



Contents

- Namespace
- Class
- Database - Query



Namespace



Namespace

```
1 a = 3
2 print a
3 a = 5
4 print a
```



Namespace

```
1 import random
2
3 print random.random()
4 random = 10
5 print random.random()
```



Namespace

```
1 import random
2
3 print random.shuffle
4 shuffle = 10
5 print random.shuffle
```



Namespace

- 이름을 구분할 수 있는 범위
- 하나의 네임스페이스에서는 하나의 이름이 하나의 개체만을 가리킴
- 모듈, 클래스 등은 각각 하나의 네임스페이스가 됨



Class



Class

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    user = db.relationship('User', foreign_keys = [user_id])
    wall_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    wall = db.relationship('User', foreign_keys = [wall_id],
        backref = db.backref('wall_posts', cascade = 'all, delete-orphan', lazy = 'dynamic'))
    body = db.Column(db.Text())
    edited_time = db.Column(db.DateTime, default = db.func.now(), onupdate = db.func.now())
    created_time = db.Column(db.DateTime, default = db.func.now())
    is_edited = db.Column(db.Boolean, default = '0')
    is_secret = db.Column(db.Boolean, default = '0')
```



Class

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    user = db.relationship('User', foreign_keys = [user_id])
    wall_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    wall = db.relationship('User', foreign_keys = [wall_id],
        backref = db.backref('wall_posts', cascade = 'all, delete-orphan', lazy = 'dynamic'))
    body = db.Column(db.Text())
    edited_time = db.Column(db.DateTime, default = db.func.now(), onupdate = db.func.now())
    created_time = db.Column(db.DateTime, default = db.func.now())
    is_edited = db.Column(db.Boolean, default = '0')
    is_secret = db.Column(db.Boolean, default = '0')
```



class?



Class, Instance

사람 : class



Class, Instance

사람 : class

홍지호 : instance



Class, Instance

사람은 이름, 나이, 키, 몸무게, 성별, ... 등의 속성이 있다



Class, Instance

#사람은 이름, 나이, 키, 몸무게, 성별,
... 등의 속성이 있다

```
class Person:  
    name = None  
    age = None  
    height = None  
    weight = None  
    gender = None
```



Class, Instance

#사람은 이름, 나이, 키, 몸무게, 성별,
... 등의 속성이 있다

```
class Person:  
    name = None  
    age = None  
    height = None  
    weight = None  
    gender = None
```

```
jiho = Person()
```



Class, Instance

```
# jiho : Person 클래스의 instance  
jiho = Person()
```



Class, Instance

```
jiho = Person()  
alex = Person()
```

```
jiho.name = "jiho"  
alex.name = "alex"
```

```
print jiho.name  
print alex.name
```



__init__()

- 새로운 인스턴스를 생성할 때 사용하는 함수



`__init__()`

```
jiho = Person()  
alex = Person()
```

```
jiho.name = "jiho"  
alex.name = "alex"
```

```
print jiho.name  
print alex.name
```



`__init__()`

```
jiho = Person()  
alex = Person()
```

```
jiho.name = "jiho"  
alex.name = "alex"
```

```
print jiho.name  
print alex.name
```



__init__()

- 새로운 인스턴스를 생성할 때 사용하는 함수
- 클래스 내의 변수들의 값을 초기화해줌



`__init__()`

```
class Person:  
    name = None  
    age = None  
    height = None  
    weight = None  
    gender = None
```

```
jiho = Person()
```



__init__()

```
class Person:
    name = None
    age = None
    height = None
    weight = None
    gender = None

    def __init__(self, name = None,
                  age = None, height = None,
                  weight = None, gender = None):
        self.name = name
        self.age = age
        self.height = height
        self.weight = weight
        self.gender = gender
```



__init__()

```
class Person:
    # 필요없어짐. __init__ 함수에서 만들어주기 때문에
    '''
    name = None
    age = None
    height = None
    weight = None
    gender = None
    '''

    def __init__(self, name = None,
                  age = None, height = None,
                  weight = None, gender = None):
        self.name = name
        self.age = age
        self.height = height
        self.weight = weight
        self.gender = gender
```



