

## 1. Overview

이 프로젝트의 주제는 온라인 서점 웹사이트를 만드는 것이다. Django와 그 내장 DB인 SQLite를 이용하여 직접 데이터베이스를 구축해보고, 그것을 웹앱에 활용해보는 것이 이 프로젝트의 목적이다. 이번 프로젝트에서 구현한 기능은 총 네 가지로 다음과 같다.

- 기본 유저 기능

- (1) 회원가입
- (2) 로그인 / 로그아웃(세션 유지)
- (3) 회원정보 수정

- 책 조회 기능

- (1) 전체 책 목록 / 카테고리별 책 목록 조회
- (2) 가격 순 정렬
- (3) 각 책의 세부 페이지

- 구매 관련 기능

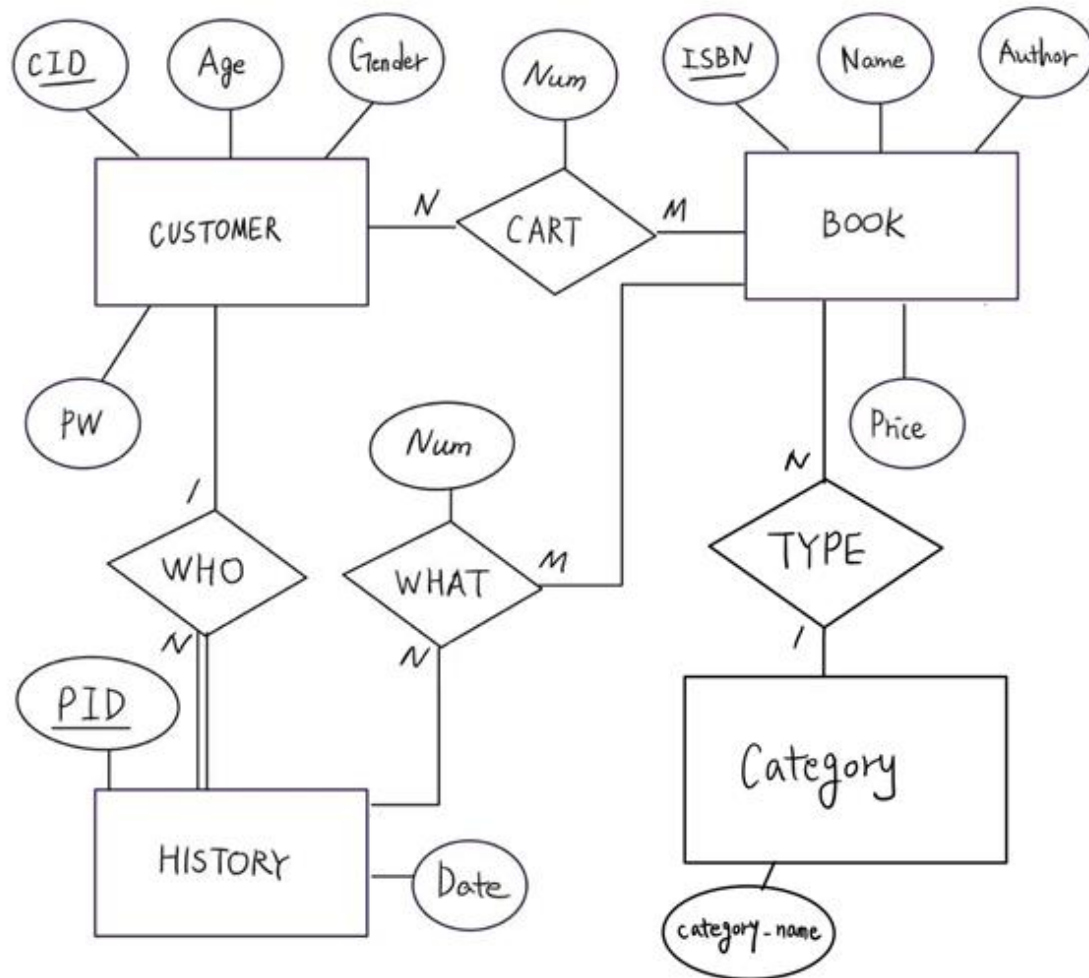
- (1) 책 세부 페이지에서 수량 결정 후 북카트에 넣기
- (2) 북카트 조회 및 총액 확인
- (3) 북카트에서 항목 선택 및 제거
- (4) 북카트 주문
- (5) 구매 내역 확인

- 책 추천 기능

- (1) 메인 화면에서 유저의 나이/성별에 따라 해당 그룹에서 판매량 제일 높은 책

세 권 추천해준다.

## 2. ER Diagram



### (1) CUSTOMER

회원 정보를 저장하는 Entity이다. Customer entity는 CID(key), PW, Age, Gender로 총 4개의 attribute를 가진다. Age와 Gender는 책 추천을 위해 입력 받는 정보이다. CID와 PW는 로그인을 위해 필요한 정보이다. 이때 CID는 고유한 값이므로 CID를 key로 했다.

### (2) BOOK

책 정보를 저장하는 Entity이다. Book entity는 ISBN(key), Name, Author, Price로 총

5개의 attribute를 가진다. ISBN은 책 고유 식별 번호로, 한국에선 13자리 ISBN을 사용한다. 고유 식별 번호이기 때문에 key로 사용하기에 적합하다고 생각해서 key로 사용하였다. Name, Author, Price에는 책에 대한 기본적인 정보들이 담긴다.

### (2.1) CART

Cart relationship은 어떤 사람이 무슨 책을 본인의 북카트에 넣었는지를 나타낸다. 한 고객은 카트 하나에 여러 책을 넣을 수 있고, 한 책은 여러 사람의 카트에 담길 수 있다. 따라서 N:M 관계이다. 고객이 같은 책을 여러 권 담을 수도 있으므로 Num attribute를 추가하였다.

### (3) HISTORY

History entity는 지난 구매 내역을 저장한다. PID(key)와 Date의 2 attribute를 가지고 있다. PID는 purchase ID로, 구매할 때마다 해당 구매 내역에 자동으로 매겨지는 번호이다. Date는 해당 책을 구매한 날짜이다.

#### (3.1) WHO

각 구매 내역이 어떤 사람에게 속하는지 나타내는 relationship이다. 이때 한 사람은 여러 구매 내역을 가질 수 있으나, 한 구매 내역은 여러 사람에게 속할 수 없다. 그리고 당연히 고객이 존재해야 그 고객의 구매내역이 존재하는 법이다. 그래서 1:N 관계를 가지고, HISTORY는 이 relationship에 total participate한다.

#### (3.2) WHAT

각 구매 내역에서 어떤 책을 샀는지 나타내는 relationship이다. 한 책은 여러 구매 내역에서 구매될 수 있고, 한 구매 내역에는 여러 책이 포함될 수 있다. 따라서 N:M 관계를 가진다. 또 한 번에 여러 책을 사는 것도 가능하므로 Num attribute도 추가했다.

