

MBIT 테스트 페이지 만들기

MBIT

My Best IT personalities

이호준

김혜원

김유진

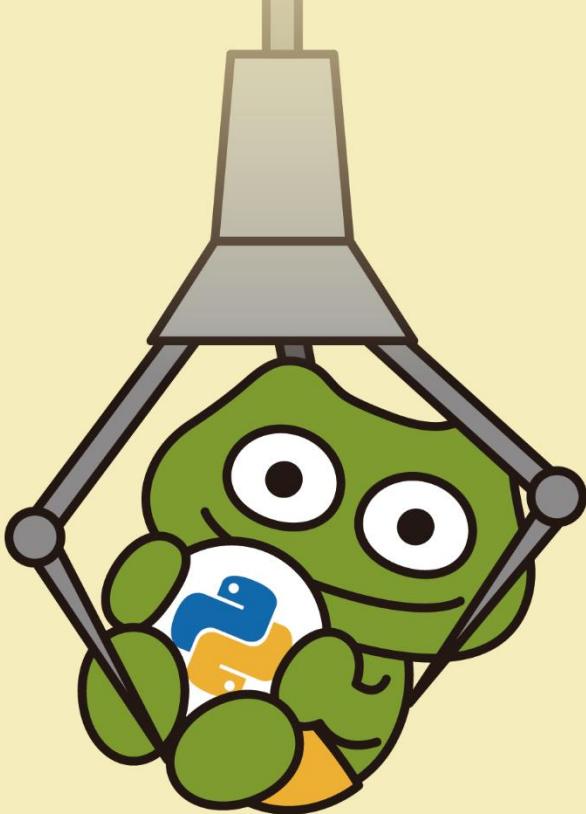
차경림

김 진

현지연

정승한

Chapter_2

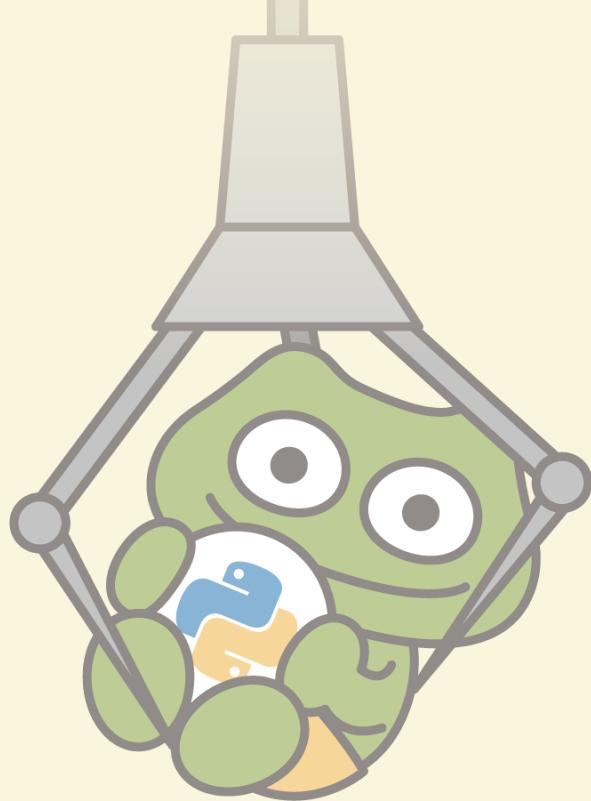


MBIT 테스트 페이지 만들기



My Best IT personalities

Chapter_2





강의 대상과 강의 듣는 방법

강의 대상

이제 막 HTML, CSS, JS, Python 중 1~2개의 언어를 공부하신 분에게 이 강의를 추천합니다. 해당 강의는 서비스를 런칭 해보며 전체적인 프로세스를 익히는 강의입니다.

아주 깊게 하나하나를 짚어드리는 강의는 아니지만, 전체적인 프로세스를 익히기에 충분한 강의라고 생각됩니다.

여러분의 성장에 함께하도록 하겠습니다.

감사합니다.

- 코딩을 전혀 해보신 적이 없으신 분은 아래 강의를 듣고 오시길 권해드립니다.

1. 나의 개발 유형을 테스트해보자! <M.B.I.T> - 넓고 얕게 Computer Science를 전반적으로 훑어보는 강의입니다.

나의 개발 유형을 테스트해보자! - 인프런 | 강의

나의 개발 유형을 테스트해보자! MBTI아닌 MBIT! My Best IT personalities! 개발에 대한 전반적인 내용을 살펴보고 나에게 가장 잘 맞

<https://inf.run/Ug9B>



2. 30분 요약강좌로 HTML, CSS, JS, Python을 전반적으로 훑을 수 있습니다.

[무료] HTML, CSS, JS, Python 30분 요약강좌 - 인프런 | 강의

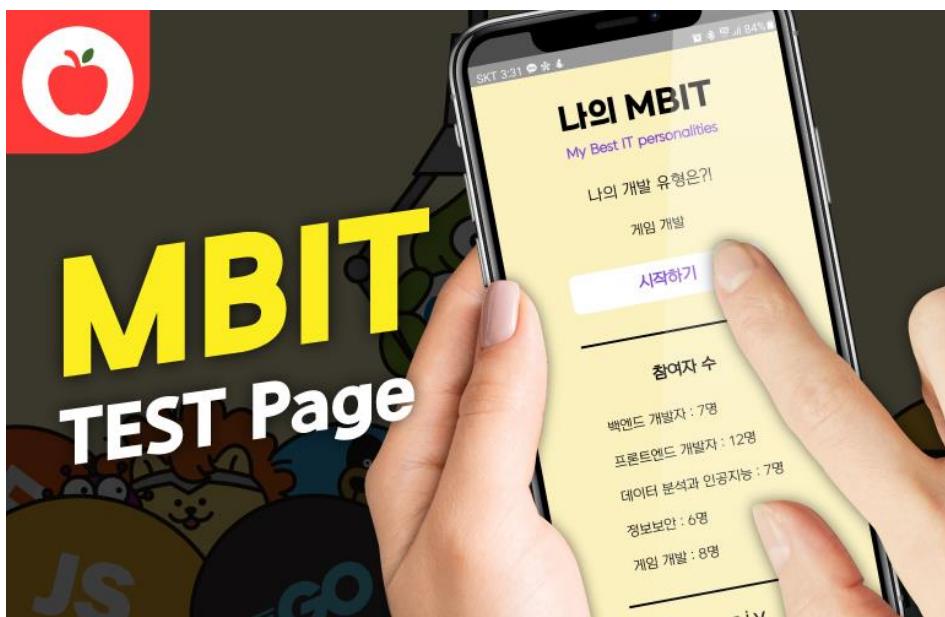
이 강좌는 코딩을 처음하시는 분들에게는 빠르게 훑을 수 있는 강의가 될 것입니다. 이미 코딩을 하시는 분들에게는 복습을 빠르게 할 수 있는

<https://inf.run/Cvkb>





영상 강의 소개



영상 강의는 인프런에서 보실 수 있습니다.

1. 강의는 책에 실린 내용과 좀 더 상세한 설명을 담고 있습니다.
2. 제코베 포트폴리오 템플릿과 해당 책을 인쇄해 볼 수 있는 PDF File을 제공합니다.
3. 함께 보실 수 있는 각종 부록 영상을 포함합니다.