__init__()

```
class Person:
    def __init__(self, name = None,
                  age = None, height = None,
                  weight = None, gender = None):
        self.name = name
        self.age = age
        self.height = height
        self.weight = weight
        self.gender = gender

jiho = Person(
    name = "jiho",
    age = 26,
    height = 183,
    weight = 67,
    gender = "M"
)
print jiho.name
```



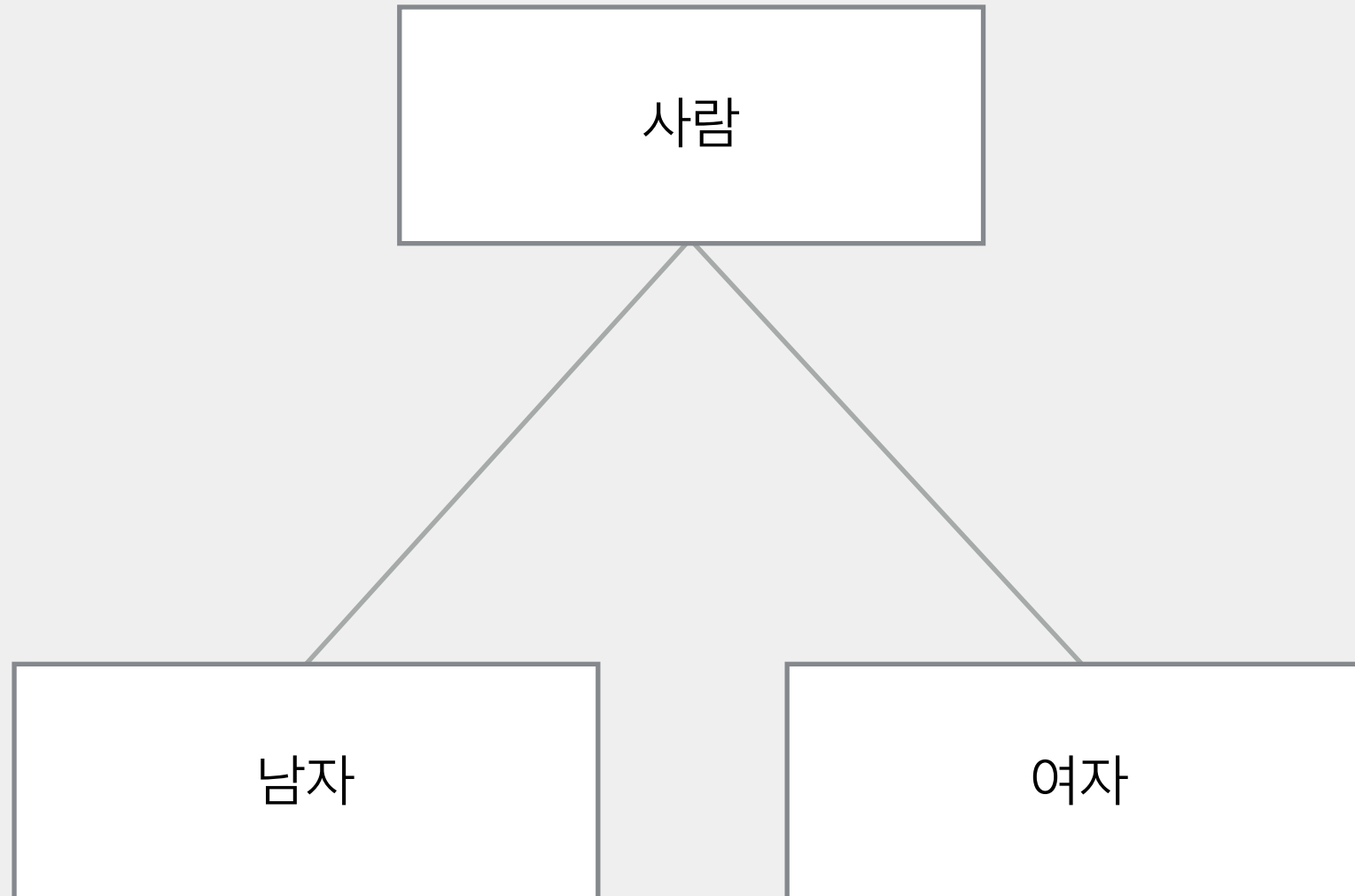
Inheritance

상속



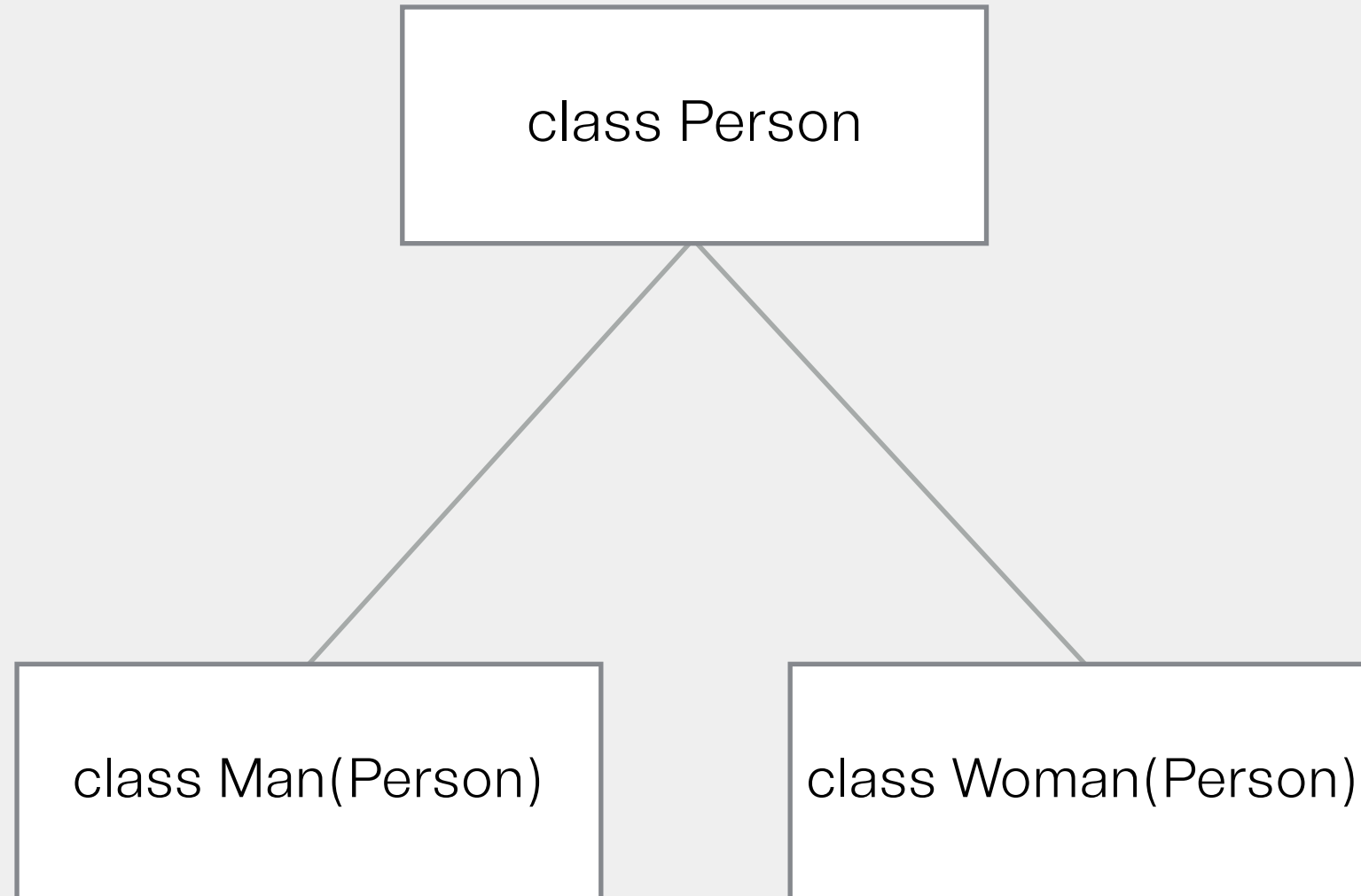
Inheritance

상속



Inheritance

상속



Inheritance

상속

```
class Post(db.Model):
    id = db.Column(db.Integer, primary_key = True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    user = db.relationship('User', foreign_keys = [user_id])
    wall_id = db.Column(db.Integer, db.ForeignKey('user.id'))
    wall = db.relationship('User', foreign_keys = [wall_id],
        backref = db.backref('wall_posts', cascade = 'all, delete-orphan', lazy = 'dynamic'))
    body = db.Column(db.Text())
    edited_time = db.Column(db.DateTime, default = db.func.now(), onupdate = db.func.now())
    created_time = db.Column(db.DateTime, default = db.func.now())
    is_edited = db.Column(db.Boolean, default = '0')
    is_secret = db.Column(db.Boolean, default = '0')
```



