Tudo sobre o arquivo build.settings do Corona SDK

5. agosto 2014 Corona SDK, Mobile

Saiba tudo sobre Corona SDK com o único livro em Português sobre este incrível framework, com quase 500 páginas. Saiba mais!

Alguns desenvolvedores, especialmente aqueles novos com Corona ou com desenvolvimento de apps em geral, têm problemas com as configurações e parâmetros que são necessários para compilar com sucesso app para testar e eventualmente, lançar no mercado. Em Corona, estas configurações são ditadas através arquivo build.settings, mas inspecionar um build.settings complexo pode ser intimidador. Entretanto, se voc entende tables Lua e mentalmente separar este arquivo em seções, você irá entender melhor este element crucial do projeto.

Overview

Essencialmente, o arquivo build.settings contém informações que determinam quais recursos devem ocorre em tempo de compilação. Estes são divididos em 5 áreas primárias:

- Orientation - controla as orientações de tela suportadas;
- iOS Settings - características especiais do iOS
- Android Settings - características para Android
- Plugins - carrega vários plugins que um app pode precisar acessar
- Excluded Files - Exclui certos arquivos das suas compilações de app.

Isto tudo começa com uma simples table Lua chamada settings:

```
settings = {}
```

Desta forma, esta table pode conter vários pares de chave-valor para cada uma das cinco seções, e cada u destas tables pode ter vários pares chave-valor aninhados, muitos dos quais exploraremos neste tutorial.

Declaração Chave-Valor

Estas seções chave-valor podem ser definidas de uma de três maneiras, ou uma combinação de todas três dependendo da sua preferência ou, e alguns casos, o nome da chave sendo definida.

1. Propriedades da Table

```
settings = {}
settings.orientation = {}
settings.iphone = {}
settings.android = {}
settings.plugins = {}
```

2. Array Associativo

```
settings = {}
settings["orientation"] = {}
settings["iphone"] = {}
settings["android"] = {}
settings["plugins"] = {}
```

Note que este método é exigido quando sua chave contém caracteres especiais como um traço. Por exemp se você precisa definir uma chave como hokey-pokey, você precisará especificar ela como settings["hokey-pokey"] e não como settings.hokey-pokey que é considerado inválido em Lua. Isto será muito importante quando definirmos os plugins.

3. Estrutura de Tables Aninhadas

```
settings = {
    orientation = {},
    iphone = {},
    android = {},
    plugins = {},
}
```

Observações Adicionais

Todos os três formatos são perfeitamente válidos, mas o formato padrão mostrado em nossos exemplos e documentação é o formato table aninhada.

Cada uma destas seções chave-valor é opcional. Se você está compilando somente para iOS, você não precisa incluir a table android. Da mesma forme, se estiver compilando para Android, você não precisa inclu table iphone. Se você não usa quaisquer plugins, você pode omitir a table plugins. E, embora você não pre de uma table orientation, você deve fornecer uma em praticamente todo app para definir as orientações permitidas ao app.

A ordem destas seções não é importante, desde que inclua-as corretamente como pares de chave-valor.

A table Orientation

A seção orientation é a mais fácil das cinco - ela meramente define qual orientação é a padrão para o app, quais outras orientações são permitidas se o usuário girar o telefone.

```
settings =
{
    orientation =
    {
        default = "landscapeRight",
        supported = { "landscapeRight", "landscapeLeft" }
    },
}
```

Existem duas chaves para esta table: default e supported. O valor default é uma string que determina a orientação na qual o app irá começar. Strings válidas incluem:

- "portrait"

- "portraitUpsideDown"
- "landscapeLeft"
- "landscapeRight"

O valor supported é uma table indexada (sem chaves) que pode conter quaisquer das quatro opções listada acima, separada por vírgulas. Note que esta table deve incluir o mesmo valor que você definiu como defaul em adição a quaisquer outras orientações que você deseja suportar. No exemplo acima, o app irá carregar "landscapeRight", mas também permite trocar para a orientação "landscapeLeft", visando fornecer uma me experiência aos usuários.

