

Index

[Introduction](#_i0udy0g4lw9u)

[Sprocket](#_rk7zkc7y3h5)

[Milestones](#_sntjs7jhej7)

[Sprocket in details](#_bg5yqfso0s8x)

[App](#_n1h1vcy3ie6)

[Browser](#_8bajg0lilje5)

[Renderer](#_t1l0ty2qfjeq)

[Common](#_etgorfg31iyk)

[Android](#_eog5hq46r41m)

[Todo](#_w1oul2qfax2u)

[Child](#_lmnryo5cp3xj)

[Utility](#_fxevy88f2w7b)

[Sandbox](#_6n7s20k4rcn6)

[Tab support (Linux)](#_8i03glwwh5oe)

# Introduction

There are multiple ways to build a web browser. You don’t need to do everything from scratch, but if you want to do that you should check [this tutorial](http://limpet.net/mbrubeck/2014/08/08/toy-layout-engine-1.html) out first. For those who would like to follow the beaten path, the open-source communities have already provided several high quality browser engines: [WebKit (Safari)](https://www.webkit.org/), [Blink (Chromium)](http://www.chromium.org/Home), [Gecko (Firefox)](https://developer.mozilla.org/en-US/docs/Mozilla/Gecko), [Servo](https://github.com/servo/servo) (Mozilla’s experimental project). You can even read an overview about their architecture design [here](http://taligarsiel.com/Projects/howbrowserswork1.htm).

There are some other popular “out-of-the-box” solutions: [CEF](https://code.google.com/p/chromiumembedded/), [NW.js (previously node-webkit)](http://nwjs.io/), [Awesomium](http://www.awesomium.com/), [OpenFin](https://openfin.co/), [QtWebEngine](http://qt-project.org/wiki/QtWebEngine). They are all based on Chromium’s framework, which is the Content API. Each of them are using their own UI, callbacks and ecosystem to support different purposes. If you find here what you are looking for, stick with it. Otherwise we can show the capacity of the Content API to build up your custom ecosystem.

The [Yandex Browser](https://browser.yandex.com/future/) (and of course [Google Chrome](https://www.google.com/chrome/)) could be a great example, what you can achieve with the Content API.

For the following tutorial we will use the Content API to build our browser, like Chrome does, but only with basic functionality.

You can read about the Chromium Content API and modules here:

[Content API](https://www.chromium.org/developers/content-module/content-api), [Content Module](https://www.chromium.org/developers/content-module)

If you are interested in how Chromium is built together and how it works, you can check out our [Chromium Architecture overview](https://docs.google.com/document/d/1QXPBeg-0Vm5xQFDybEXgIubg-3AUgD0-doXBAo9r6os).

# Sprocket

There are lots of very good standalone and embedded browsers on the market, but each one of them is connected to and based on their separate ecosystems. In general it is a good strategy to join an already existing system, but sometimes there is a need to build up a totally different one. Our guide is about to show how to do this, “How to build a Web Browser”.

In the beginning we started with ideas from already existing embedder implementations. Although the concept of already existing browsers are fine, we felt our hands were tied. So, we dropped everything and went to the beginning to build up our ideas from scratch. This is the way how Sprocket was born.

Sprocket has three branches currently: master, core and testing. Core Sprocket is about to be as minimal as possible compared to the existing implementations. In this branch the code size is reduced to a minimum, and unnecessary requirements were dropped. Now - after the bases have been done - we are continuously extending Sprocket with well-documented new features. You can read more about our branches later in this documentation.

For those who cannot wait, the source code can be found on [GitHub](https://github.com/szeged/sprocket). You will also see a guide there about how to compile and execute the browser. You can find some sneak peek pictures [here](https://drive.google.com/folderview?id=0B4-vJLnNbTOLfmtmWENaTEd0STBfZTcxS0ZYNnQtTXd5WktXX0Vtb01ySENnWS1QVE10clE&usp=sharing).

# Milestones

Our project has just started and has a long way to go. So, we collected our goals and broke them down into milestones.

|  |  |  |  |
| --- | --- | --- | --- |
| **Milestone** | **Feature Name** | **Description** | **Status** |
| M0 | Core codebase | A stable browser based on Content API. Supported platform: Linux (PC and ARM) | Done |
| Android Port | Extend the platform support with Android. | Done |
| Documentation | Document the source code and make a proper documentation about how it works. | Done |
| M1 | Tabs | Tab support on Linux | Done |
| Tab support on Android | In progress |
| Sandbox | Sandbox support. | Done |
| Error Page | Add support for error pages. | Not implemented |
| Dialogs #1 | Add javascript dialog support (alert, confirm, prompt…). | Done |
| Dialogs #2 | Support color chooser and file picker dialogs. | Not implemented |
| Dialogs #3 | Add support for HTTP authentication. | Done |
| M2 | Custom URL scheme handler | sprocket:// | Not implemented |
| Settings | A settings page.  (sprocket://settings) | Not implemented |
| History | Manage history.  (sprocket://history) | Not implemented |
| Downloads | Manage downloads.  (sprocket://downloads) | Not implemented |
| Bookmarks | Manage bookmarks.  (sprocket://bookmarks) | Not implemented |
| Custom UI | Redesign the UI. | Not implemented |
| Incognito | Use off-the-record browser context. | Not implemented |

Currently we have three branches: master, core and testing.

