07

Detectando clicks no menu

Transcrição

O menu já está sendo gerado com sucesso a partir dos arquivos JSON dos cursos que possuímos, mas ao trocar o curso no menu, o mesmo também deve ser alterado na aplicação. Então, nesse vídeo iremos implementar essa comunicação, para que assim que mudarmos o curso no menu, um sinal seja disparado para o aplicativo visual, dizendo que o curso também deve ser alterado nele.

Quando o curso for clicado no menu, o evento `curso-trocado` será disparado, e a *view* ficará escutando-o, para ela ser atualizada com o novo curso em seguida.

Detectando um clique o item do menu

Já vimos como mandar um evento para da *view* para o processo principal, mas no nosso caso o menu está no processo principal, logo devemos fazer o inverso, mandar um evento do processo principal para a *view*.

O processo principal da nossa aplicação pode ter várias janelas-filhas, mas o contrário não acontece, já que a janela-filha só possui um processo principal. Por isso, em uma janela-filha, como **renderer.js**, quando fazemos `ipcRenderer.send(...)`, ele sabe que isso será enviado para o processo principal. Mas no processo principal, como o **main.js**, se fizermos `ipcMain.send(...)`, ele não saberá para qual janela-filha enviar o evento, por isso não há essa função `send` no processo principal.

Logo, não devemos usar a função `send` no `ipcMain`, e sim na nossa janela, que é quem possui essa função. Então, no exemplo do nosso aplicativo, faríamos `mainWindow.send(...)`.

Se só queremos enviar o evento quando o curso for trocado, clicado, como colocamos um evento de `click` no menu? Na hora que estamos montando o objeto do item do menu, podemos colocar a propriedade `click`, que executará algo quando o item for clicado:

```
// template.js

const data = require('./data');

module.exports = {
    geraTrayTemplate() {
        let template = [
            { 'label': 'Cursos' },
            { type: 'separator' }
        ];

        let cursos = data.pegaNomeDosCursos();
        cursos.forEach((curso) => {
            let menuItem = {
                label: curso,
                type: 'radio',
                // adicionando a propriedade click
                click: () => {

                }
```

```
            }
            template.push(menuItem);
        });
        return template;
    }
}
```

Assim que clicarmos no item, enviaremos um evento através da janela de visualização para o processo de renderização (**renderer.js**) e como vimos anteriormente, precisamos da janela para enviar o evento, logo vamos recebê-la por parâmetro na função e chamar a sua função `send`, enviando o evento e o curso selecionado:

```
// template.js

const data = require('./data');

module.exports = {
    // recebendo a janela na função
    geraTrayTemplate(win) {
        let template = [
            { 'label': 'Cursos' },
            { type: 'separator' }
        ];

        let cursos = data.pegaNomeDosCursos();
        cursos.forEach((curso) => {
            let menuItem = {
                label: curso,
                type: 'radio',
                // adicionando a propriedade click
                click: () => {
                    win.send('curso-trocado', curso);
                }
            }
            template.push(menuItem);
        });
        return template;
    }
}
```

E no **main.js**, enviamos a `mainWindow` para a função:

```
// main.js

// código omitido comentado

app.on('ready', () => {
    console.log('Aplicacão iniciada');
    mainWindow = new BrowserWindow({
        width: 600,
        height: 400,
    });
    tray = new Tray(__dirname + '/app/img/icon-tray.png');
    // Enviando a mainWindow para a função geraTrayTemplate
    let template = templateGenerator.geraTrayTemplate(mainWindow);
```

```
    let trayMenu = Menu.buildFromTemplate(template);
    tray.setContextMenu(trayMenu);

    mainWindow.loadURL(`file://${__dirname}/app/index.html`);
});

// restante do código comentado
```

Agora, no **renderer.js**, devemos escutar o evento `curso-trocado` :

```
// renderer.js

// restante do código omitido

ipcRenderer.on('curso-trocado', (event, nomeCurso) => {

});
```

Trocando o curso e carregando os seus dados

Como já selecionamos o curso no **renderer.js**, basta trocar o seu `textContent` para ser igual ao parâmetro recebido:

```
// renderer.js

// restante do código omitido

ipcRenderer.on('curso-trocado', (event, nomeCurso) => {
    curso.textContent = nomeCurso;
});
```

Ao executar a aplicação, vemos que já conseguimos trocar o nome do curso, resta agora carregar os seus dados. Já fizemos isso no próprio **renderer.js**, basta chamar a função `pegaDados` de data e atualizar o tempo:

```
// renderer.js

// restante do código omitido
ipcRenderer.on('curso-trocado', (event, nomeCurso) => {
    data.pegaDados(nomeCurso)
        .then((dados) => {
            tempo.textContent = dados.tempo;
        })
    curso.textContent = nomeCurso;
});
```

Com isso, a integração fica completa, ao alterar o curso no menu, ele também é alterado na aplicação e tem os seus dados carregados.