

# IdeaVim活用術

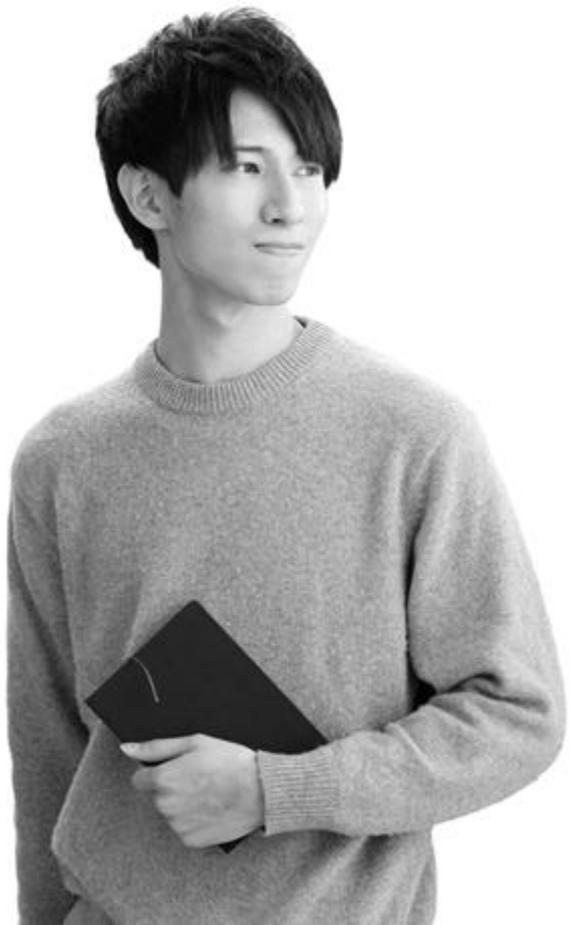
## ～複雑な繰り返し操作をやっつけよう～

Sansan株式会社

Sansan Engineering Unit Mobile Application Group 小林 慎悟

**sansan**





# 小林 慎悟

Sansan株式会社

技術本部 Sansan Engineering Unit  
Mobile Application Group

Androidエンジニア

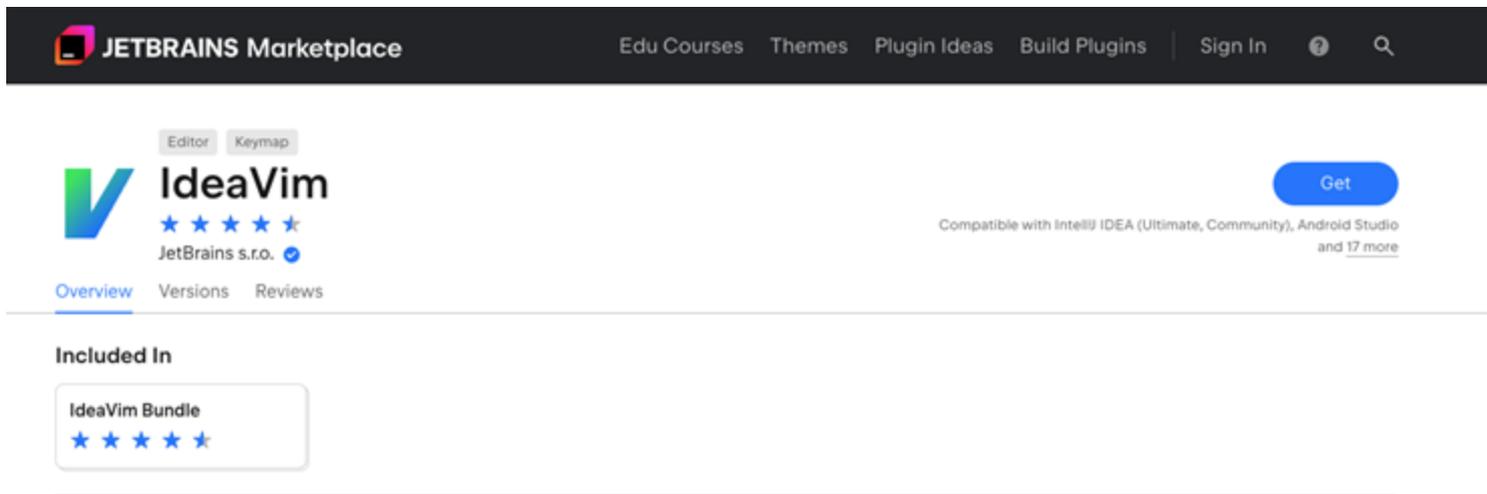
---

Slerに4年務めた後、2022年にSansan株式会社に入社。  
Androidメインにかかれこれ6年くらいアプリ開発をやっている。

半年前くらいからVimにハマってる（なので今日もVimのはなし）

# IdeaVimとは

- AndroidStudioなどJetBrains製IDEでVimのキーバインドを使えるようにするプラグイン



The screenshot shows the JetBrains Marketplace interface for the IdeaVim plugin. At the top, the 'JETBRAINS Marketplace' logo is on the left, and navigation links for 'Edu Courses', 'Themes', 'Plugin Ideas', 'Build Plugins', 'Sign In', and a search icon are on the right. The main content area features the 'IdeaVim' plugin card, which includes a 'Get' button, a 5-star rating, and compatibility information: 'Compatible with IntelliJ IDEA (Ultimate, Community), Android Studio and 17 more'. Below the main card, there is a section titled 'Included In' which contains a box for 'IdeaVim Bundle' with a 5-star rating.

実装時にこんな場面ないでしょうか？



```
data class User(  
    val id: String, // ユーザーの一意的ID  
    val firstName: String, // ユーザーの名  
    val lastName: String, // ユーザーの姓  
    val department: String, // ユーザーが所属する部署  
    val email: String, // ユーザーのメールアドレス  
    val phoneNumber: String, // ユーザーの電話番号  
    val dateOfBirth: String, // ユーザーの生年月日  
    val hireDate: String, // ユーザーの採用日  
    val salary: Double, // ユーザーの給料  
    val position: String, // ユーザーの役職  
