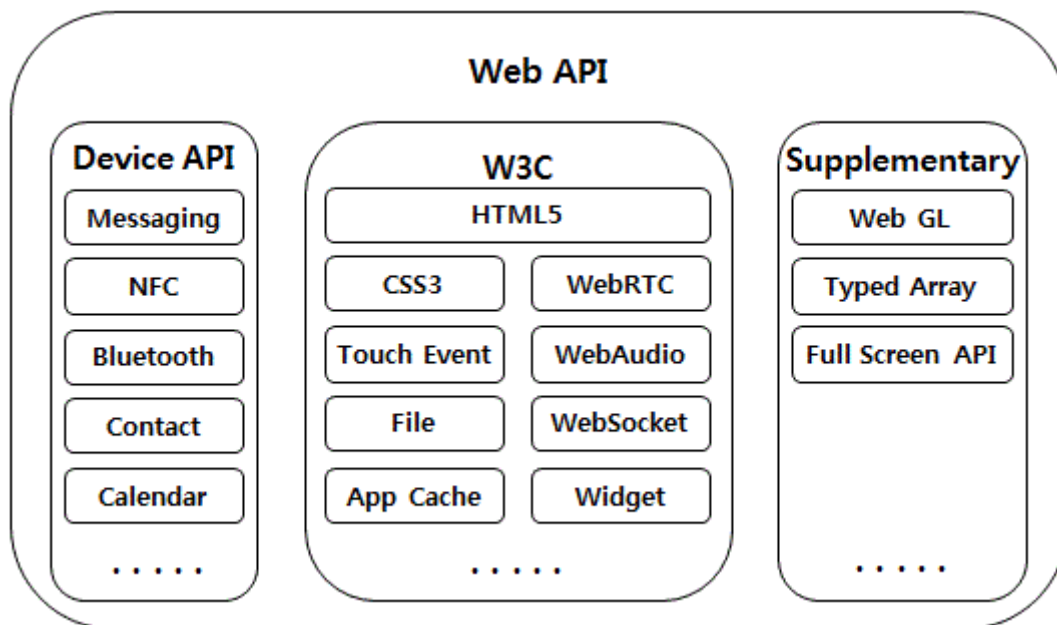


图 2: Tizen Web API

Tizen 设备 API

设备 API 基于 JavaScript 并且提供对设备的平台功能的高级访问。

主要设备 API 是：

- **警报 API**

警报 API 允许您计划系统警报时间。通过警报，您可以运行其他应用程序以便在特定时间执行特定操作。您可以将警报计划为只触发一次，也可以计划为按预定义间隔反复触发。

- **应用程序 API**

通过应用程序 API，您可以检索与在设备上安装或者当前正在设备上运行的应用程序有关的信息。在安装、更新或删除应用程序时您还可以接收通知，并且执行应用程序管理任务，例如启动或退出应用程序。

- **蓝牙 API**

通过蓝牙 API，您可以使用设备的蓝牙功能，例如管理本地蓝牙适配器、将设备与其他蓝牙设备配对，以及通过蓝牙连接交换数据。

- **日历 API**

通过日历 API，您可以在日历中管理事件和任务。日历是事件或任务的集合，取决于日历类型。每个事件或任务都具有一系列属性，例如目的、开始时间和持续时间。

- **联系人 API**

通过联系人 API，您可以管理联系人以及在地址簿中列出的人士。联系人对象始终与特定地址簿相关联。人士对象是与同一个人相关联的一个或多个联系人的聚合。

- **内容 API**

通过内容 API，您可以搜索内容，例如在设备上本地存储的图像、视频和音乐。您还可以执行内容管理任务，例如查看和更新内容属性。

- **下载 API**

通过下载 API，您可以从 Internet 下载文件。还可以监视下载进度和状态。

- **文件系统 API**

通过文件系统 API，您可以管理设备文件系统的文件和文件夹。该文件系统 API 提供对文件系统的非受限部分的访问，这些部分表示为虚拟根位置。这些虚拟根构成了多个位置的集合，它们结合在一起作为您的应用程序可以访问的单个虚拟文件系统。

- **消息传递 API**

通过消息传递 API，您可以将消息传递功能用于 SMS、MMS 和电子邮件通信。HTML5 消息传递进程使用统一资源标识符 (URI)。

- **NFC API**

通过 NFC API，您可以使用近场通信 (NFC) 服务在 NFC 设备（“对等方”）或标记之间交换数据。这些设备可以共享联系人、照片和视频，并且还可以充当智能卡。您可以使用 NFC 设备针对多种行为（例如支付账单或下载优惠券）与 NFC 标记进行通信。

- **通知 API**

通过通知 API，您可以提供与应用程序事件有关的通知。

- **程序包 API**

通过程序包 API，您可以检索有关程序包的详细信息，例如 ID、图标路径和版本。在安装、更新或删除程序包时您还可以接收通知，并且执行程序包管理任务，例如安装或卸载程序包。

- **电源 API**

通过电源 API，您可以访问设备的电力资源。目前，支持屏幕和 CPU 电力资源，使您可以请求特定的电源状态并且控制屏幕的亮度。

- **系统信息 API**

通过系统信息 API，您可以访问设备的系统属性。您可以检索不同的设备和系统详细信息，例如当前电池电量水平、可用存储量以及蜂窝网络连接的状态。

- **系统设置 API**

通过系统设置 API，您可以访问设备的针对家庭的设置，以及锁定屏幕墙纸、来电铃声和电子邮件通知音。

- **时间 API**

通过时间 API，您可以通过检索日期和时间信息使用区域设置特定的日历功能。您还可以更改设备的日期、时间和时区，以及执行与日期和时间相关的计算。

- **Tizen API**

通过 Tizen API，您可以使用常见的 Tizen 功能。Tizen API 包含用于查询方法、一般的成功和错误事件处理程序、一般的错误界面以及用于定义位置信息的简单坐标界面的筛选器和排序模式的筛选器和排序模式。

- **Web 设置 API**

通过 Web 设置 API，您可以设置 Web 视图属性。

有关设备 API 的详细信息，请参阅 Tizen SDK 帮助。

4.Web 应用程序开发入门

Tizen 提供工具以便管理您的应用程序的从构思和设计、到开发和发布、再到最终的应用程序停用的整个生命周期。

开发 Tizen Web 应用程序：

1. [计划 and 设计应用程序](#)
2. [安装 Tizen SDK](#)
3. [创建应用程序项目](#)
4. [创建应用程序 UI](#)
5. [生成应用程序](#)
6. [运行和调试应用程序](#)
7. [打包应用程序](#)
8. [根据需要本地化应用程序](#)

计划 and 设计应用程序

开发一个 Tizen Web 应用程序的第一步是使用您选择的设计工具计划 and 设计应用程序。

在您完成了对您的应用程序的计划和设计后，安装 Tizen SDK 并且创建应用程序项目。

安装 Tizen SDK

Tizen SDK 是用于开发 Tizen Web 和本机应用程序的一组全面的工具。它包含 IDE、仿真器、工具链、示例代码和帮助文档。

在安装该 SDK 时，您可以选择[在线安装](#)或[脱机安装](#)。

先决条件

在开始安装前，请检查您的计算机是否满足以下系统要求：

- 操作系统：
 - Mac OS® X 10.7 Lion (64 位) 或 Mac OS® X 10.8 Mountain Lion (64 位) 或 Mac OS® X 10.9 Mavericks (64 位)
 - Microsoft Windows® 7 (32 位或 64 位) 或 Microsoft Windows® 8 (32 位或 64 位)
 - Ubuntu® 12.04、12.10 或 14.04 (32 位或 64 位)
- 双核 2 GHz CPU
- 2 GB RAM
- 6 GB 可用磁盘空间
- 本地管理员权限

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅[内容许可](#)。

有关先决条件的详细信息，请参阅 <https://developer.tizen.org/downloads/sdk/installing-sdk/prerequisites-tizen-sdk>。

安装 Tizen SDK（在线）

在线安装 Tizen SDK：

1. 下载 [SDK 安装管理器](#)。
2. 运行 SDK 安装管理器。
3. 单击 **Next**（下一步）。
4. 接受条款和条件，然后单击 **Next**（下一步）。
5. 选择您要安装的组件，然后单击 **Next**（下一步）。
6. 选择 SDK 主文件夹，然后单击 **Install**（安装）。

SDK 安装向导将在安装完毕时通知您。您现在可以在 IDE 中创建应用程序项目。

安装 Tizen SDK（脱机）

从映像文件脱机安装 SDK：

1. 下载 [SDK 安装管理器](#)。
2. 下载 [SDK 映像文件](#)。
3. 运行 SDK 安装管理器。
4. 单击 **Advanced**（高级）。
5. 在 **Advanced Configuration**（高级配置）窗口中，选择 **SDK Image**（SDK 映像）。
6. 单击文件夹按钮，浏览到 SDK 映像文件，然后单击 **OK**（确定）。
7. 单击 **Next**（下一步）。
8. 接受条款和条件，然后单击 **Next**（下一步）。
9. 选择您要安装的组件，然后单击 **Next**（下一步）。
10. 选择 SDK 主文件夹，然后单击 **Install**（安装）。

SDK 安装向导将在安装完毕时通知您。您现在可以在 IDE 中创建应用程序项目。

创建应用程序项目

在计划和设计了您的应用程序并且安装了 SDK 后，您现在就可以在 IDE 中创建 [Web 应用程序项目](#)了。

在创建应用程序项目时，请使用适当的项目 [模板或示例](#)。基于该模板或示例，项目向导将自动实现应用程序需要运行的基本功能。您可以从多种模板和示例中进行选择。还可以创建您自己的 [用户模板](#)。

最后，要实现所需功能，请仔细设置 [项目属性](#)和 [配置应用程序](#)。

创建 Web 应用程序项目

为您的应用程序创建 Web 应用程序项目：

1. 在 IDE 中，依次选择 **File > New > Tizen Web Project**（文件 > 新建 > Tizen Web 项目）。
如果看不到想要的项目选项，请确保您在使用 Tizen Web 视角：
 - a. 选择 **Window > Open Perspective > Other**（窗口 > 打开视角 > 其他）。
 - b. 选择 **Tizen Web** 视角。
2. 在 **New Tizen Web Project**（新建 Tizen Web 项目）窗口中，定义项目详细信息：
 - a. 选择应用程序要基于的模板或示例。
 - b. 定义项目的名称。

注意：

Tizen API 名称不能用作项目名称。项目名称的长度必须介于 2 到 49 个字符之间，并且只能包含以下字母数字字符：a-z、A-Z、0-9。

3. 如果启用，则单击 **Next**（下一步）按钮，然后自定义 jQuery Mobile 或 Tizen Web UI Framework 模板设置。
4. 单击 **Finish**（完成）。

新的应用程序项目将显示在 **Project Explorer**（项目资源管理器）视图中，其中包含 `config.xml` 文件中的默认内容以及[若干项目文件夹](#)中的默认内容。

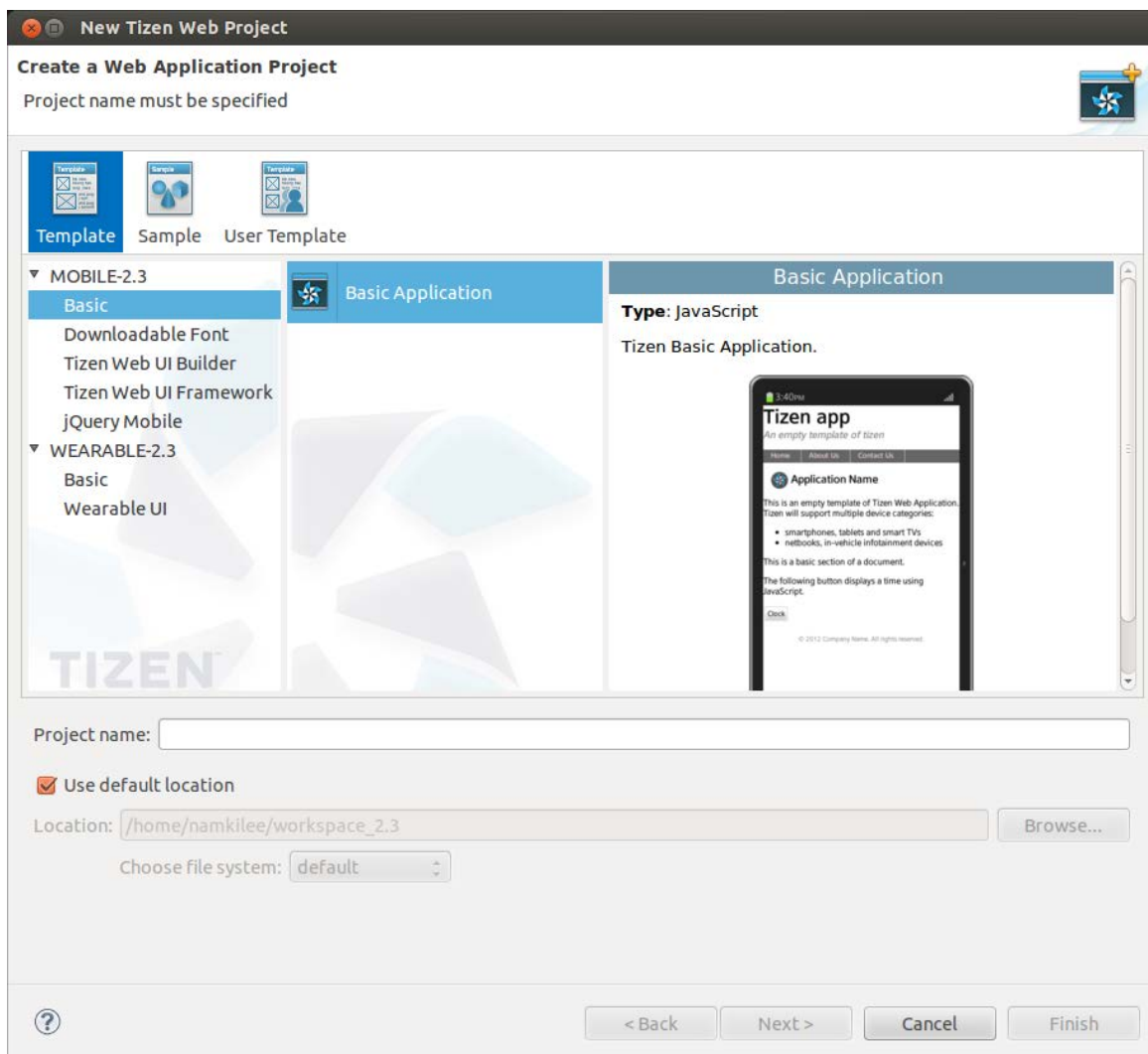


图 3: 创建 Tizen Web 应用程序项目

导入项目

将现有 Tizen 应用程序项目导入到 IDE 中:

1. 在 IDE 中, 依次选择 **File > Import > Tizen > Tizen Web Projects and Widget file**。
2. 单击 **Browse** (浏览), 然后选择包含您的现有项目或小程序文件的根文件夹 (.wgt)。

在选择项目后, 您可以通过选择 **Copy projects into workspace** (将项目复制到工作区) 将其复制到工作区中。

3. 单击 **Finish** (完成)。

支持的模板和示例

您可以基于多种模板和示例创建 Tizen Web 应用程序。除了预定义的模板之外，还可以创建您自己的用户模板。

下表列出了可用于 Tizen Web 应用程序开发的预定义的模板。

表 1: 预定义模板

项目类型	应用程序类型	说明
Basic	空白应用程序	这是空的 Tizen 应用程序模板。
Tizen Web UI Builder	<ul style="list-style-type: none"> 空应用程序 单页应用程序 多页应用程序 导航应用程序 	这些模板基于 Tizen Web UI Builder。
Tizen Web UI Framework	<ul style="list-style-type: none"> 单页应用程序 多页应用程序 主详细信息应用程序 导航应用程序 	这些模板基于 Tizen Web UI Framework。
jQuery Mobile	<ul style="list-style-type: none"> 单页应用程序 多页应用程序 主详细信息应用程序 导航应用程序 	这些模板基于 jQuery Mobile。 您可以基于以下颜色主题更改应用程序页眉、正文和页脚： <ul style="list-style-type: none"> A: 黑色 B: 蓝色 C: 灰色 D: 浅灰色 E: 黄色

创建用户模板

您可以通过复制现有用户模板或通过将现有项目作为用户模板导出，创建您自己的用户模板。然后，您可以使用这些模板创建新项目。

用户模板位于 `<TIZEN_SDK_DATA>/ide/user-templates/web` 文件夹中以及用户定义的文件夹中。`<TIZEN_SDK_DATA>` 文件夹的确切位置取决于操作系统：

- Linux/Mac OS® X: `/home/${USER_NAME}/tizen-sdk-data`
- Windows: `C:\tizen-sdk-data`

用户模板的项目文件夹包含项目相关的文件，这些文件复制到从模板创建的新项目。可以在您的新项目中修改这些文件。

下表描述某一用户模板随附的文件以及项目文件夹内容。

表 2：用户模板文件

文件	内容
description.xml	<p>该文件由以下在项目向导中显示的模板描述信息构成：</p> <ul style="list-style-type: none"> • <SampleName>：项目类型名称。可以使用相同的用户模板创建若干项目类型。 • <SampleVersion>：项目类型版本。 • <Preview>：预览文件名。 • <Description>：说明标题。
tizen-app-template.xml	<p>该文件由以下在项目向导中显示的模板定义信息构成：</p> <ul style="list-style-type: none"> • <template-name>：用户模板名称。 • <widget-type>：小程序类型。默认设置为 TIZEN。 • <description-file-name>：说明文件名称。默认设置为 description.xml。 • icon64 和 icon32 属性：图标图像文件名称。数字后缀指示图标大小（以像素为单位）。