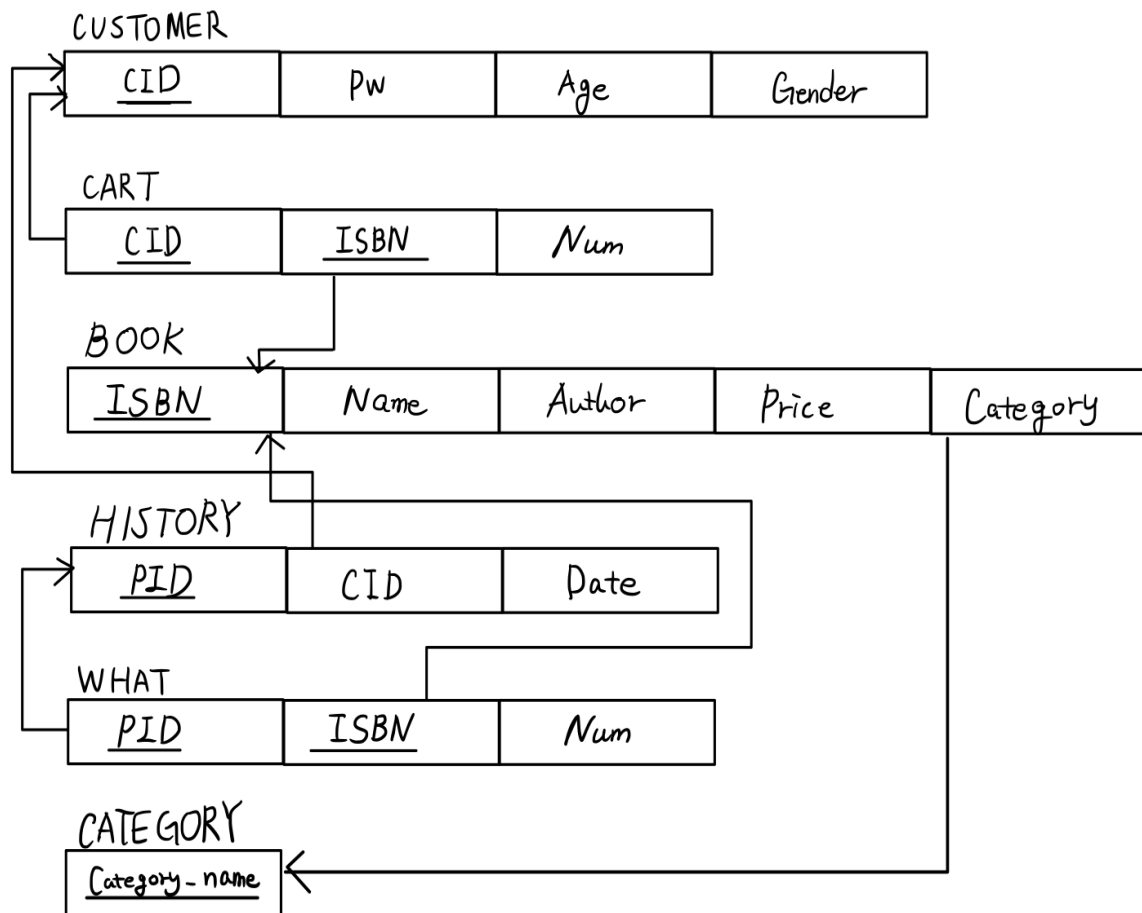
### (4) CATEGORY

카테고리의 종류를 저장하는 entity이다.

#### (4-1) TYPE

책이 어떤 카테고리에 속하는지를 나타내는 relationship이다.

### 3. Relational Data Model



#### (1) CUSTOMER

회원 정보를 저장하는 table이다.

CID : 아이디이자 primary key이다. 최대 20자를 가질 수 있다.

PW : 비밀번호이다. 최대 20자를 가질 수 있다.

Birthyear : 태어난 년도이다. 기본 값은 1990이다.

Gender : 성별이다. F(Female), M(Male), U(Unknown) 중 하나를 선택할 수 있게 하였다.

## (2) CART

북카트 정보를 저장하는 table이다. 각각의 row는 "어떤 회원이 어떤 책을 얼마나 카트에 넣었다"는 것을 표현한다.

CID : 해당 책을 북카트에 넣은 유저의 아이디이다. Customer table을 참조한다.

ISBN : 해당 책의 고유 ISBN이다. Book table을 참조한다.

Num : 해당 책을 넣은 수량이다.

(CID, ISBN) 쌍은 중복될 수 없다. 한 북카트에 책을 더 넣으면 Num만 늘어날 뿐이다. 따라서 (CID, ISBN) 쌍을 primary key로써 사용했다.

## (3) BOOK

책 정보를 저장하는 table이다.

ISBN : 책 고유의 ISBN을 저장한다. ISBN은 고유하므로 primary key로 사용한다.

Name : 책 제목을 저장한다.

Author : 작가 이름을 저장한다.

Price : 책 가격을 저장한다.

Category : 해당 책이 포함되는 카테고리이다. Category table을 참조한다.

## (4) HISTORY

구매 내역을 저장하는 table이다.

PID : 각 구매 내역의 고유 ID (primary key)이다.

CID : 해당 구매를 한 유저의 ID이다. Customer table을 참조한다.

Date : 구매 날짜를 저장한다.

## (5) WHAT

어떤 구매 내역에 대해 어떤 책을 샀는지를 저장하는 table이다.

PID : 어떤 구매 내역에 대한 정보인지를 저장한다. HISTORY table을 참조한다.

ISBN : 어떤 책을 샀는지를 저장한다. BOOK table을 참조한다.

Num : 해당 책을 얼마나 샀는지를 저장한다.

(PID, ISBN) 쌍은 중복될 수 없다. 왜냐하면 구매라는 건 한 북카트에 든 걸 통째로 구매하는 행위인데, 한 북카트에 동일한 ISBN이 여러 번 들어갈 수 없기 때문이다.(여러 권을 구매한다면 Num만 늘어나도록 구현한다) 따라서 (PID, ISBN) 쌍을 primary key 로써 사용했다.