저자 소개

이호준

주식회사 위니브와 바울랩이라는 회사를 이끌고 있어요. 청소년에게는 ICT 관련 진로와 직업을 찾을 수 있도록, 청년에게는 ICT 업을 찾아 주는 일을 하고 있습니다.

김혜원, 김유진, 차경림

주식회사 위니브의 COO, CTO, CDO입니다.

김진, 현지연

주식회사 위니브의 연구원입니다.

정승한

주식회사 위니브의 Back-end 개발자입니다.



강의 자료

강의자료는 아래 링크들을 통해서 확인 및 다운로드 하실 수 있습니다!

노션 페이지 : <https://bit.ly/3cXZJL9>

이미지 다운 링크 : <https://bit.ly/3p4ywIW>



MBIT 테스트

MBIT 테스트는 [👉 https://mbit.weniv.co.kr/](https://mbit.weniv.co.kr/) 을 확인해 주세요!

나의 MBIT

My Best IT personalities

나의 개발 유형은?!

게임 개발

시작하기

참여자 수

백엔드 개발자 : 2명

프론트엔드 개발자 : 3명

데이터 분석과 인공지능 : 1명

정보보안 : 0명

게임 개발 : 1명



Chapter 2. 나의 MBIT 찾기 직접 개발해보기

Chapter 2-1. 프론트엔드

001. 메인페이지 만들기

002. 설문페이지 만들기

003. 결과페이지 만들기

Chapter 2-2. 백엔드

001. 서버 환경 구축(by 구름IDE)

002. 장고 프로젝트 생성하기

003. 모델 작성하기

004. 관리자 페이지

005. 페이지 작성하기

006. 장고 shell과 ORM

007. Database Seeding

008. static 파일

009. 템플릿 적용하기

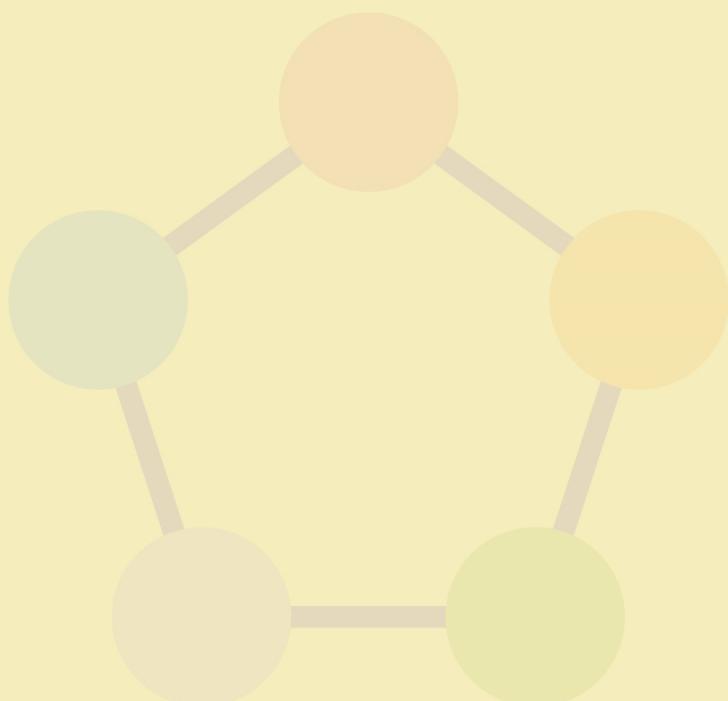
[심화] 배포하기

[심화] 프로젝트 개선

[심화] 커스텀 도메인

Chapter 2-1

프론트엔드 (Front-end)





001. 메인페이지 만들기

1. 화면 레이아웃 구성하기

1.1 우리가 만들 화면

1.2 화면 나누기

2. HTML

2.1 기본 구조

2.2 태그 입력하기

2.3 id, class 입력하기

2.4 콘텐츠 넣기

2.4.1 제목

2.4.2 개발 유형 리스트

2.4.3 버튼

2.4.4 데이터

2.4.5 weniv 로고

2.5 실행화면

3. CSS 적용하기

3.1 CSS 파일 만들기

3.1.1 style.css

3.1.2 reset.css

3.1.3 html 파일에 적용하기

3.2 스타일 적용하기

3.2.1 기본 설정

3.2.2 제목

3.2.3 개발 유형 리스트

3.2.4 버튼

3.2.5 데이터

3.2.6 로고

3.3 실행화면

4. 전체 코드

4.1 index.html

4.2 style.css



1. 화면 레이아웃 구성하기

1.1 우리가 만들 화면



1.2 화면 나누기



1.1에서 확인한 우리가 만들 화면을 위의 이미지와 같이 나눕니다. 콘텐츠들을 넣을 하나의 큰 컨테이너(빨간 상자)와 그 안을 다섯 부분(파란 상자)으로 나누었습니다.

2. HTML

2.1 기본 구조

메인페이지를 만들기 위해 `index.html` 파일을 생성합니다. 우리가 실습할 Visual Studio Code에서 `!` (느낌표)를 입력 후 `Tab` 을 누르면 기본 구조가 자동으로 생성됩니다. (Emmet이 설치되어 있어야 합니다.)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

여기서, `<title></title>` 에 '나의 개발 유형찾기'를 입력하여 제목을 설정합니다.

```
<title>나의 개발 유형찾기</title>
```

2.2 태그 입력하기

1.2에서 나누었던 화면을 태그를 사용하여 실제로 구성해 봅시다. 콘텐츠들을 넣을 `section` 과 콘텐츠들을 감쌀 `div` 하나, 그리고 실제 콘텐츠들을 넣을 `div` 5개로 구성됩니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section>
        <div>
            <div></div>
            <div></div>
            <div></div>
            <div></div>
            <div></div>
        </div>
    </section>
</body>
</html>
```



`<section><div><div>*5` 를 입력 후 `Tab` 을 누르면 자동으로 입력됩니다.

2.3 id, class 입력하기

CSS 적용을 위해 id와 class명을 각 태그에 입력합니다.

```
<section id="main_contents">
    <div class="wrapper">
        <div class="title"></div>
        <div class="intro"></div>
        <div class="buttons"></div>
        <div class="result_data"></div>
        <div class="weniv"></div>
    </div>
</section>
```

2.4 콘텐츠 넣기

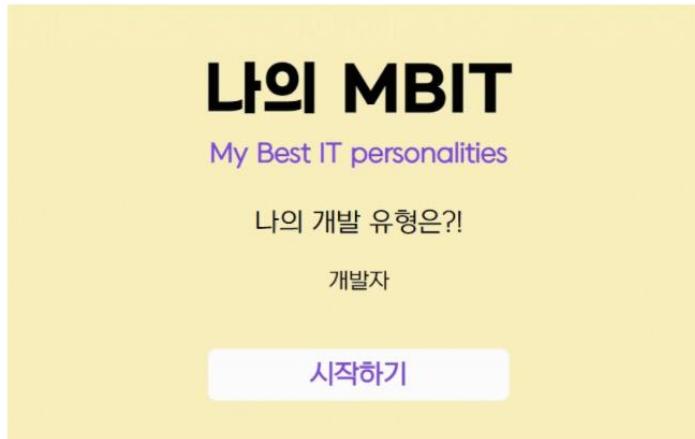
2.4.1 제목

`<div class="title"></div>`에 제목을 넣기 위해서 `h3` 태그 2개를 입력하고 각 태그들의 class 명을 `main_title`과 `sub_title`로 설정하고 콘텐츠를 채워넣어 줍니다.