* **Core** branch contains the relevant code to a minimal browser which can be easily extended with new features. The minimal browser supports Linux and Android as well, and it runs in fullscreen mode on both platforms. On Linux you can toggle between the full screen and windowed mode with F11 key.
* **Master** branch contains (will contain) the above listed features.
* **Testing** branch is the same as core, but it contains test specific features, for example devtools.

# Sprocket in details

In this section, we will discuss a simplified version of Content API’s architecture, what kind of classes exist, the connections between them and what we’ve done in our implementation.

To understand how the different units are implemented both in Content API and in Sprocket engine, and what the connections are between them, we summarized the Content API drawing mechanism in this figure: [Content API drawing](https://docs.google.com/drawings/d/11Hm1GOtoPb297RIzM1E2xg_TbO4tjitNfRisi3G2_m0/edit?usp=sharing)

Here you can see an overview of Sprocket's architecture: [High-Level,](https://docs.google.com/drawings/d/16Q6xKqj4OkoZV_6RugMbLDdvDfMsnSs3VAz38Sw9VWs/edit?usp=sharing) [Detailed.](https://docs.google.com/drawings/d/1ABSZHQ11SLVI9pNcxJxm0iPHGH6rR6ebVAgycFCfO_o/edit?usp=sharing)

In this documentation we will summarize the most important details about Sprocket’s architecture. Under each heading you can read about the relevant classes which the given folder contains.

## [App](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/)

The “app” folder contains the startup dependent classes.

The Content API’s entry point is the [ContentMain](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/content_main.h&l=68). This will start the whole program. We can customize the startup with implementing the [ContentMainDelegate](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/content_main_delegate.h) interface. Sprocket’s implementation for ContentMainDelegate is the [SprocketMainDelegate](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/app/main_delegate.h) class.

In Sprocket, the Android and Linux main processes are different, so we have two different paths for them.

On Linux the [main()](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fapp%2Fmain.cc&sa=D&sntz=1&usg=AFQjCNHtuQfUdINGdfBeO4mneH6NCC-JUw) function is used as an entry point which calls the [ContentMain()](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/content_main.h&l=68), passing our [SprocketMainDelegate](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.h) implementation.

On Android [Init()](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fandroid%2Flibrary_loader.cc%23L21&sa=D&sntz=1&usg=AFQjCNF9CS2HsTEkbrsFkZAi5qmvs09z5g) will call the [SetContentMainDelegate()](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/content_main.h&l=62) with the [SprocketMainDelegate](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/app/main_delegate.h) when the shared lib has been loaded.

In the [ContentMainDelegate](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/app/content_main_delegate.h) we can customize the startup. [Loading resources](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fapp%2Fmain_delegate.cc%23L85&sa=D&sntz=1&usg=AFQjCNHxLVDzd6bOFUjRbePUyyiBwZGhWg), [checking the command line](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.cc#L42) flags or [expanding it](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.cc#L42), [starting processes](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.cc#L42), etc.

## [Browser](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/)

The ‘browser’ part of the Content API is the backend for the application which handles all I/O and communication with the child processes. This also talks to the renderer to manage web pages to be painted on the screen.

See: <https://www.chromium.org/developers/design-documents/multi-process-architecture>

The [SprocketMainDelegate](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.h#L23) creates [clients](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fapp%2Fmain_delegate.cc%23L42&sa=D&sntz=1&usg=AFQjCNFGVA0SlURJhUN2-88FzAVRVbTwqA), in order to customize the content in different process types. The embedder can participate in the browser logic through its [SprocketContentBrowserClient](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/content_browser_client.h#L21) class.

The browser’s main process is created in [SprocketMainDelegate](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/app/main_delegate.cc#L60), and [BrowserMain](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fapp%2Fmain_delegate.cc%23L76&sa=D&sntz=1&usg=AFQjCNH-IxRseVWMPyTku4hO7cB61dciRg) uses this [BrowserMainRunner](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/browser_main_runner.h&l=16) to Initialize, Run and Shutdown. The Android port’s logic is different from this, and will be discussed later in the [Android](#_eog5hq46r41m) section.

