Tecnologie e applicazioni web

Electron

Filippo Bergamasco (filippo.bergamasco@unive.it)
http://www.dais.unive.it/~bergamasco/
DAIS - Università Ca'Foscari di Venezia
Anno accademico: 2017/2018

Electron

Electron è una libreria open source sviluppata nel 2014 da GitHub per creare applicazioni desktop cross-platform utilizzando tecnologie web:

- Runtime Javascript Node.js per la parte di backend
- Browser Chromium (HTML/CSS) per il rendering dell'interfaccia grafica

Electron crea un unico package eseguibile su Windows, Linux e OSX

Electron

Permette lo sviluppo di applicazioni desktop in puro Javascript, fornendo delle API per l'interfaccia verso il sistema operativo (derivate da Node.js)

Nota: Electron non fornisce un binding Javascript alle API grafiche native. La GUI è realizzata visualizzando un browser Chromium "minimale", controllabile da Javascript

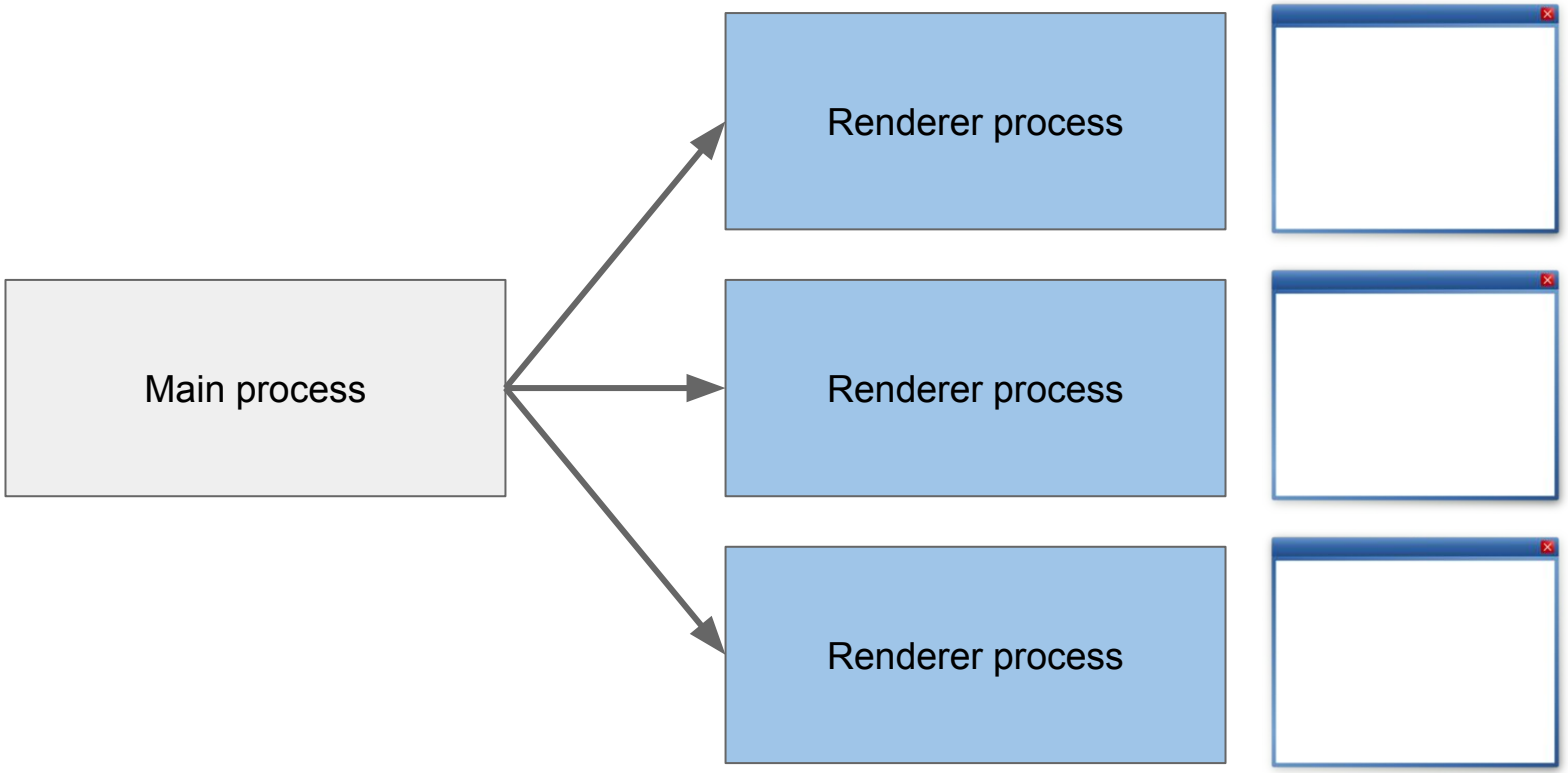
Main & renderer process

Quando l'applicazione viene eseguita, il codice Javascript dell'entry-point risiede in un processo chiamato **main process**.

Il main process può istanziare uno o più oggetti di tipo **`BrowserWindow`**, per creare delle finestre sullo schermo su cui possono essere visualizzati contenuti HTML.