#### 4. Implementation and Results

우선 데이터베이스 구축을 위해 models.py를 작성하였다. 3에서 설명한 것처럼 모델들을 구성하였다. 코드는 다음과 같다.

```
import datetime
from django.db import models
from django.utils import timezone
from django.core.validators import import MaxValueValidator, MinValueValidator

class Category(models.Model):
    name = models.CharField(max_length=30)

class Customer(models.Model):
    cid = models.CharField(max_length=20, primary_key=True)
    pw = models.CharField(max_length=20)
    birthyear =
models.PositiveIntegerField(validators=[MinValueValidator(1900),
MaxValueValidator(9999)],default=1990)
    GENDERS = (('F', 'Female'), ('M', 'Male'),('U', 'Unknown'))
    gender = models.CharField(max_length=1, default='U', choices=GENDERS)

class Book(models.Model):
    isbn = models.CharField(max_length=13, primary_key=True)
    name = models.CharField(max_length=255)
    author = models.CharField(max_length=50)
    price = models.IntegerField(default=0)
    category = models.ForeignKey(Category, on_delete=models.CASCADE)

class Cart(models.Model):
    cid = models.ForeignKey(Customer, on_delete=models.CASCADE)
    isbn = models.ForeignKey(Book, on_delete=models.CASCADE)
    num = models.IntegerField(default=1)
```

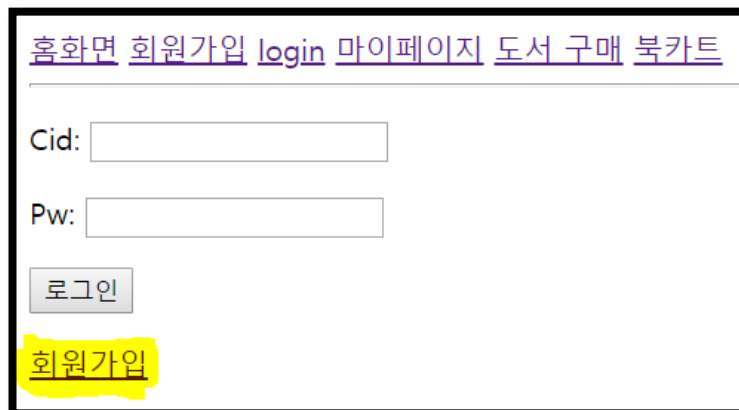
```
class History(models.Model):
    pid = models.AutoField(primary_key=True)
    cid = models.ForeignKey(Customer, on_delete=models.CASCADE)
    date = models.DateTimeField()

class What(models.Model):
    pid = models.ForeignKey(History, on_delete=models.CASCADE)
    isbn = models.ForeignKey(Book, on_delete=models.CASCADE)
    num = models.IntegerField(default=1)
```

각각의 클래스는 각각 3에서 설명한 동명의 table에 해당한다.

- 기본 유저 기능

(1) 회원가입 <http://localhost:8000/signup/>



상단 바 또는 로그인 창 하단부에서 회원가입으로 들어갈 수 있다.

[홈화면](#) [회원가입](#) [login](#) [마이페이지](#) [도서](#)

---

Cid:

Pw:

Birthyear:

Gender:

ID, Pw, Birthyear, Gender를 입력하고 회원가입 할 수 있다.

[홈화면](#) [회원가입](#) [login](#) [마이페이지](#) [도서](#) [구매](#) [북카트](#)

---

- Customer with this Cid already exists.

Cid:

Pw:

- Ensure this value is less than or equal to 9999.

Birthyear:

Gender:

존재하는 아이디로 가입하려고 하거나, 출생년도가 1900 미만 또는 9999 이상이면 회원가입이 되지 않고 경고 메시지가 나오게 된다.

해당하는 코드는 다음과 같다.



home/views.py

```
def signup(request):
    cid = request.session.get('cid', False)
    login = 'logout'
    if cid == False: login = 'login'

    if request.method == 'POST':
        form = CustomerForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect('/')
    else:
        form = CustomerForm()
    return render(request, 'home/signup.html', {'login':login, 'form':form})
```

만약 POST 방식의 request가 들어왔다면 CustomerForm을 받은 후 form이 적절한 형태인지 검사한다. 만약 적절하다면 저장한 후 첫 페이지로 돌아간다.

GET 방식의 request가 들어왔다면 초기화된 CustomerForm을 home/signup.html에 렌더링 한다.

home/forms.py

```
class CustomerForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ['cid', 'pw', 'birthyear', 'gender']
        widgets = {
            'pw': forms.PasswordInput,
        }
```

Customer model을 사용한 Form이다. 이때 비밀번호는 PasswordInput을 사용하게 했다.

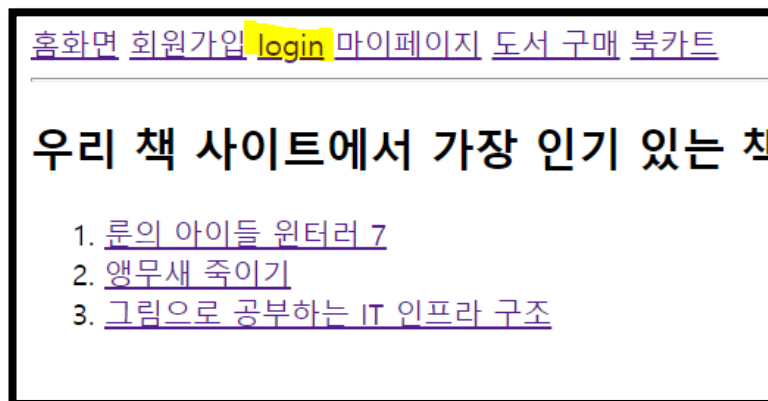
home/templates/home/signup.html

```
{% extends "home/base.html" %}

{% block content %}
    <div>
        <form method="POST">
            {% csrf_token %}
            {{ form.as_p }}
            <button type="submit">저장</button>
        </form>
    </div>
{% endblock content %}
```

전달받은 form을 표시한다. Submit button을 누르면 POST 방식으로 request를 보내게 된다.

(2) 로그인 / 로그아웃(세션 유지) <http://localhost:8000/login/>



상단 바로 들어가거나, 아니면 로그인을 하지 않았는데 로그인이 필요한 페이지(마이페이지, 북카트 등)로 들어가면 로그인 창이 나온다.

[홈화면](#) [회원가입](#) [login](#) [마이페이지](#) [도서](#)

---

Cid:

Pw:

[회원가입](#)

아이디와 비밀번호를 입력하면 된다.

[홈화면](#) [회원가입](#) [login](#) [마이페이지](#) [도서](#) [구독](#)

---

로그인 정보가 올바르지 않습니다.

Cid:

Pw:

[회원가입](#)

만약 틀린 아이디 또는 비밀번호를 넣게 되면, 로그인이 되지 않고 다음과 같은 메시지가 나오게 된다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

wind님을 위한 책 추천,

wind님 또래의 여자가 가장 많이 선택한 책

1. [앵무새 죽이기](#)
2. [룬의 아이들 윈터러 7](#)
3. [세월의 돌 1](#)

로그인이 되면 index 페이지로 돌아가고, 상단 바에 login이 아니라 logout이 표기되게 된다. Session을 유지하기 때문에 로그인이 풀리지 않는다.

만약 로그인이 된 상태라면, 상단 바의 logout을 누르면 로그아웃 되게 된다.