复制用户模板

从现有用户模板创建新用户模板：

1. 复制并且重命名相应的 <TIZEN_SDK_DATA>/ide/user-templates/web 文件夹。
该文件夹包含默认模板布局。
2. 编辑新文件夹中的文件，或者根据需要添加新文件，例如屏幕快照或图标图像以及 CSS 和 JavaScript 文件。

将项目作为用户模板导出

将现有项目作为新用户模板导出：

1. 在 IDE 中，依次选择 **File > Export > Tizen > User Template**（文件 > 导出 > Tizen > 用户模板）。
2. 在 **User Template Export Wizard**（用户模板导出向导）窗口中，定义项目和用户模板详细信息：
 - a. 选择源项目。
 - b. 定义用户模板的名称。
 - c. 单击 **Browse**（浏览），然后选择用户模板的导出位置。

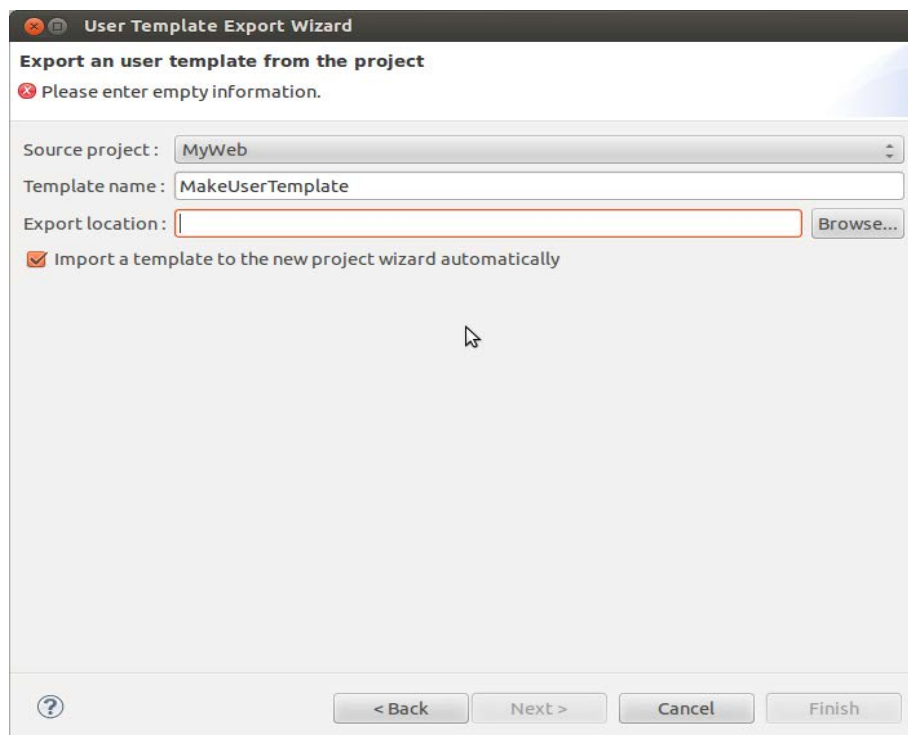


图 4：定义用户模板的导出位置

- d. 要将模板添加到 **New Tizen Web Project**（新建 Tizen Web 项目）窗口的 **User Template**（用户模板）选项卡，请选择 **Import a template to the new project wizard automatically**（自动将模板导入到新项目向导）。
 - e. 或者，要选择要在项目向导中显示的模板说明、预览图像和图标文件，请单击 **Next**（下一步）。
3. 单击 **Finish**（完成）。

设置项目属性

在实现实际应用程序功能前，为您的应用程序项目设置所有必需的属性。

设置项目属性：

1. 在 IDE 中，在 **Project Explorer**（项目资源管理器）视图中右键单击项目，然后选择 **Properties**（属性）。
2. 设置以下属性：
 - [API 权限](#)
 - [内部版本属性](#)
 - [JSON 属性](#)
3. 单击 **Apply**（应用）。

注意：

仅使用 IDE 中的配置编辑器设置项目属性。如果您使用其他编辑器创建或编辑 `config.xml` 文件，您的应用程序可能不会像预期一样工作。

在完成了对项目属性的设置后，[配置应用程序](#)。

检查 API 权限

检查您的项目中的源代码以便确定是否违反了任何 API 权限：

1. 在 **Properties**（属性）窗口中，依次选择 **Project > Properties > Tizen SDK > Web > Privilege**（项目 > 属性 > Tizen SDK > Web > 权限）。
2. 如果您想要在特定文件夹中排除 JavaScript 文件，请在 **Excluded Folder Settings**（排除的文件夹设置）窗格中定义文件夹。
排除文件夹可避免对库文件（例如 `tizen-ui-fw` 和 `jQuery`）进行不必要的分析。默认排除 `tizen-web-ui-fw` 文件夹。
3. 要自动执行权限检查，请在 **Mode Settings**（模式设置）面板中选择所需选项。
4. 要在生成过程中执行权限检查，请选择 **Run privilege checks with build**（针对生成运行权限检查）。

还可以通过在上下文菜单中选择 **Project > Check Privilege**（项目 > 检查权限），执行权限检查。结果将显示在 **Problems**（问题）视图中。

设置生成属性

设置生成属性：

1. 在 **Properties**（属性）窗口中，依次选择 **Tizen SDK > Package > Web**（Tizen SDK > 程序包 > Web）。
2. 选择 **Optimize Web resources**（优化 Web 资源）。
3. 在 **Optimization**（优化）窗格中，添加要从优化中排除的任何文件。

设置 JSON 属性

设置 JSON 属性：

1. 在 **Properties**（属性）窗口中，依次选择 **Tizen SDK > Web > JSON Properties**（Tizen SDK > Web > JSON 属性）。
2. 选择 **Enable JSON validation in project**（在项目中启用 JSON 验证）。

配置应用程序

每个应用程序都包含 `config.xml` 文件，该文件包含用于配置应用程序的各种属性。要编辑该 `config.xml` 文件，请通过在 **Project Explorer**（项目资源管理器）视图中双击该文件来打开它。

可以通过使用编辑器中的相应选项卡来编辑以下属性：

- [概述](#)
定义一般信息。
- [小程序](#)
定义许可信息和 UI 首选项。
- [功能](#)
定义软件和硬件功能要求。
- [权限](#)
定义用于访问受限的 API 或 API 组的权限。
- [策略](#)
定义用于请求外部网络资源权限的网络 URL。
- [本地化](#)
定义 config.xml 文件的名称、说明和许可元素的本地化信息。
- [首选项](#)
定义可以使用 Widget API 设置和检索的键值对。
- [Tizen](#)
定义特定于 Tizen 的要求信息。
- [源](#)
查看和编辑 config.xml 文件的原始 XML 代码。

在完成了对应用程序的配置后，[创建应用程序 UI](#)。

概述

通过 **Overview**（概述）选项卡，您可以定义应用程序的一般信息。

下表列出了您可以在该选项卡中编辑的属性。

表 3：概述属性

属性	说明
标识符	应用程序 ID。
版本	应用程序的当前版本。
名称	在应用程序菜单或其他上下文中显示的应用程序名称。
内容	应用程序的安装文件。
图标	应用程序的自定义图标，显示在主菜单中。 图标格式必须为具有 alpha 的 32 位 PNG，并且图标大小必须为 117 x 117 像素。

小程序

通过 **Widget**（小程序）选项卡，您可以定义应用程序的许可信息和 UI 首选项。

下表列出了您可以在该选项卡中编辑的属性。

表 4：小程序属性

属性	说明
管理小程序	
作者	创建了该应用程序的个人或组织。
电子邮件	作者的电子邮件地址。
网站	与该应用程序相关联的 IRI，例如社交网络上的主页或配置文件。
许可证	提供该应用程序的内容所基于的软件许可证。该许可证可以包含内容（例如使用协议）、再分发声明和版权许可条款。
许可 URL	与软件或内容许可证相关联的有效 IRI 或路径。
说明	应用程序的自由形式说明。
管理小程序 UI	
宽度	启动文件视口宽度。
高度	启动文件视口高度。
查看模式	首选查看模式。

功能

通过 **Features**（功能）选项卡，您可以定义应用程序的软件和硬件功能要求。定义可用于 Tizen Store 中的应用程序 filtering（筛选）。

添加所需功能：

1. 在 **Features**（功能）选项卡中，单击 **Add**（添加）。
2. 从列表中选择某一功能。
3. 单击 **Save**（保存）。

在添加功能后，您可以在 **Source**（源）选项卡中查看代码，例如：

```
<tizen:feature name="http://tizen.org/feature/network.nfc"/>
```

权限

通过 **Privileges**（权限）选项卡，您可以定义用于从应用程序访问受限的 API 或 API 组的权限。该选项卡充当用于请求要在运行时使用的应用程序的 IRI 可标识运行时组件绑定的标准化工具。

添加权限：

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

1. 在 **Privileges**（权限）选项卡中，单击 **Add**（添加）。
2. 在 **Add privilege**（添加权限）窗口中，设置以下选项之一：
 - **Internal**（内部）：从预定义的 API 列表选择一个权限。您可以通过按住 CTRL 键选择多个权限。
 - **Privilege name**（权限名称）：手动输入包含权限定义的 URL。
 - **File**（文件）：单击 **Browse**（浏览），然后选择一个权限文件（.xml 或 .widlprocxml）。
3. 单击 **Finish**（完成）。

在添加权限后，您可以在 **Source**（源）选项卡中查看代码，例如：

```
<tizen:privilege name="http://tizen.org/privilege/application.launch"/>
```

策略

通过 **Policy**（策略）选项卡，您可以定义用于请求外部网络资源权限的网络 URL。

根据 W3C 访问请求策略 (WARP)，您默认不能访问外部网络资源。如果您需要访问外部网络资源，则必须为其请求网络资源权限。

下表列出了您可以在该选项卡中编辑的属性。

表 5：策略属性

属性	说明
content-security-policy	<p>针对打包或托管应用程序的其他内容安全策略。根据 W3C 内容安全策略 1.0 定义策略字符串。</p> <p>下面的示例展示了 Source（源）选项卡的代码：</p> <pre><tizen:content-security-policy> script-src 'self' </tizen:content-security-policy></pre>
content-security-policy-report-only	<p>仅限监控目的的针对打包或托管应用程序的其他内容安全策略。</p> <p>下面的示例展示了 Source（源）选项卡的代码：</p> <pre><tizen:content-security-policy-report-only> script-src 'self'; report-uri="http://example.com/report.cgi" </tizen:content-security-policy-report-only></pre>
allow-navigation	<p>对于 Web 应用程序允许的 URL 域的列表。</p> <p>该属性是可选的。</p> <p>下面的示例展示了 Source（源）选项卡的代码：</p> <pre><tizen:allow-navigation> tizen.org *.tizen.org </tizen:allow-navigation/></pre>
access	<p>网络资源权限。</p> <p>请求网络资源权限：</p> <ol style="list-style-type: none"> 1. 单击 Add（添加）。 2. 在 Network URL（网络 URL）列中，输入您要访问的资源 URL。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

	<p>3. 要允许应用程序访问 URL 子域，请通过单击该值将 Allow subdomain（允许子域）中的值设置为 true。</p> <p>下面的示例展示了 Source（源）选项卡的代码：</p> <pre><access origin="http://www.tizen.org" subdomains="true"/></pre>
--	--

本地化

通过 **Localization**（本地化）选项卡，您可以定义 `config.xml` 文件的名称、说明和许可元素的本地化信息。

添加本地化信息：

1. 在 **Name**（名称）面板中，单击 **Add**（添加）。
2. 选择某一区域设置，并且输入您的名称。
3. 在 **Description**（说明）面板中，单击 **Add**（添加）。
4. 选择某一区域设置，然后输入应用程序的说明。
5. 在 **License**（许可证）面板中，单击 **Add**（添加）。
6. 选择某一区域设置，然后输入许可证和许可证 URI。

在添加本地化信息后，您可以在 **Source**（源）选项卡中查看代码，例如：

```
<name xml:lang="en-gb">Lee</name>

<description xml:lang="en-gb">Widget</description>

<license xml:lang="en-gb" href=" http://www.apache.org/licenses/LICENSE-2.0.html">
  Apache License, Version 2.0
</license>
```

首选项

通过 **Preferences**（首选项）选项卡，您可以定义可以使用 Widget API 设置和检索的键值对。应用程序在运行时期间使用这些键值对。

添加键值对：

1. 在 **Preferences**（首选项）选项卡中，单击 **Add**（添加）。
2. 在 **Name**（名称）列中输入键，在 **Value**（值）列中输入该键的值。
3. 要将键值对设置为只读，请通过单击该值将 **Read-only**（只读）中的值设置为 **true**。

在添加键值对后，您可以在 **Source**（源）选项卡中查看代码，例如：

```
<preference name="key" value="value" readonly="false"/>
```

Tizen

Tizen 选项卡可用于定义特定于 Tizen 的要求信息。该选项卡显示 Tizen 架构扩展。该选项卡上的某些属性是必需的，其他是可选的。

下表列出了您可以在该选项卡中编辑的属性。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅[内容许可](#)。

表 6: Tizen 属性

属性	说明
应用程序	
ID	Tizen 应用程序 ID, 它是从 Tizen 程序包 ID 和项目名称随机生成的。 该属性是必需的。
所需版本	指示应用程序支持的最低 API 版本。 此属性是必需的并且必须是浮点值, 例如 1.0 或 2.0。
内容	
src	在 W3C 小程序包装和 XML 配置中, Web 应用程序起始页是小程序包内的文档。Tizen WRT 允许在外部服务器上托管起始页。 <tizen:content /> 元素用于指向相关文档。
设置	
screen-orientation	屏幕方向模式: <ul style="list-style-type: none"> • 横向 • 纵向 • 自动旋转 该属性是可选的。默认值为 portrait 。
context-menu	支持上下文菜单。 该属性是可选的。默认值为 enable 。
background-support	设置在应用程序发送到后台时应用程序是否继续执行。 该属性是可选的。默认值为 disable 。
encryption	应用程序资源 (HTML、CSS 和 JavaScript 文件) 的加密。 该属性是可选的。默认值为 disable 。
install-location	安装位置, 例如 SD 卡。 该属性是可选的。默认值为 auto 。
hwkey-event	支持硬件键。 该属性是可选的。默认值为 enable 。
app-control	
	<p>描述应用程序提供的应用程序控制功能。operation (操作)、uri 和 mime 字段描述其他应用程序可以请求的功能, src 字段描述处理请求的应用程序页。</p> <p>下面的示例展示了 Source (源) 选项卡的代码:</p> <pre><tizen:app-control> <tizen:src name="edit.html"/> <tizen:operation name="http://tizen.org/appcontrol/operation/edit"/> <tizen:uri name="file"/> <tizen:mime name="image/jpeg"/></pre>

除非另有注明, 否则, 本文中的内容 (代码示例除外) 基于 [Creative Commons Attribution 3.0](#) 获得许可, 而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息, 请参阅 [内容许可](#)。

	<code></tizen:app-control></code>
app-widget	
id	Dynamic Box ID。 ID 格式为 <code>[<tizen:application> id].[any string]</code> 。
primary	在您单击应用程序图标以便添加 Dynamic Box 并且使用 Add from app tray （从应用程序托盘添加）选项后，主 Dynamic Box 将显示在主屏幕上。您可以通过使用 Add more app-widget （添加更多程序和小程序）选项添加辅助 Dynamic Box。 只能有一个 Dynamic Box 可以将该属性的值设为 true 。
auto-launch	要在单击 Dynamic Box 时启动应用程序，请将该属性的值设置为 true 。
update-period	用于确定重新加载 Dynamic Box 的频率（以秒为单位）。最小值为 1800.0。默认行为是 no-update 。
box-label	Dynamic Box 的名称。
box-icon	在您选择 Add more app-widget （添加更多程序和小程序）选项时该图标显示在该列表上。
box-content	src 属性指定基于应用程序根文件夹的起始页的相关路径。 mouse-event 属性指定指示 Dynamic Box 内容接收鼠标向下和向上事件的标志。 touch-effect 属性指定指示在用户单击 Dynamic Box 时必须应用于 Dynamic Box 的特殊 UI 效果的标志。
box-size	preview 属性指定在 Dynamic Box 上显示的图像文件的路径。 use-decoration 属性指定 Dynamic Box Viewer 是否将框装饰应用于框大小。
pd	src 属性指定基于应用程序根文件夹的起始页的相关路径，并且用于显示 Drop View。还可以定义 Drop View 高度。Drop View 宽度始终固定到设备屏幕宽度。
帐户	
显示名称	帐户提供程序的显示名称。 该属性是必需的。
多个帐户	设置是否支持多个帐户。 该属性是必需的。
图标	表示帐户提供程序的图标的路径。该图标图像由帐户设置使用，并且必须放置于共享文件夹中。该大小为 72 x 72 像素。 该属性是必需的。
小图标	表示帐户提供程序的小图标的路径。该图标图像由帐户设置使用，并且必须放置于共享文件夹中。该大小为 45 x 45 像素。 该属性是必需的。
功能	以 IRI 格式定义的帐户提供程序的功能： <code>http://<vendor information>/accounts/capability/<name></code> 可以使用以下预定义功能：