    val address: String, // ユーザーの自宅住所  
    val supervisorId: String?,  
    val isActive: Boolean,  
    val lastLogin: String?  
)
```



すべてのプロパティをprivateにして、  
プロパティ名の先頭にアンダーバーをつけたい

```
data class User(  
    private val id: String, // ユーザーの一意的ID  
    private val firstName: String, // ユーザーの名  
    private val lastName: String, // ユーザーの姓  
    private val department: String, // ユーザーが所属する部署  
    private val email: String, // ユーザーのメールアドレス  
    private val phoneNumber: String, // ユーザーの電話番号  
    private val dateOfBirth: String, // ユーザーの生年月日  
    private val hireDate: String, // ユーザーの採用日  
    private val salary: Double, // ユーザーの給料  
    private val position: String, // ユーザーの役職  
    private val address: String, // ユーザーの自宅住所  
    private val supervisorId: String?,  
    private val isActive: Boolean,  
    private val lastLogin: String?  
)
```



すべてのプロパティをprivateにして、  
プロパティ名の先頭にアンダーバーをつけたい

```
5
6 data class User(
7     val id: String,           // ユーザーの一意のID
8     val firstName: String,    // ユーザーの名
9     val lastName: String,     // ユーザーの姓
10    val department: String,    // ユーザーが所属する部署
11    val email: String,        // ユーザーのメールアドレス
12    val phoneNumber: String,   // ユーザーの電話番号
13    val dateOfBirth: String,   // ユーザーの生年月日
14    val hireDate: String,      // ユーザーの採用日
15    val salary: Double,        // ユーザーの給料
16    val position: String,      // ユーザーの役職
17    val address: String,       // ユーザーの住所
18    val supervisorId: String?, //
19    val isActive: Boolean,     //
20    val lastLogin: String?     //
21 )
22 | I
```