해당하는 코드는 다음과 같다.

home/views.py

```
def login(request):
    cid = request.session.get('cid', False)
    if cid == False: # Login
        msg = ''
        if request.method == 'POST':
            form = LoginForm(request.POST)
            try:
                customer = Customer.objects.get(cid=form.data['cid'],
pw=form.data['pw'])
                request.session['cid'] = form.data['cid']
                return redirect('/')
            except:
                msg = '로그인 정보가 올바르지 않습니다.'
                form = LoginForm()
        else:
            form = LoginForm()
        return render(request, 'home/login.html',
{'login': 'login', 'form': form, 'msg': msg})
    else:
        try:
            del request.session['cid']
        except KeyError:
```

```
pass
return redirect('/')
```

우선 session을 확인한 후 로그인이 되어 있으면 로그아웃 한다(session에서 아이디를 지운다).

로그인이 되어 있지 않고, POST 방식이 아닐 때는 빈 LoginForm을 home/login.html에 렌더링한다.

로그인이 되어 있지 않고, POST 방식이면 LoginForm을 받아온 후 해당하는 회원정보가 있는지, 비밀번호가 그에 일치하는지 확인한다. 확인한 결과 아이디가 존재하고 비밀번호가 일치하면 session에 아이디를 추가한 후 인덱스 페이지로 돌아가고, 아니면 에러 메시지와 함께 새로운 빈 LoginForm을 렌더링한다.

home/forms.py

```
class LoginForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ['cid', 'pw']
        widgets = {
            'pw': forms.PasswordInput,
        }
```

Customer model을 사용한 Form이다. 이때 비밀번호는 PasswordInput을 사용하게 했다.

home/templates/home/login.html

```
{% extends "home/base.html" %}

{% block content %}
    <div>
        {{msg}}
        <form method="POST">
```

```

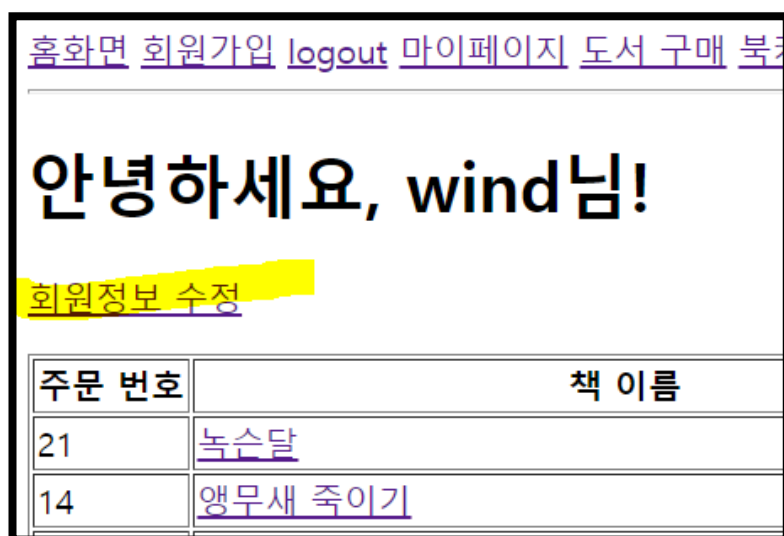
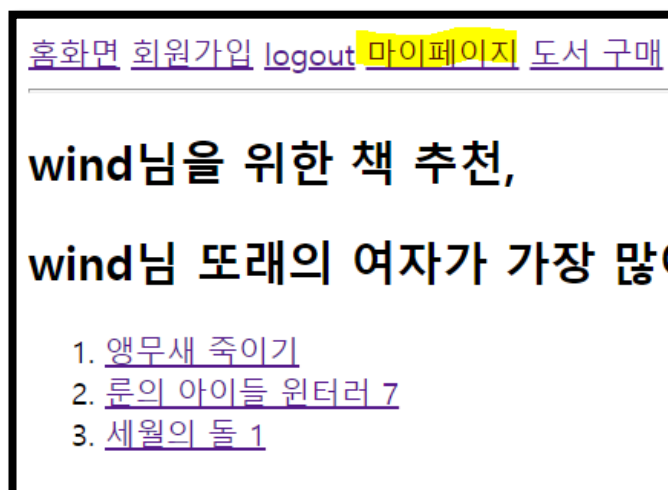
{% csrf_token %}
{{ form.as_p }}
<button type="submit">로그인</button>
</form>
<a href="{% url 'signup' %}">회원가입</a>
</div>

{% endblock content %}

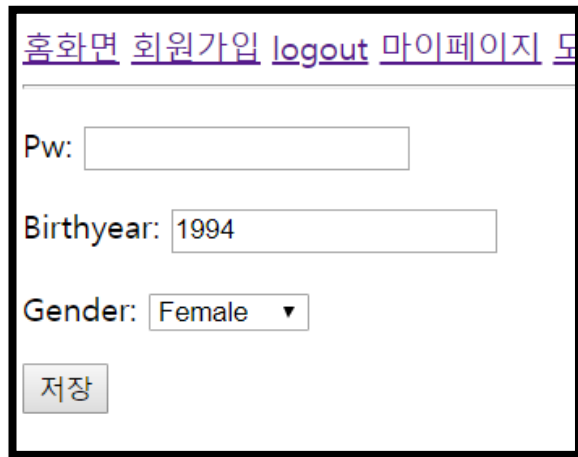
```

전달받은 form을 표시한다. Submit button을 누르면 POST 방식으로 request를 보내게 된다.

### (3) 회원정보 수정

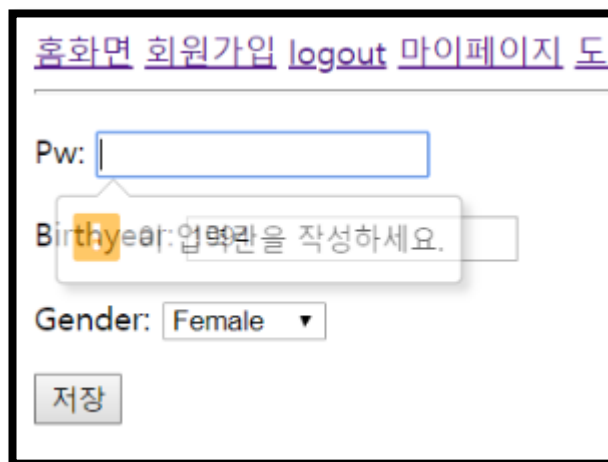


로그인 된 상태에서 마이페이지-회원정보 수정으로 들어가면 회원정보 수정이 가능하다.



The screenshot shows a web form titled '홈화면 회원가입 logout 마이페이지 도'. The form contains three input fields: 'Pw:' (password), 'Birthyear:' (year), and 'Gender:' (dropdown menu). The 'Birthyear' field is filled with '1994' and the 'Gender' dropdown is set to 'Female'. A '저장' (Save) button is at the bottom.