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

	<ul style="list-style-type: none"> • http://tizen.org/account/capability/contact 在帐户与联系人相关时使用。 • http://tizen.org/account/capability/calendar 在帐户与日历相关时使用。 <p>该属性是可选的。</p>
元数据	
键	<p>可通过 Tizen 应用程序 API 访问（只读）元数据。 下面的示例展示了 Source（源）选项卡的代码：</p> <pre><tizen:metadata key="key1" /></pre> <p>该属性是可选的。该值必须是唯一字符串。</p>
值	<p>可通过 Tizen 应用程序 API 访问（只读）元数据。 下面的示例展示了 Source（源）选项卡的代码：</p> <pre><tizen:metadata key="key2" value="value" /></pre> <p>该属性是可选的。该值必须是字符串。</p>

源

通过 **Source**（源）选项卡，可以查看和编辑 `config.xml` 文件的原始 XML 代码。对其他选项卡进行的任何更改都将显示在 `config.xml` 文件中。

注意：

该 `config.xml` 文件必须不仅要符合 XML 文件格式，还要符合 W3C 规格要求。以 XML 格式编辑配置文件仅供高级用户使用。

使用 UI Builder 创建应用程序 UI

Tizen IDE 提供 UI Builder 以便创建应用程序 UI。UI Builder 通过允许您使用拖放 WYSIWYG（所见即所得）编辑器排列小程序，简化了 UI 创建过程。

有关使用其他方法创建应用程序 UI 的详细信息，请参阅[使用 Tizen 高级 UI](#)。

使用 UI Builder 创建应用程序 UI

1. [创建 UI Builder 项目](#)
2. [添加新页](#)
3. [通过添加和排列 UI 小程序设计页面](#)
4. [处理 UI 小程序事件](#)
5. [测试 UI Builder 项目](#)

有关 UI Builder 的详细信息，请参阅 Tizen 开发指南中的 [UI Builder](#)。

创建 UI Builder 项目

UI Builder 项目在 IDE 工作区中创建用于生成 Tizen Web 应用程序的骨干源代码以及所需文件和文件夹。您可以使用 UI Builder 创建和修改页面布局。UI Builder project 项目使用 [Tizen Web UI Builder 框架](#) 作为编程模型的指导。

创建 UI Builder 项目类似于[创建 Web 应用程序项目](#)。在 **New Web Project**（新建 Web 项目）窗口中，选择 Tizen Web UI Builder 项目模板。

添加页面

您可以将多个页面添加到 UI Builder 应用程序。

添加新页：

1. 在 IDE 中，依次选择 **File > New > Other**（文件 > 新建 > 其他）。
2. 在 **New**（新建）窗口中，依次选择 **Tizen > Tizen Web UI Builder Page**（Tizen > Tizen Web UI Builder 页）。
3. 在 **New Page File**（新建页文件）窗口中，为页面选择 UI Builder 项目并且输入新的页 ID。
4. 单击 **Finish**（完成）。

新页将显示在 **Page Designer**（页设计器）视图中。

设计页

您可以使用 **Page Designer**（页设计器）视图从页面中添加和删除 UI 小程序、将小程序排列为您想要的布局以及设置小程序属性。您可以将多种不同类型的小程序添加到页面。

您可以通过设置目标设备分辨率、放大和缩小以及将设计区域缩放到可用屏幕空间，配置页面的设计区域。

向页面添加小程序：

1. 在 IDE 的 **Pages**（页）视图中，单击要将小程序添加到的页面的缩略图。

Page Designer（页设计器）视图将显示该页的设计区域。

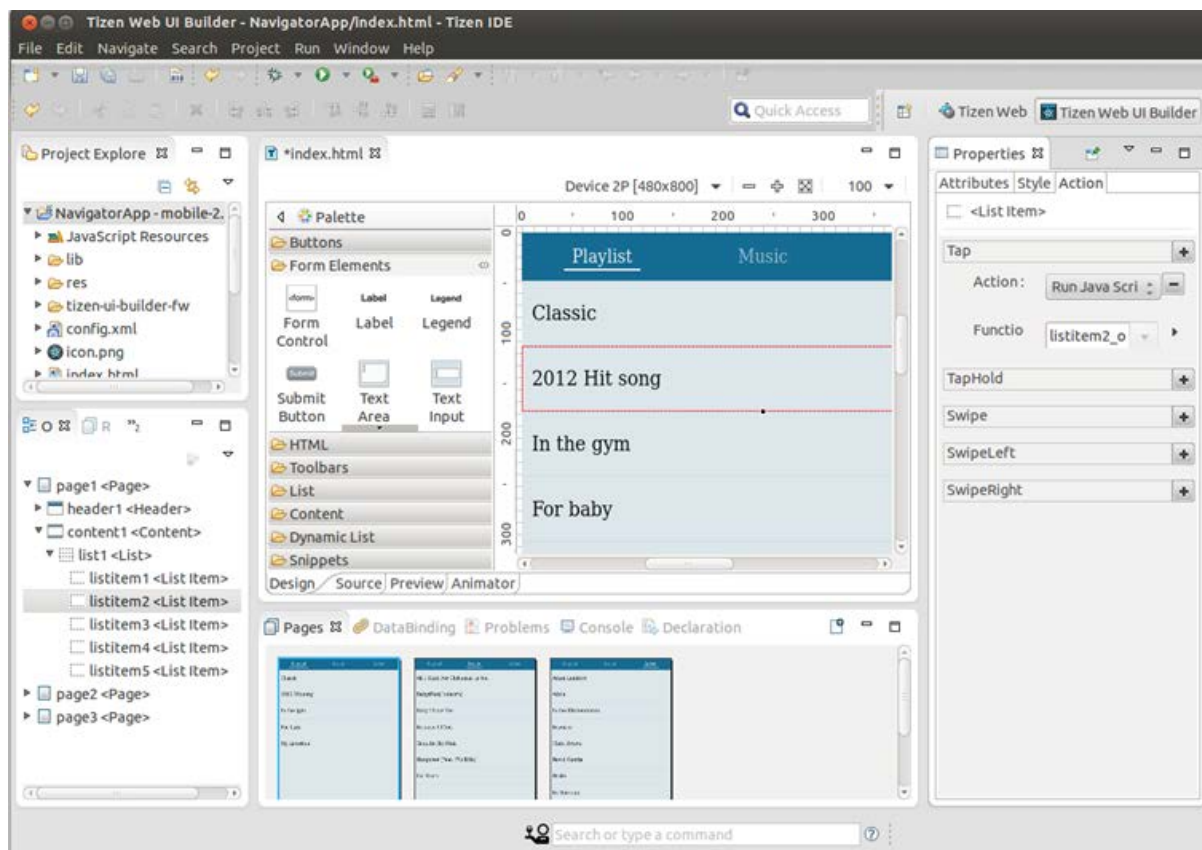


图 5: Tizen IDE 中的 UI Builder 项目

2. 将小程序从 **Palette**（调色板）区域拖到设计区域。

设计区域中下一个空位置用虚点构成的长方形标记。父小程序或内容对象为蓝色。有关添加小程序的详细信息，请参阅[放置小程序](#)。

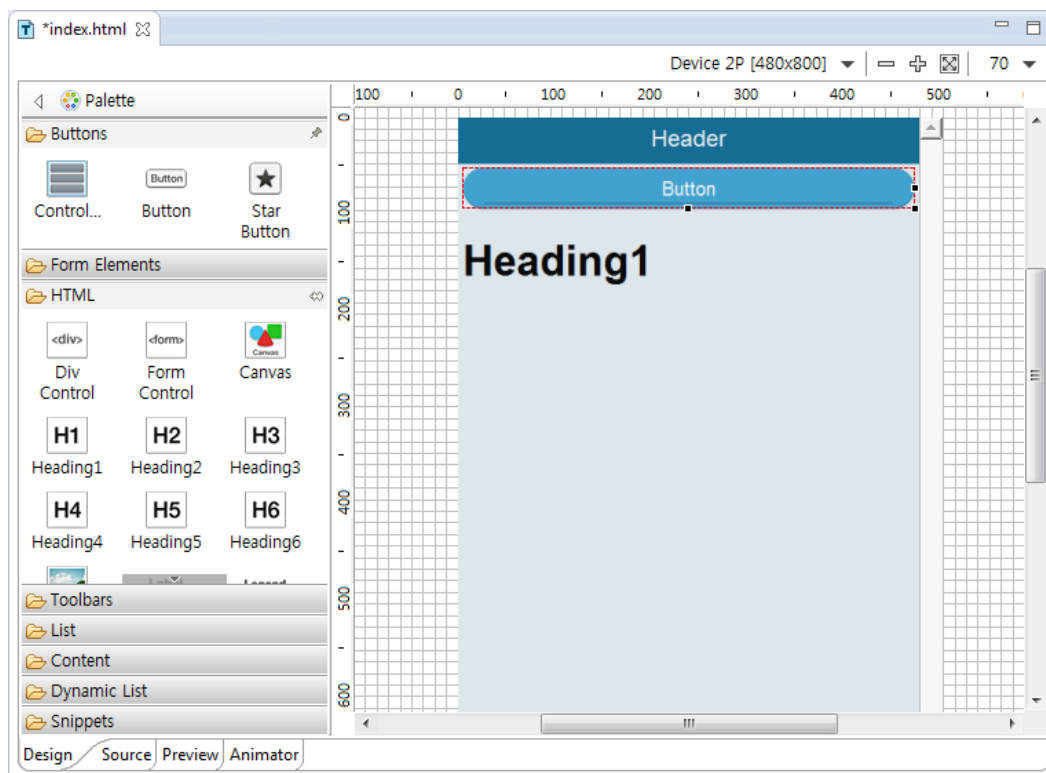


图 6：将小程序从调色板添加到页面

3. 在设计区域中或 **Outline**（大纲）视图中选择小程序。

下图显示 **Outline**（大纲）视图的特写，列表项小程序嵌套在列表小程序内。

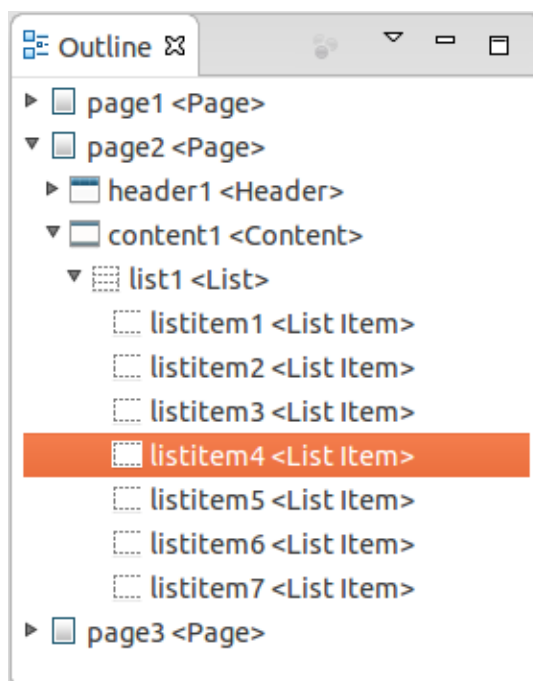


图 7：大纲视图

- 在 **Properties**（属性）视图中，设置小程序属性。

下图显示列表项小程序的 **Properties**（属性）视图，列表项文本突出显示以便用于编辑。

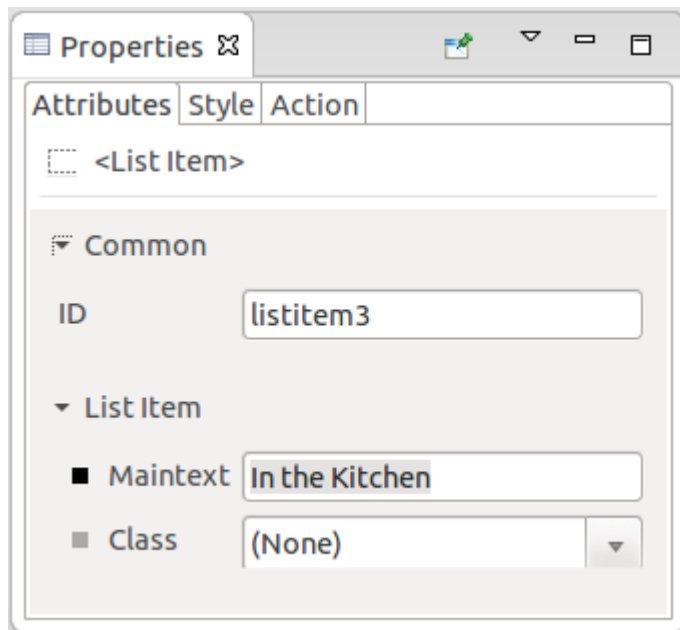


图 8：属性视图

- 根据需要在设计区域中移动小程序。

放置小程序

您可以将子小程序添加到容器小程序中，例如 **List** 中。例如，可以将一组 **ListItem** 小程序添加到 **List** 小程序中。如果放置于设计区域中的某个 **List** 小程序不包含任何 **ListItem** 小程序，则 **List** 小程序将显示消息“Drop a List Item”（删除列表项）。

只能将特定类型的小程序添加到容器小程序中。例如，只能将 **ListItem** 小程序添加到 **List** 小程序中。如果您尝试将 **ListItem** 小程序添加到其他类型的小程序中，则设计区域将变为红色，指示该小程序不能放置于那里。如果您将某一小程序添加到正确类型的容器小程序中，则设计区域将变为蓝色，指示这是有效放置。

处理针对页面的事件

为小程序添加事件处理程序：

- 在 IDE 的 **Page Designer**（页设计器）视图的设计区域中，选择该小程序。
- 在 **Properties**（属性）视图中，选择 **Action**（操作）选项卡。
- 单击要添加事件处理程序的事件的加号按钮，然后从 **Action**（操作）下拉列表中选择 **Run JavaScript Function**（运行 JavaScript 函数）。
- 在 **Function**（函数）字段中，定义事件处理程序的名称，然后单击该字段旁的播放按钮。

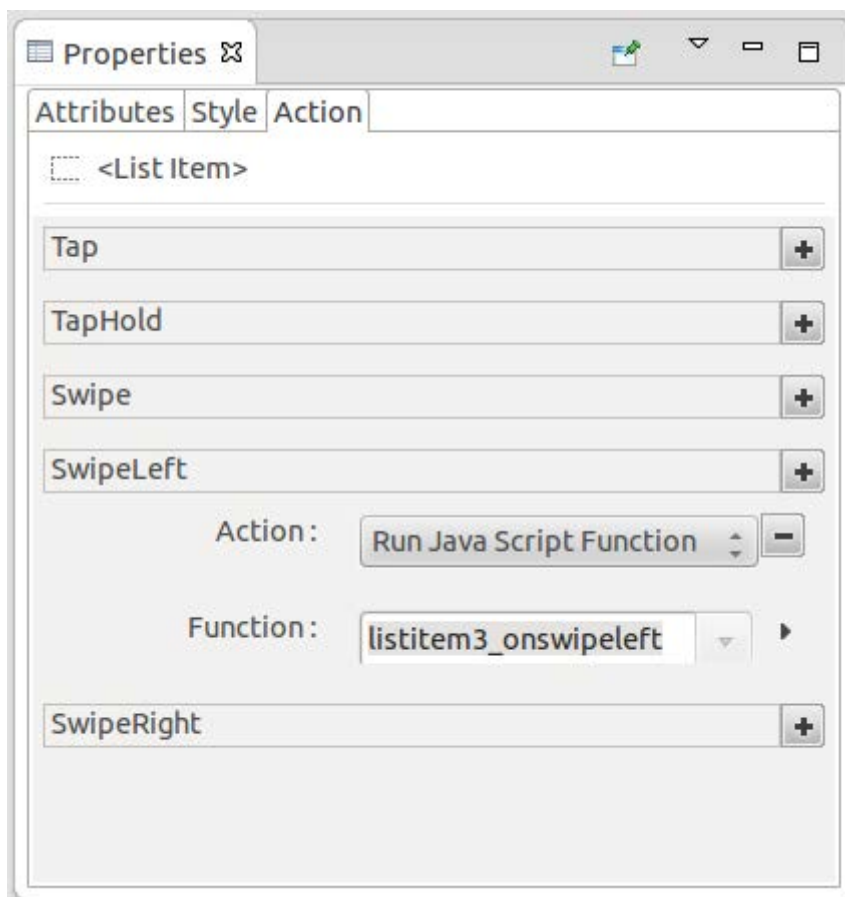


图 9：将 SwipeLeft 处理程序添加到列表项。

UI Builder 为事件处理程序生成骨干代码，例如：

```
/**
 * @param {Object} event
 * @base _page1_page
 * @returns {Boolean}
 */
_page1_page.prototype.listitem3_onswipeleft = function(event) {
};
```

5. 输入特定于应用程序的事件处理程序代码。

每个小程序都具有自己的唯一 ID，并且您可以通过使用该 ID 从事件处理程序访问该小程序。可以在 **Outline**（大纲）视图中查看和编辑该小程序 ID。

例如，下面的代码在用户从左侧扫过 listitem3 小程序上方时显示警报窗口：

```
_page1_page.prototype.listitem3_onswipeleft = function(event)
{
    window.alert("Hello World");
};
```

测试 UI Builder 项目

测试 UI Builder 项目：

1. 在 IDE 中，在 **Project Explorer**（项目资源管理器）视图中选择该项目。
2. 要在仿真器或目标设备中测试该项目，请依次选择 **Run > Run as > Tizen Web Application**（运行 > 运行方式 > Tizen Web 应用程序）。
3. 要在模拟器中测试该项目，请依次选择 **Run > Run as > Tizen Web Simulator Application**（运行 > 运行方式 > Tizen Web Simulator 应用程序）。