```
<div class="title">
  <h3 class="main_title">나의 MBIT</h3>
  <h3 class="sub_title">My Best IT personalities</h3>
</div>
```

2.4.2 개발 유형 리스트

`<div class="intro"></div>`에는 animation을 사용하여 개발 유형 리스트를 아래와 같이 넣을 것입니다.



```
<div class="intro">
  <h1>나의 개발 유형은?!</h1>
  <div class="type">
    <ul class="type_list">
      <li>개발자</li>
      <li>데이터 분석과 인공지능</li>
      <li>정보보안</li>
      <li>게임 개발</li>
      <li>개발자</li>
    </ul>
  </div>
</div>
```

"나의 개발 유형은?!"은 `h1` 태그를 사용하고, 개발 유형 리스트는 `ul` 태그를 사용하여 입력합니다. animation 적용은 css 파트에서 진행하도록 하겠습니다.

2.4.3 버튼

이번에는 테스트 시작을 위한 '시작하기' 버튼을 넣어주도록 합시다. `<div class="button"></div>`에 `<button>` 태그를 사용하여 버튼을 생성합니다. 이때 링크 연결을 위해 각 버튼을 `a` 태그로 감싸줍니다.

```
<div class="button">
  <a href="#">
    <button class="start" type="button">시작하기</button>
  </a>
</div>
```

여기서 `a` 태그는 링크를 통해 다른 웹페이지 또는 문서로의 이동을 위해 사용되며, `href` 는 **hypertext reference**의 약자로 연결할 주소를 지정하는 속성입니다. 주소는 #으로 설정하고 이후에 수정해 주도록 합시다.

`button` 태그에서 `class`명은 `start`로, `type` 속성은 `button`으로 설정하였습니다.

2.4.4 데이터

`<div class="result_data"></div>`에는 테스트에 참여한 참여자 수를 넣도록 합시다. '참여자 수'는 `h3` 태그, 각 유형별 참여자 수는 `ul` 태그를 사용하도록 하겠습니다. 각 콘텐츠들은 `div` 태그로 감싸주고 `class`명은 `data_wrap`으로 입력합니다.

```
<div class="result_data">
  <div class="data_wrap">
    <h3>참여자 수</h3>
    <ul>
      <li>백엔드 개발자 : 0명</li>
      <li>프론트엔드 개발자 : 0명</li>
      <li>데이터 분석과 인공지능 : 0명</li>
      <li>정보보안 : 0명</li>
      <li>게임 개발 : 0명</li>
    </ul>
  </div>
</div>
```



실제 데이터(참여자 수)를 넣어보는 실습은 백엔드 파트에서 진행됩니다.

2.4.5 weniv 로고

마지막으로 weniv 로고 이미지를 배치하겠습니다. 이미지 삽입을 위해 `img` 태그를 사용합니다. 이미지를 클릭하면 위니브 홈페이지(<http://www.paullab.co.kr>)로 이동되도록 `a` 태그를 사용합니다.

```
<div class="weniv">
    <a href="http://www.paullab.co.kr">
        
    </a>
</div>
```

이미지는 img 폴더에 있는 `weniv_logo_black.png`를 사용합니다. `src` 속성에 이미지 주소인 `img/weniv_logo_black.png`를 입력합니다.

`alt` 속성은 이미지의 주소가 잘못되었거나 해당 이미지가 존재하지 않을 경우 이미지를 대신하여 나타낼 대체 텍스트를 지정할 수 있는 속성입니다. 여기서는 'weniv'라고 입력하겠습니다.



태그.클래스명 또는 태그#아이디명을 입력 후 Tab 을 누르면 클래스명 또는 아이디명이 입력된 태그가 자동으로 생성됩니다.

2.5 실행화면

나의 MBIT

My Best IT personalities

나의 개발 유형은?!

- 개발자
- 데이터 분석과 인공지능
- 정보보안
- 게임 개발
- 개발자

[시작하기](#)

참여자 수

- 백엔드 개발자 : 0명
- 프론트엔드 개발자 : 0명
- 데이터 분석과 인공지능 : 0명
- 정보보안 : 0명
- 게임 개발 : 0명

weniv

3. CSS 적용하기

3.1 CSS 파일 만들기

3.1.1 style.css

이제부터 메인페이지를 꾸며주기 위한 css 파일을 만들어 봅시다. css 폴더를 만든 후 `style.css` 파일을 만들어 주세요.

메인페이지에서의 모든 스타일 설정은 `style.css`에서 진행됩니다.

3.1.2 reset.css

css 폴더 내에 `reset.css` 파일을 생성한 후 아래 링크에서 코드를 복사하여 붙여넣기합니다.

meyerweb.com

The goal of a reset stylesheet is to reduce browser inconsistencies in things like default line heights, margins and font sizes of headings, and so on. The general

 <https://meyerweb.com/eric/tools/css/reset/>



행 높이, 여백 및 글꼴 크기 등과 같은 기본 스타일 설정은 브라우저마다 모두 다릅니다. 이에 따른 문제를 개선하고 기본 설정을 모두 통일시켜주기 위해 `reset.css`를 통해 스타일을 초기화합니다.

`reset.css`는 스타일 초기화에만 사용되며 추가 작업이 이루어지지 않습니다.

3.1.3 html 파일에 적용하기

앞서 생성한 두 개의 css 파일을 `index.html`에 적용합시다. `<head></head>` 안에 아래 코드를 넣어줍니다.

```
<link rel="stylesheet" type="text/css" href="css/reset.css">
<link rel="stylesheet" type="text/css" href="css/style.css">
```

3.2 스타일 적용하기

3.2.1 기본 설정

기본적으로 설정할 것은 글꼴, 배경, 그리고 기본 배치입니다.

우선 `@font-face` 를 사용하여 폰트를 적용하여 봅시다. 우리가 사용할 폰트는 'Gmarket Sans B'와 '넥슨 Lv.1 고딕 Regular' 이렇게 두가지입니다.

아래와 같이 css 코드 제일 상단에 입력합니다.

```
@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
    format("woff");
    font-weight: normal;
    font-style: normal;
}
```



폰트는 상업용 무료 한글 폰트 사이트인 '[눈누](https://noonnu.cc)'(<https://noonnu.cc>)에서 가져왔으며, 다양한 폰트를 확인할 수 있습니다.

이제 `body` 에 기본 폰트, 글자 색상, 그리고 배경색을 지정해 봅시다. 앞서 설명했듯이 우리가 사용할 폰트는 두가지이지만, 우선 '넥슨 Lv.1 고딕 Regular'를 기본 폰트로 설정하겠습니다.

```
body {
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}
```

모든 콘텐츠를 감싼 `<section id="main_contents"></seciton>` 를 전체 화면으로 설정해 주기 위해 `width` 와 `height` 를 모두 100%로 지정합니다.

