2.6. 文件验证

按照设计，MongoEngine 在将文档插入 MongoDB 之前严格验证文档，并确保它们与模型中定义的字段一致。

MongoEngine 假设数据库中存在的文档符合模式。这意味着当对象从数据库加载到模型实例时，Mongoengine 不会验证文档，但在某些情况下此操作可能会失败（例如，如果从数据库中获取的文档中有一个字段未在你的模型）。

2.6.1. 内置验证

Mongoengine 提供了不同的字段，这些字段封装了开箱即用的相应验证。调用*.validate()*或*.save()时运行验证*

```
from mongoengine import Document, EmailField, IntField

class User(Document):
    email = EmailField()
    age = IntField(min_value=0, max_value=99)

user = User(email='invalid@', age=24)
user.validate()     # raises ValidationError (Invalid email address: ['email'])
user.save()         # raises ValidationError (Invalid email address: ['email'])

user2 = User(email='john.doe@garbage.com', age=1000)
user2.save()        # raises ValidationError (Integer value is too large: ['age'])
```

2.6.2. 自定义验证

以下功能可用于自定义验证：

字段*验证*参数

```
def not_john_doe(name):
    if name == 'John Doe':
        raise ValidationError("John Doe is not a valid name")

class Person(Document):
    full_name = StringField(validation=not_john_doe)

Person(full_name='Billy Doe').save()
Person(full_name='John Doe').save()  # raises ValidationError (John Doe is not a valid
name)
```

文档*清理*方法

此方法被称为`save()`并应用于提供自定义模型验证和/或在验证之前修改某些字段值。例如，您可以使用它自动为字段提供值，或者进行需要访问多个字段的验证。

```
class Essay(Document):
    status = StringField(choices=('Published', 'Draft'), required=True)
    pub_date = DateTimeField()

    def clean(self):
        # Validate that only published essays have a `pub_date`
        if self.status == 'Draft' and self.pub_date is not None:
            raise ValidationError('Draft entries should not have a publication date.')
        # Set the pub_date for published items if not set.
        if self.status == 'Published' and self.pub_date is None:
            self.pub_date = datetime.now()
```

笔记

只有在打开验证和调用`save()`.

添加自定义 Field 类

我们建议尽可能使用 MongoEngine 提供的字段。但是，也可以通过重写*validate*方法来子类化 Field 并封装一些验证

```
class AgeField(IntField):

    def validate(self, value):
        super(AgeField, self).validate(value)     # let IntField.validate run first
        if value == 60:
            self.error('60 is not allowed')

class Person(Document):
    age = AgeField(min_value=0, max_value=99)

Person(age=20).save()   # passes
Person(age=1000).save() # raises ValidationError (Integer value is too large: ['age'])
Person(age=60).save()   # raises ValidationError (Person:None) (60 is not allowed:
['age'])
```

笔记

重写*validate*时，使用*self.error("your-custom-error")*而不是显式引发 ValidationError，它将提供更好的错误消息上下文

2.6.3. 跳过验证

尽管不鼓励这样做，因为它允许违反字段约束，但如果出于某种原因您需要在调用时禁用文档的验证和清理`save()`，您可以使用*.save(validate=False)*。

```
class Person(Document):
    age = IntField(max_value=100)

Person(age=1000).save()    # raises ValidationError (Integer value is too large)

Person(age=1000).save(validate=False)
person = Person.objects.first()
assert person.age == 1000
```