

Опциональные значения в моделях

Цель

В некоторых случаях передача всех полей модели от клиента не имеет смысла. Например, клиент хочет обновить только наименование какой-то сущности. Необходимо предоставить простой программный интерфейс для объявления таких полей, получения значений только заполненных полей перед выполнением запроса к базе данных.

Пакет

gitlab.corp.rarus.cloud/rarus-lms/backend/pkg/optional

Пакет предоставляет обобщенный тип `optional.Optional[T any]`, который оборачивает другой тип и хранит указатель на значение этого типа. Таким образом можно легко различить наличие и отсутствие значения в опциональном поле.

Например, имеем модель клиента, у которого есть необязательное для передачи поле `age`:

```
type Client struct {
    Id      uint64          `json: "id" `
    Name    string         `json: "name" `
    Age     optional.Optional[uint8] `json: "age" `
}
```

При входном JSON отсутствующим полем `age`, поле не имеет значения:

```
jsonData := []byte(`{
    "id": 1,
    "name": "Василий",
}`)
c := Client{}
json.Unmarshal(jsonData, &c) // Обертка использует парсер внутреннего типа
```

```

c.Age.IsSet() // если значение не установлено - false

v, err := c.Age.GetValue() //если значение == nil, возвращает error

c.Age.Value // указатель на внутренний тип, *uint8 в данном случае

v = c.Age.OrDefault(30) // если значение не установлено, вернет значение по-умолчанию
// v == 30, но c.Age.IsSet() == false

c.Age.SetValue(30) // Теперь значение изменилось и изменения будут учтены при получении
map[string]any

```

Для записи в базу не фиксированного набора значений, обычно нужно получить ключи и значения из структуры. Для этого можно использовать функции:

```

jsonData := []byte(`{
  "id": 1,
  "name": "Василий",
}`)

c := Client{}
json.Unmarshal(jsonData, &c)

m := optional.StructToMap(c) // здесь будет map[string]any, но содержащая только обязательные
поля и Optional поля с заполненными значениями
// m = {"id":1, "name":"Василий"}

k, v := optional.GetKeysAndValues(m) // далее можно получить ключи и значения, для передачи в
аргументы драйвера к базе
// k = ["id", "name"]
// v = [1, "Василий"]

squirrel.Update("some_table").Columns(k).Values(v).Where(...)

```

Функция `optional.StructToMap` формирует ключи хэш-таблицы по следующим правилам:

- Приватные поля всегда игнорируются
- Если у поля не указаны тэги - оно игнорируется
- Если у поля указан тэг ``db: "some_name"``, ключ берется из него
- Если у поля указан тэг ``json: "some_name"``, но нет тэга ``db``, ключ берется из него