```
#main_contents{  
    width: 100%;  
    height: 100%;  
}
```

하지만 아래 사진과 같이 `#main_contents` 는 전체 화면이 아닌 콘텐츠만큼의 영역만 차지합니다.



이러한 문제를 해결하기 위해서는 `html` 과 `body` 각각의 width와 height에도 100%씩 설정해 주어야 합니다. 아래와 같이 입력하면 `#main_contents` 의 영역이 전체 화면만큼 차지하는 것을 확인할 수 있습니다.

```
html {
    width: 100%;
    height: 100%;
}

body {
    width: 100%;
    height: 100%;
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}
```



개발자 도구(F12 또는 Ctrl+Shift+i)를 사용하면 태그 및 콘텐츠의 영역과 스타일을 쉽게 확인 할 수 있습니다.

1.1의 우리가 만들 화면은 중앙 정렬이 되어 있고 화면의 상단에는 여백이 있습니다.

여기서는 **플렉스 박스(flex box)**를 사용하여 레이아웃을 설정합니다. 해당 요소를 블록 타입으로 정의하기 위해 `display` 속성을 flex로 지정하고, 플렉스 박스의 속성 중 수평 방향 정렬을 설정할 수 있는 `justify-content` 속성을 center로 설정하여 콘텐츠들을 중앙으로 배치합니다.

상단의 여백은 `margin-top` 을 100px 만큼 설정하여 만들어 줍니다.

```
#main_contents{
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100%;
    margin-top: 100px;
}
```

콘텐츠를 다시 한 번 감싸주는 `div` 인 `<div class="wrapper"></div>` 의 width, height, 그리고 배치도 설정해 줍시다.

width와 height, 그리고 display는 `#main_contents` 와 동일하게 설정합니다. 하지만 `.wrapper` 에서는 플렉스 박스의 속성 중 하나인 `flex-direction` 속성을 `column`으로 설정하여 요소들의 배치 방향을 수직으로 변경시킵니다.

또한, `max-width` 를 사용하여 500px 만큼의 최대 너비를 설정합니다. max-width는 내부 콘텐츠들의 너비가 최대 너비보다 커지는 것을 방지하기 위하여 사용합니다.

```
#main_contents .wrapper {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: 100%;
    max-width: 500px;
}
```

3.2.2 제목

이제 제목의 스타일을 설정합시다. `text-align` 을 center로 설정하여 중앙 정렬을 하고, `padding-bottom` 을 사용하여 하단에 50px만큼 여백을 줍니다.

```
.title {
    text-align: center;
    padding-bottom: 50px;
}
```

메인 제목과 서브 제목은 서로 글자 크기와 색상 등의 스타일이 다르기 때문에 각각의 class명을 사용하여 설정합니다.

`.main_title` 의 폰트는 'Gmarket Sans B'로 지정하고 `font-size` 는 50px로 설정합니다. 또한 하단의 `.sub_title` 과의 간격을 주기 위해 `padding-bottom` 을 사용하여 15px만큼 여백을 설정합니다.

```
.main_title {
    font-family: "GmarketSansBold";
    font-size: 50px;
    padding-bottom: 15px;
}
```

.sub_title 에서는 글자 크기와 굵기, 그리고 색상을 설정합니다.

```
.sub_title {
    font-size: 20px;
    font-weight: bold;
    color: #7F47DD;
}
```

3.2.3 개발 유형 리스트

지금부터는 CSS 애니메이션을 사용해서 개발 유형들이 하나씩 돌아가면서 나타나도록 하겠습니다.

우선 애니메이션 적용에 앞서 .intro 에 스타일을 적용하여 요소들을 배치하고 아래 버튼과 간격을 주도록 하겠습니다.

text-align 속성을 center로 설정하여 중앙 정렬시키고, padding-bottom 을 사용하여 40px만큼 여백을 줍니다.

```
.intro {
    text-align: center;
    padding-bottom: 40px;
}
```

"나의 개발 유형은?!"에 스타일을 설정해 주도록 합시다. .intro 의 자손인 h1 을 선택하기 위해

.intro h1 선택자를 사용하여 스타일을 적용합니다.

```
.intro h1 {
    font-size: 30px;
    padding-bottom: 15px;
}
```

개발 유형 리스트의 font-size 를 25px로 설정합니다. .type 의 line-height 와 height 를 동일한 값으로 설정합니다.

```
.type {
    font-size: 25px;
    line-height: 3em;
    height: 3em;
}
```

실행화면을 보면 아래와 같이 리스트가 넘친 것을 확인할 수 있습니다. 이를 해결하기 위해 `overflow: hidden` 을 추가하여 넘친 콘텐츠를 잘라내어 보이지 않게 합니다.



```
.type {
    font-size: 25px;
    line-height: 3em;
    height: 3em;
    overflow: hidden;
}
```

이제 애니메이션을 적용하기 위해 `animation` 속성과 애니메이션 중간 상태를 기술하기 위한 `@keyframes` 규칙을 사용하도록 하겠습니다.

우리는 아래와 같이 5개의 중간 상태를 기술하여 5개의 `li` 요소의 위치를 이동시키며 하나씩 나타나게 할 것입니다. `transform` 속성의 `translateY()`를 사용하여 Y축(수직 방향)으로 이동 효과를 부여합니다.

```
@keyframes rotate {
    0% {
        transform: translateY(0);
    }
    25% {
        transform: translateY(-20%);
    }
    50% {
        transform: translateY(-40%);
    }
    75% {
        transform: translateY(-60%);
    }
    100% {
        transform: translateY(-80%);
    }
}
```

키프레임을 통해 애니메이션의 중간 상태를 기술하였으면 이제 적용하도록 합시다. 스타일은 `.type_list`에 적용합니다.

`animation` 속성에 중간 상태를 기술한 `animation-name` 과 한 사이클을 완료하는 데 걸리는 시간을 지정하는 `animation-duration`, 그리고 애니메이션이 무한히 반복되도록 `animation-iteration-count`를 설정합니다. 이때, 따로 설정하지 않은 속성들은 기본값으로 설정됩니다.

```
.type_list {
    animation: rotate 7s infinite;
}
```

위의 코드처럼 `animation` 속성에 한 번에 입력할 수도 있으나, 아래와 같이 풀어서 입력할 수도 있습니다.

```
animation-name: rotate;
animation-duration: 7s;
animation-iteration-count: infinite;
```

3.2.4 버튼

이제 버튼을 꾸며줄 차례입니다. 우선 앞에서 해왔던 것과 동일한 방법으로 배치부터 해보도록 합시다. `text-align` 을 center로 설정하여 중앙 정렬하고, `padding-bottom` 으로 안쪽 아래 50px만큼의 여백을 줍니다.

```
.button {
    text-align: center;
    padding-bottom: 50px;
}
```

버튼의 `width` 와 `height` 를 각각 300px, 50px로 하여 크기를 변경합니다. 또한, `padding` 을 5px로 설정하여 버튼 안쪽 여백을 상, 하, 좌, 우, 모두 균일하게 5px씩 줍니다.

`border-style` 을 none으로 지정하여 버튼의 테두리를 없애고, 버튼의 각 모서리를 부드럽게 만들어 주기 위해 `border-radius` 를 10px로 설정합니다.

```
.button button{
    width: 300px;
    height: 50px;
    padding: 5px;
    border-style: none;
    border-radius: 10px;
}
```

이어서 버튼의 폰트, 크기, 굵기 등의 스타일을 설정합니다.

