Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Development and automated testing in Firefox OS environment

Szakdolgozat

Készítette:

Sánta Gergely

gazdaságinformatikus szakos hallgató

Témavezető:

Lengyel Zsolt

Szeged

2014

Tartalomjegyzék

Feladatkiírás

The goal of the thesis is to design an effective development methodology for Firefox OS, with great emphasis on the automated testing and debugging tasks and opportunities occuring at various stages of the development. During the selection of the various tools preference should be given to already proven and mature web development technologies (since this is the main goal of Firefox OS), nonetheless the specific requirements and toolkit of the new platform should also be presented. The previously described items should be utilized with an example application besides the theoretical implementation.

Tartalmi összefoglaló

Introduction

Firefox OS is one of the latest operating systems for mobile phones and tablets. It was released in February 2012 by Mozilla as an open-source project based on a Linux kernel code named as boot to gecko (B2G).

One important concept behind Firefox OS is to provide full smartphone experience at an affordable price targeting primarily the developing countries. In many of them Firefox OS is already present (including Hungary), and the expansion continues this year, with more device manufacturer (inclding ZTE) shipping their mobile phones with Firefox OS besides android.

The other major idea of the OS is the broad development community it intends to target. It uses open tools used throughout the web like javascript and HTML5, enabling the use of millions of already existing web application with little or no modification. More and more companies needs to be present on the mobile app market in addition to the web, spending an increasing amount of mony for application development to support the various mobile platforms. There is an emerging new era of mobile app development, due to the sophisticated mobile browsers of today, where it is possible to use full screen sized web applications, which look and work almost like a native app. These modern browsers have built in support for native APIs such as camera and GPS. Firefox OS successfully sensed this new trend making the environment of the web as their native environment. It is even possible to install apps from the Firefox OS marketplace in desktop or android environmentis, that work similarly as native applications.

However the many advantages of the platform does not make it automatically a market success. With the release of 2.2 there are still performance issues and unexpected system crashes, that can ruin the platforms reputation, since in some countries it is already available for production. The other problem is the lack of applications in the Firefox OS marketplace. Despite the relative ease of app development for this paltform, and the huge community of potential developers, it is hard to make the software vendors port their apps, without a significant market share.

In my thesis I focus on the development and testing of Firefox OS application. The

motive behind this is to contribute to the development of this promising platform by collecting the best of the tools and practices, the potential developers need. Most of these tools are already widely used in other areas of software engineering (mainly in web development), but the lack of documentation, experience, tutorials and other resources make it difficult for the developers to utilize their existing experience. Some other tools are were created specifically to suppport development for the Firefox OS platform. These tools are very young and change quickly so I try to empahsize the general idea behind them rather than the exact usage of each one. One of the biggest and most important part of my thesis will explore and explain the process of automated testing, which can help to produce and maintain high quality applications.

Throughout my thesis I demonstrate the concepts and tools through a real app I develped. This is a bycicle application, where the bycicle routes are created and maintained by the community. The users adds the routes' points to the map, and other users can rate these, so that to be able to maintain a high quality route system. The application is by no means production quality, but good enough for demonstration purposes and to bring the concepts closer to the user.

My goal was to make my research more digestible for anyone interested in this topic. Every chapter starts with a theoretical part where I iterate over the possible solutions for the given subject and try to chose the most appropriate one based on usability, popularity, and other characteristics of the actual situation. After that I dig into the chosen tool more deeply to get the reader more familiar with it, and to show how this tool can solve the present problem. At the end of the chapter I guide the reader through a concrete use case with the help of the application I developed.

1. fejezet

Introduction to Firefox OS

1.1. The Firefox OS ecosystem

Firefox OS is an open source operating system aiming mobile and tablet devices. It is based on a Linux kernel and oficially named as Boot to Gecko (B2G). The main idea behind Fireox OS is to embrace the ever expanding ecosystem of the open web, and to enable the use of these tools throughout the whole application development process. One of the hottest topics of the web today is HTML5, which encompasses open web standards like HTML, CSS and Javascript. Firefox OS is built with these in mind, by developing Web APIs so that HTML5 applications can communicate with the device's hardver. It works like a web browser running on a lightweight Linux kernel, where every application is a standalone web app, even the native ones like the Camera or the Phone.

The User Interface of Firefox OS looks a lot like Android, with a lock screen, home screen and notification area. The deletion and closing of application works also similarly. However as it is getting more mature, it's own characteristics have started to appear. For example it does not use the paging structure of android for the apps, but rather a vertical scrollbar similar to the solution of modern web applications for handling it's contents. This infinite scrolling style does not olny charachterize the home screen, but it is utilized by also the applications, making the look and feel smooth and consistent throuthout the different views of the phone.

The platform differentiates two kinds of applications. Both of them are distributed through the Firefox Marketplace. Hosted apps resides on a server and work similar to traditional web pages. The only difference is the app manifest in the app's root directory, which provides important details about the app, such as the path of the main html file and the name of the application. This information is needed to be able to install the application

with a native-like procedure. There are some security constraints concerning hosted apps that does not enable them to use privileged and certified APIs. If an app needs greater controll over the phones resources, it should be distributed as a packaged app. A packaged app containes all of its resources in a zip file, with the app manifast at the root directory. Packaged apps can be further divided into three categories: web app, privileged app and certified app, going from the most restrictive one to the digitally signed categories, that enable the use of priviledged and certified APIs. Privileged apps are signed as part of the Marketplace review process, while certified apps are signed by device manufacturers. Packaged apps tend to open up more quickly than hosted apps, because all of their resources are stored locally, making the user experience more smooth.

Considering the applications, the Firefox OS ecosystem does not stay within the limits of devices running Firefox OS. The notions of Open Web Apps for Android and Open Web Apps for Desktop enable Marketplace apps to be installed into Android and Windows respectively. These are executed by Web Runtime, a compontent of the given platform's Firefox based browser.