Il codice Javascript di ciascuna BrowserWindow risiede in un proprio **renderer process**

Main & renderer process



Main & renderer process

Ciascun render process è isolato dagli altri render process e dal main process

Il codice dei vari interpreti Javascript viene eseguito in parallelo: codice bloccante su un process non blocca gli altri

Per evitare problemi di concorrenza, non è consigliabile condividere oggetti tra i vari processi

Electron mette a disposizione un meccanismo di ICP per la comunicazione

Prima applicazione

Un'app Electron è realizzata come un normale modulo Node.js. Il file `package.js` deve indicare l'entry point dell'applicazione nella property "main"

```
{
 "name": "MyFistApplication",
 "main": "test",
 "version": "0.0.1",
 "dependencies": {
   "electron-prebuilt": "^1.4.13"
 },
 "private": true,
 "scripts": {
   "start": "electron ."
 }
}
```

Prima applicazione

Esempio di entry point:

```
const {app, BrowserWindow} = require('electron')
const url = require('url')
const path = require('path')

let win


function createWindow() {
  win = new BrowserWindow({width: 800, height: 600})
  win.loadURL(url.format ({
     pathname: path.join(__dirname, 'index.html'),
     protocol: 'file:',
     slashes: true
  }))
}

app.on('ready', createWindow)
```

Prima applicazione

index.html

```
<!DOCTYPE html>
<html>
  <head>
     <meta charset = "UTF-8">
     <title>Hello World!</title>
  </head>

  <body>
     <h1>Hello World!</h1>
     We are using node <script>document.write(process.versions.node)</script>,
     Chrome <script>document.write(process.versions.chrome)</script>,
     and Electron <script>document.write(process.versions.electron)</script>.
  </body>
</html>
```

Eventi

Come per le app mobile, l'oggetto app può emettere diversi eventi corrispondenti alle diverse fasi del lifecycle dell'applicazione

ready: emesso quando il framework ha completato l'inizializzazione ed è pronto per creare finestre

window-all-closed: emesso quando tutte le finestre sono state chiuse

Quit: emesso prima della chiusura dell'applicazione

Eventi

Alcuni eventi sono caratteristici del browser chromium che gira in background

login: emesso quando il contenuto web richiede username/password per basic authentication

select-client-certificate: emesso quando la connessione HTTPS richiede di selezionare un certificato

IPC

Girando su processi diversi, main e renderer process possono comunicare attraverso due oggetti:

- **ipcMain**: usato nel main process per ricevere ed inviare messaggi asincroni verso i render process
- **ipcRenderer**: usato in un render process per ricevere ed inviare messaggi asincroni verso il main process

Combinando i due è possibile far comunicare fra loro più render process

System dialogs

Electron fornisce un oggetto “dialog” che permette di aprire delle finestre di dialogo standard fornite dal sistema operativo:

- Open / save dialogs
- Message boxes

I dialog possono essere aperti soltanto dal main process. Pertanto, occorre utilizzare l’IPC per richiedere l’apertura del dialog e comunicare il risultato al rendering process

Angular & electron

E'possibile trasformare una web application Angular in un applicazione stand-alone desktop grazie al framework Electron.

In modo simile a quanto fatto per la realizzazione di un'app mobile, è sufficiente produrre i files html+js con il comando `$ ng build` e aprire il file `index.html` in una BrowserWindow

Gestione dei moduli

Se l'applicazione Angular non utilizza nessuna API fornita da Electron, non è necessario fare alcuna modifica eccetto modificare il tag "base" di index.html

```
<base href="./">
```

In caso contrario (ad esempio per usare l'infrastruttura di IPC), è necessario modificare il "target" di webpack per il caricamento corretto dei moduli

Gestione dei moduli

Se l'app Angular è stata creata con angular-cli, è necessario "scollegare" il processo di build dall'utility cli e gestirlo manualmente.

In questo modo possiamo modificare il file webpack.config.js per inserire il target corretto per il caricamento di moduli

Step 1

“Scollegare” il processo di build da angular-cli:

```
$ ng eject
```

ed inserire i seguenti script in package.json

```
"scripts": {
  "ng": "ng",
  "start": "webpack-dev-server --port=4200",
  "build": "webpack",
  "electron": "electron .",
  "package": "electron-packager . pmdesktop --ignore=src",
  "pree2e": "webdriver-manager update --standalone false --gecko false --quiet"
}
```

Step 2

Modificare il file webpack.config.js. Nell'oggetto module.exports inserire la property:

```
"target": "electron-renderer"
```

A questo punto è possibile compilare ed eseguire l'applicazione electron con i comandi:

```
$ ng run build
$ ng run electron
```

Packaging

Electron viene fornito con un tool chiamato electron-packager che consente in modo semplice di creare un unico pacchetto eseguibile per ciascuna piattaforma. Utilizzo:

```
electron-packager . <appName>
--platform=<platform>
--arch=<architecture>
--out=<outputDirectory>
--overwrite
--icon=path/to/custom/icon
```

Per saperne di più...

https://electronjs.org/docs/tutorial