Gestion d'erreurs

Gestion d'erreurs dans les langages

Il y a plusieurs stratégies possibles :

- Code d'Erreurs
- Exceptions
- Pattern Matching
- ...

Go et le retour multiple

En Go, nous allons exploiter le retour multiple des fonctions pour gérer nos erreurs

Exemple classique

```
func MyFunc() (int, error) {
	// code
	return 1
}

func main() {
	v, err := MyFunc()

	if err != nil {
		fmt.Printf("Error in MyFunc: %v", err)
	}
}
```

> nil = NULL

Gestion d'erreur standard en Go

En Go, on peut retrouver souvent des codes qui auront cette forme-là pour gérer les erreurs

```
v1, err := MyFunc1()
if err != nil {
	return err
}

v2, err := MyFunc2()
	if err != nil {
return err
}

v3, err := MyFunc3()
	if err != nil {
return err
}

v4, err := MyFunc4()
	if err != nil {
return err
}
```

C'est un peu répétitif... Mais efficace ! Cela apporte une lecture du code progressivement.

Early return

En Go, on va favoriser les tests et retour d'erreurs en tout début de fonction, pour faire un retour d'erreur le plus rapidement possible, permettre à notre code qui suit de grandir plus facilement.

Code non-early return

```
func MyFunc(condition bool) (int, err) {
	if (condition) {
		if (!condition2) {
			return 0, errors.New("Error 2!")
```

```
□□}
□□// code
□□return 42, nil
□}
□return 0, errors.New("Error!")
}
```

Code early-return

```
func MyFunc(condition bool) (int, err) {
□if (!condition) {
□□return 0, errors.New("Error!")
□}
□if (!condition2) {
□□return 0, errors.New("Error 2!")
□}
□// code
□return 42, nil
}
```

Revision #2
Created 2022-05-09 19:12:13 UTC by Noé Larrieu-Lacoste
Updated 2022-05-09 19:13:13 UTC by Noé Larrieu-Lacoste