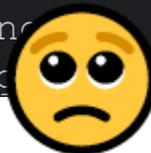
IDEのマルチカーソルを活用すれば一瞬だ

```
data class User(  
    private val _idLocal: String, // ユーザーの一意的ID  
    private val _firstNameLocal: String, // ユーザーの名  
    private val _lastNameLocal: String, // ユーザーの姓  
    private val _departmentLocal: String, // ユーザーが所属する部署  
    private val _emailLocal: String, // ユーザーのメールアドレス  
    private val _phoneNumberLocal: String, // ユーザーの電話番号  
    private val _dateOfBirthLocal: String, // ユーザーの生年月日  
    private val _hireDateLocal: String, // ユーザーの採用日  
    private val _salaryLocal: Double, // ユーザーの給料  
    private val _positionLocal: String, // ユーザーの役職  
    private val _addressLocal: String, // ユーザーの自宅住所  
    private val _supervisorIdLocal: String, // ユーザーの上司ID  
    private val _isActiveLocal: Boolean, // ユーザーがアクティブかどうか  
    private val _lastLoginLocal: String, // ユーザーの最終ログイン日時  
)
```



次に、プロパティ名の末尾にLocalをつけよう。

```
data class User(  
    private val _idLocal: String, // ユーザーの一意的ID  
    private val _firstNameLocal: String, // ユーザーの名  
    private val _lastNameLocal: String, // ユーザーの姓  
    private val _departmentLocal: String, // ユーザーが所属する部署  
    private val _emailLocal: String, // ユーザーのメールアドレス  
    private val _phoneNumberLocal: String, // ユーザーの電話番号  
    private val _dateOfBirthLocal: String, // ユーザーの生年月日  
    private val _hireDateLocal: String, // ユーザーの採用日  
    private val _salaryLocal: Double, // ユーザーの給料  
    private val _positionLocal: String, // ユーザーの役職  
    private val _addressLocal: String, // ユーザーの自宅住所  
    private val _supervisorIdLocal: String, // ユーザーの上司ID  
    private val _isActiveLocal: Boolean, // ユーザーがアクティブかどうか  
    private val _lastLoginLocal: String, // ユーザーの最終ログイン日時  
)
```



あれ？今回は縦のラインが揃ってないからマルチカーソルでさっきみたいにできないぞ

こんな時、IdeaVimがあれば



```
5
6 R data class User(
7     private val _id: String, // ユーザーの一意的ID
8     private val _firstName: String, // ユーザーの名
9     private val _lastName: String, // ユーザーの姓
10    private val _department: String, // ユーザーが所属する部署
11    private val _email: String, // ユーザーのメールアドレス
12    private val _phoneNumber: String, // ユーザーの電話番号
13    private val _dateOfBirth: String, // ユーザーの生年月日
14    private val _hireDate: String, // ユーザーの採用日
15    private val _salary: Double, // ユーザーの給料
16    private val _position: String, // ユーザーの役職
17    private val _address: String, // ユーザーの自宅住所
18    private val _supervisorId: String?, // ユーザーの上司のID (オプション)
19    private val _isActive: Boolean, // ユーザーがアクティブかどうか (雇用中か)
20    private val _lastLogin: String? // ユーザーの最後のログイン日時 (オプション)
21 )
22
```

```

5
6 R data class User(
7     private val _id: String,
8     private val _firstName: String,
9     private val _lastName: String,
10    private val _department: String,
11    private val _email: String,
12    private val _phoneNumber: String,
13    private val _dateOfBirth: String,
14    private val _hireDate: String,
15    private val _salary: Double,
16    private val _position: String,
17    private val _address: String,
18    private val _supervisorId: String?,
19    private val _isActive: Boolean,
20    private val _lastLogin: String?
21 )
22