Static

- 인스턴스의 속성이 아닌, 클래스 자체의 속성



Static

```
user = User(  
    email = data['email'],  
    username = data['username'],  
    gender = data['gender'],  
    password = db.func.md5(data['password']),  
    mobile = data['mobile'],  
    birthday = data['birthday']  
)
```



Static

```
user = User(  
    email = data['email'],  
    username = data['username'],  
    gender = data['gender'],  
    password = db.func.md5(data['password']),  
    mobile = data['mobile'],  
    birthday = data['birthday']  
)  
  
print user.name  
print user.email  
  
user.mobile = '010-4573-6906'
```



Static

```
user = User.query.get(id)
```



Static

```
user = User.query.get(id)
```



...



Database - Query



Query

- 데이터베이스에서 데이터를 가져오기 위해 쿼리 사용



Query

```
class User(db.Model):  
    id = db.Column(db.Integer, primary_key = True)  
    email = db.Column(db.String(60))  
    username = db.Column(db.String(45))  
    gender = db.Column(db.Enum('F', 'M'))  
    password = db.Column(db.String(100))  
    mobile = db.Column(db.String(15))  
    birthday = db.Column(db.Date)
```



Query

```
class User(db.Model):  
    id = db.Column(db.Integer, primary_key = True)  
    email = db.Column(db.String(60))  
    username = db.Column(db.String(45))  
    gender = db.Column(db.Enum('F', 'M'))  
    password = db.Column(db.String(100))  
    mobile = db.Column(db.String(15))  
    birthday = db.Column(db.Date)
```

id	username	gender	password	mobile	birthday	email
----	----------	--------	----------	--------	----------	-------



Query

```
class User(db.Model):  
    id = db.Column(db.Integer, primary_key = True)  
    email = db.Column(db.String(60))  
    username = db.Column(db.String(45))  
    gender = db.Column(db.Enum('F', 'M'))  
    password = db.Column(db.String(100))  
    mobile = db.Column(db.String(15))  
    birthday = db.Column(db.Date)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query

- User.query : user 테이블의 row들을 가져옴
- 쿼리 결과 클래스의 인스턴스
- 쿼리 결과 클래스가 가지고 있는 함수 : get(), filter(), order_by(), limit()



User.query.get(pk)



User.query.get(pk)

```
user = User.query.get(8)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.get(pk)

```
user = User.query.get(8)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.get(pk)

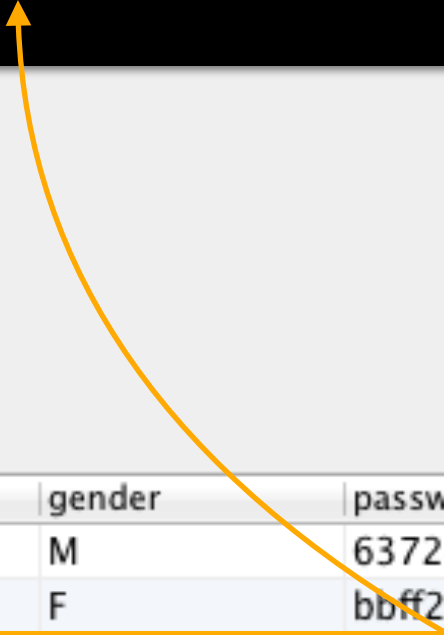
```
user = User.query.get(8)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.get(pk)

```
user = User.query.get(8)
```



id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.get(pk)

```
user = User.query.get(8)  
  
print user.name  
print user.email
```



User.query.get(pk)

- User.query.get(pk)
 - user 테이블의 row 하나를 User 클래스의 인스턴스 형태로 리턴
 - Primary key로 row를 찾음



User.query.filter(조건)



User.query.filter(조건)

```
User.query.filter(  
    User.gender == 'M',  
    User.birthday < '1990-01-01'  
)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.filter(조건)