The browser startup process consists of different stages, like EarlyInitialization, MainMessageLoopStart, etc. The [BrowserMainParts](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/browser_main_parts.h&l=50) class contains these and [here](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_main_parts.cc%23L93&sa=D&sntz=1&usg=AFQjCNGgqtQBNBG8fWU-2SSGFubuNzWxUw) we initialize the [SprocketBrowserContext](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/browser_context.h#L19) and create the [SprocketWebContents](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/web_contents.h#L26).

The [SprocketBrowserContext](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/browser_context.h#L19) class holds the context which is required for a browsing session. A typical browser application has two contexts: one for normal browsing and the other for the private/incognito session. [SprocketBrowserContext](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/browser_context.h#L19) stores information about the nature of the session:

* It holds the path of the directory where this context's data is stored.
* Is it off-the-record? (also known as “incognito mode”)

... and it gives us a chance to implement various interfaces:

* [ZoomLevelDelegate](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L80&sa=D&sntz=1&usg=AFQjCNEWSfMSi8tA4400dS2TvO5aZSs2zw): Maps hostnames to custom zoom levels and persist its information.
* [URLRequestContextGetter](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L90&sa=D&sntz=1&usg=AFQjCNGZjRjEHHBSo77iXBdzfUd9wVOS2A): Customization of the main networking behavior.
* [ResourceContext](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L119&sa=D&sntz=1&usg=AFQjCNFQSvTiwM9rTMRNpQI5EWt1lh6RjA): It contains the relevant context information required for resource loading.
* [DownloadManagerDelegate](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L124&sa=D&sntz=1&usg=AFQjCNFEJPY35oc4_mjVecW4iu9B9uw6nA): Manages all downloads and destination view.
* [BrowserPluginGuestManager](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L129&sa=D&sntz=1&usg=AFQjCNGl2-D0cryhAV6X_ktISiN5QhXY_w): Offloads guest management and routing operations outside of the content layer.
* [SpecialStoragePolicy](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L134&sa=D&sntz=1&usg=AFQjCNGQP-xBmtiArCrtvTbWELnndkSFlA): Access to the storage subsystems: special rights are granted to 'extensions' and 'applications'.
* [PushMessagingService](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L139&sa=D&sntz=1&usg=AFQjCNHYt_rcUJ9kFTp6HP6DMGlZe-HyLw): The Push API uses for talking to push messaging services.
* [SSLHostStateDelegate](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L144&sa=D&sntz=1&usg=AFQjCNF8bt2OwdWXM3TBRBA9Z9j6ePc5cg): Encapsulates the host-specific state for SSL errors.
* [PermissionManager](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fbrowser_context.cc%23L148&sa=D&sntz=1&usg=AFQjCNGK-XAt-F0lIkh584yzNA-srqSUMA): Manages feature (e.g. geolocation) permissions (requesting, cancellation, etc...).

The [URLRequest](https://code.google.com/p/chromium/codesearch#chromium/src/net/url_request/url_request.h) class represents the asynchronous load of a data stream from an URL and the [URLRequestContext](https://www.google.com/url?q=https%3A%2F%2Fcode.google.com%2Fp%2Fchromium%2Fcodesearch%23chromium%2Fsrc%2Fnet%2Furl_request%2Furl_request_context.h%26l%3D51) provides application-specific context for its instances. An embedder can customize the [URLRequestContext](https://code.google.com/p/chromium/codesearch#chromium/src/net/url_request/url_request_context.h&l=51) objects through the [URLRequestContextGetter](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fnet%2Furl_request_context_getter.h%23L34&sa=D&sntz=1&usg=AFQjCNGcdo2k5QEM7KJIpZBwRtFmqNGUBg) interface: [Storage](https://code.google.com/p/chromium/codesearch#chromium/src/net/url_request/url_request_context_storage.h&l=37), [Proxy Service](https://code.google.com/p/chromium/codesearch#chromium/src/net/proxy/proxy_service.h&rcl=1430177881&l=46), [Cert Verifier](https://code.google.com/p/chromium/codesearch#chromium/src/net/cert/cert_verifier.h&rcl=1430177881&l=27) (ignore cert error), [SSL Config Service](https://code.google.com/p/chromium/codesearch#chromium/src/net/ssl/ssl_config_service.h&l=25), [Host Resolver](https://code.google.com/p/chromium/codesearch#chromium/src/net/dns/host_resolver.h&rcl=1430177881&l=41), [Http Auth Handler](https://code.google.com/p/chromium/codesearch#chromium/src/net/http/http_auth_handler_factory.h&rcl=1430177881&l=30), [Cookie Store](https://code.google.com/p/chromium/codesearch#chromium/src/net/cookies/cookie_store.h&rcl=1430192095&l=30), [User Agent](https://code.google.com/p/chromium/codesearch#chromium/src/net/url_request/http_user_agent_settings.h&rcl=1430192095&l=19), etc...