```
.button button{
    ...
    font-family: "NEXON Lv1 Gothic OTF";
    font-size: 20px;
    font-weight: bold;
}
```

버튼의 배경색과 글자색도 함께 설정합니다. 요소 위에 마우스 커서가 올라갔을 때 모양을 지정하는 `cursor` 속성을 `pointer`로 지정합니다. 또한 `margin-bottom` 을 20px로 설정하여 아래 요소와의 여백을 줍니다.

```
.button button{
    ...
    background-color: #fff;
    color: #7F47DD;
    cursor: pointer;
    margin-bottom: 20px;
}
```



`padding` 은 요소의 안쪽 여백을 설정하고, `margin` 은 요소의 바깥 여백을 설정합니다.

버튼에 마우스 커서가 올라갔을 때 효과를 주고 싶습니다. 이때 사용하는 것이 가상 클래스 `:hover`입니다

`.button button:hover` 는 `.button` 의 자손인 `button` 에 마우스 오버했을 때 `button` 이 선택되며, 이때 선택된 요소(button)에 스타일을 설정할 수 있습니다.

여기서는 `text-decoration` 속성을 `underline`으로 설정하여 마우스 오버되면 밑줄이 생기도록 합니다.

```
.button button:hover{
    text-decoration: underline;
}
```

3.2.5 데이터

'참가자 수'를 표시할 `.result_data` 를 중앙 정렬시키기 위해 `display` 를 flex로 설정하고, 수평 방향으로 정렬하는 `justify-content` 를 center로 설정합니다.

```
.result_data {
    display: flex;
    justify-content: center;
}
```

데이터 영역은 버튼과 같은 `width` 를 설정할 것입니다. 하지만 버튼은 300px로 설정해 주었지만 `.data_wrap` 에서 `width` 는 270px로 설정합니다. 그 이유는 `padding` 값을 15px씩 주었으므로 300px로 설정하면 실제 차지하는 width는 330px이 되어버리기 때문입니다. 따라서 270px로 설정해야 실제 차지하게 되는 width는 300px이 됩니다.

그 다음은 위, 아래에 각각 수평선을 그리기 위해 요소의 테두리를 설정하는 `border` 를 사용합니다. `border-top` 과 `border-bottom` 에는 각각 "3px solid #000"의 동일한 값을 입력합니다. 이는 하나의 직선(solid)으로 두께가 3px인 검은색 테두리를 그린다는 뜻입니다.

```
.result_data .data_wrap {
    width: 270px;
    padding: 15px;
    border-top: 3px solid #000;
    border-bottom: 3px solid #000;
}
```

이어서 `line-height` 을 1.5로 설정하여 행간을 띄우고, 아래 로고와의 여백을 50px만큼 줍니다.

```
.result_data .data_wrap {
    ...
    line-height: 1.5;
    margin-bottom: 50px;
}
```

'참여자 수'를 중앙 정렬하고 폰트 사이즈를 20px로 설정합니다. 그리고 글자 굵기는 bold로 설정합니다.

안쪽 여백을 상, 하에만 각각 20px과 30px로 설정하기 위해 `padding: 20px 0 30px` 로 입력합니다.

```
.result_data h3 {
    text-align: center;
    font-size: 20px;
    font-weight: bold;
    padding: 20px 0 30px;
}
```

유형별 참여자 수는 '참여자 수'보다 좀 작게 표현하기 위해 폰트 사이즈를 15px로 합니다. 또한 안쪽 아래 여백을 위의 여백과 동일하게 20px로 설정합니다.

```
.result_data li {
    font-size: 15px;
    padding-bottom: 20px;
}
```

3.2.6 로고

이제 메인페이지의 마지막 작업인 로고입니다. 이전과 마찬가지로 우선 배치부터 해보도록 합시다.

`justify-content` 를 center로 설정하여 중앙 정렬을 시켜주고 아래 여백은 50px만큼 줍니다.

```
.weniv {
    display: flex;
    justify-content: center;
    padding-bottom: 50px;
}
```

로고의 크기를 조절하기 위해 `img` 태그를 감싸고 있는 `a` 태그의 `width` 를 100px로 설정하고, `img` 의 `width` 와 `height` 는 각각 100%, auto로 설정합니다.

```
.weniv a {  
    width: 100px;  
}  
  
.weniv img{  
    width: 100%;  
    height: auto;  
}
```

3.3 실행화면

나의 MBIT

My Best IT personalities

나의 개발 유형은?!

개발자

시작하기

참여자 수

백엔드 개발자 : 0명
프론트엔드 개발자 : 0명
데이터 분석과 인공지능 : 0명
정보보안 : 0명
게임 개발 : 0명

weniv

4. 전체 코드

4.1 index.html

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="css/reset.css">
    <link rel="stylesheet" type="text/css" href="css/style.css">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section id="main_contents">
        <div class="wrapper">
            <div class="title">
                <h3 class="main_title">나의 MBIT</h3>
                <h3 class="sub_title">My Best IT personalities</h3>
            </div>
            <div class="intro">
                <h1>나의 개발 유형은?!</h1>
                <div class="type">
                    <ul class="type_list">
                        <li>개발자</li>
                        <li>데이터 분석과 인공지능</li>
                        <li>정보보안</li>
                        <li>게임 개발</li>
                        <li>개발자</li>
                    </ul>
                </div>
            </div>
            <div class="button">
                <a href="#">
                    <button class="start" type="button">시작하기</button>
                </a>
            </div>
            <div class="result_data">
                <div class="data_wrap">
                    <h3>참여자 수</h3>
                    <ul>
                        <li>백엔드 개발자 : 0명</li>
                        <li>프론트엔드 개발자 : 0명</li>
                        <li>데이터 분석과 인공지능 : 0명</li>
                        <li>정보보안 : 0명</li>
                        <li>게임 개발 : 0명</li>
                    </ul>
                </div>
            </div>
        </div>
    </section>
</body>

```

```

<div class="weniv">
    <a href="http://www.paullab.co.kr">
        
    </a>
</div>
</div>
</section>
</body>
</html>

```

4.2 style.css

```

@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
        format("woff");
    font-weight: normal;
    font-style: normal;
}

@keyframes rotate {
    0% {
        transform: translateY(0);
    }
    25% {
        transform: translateY(-20%);
    }
    50% {
        transform: translateY(-40%);
    }
    75% {
        transform: translateY(-60%);
    }
    100% {
        transform: translateY(-80%);
    }
}