```
User.query.filter(  
    User.gender == 'M',  
    User.birthday < '1990-01-01'  
)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.filter(조건)

```
User.query.filter(  
    User.gender == 'M',  
    User.birthday < '1990-01-01'  
)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.filter(조건)

```
User.query.filter(  
    User.gender == 'M',  
    User.birthday < '1990-01-01'  
)
```

id	username	gender	password	mobile	birthday	email
6	홍지호	M	6372434fa6b...	010-4573-6906	1989-01-26	s_polaris@nav...
7	남다예	F	bbff2878378c...	010-4010-8075	1993-04-29	namdy0429@...
8	alexhan46	M	a700db7a6cfc...	010-1234-1234	1991-04-06	polaris341@g...
9	alexhan46	M	a700db7a6cfc...	010-1234-1234	1000-04-06	polaris34@gm...
10	우동	M	313cb4f59b1...	01029499451	1994-05-01	ldw9451@gm...
11	monica	F	1097fa0db5d...	01000000000	1990-07-06	k.young706@...
12	hellohello	F	cbe3d1cdc7e...	01065350305	1988-03-01	whatssun@gm...
13	장진욱	M	45d136a1599...	01064214990	1990-04-10	j900410@nav...
14	이혜리	F	4f63c6d1549...	01092237846	2014-12-31	gpfl7846@gm...
15	segg	M	49dec5fb8af4...	010-1234-1234	1991-05-20	a@naver.com
16	바보	F	1adbb317859...	01010101010	0000-00-00	33@gmail.com
17	alexhan46	M	a700db7a6cfc...	01087611661	1991-04-06	alexhan46@g...



User.query.filter(조건)

- user 테이블에서 조건에 맞는 row들을 모두 가져옴
- User.query.filter() 함수의 리턴값 : 쿼리 결과 클래스의 인스턴스



User.query.filter(조건)

```
user = User.query.filter(      ).first()  
user = User.query.filter(      ).one()  
users = User.query.filter(      ).all()
```



.first(), .one(), .all()

- (쿼리 결과).first()
 - 쿼리 결과 중 첫번째 row를 리턴
 - User.query.filter(조건).first() : User 클래스의 인스턴스 리턴



.first(), .one(), .all()

- (쿼리 결과).first()
 - 쿼리 결과 중 첫번째 row를 리턴
 - User.query.filter(조건).first() : User 클래스의 인스턴스 리턴
- (쿼리 결과).one()
 - 쿼리 결과가 하나인 경우 해당 row 리턴
 - 쿼리 결과가 하나가 아닌 경우 에러



.first(), .one(), .all()

- (쿼리 결과).first()
 - 쿼리 결과 중 첫번째 row를 리턴
 - User.query.filter(조건).first() : User 클래스의 인스턴스 리턴
- (쿼리 결과).one()
 - 쿼리 결과가 하나인 경우 해당 row 리턴
 - 쿼리 결과가 하나가 아닌 경우 에러
- (쿼리 결과).all()
 - 쿼리 결과 전체를 리스트 형태로 리턴
 - User.query.all()
 - user 테이블의 전체 내용이 들어있는 리스트 리턴
 - 리턴하는 리스트에는 User 클래스의 인스턴스들이 들어있음



Post.query.order_by(정렬 기준)



Post.query.order_by(정렬 기준)

```
Post.query.filter(  
    Post.wall_id == 10  
) .order_by(  
    db.desc(Post.edited_time)  
)
```

id	user_id	wall_id	body	edited_time	created_time	is_edited	is_secret
31	11	10	<p>열심히일하...	2014-08-07 08:24:54	2014-08-07...	0	0
15	8	10	<p>후훗</p>	2014-08-06 15:05:00	2014-08-06...	0	1
14	6	10	<p>asdfasdfs...	2014-08-06 15:04:07	2014-08-06...	0	1



Post.query.order_by(정렬 기준)

```
Post.query.filter(  
    Post.wall_id == 10  
) .order_by(  
    db.desc(Post.edited_time)  
)
```

id	user_id	wall_id	body	edited_time	created_time	is_edited	is_secret
31	11	10	<p>열심히일하...	2014-08-07 08:24:54	2014-08-07...	0	0
15	8	10	<p>후훗</p>	2014-08-06 15:05:00	2014-08-06...	0	1
14	6	10	<p>asdfasdfs...	2014-08-06 15:04:07	2014-08-06...	0	1



Post.query.order_by(정렬 기준)