A table iphone

A table iphone contém todas as configurações necessárias para compilar para dispositivos iOS. Note o seguinte:

Esta table deve ser digitada como iphone (tudo em minúscula) mesmo que o dispositivo seja normalmente chamado de "iPhone" - este é um problema comum para novos usuários, então certifique-se de digitar corretamente.

A table iphone cobre todos os dispositivos iOS incluindo iPhone, iPad e iPod Touch, então você só precisa desta seção para compilar para todos eles.

A sub-table plist

A table iphone sempre contém outra table aninhada chamada plist. Esta é uma table cujos valores são copiados dentro do dicionários plist do projeto XCode. Logo, se você precisa incluir itens nessa lista, adicio os aqui.

As chaves dentro da table plist são as mesmas do XCode, mas os valores são orientados ao Lua. Por exem em XCode existe um par chave-valor chamado UIApplicationExitsOnSuspend que aceita valores booleanos XCode como YES e NO. Entretanto, Lua usa true ou false, logo, para definir esta preferência para um app Corona, sua table iphone deve incluir isto:

```
settings =
{
    iphone =
    {
        plist =
        {
            UIApplicationExitsOnSuspend = false,
        }
    },
}
```

A maioria das tables iphone incluirão mais do que isto, e alguns desenvolvedores comumente copiam esta table de build.settings existentes sem entender seu conteúdo. Considere o seguinte exemplo:

```
settings =
{
    iphone =
    {
        plist =
        {
            CFBundleIconFile = "Icon.png",
            CFBundleIconFiles =
            {
                "Icon.png",
                "Icon@2x.png",
                "Icon-60.png",
                "Icon-60@2x.png",
                "Icon-72.png",
                "Icon-72@2x.png",
                "Icon-76.png",
                "Icon-76@2x.png",
                "Icon-Small.png",
                "Icon-Small@2x.png",
                "Icon-Small-40.png",
                "Icon-Small-40@2x.png",
                "Icon-Small-50.png",
                "Icon-Small-50@2x.png",
            },
            UIApplicationExitsOnSuspend = false,
            FacebookAppID = "383847392938347383473",
            CFBundleURLTypes =
            {
                {
                    CFBundleURLSchemes =
                    {
                        "fb383847392938347383473",
                    }
                }
            }
        }
    }
}
```

Esta table pode parecer complexa, mas vamos dar uma olhada e explorar os vários elementos:

- CFBundleIconFile - esta String declara o nome base do seu ícone do app. Este valor pode ser geralmente "Icon.png" (note que é case-sensitive com I maiúsculo).
- CFBundleIconFiles - esta é uma table indexada (sem chaves) que diz ao iOS quais arquivos de ícone devem ser usados nos vários dispositivos. Para uma experiência de usuário melhor, você deve inclui todas estas opções, e deve criar todos os arquivos de ícones associados nos tamanhos apropriados.
- UIApplicationExitsOnSuspend - este valor boolean instrui o IOS que o app não deve terminar quando suspenso. Este valor deve ser false na maioria das vezes.
- FacebookAppID - para implementar a funcionalidade do Facebook, você precisará incluir este par ch valor. Seu valor único pode ser obtido no painel de desenvolvedor do Facebook.
- CFBundleURLTypes - este é um array indexado de chaves-valores que descrevem os esquemas de suportados pelo app (consulte a documentação da Apple para maiores detalhes). Por causa desta estrutura, esta table começa com uma sub-table sem chaves adicional. No exemplo acima, relaciona um esquema de URL para o Facebok.

Outros Elementos

Este tutorial não pode discutir cada elementos que possa existir na table iphone, então consulte a documentação oficial do Corona e da Apple para maiores informações.