生成应用程序

在您可以运行和调试应用程序之前，必须首先生成该应用程序。您可以自动或手动生成应用程序：

- 要自动生成该应用程序，请在 IDE 中，依次选择 **Project > Build Automatically**（项目 > 自动生成）。
如果您选择该选项，则只要更改和保存源代码或资源，该 IDE 就自动重新生成该应用程序。
- 要手动生成该应用程序，请在 IDE 中，依次选择 **Project > Build Project**（项目 > 生成项目）。
如果您选择该选项，则可以在方便时生成您的项目。

注意：

如果您选择手动生成应用程序：

- 确保您在运行或调试项目前具有最新的生成输出。
- 要删除项目生成输出，请在 IDE 中，依次选择 **Project > Clean**（项目 > 清除）。

运行和调试应用程序

在生成您的应用程序后，在以下一个或多个环境中运行并且调试该应用程序：

- **模拟器**
通过 Tizen Web Simulator，您可以运行使用 Tizen Web API 的应用程序。
- **仿真器**
Tizen SDK Emulator 模拟用于运行 Tizen Web 应用程序的目标环境。使用该环境，您可以在将您的应用程序部署到实际目标设备前测试该应用程序。
- **目标设备**
通过在目标设备上运行您的应用程序，您可以在实际目标环境中调试和测试您的应用程序。

您可以通过使用快速开发支持 (RDS) 更快地运行和调试您的应用程序。有关您可以使用的调试方法和工具の詳細信息，请参阅[调试应用程序](#)。

在完成对您的应用程序的调试后，您可以打包应用程序了。

在模拟器中运行应用程序

注意：

Tizen Web Simulator 要求 Google Chrome 以便运行。要使用 Simulator，请首先下载并且安装 Google Chrome，然后在模拟器首选项中指定浏览器安装位置。

在模拟器中运行应用程序：

1. 如果您是首次运行该应用程序，则通过从菜单中依次选择 **Run > Run Configurations > Tizen Web Simulator Application**（运行 > 运行配置 > Tizen Web Simulator 应用程序）来运行配置。运行配置包含必需的应用程序启动设置。
2. 通过执行以下操作之一运行应用程序：
 - 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Run As > Tizen Web Simulator Application**（运行方式 > Tizen Web Simulator 应用程序）。
 - 在菜单中，依次选择 **Run > Run As > Tizen Web Simulator Application**（运行 > 运行方式 > Tizen Web Simulator 应用程序）。
 - 在工具栏中，单击 **Run**（运行）。

在应用程序启动后，该模拟器将加载在 `config.xml` 文件的 **Content**（内容）字段中指定的文件。最通常指定的文件是 `index.html`。

该模拟器将使用 WebKit 在浏览器中呈现您的应用程序。所有 Google Chrome 开发功能均在模拟器中提供，包括 Remote Inspector 工具，您可以通过按下 **F12** 键打开该工具。

通过模拟器，您可以设置设备屏幕尺寸和方向，将事件和消息（例如地理位置数据和传感器输入事件）发送到您的应用程序以便进行调试。

在仿真器中运行应用程序

注意：

运行仿真器要求最少 20 MB 的可用磁盘空间。该映像可能占用最多 10 GB 的磁盘空间。

在仿真器中运行应用程序：

1. 启动仿真器：
 - a. 从桌面 (Linux) 或“开始”菜单 (Windows) 或者从命令行启动 Emulator Manager。Mac OS® X 并不提供针对 Emulator Manager 的快捷方式。
 - b. 在 Emulator Manager 的 **Name**（名称）列中选择该仿真器映像
如果您是首次使用 Emulator Manager，将创建仿真器映像：
 - i. 单击 **Create New**（新建）。
 - ii. 设置映像详细信息。
 - iii. 单击 **Save**（保存）。
 - c. 单击 **Launch Emulator**（启动仿真器）。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅[内容许可](#)。

2. 通过执行以下操作之一运行应用程序：
 - 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Run As > Tizen Web Application**（运行方式 > Tizen Web 应用程序）。
 - 在菜单中，依次选择 **Run > Run As > Tizen Web Application**（运行 > 运行方式 > Tizen Web 应用程序）。
 - 在工具栏中，单击 **Run**（运行）。
3. 要停止该仿真器，请右键单击该仿真器，然后单击 **Close**（关闭）。

在目标设备上运行应用程序

在目标设备上运行应用程序：

1. 将目标设备连接到您的计算机。
2. 在 IDE 中，依次选择 **Run > Run Configurations**（运行 > 运行配置）。
3. 在 **Run Configurations**（运行配置）窗口中，单击 **New Launch Configuration**（新的启动配置）。

快速开发支持用于跳过程序包上载并且加快应用程序运行速度。RDS 默认启用。要禁用它，请依次选择 **Window > Preferences > Tizen SDK > Rapid Development Support**（窗口 > 首选项 > Tizen SDK > 快速开发支持）。
4. 使用 **Timeout**（超时）值滑块设置超时。

超时值表示应用程序启动的等待时间。如果您在使用较低配置的计算机，则设置一个较高的超时值，以免发生应用程序启动失败错误。
5. 单击 **Run**（运行）。

如果在运行配置中不需要任何更改，则可以通过执行以下操作之一在目标设备上运行应用程序：

- 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Run As > Tizen Web Application**（运行方式 > Tizen Web 应用程序）。
- 在菜单中，依次选择 **Run > Run As > Tizen Web Application**（运行 > 运行方式 > Tizen Web 应用程序）。
- 在工具栏中，单击 **Run**（运行）。

如果您的应用程序成功在目标设备上启动，则 **JavaScript Log Console**（JavaScript 日志控制台）将自动在 IDE 中启动。该视图显示应用程序 JavaScript 日志。

调试应用程序

通过调试应用程序，您可以了解其控制流。您可以通过在目标设备上运行应用程序并且调试其 JavaScript 代码来调试该应用程序。JavaScript 代码调试使用 Remote Inspector 工具。

在目标设备上开始调试应用程序：

1. 将目标设备连接到您的计算机。
2. 在 IDE 中，通过执行以下操作之一打开 **Debug Configurations**（调试配置）窗口：
 - 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Debug As > Debug Configurations**（调试方式 > 调试配置）。

- 在菜单中，依次选择 **Run > Debug Configurations**（运行 > 调试配置）。
3. 在 **Debug Configurations**（调试配置）窗口中，单击 **New Launch Configuration**（新的启动配置）。

[快速开发支持](#) (RDS) 用于跳过程序包上载并且加快应用程序运行速度。RDS 默认启用。要禁用它，请依次选择 **Window > Preferences > Tizen SDK > Rapid Development Support**（窗口 > 首选项 > Tizen SDK > 快速开发支持）。

4. 使用 **Timeout**（超时）值滑块设置超时。

超时值表示应用程序启动的等待时间。如果您在使用较低配置的计算机，则设置一个较高的超时值，以免发生应用程序启动失败错误。

5. 单击 **Debug**（调试）。

如果在调试配置中无需任何更改，则可以通过执行以下操作之一在目标设备上调试应用程序：

- 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Debug As > Tizen Web Application**（调试方式 > Tizen Web 应用程序）。
- 在菜单中，依次选择 **Run > Debug As > Tizen Web Application**（运行 > 调试方式 > Tizen Web 应用程序）。
- 在工具栏中，单击 **Debug**（调试）。

在调试开始后，Remote Inspector 将显示在一个新的 Google Chrome 窗口中。您可以使用 Remote Inspector 执行以下调试任务：

- 检查样式
- 检查 DOM
- 检查资源
- [调试 JavaScript 代码](#)

注意：

Remote Inspector 始终在新浏览器窗口中打开。不支持在要调试的应用程序和 Remote Inspector 之间的生命周期同步。

为使 Remote Inspector 工作，必须在设备上安装 Google Chrome。当 Google Chrome 安装在该设备时，Tizen SDK 会自动检测到它。要选择浏览器路径，请转到 **Window > Preferences > Tizen SDK > Web > Chrome**（窗口 > 首选项 > Tizen SDK > Web > Chrome）。

调试 JavaScript 代码

注意：

您在调试 JavaScript 代码前必须启用调试。






要调试 JavaScript 代码，请从 Remote Inspector 菜单中单击 **Sources**（源）。

您可以通过在编辑器左侧的标记栏区域中右键单击，然后选择 **Toggle Breakpoint**（切换断点），在代码中设置断点。

在设置了断点后，您可以查看变量、表达式和当前回调。还可以通过使用以下控制按钮控制调试。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

表 7：用于在断点之间进行调试的控制按钮

按钮	说明
	恢复当前执行。
	逐句跳过突出显示的语句。 执行当前行，并且如果该行包含某一方法，则执行该方法而不进入该方法。
	单步执行突出显示的语句。 执行当前行，并且如果该行包含某一方法，则单步执行该方法。
	跳出当前方法。
	停用所有断点。

如果您的应用程序成功在目标设备上启动，则 **JavaScript Log Console**（JavaScript 日志控制台）将自动在 IDE 中启动。该视图显示应用程序 JavaScript 日志。

快速开发支持

借助于快速开发支持 (RDS)，您可以通过节约部署时间快速开发 Tizen 应用程序。

在使用 RDS 时，将首先正常启动应用程序（正常模式）。在首次启动后，将跳过打包过程，并且只有已修改、添加和删除的文件将安装在目标设备上（RDS 模式）。如果在启动时出现某个错误，则应用程序将在正常模式下启动。

在正常模式下启动应用程序：

1. 打包应用程序。
2. 将打包后的文件传输到目标设备。
3. 在目标设备上安装打包后的文件。如果应用程序已安装，则在安装前卸载已安装的应用程序。

在 RDS 模式下启动应用程序：

1. 搜索增量文件（已更改、已添加和已删除的文件）。
2. 将增量文件传输到目标设备。
3. 如果修改了 `config.xml` 文件，则运行文件夹配置。

RDS 选项默认启用。要禁用它，请在 IDE 中，依次选择 **Window > Preferences > Tizen SDK > Rapid Development Support**（窗口 > 首选项 > Tizen SDK > 快速开发支持）。

注意：

在多应用程序项目中不支持 RDS。

如果您想要从设备中删除某一应用程序，则必须手动删除已安装的应用程序，因为启动过程不包含卸载。

打包应用程序

在成功运行并调试您的应用程序后，您需要将其打包。您可以打包应用程序并且启用单次下载并安装。通过 IDE 提供的应用程序包装管理器，您可以在仿真器中或目标设备上安装并卸载该程序包。

注意：

必须使用作者证书对应用程序进行签名。如果您不具有证书，则使用证书生成程序创建证书，然后在 IDE 中注册该证书。

Web 应用程序包装基于 W3C 包装和配置规范。您可以使用以下生成选项管理包装：

• 自动

在使用自动生成选项时，IDE 将自动识别源和资源中所有已保存的更改，然后重新生成项目。自动生成选项默认启用。要禁用或启用该选项，请在 IDE 中，依次选择 **Project > Build Automatically**（项目 > 自动生成）。

• 手动

通过手动生成选项，您可以手动使用您所做更改手动生成项目。要使用该选项：

- 在 IDE 中，通过选择 **Project > Build Automatically**（项目 > 自动生成）禁用自动生成选项。
- 通过选择 **Project > Clean**（项目 > 清除）删除项目生成输出。
- 通过选择 **Project > Build Project**（项目 > 自动生成）重新生成项目。

注意：

在使用手动生成选项时，确保在运行或调试项目时生成输出是最新版本。

还可以通过从命令行运行 `web-packaging` 命令打包您的应用程序：

```
web-packaging project.wgt project/
```

默认情况下，将创建应用程序包装一次。您可以在应用程序开发过程中的任何点查看程序包内容。

查看应用程序包

要查看某一应用程序包的内容，请在 **Project Explorer**（项目资源管理器）视图中双击项目 `.wgt` 文件。在应用程序项目中提供的所有文件将显示在列表中。

对程序包内容列表中的文件的任何更改（例如删除文件或拖放文件）将不会反映在实际项目文件中。

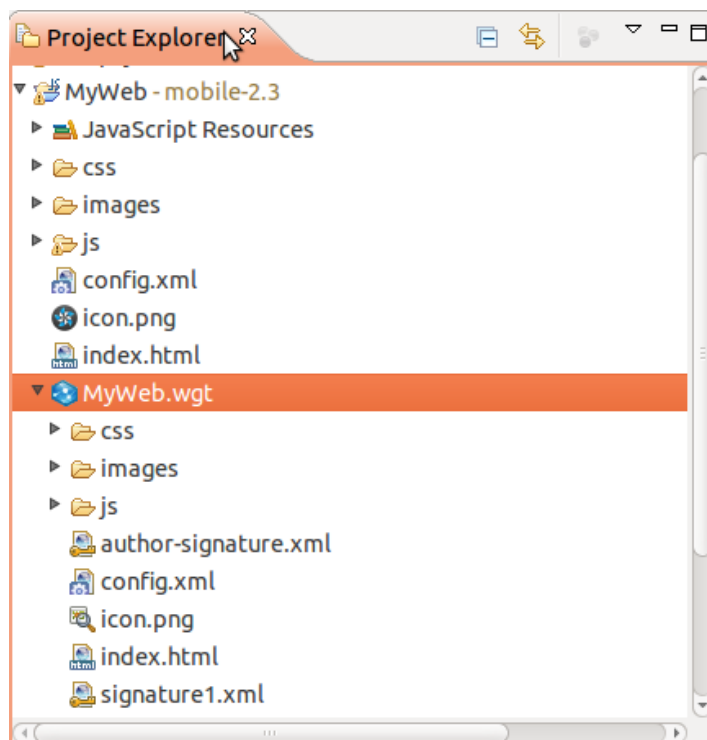


图 10: 查看应用程序包的内容

本地化应用程序

您可以通过为不同区域设置创建单独的应用程序版本，为不同语言和区域性本地化您的应用程序。

本地化您的应用程序：

1. 通过执行以下操作之一启动本地化向导：
 - 在 **Project Explorer**（项目资源管理器）视图中，右键单击该项目，然后选择 **Localization > Localization Wizard**（本地化 > 本地化向导）。
 - 在菜单中，依次选择 **Project > Localization > Localization Wizard**（项目 > 本地化 > 本地化向导）。
2. 在 Localization Wizard（本地化向导）中，从文件列表中选择要本地化的文件，然后单击 **Next**（下一步）。

已本地化的文件将灰显。
3. 在 **Available locales**（可用区域设置）列表中，选择区域设置并且将它们添加到 **Selected locales**（所选区域设置）列表中。单击 **Next**（下一步）。

已支持的区域设置将灰显。
4. 通过选中所需区域设置列下的复选框，将文件与特定区域设置相关联。

已为特定区域设置本地化的文件的复选框将灰显。
5. 单击 **Finish**（完成）。

在 **Project Explorer**（项目资源管理器）视图中，将创建一个新的 `locales`（区域设置）文件夹，其中对于每个区域设置都包含单独的子文件夹。子文件夹包含所选文件的副本。使用这些副本创建文件的本地化版本。

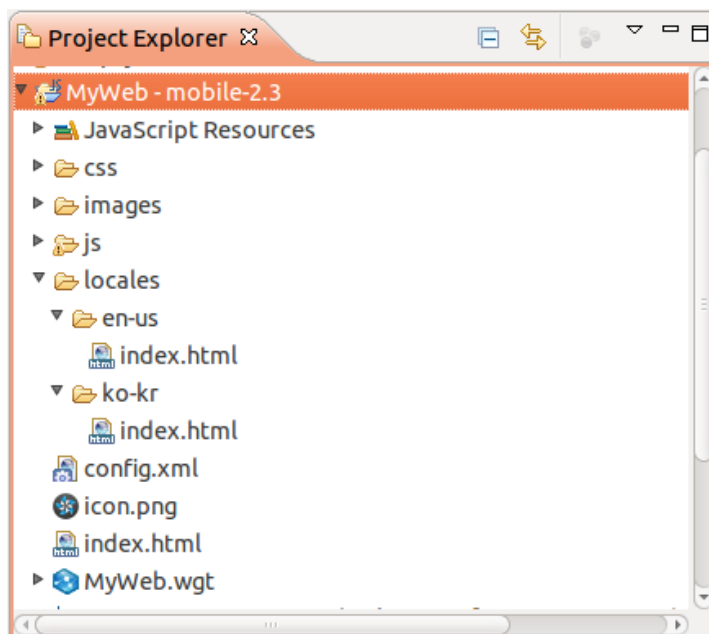


图 11：区域设置子文件夹

5.使用 Tizen 高级 UI

本章介绍 Tizen 高级 UI (TAU) 框架以及它所提供的基本 UI 元素，并且向您展示 Tizen 对最新 CSS 功能的支持是如何允许您创建类似于 Flipboard 的效果的。

有关使用其他方法创建应用程序 UI 的详细信息，请参阅[使用 Tizen 高级 UI](#)。

Tizen 高级 UI 简介

Tizen 高级 UI 是用于开发 Web 应用程序的 HTML、CSS 和 JavaScript 框架。该框架提供向您的应用程序提供唯一 Tizen 风格的 UI 元素。该框架还提供用于处理 UI 元素的事件和实用程序方法。

因为 HTML 已支持有意义的基元元素，例如页眉、内容、页脚和按钮，所以，TAU 仅为基元元素以及其他一些功能提供主题。

开始使用简单的应用程序

开始使用简单的单页 Web 应用程序：

1. [创建 Web 应用程序项目](#)并且选择 **Tizen Web UI Framework > Single Page Application**（Tizen Web UI 框架 > 单页应用程序）作为应用程序模板。