```
Post.query.filter(  
    Post.wall_id == 10  
)  
    .order_by(  
        db.desc(Post.edited_time)  
    )
```

id	user_id	wall_id	body	edited_time	created_time	is_edited	is_secret
31	11	10	<p>열심히일하...	2014-08-07 08:24:54	2014-08-07...	0	0
15	8	10	<p>후훗</p>	2014-08-06 15:05:00	2014-08-06...	0	1
14	6	10	<p>asdfasdfs...	2014-08-06 15:04:07	2014-08-06...	0	1



Post.query.order_by(정렬 기준)

- '쿼리 결과'를 기준에 따라 정렬
- 쿼리 결과 클래스의 인스턴스 리턴
- `posts = Post.query.order_by(db.desc(Post.edited_time)).all()`
 - `posts` 변수에는 전체 글들이 수정 시간(`edited_time`) 역순으로 정렬되어 리스트 형태로 들어감



Post.query.limit(최대 결과 수)



Post.query.limit(최대 결과 수)

Post.query

id	user_id	wall_id	body	edited_time	created_time	is_edited	is_secret
3	6	6	<p>미ㄸㅈㄴㄴ...	2014-08-06...	2014-08-06...	0	0
4	7	6	<p>àbwerwer...	2014-08-06...	2014-08-06...	0	0
6	8	8	<p>test</p>	2014-08-06...	2014-08-06...	0	0
7	8	6	<p>test</p>	2014-08-06...	2014-08-06...	0	0
8	6	8	<p>멍청아 ㅋㅋ...	2014-08-06...	2014-08-06...	0	0
9	11	8	<p>Babo</p>	2014-08-06...	2014-08-06...	0	0
10	6	8	<p>왜 년 이상...	2014-08-06...	2014-08-06...	0	0
11	10	6	<p>지호님 안녕...	2014-08-06...	2014-08-06...	0	0
14	6	10	<p>asdfasdfs...	2014-08-06...	2014-08-06...	0	1
15	8	10	<p>후훗</p>	2014-08-06...	2014-08-06...	0	1
16	6	6	<p>asdfasdf...	2014-08-06...	2014-08-06...	0	0
17	6	6	<p><span cla...	2014-08-06...	2014-08-06...	0	0
18	6	6	<p>asdfsadf...	2014-08-06...	2014-08-06...	0	0
19	6	6	<p>asdfagwe...	2014-08-06...	2014-08-06...	0	0
20	6	6	<p>aw;kla</...>	2014-08-06...	2014-08-06...	0	0
21	6	6	<div contente...	2014-08-07...	2014-08-06...	0	1
22	6	13	<div contente...	2014-08-07...	2014-08-07...	0	0
23	12	12	<p>호지승</p>	2014-08-07...	2014-08-07...	0	0



Post.query.limit(최대 결과 수)

Post.query.limit(5)

id	user_id	wall_id	body	edited_time	created_time	is_edited	is_secret
3	6	6	<p>미 ㅏ ㅓ ㅓㅓ...	2014-08-06...	2014-08-06...	0	0
4	7	6	<p>àbwerwer...	2014-08-06...	2014-08-06...	0	0
6	8	8	<p>test</p>	2014-08-06...	2014-08-06...	0	0
7	8	6	<p>test</p>	2014-08-06...	2014-08-06...	0	0
8	6	8	<p>멍청아 ㅋㅋ...	2014-08-06...	2014-08-06...	0	0



Post.query.limit(최대 결과 수)

- 쿼리 결과 중에서 최대 결과 수 만큼의 row들만 가져옴
- 쿼리 결과 클래스의 인스턴스 리턴



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```

Post 테이블의



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```

데이터를 가져와서



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```

wall_id가 10인 row들만 걸러내서



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```

수정 시간의 역순으로 정렬한 결과의



filter() + order_by() + limit()

```
Post.query.filter(Post.wall_id == 10).order_by(db.desc(Post.edited_time)).limit(5)
```

처음 다섯개



...



실습





- 타임라인에 들어가면 처음부터 글을 보여주지 않고, AJAX 요청을 보내 최근 다섯 개의 글만 가져와서 보여주기



과제



과제

- Autoload