들어가면 비밀번호는 비어 있고, Birthyear과 Gender는 저장해놓은 정보가 나온다. 회원정보를 수정하고 저장을 누르면 바뀐 정보가 저장된다. 이때 Pw란은 반드시 채워야 한다. 채우지 않으면 다음과 같은 메시지가 뜬다.



The screenshot shows the same web form as before, but with an error message displayed. The 'Pw:' field is empty, and a yellow tooltip message says '비밀번호를 작성하세요.' (Please enter a password). The 'Birthyear' and 'Gender' fields remain filled with '1994' and 'Female' respectively. The '저장' (Save) button is still visible.

해당하는 코드는 다음과 같다.

home/views.py

```
def modifyAcc(request):
    cid = request.session.get('cid', False)
    login = 'logout'
    if cid == False: login = 'login'

    customer = Customer.objects.get(cid=cid)
    if request.method == 'POST':
```

```

form = ModifyForm(request.POST)
if form.is_valid():
    customer.pw = form.data['pw']
    customer.birthyear = form.data['birthyear']
    customer.gender = form.data['gender']
    customer.save()
    return redirect('/mypage')
else:
    form = ModifyForm(initial={'birthyear':customer.birthyear,
'gender':customer.gender})
    return render(request, 'home/modifyAcc.html', {'login':login,'form':form})

```

로그인 되어 있지 않으면 로그인 창으로 redirect 해버린다.

POST 방식이면 ModifyForm을 받아들인다. 만약 form의 데이터가 유효하다면 현재 세션의 cid에 해당하는 customer 정보를 바꾸고 mypage로 리다이렉트한다. 그렇지 않다면 새 ModifyForm을 modifyAcc.html에 렌더링한다. 이때 현재 세션의 cid에 해당하는 customer에 저장된 정보로 초기화 시켜서 form을 만든다.

home/forms.py

```

class ModifyForm(forms.ModelForm):
    class Meta:
        model = Customer
        fields = ['pw', 'birthyear', 'gender']
        widgets = {
            'pw': forms.PasswordInput,
        }

```

Customer model을 사용한 Form이다. 이때 비밀번호는 PasswordInput을 사용하게 했다.

home/templates/home/modifyAcc.html

```
{% extends "home/base.html" %}
```



```
{% block content %}
  <div>
    <form method="POST">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">저장</button>
    </form>
  </div>
{% endblock content %}
```

전달받은 form을 표시한다. Submit button을 누르면 POST 방식으로 request를 보내게 된다.

- 책 조회 기능 <http://localhost:8000/books/>

- (1) 전체 책 목록 / 카테고리별 책 목록 조회
- (2) 가격 순 정렬

[홈화면](#)
[회원가입](#)
[logout](#)
[마이페이지](#)
[도서 구매](#)
[북카트](#)

---

[전체](#)

[컴퓨터](#)
[자기계발](#)
[소설](#)

---

[드래곤 라자 세트](#) | 소설 | 이영도 | 96000

[알고리즘 문제 해결 전략 세트](#) | 컴퓨터 | 구종만 | 50000

[Clean Code](#) | 컴퓨터 | 로버트 C. 마틴 | 33000

[호빗](#) | 소설 | J.R.R. 톨킨 | 28000

[그림으로 공부하는 IT 인프라 구조](#) | 컴퓨터 | 야마자키 야스시 | 25000

[인공지능을 위한 수학](#) | 컴퓨터 | 이시카와 아키히코 | 25000

[파이토치 첫걸음](#) | 컴퓨터 | 두세교 | 24000

[밑바닥부터 시작하는 딥러닝](#) | 컴퓨터 | 사이토 고키 | 24000

상단 바의 도서 구매를 누르면 책 목록을 볼 수 있다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북마크](#)

[전체](#)

[컴퓨터](#) [자기계발](#) [소설](#)

-----▼

[아주 작은 습관의 힘](#) | 자기계발 | 제임스 클리어 |

[말센스](#) | 자기계발 | 셀레스트 헤들리 | 14500

[40일 만에 두뇌력 천재가 된다](#) | 자기계발 | 개러스

[내 머릿속 청소법](#) | 자기계발 | 김경록 | 14000

[왓칭 1](#) | 자기계발 | 김상운 | 13000

카테고리를 선택함으로써 원하는 카테고리의 책만 볼 수 있다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북마크](#)

[전체](#)

[컴퓨터](#) [자기계발](#) [소설](#)

-----▼

-----

가격낮은순 | 전민희 | 8000

가격높은순

[죄와 벌](#) | 소설 | 도스토옙스키 | 8500

[세월의 돌 1](#) | 소설 | 전민희 | 10000

[얼음나무 숲](#) | 소설 | 하지은 | 10000

[태양의탑 6](#) | 소설 | 전민희 | 12000

옵션을 선택하면 가격 순 정렬도 가능하다.

해당하는 코드는 다음과 같다.

home/views.py

```
def bookList(request, category=''):
    login = 'logout'
    cid = request.session.get('cid', False)
    if cid == False: login = 'login'

    categories = Category.objects.values('name').distinct()
    books = None

    sort=request.GET.get('sort','')
    if sort == 'lower-price':
        if category == '': books = Book.objects.all()
        else : books = Book.objects.filter(category__name=category)
        books = books.order_by('price')
    elif sort == 'higher-price':
        if category == '': books = Book.objects.all()
        else : books = Book.objects.filter(category__name=category)
        books = books.order_by('-price')
    else :
        if category == '': books = Book.objects.all()
        else : books = Book.objects.filter(category__name=category)
        books = books.order_by('-price')
    return render(request, 'home/books.html',
{'login':login, 'books':books, 'category':categories})
```

우선 카테고리 테이블에서 카테고리 목록을 받아온다. 함수 인자로 카테고리 명을 받게 되는데, 따로 인자를 받지 않으면 Book table의 모든 row를 books 변수에 저장하고 아니면 해당 카테고리에 해당하는 row만 books 변수에 저장한다.

