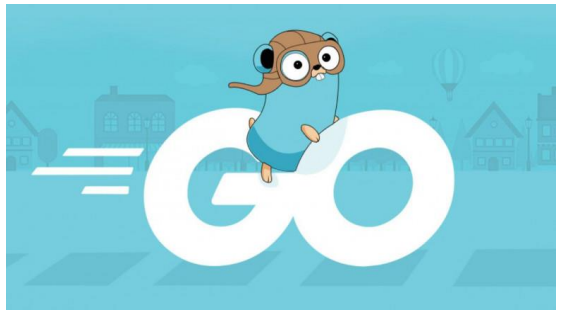
Go(Goland)



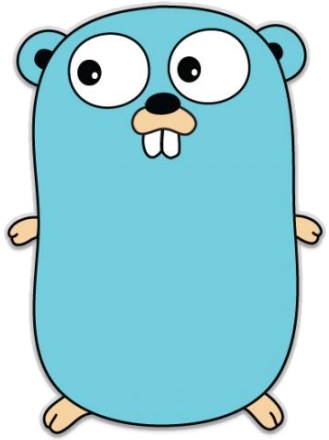
Leon Maciejewicz i Olgierd Mizgier

Wprowadzenie do języka Go (Golang)

```go
// Categoryrepository ...
var (
    Categoryrepository CategoryRepoInterface = &categoryrepository{}
    ctx                                      = context.TODO()
    collection                               = "category"
)

type CategoryRepoInterface interface {
    Create(category *Category) (*Category, httperrors.HttpErr)
    GetOne(id string) (*Category, httperrors.HttpErr)
    GetAll() ([]*Category, httperrors.HttpErr)
    GetAll1(search support.Paginator) (*Results2, httperrors.HttpErr)
    Update(id string, category *Category) (*Category, httperrors.HttpErr)
    Delete(id string) (string, httperrors.HttpErr)
    GetAllInfo() ([]Result, httperrors.HttpErr)
}

type categoryrepository struct {
    Mongodb *mongo.Database
    Bizname string
    Cancel  context.CancelFunc
}

func NewCategoryRepo() CategoryRepoInterface {
    return &categoryrepository{}
}
```

- Definicja: Go, znany również jako Golang, to język programowania stworzony przez Google w 2007.

Cel: Zapewnia prostotę, wydajność i skalowalność.



Twórcy

- ***Jego projektantami są:***
- Robert Griesemer
- Rob Pike oraz
- Ken Thompson

Filozfia projektowa

- Go powstał z kilku kluczowych założeń projektowych:
- Prostota i zwięzłość:
- Wydajność:
- Współbieżność:
- Bezpieczeństwo typów:
- Przenośność:

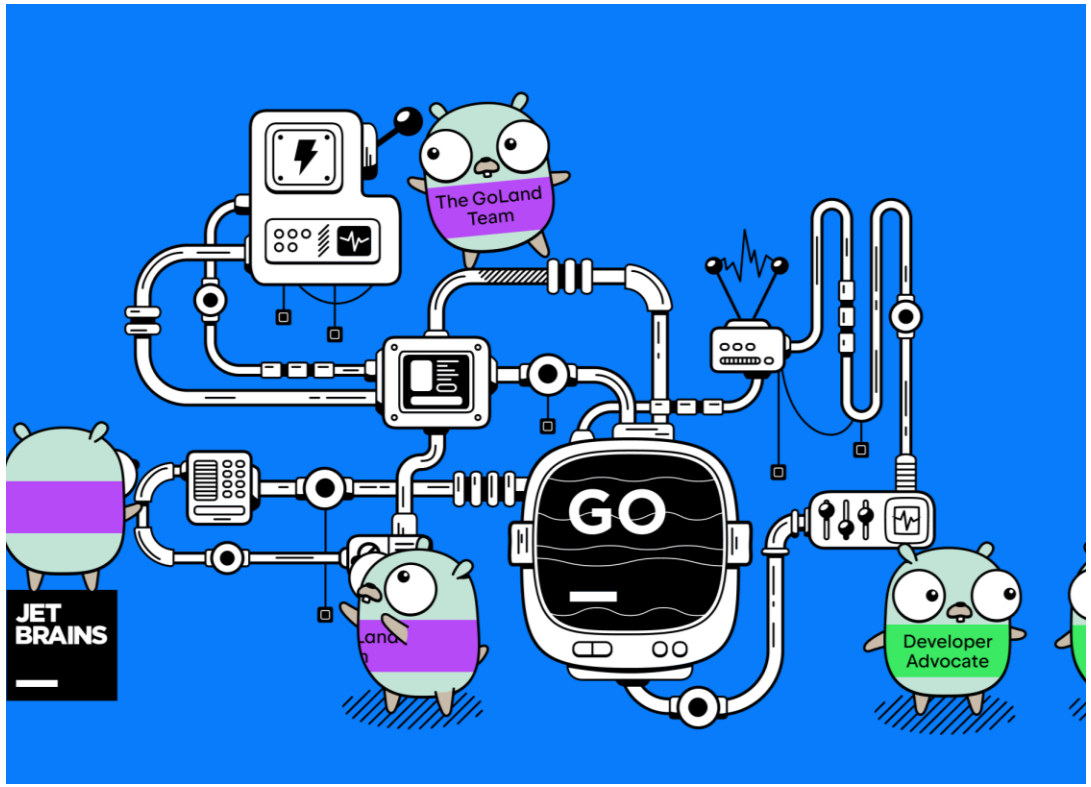
Typy zmiennych i deklaracje

```go
package main

import "fmt"

func main() {
    var age int = 25         // Liczba całkowita
    var height float64 = 5.9 // Liczba zmiennoprzecinkowa
    var grade rune = 'A'     // Znak
    var isStudent bool = true // Boolean

    fmt.Println(age, height, grade, isStudent)
}
```

- **Liczby całkowite (int)**: Liczby całkowite.
- **Liczby zmiennoprzecinkowe (float32, float64)**: Liczby z przecinkiem.
- **Znaki (rune)**: Pojedyncze litery lub symbole.
- **Boolean (bool)**: Wartości prawda/fałsz.