[Sprocket](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fnet%2Furl_request_context_getter.h%23L34&sa=D&sntz=1&usg=AFQjCNGcdo2k5QEM7KJIpZBwRtFmqNGUBg) uses the default classes, but we will continuously extend these with custom implementations (e.g. M1: Auth handler).

The [SprocketWebContents](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fweb_contents.h%23L26&sa=D&sntz=1&usg=AFQjCNGblrIZ3ApsOE1JszAdxe0ZX1bBnQ) class is our wrapper for Chromium’s [WebContents](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/web_contents.h). This is the core class in src/content/. It renders web content (usually HTML) in a rectangular area and it will do all the [multi-process stuff](http://www.chromium.org/developers/design-documents/multi-process-architecture) behind the scenes. In this unit we handle the navigation, toggle fullscreen, open URL, etc. all the web content related operations.  
Each WebContents has exactly one [NavigationController](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/navigation_controller.h); each NavigationController [belongs](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/web_contents.h&l=170) to one WebContents. The NavigationController can be obtained from [GetController()](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/web_contents.h&l=172), and is used to load URLs into the WebContents, to navigate backwards/forwards in the history, etc.

[SprocketWindow](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow.h&sa=D&sntz=1&usg=AFQjCNGf1bYVOqsAnGXMraj8US-hTkd-og) represents one window, all the UI elements - including buttons and URL bar logic, as well as the web content area. As you might know the browser’s UI is different on Android compared to Linux (for more information check out the [Android](#_eog5hq46r41m) section).

[On Linux](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_aura.cc&sa=D&sntz=1&usg=AFQjCNGKZD4_DPQLQ3Rf3e7ylEyAAksI_g) we use [Aura](https://www.chromium.org/developers/design-documents/aura/aura-overview). Our [View (SprocketWindowDelegateView)](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.h%23L31&sa=D&sntz=1&usg=AFQjCNHVegqP_vKG3LmjCpHSKaUeuf8KkQ) is a [WidgetDelegate](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/widget/widget_delegate.h) - which handles all the window related operations - with a [TextfieldController](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/textfield/textfield_controller.h) - for the url bar - and a [ButtonListener](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/button/button.h) - for the buttons as well.

In [SprocketViewsDelegateAura](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/views_delegate_aura.h) we [set our native widget](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/views_delegate_aura.cc#L32), which is the [DesktopNativeWidgetAura](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/widget/desktop_aura/desktop_native_widget_aura.h&l=53), and the [default icon](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/views_delegate_aura.cc#L20) and [application name](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/views_delegate_aura.cc#L48).

On master branch - which has tab support - the view contains two subviews: a [toolbar](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.h%23L22&sa=D&sntz=1&usg=AFQjCNG2ssUhU1FGAXdLgABBeKOi46LXUA) and a [tabbed pane](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftabbed_pane.h%23L25&sa=D&sntz=1&usg=AFQjCNEG2y1Wp_CxU_8dUWd0t8MgNZ5uXg). Sprocket’s UI layout is a simple [grid](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L128&sa=D&sntz=1&usg=AFQjCNFHYl_CmkH32B7kCvtiKXfmuv_TTQ) with two rows. These rows are for the toolbar and the tabbed pane.

The [toolbar](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.h%23L22&sa=D&sntz=1&usg=AFQjCNG2ssUhU1FGAXdLgABBeKOi46LXUA) contains the buttons ([back](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.cc%23L35&sa=D&sntz=1&usg=AFQjCNET3onBSX7EigC0bq4BM9BMWklV1w), [forward](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.cc%23L44&sa=D&sntz=1&usg=AFQjCNFCSVC4SctF6Wx9e0cU2k8cay5yig), [refresh and stop](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.cc%23L54&sa=D&sntz=1&usg=AFQjCNE0p3_QiYnEEWn4XMAdhvrzWAv5HA)) and the [URL bar](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Ftoolbar.cc%23L54&sa=D&sntz=1&usg=AFQjCNE0p3_QiYnEEWn4XMAdhvrzWAv5HA). We can define [accelerators](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L159&sa=D&sntz=1&usg=AFQjCNEI7bM7M78jUR1qrWYG5g_kdGNZ-g) for the UI elements and handle [button presses](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L160&sa=D&sntz=1&usg=AFQjCNEstP-CEtheg3cBp47u2Bhk3sKQpA). It is also possible to [load](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L160&sa=D&sntz=1&usg=AFQjCNEstP-CEtheg3cBp47u2Bhk3sKQpA) the typed URL from the URL bar. A good example how to [control](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_aura.cc%23L90&sa=D&sntz=1&usg=AFQjCNHN1U4MQ7aD89qYs7EXIFgV1WBAYw) which UI element can be enabled can be seen in Sprocket.