```

```
html {
    width: 100%;
    height: 100%;
}

body {
    width: 100%;
    height: 100%;
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}

#main_contents {
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100%;
    margin-top: 100px;
}

#main_contents .wrapper {
    display: flex;
    flex-direction: column;
    width: 100%;
    height: 100%;
    max-width: 500px;
}

.title {
    text-align: center;
    padding-bottom: 50px;
}

.main_title {
    font-family: "GmarketSansBold";
    font-size: 50px;
    padding-bottom: 15px;
}

.sub_title {
    font-size: 20px;
    font-weight: bold;
    color: #7F47DD;
}
```

```
.intro {  
    text-align: center;  
    padding-bottom: 40px;  
}  
  
.intro h1 {  
    font-size: 30px;  
    padding-bottom: 15px;  
}  
  
.type {  
    font-size: 25px;  
    line-height: 3em;  
    height: 3em;  
    overflow: hidden;  
}  
  
.type_list {  
    animation: rotate 7s infinite;  
}  
  
.button {  
    text-align: center;  
    padding-bottom: 50px;  
}  
  
.button button {  
    width: 300px;  
    height: 50px;  
    padding: 5px;  
    border-style: none;  
    border-radius: 10px;  
    font-family: "NEXON Lv1 Gothic OTF";  
    font-size: 20px;  
    font-weight: bold;  
    cursor: pointer;  
    background-color: #fff;  
    color: #7F47DD;  
    margin-bottom: 20px;  
}  
  
.button button:hover {  
    text-decoration: underline;  
}  
  
.result_data {  
    display: flex;  
    justify-content: center;  
}
```

```
.result_data .data_wrap {  
    width: 270px;  
    padding: 15px;  
    border-top: 3px solid #000;  
    border-bottom: 3px solid #000;  
    line-height: 1.5;  
    margin-bottom: 50px;  
}  
  
.result_data h3 {  
    text-align: center;  
    font-size: 20px;  
    font-weight: bold;  
    padding: 20px 0 30px;  
}  
  
.result_data li {  
    font-size: 15px;  
    padding-bottom: 20px;  
}  
  
.weniv {  
    display: flex;  
    justify-content: center;  
    padding-bottom: 50px;  
}  
  
.weniv a {  
    width: 100px;  
}  
  
.weniv img {  
    width: 100%;  
    height: auto;  
}
```



002. 설문페이지 만들기

1. 화면 레이아웃 구성하기

1.1 우리가 만들 화면

1.2 화면 나누기

2. HTML

2.1 기본 구조

2.2 태그 입력하기

2.3 id, class 입력하기

2.4 콘텐츠 넣기

2.4.1 문제

2.4.2 설문 항목

2.4.3 그 외 문항

2.5 실행화면

3. CSS 적용하기

3.1 CSS 파일 만들기

3.2 스타일 적용하기

3.2.1 기본 설정

3.2.2 문제

3.2.3 설문 항목

3.2.4 버튼

3.2.5 추가 작업

3.3 실행화면

4. 기능 추가하기

4.1 js 파일 만들기

4.2 슬라이드 효과

4.2.1 scrollDown()

4.2.2 scrollUp()

4.3 유효성 검사

4.3.1 다음 문항으로 넘어가기

4.3.2 제출하기

4.4 새로고침

5. 전체 코드

5.1 form.html

5.2 form.css

5.3 form.js

1. 화면 레이아웃 구성하기

1.1 우리가 만들 화면

 실제 테스트는 10개의 문항으로 설문지가 구성되어 있지만, 프론트엔드 강의에서는 3개의 문항으로 실습을 진행합니다. 이후에 진행될 백엔드 파트에서는 모든 문항을 불러오는 실습이 진행됩니다.

1/3

당신이 가장 재밌었던 수업은?

- 게임 개발(유니티, 언리얼, Pygame)
- 백엔드 또는 인프라(Cassandra, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)
- 정보보안(해킹과 방어)
- 프론트엔드(HTML, CSS, Javascript, etc)
- 데이터 분석과 인공지능(천자리, 분석, 시각화, 머신러닝, 딥러닝)
- 이 같은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.

[다음](#)

2/3

여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?

- 데이터 분석을 통한 효과적인 개인 맞춤형 서비스
- 회의하거나 품격있는 디자인, 흥미로운 콘텐츠 제작
- 게임적인(Gamification, 게이미피케이션) 요소 도입
- 안정적인 서비스 운영
- 믿을 수 있는 경보 관리

[이전](#)[다음](#)

3/3

당신은 친구의 집들이에 가려고 합니다. 당신이 선택한 선물은?

- 집 꾸미기에 도움이 되는 예쁜 인테리어 소품을 골라볼까?
- 실용성이 강이지! 화장지랑 수건 세트랑...
- 음... 우선 센스있는 집들이 선물 먼저 검색해봐야지!
- 집들이 가서 뭐하겠어! 보드 게임이나 사 기자!
- 역시 집은 안전이 최고!! 험관문 보안감지 사줘야겠다.

[이전](#)[제출](#)

1.2 화면 나누기

화면은 아래와 같이 나누어집니다. 모든 문항들을 설문지페이지에 뿌려주고 실제 화면에는 한 문항만 보여주기 위해 각 문항들은 전체화면으로 설정해 주어야 합니다.

1/3

당신이 가장 재밌었던 수업은?

- 게임 개발(유니티, 언리얼, Pygame)
- 백엔드 또는 인프라(C계열-도전, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 무한문자, etc)
- 정보보안(해킹과 방어)
- 프론트엔드(HTML, CSS, Javascript, etc)
- 데이터 분석과 인공지능(천자리, 분석, 시각화, 마신러닝, 딥러닝)
- 이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.

[다음](#)

2/3

여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?

- 데이터 분석을 통한 효과적인 개인 맞춤형 서비스
- 화려하거나 품격있는 디자인, 음미로운 콘텐츠 제작
- 게임적인(Gamification, 게이미피케이션) 요소 도입
- 안정적인 서비스 운영
- 믿을 수 있는 정보 관리

[이전](#)[다음](#)

3/3

당신은 친구의 집들이에 가려고 합니다. 당신이 선택한 선물은?

- 집 꾸미기에 도움이 되는 예쁜 인테리어 소품을 골라볼까?
- 실용성이 같아지! 화장지랑 수건 세트랑...
- 음... 우선 센스 있는 집들이 선물 먼저 검색해봐야지!
- 집들이 가서 뭐하겠어! 보드 게임이나 사기자!
- 역시 집은 안전이 최고!! 현관문 보안장치 사줘야겠다.

[이전](#)[제출](#)

2. HTML

2.1 기본 구조

설문지페이지를 만들기 위해 `form.html` 파일을 생성합니다. 기본 구조는 메인페이지의 `index.html`과 동일합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>나의 개발 유형찾기</title>
</head>
<body>

</body>
</html>
```

2.2 태그 입력하기

`section` 과 콘텐츠들을 감쌀 `div` 하나, 사용자가 설문 데이터를 전송하기 위한 `form`, 그리고 그 안에 문항들을 넣을 `div` 3개로 구성됩니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section>
        <div>
            <form>
                <div></div>
                <div></div>
                <div></div>
            </form>
        </div>
    </section>
</body>
</html>
```

2.3 id, class 입력하기

CSS 적용을 위해 id와 class명을 각 태그에 입력합니다.

```
<section id="survey">
  <div class="wrapper">
    <form id="form">
      <div class="test"></div>
      <div class="test"></div>
      <div class="test"></div>
    </form>
  </div>
</section>
```

2.4 콘텐츠 넣기

2.4.1 문제

문항 번호와 문제는 `h3` 태그를 사용하여 `<div class="question_container"></div>` 안에 넣습니다. class명은 각각 `number`와 `question`으로 입력합니다.

```
<div class="question_container">
  <h3 class="number">1/3</h3>
  <h3 class="question">당신이 가장 재밌었던 수업은?</h3>
</div>
```