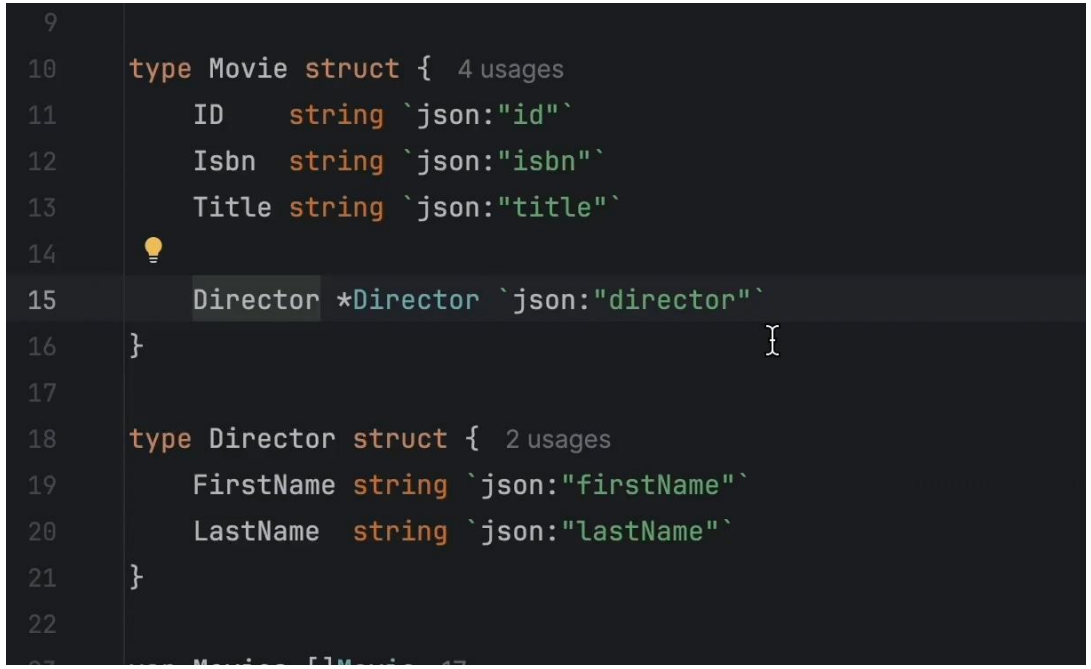
Ekosystem i narzedzia

- Go posiada bogaty ekosystem narzędzi, które wspierają programistów w tworzeniu, testowaniu i wdrażaniu aplikacji:
- **Go tools**: Zestaw narzędzi wbudowanych w język, takich jak `go build`, `go test`, `go fmt`, `go vet`, `go run` i `go get`, które ułatwiają zarządzanie projektem.
- **Biblioteka standardowa**: Go ma rozbudowaną bibliotekę standardową, która obejmuje różnorodne funkcjonalności, od obsługi sieci i formatowania tekstu, po manipulację czasem i serializację danych.
- **Pakiety zewnętrzne**: Go wspiera zarządzanie zależnościami za pomocą `go modules`, co ułatwia integrację zewnętrznych bibliotek.

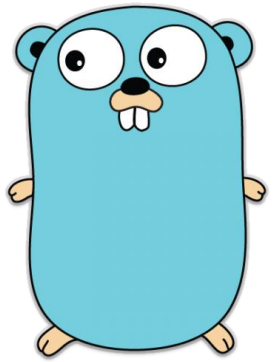


Automatyczne zarządzanie pamięcią

- Go posiada wbudowany mechanizm garbage collection, który automatycznie zarządza pamięcią, zwalniając programistów od konieczności manualnego zarządzania cyklem życia obiektów. To redukuje ryzyko błędów takich jak wycieki pamięci i błędy segmentacji.



Funkcje



- **Go (Golang)**:
  - Funkcje definiowane przy użyciu **`func`**.
  - Przykład: Funkcja **add** zwracająca sumę dwóch liczb.

```go
func greet(name string) string {
    return "Hello, " + name
}

func main() {
    message := greet("Alice")
    fmt.Println(message)
}
```

Klasy

- Go nie posiada tradycyjnych klas, zamiast tego używa struktur (structs).
- Struktury działają jako plany do tworzenia obiektów z określonymi atrybutami i zachowaniami.

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

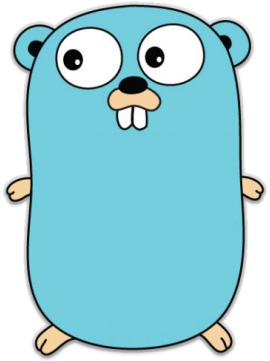
```go
package main

import "fmt"

type Person struct {
    Name string
    Age  int
}

func main() {
    p := Person{Name: "Alice", Age: 30}
    fmt.Println(p)
}
```

Konstruktorzy i tworzenie obiektów

```go
package main

import "fmt"

type Person struct {
    Name string
    Age  int
}

func NewPerson(name string, age int) Pers
    return Person{Name: name, Age: age}
}

func main() {
    p := NewPerson("Alice", 30)
    fmt.Println(p)
}
```

- Go nie ma specjalnych konstruktorów, ale można tworzyć funkcje, które inicjalizują obiekty.
- Przykład: Tworzenie funkcji inicjalizującej strukturę.

Zastosowanie języka



- 1. Rozwój aplikacji Go
- 2. Debugowanie
- 3. Testowanie
- 4. Integracja z narzędziami zewnętrznymi5. Kontrola wersji
- 6. Refaktoryzacja kodu
- 7. Analiza statyczna
- 8. Profilowanie aplikacji9. Wsparcie dla wielu platform

Koniec