A table android

O Android é razoavelmente simples na maioria dos casos - tipicamente é somente uma lista de permissões características. Note que você deve incluir uma table usesPermission dentro da table Android que é uma lis de permissões que você quer conceder ao app. No exemplo seguinte, permitimos ao app Corona acesso à Internet.

```
settings =
{
    android =
    {
        usesPermissions =
        {
            "android.permission.INTERNET",
        },
    },
}
```

Uma versão um pouco mais complexa pode incluir uma table usesFeatures que diz ao Android Marketplace quais capacidades de hardware o app precisa ou não precisa.

```
settings =
{
    android =
    {
        usesPermissions =
        {
            "android.permission.INTERNET",
            "android.permission.WRITE_EXTERNAL_STORAGE",
            "android.permission.ACCESS_FINE_LOCATION",
            "android.permission.ACCESS_COURSE_LOCATION",
        },
        usesFeatures =
        {
            { name="android.hardware.camera", required=true },
            { name="android.hardware.location", required=false },
            { name="android.hardware.location.gps", required=false },
        },
    },
}
```

Outros Elementos

Este tutorial não pode discutir cada elemento que pode existir na table android, para isso consulte a documentação oficial do Corona e a documentação do Google, na parte de permissions e features.

A table plugins

A seção final a ser discutida é a table que suporta os plugins do Corona SDK. Um exemplo básico que habi o plugin AdMob se pareceria com isso:

```
settings =
{
    plugins =
    {
        ["CoronaProvider.ads.admob"] =
        {
            publisherId = "com.coronalabs",
        },
    },
}
```

Note que o nome da chave para est plugin (CoronaProvider.ads.admob) inclui pontuação. Devido a isto, vo deve usar colchetes e aspas ao definir esta chave. Mesmo que o nome do plugin não inclua pontuação, est método fornece um nível de consistência:

```
settings =
{
    plugins =
    {
        ["facebook"] =
        {
            publisherId = "com.coronalabs",
            supportedPlatforms = { iphone = true, ["iphone-sim"] = true },
        },
    },
}
```

Neste exemplo mostramos um aspecto importante de incluri plugins: a sub-table supportedPlatforms. Se vo usar um plugin que é específico de uma plataforma, você pode usar a chave supportedPlatforms para instru os servidores de build do Corona a somente incluírem o plugin na plataforma específica.

Porque Facebook é considerado "platform specific" no contexto de plugins? A funcionalidade do Facebook na verdade, disponível para iOS e Android, mas desde algum tempo que o Facebook para iOS foi removidc core do Corona e agora é implementado via plugin. Se você está usando uma versão mais recente do Corc está compilando para iOS, você deve incluir a chave acima ["facebook"] na sua table plugins, juntamente cc sub-table supportedPlatforms.

Precauções Importantes

1. Copiando e Colando Blocos de Código

Comumente, desenvolvedores copiam e colam exemplo de código a documentação do Corona. Tudo bem, você deve estar atento a manter a organização de todas tables dentro do build.settings. Não copie simplesmente os exemplos mostrados acima e se copiar tome cuidado em não colar as sub-tables nas seçõ errradas, como colar um bloco plugin dentro de uma table iphone.

2. Indentação Apropriada

Realmente ajuda indentar seu código. A maioria dos editores de código modernos manipulam a indentação código automaticamente, e é ainda mais importante indentar o código do build.settings porque ele será normalmente uma miríade de tables e sub-tables. Com a indentação apropriada, será mais fácil de manter s elementos organizados e legíveis.

Conclusão

O build.settings pode ser bem complexo, mas no fundo, nada mais é do que tables Lua aninhadas com seu pares de chave-valor, e quanto mais confortável você estiver com esses aspectos, mais fácil será construir build sem falhas para todas plataformas.

Saiba tudo sobre Corona SDK com o único livro em Português sobre este incrível framework, com quase 500 páginas. Saiba mais!