1.2. The architecture of Firefox OS

This section reveals how Firefox OS builds up, from bottom to top, explaining each layer more deeply, because an application developer's main concerns connect to layers in higher levels, where the app interacts with the system.

The lowest level operating system, named Gonk, encompasses the Linux kernel and other hardware related layers. It is a porting target of Gecko, which means, there is a port of Gecko to Gonk, just like there is a port of Gecko to Android or Windows. One difference is that Gonk is part of the Firefox OS project, so it is more lenient when it comes to Gecko, exposing direct access to the full telephony stack and display frame buffer.

The next layer is the application runtime, named Gecko. It is actually the web browser engine used in many applications developed by Mozilla. It supports open web standards like HTML, CSS, and Javascript, and makes sure those APIs work well on every operating system it supports. Gecko includes, among other things, a networking stack, graphics stack and a Javascript virtual machine.

At the top of the hierarchy resides the application layer, that contains the various kinds of apps, javascript libraries and Gaia. Gaia, written entirely in HTML, CSS and Javascript, is the platform's user interface and contains all the system and certified apps. It communicates with the underlying operating system through standard Web APIs, implemented by

Gecko. The separation of Gaia from the system specific components enables it to run on other operating systems and web browsers, widening the potential market Firefox OS apps can reach. More information about how Gaia handles and stuctures apps can be found in the next section.

(ide egy kisebb képet beszúrni a Firefox OS felépítéséről)

1.2.1. How Gaia handles custom apps

To imitate the behaviour of desktop browsers, where webpages live in different windows or tabs, Firefox OS opens each app in a new iframe. It has a lot of security adventages because it sandboxes the apps. This means each app has access onlty to it's own resources (cookies, IndexedDB, etc.). A practical example can be the case when the user logs in to Facebook in App A, which has no effect on App B's ability to interact with the user's account on Facebook. The life cycle of an app is the following:

(ide egy képet beszúrni az alkalmazást tartalmazó iframe html kódjáról)

1.3. Development tools for Firefox OS

There are multiple possibilities for developers to try their Firefox OS apps during development.

1.3.1. Firefox OS Simulator

The simplest solution is the Firefox OS simulator developed by Mozilla and available as an add-on in the desktop browser. It simulates the higher layers of Firefox OS, and makes it easy for developers to test and verify their apps behaviour quickly, without a device. It runs in a window the same size as a Firefox OS device, includes the Firefox OS user interface and built-in apps, and simulates many of the Firefox OS device APIs.

After installing one of the simulator add-ons (there are multiple simulators corresponding to the version of Firefox OS), it appears in the Firefox App Manager. The app manager can be opened by typing in the URL bar the following: about:app-manager. At the bottom of the page appears the Start Simulator button, clicking it reveals the installed simulators and gives the option to install other versions of it. Choosing one of them and clicking it opens the simulator in a new window. The apps tab of the App Manager can be used to add packaged or hosted apps to the App Manager, that can be later uploaded to the simulator with the update button of the actual app's window. The debug button does the

same, but it also connects a toolbox to the app, allowing to debug its code directly with the usual debugging tools the Firefox browser offers.

From Firefox 33 onwards Firefox includes a tool called WebIDE, whose purpose is to replace the App Manager. It's functionality is similar to App Manager's, but it also provides an editing environment for developers to create Firefox OS apps. Since the App Manager was more stable at the time of this writing, I use it throughout my thesis instead of the WebIde.

(ide egy képet beszúrni az App Manager-ről)

1.3.2. Firefox OS desktop build

The Firefox OS desktop client (B2G desktop cliend) can be used to run Web apps in a Gecko-based environment. It does not emulate device hardware, but can be useful during the development process for quick tests and user interface verification. As I noticed through my research the Desktop Build is the least popular of the possible to run Firefox apps, and mainly used by Gaia developers working on the Gaia user interface. However it has a big advantage over the simulator and the real device: it containes a pre-installed marionette server, that can be used to run user interface tests. This topic is covered deeply at Chapter 3.

Adding a custom app to the desktop build is difficult, because the developer has to package the app's files himself, copy it to the appropriate location and put the app's information to a configuration json file. However from February 2014 it is possible to run a custom B2G binary (a.k.a desktop build) and/or a custom Gaia profile from the simulator, making app uploading easy thorugh the App Manager, or WebIDE. It can be done by typing about:addons to the URL bar. The Extensions tab lists the installed simulator, clicking on the Preferences button of the chosen simulator, it jumps to a page where the path of the custom B2G or Gaia profile can be selected. From then on, when running the simulator it uses the given B2G or Gaia profile instead of the built-in one, enabling to run user interface tests via marionette. I had connection problems to B2G, when trying to run it from the App Manager and WebIDE on my Ubuntu machine, so in Chapter 3, where I describe marionette tests, I do not use this solution and add my app to the desktop build by hand.

1.3.3. Running apps on a real device

The simulator and destop build are good for running quick tests, but they lack some hardware functionlity, that makes it necessarry for developers to try their apps from time to

time on a real device. Fortunately it is very easy to do. On the device Remote debugging has to be checked, which is located in Device Information > More Information > Developer. The device communicates with the computer through the Android Debug Bridge, which means on Ubuntu the ADB Helper add-on needs to be installed. From then on the attached device appears beside the Start Simulator button in App Manager or WebIDE, and can be used the same way as the simulator.

Marionette is available in all Firefox OS builds, but manufacturers disable it in their image files. To run user interface tests in the same way when using a desktop build, the user has to build Firefox OS himself. It is not an easy process, and I will describe it briefly in chapter 3, where I iterate through the options of running automated tests.