그리고 sort가 어떻게 되어 있는지가 request에 담겨서 오는데, sort 종류에 따라 books를 가격에 대해 다르게 정렬하게 된다.

books를 home/books.html으로 렌더링한 결과를 리턴하게 된다..

home/templates/home/books.html

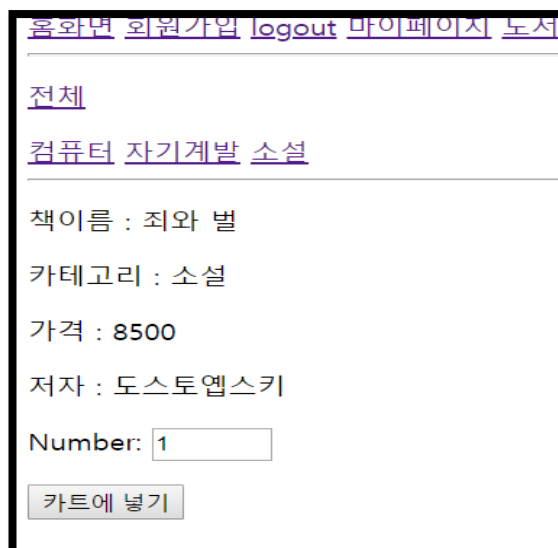
```
{% extends "home/base.html" %}

{% block content %}
{% include 'home/booksCategory.html' %}
    <select id="sort-select"
onchange="location=this.value;" style="width:200px;height:50px;font-
weight:bold;">
        <option class="sort-nothing" value="/">-----</option>
        <option class="sort-lower-price" value="?sort=lower-
price">가격낮은순</option>
        <option class="sort-higher-price" value="?sort=higher-
price">가격높은순</option>
    </select>
    <div>

        {% for obj in books %}
        <p><a href="{% url 'detail' obj.category.name
obj.isbn %}">{{ obj.name }}</a>
        | {{obj.category.name}} | {{obj.author}} | {{obj.price}}</p>
        {% endfor %}
    </div>
{% endblock content %}
```

우선 가격을 선택하기 위한 select 박스를 만든다. 그리고 books의 책들을 표시한다. 페이지에 책 제목, 카테고리명, 작가 이름, 가격이 표시되게 된다. 각 책들은 세부페이지로 가기 위한 링크가 걸리게 된다.

(3) 각 책의 세부 페이지 <http://localhost:8000/books/<category>/<isbn>/>



The screenshot shows a web page with a navigation bar at the top containing links: [홈](#), [회원가입](#), [logout](#), [마이페이지](#), and [도서](#). Below the navigation bar, there are two sub-navigation links: [전체](#) and [컴퓨터 자기계발 소설](#). The main content area displays the following information for a book: '책이름 : 죄와 벌', '카테고리 : 소설', '가격 : 8500', and '저자 : 도스토옙스키'. Below this information, there is a 'Number:' label followed by a text input field containing the value '1'. At the bottom of the form, there is a button labeled '카트에 넣기'.

아까 책 리스트에서 각 책 제목을 클릭하면 그 책의 세부 페이지로 들어가게 된다. 세부 페이지에는 책 정보가 나오고, 카트에 넣을 수량을 결정해서 카트에 넣을 수도 있다.

코드는 다음과 같다.

home/views.py

```
def detail(request, category, isbn):
    try:
        book = Book.objects.get(isbn=isbn, category__name=category)
    except :
        raise Http404("Book does not exist")
    form = AddCartForm()
    cid = request.session.get('cid', False)
    if request.method == 'POST':
        if cid == False :
            return redirect('/login')
        else :
            form = AddCartForm(request.POST)
            number = int(form.data['number'])
            try :
                already = Cart.objects.get(cid_id=cid, isbn_id=isbn)
                already.num += number
                already.save()
            except:
                cart = Cart(cid_id=cid, isbn_id=isbn, num=number)
                cart.save()
            messages.success(request, '책이 카트에 추가되었습니다.')
    else:
        form = AddCartForm()
        login = 'logout'
        if cid == False: login = 'login'
        categories = Category.objects.values('name').distinct()
        return render(request, 'home/detail.html', {'login':login,
            'book':book, 'form':form, 'category':categories})
```

해당 카테고리 와 isbn에 해당하는 책의 정보를 Book table에서 가져온다. 없으면 404 Error를 띄운다. 만약 request가 POST라면 카트에 넣기를 했을 때이다. 이때 로그인 이 안 되어 있으면 로그인 페이지로 리다이렉트하고, 아니면 AddCartForm을 읽어들여 수량을 알아낸다. 만약에 cid가 일치하고

isbn도 일치하는 row가 Cart에 존재한다면 숫자만 그만큼 올린다. 그렇지 않다면 새로운 row를 추가한다.

만약에 POST 방식이 아니라면 새로운 AddCartForm을 만든다. 그리고 detail.html에 렌더링 한다.

home/templates/home/detail.html

```
{% extends "home/base.html" %}

{% block content %}
{% include 'home/booksCategory.html' %}
<div>
  <p> 책이름 : {{ book.name }} </p>
  <p> 카테고리 : {{ book.category.name }}</p>
  <p> 가격 : {{ book.price }} </p>
  <p> 저자 : {{ book.author }} </p>
</div>
<form method="POST" id="addCart">
  {% csrf_token %}
  {{ form.as_p }}
  <input type="submit" value="카트에 넣기">
</form>

{% if messages %}
<ul class="messages">
  {% for message in messages %}
    <li{% if message.tags %} class="{{ message.tags }}" {%
endif %}>{{ message }}</li>
    <li> 카트로 이동 : <a href="{% url 'cart' %}">북카트</a> </li>
  {% endfor %}
</ul>
{% endif %}

{% endblock content %}
```

우선 책 정보를 표시한다. 그리고 AddCartForm을 표시한다. 카트에 넣기 버튼을 누르면 POST 방식으로 form이 전해지게 된다.

만약에 메시지가 있으면(views.py에서 카트에 넣었을 때만 메시지가 표시되도록 했다) 카트로 이동하는 링크가 표시되도록 했다.

home/forms.py

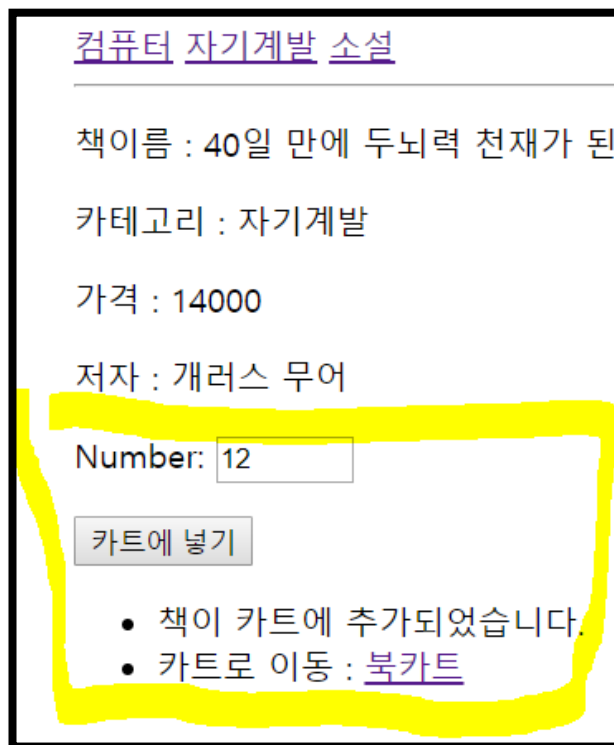
```
class AddCartForm(forms.Form):  
    number = forms.IntegerField(initial=1, min_value=1, max_value=300,  
                                required=True)
```

AddCartForm은 아주 간단한 form으로, 하나의 IntegerField만 존재한다. 이때 한 번에 카트에 넣을 수 있는 최소 수량은 1, 최대 수량은 300으로 설정했다.