The [tabbed pane](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.h#L25) contains a [scrollview](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.cc#L128) and the contents view. The scrollview contains the tabs within a [tab strip](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.cc#L34) and a [new tab button](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.cc#L240). The contents view’s child views are the webviews associated with the [tabs](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tab.h#L23). When a [tab gets selected](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.cc#L171), the associated child view of the contents view becomes visible. [Adding a tab](https://github.com/szeged/sprocket/blob/0db915417cefadab977b2a5c0df11d33b0961f22/browser/ui/tabbed_pane.cc#L157) consists of two steps: adding a tab as a child view to the tabstrip, and adding the tab’s content view as a child view to tabbed pane’s contents. The tab’s content view is a [web view](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L20&sa=D&sntz=1&usg=AFQjCNEuXDnVCwB8UMzIpQw7V7NaMDq_nA), which has a [web content](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L21&sa=D&sntz=1&usg=AFQjCNFbhNMedIMefZWbb1y6NWbyi4xrAQ).

The [context menu](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fbrowser%2Fui%2Fwindow_delegate_view_aura.cc%23L67&sa=D&sntz=1&usg=AFQjCNF9naqjNRhdQK7XQ4hddRaKGIt9uA) has the same functionality as the buttons. It can be extended, and we are planing to add the the cut, copy and paste functions to the context menu.

On core and testing branch there is no toolbar or tabbed pane. The window uses a [FillLayout](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_delegate_view_aura.cc#L46) to stretch its content, which is the web view.

## [Renderer](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/renderer/)

Currently Sprocket has no renderer folder, because we use the default implementation of these classes. In M2 we will extend this to show how easily you can add new features.

Sprocket specific code that runs in the renderer process could be put here. Features like auto fill, translate, etc. can be added to the content module with extension of these classes

## [Common](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/common/)

You can find here those files which are shared between the multiple processes (i.e. browser and renderer, renderer and plugin, etc...). Here you can declare constants, switches, favicon, etc, basically everything you will need to access from all processes.

Currently Sprocket [implements](https://www.google.com/url?q=https%3A%2F%2Fgithub.com%2Fszeged%2Fsprocket%2Fblob%2F0db915417cefadab977b2a5c0df11d33b0961f22%2Fcommon%2Fcontent_client.h&sa=D&sntz=1&usg=AFQjCNFKVteQcKn2_NJ-MsI2uomfrAfoQA) only the [ContentClient](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/common/content_client.h&l=68).

## [Android](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/android/)

[In this folder](https://github.com/szeged/sprocket/tree/master/android) you can find source codes for the Android port’s implementation. The code of an Android application is based on Java. Most of the Content API is written in C++, but to work with Android, it has Java (JNI) bindings. Fortunately with [JNI](http://en.wikipedia.org/wiki/Java_Native_Interface) we can call C++ functions and vica versa. To make this work, you need to [bind](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/library_loader.cc#L33) the [Init](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/library_loader.cc#L21) and [RegisterJNI](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/library_loader.cc#L17) functions. This will be called when the shared library is loaded first.

We have defined some [helper functions](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/manager.h) which will provide a bridge between the java and C++ code. It is not a hard requirement, but could be handy.

The following part requires some knowledge about Android. If you are new in this topic, we can suggest to read this basic [tutorial](https://developer.android.com/training/basics/firstapp/index.html) at least. If you know the following terms it will be easy to follow us in this part of the documentation: [SDK Manager](https://developer.android.com/tools/help/sdk-manager.html), [Apk](http://en.wikipedia.org/wiki/Android_application_package), [AVD](https://developer.android.com/tools/devices/index.html), [Application](https://developer.android.com/guide/components/fundamentals.html), [Activities](https://developer.android.com/guide/components/activities.html), [Resources](https://developer.android.com/guide/topics/resources/overview.html), [Manifest](https://developer.android.com/guide/topics/manifest/manifest-intro.html), [LinearLayout](https://developer.android.com/reference/android/widget/LinearLayout.html), [EditText](https://developer.android.com/reference/android/widget/EditText.html), [Button](https://developer.android.com/reference/android/widget/Button.html), [FrameLayout](http://developer.android.com/reference/android/widget/FrameLayout.html), [Intent](https://developer.android.com/guide/components/intents-filters.html), [Context](https://developer.android.com/reference/android/content/Context.html).

Implemented java classes:

* [Application](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketApplication.java): Entry point of the Sprocket application. Handles initialization of information that needs to be shared across the main Activity and the created child services.
* [Activity](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java): Activity for managing the SprocketWindow.
* [Manager](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketManager.java): Container and generator of the views.
* [Window](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketWindow.java): Container for various UI components that make up a window. (Currently we support only one window.)