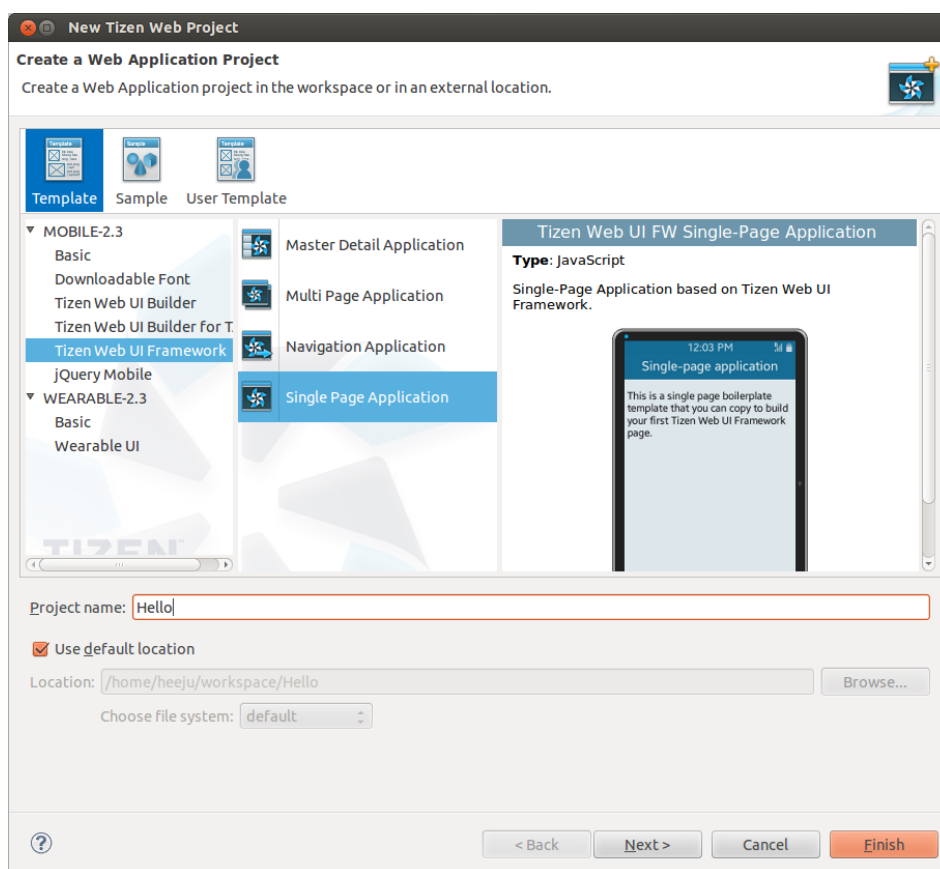


图 12: 为新的 Web 应用程序项目选择模板

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅[内容许可](#)。

2. 为应用程序声明视口元标记:

```
<meta name="viewport" content="..." />
```

视口是 Web 引擎在其中显示 UI 的屏幕区域。视口元数据标记指示 Web 引擎以某一特定的形式系数（例如特定屏幕宽度）为目标。这允许 Web 引擎为当前设备和屏幕上的应用程序 UI 确定正确的比例系数。例如，要为您的应用程序指定 360 个像素的目标宽度，请声明以下视口元标记：

```
<meta name="viewport" content="width=360" />
```

如果您在创建为每个设备动态调整其 UI 的响应性应用程序，则声明以下视口元标记：

```
<meta name="viewport" content="width=device-width" />
```

device-width 值将目标宽度设置为当前设备屏幕的理想宽度。

除了 width 之外，还可以在视口元标记中指定其他形式系数属性，例如 height、initial-scale、minimum-scale、maximum-scale 和 user-scalable。但是，使用其他这些属性时要小心。错误的值可能导致 Web 引擎显示不正确的 UI。

3. 导入 TAU CSS 和 JavaScript 文件:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width" />
<link rel="stylesheet" href="lib/tau/mobile/theme/default/tau.min.css">
<script type="text/javascript" src="lib/tau/mobile/js/tau.min.js">
...
</head>
<body>
...
</body>
</html>
```

该模板提供您的应用程序代码的基本框架，这样，您现在就可以开发应用程序。

添加页面

要在设备屏幕上显示 UI，您需要将至少一个视图添加到应用程序。每个视图都会占满整个屏幕。在 TAU 中，视图声明为页。

要将页添加到应用程序，请使用以下代码：

```
<div data-role="page"><!--Page area (mandatory)-->
  <div data-role="header"><!--Header area (optional)--></div>
  <div data-role="content"><!--Content area (optional)--></div>
  <div data-role="footer"><!--Footer area (optional)--></div>
</div>
```

该页声明为具有 data-role="page" 属性的 <div> 元素。在 <div> 元素内，您可以如下图中所示为页定义页眉、内容和页脚。

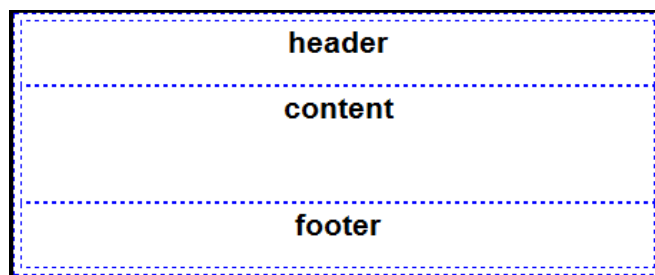


图 13: 页结构

处理针对页面的事件

TAU 还提供不同的自定义事件，以便在应用程序 UI 和 JavaScript 代码之间定义挂钩。这些事件补充了平台的固有事件，例如，包括了基于触摸的手势事件，例如轻擦、拖动和轻捏。

例如，要处理 UI 中的轻擦和拖动事件，请使用以下代码作为基础：

```
var gesture = new Gesture(element,
{
  dragOrientation:tau.gesture.Orientation.VERTICAL
})
.on("drag dragend dragcancel swipe", function(event)
{
  /* Handle swipe and drag events */
});
```

还可以将 [jQuery](#) 用于 TAU。

可用的 UI 小程序

下表列出了在 TAU 中提供的 UI 小程序。

表 8: 在 TAU 中提供的 UI 小程序

小程序	说明
Autodividers	Autodividers 小程序自动将列表视图小程序的分配器作为扩展名创建。
Button	Button 小程序将默认浏览器按钮更改为具有附加功能（例如图标、边角和阴影）的特殊按钮。
Checkbox radio	Checkbox radio 小程序将默认浏览器复选框和单选按钮更改为更适合于移动环境的形式。
Collapsible	Collapsible 小程序允许您在点击时展开或折叠内容。
Control group	Control group 小程序通过将多个按钮组合成单个块，改进了多个按钮的风格。
Date-time picker	Date-time picker 小程序显示一个控件，用户可以使用该控件输入日期和时间值。
Drawer	Drawer 小程序允许您打开和关闭一个 Drawer，以便显示或隐藏其中的内容。
Fast scroll	Fast scroll 小程序显示一个快捷方式列表，它绑定到其父滚动条和相应的列表视图。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。有关详细信息，请参阅 [内容许可](#)。

Flip toggle switch	Flip toggle switch 小程序是用于二进制开/关或 true/false 数据输入的常见 UI 元素。
Gallery	Gallery 小程序在屏幕上显示库中的元素。
List divider	List divider 小程序创建可用于对列表项进行分组的列表分隔符。
List view	List view 小程序用于以列表格式进行显示，例如，显示导航数据、结果和数据条目等。
Loader	Loader 小程序在页面发生变化时显示加载程序弹出窗口。
Notification	Notification 小程序在屏幕上显示一个弹出窗口以便提供通知。
Popup	Popup 小程序支持两个弹出窗口：定位到窗口弹出窗口（例如系统弹出窗口）和上下文弹出窗口。
Progress	Progress 小程序显示某一操作正在进行中。
Progress bar	Progress bar 小程序显示一个控件，该控件指示正在进行中的操作的进度百分比。
Scroll handler	Scroll handler 小程序是 Scroll view 小程序的扩展，并且添加了类似于控点的滚动按钮。
Search bar	Search bar 小程序用于搜索页面内容。
Slider	Slider 小程序将范围类型浏览器输入更改为滑块。
Swipe	Swipe 小程序在屏幕上显示一个列表视图，可在其中垂直请擦过列表项以便显示一个菜单。
Tab bar	Tab bar 小程序在屏幕上显示按钮的未排序的列表，这些按钮一起包装到单个组中。
Token text area	Token text area 小程序将文本项更改为按钮。
Virtual list view	Virtual list view 小程序用于使用动态数据管理以列表格式显示数据元素。

处理多个页面、小程序和事件

下面的示例显示如何创建一个应用程序，该应用程序具有两页，每一页上都有一个 Button 小程序以及用于更改当前显示的页的事件处理程序：

1. 通过使用 UI Builder 或手动编辑 HTML 代码向应用程序添加一页：

```
<div data-role="page" id="page1">
  <div data-role="header">
    <h1>Page 1</h1>
  </div>
  <div data-role="content">
    Swipe right or
    <button id="button1" data-role="button">Press button</button>
    to change to page 2
  </div>
</div>
```

上面的代码添加一页，该页具有一些文本以及 Button 小程序。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。有关详细信息，请参阅 [内容许可](#)。

2. 将具有一些文本以及 Button 小程序的第二页添加到该应用程序:

```
<div data-role="page" id="page2">
  <div data-role="header">
    <h1>Page 2</h1>
  </div>
  <div data-role="content">
    Swipe left or
    <button id="button2" data-role="button">Press button</button>
    to change back to page 1
  </div>
</div>
```

3. 定义用于在用户从左或从右轻擦过屏幕或单击按钮时更改当前显示的页面的事件和事件处理程序:

```
<script>
tau.event.one(document, "pageshow", function()
{
  /* Swipe handlers */
  $("#page1").on("swipeleft", function(event)
  {
    tau.changePage("#page2");
  });
  $("#page2").on("swiperight", function(event)
  {
    tau.changePage("#page1");
  });

  /* Button click handlers */
  $("#button1").on("click", function(event)
  {
    tau.changePage("#page2");
  });
  $("#button2").on("click", function(event)
  {
    tau.changePage("#page1");
  });
})
</script>
```

tau.changePage() 方法将当前显示的页面更改为具有指定 ID 的页面。如果用户在第 1 页上从左或从右轻擦过或单击按钮, 则应用程序将切换到第 2 页, 反之亦然。

6.使用 Tizen Web API

本章展示如何使用 Tizen Web API 开发以下类型的功能：

- [日历](#)
- [联系人](#)
- [消息传递](#)
- [多媒体](#)
- [NFC](#)

日历

通过日历 API，您可以在日历中管理事件和任务。

日历是事件或任务的集合，取决于日历类型。每个事件和任务都具有一系列属性，例如目的、开始时间和持续时间。

使用 `CalendarItemId` 类型定义标识事件和任务，该类型定义或者是 `CalendarTaskId`（对于任务），或者是 `CalendarEventId`（对于事件）。在重复发生的事件中，`CalendarEventId` 包含重复发生 ID (RID) 以及实际事件 ID，以便单独标识事件的每次发生。

日历 API 使用时间 API 的 `TZDate` 对象（而非标准 JavaScript `Date` 对象）来处理与时区相关的难题，因为 `TZDate` 对象处理确切时间并且提供不同的实用程序方法。

注意：

由于时区和夏令时，针对“今天”的事件可能实际上发生在过去或将来。

日历 API 的主要功能

日历 API 的主要功能包括：

- **日历管理**

要访问某一日历项，您必须首先检索适用类型的 `Calendar` 对象。要访问设备日历，您可以使用：

- `CalendarManager` 接口的 `getDefaultCalendar()` 方法，以便检索默认日历
- `getCalendars()` 方法，以便将所有可用日历作为数组检索
- `CalendarManager` 接口的 `getUnifiedCalendar()` 方法，以便检索合并来自相同类型的所有日历的事件和任务的特殊日历

- **日历项管理**

使用 `Calendar` 接口的适用方法可以管理日历项（将新项添加到日历或者管理单个日历事件）。

要更新或删除重复性事件的单个实例，请首先使用 `CalendarEvent:CalendarItem` 对象的 `expandRecurrence()` 方法获取事件实例的列表。然后，删除适用的事件实例，或者通过调用 `update()` 方法并且将 `updateAllInstances` 参数设置为 `false` 来更新事件实例。

您可以通过使用以下适用的批处理方法来同时创建和管理多个日历项：`addBatch()`、`updateBatch()` 和 `removeBatch()`。该批处理模式提供针对多个日历项的更快、优化的处理。

在搜索日历项时，您可以基于特定的筛选器属性创建属性筛选器、属性范围筛选器和组合筛选器。还可以对搜索结果进行排序。

注意：

批处理模式不提供与操作有关的进度信息。要确保您可以查看进度，请将批处理操作划分为若干更小的批处理操作。例如，将一个处理 100 个更新请求的批处理拆分成一次更新 10 个记录的 10 个批处理操作。拆分某一批处理操作还可以帮助您避免阻止其他数据库操作，例如添加或删除。

• 日历项警报

可以通过使用 `CalendarAlarm` 接口为重要事件（例如，每月会议）或特定任务（例如，支付公用事业帐单）设置警报。该警报将在指定的时间触发，以便向用户提醒该事件或任务。

• 日历更改通知

可以在更改日历项时在您的应用程序中接收通知，将您的应用程序中的日历与用户特定的日历（例如特定网站上的日历）保持同步。

`Calendar` 接口的 `addChangeListener()` 方法注册一个事件侦听器，在 `addChangeListener()` 方法返回该侦听器的订阅标识符时该侦听器以异步方式启动。可以使用 `CalendarChangeCallback` 接口定义用于接收通知的侦听器事件处理程序。

对于日历数据库进行的每个更改都将触发您可为其定义通知的事件。对于批处理模式操作，每个操作仅生成单个事件。重复发生的日历事件作为一个事件处理。

• iCalendar 2.0 格式转换

您可以使用 `CalendarEvent` 对象构造函数和 `CalendarItem` 接口的 `convertToString()` 方法相应地在日历项与 iCalendar 格式之间来回进行转换。

该转换通过使用 `.ics` 扩展名共享文件，允许您在应用程序之间交换日历数据。iCalendar 格式独立于基础传输协议，这意味着可以使用多种协议（包括 HTTP、SMTP 和红外）交换日历项。

iCalendar 格式可用于存储日历项信息并且通过 Internet 交换日历数据。

下面的示例展示了采用 iCalendar 格式的示例事件：

```
BEGIN:VCALENDAR
BEGIN:VEVENT
DTSTART:20110714T150000Z
DTEND:20110715T173000Z
SUMMARY:Team meeting
END:VEVENT
END:VCALENDAR
```

有关日历 API 的详细信息，请参阅 Tizen [日历 API 参考](#)。

向日历添加事件

向默认系统日历添加事件：

1. 使用 `CalendarManager` 接口的 `getDefaultCalendar()` 方法检索默认系统日历。将日历类型“EVENT”指定为参数。

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

```
var calendar = tizen.calendar.getDefaultCalendar("EVENT");
```

2. 创建 CalendarEvent 对象并且定义事件属性:

```
var ev = new tizen.CalendarEvent(
{
  description:"HTML5 Introduction",
  summary:"HTML5 Webinar",
  startDate:new tizen.TZDate(2011, 3, 30, 10, 0),
  duration:new tizen.TimeDuration(1, "HOURS"),
  location:"Huesca",
});
```

3. 要定义重复性事件, 请定义重复性规则。在此示例中, 该事件将在三天中每天重复一次。

```
recurrenceRule:new tizen.CalendarRecurrenceRule("DAILY",
{occurrenceCount:3})
});
```

4. 要设置一个警报以便针对此事件进行提醒, 请使用 CalendarAlarm 接口创建一个警报, 然后将该警报添加到此事件:

```
/* Alarm is triggered with sound 30 minutes before the event's start
time */
var alarm = new tizen.CalendarAlarm(new tizen.TimeDuration
(30, "MINS"), "SOUND");
ev.alarms = [alarm];
```

5. 使用 Calendar 对象的 add() 方法将 CalendarEvent 对象添加到默认日历:

```
calendar.add(ev); /* ev.id attribute is generated */
```

注意:

您可以通过使用 "TASK" 日历类型和 CalendarTask 构造函数按照与上述相同的方式添加任务项。

在批处理模式下向日历添加事件

在批处理模式下将多个事件添加到默认系统日历:

1. 使用 CalendarManager 接口的 getDefaultCalendar() 方法检索默认系统日历:

```
var calendar = tizen.calendar.getDefaultCalendar("EVENT");
```

2. 多要作为数组添加的项:

```
var ev = new tizen.CalendarEvent(
{
  description:"HTML5 Introduction",
  summary:"HTML5 Webinar",
  startDate:new tizen.TZDate(2011, 3, 30, 10, 0),
  duration:new tizen.TimeDuration(1, "HOURS"),
  location:"Huesca"
});
```

注意：

为了保持示例尽可能简单，下面的数组仅包含一个事件。

注意：

`addBatch()` 方法是异步的。通常，应使用同步回调对其成功或失败做出反应。

3. 使用 `Calendar` 对象的 `addBatch()` 方法将数组中的事件添加到日历：

```
calendar.addBatch([ev]);
```

管理事件

管理单个日历事件：

1. 使用 `CalendarManager` 接口的 `getDefaultCalendar()` 方法检索默认系统日历。将日历类型“EVENT”指定为参数。

```
var myCalendar = tizen.calendar.getDefaultCalendar("EVENT");
```

2. 通过使用 `Calendar` 对象的 `find()` 方法检索日历中存储的所有事件：