- 구매 관련 기능

- (1) 책 세부 페이지에서 수량 결정 후 북카트에 넣기

<http://localhost:8000/books/<category>/<isbn>/>



컴퓨터 자기계발 소설

---

책이름 : 40일 만에 두뇌력 천재가 된

카테고리 : 자기계발

가격 : 14000

저자 : 개러스 무어

Number:

- 책이 카트에 추가되었습니다.
- 카트로 이동 : [북카트](#)

이렇듯 책 세부 페이지에서 수량을 결정한 후 '카트에 넣기' 버튼을 누르면 메시지와 함께 북카트로 이동하는 링크가 생긴다.

해당 코드는 책 조회 기능 — (3) 각 책의 세부 페이지에 기술되어 있으며

로 코드는 생략한다.

- (2) 북카트 조회 및 총액 확인 <http://localhost:8000/cart/>
- (3) 북카트에서 항목 선택 및 제거
- (4) 북카트 주문

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

---

wind님의 북카트입니다

	책 이름	수량	권당 가격
<input type="checkbox"/>	<a href="#">호빗</a>	3권	28000원
<input type="checkbox"/>	<a href="#">40일 만에 두뇌력 천재가 된다</a>	12권	14000원

총액 : 252000원

북카트에 넣은 항목들을 확인할 수 있다. 책 이름을 누르면 역시 각 책의 상세 페이지로 이동 가능하다. 총액이 하단부에 표시된다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

---

wind님의 북카트입니다

	책 이름	수량
<input type="checkbox"/>	<a href="#">호빗</a>	3권
<input checked="" type="checkbox"/>	<a href="#">40일 만에 두뇌력 천재가 된다</a>	12권

총액 : 252000원



항목들을 체크하고 '체크한 항목 삭제하기'를 누르면

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

---

wind님의 북카트입니다

	책 이름	수량	권당 가격
<input type="checkbox"/>	<a href="#">호빗</a>	3권	28000원

총액 : 84000원

[체크한 항목 삭제하기](#)

[전부 구매하기](#)

체크되었던 항목이 카트에서 사라진다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

---

## 구매번호 : 22

구매해주셔서 감사합니다.

구매내역은 [마이페이지](#)에서 확인하실 수 있습니다.

'전부 구매하기'를 누르면 북카트의 모든 책들이 구매되며 위와 같은 화면이 나오게 된다.

home/views.py

```
def cart(request):
    cid = request.session.get('cid', False)
    if cid == False :
        return redirect('/login')

    if request.method == 'POST': # 아이템 제거 명령
        keys = request.POST.keys()
        for key in keys :
            if key == 'csrfmiddlewaretoken' : continue
            Cart.objects.get(cid=cid, isbn=key).delete()

    cart = Cart.objects.filter(cid=cid).select_related('isbn')
    price = 0
    for obj in cart:
        price += obj.num * obj.isbn.price
    return render(request, 'home/cart.html', {'login':'logout', 'cart':cart,
        'cid':cid, 'price':price})
```

POST일 때는 책을 제거해야 할 때이다. 책을 체크해서 제출하면 check box의 이름인 isbn을 키로 하여 정보를 넘겨주게 된다. Csrf\_token일 때는 건너뛰고, 아닐 때는 isbn을 받아들여 현재 유저의 카트에서 해당 isbn을 가진 책을 삭제한다.

그리고 카트를 보여줘야 하므로, 현재 유저의 cid를 가진 카트 데이터를 모두 꺼낸다. 이때 책 정보도 보여줘야 하므로, Book table과 join한다. (isbn 정보를 통하여) 그리고 우선 총액을 계산한다. 그리고 카트 데이터를 home/cart.html에 렌더링한다.

home/templates/home/cart.html

```
{% extends "home/base.html" %}

{% block content %}
    <div>
        <p>{{cid}}님의 북카트입니다</p>

        <div>
            <form method="POST" name="deleteBooks">
```

```

{% csrf_token %}
<table width="500" border="1">
  <tr> <th></th> <th>책 이름</th> <th>수량</th> <th>권당
가격</th></tr>

  {% for obj in cart %}
  <tr>
    <td>
      <input type="checkbox" name="{{obj.isbn.isbn}}">
    </td>
    <td><a href="{% url 'detail' obj.isbn.category.name
obj.isbn.isbn %}">{{obj.isbn.name}}</a></td>
    <td>{{obj.num}}권</td>
    <td>{{obj.isbn.price}}원 </td>
  </tr>
  {% endfor %}
</table>
<p> 총액 : {{price}}원 </p>
<input type="submit" value="체크한 항목 삭제하기">
</form>
</div>
<div>
  <form method="POST" name="buy" action="{% url 'purchase' %}">
    {% csrf_token %}
    <input type="submit" value="전부 구매하기">
  </form>
</div>
{% endblock content %}

```

받은 정보를 for문으로 테이블을 만든다. Form이 두 개 있는데, 하나는 삭제를 위한 form이고 하나는 구매를 위한 form이다. 두 form은 간단한 형태이기 때문에 forms.py에는 추가하지 않았다. 삭제를 위한 form은 체크박스의 데이터를 cart 함수로 넘기게 되고, 구매를 위한 form은 purchase 함수로 POST 방식으로 request를 보내게 된다.

home/views.py

```

def purchase(request):
    cid = request.session.get('cid', False)
    if cid == False :
        return redirect('/login')
    if request.method == 'POST':

```

```

    cart = Cart.objects.filter(cid=cid)
    history = History(cid_id=cid, date=datetime.datetime.now())
    history.save()
    pid = history.pid
    for book in cart:
        what = What(pid_id=pid, isbn=book.isbn, num=book.num)
        what.save()
    cart.delete()
else :
    return redirect('/')
return render(request, 'home/purchase.html', {'login':'logout','pid':pid})

```

구매를 위한 함수이다. cid를 가진 cart의 모든 row를 꺼낸 다음에 history와 what에 그 데이터를 추가한다. 그리고 북카트를 비우기 위해 cid를 가진 cart의 모든 row를 삭제한다. 그리고 구매 번호를 home/purchase.html로 넘겨준다.

home/templates/home/purchase.html

```

{% extends "home/base.html" %}

{% block content %}
<div>
    <p><h1>구매번호 : {{ pid }}</h1></p>
    <p>구매해주셔서 감사합니다.</p>
    <p>구매내역은 <a href="{% url 'mypage' %}">마이페이지</a>에서 확인하실 수
있습니다.</p>
</div>
{% endblock content %}

```

구매번호를 출력하고 마이페이지로의 링크를 제공한다.