Although in general there is no need to subclass [Application,](http://developer.android.com/reference/android/app/Application.html) but we will need to load our dependencies (paks) and initialize the command line. The entry point of the Content API can be found [here](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/android/java/src/org/chromium/content/app/ContentApplication.java).

Our [Activity](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java) redefines the [onCreate](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L50) and [onStart](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L177) methods of the [Activity Lifecycle](http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle).

On Create: the [command line initialization](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L53) has to be done before loading the library, because we need to pass the [command line arguments](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L55) first. After that, the engine tries to load the native [libraries](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L64) with help of the [LibraryLoader](https://code.google.com/p/chromium/codesearch#chromium/src/base/android/java/src/org/chromium/base/library_loader/LibraryLoader.java&q=libraryloader.java&sq=package:chromium&type=cs&l=42). If it [fails](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L66), nothing can work, so we have to [kill the whole application](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L69), not just the activity. After loading the libraries we create the [Manager](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketManager.java) - also [create](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketManager.java#L49) a [C++ reference](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/manager.cc#L51) of it - and the Activity Window. Next, we [get](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketManager.java#L33) the startup URL and if it is [not empty](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L85), we [replace](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L86) our defaultURL. And finally we can [start](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L92) the browser process. If the startup fails, we [notify](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L118) the user. If succeed, we [launch](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L115) the web contents, which basically [creates](https://github.com/szeged/sprocket/blob/master/android/manager.cc#L63) a SprocketWindow and a SprocketWebContents. This is where [Java](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/java/src/hu/uszeged/sprocket/SprocketManager.java#L132) calls a [C++](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/manager.cc#L60) function. For now, if we have an active window, we close it, and open a new one with the URL which was set before. There should not be any active window when onCreate is called, but there is a [corner case (left side)](http://developer.android.com/reference/android/app/Activity.html#ActivityLifecycle).

OnStart: We should [have](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L180) an active content view, so let’s [show](https://github.com/szeged/sprocket/blob/f998f1bf50454636d42a0efa115c9e14a5b8f8a5/android/sprocket_apk/src/hu/uszeged/sprocket_apk/SprocketActivity.java#L181) it.

After this, we should have the browser up, and the Window will handle all the UI interactions.

# Todo

In this section we are listing those features, guides, notes, etc. which will be added later to the document. Feel free to comment or extend them! Moreover any comments are welcome on any part of this document!

## [Child](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/child/)

/\* TODO.\*/

## [Utility](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/utility/)

Code for running operations in a sandboxed process. The browser process uses it when it wants to run an operation on untrusted data.

/\* TODO.\*/

## [Sandbox](https://www.chromium.org/developers/design-documents/sandbox)

By default, we [disable](https://github.com/szeged/sprocket/blob/207d7ebdd6d5fbe7ac18f7ed8de64ffe83397e7b/main_delegate.cc#L28) the sandbox feature.

In this tutorial (guide?), we will show how to add the sandbox feature.

First, we will need a command line switch to enable the sandbox. Let’s add our [switches file](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/common/switches.h) to the [common](https://github.com/szeged/sprocket/tree/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/common) directory, so we could access it from the entire application. Next, we need to [define](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/common/switches.cc#L11) the switch: “use-sandbox”. By default, we [disable](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/app/main_delegate.cc#L44) the sandbox mode, and only enable it when the use-sandbox flag is [set](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/app/main_delegate.cc#L43). The [browser process](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/app/main_delegate.cc#L42) will handle these command line flags.

Finally, we need to add the sandbox [target](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/sprocket.gyp#L170), and here is a guide [how to use it](https://github.com/szeged/sprocket/blob/668eeed3d62c3fadf62c9edd5ec29d4772c2fd91/README.md#sandbox-linux).

Now if we use the --use-sandbox command line argument, the application will run in sandboxed mode.

On Android, sandboxing is default. With version 4.1 (Jelly Bean), a new feature has been added to Android, it is called [process isolation](http://developer.android.com/guide/topics/manifest/service-element.html#isolated). Setting a process isolated means, that it will not have any permissions on its own, and it is isolated from the rest of the system. For instance an isolated process can only read the files / folders from the APK itself, but can not read any other data which is on the external or internal storage of the device. To sandbox a process on Android, you can set the [android:isolatedProcess=true](https://github.com/szeged/sprocket/blob/b2b53a0ef8b22717cd9ea39fa8ec586ffb2bd30e/android/sprocket_apk/AndroidManifest.xml.jinja2#L50) on the specified process. In Sprocket, this method is used for sandboxing.