```
myCalendar.find(eventSearchSuccessCallback)
```

注意：

要检索一组特定的事件，您可以通过 `filter` 和 `sortMode` 参数为搜索操作指定筛选器和排序顺序。

注意：

在该示例中，由于未使用筛选器而检索了所有事件。

3. 更新或删除 `eventSearchSuccessCallback` 事件处理程序内找到的事件。

在该示例中，更改了第一个事件的说明参数，并且使用 `update()` 方法更新了日历中的事件。使用 `remove()` 方法删除了第二个事件。

```
/* Define the event success callback */
function eventSearchSuccessCallback(events)
{
    /* Update the first existing event */
    events[0].description = "New Description";
    myCalendar.update(events[0]);

    /* Delete the second existing event */
    myCalendar.remove(events[1].id);
}
```


在批处理模式下管理多个日历事件

在批处理模式下管理多个日历事件：

1. 使用 CalendarManager 接口的 getDefaultCalendar() 方法检索默认系统日历。将日历类型“EVENT”指定为参数。

```
var myCalendar = tizen.calendar.getDefaultCalendar("EVENT");
```

2. 通过使用 Calendar 对象的 find() 方法检索日历中存储的所有事件：

```
myCalendar.find(eventSearchSuccessCallback);
```

注意：

要检索一组特定的事件，您可以通过 filter 和 sortMode 参数为搜索操作指定筛选器和排序顺序。

注意：

在该示例中，由于未使用筛选器而检索了所有事件。

3. 更新事件：

- a. 在 find() 方法的成功事件处理程序中定义要更新的项：

```
function eventSearchSuccessCallback(events)
{
    events[0].description = "New Description 1";
    events[1].description = "New Description 2";
}
```

- b. 使用 updateBatch() 方法以异步方式更新多个日历项：

```
/* Update the first 2 existing events */
myCalendar.updateBatch(events.slice(0, 2));
}
```

4. 要删除多个事件，请在 find() 方法的成功事件处理程序中使用 removeBatch() 方法以异步方式删除多个日历项：

```
function eventSearchSuccessCallback(events)
{
    /* Delete the first 2 existing events */
    myCalendar.removeBatch([events[0].id, events[1].id]);
}
```

更新重复性日历事件

更新重复性日历事件：

1. 使用 CalendarManager 接口的 getDefaultCalendar() 方法检索默认系统日历。通过指定事件 ID 使用 get() 方法检索日历事件。

```
var calendar = tizen.calendar.getDefaultCalendar("EVENT");
var event = calendar.get(evId);
```

2. 通过使用 `CalendarEvent` 对象的 `expandRecurrence()` 扩展重复性事件以便获取其实例:

```
event.expandRecurrence(new tizen.TZDate(2012, 2, 1),
                      new tizen.TZDate(2012, 2, 15),
                      eventExpandSuccessCB);
```

扩展的事件实例具有自己的 `id.uid` 和 `id.rid` 属性, 其中, `id.uid` 属性对于所有实例而言是相同的。

3. 更新扩展的重复性事件的单个实例。

对于重复性事件, 您可以使用 `update()` 方法的第二个参数确定是更新该事件的单个实例还是所有发生的事件。如果该参数设置为 `true`, 则更新所有实例; 而如果该参数设置为 `false`, 则只更新重复性事件的指示的实例 (基于 `id.rid` 属性)。

在此示例中, 将更新该事件的第二个实例。

```
/* Success event handler */
function eventExpandSuccessCB(events)
{
    events[1].summary = 'updated summary';
    calendar.update(events[1], false);
}
```

接收有关日历更改的通知

在添加、更新或删除日历项时接收通知:

1. 定义所需变量:

```
/* Watcher identifier */
var watcherId = 0;

/* This example assumes that the calendar is initialized */
var calendar;
```

2. 使用 `CalendarChangeCallback` 侦听器接口定义不同通知的事件处理程序:

```
var watcher =
{
    /* When new items are added */
    onitemsadded:function(items)
    {
        console.log(items.length + " items were added");
    },

    /* When items are updated */
    onitemsupdated:function(items)
    {
        console.log(items.length + " items were updated");
    },

    /* When items are deleted */
    onitemsremoved:function(ids)
    {
        console.log(ids.length + " items were removed");
    }
}
```

```
};
```

- 注册侦听器以便使用定义的事件处理程序：

```
watcherId = calendar.addChangeListener(watcher);
```

- 要停止通知，请使用 `removeChangeListener()` 方法：

```
function cancelWatch()
{
    calendar.removeChangeListener(watcherId);
}
```

转换日历项

下面的示例展示如何通过将日历事件转换为 iCalendar 格式和从 iCalendar 格式进行转换，使您的应用程序中的日历项转换更有效：

- 将 iCalendar 字符串转换为日历事件：
 - 使用 `CalendarManager` 接口的 `getDefaultCalendar()` 方法检索默认系统日历。将日历类型“EVENT”指定为参数。

```
var calendar = tizen.calendar.getDefaultCalendar("EVENT");
```

- 从 iCalendar 字符串创建一个新的 `CalendarEvent` 对象并且将其添加到默认日历：

```
try
{
    var ev = new tizen.CalendarEvent
    ("BEGIN:VCALENDAR\r\n" +
     "BEGIN:VEVENT\r\n" +
     "DTSTAMP:19970901T1300Z\r\n" +
     "DTSTART:19970903T163000Z\r\n" +
     "DTEND:19970903T190000Z\r\n" +
     "SUMMARY:Annual Employee Review\r\n" +
     "CATEGORIES:BUSINESS,HUMAN RESOURCES\r\n" +
     "END:VEVENT\r\n" +
     "END:VCALENDAR", "ICALNDAR_20");

    calendar.add(ev);
    console.log('Event added with UID ' + ev.id.uid);
}
```

- 要转换多个 iCalendar 字符串并且将其导入到日历中，请逐个转换这些字符串，然后使用 `addBatch()` 方法在批处理模式下一次添加所有这些事件。
- 将日历事件转换为 iCalendar 格式：
 - 检索默认系统日历并且查找在 `summary` 属性中包含“Tizen”字符串的所有日历项：

```
var myCalendar;

myCalendar = tizen.calendar.getDefaultCalendar("EVENT");

/* Define a filter */
var filter = new tizen.AttributeFilter("summary",
                                       "CONTAINS",
```

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](https://creativecommons.org/licenses/by/3.0/) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](https://opensource.org/licenses/BSD-3-Clause) 获得许可。有关详细信息，请参阅 [内容许可](#)。

```

        "Tizen");

/* Search for the events */
myCalendar.find(eventSearchSuccessCallback, errorCallback, filter);

```

- b. 使用 `convertToString()` 方法在 `find()` 方法的成功事件处理程序中将日历项转换为 `iCalendar` 字符串。

```

function eventSearchSuccessCallback(events)
{
    /* Convert the first event */
    var vevent = events[0].convertToString("ICALNDAR_20");
}

```

- 要从日历导出并转换多个事件，请使用具有适用筛选器的 `find()` 方法查找所需事件，然后逐一转换找到的事件。

联系人

通过联系人 API，您可以管理联系人以及在地址簿中列出的人士。联系人对象始终与特定地址簿相关联。人士对象是与同一个人相关联的一个或多个联系人的聚合。

联系人 API 的主要功能

联系人 API 的主要功能包括：

- **地址簿管理**

要访问某一联系人，您必须首先检索 `AddressBook` 对象。要访问设备地址簿，您可以使用：

- `ContactManager` 接口的 `getDefaultAddressBook()` 方法，以便检索默认地址簿
- `getAddressBooks()` 方法，以便将所有可用地址簿作为数组检索
- `getAddressBook()` 方法，以便检索特定的地址簿

- **联系人管理**

可以通过使用 `AddressBook` 接口的适用方法添加和管理联系人。在一次管理单个联系人时，以同步方式处理操作。

您可以通过使用以下适用的批处理方法来同时创建和管理多个联系人：`addBatch()`、`updateBatch()` 和 `removeBatch()`。该批处理模式提供针对多个联系人的更快、优化的处理。

注意：

批处理模式不提供与操作有关的进度信息。要确保您可以查看进度，请将批处理操作划分为若干更小的批处理操作。例如，将一个处理 100 个更新请求的批处理拆分成一次更新 10 个记录的 10 个批处理操作。拆分某一批处理操作还可以帮助您避免阻止其他数据库操作，例如添加或删除。

在搜索联系人时，您可以基于特定的筛选器属性创建属性筛选器、属性范围筛选器和组合筛选器。还可以对搜索结果进行排序。

- **联系人更改通知**

可以通过在联系人信息变化时在您的应用程序中接收通知，将您的应用程序中的地址簿保持与外部联系人管理器同步。

AddressBook 接口的 addChangeListener() 方法注册事件侦听器。在首次调用该方法后事件订阅自动开始。该方法返回侦听器的订阅标识符。可以使用 AddressBookChangeCallback 接口定义用于接收通知的侦听器事件处理程序。

注意：

作为 addChangeListener() 方法的第一个参数的侦听器对象必须定义了至少一个事件处理程序。如果未定义任何处理程序，则 TypeMismatchError 异常将发生。

对于地址簿进行的每个更改都将触发您可为其定义通知的事件。对于批处理模式操作，每个操作仅生成单个事件。

- **人员管理**

可以通过使用 ContactManager 接口的适用方法管理联系人，包括搜索、更新和删除。在一次管理单个联系人时，以同步方式处理操作。

您可以通过使用以下适用的批处理方法来同时管理多个联系人：updateBatch() 和 removeBatch()。该批处理模式提供针对多个联系人的更快、优化的处理。

在添加联系人或者取消与现有联系人的链接时将自动添加或修改联系人。不能直接添加某个人。

在搜索某个人时，您可以基于特定的筛选器属性对搜索结果进行筛选和排序。

- **vCard 格式转换**

您可以将联系人转换为 vCard 格式或者从 vCard 格式转换回联系人，以便导入和导出联系人。

vCard (RFC 2426) 文件格式 (.vcf 或 .vcard) 是针对电子名片的标准，包含联系信息，例如姓名、地址、电话号码、电子邮件地址、URL、徽标、照片和音频剪辑。

联系人 API 支持 vCard 版本 3.0。

有关联系人 API 的详细信息，请参阅 Tizen [联系人 API 参考](#)。

检索地址簿

检索地址簿：

- 要检索默认地址簿，请使用 ContactManager 接口的 getDefaultAddressBook() 方法。该方法将地址簿作为 AddressBook 对象返回。

```
var myAddressbook;

/* Get the default address book */
myAddressbook = tizen.contact.getDefaultAddressBook();
```

- 要检索所有可用地址簿，请使用 getAddressBooks() 方法。该方法将 AddressBook 对象的数组传递到成功事件处理程序。该数组包含设备上的所有可用地址簿。

```
var addressBooks;

function addressBooksCB(addressBooks)
```

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

```

{
  if (addressBooks.length > 0)
  {
    addressBook = addressBooks[0];
    console.log("The addressbook name is " + addressbook.name);
  }
}

/* Get the list of available address books */
tizen.contact.getAddressBooks(addressBooksCB);

```

检索设备上的所有可用地址簿。

- 如果您已经知道某一地址簿的 ID，则可以使用 `getAddressBook()` 方法检索该特定地址簿。

添加联系人

向默认系统地址簿添加单个联系人：

1. 使用 `ContactManager` 接口的 `getDefaultAddressBook()` 方法检索默认系统地址簿：

```
var addressbook = tizen.contact.getDefaultAddressBook();
```

2. 创建一个 `Contact` 对象并且将其属性定义为实现 `ContactInit` 接口 (`Contact` 构造函数的参数) 的对象。

```

var contact = new tizen.Contact(
{
  name:new tizen.ContactName({firstName:"Jeffrey", lastName:"Hyman"}),
  emails:[new tizen.ContactEmailAddress("user@example.com")]
});

```

3. 使用 `AddressBook` 接口的 `add()` 方法将 `Contact` 对象添加到默认地址簿：

```
addressbook.add(contact);
```

在批处理模式下添加多个联系人

在批处理模式下添加多个联系人：

1. 使用 `ContactManager` 接口的 `getDefaultAddressBook()` 方法检索默认系统地址簿：

```
addressbook = tizen.contact.getDefaultAddressBook();
```

2. 多要作为数组添加的项：

```

var c1 = new tizen.Contact(
{
  name:new tizen.ContactName({firstName:"Jeffrey", lastName:"Hyman"}),
  emails:[new tizen.ContactEmailAddress("user1@example.com")]
});

var c2 = new tizen.Contact(
{
  name:new tizen.ContactName({firstName:"Elton", lastName:"John"}),
  emails:[new tizen.ContactEmailAddress("user2@example.com")]
});

```

```
});
```

3. 使用 AddressBook 接口的 addBatch() 方法将数组中的联系人添加到地址簿:

```
addressbook.addBatch([c1, c2]);
```

注意:

addBatch() 方法是异步的并且应使用回调对其成功或失败做出反应。

管理联系人

管理单个联系人:

- 要检索单个联系人, 请使用 AddressBook 接口的 get() 方法并且使用联系人 ID 作为参数。下面的示例使用 ContactRef 接口的对象检索联系人 ID。

```
/* contactRef is retrieved by other APIs */
var contactRef;
try
{
  /* Retrieve the contact corresponding to the given reference */
  var addressBook = tizen.contact.getAddressBook(
    contactRef.addressBookId);
  var contact = addressBook.get(contactRef.contactId);
}
```

- 更新或删除单个联系人:
 - 使用 ContactManager 接口的 getDefaultAddressBook() 方法检索默认地址簿:

```
var addressbook = tizen.contact.getDefaultAddressBook();
```

- 通过使用 AddressBook 接口的 find() 方法检索在地址簿中存储的联系人:

```
var filter = new tizen.AttributeFilter("name.firstName", "CONTAINS",
  "Chris");
var sortMode = new tizen.SortMode("name.lastName", "ASC");

try
{
  addressbook.find(contactsFoundCB, null, filter, sortMode);
}
```

注意:

要检索某一特定的联系人, 您可以通过 filter 和 sortMode 参数为搜索操作指定筛选器和排序顺序。

与筛选器匹配的联系人按所选排序顺序作为数组传递到注册的成功事件处理程序。

- 更新或删除 contactsFoundCB 事件处理程序内找到的联系人。

在该示例中, 更改了第一个联系人的名字, 并且使用 update() 方法更新了地址簿中的联系人。使用 remove() 方法删除了第二个联系人。

```

/* Define the event success callback */
function contactsFoundCB(contacts)
{
    contacts[0].name.firstName = "Christopher";
    try
    {
        /* Update the first found contact */
        addressbook.update(contacts[0]);

        /* Delete the second found contact */
        addressbook.remove(contacts[1].id);
    }
}

```

在批处理模式下管理多个联系人

在批处理模式下管理多个联系人：

1. 使用 ContactManager 接口的 getDefaultAddressBook() 方法检索默认地址簿：

```
var addressbook = tizen.contact.getDefaultAddressBook();
```

2. 通过使用 AddressBook 接口的 find() 方法检索在地址簿中存储的联系人：

```

var filter = new tizen.AttributeFilter("name.firstName", "CONTAINS",
                                      "Chris");
var sortMode = new tizen.SortMode("name.lastName", "ASC");

try
{
    addressbook.find(contactsFoundCB, null, filter, sortMode);
}

```

3. 要更新联系人，请在 find() 方法的成功事件处理程序中定义要进行的联系人更改：

```

function contactsFoundCB(contacts)
{
    /* Change the first names of all the found contacts */
    for (var i=0; i<contacts.length; i++)
    {
        contacts[i].name.firstName = "Christopher";
    }
}

```

4. 使用 updateBatch() 方法以异步方式更新多个联系人：

```

/* Update all found contacts */
addressbook.updateBatch(contacts);
}

```

5. 要删除多个联系人，请在 find() 方法的成功事件处理程序中使用 removeBatch() 方法以异步方式删除多个联系人：

```

function contactsFoundCB(contacts)
{
    /* Delete the first 2 found contacts */
    addressbook.removeBatch([contacts[0].id, contacts[1].id]);
}

```


接收有关联系人更改的通知

在添加、更新或删除联系人时接收通知：

1. 定义所需变量：

```
/* Watcher identifier */
var watcherId = 0;

/* This example assumes that the address book is initialized */
var addressbook;
```

2. 使用 AddressBookChangeCallback 侦听器接口定义不同通知的事件处理程序：

```
var watcher =
{
  /* When contacts are added */
  oncontactsadded:function(contacts)
  {
    console.log(contacts.length + " contacts were added");
  },

  /* When contacts are updated */
  oncontactsupdated:function(contacts)
  {
    console.log(contacts.length + " contacts were updated");
  },

  /* When contacts are deleted */
  oncontactsremoved:function(ids)
  {
    console.log(ids.length + " contacts were deleted");
  }
};
```

3. 注册侦听器以便使用定义的事件处理程序：

```
watcherId = addressbook.addChangeListener(watcher);
```

4. 要停止通知，请使用 AddressBook 接口的 removeChangeListener() 方法：

```
addressbook.removeChangeListener(watcherId);
```

导入联系人

将联系人导入到默认系统地址簿：

1. 使用 ContactManager 接口的 getDefaultAddressBook() 方法检索默认系统地址簿：

```
var addressbook = tizen.contact.getDefaultAddressBook();
```

2. 从 vCard 字符串创建一个新的 Contact 对象并且将其添加到默认地址簿：

```
var contact = null;

try
{
  contact = new tizen.Contact
```

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。
有关详细信息，请参阅 [内容许可](#)。