## (5) 구매 내역 확인

[옴와면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카드](#)

## 안녕하세요, wind님!

[회원정보 수정](#)

주문 번호	책 이름	수량	주문 날짜
22	<a href="#">호빗</a>	3	June 22, 2019, 11:26 p.m.
21	<a href="#">녹슨달</a>	1	June 20, 2019, 1:18 p.m.
14	<a href="#">앵무새 죽이기</a>	1	May 31, 2019, 11:54 p.m.
13	<a href="#">앵무새 죽이기</a>	1	May 31, 2019, 11:48 p.m.
12	<a href="#">앵무새 죽이기</a>	1	May 31, 2019, 11:48 p.m.
11	<a href="#">2억 빛을 진 내게 우주님이 가르쳐준 윤이 풀리는 말버릇</a>	1	May 31, 2019, 11:22 a.m.
11	<a href="#">호빗</a>	1	May 31, 2019, 11:22 a.m.
11	<a href="#">그림으로 공부하는 IT 인프라 구조</a>	1	May 31, 2019, 11:22 a.m.
10	<a href="#">룬의 아이들 원터러 7</a>	3	May 31, 2019, 11:21 a.m.

상단 바의 마이페이지로 들어가거나 구매 후 나오는 페이지에서 마이페이지로 들어갈 수 있다. 지금까지 구매한 책의 내역을 확인할 수 있다. 시간 역순으로 정렬된다.

home/views.py

```
def mypage(request):
    cid = request.session.get('cid', False)
    if cid == False :
        return redirect('/login')
    else:
        what =
What.objects.select_related('pid').filter(pid__cid=cid).order_by('-pid__date')
        return render(request, 'home/mypage.html',
        {'login':'logout', 'cid':cid, 'what':what})
```

what은 history의 pid에서 어떤 책을 샀는지를 담은 테이블이다. 현재 세션의 cid와 일치하는 모든 pid에 대해서 책 목록을 가져오고, 시간 역순으로 정렬한다.

home/templates/home/mypage.html

```
{% extends "home/base.html" %}

{% block content %}
    <div>
        <h1>안녕하세요, {{cid}}님!</h1>
        <p> <a href="{% url 'modify' %}"> 회원정보 수정 </a> </p>
        <table border="1">
            <tr> <th>주문 번호</th> <th>책 이름</th> <th>수량</th> <th>주문
날짜</th></tr>
            {% for obj in what %}
                <tr>
                    <td> {{ obj.pid_id }} </td>
                    <td> <a href="{% url 'detail' obj.isbn.category.name
obj.isbn.isbn %}">{{obj.isbn.name}}</a> </td>
                    <td> {{obj.num}} </td>
                    <td> {{obj.pid.date}} </td>
                </tr>
            {% endfor %}
        </table>
    </div>
{% endblock content %}
```

받은 데이터를 테이블에 순서대로 출력한다.

- 책 추천 기능

- (1) 메인 화면에서 유저의 나이/성별에 따라 해당 그룹에서 판매량 제일 높은 책 세 권 추천해준다.

[홈화면](#) [회원가입](#) [logout](#) [마이페이지](#) [도서 구매](#) [북카트](#)

wind님을 위한 책 추천,

wind님 또래의 여자가 가장 많이 선택한 책 세권을 만나보세요.

1. [앵무새 죽이기](#)
2. [호빗](#)
3. [툰의 아이들 윈터러 7](#)

홈 화면에 들어가면 이런 식으로 개인 정보를 활용하여 책을 추천해준다.

[홈화면](#) [회원가입](#) [login](#) [마이페이지](#) [도서 구매](#) [북카트](#)

**우리 책 사이트에서 가장 인기 있는 책 세권을 소개합니다.**

1. [룬의 아이들 윈터러 7](#)
2. [앵무새 죽이기](#)
3. [그림으로 공부하는 IT 인프라 구조](#)

로그인하지 않으면 나이, 성별 상관 없이 가장 인기 있는 책을 추천해준다.

home/views.py

```
def index(request):
    cid = request.session.get('cid', False)
    rec = recommend(cid)
    if cid != False:
        g = Customer.objects.get(cid=cid).gender
        if g == 'F':
            gender = '여자'
        elif g == 'M':
            gender = '남자'
        else:
            gender = '유저'
        return render(request, 'home/index.html', {'login':'logout','cid':cid,
'gender':gender, 'recommend':rec})
    else:
        return render(request, 'home/index.html', {'login':'login','cid':None,
'recommend':rec})
```

recommend 함수에서 실행한 결과를 index.html 에 렌더링 한다.

```
def recommend(cid):
    if cid == False:
        what = What.objects.all()
```

```

        bestList =
what.values('isbn').annotate(total=Count('isbn')).order_by('-total')[:3]
    else:
        customer = Customer.objects.get(cid=cid)
        birth = customer.birthyear
        gender = customer.gender

        if customer.gender == "U":
            what =
What.objects.select_related('pid__cid').filter(pid__cid__birthyear__gte=birth-
5, pid__cid__birthyear__lte=birth+5)
        else :
            what =
What.objects.select_related('pid__cid').filter(pid__cid__gender=gender,
pid__cid__birthyear__gte=birth-5, pid__cid__birthyear__lte=birth+5)
            # what ; 비슷한 사람들이 산 책
            bestList =
what.values('isbn').annotate(total=Count('isbn')).order_by('-total')[:3]
            bookList = []
            for best in bestList:
                bookList.append(Book.objects.get(isbn=best['isbn']))
            return bookList

```

로그인 되어 있지 않을 때는 what에서 같은 isbn끼리 count한 후 판매량 많은 세 권의 책을 리턴한다.

로그인 되어 있을 때는 경우가 두 가지다. Unknown일 때, Female 또는 Male일 때. Unknown일 때는 우선 남자 여자 가리지 않고 내 나이보다 +5 ~ -5살 많은 사람들이 산 책 목록을 뽑는다. Female이나 male일 때는 나랑 같은 성별이면서 내 나이보다 +5 ~ -5살 많은 사람들이 산 책 목록을 뽑는다. 그리고 나서 같은 isbn끼리 count한 후 판매량 많은 세 권의 책을 리턴한다.

home/templates/home/index.html

```

{% extends "home/base.html" %}
{% block content %}
    <div>
        {% if cid is not None %}
            <h2>{{cid}}님을 위한 책 추천,</h2>

```



```
<h2>{{cid}}님 또래의 {{gender}}가 가장 많이 선택한 책 세권을  
만나보세요.</h2>  
{% else %}  
<h2> 우리 책 사이트에서 가장 인기 있는 책 세권을 소개합니다. </h2>  
{% endif %}  
<ol>  
    {% for book in recommend %}  
        <li><a href="{% url 'detail' book.category.name  
book.isbn %}">{{ book.name }}</a> </li>  
    {% endfor %}  
</ol>  
</div>  
{% endblock content %}
```

책 목록을 출력한다.