```

## 解説

```

// ユーザーの一意のID
// Vimにはマクロという一連の操
// 作を記録し、あとから繰り返し
// ユーザーが所属する部署
// 実行できる機能がある
// Vimにはマクロという一連の操
// 作を記録し、あとから繰り返し
// ユーザーの生年月日
// ユーザーの採用日
// ユーザーの採用日
// ユーザーの採用日
// プロパティが例え100個あっても
// 余裕で対応できる
// ユーザーの生年月日
// ユーザーの採用日
// ユーザーの上司のID (オプション)
// ユーザーがアクティブかどうか (雇用中かど
// ユーザーがアクティブかどうか (雇用中かど

```

キー入力: 最後のログイン日時 (オプション)

qqf\_eaLocal<Esc>+q13@q

もう少し複雑な繰り返し操作



```
data class User(  
    // ユーザーの一意的ID  
    private val _idLocal: String, // ユーザーの一意的ID  
    // ユーザーの名  
    private val _firstNameLocal: String, // ユーザーの名  
    // ユーザーの姓  
    private val _lastNameLocal: String, // ユーザーの姓  
    // ユーザーが所属する部署  
    private val _departmentLocal: String, // ユーザーが所属する部署  
    // ユーザーのメールアドレス  
    private val _emailLocal: String, // ユーザーのメールアドレス  
    ...  
    // ユーザーがアクティブかどうか (雇用  
    private val _isActiveLocal: Boolean  
    // ユーザーの最後のログイン日時 (オフ  
    private val _lastLoginLocal: String  
)
```



コメントを上の方に移動させたい

```
5
6  data class User(
7     private val _idLocal: String,           // ユーザーの一意のID
8     private val _firstNameLocal: String,    // ユーザーの名
9     private val _lastNameLocal: String,     // ユーザーの姓
10    private val _departmentLocal: String,    // ユーザーが所属する部署
11    private val _emailLocal: String,         // ユーザーのメールアドレス
12    private val _phoneNumberLocal: String,   // ユーザーの電話番号
13    private val _dateOfBirthLocal: String,   // ユーザーの生年月日
14    private val _hireDateLocal: String,      // ユーザーの採用日
15    private val _salaryLocal: Double,        // ユーザーの給料
16    private val _positionLocal: String,      // ユーザーの役職
17    private val _addressLocal: String,       // ユーザーの自宅住所
18    private val _supervisorIdLocal: String?, // ユーザーの上司のID (オプション)
19    private val _isActiveLocal: Boolean,     // ユーザーがアクティブかどうか (雇用中かどうか)
20    private val _lastLoginLocal: String?     // ユーザーの最後のログイン日時 (オプション)
21 )
22
```

```

5
6 R  data class User(
7     private val _idLocal: String,
8     private val _firstNameLocal: String,
9     private val _lastNameLocal: String,
10    private val _departmentLocal: String,
11    private val _emailLocal: String,
12    private val _phoneNumberLocal: String,
13    private val _dateOfBirthLocal: String,
14    private val _hireDateLocal: String,
15    private val _salaryLocal: Double,
16    private val _positionLocal: String,
17    private val _addressLocal: String,
18    private val _supervisorIdLocal: String?,
19    private val _isActiveLocal: Boolean,
20    private val _lastLoginLocal: String?
21 )
22

```

## 解説

```

// ユーザーの一意のID
// ユーザーの名
// ユーザーのメールアドレス
// ユーザーの電話番号
// ユーザーの生年月日
// ユーザーの採用日
// ユーザーの給料
// ユーザーの役職
// ユーザーの自宅住所
// ユーザーの上司のID (オプション)
// ユーザーの最後のログイン日時 (オプション)

```

- 先ほどと同様にマクロを活用
- 1行目のコメント手前までを一度削除、下に行を追加して貼り付け
- それを13回繰り返し

キー入力:

qqf/hv^do<Esc>p+q13@q

# おわりに

IdeaVimのマクロを活用して繰り返し操作を一瞬でやっつける方法を紹介しました。

今回の発表の内容は記事としても公開しています。

- [IntelliJユーザーに贈るIdeaVim活用術 ～複雑な繰り返し作業をやっつけよう～](#)

Vimをこれから始める人向けの記事も書いているので、そちらもぜひ。

- [今度こそ挫折しないVim](#)

***sansan***