## Fullscreen support

### Linux

To support fullscreen mode, first, some [WebContentsDelegate](https://code.google.com/p/chromium/codesearch#chromium/src/content/public/browser/web_contents_delegate.h&q=webcontentsdelegate&sq=package:chromium&type=cs&l=71) methods should be overridden in our [SprocketWebContents](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/web_contents.h) class. These methods are the following:

* [EnterFullscreenModeForTab](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/web_contents.h#L81)(...)
* [ExitFullscreenModeForTab](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/web_contents.h#L83)(...)
* [IsFullscreenForTabOrPending](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/web_contents.h#L84)(...)

These methods are called from the Content API when a website requests to be (or not to be anymore) in full screen: for example when the fullscreen button is pressed on a HTMLVideoElement. In the Enter… and Exit… methods the resizing behaviors can be configured. Besides Content API requesting the fullscreen view, the user should be able to toggle fullscreen mode as well. In Sprocket, this is possible with pressing the F11 key. The key event gets catched in [SprocketWebContents::HandleKeyboardEvent](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/web_contents.cc#L171) method and it is delegated to [SprocketWindowDelegateView](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_delegate_view_aura.cc#L99) which does the handling.

There is another method which plays a key role in the full screen toggling: [SprocketWindow::PlatformToggleFullscreenModeForTab](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_aura.cc#L81). It is important to save the fullscreen state of the window, before toggling to fullscreen mode on a website, because after switching to non-fullscreen view, the same state as before should be restored. For example, when a video is being watched, the user can toggle the *window* to fullscreen mode, but not the *video*. After this, the user can toggle the *video* to fullscreen as well. After switching to a non-fullscreen video mode, it is important that the *window* stays fullscreen, because only the *video* was toggled. To solve this problem, we added a [was\_fullscreen](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window.h#L99) variable to SprocketWindow’s implementation. Another common case is, when the user sets the window and the video to full screen at the same time. In this case, when the *window* fullscreen mode gets toggled off, the *video* (web content) should be toggled to normal mode as well. This logic is also implemented in [PlatformToggleFullscreenModeForTab](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_aura.cc#L86). The current states of the window and webcontents are kept in the [is\_fullscreen](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window.h#L98) variable in SprocketWindow and SprocketWebContents.

The above description applies for all branches. The difference in the implementation between core (testing) and master branch is in SprocketWindowDelegateView class. This class is responsible for the UI layout.

So first, let’s see, how it works on core and testing branch, because it’s the simpler way. The window’s layout is a simple [FillLayout](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_delegate_view_aura.cc#L24), which is completely filled with the WebView. To toggle fullscreen mode it is enough to call [SetFullscreen(bool fullscreen)](https://github.com/szeged/sprocket/blob/812db6836fa5df94f306769ad59ab696f6164f18/browser/ui/window_aura.cc#L85) on the windowWidget.

On master Sprocket, toggling fullscreen mode is more complicated because the task is not just to stretch a FillLayout to fullscreen, but to rearrange the whole layout in order to hide the toolbar, tabs, etc. To completely understand our solution, first you should have read the above *Browser* section. When fullscreen layout is [requested](https://github.com/szeged/sprocket/blob/6e217e44a8f8bbac7ce64e13ee8ec584ea4aa43a/browser/ui/window_delegate_view_aura.cc#L178), the gridView should be removed and replaced by the webView. When normal layout is [requested](https://github.com/szeged/sprocket/blob/6e217e44a8f8bbac7ce64e13ee8ec584ea4aa43a/browser/ui/window_delegate_view_aura.cc#L185), the webView is removed and replaced by the gridView.

### Android

On Android, the case is much simpler, because toggling the browser window to fullscreen is not an option on mobile browsers, so we only have to deal with toggling of the webcontents which was already described in the Linux section above.

## Tab support (Linux)

To support multiple browsing tabs, you can use the classes from [ui/view/controls/tabbed\_pane/](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/tabbed_pane/). These classes contain the most important features regarding tab support, but they need some customization for the proper use. So the simplest way is to implement these classes yourself, so that you can keep the implemented features and extend them in your own way. Three classes play the key role in tab support: [TabbedPane](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L25), [Tab](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L23), [TabStrip](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L34). TabbedPane contains a [tab strip](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L88) and the [contents view](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L89). TabStrip contains the tabs as its child views, and the contents view's child views are the webviews associated with each tab.

TabStrip is just a simple View subclass which contains unique [size](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L63) and [layout](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L73) calculations.

Tab represents a tab with its header and [content view](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L65) (webview). It contains a reference to the [tabbed pane](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L59) which contains it, and also a reference to its associated content view. We also added a [SprocketWebContents](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L67) member to the Tab class so that it could keep track of its own web content (back, forward, stop, load URL, etc). If you prefer, you can also add a [close button](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L60) to the Tab, but then you should change the [Layout()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.cc#L82) method. You can configure your tab's size in [GetPreferredSize()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.cc#L78) function. You can use dynamically changing widths, or just use constants. If you're looking for a simple solution you can put the TabStrip into a [ScrollView](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L128) and return constant sizes in this method. Originally there is no accessor provided to Tab's [title](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.h#L61) member, so dynamic title changes are not supported. You can provide a [setter](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tab.cc#L51) in order to support this.