2.4.2 설문 항목

설문 항목은 `<div class="answer"></div>`에 들어갑니다. 설문지에서 사용자는 한 개의 답안을 선택해야 합니다. 그렇게 때문에 우리는 `input` 태그의 타입 속성인 **라디오 버튼**(radio button)을 사용할 것입니다. 또한 항목 설명을 위해 `label` 태그를 사용하며, `input` 태그와 `label` 태그는 `div` 태그로 감싸줍니다.

```
<div class="answer">
  <div>
    <input type="radio" name="answer_1" />
    <label>개임 개발(유니티, 언리얼, Pygame)</label>
  </div>
  <div>
    <input type="radio" name="answer_1" />
    <label>백엔드 또는 인프라(C계열-닷넷, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)</label>
  </div>
  <div>
    <input type="radio" name="answer_1" />
    <label>정보보안(해킹과 방어)</label>
  </div>
  <div>
    <input type="radio" name="answer_1" />
    <label>프론트엔드(HTML, CSS, Javascript, etc)</label>
  </div>
  <div>
    <input type="radio" name="answer_1" />
    <label>데이터 분석과 인공지능(전처리, 분석, 시각화, 머신러닝, 딥러닝)</label>
  </div>
  <div>
    <input type="radio" name="answer_1" />
    <label>이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.</label>
  </div>
</div>
```

`input` 태그의 `name` 속성은 "제출" 버튼을 클릭했을 때 서버로 전달되는 이름을 지정할 수 있는 속성입니다. 여기서 문항(문제)별 `name`을 동일하게 지정해 주어야 같은 그룹으로 설정할 수 있습니다.



`input` 태그의 `id` 속성과 `label` 태그의 `for` 속성을 동일한 값으로 입력하면 텍스트(항목)를 클릭하여도 라디오 버튼을 선택할 수 있습니다.

`input` 태그의 `id`, `value` 등의 값과 `label` 태그의 `for` 속성값을 설정하는 작업은 백엔드 파트에서 진행됩니다.

2.4.3 버튼

다음 문항으로 넘어가기 위한 "다음" 버튼을 만들어 줍니다. `<div class="btn_wrap"></div>` 안에 `button` 태그를 만들어 주고 class명은 `next_btn`으로 입력합니다.

```
<div class="btn_wrap">
    <button class="next_btn">다 음</button>
</div>
```

2.4.3 그 외 문항

아래는 2번 문항입니다. 기본 구조는 앞서 설명한 구조와 같기 때문에 그대로 복사하여 붙여넣기하겠습니다. 안에 들어갈 문제와 설문 항목 등의 내용은 아래와 같이 변경합니다.

1번 문항과 2번 문항은 별도의 그룹이기 때문에 `input` 태그의 name 속성은 `answer_2`로 입력합니다. 또한 2번 문항부터는 이전으로 돌아갈 수 있도록 "이전" 버튼을 추가합니다.

```
<div class="question_container">
    <div class="number">2/3</div>
    <div class="question">여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?</div>
</div>
<div class="answer">
    <div>
        <input type="radio" name="answer_2" />
        <label>데이터 분석을 통한 효과적인 개인 맞춤형 서비스</label>
    </div>
    <div>
        <input type="radio" name="answer_2" />
        <label>화려하거나 품격있는 디자인, 흥미로운 콘텐츠 제작</label>
    </div>
    <div>
        <input type="radio" name="answer_2" />
        <label>게임적인(Gamification, 게이미피케이션) 요소 도입</label>
    </div>
    <div>
        <input type="radio" name="answer_2" />
        <label>안정적인 서비스 운영</label>
    </div>
    <div>
        <input type="radio" name="answer_2" />
        <label>믿을 수 있는 정보 관리</label>
    </div>
<div class="btn_wrap">
    <button class="prev_btn">이 전</button>
    <button class="next_btn">다 음</button>
</div>
```

아래는 3번 문항입니다. 앞에서 해본 것과 마찬가지로 `input` 태그의 name 속성과 문제 및 설문 항목 등의 내용 외에는 구조가 동일합니다.

하지만 3번 문항은 마지막 문항이기 때문에 "다음" 버튼이 아닌 "제출" 버튼이 필요합니다. 여기서 제출 버튼은 `button` 태그가 아닌 `input` 태그를 사용합니다. type 속성을 `submit`으로 입력하고 value 속성에는 "제출"이라고 입력하여 제출 버튼을 만들어 줍니다. class 명은 `submit_btn`이라고 하겠습니다.

```
<div class="question_container">
    <div class="number">3/3</div>
    <div class="question">당신은 친구의 집들이에 가려고 합니다. 당신이 선택한 선물은?</div>
</div>
<div class="answer">
    <div>
        <input type="radio" name="answer_3" />
        <label>집 꾸미기에 도움이 되는 예쁜 인테리어 소품을 골라볼까?</label>
    </div>
    <div>
        <input type="radio" name="answer_3" />
        <label>실용성이 같이지! 화장지랑 수건 세트랑...</label>
    </div>
    <div>
        <input type="radio" name="answer_3" />
        <label>음... 우선 센스있는 집들이 선물 먼저 검색해봐야지!</label>
    </div>
    <div>
        <input type="radio" name="answer_3" />
        <label>집들이 가서 뭐하겠어! 보드 게임이나 사 가자!</label>
    </div>
    <div>
        <input type="radio" name="answer_3" />
        <label>역시 집은 안전이 최고!! 현관문 보안장치 사줘야겠다.</label>
    </div>
</div>
<div class="btn_wrap">
    <button class="prev_btn">이전</button>
    <input type="submit" value="제출" class="submit_btn"/>
</div>
```

2.5 실행화면

1/3

당신이 가장 재밌었던 수업은?

- 게임 개발(유니티, 언리얼, Pygame)
- 백엔드 또는 인프라(C계열-닷넷, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)
- 정보보안(해킹과 방어)
- 프론트엔드(HTML, CSS, Javascript, etc)
- 데이터 분석과 인공지능(머신러닝, 딥러닝)
- 이 같은 내 길이 아닌 것 같다.. 고로 재미있는지 모르겠다.

[다음]

2/3

여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?

- 데이터 분석을 통한 효과적인 개인 맞춤형 서비스
- 화려하거나 즐거운 디자인, 큐비로운 클렌즈 제작
- 게임적인(Gamification, 게이미피케이션) 요소 도입
- 안정적인 서비스 운영
- 넓을 수 있는 정보 관리

[이전] **[다음]**

3/3

당신은 친구의 집들이에 가려고 합니다. 당신이 선택한 선물은?

- 집 구미기에 도움이 되는 예쁜 인테리어 소품을 골라볼까?
- 실용성이 입이지! 화장지랑 수건 세트랑...
- 음.. 우선 샌스있는 집들이 선물 먼저 검색해봐야지!
- 집들이 가서 뭐하겠어! 보드 게임이나 사 가치!
- 역시 집은 안전이 최고!! 헌관문 보안장지 사줘야겠다.

[이전] **[제출]**

3. CSS 적용하기

3.1 CSS 파일 만들기

설문지페이지를 꾸며주기 위한 `form.css` 파일을 css 폴더 내에 만듭니다. 앞서 만들었던 `reset.css` 와 `form.css` 파일을 `form.html`에 아래와 같이 적용시킵니다.

```
<link rel="stylesheet" type="text/css" href="css/reset.css">
<link rel="stylesheet" type="text/css" href="css/form.css">
```

3.2 스타일 적용하기

3.2.1 기본 설정

메인페이지에서 기본 스타일 설정을 했던 것과 같이 `@font-face` 를 통해 폰트를 적용합니다.

```
@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
    format("woff");
    font-weight: normal;
    font-style: normal;
}
```

설문페이지의 배경색, 글자색, 그리고 기본 폰트 설정을 아래와 같이 합니다.

```
body {
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}
```

모든 콘텐츠들을 담고 있는 `#survey` 의 스타일을 설정합니다. 우선 `display` 를 flex로 설정하고, `justify-content` 를 center로 설정하여 콘텐츠들을 중앙으로 배치합니다.