```

("BEGIN:VCARD\n"+
"VERSION:3.0\n"+
"N:Gump;Forrest\n"+
"FN:Forrest Gump\n"+
"ORG:Bubba Gump Shrimp Co.\n"+
"TITLE:Shrimp Man\n"+
"TEL;WORK:(111) 555-1212\n"+
"TEL;HOME:(404) 555-1212\n"+
"EMAIL;WORK;PREF:test@example.com\n"+
"END:VCARD");

addressbook.add(contact);
console.log("Contact was added with ID " + contact.id);
}

```

要转换多个字符串并且将其导入到某一地址簿中，请逐个转换这些字符串，然后使用 `AddressBook` 接口的 `addBatch()` 方法在批处理模式下一次添加所有这些联系人。

导出联系人

从默认系统地址簿导出联系人：

1. 使用 `ContactManager` 接口的 `getDefaultAddressBook()` 方法检索默认系统地址簿并且使用 `find()` 方法从该地址簿获取联系人：

```

var addressbook;

var addressbook = tizen.contact.getDefaultAddressBook();

/* Define a filter */
var filter = new tizen.AttributeFilter("name.firstName", "CONTAINS",
"Chris");

/* Search for the contacts */
addressbook.find(contactsFoundCB, errorCallback, filter);

```

2. 在 `find()` 方法的成功事件处理程序中将各联系人转换为 `vCard` 字符串。

在下面的示例中，通过将最先找到的联系人转换为 `vCard` 版本 3.0 格式导出该联系人：

```

function contactsFoundCB(contacts)
{
    /* Convert the first contact */
    var vcard = contacts[0].convertToString("VCARD_30");
}

```

管理人士

管理单个人士：

1. 要检索人士，请使用 ContactManager 接口的 find() 方法：

```
tizen.contact.find(personsFoundCB);
```

2. 更新或删除 personsFoundCB() 事件处理程序中找到的人士。

在该示例中，更改了第一个人士的喜爱的标志，并且使用 update() 方法更新了联系人。使用 remove() 方法删除了第二个人士。

```
/* Define the event success callback */
function personsFoundCB(persons)
{
    persons[0].isFavorite = true;
    /* Update the first found person */
    tizen.contact.update(persons[0]);

    /* Delete the second found person */
    tizen.contact.remove(persons[1].id);
}
```

将多个人士合并为单个人士：

1. 要检索人士，请使用 ContactManager 接口的 find() 方法：

```
tizen.contact.find(personsFoundCB);
```

2. 在 personsFoundCB() 事件处理程序中定义要合并的人士：

```
function personsFoundCB(persons)
{
    var sourcePerson = persons[0];
    var targetPerson = persons[1];
```

3. 使用 link() 方法链接要连接到其他人士的联系人：

```
/* Link 2 persons, contacts from sourcePerson are added to
   targetPerson and sourcePerson is removed */
targetPerson.link(sourcePerson.id);
}
```

消息传递

通过消息传递 API，您可以将消息传递功能用于 SMS、MMS 和电子邮件通信。

消息传递 API 通过提供单步式功能来执行所有与消息传递相关的高级操作，尽量减少您的编码工作。

消息传递 API 的主要功能

消息传递 API 的主要功能包括：

- **消息编写**

您可以使用 Message 对象构造函数编写消息，并且可以使用实现 MessageInit 接口的对象设置消息属性和参数。

您可以通过以下方法将附件添加到您的 MMS 和电子邮件：创建 MessageAttachment 对象，为每个对象定义文件路径和 MIME 类型（image/png、text/pdf 或 text/html），将这些对象以数组形式分配给 Message 对象的 attachments 属性。

要保存消息草稿，请使用 MessageStorage 接口的 addDraftMessage() 方法。

注意：

在首次处理消息时，例如在发送消息或将消息另存为草稿时，系统会将一个唯一的只读消息 ID 分配给各消息。

- **消息发送**

可以使用 MessageService 接口的 sendMessage() 方法发送消息。该方法要求成功和错误事件处理程序。根据发送操作的结果，该消息将移到设备的“已发送邮件”文件夹或“草稿”文件夹，并且还存储于消息存储数据库中。

- **消息管理**

您可以通过使用 MessageStorage 接口提供的以下方法查找、更新和删除已存储的消息：findMessages()、updateMessages() 和 removeMessages()。

在搜索消息时，您可以基于特定的筛选器属性创建属性筛选器、属性范围筛选器和组合筛选器。还可以对搜索结果进行排序。

- **消息存储更改通知**

可以注册事件侦听器以便监视消息存储中的更改、特定转换或特定消息文件夹。

MessageStorage 接口的 addMessagesChangeListener()、addConversationsChangeListener() 和 addFoldersChangeListener() 方法注册一个事件侦听器，在该方法返回侦听器的订阅标识符后该事件侦听器将以异步方式启动。可以使用 MessagesChangeCallback、MessageConversationsChangeCallback 和 MessageFoldersChangeCallback 接口定义用于接收与更改有关的通知的侦听器事件处理程序。

- **服务器同步方法**

可以使用 MessageService 接口的 loadMessageBody() 和 loadMessageAttachment() 方法从电子邮件服务加载电子邮件消息和附件。

要保持您的电子邮件服务帐户最新，您可以使用 sync() 方法将它们与其各自的外部服务器（例如 Gmail 和 Microsoft Exchange）保持同步。还可以使用 syncFolder() 方法只同步一个文件夹，例如收件箱。

可以指定可在各文件夹中检索的消息的最大数目。

有关消息传递 API 的详细信息，请参阅 Tizen [消息传递 API 参考](#)。

发送消息

发送消息：

1. 使用 `getMessageServices()` 方法检索消息传递服务。消息传递服务确定您要处理的消息类型（SMS、MMS 或电子邮件）。

在此示例中，检索 SMS 服务：

```
tizen.messaging.getMessageServices("messaging.sms", serviceListCB,
                                   errorCallback);
```

2. 在 `getMessageServices()` 方法的成功事件处理程序中，使用 `Message` 接口定义消息的内容和属性，然后使用 `MessageService` 接口的 `sendMessage()` 方法发送消息。

如果消息尚未发送就绪，则使用 `MessageStorage` 接口的 `addDraftMessage()` 方法保存消息草稿。

```
function serviceListCB(services)
{
  /* Define SMS message */
  var msg = new tizen.Message("messaging.sms",
  {
    plainBody:"I will arrive in 10 minutes.",
    to:["+346666666666", "+348888888888"]
  });
  /* Assume msgReady was defined */
  if (msgReady) {
    /* Send SMS message */
    services[0].sendMessage(msg, messageSent, messageFailed);
  }
  else
  {
    /* Save a draft */
    services[0].messageStorage.addDraftMessage(msg, successCallback,
                                               errorCallback);
  }
}
```

如果您在发送具有附件的 MMS 或电子邮件，则将附件作为 `MessageAttachment` 对象的数组添加：

```
var msg = new tizen.Message("messaging.email");
msg.attachments = [new tizen.MessageAttachment("images/myimage.png",
                                              "image/png"),
                  new tizen.MessageAttachment("documents/mydoc.pdf",
                                              "text/pdf")];
```

3. 定义消息发送成功事件处理程序，如果成功发送将调用该处理程序。对于电子邮件，成功意味着电子邮件已发送给传递系统，而非电子邮件的最终收件人。对于消息单独发送给每个消息收件人的消息传递技术，例如 SMS，将为每个收件人单独调用成功回调。

```
function messageSent(recipients)
{
  for (var i = 0; i < recipients.length; i++)
  {
    console.log("The SMS has been sent to " + recipients[i]);
  }
}
```

通过定义 `errorCallback`，您可以处理在消息传递失败时会发生的所有可能错误和异常。

管理消息

管理消息：

1. 要从消息存储中检索其发件人是“me”的消息，请将 `MessageStorage` 接口的 `findMessages()` 方法与某一筛选器一起使用：

```
function serviceListCB(services)
{
  emailService = services[0];
  /* Set the attribute filter */
  var filter = new tizen.AttributeFilter("from", "CONTAINS", "me");
  emailService.messageStorage.findMessages(filter, messageArrayCB,
                                          errorCallback);
}

tizen.messaging.getMessageServices("messaging.email", serviceListCB,
                                  errorCallback);
```

`findMessages()` 方法将 `Message` 对象的数组作为搜索结果返回。搜索结果不包含消息的实际正文。要加载消息正文，请使用 `MessageService` 接口的 `loadMessageBody()` 方法。

2. 要更新消息存储中的消息，请使用 `updateMessages()` 方法。该方法使用在步骤 1 中由 `findMessages()` 方法检索的 `Message` 对象的数组。

在此示例中，数组中第一个 `Message` 对象的 `isRead` 属性更新为 `true`：

```
function messageArrayCB(messages)
{
  messages[0].isRead = true;
  emailService.messageStorage.updateMessages(messages, successCallback,
                                             errorCallback);
}
```

3. 要从消息存储中删除消息，请使用 `removeMessages()` 方法：

```
function messageArrayCB(messages)
{
  emailService.messageStorage.removeMessages(messages, successCallback,
                                             errorCallback);
}
```

同步电子邮件

同步电子邮件：

1. 使用 `getMessageServices()` 方法检索消息传递服务：

```
tizen.messaging.getMessageServices("messaging.email", serviceListCB,
                                   errorCallback);
```

2. 使用 `MessageStorage` 接口的 `findMessages()` 方法搜索具有附件的所有电子邮件：

```
service.messageStorage.findMessages(
    new tizen.AttributeFilter("hasAttachment", "EXACTLY", true),
    messageQueryCallback);
```

3. 要加载消息正文，请使用 `MessageService` 接口的 `loadMessageBody()` 方法：

```
/* Success callback for the search operation */
function messageQueryCallback(messages)
{
    for (var i = 0; i < messages.length; i++) {
        var message = messages[i];
        if (!message.body.loaded) {
            tizen.messaging.loadMessageBody(message, successCallback,
                                             errorCallback);
        }
    }
}
```

4. 要下载消息附件，请使用 `loadMessageAttachment()` 方法并且将附件数组（具有有效文件路径）作为参数：

```
tizen.messaging.loadMessageAttachment(message.attachments[0],
                                       successCallback,
                                       errorCallback);
}
}
```

5. 将消息与外部服务器同步：

- 要同步所有帐户文件夹，请使用 `sync()` 方法：

```
/* Synchronize the folders in the success event handler */
function servicesListSuccessCB(services)
{
    services[0].sync(serviceSyncedCB, null, 30);
}
/* Get the email service */
tizen.messaging.getMessageServices("messaging.email",
                                   servicesListSuccessCB);
```

- 要同步特定文件夹，请使用 `syncFolder()` 方法。

在此示例中，只同步“INBOX”类型的文件夹：

```
var emailService; /* Assume email service is initialized */
function serviceCallback(services)
{
    emailService = services[0];
}
```

```
/* Synchronize in the search success event handler */
function folderQueryCallback(folders)
{
    for (var i = 0; i < folders.length; i++) {
        if (folders[i].type === "INBOX") {
            emailService.syncFolder (folders[i], folderSyncedCB, null, 30);
        }
    }
}

/* Get the email service */
tizen.messaging.getMessageServices("messaging.email",
                                   serviceCallback, errorCallback);

/* Search for specific folders */
var filter = new tizen.AttributeFilter("serviceId", "EXISTS");

emailService.messageStorage.findFolders(filter,
                                         folderQueryCallback);
```


接收有关消息存储更改的通知

在添加、更新或删除消息和消息文件夹时接收通知：

1. 定义所需变量：

```
/* Watch identifier */
var watchId;
```

2. 通过实现 MessagesChangeCallback 侦听器接口定义不同通知的事件处理程序：

```
var messageChangeCallback =
{
  /* When messages are updated */
  messagesupdated:function(messages)
  {
    console.log(messages.length + " message(s) updated");
  },

  /* When messages are added */
  messagesadded:function(messages)
  {
    console.log(messages.length + " message(s) added");
  },

  /* When messages are deleted */
  messagesremoved:function(messages)
  {
    console.log(messages.length + " message(s) removed");
  }
};
```

3. 注册侦听器以便使用定义的事件处理程序：

```
watchId = msgService.messageStorage.addMessagesChangeListener(
  messageChangeCallback);
```

4. 要停止接收通知，请使用 MessageStorage 接口的 removeChangeListener() 方法：

```
msgService.messageStorage.removeChangeListener(watchId);
```

注意：

要针对特定会话或消息文件夹中的更改提供通知，请如上所述使用适用方法和事件处理程序。

多媒体

Tizen 使您能够使用内容 API 搜索位于本地设备存储上的内容（图像、视频、音乐或其他）。您还可以执行内容管理任务，例如查看和更新内容属性。

发现内容

Tizen 系统维护一个可用多媒体内容的数据库。该数据库允许快速搜索媒体文件以及将附加信息分配给内容，例如将某一视频剪辑标记为收藏。

通过内容 API，您可以搜索和更新数据库。ContentManager 接口提供 find() 方法以便检索与媒体文件有关的信息。

每个多媒体文件都表示为以下 Content 接口之一的实例：

- AudioContent 表示音频文件。
- ImageContent 表示图像文件。
- VideoContent 表示视频文件。

列出所有音频文件

检索系统中所有音频文件的列表：

1. 定义用于搜索的回调方法。findSuccess() 回调接收 Content 对象的列表并且显示各对象的标题。

```
function findSuccess(items)
{
  console.log("Found " + items.length + " audio tracks:");
  for(var i=0; i<items.length; i++)
  {
    console.log(i.toFixed() + "." + items[i].title + " (" +
      items[i].name + ")");
  }
}
function findError(err)
{
  console.log("Error:" + err.message);
}
```

2. 要检索音频文件，请将 ContentManager 接口的 find() 方法与属性筛选器一起使用以便将搜索限制为仅限音频文件。

find() 方法的第三个参数允许您将搜索限制为单个文件夹。在此示例中，使用 null 值搜索所有文件夹。

```
var audioOnly = new tizen.AttributeFilter("type", "EXACTLY", "AUDIO");

tizen.content.find(findSuccess, findError, null, audioOnly);
```

可用类似方式发现其他类型的媒体内容：

```
var videoOnly = new tizen.AttributeFilter("type", "EXACTLY", "VIDEO");

var imagesOnly = new tizen.AttributeFilter("type", "EXACTLY", "IMAGE");
```

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。有关详细信息，请参阅 [内容许可](#)。

发现新添加的内容

通过内容 API，您还可以接收与内容数据库中的更改有关的通知。例如，要在添加新内容时接收通知，请使用 ContentManager 接口的 `setChangeListener()` 方法设置事件侦听器：

```
tizen.content.setChangeListener(
{
  oncontentadded:function(item)
  {
    console.log("New " + item.type + " detected:" + item.contentURI);
  }
});
```

要停止接收通知，请使用 `unsetChangeListener()` 方法：

```
tizen.content.unsetChangeListener();
```

捕获图像和视频

Web API 未提供对相机设备的直接访问，但您可以通过 Tizen 应用程序管理器启动本机相机应用程序。

通过 Tizen 应用程序管理器，应用程序可以使用其他应用程序执行特定操作，例如使用相机设备捕获图像和视频。您可以使用应用程序 API 访问 Tizen 应用程序管理器。要使用应用程序 API，您的应用程序必须具有在 `confix.xml` 文件中定义的以下权限：

```
https://tizen.org/privilege/application.launch
```

要启动其他应用程序，您必须创建描述您要执行的操作的 `ApplicationControl` 对象。`ApplicationControl` 对象是启动其他应用程序的 `launchAppControl()` 方法所必需的。

您可以通过指定其 ID 运行任何系统应用程序。要启动某一特定的应用程序，请将其 ID 定义为 `launchAppControl()` 方法的第二个参数。您还可以允许系统决定哪一应用程序最适合执行请求的操作。

`launchAppControl()` 方法是异步的，这意味着可通过回调机制提供其结果。将成功和失败回调相应定义为 `launchAppControl()` 方法的第三个参数和第四个参数。在应用程序成功启动时将触发成功回调。否则，将触发失败回调。要接收请求操作的结果，请将回复回调定义为第五个参数。

有关应用程序 API 的详细信息，请参阅 Tizen [应用程序 API 参考](#)。

拍摄照片

使用相机设备拍摄照片：

1. 在 `config.xml` 文件中定义以下权限：

```
<tizen:privilege name="http://tizen.org/privilege/application.launch" />
```

2. 为 `http://tizen.org/appcontrol/operation/create_content` 操作创建 `ApplicationControl` 对象，并且将 MIME 类型设置为 `image/jpg`：

```
var appControl = new tizen.ApplicationControl(
  "http://tizen.org/appcontrol/operation/create_content",
  null,
  "image/jpg");
```

3. 定义成功、错误和回复回调。回复回调必须实现 `ApplicationControlDataArrayReplyCallback` 接口。

```
function successCb()
{
    console.log("Camera application launched successfully");
}
function errCb(err)
{
    console.log("Error:" + err.message);
}
var replyCB =
{
    onsuccess:function(pairs)
    {
        for(var i=0; i<pairs.length; i++)
        {
            if(pairs[i].key ===
                "http://tizen.org/appcontrol/data/selected")
            {
                console.log("picture taken:" + pairs[i].value[0]);
            }
        }
    },
    onfailure:function()
    {
        console.log("FAILED");
    }
};
```

4. 通过调用 `launchAppControl()` 方法请求系统运行相机应用程序：