TabbedPane is the class that is responsible for most of the tab related functions. It manages the tabs in the TabStrip and the associated content views. It ensures the non-null 1:1 relationship between content views and tabs. TabbedPane is responsible for tab additions, selections and closing. Tab closing is not supported originally, you can implement your custom function for it. [Here](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L198) you can find our example. If you added a close button to the Tab, you should [extend](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L26) the ButtonListener class in TabbedPane as well, and [override](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L298) ButtonPressed(...) method in order to catch the proper event. [GetTabAt(int index)](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L55) method is private originally, but it can be modified to be public, because in some cases it is necessary to call it from outside. We extracted the accelerator addition to a separate method called [InitAccelerators()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L232). In [AcceleratorPressed(...)](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L264) method you can bind other behaviours to specific key presses, but first you have to add the proper accelerator in InitAccelerators(). In order to get the window resizing to work correctly, [PreferredSizeChanged()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L283) method should be overridden, and you should provide the correct size calculations of the tab associated webviews. In our case we set all of the tabs webview’s sizes to [match](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L289) the selected tab's webview's size. You should also modify [GetPreferredSize()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L217) function: it's not necessary to calculate the [maximum](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/tabbed_pane/tabbed_pane.cc&l=333) size of the content views, you can use the [selected](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L221) tab's content view's size. We added an [extra button](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L91) to the tabstrip for opening new tabs. If you do this our way, you should change the [GetTabCount()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L143) method, because now tab\_strip\_->child\_count() won't be equal to contents\_->child\_count(), you should write [tab\_strip\_->child\_count() - 1](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L144) instead. (Because the new tab button is also a child view of the tab strip, but has no associated content view.) You should also [differentiate](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L299) two cases in ButtonPressed() method, based on who the sender is (new tab or close tab button).

You should also remove the early return from [SelectTabAt(int index)](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/tabbed_pane/tabbed_pane.cc&l=304), because it can create confusing situations in case of tab deletion (you delete a selected tab, and the tab that comes in place of the deleted should be selected, but the index stays the same). Also when implementing the [tab closing](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L198) you should take care of different cases ([selected tab deletion](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L206), which tab can come to its place, unselected tab deletion). You should consider a unified behavior to simplify your implementation. In our case when a selected tab gets deleted and it has other tabs on the right, then the tab on its right will replace it, if it is the rightmost tab, then the tab on its left becomes selected. If unselected tab gets deleted then the selected tab remains the same.

As mentioned before you can choose to put the tab strip into a scroll view instead of dynamic tab width changing, that is what we are using as well. In this case you should take care of the proper size calculations in [Layout()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L246). In the [scrollview size calculation](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L251) you should consider different cases: tab scrollbar is shown or not. In the content view size calculation you should use the [previously calculated scrollview height](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L259).

There is a fourth class that should be used, that's [TabbedPaneListener](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/tabbed_pane/tabbed_pane_listener.h&l=14). It is an abstract class so some of your own classes should extend it. TabbedPane has a [reference](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.h#L84) to a listener and it can communicate with the rest of the application through it. Originally it has one method: [TabSelectedAt(int index)](https://code.google.com/p/chromium/codesearch#chromium/src/ui/views/controls/tabbed_pane/tabbed_pane_listener.h&l=17). When a tab gets selected, TabbedPane [calls](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane.cc#L190) this method on it's listener, in order to perform certain operations depending on tab selection. We [expanded](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane_listener.h#L12) the listener interface with two other methods: [LastTabClosed()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane_listener.h#L16) and [OpenNewEmptyTab()](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/tabbed_pane_listener.h#L17). For example, the first method is necessary because it lets [SprocketWindowDelegateView](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/window_delegate_view_aura.cc#L109) know that the last tab has been closed, so it can call specific methods in order to close the whole window.

Besides these classes you should also modify your existing code. You should add methods to the window class regarding tab [addition](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/window_aura.cc#L66) and [selection](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/ui/window_aura.cc#L72). Before implementing the tab support the SprocketWebContents instance belonged to the SprocketWindowDelegateView, now it should belong to the tab.

The navigation controls, URL bar and window title should also change in regard of the selected tab. Therefore you should check if the tab is selected before [updating the navigation controls](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/web_contents.cc#L90), [setting the title](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/web_contents.cc#L150) or [setting the text of the url bar](https://github.com/szeged/sprocket/blob/93a988605622e2f619e6c684425029c2f5558902/browser/web_contents.cc#L234).