```
tizen.application.launchAppControl(appControl, null, successCb, errCb,
replyCB)
```

播放音频和视频

Tizen 支持 HTML5 音频和视频元素，可以使用这些元素播放多媒体文件和流且无需单独的查看。

使用 JavaScript，可以使用媒体事件控制播放。用作媒体元素的音频和视频元素将继承 `HTMLMediaElement` 接口的所有属性和方法。

具有 JavaScript 的音频和视频元素的主要功能包括：

- **创建播放器**

您可以创建简单的音频和视频播放器。

- **控制播放**

可以使用 `Media` 对象的 `play()` 和 `pause()` 方法控制媒体文件的播放和暂停。对于媒体事件，可以使用附加功能。

- **检索持续时间和播放时间**

如果加载了其元数据（例如播放时间、持续时间以及视频宽度和高度），则您可以检索媒体文件的持续时间和播放时间。

- **从随机位置播放**

可以通过从随机位置播放媒体文件，指示播放时间。为此，您必须更改 Media 对象的 `currentTime` 属性值以便触发 `timeupdate` 事件。

- **检索进度状态**

您可以使用 `Progress` 媒体事件检索和显示下载进度状态，在更新与加载媒体内容的媒体对象的进度相关的信息时将触发该媒体事件。

- **查看支持的媒体格式**

可以通过使用 `canPlayType()` 方法查看是否可以播放媒体数据。必须在 Web 服务器中以 Tizen 中支持的格式设置 MIME 类型。如果使用了非支持的 MIME 类型，您可以提前处理异常。

Tizen 支持以下媒体编解码器：

- 对于音频：AAC、AMR-NB、AMR-WB、MP3、Vorbis
- 对于视频：H.263、H.264、MPEG-4、Theora

有关 HTML5 音频和视频元素的详细信息，请参阅 [W3C HTML5 规范](#)。

创建音频和视频播放器

为流式传输音频和视频创建一个简单的 HTML5 播放器：

1. 要创建音频播放器，请创建包含必需属性的音频元素：

```
<audio id="audio" src="media/audio_sample.mp3"
      preload="auto" controls>
</audio>
```

2. 要创建视频播放器，请创建包含必需属性的视频元素：除了可用于 audio 元素的属性之外，还可以使用 `width`、`height` 和 `poster` 属性。

```
<video id="video" src="media/video_sample.mp4"
       width="400" height="220" poster="media/poster_sample.png"
       preload="auto" controls>
</video>
```

将创建具有播放控件（在浏览器中提供的内置控件）的播放器。仅当添加了 `controls` 属性时该控件才可见。如果未定义 `poster` 属性，则在播放前在屏幕上将显示视频的第一个框架。

注意：

`preload` 属性默认设置为 `auto`，这意味着将自动加载媒体元数据。如果不想加载元数据，则将属性值设置为 `metadata` 或 `none`。

注意：

在使用自动播放功能（无需用户交互即自动播放内容）前要仔细考虑用户体验。在移动网络中，用户可能招致意外的 Internet 数据包费用或干扰因素，例如播放意外停止。

播放媒体文件

使用自定义控件播放和暂停媒体文件：

1. 创建用于控制播放和暂停操作的视频元素和按钮：

```
<div class="media">
  <video id="video" src="media/video_sample.mp4"></video>
  <div>
    <button id="v-play" type="button">play</button>
    <button id="v-pause" type="button" disabled>pause</button>
  </div>
</div>
```

在播放事件发生前，**Pause** 按钮将被禁用。

2. 定义按钮功能。使用 HTMLMediaElement 接口的 play() 和 pause() 方法播放和暂停媒体文件。

```
<script>
  var play_button = document.getElementById("v-play");
  var pause_button = document.getElementById("v-pause");
  var video = document.getElementById("video");

  play_button.addEventListener("click", function()
  {
    video.play(); /* Play movie */
  }, false);

  pause_button.addEventListener("click", function()
  {
    video.pause(); /* Pause movie */
  }, false);

  video.addEventListener("play", function()
  {
    pause_button.disabled = false;
  }
  )
```

流式传输多媒体

您可以从本地设备（例如摄像机和麦克风）访问多媒体流。该功能基于 W3C `getUserMedia()` 规范。`getUserMedia()` 规范尚在制订中并且没有完全标准化。许多浏览器项目实现了该规范的自己的版本。Tizen 平台使用 WebKit 并且通过 `webkitGetUserMedia()` 方法提供对本地多媒体流的访问。

注意：

`getUserMedia()` 规范的实现是非常尖端的技术。本节中论述的所有 API 或功能在该规范的将来版本中很可能会更改，并且在 Tizen 的更高版本中可能会以不同方式工作。

有关媒体流的详细信息，请参阅 [W3C 媒体捕获和流规范](#)。

该规范定义一个 URL 接口，该接口提供 `createObjectURL()` 方法，您可以使用该方法从媒体流对象创建 Blob URL。与 `getUserMedia()` 方法一样，WebKit 项目实现其自己版本的 `createObjectURL()` 方法并且提供包含该 `createObjectURL()` 方法的 `webkitURL` 接口。

`createObjectURL()` 方法采用 `webkitGetUserMedia()` 方法生成的媒体流对象，并且创建适合作为视频元素的源属性的输入的 URL。

对来自相机设备的视频进行流式处理

访问视频流：

1. 创建 HTML5 视频元素和按钮以便用于控制视频流访问：

```
<body>
  <video id="videoPlay" src="" autoplay controls></video>
  <br />
  <input type="button" value="START" onclick="getVideoStream();"
        id="btnStart">
</body>
```

2. 使用 `navigator.webkitGetUserMedia()` 方法访问视频流：

```
<script>
  function getVideoStream()
  {
    navigator.webkitGetUserMedia({video:true}, successCallback,
                                errorCallback);
  }
</script>
```

第一个参数是必需的并且分配 JSON 对象以便确定要使用的媒体元素（音频或视频）。

系统会要求用户提供权限，以便提供对相机视频流的应用程序访问权限。

3. 检索视频流信息、创建流 URL 并且将视频附加到视频元素：

```
<script>
  function successCallback(stream)
  {
    var URL = window.webkitURL;
    document.getElementById("videoPlay").src =
      URL.createObjectURL(stream);
  }
</script>
```

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](#) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](#) 获得许可。有关详细信息，请参阅 [内容许可](#)。

NFC

通过 NFC API，您可以使用近场通信 (NFC) 服务在 NFC 设备（“对等方”）或标记之间交换数据。这些设备可以共享联系人、照片和视频，并且还可以充当智能卡。您可以使用 NFC 设备针对多种行为（例如支付帐单或下载优惠券）与 NFC 标记进行通信。

与其他短范围通信技术相比，NFC 提供以下优势：

- 更快设置
- 更低功耗
- 无设备配对要求
- 减少了不想要的中断

NFC 标记是可以安全地存储个人信息（例如借记卡号码或联系人详细信息）的芯片。您可以使用 `NFCTag` 接口的方法访问 NFC 标记以便读取或写入信息。NFC 标记类型是使用 `NFCTagType` 类型定义的类型属性标识的。

注意：

Tizen 支持以下 NFC 标记类型：GENERIC_TARGET、ISO14443_A、ISO14443_4A、ISO14443_3A、MIFARE_MINI、MIFARE_1K、MIFARE_4K、MIFARE_ULTRA、MIFARE_DESFIRE、ISO14443_B、ISO14443_4B、ISO14443_BPRIME、FELICA、JEWEL 和 ISO15693。

NFC 论坛定义了 NFC 数据交换格式 (NDEF)，以便封装两个 NFC 设备之间或 NFC 设备和 NFC 标记之间交换的数据。NDEF 消息可以存储不同格式的数据，例如文本、MIME 类型对象或者基于 RTD（记录类型定义）规范的超短记录。NFC 标记使用 NDEF 来交换消息。

NFC API 的主要功能

NFC API 的主要功能包括：

- **NFC 设备管理**

可以通过启用或禁用 NFC 服务来管理 NFC 连接性。

要使用 NFC，请使用 `NFCManager` 接口的 `getDefaultAdapter()` 方法检索默认 NFC 适配器。可以使用 `setPowered()` 方法启用和禁用 NFC。

- **NFC 标记和对等方检测**

要在检测到 NFC 标记或对等设备时接收通知，请使用 `NFCAdapter` 接口的 `setTagListener()` 和 `setPeerListener()` 方法。这些方法注册事件侦听器，事件侦听器将在检测到 NFC 标记或对等设备时相应触发通知。可以使用 `NFCTagDetectCallback` 和 `NFCPeerDetectCallback` 接口定义事件处理程序，以便相应接收有关附加和断开 NFC 标记和对等方的通知。

- **NDEF 消息操作**

可以通过以下方式处理 NDEF 消息：首先创建 NDEF 记录并且使用 `NDEFRecord` 构造函数（还可以使用 `NDEFRecordText`、`NDEFRecordURI` 和 `NDEFRecordMedia` 接口），然后使用 `NDEFMessage` 接口的记录属性将记录添加到 NDEF 消息。

• NDEF 数据交换

可以在标记和对等方之间交换 NDEF 数据。要交换标记之间的数据，请使用 NFCTag 接口的 readNDEF() 和 writeNDEF() 方法。

要在对等方之间交换数据，请使用 sendNDEF() 方法发送消息，并且使用 NFCPeer 接口的 setReceiveNDEFListener() 方法触发一个事件侦听器，在从对等方接收 NDEF 消息时该事件侦听器将触发一个事件。

可以使用 NDEFMessageReadCallback 接口定义事件处理程序，以便从标记和对等设备读取 NDEF 消息。

注意：

如果某一应用程序处于后台（隐藏的）并且使用 writeNDEF()、transceive() 或 sendNDEF() 方法，将启动错误回调。只能在应用程序可见时使用这些方法。

该 NFC 服务可以使用应用程序控制功能基于 NDEF 消息内容启动 NFC 应用程序。如果应用程序设计为以此方式启动，则它必须具有在 config.xml 文件中定义的以下操作：

<http://tizen.org/appcontrol/operation/nfc/wellknown>

在 NFC 设备（已开机）读取 NFC 标记或接收 NDEF 消息（其第一个记录 (NDEFRecord) 的记录类型设置为 NFC_RECORD_TNF_WELL_KNOWN）时，将启动 NFC 应用程序。

还可以通过卡模拟交易启动 NFC 应用程序。NFC 设备可使用卡模拟功能与销售点 (POS) 诊断进行通信，以便执行付款等功能。如果在 config.xml 文件中定义了应用程序控制以及

<http://tizen.org/appcontrol/operation/nfc/transaction> 操作，并且发生了卡模拟功能导致的交易，则启动 NFC 应用程序。

下表列出了 NFC 操作、架构和 MIME 类型。

表 9：NFC 操作

操作	架构	MIME
http://tizen.org/appcontrol/operation/nfc/empty	NULL	NULL
http://tizen.org/appcontrol/operation/nfc/wellknown	<scheme>:<host>/<path> URL 示例： <ul style="list-style-type: none"> • http • http://tizen.org/ • http://tizen.org/about/devices • http://tizen.org/about/* URN 示例： <ul style="list-style-type: none"> • tel • mailto • mailto:tommy@tizen.org <protocol_code> 和 <scheme> 必须匹配。有关详细信息，请参阅 NFC 论坛上的 NFCForum-TS-RTD_URI_1.0 和 NFC RTD（记录类	U/<protocol_code> 例如： <ul style="list-style-type: none"> • U/0x03 • U/0x05 • U/*

除非另有注明，否则，本文中的内容（代码示例除外）基于 [Creative Commons Attribution 3.0](https://creativecommons.org/licenses/by/3.0/) 获得许可，而本文中包含的所有代码示例都基于 [BSD-3 条款](https://creativecommons.org/licenses/by/3.0/) 获得许可。有关详细信息，请参阅 [内容许可](#)。

	型定义) 文档。	
	NULL	<code><type_string>/*</code> 例如: <ul style="list-style-type: none"> • <code>sp/*</code> • <code>T/*</code> • <code>*/*</code>
<code>http://tizen.org/appcontrol/operation/nfc/mime</code>	NULL	<code><type_string>/<subtype_string></code> (不区分大小写) 例如: <ul style="list-style-type: none"> • <code>text/x-vard</code> • <code>text/*</code> • <code>*/*</code>
<code>http://tizen.org/appcontrol/operation/nfc/uri</code>	<code><uri></code> 例如: <ul style="list-style-type: none"> • <code>http://tizen.org/about/devices</code> 	NULL
<code>http://tizen.org/appcontrol/operation/nfc/external</code>	<code><scheme>:<string></code> (不区分大小写) 例如: <ul style="list-style-type: none"> • <code>nfc:ext.tizen.org.ABC</code> 	NULL
<code>http://tizen.org/appcontrol/operation/nfc/transaction</code>	<code>nfc://secure/<SE name>/aid/<aid></code> 例如: <ul style="list-style-type: none"> • <code>nfc://secure/SIM1/aid/123456789</code> • <code>nfc://secure/SIM1/aid/1234*</code> • <code>nfc://secure/SIM1/aid/*</code> 	NULL

有关 NFC API 的详细信息，请参阅 Tizen [NFC API 参考](#)。

管理 NFC 连接性

管理 NFC 连接性：

1. 要访问 NFC 适配器，请使用 `getDefaultAdapter()` 方法：

```
var nfcAdapter = tizen.nfc.getDefaultAdapter();
```

2. 要启用 NFC，请使用 `NFCAdapter` 接口的 `setPowered()` 方法并且将第一个参数设置为 `true`：

```
nfcAdapter.setPowered(true, onPowerOn, onPowerOnFails);
```

3. 要禁用 NFC，请使用 `setPowered()` 方法并且将第一个参数设置为 `false`：

```
nfcAdapter.setPowered(false);
```

检测 NFC 标记和对等设备

检测 NFC 标记和对等设备：

1. 要访问 NFC 适配器，请使用 `getDefaultAdapter()` 方法：

```
var nfcAdapter = tizen.nfc.getDefaultAdapter();
```

2. 使用 `NFCtagDetectCallback` 侦听器接口为 NFC 标记检测定义事件处理程序：

```
var setTagDetect =
{
  /* When an NFC tag is detected */
  onattach:function(nfcTag)
  {
    console.log("NFC Tag detected.Its type is:" + nfcTag.type);
  }

  /* When an NFC tag becomes unavailable */
  ondetach:function()
  {
    console.log("NFC Tag unavailable");
  }
}
```

3. 注册侦听器以便使用定义的事件处理程序。

您可以通过将标记类型定义为 `setTagListener()` 方法的第二个参数，将侦听器限制为仅检测特定的 NFC 标记类型。在此示例中，仅检测 MIFARE 标记：

```
/* Defines the tag types to be detected */
var tagFilter = ["MIFARE_MINI", "MIFARE_1K", "MIFARE_4K",
                "MIFARE_ULTRA", "MIFARE_DESFIRE"];

/* Registers the event listener */
nfcAdapter.setTagListener(setTagDetect, tagFilter);
```

4. 要停止标记检测，请使用 `unsetTagListener()` 方法：

```
nfcAdapter.unsetTagListener();
```

NFC 对等方的检测方式类似于 NFC 标记，只是 `setPeerListener()` 方法用于注册 `NFCPeerDetectCallback` 侦听器接口，`unsetPeerListener()` 方法用于停止对等方检测。

处理 NDEF 消息

处理 NDEF 消息：

1. 要创建 NDEF URI 记录，请创建 NDEFRecordURI 接口实例并且指定 URI 参数：

```
var newRecord = new tizen.NDEFRecordURI("https://www.tizen.org/");
```

还可以基于要创建的记录类型创建 NDEFRecord、NDEFRecordText 和 NDEFRecordMedia 接口的实例。

2. 创建 NDEFMessage 接口实例：

```
var newMessage = new tizen.NDEFMessage();
```

3. 要将 NDEF 记录添加到 NDEF 消息，请使用 NDEFMessage 接口的 records 属性：

```
newMessage.records[0] = newRecord;
```

与 NFC 标记交换 NDEF 数据

与 NFC 标记交换 NDEF 数据：

1. 要从 NFC 标记读取数据，请使用 NFCTag 接口的 readNDEF() 方法。该方法检索在 NFC 标记上存储的 NDEF 消息并且将这些消息传递到 NDEFMessageReadCallback 侦听器。

```
/* NDEFMessageReadCallback listener */
function readMessage(message)
{
    console.log("Record Count is " + message.recordCount);
}

/* Check whether the NFC tag supports NDEF format */
if (Tag.isSupportedNDEF)
{
    /* Read NDEF data */
    Tag.readNDEF(readMessage);
}
```

2. 要将数据写入 NFC 标记，请使用 writeNDEF() 方法：

```
var newMessage = new tizen.NDEFMessage();
function writeCallback()
{
    console.log("Success!");
}
Tag.writeNDEF(newMessage, writeCallback);
```

3. 可以使用 transceive() 方法将原始数据以字节数组的形式传递到 NFC 标记。但是，将数据发送到 NFC 标记的方法要求了解该标记的基本细节。

与对等设备交换 NDEF 数据

与对等设备交换 NDEF 数据：

1. 要从对等设备接收 NDEF 消息，请使用 NFCPeer 接口的 `setReceiveNDEFListener()` 方法。该方法注册 `NDEFMessageReadCallback` 侦听器接口，在从对等设备读取 NDEF 消息时将调用该侦听器接口。

```
/* NDEFMessageReadCallback listener */
function readMessage(message)
{
    console.log("Record Count is " + message.recordCount);
}

/* Set a listener to receive an NDEF message */
peer.setReceiveNDEFListener(readMessage);
```

2. 要将 NDEF 消息发送到对等设备，请使用 `sendNDEF()` 方法：

```
var newMessage = new tizen.NDEFMessage();
peer.sendNDEF(newMessage);
```