`width` 와 `height` 는 각각 100vw, 100vh로 뷰포트의 너비값, 높이값을 갖도록 합니다. 그리고 `margin-top` 을 통해 상단의 50px만큼의 여백을 만들어 줍니다.

```
#survey{
    display: flex;
    justify-content: center;
    width: 100vw;
    height: 100vh;
    margin-top: 50px;
}
```

문제와 각 답안들이 들어가는 `.test` 는 `max-width` 를 500px로 설정하여 내부 콘텐츠들의 너비가 600px를 넘지 못하도록 합니다. 또한 화면에는 한 개의 문제만 보여져야 하므로 `height` 는 뷰포트의 높이값을 갖도록 100vh로 설정합니다.

```
.test {
    max-width: 600px;
    height: 100vh;
}
```

3.2.2 문제

`display` 를 flex로 설정하고 `flex-direction` 을 column으로 설정하여 콘텐츠를 수직으로 배치합니다. 그리고 `align-items` 를 center로 입력하여 문제를 중앙으로 배치합니다.

`font-size` 는 30px로 설정하고 행간을 주기 위해 `line-height` 를 1.3으로 입력합니다. 또한 하단의 설문 항목과의 간격을 주기 위해 `padding-bottom` 을 30px로 설정합니다.

```
.question_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    font-size: 30px;
    line-height: 1.3;
    padding-bottom: 30px;
}
```

문제 번호와 문제의 간격을 주기위해 `padding-bottom` 을 30px로 설정하여 `.number` 의 안쪽 아래에 여백을 줍니다.

```
.question_container .number {
    padding-bottom: 30px;
}
```

문제의 폰트는 'GmarketSansBold' 로 지정하여 강조합니다.

```
.question_container .question {
    font-family: "GmarketSansBold";
}
```

3.2.3 설문 항목

설문 항목의 `font-size` 는 17px로 합니다. 또한 `line-height` 는 1.3으로, `padding-bottom` 은 30px로 동일하게 설정합니다.

```
.answer {
    font-size: 17px;
    line-height: 1.3;
    padding-bottom: 30px;
}
```

각 항목들이 서로 간격을 갖도록 `padding-bottom` 을 15px로 설정합니다.

```
.answer div {
    padding-bottom: 15px;
}
```

`input` 태그의 마우스 커서를 가져가면 커서의 모양이 손가락 모양이 되도록 `cursor` 를 `pointer`로 지정합니다.

```
.answer input {
    cursor: pointer;
}
```

3.2.4 버튼

1번 문항의 버튼의 경우는 중앙으로 배치하고, 그 외 다른 문항들은 두 개의 버튼이 있기 때문에 양쪽으로 배치합니다.

중앙 정렬은 `justify-content` 를 `center`로 설정합니다. 두 개의 버튼을 배치할 때는 `space-between` 으로 설정하여 요소 사이의 공간을 두게 합니다.

```
.btn_wrap {
    display: flex;
    justify-content: center;
}

.btn_sort{
    justify-content: space-between;
}
```



`justify-content` 의 `space-between`은 요소들 사이에 동일한 간격을 두고 배치됩니다.

2개의 버튼을 감싸고 있던 `div` 의 class에 `btn_sort`를 추가로 입력합니다.

```
<div class="btn_wrap btn_sort">
    <button class="prev_btn">이전</button>
    <button class="next_btn">다음</button>
</div>
```

아래와 같이 "제출" 버튼을 포함한 모든 버튼들의 스타일을 설정합니다.

```
button, .submit_btn {
    width: 100px;
    height: 40px;
    padding: 5px;
    border-style: none;
    border-radius: 10px;
    font-family: "NEXON Lv1 Gothic OTF";
    font-size: 20px;
    background-color: #7F47DD;
    color: #fff;
    cursor: pointer;
}
```

버튼에 마우스 커서를 가져갔을 때의 스타일도 함께 설정합니다. 여기서는 글자색과 배경색을 변경시켜 줍니다.

```
button:hover, .submit_btn:hover{
    color: #7F47DD;
    background-color: #fff;
}
```

3.2.5 추가 작업

설문지페이지에서는 마우스 스크롤이 아닌 "이전", "다음" 버튼으로만 화면이 이동되도록 할 것입니다. `body` 에 `overflow: hidden` 을 추가로 입력하여 넘친 부분을 잘라내어 스크롤이 보이지 않도록 합니다.

```
body {
    background-color: #FAF1BE;
    color: #000;
    overflow: hidden;
    font-family: "NEXON Lv1 Gothic OTF";
}
```

3.3 실행화면

현재 실행화면입니다. 이제 "다음" 버튼을 클릭하면 다음 문항으로 넘어가도록 기능을 추가해야 합니다. 기능 추가는 Javascript를 활용하는 다음 파트인 "기능 추가하기"에서 진행하겠습니다.

1/3

당신이 가장 재밌었던 수업은?

- 게임 개발(유니티, 언리얼, Pygame)
- 백엔드 또는 인프라(C계열-도네트, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)
- 정보보안(해킹과 방어)
- 프론트엔드(HTML, CSS, Javascript, etc)
- 데이터 분석과 인공지능(잔차리, 분석, 시각화, 머신러닝, 딥러닝)
- 이 같은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.

다음

4. 기능 추가하기

4.1 js 파일 만들기

여러가지 기능을 추가하기 위해 js 폴더를 만들고 그 안에 `form.js` 파일을 생성합니다. 그리고 `form.html`에 적용합니다. 아래 코드는 `body` 태그 내 제일 하단에 입력합니다.

```
<script type="text/javascript" src="js/form.js"></script>
```

그리고 jQuery를 사용할 것이기 때문에 `form.html`에 아래와 같은 코드를 추가합니다. 이 코드는 `<head></head>` 안에 `title` 태그 위에 입력합니다.

```
<script src="https://code.jquery.com/jquery-3.5.1.js"></script>
```



jQuery를 사용하는 방법은 대표적으로 API를 직접 다운로드 받아 사용하는 방법과 CDN을 이용하는 방법이 있습니다. 본 실습에서는 CDN을 이용하여 진행합니다.

4.2 슬라이드 효과

4.2.1 scrollDown()

"다음" 버튼을 클릭하였을 때 다음 문항으로 넘어가도록 기능을 추가하기 위해 `scrollDown()` 함수를 정의합니다.

```
function scrollDown() {  
}
```

우리는 `animate` 메소드의 `scrollTop` 속성을 사용하여 스크롤의 위치를 이동시켜 다음 문항으로 넘어가도록 할 것입니다.

스크롤은 `.test` 의 height만큼 이동되어야 하기 때문에 현재 `.test` 의 height 값을 변수 `vheight`에 저장합니다.

```
function scrollDown() {  
    const vheight = $('.test').height();  
}
```

`scrollTop` 속성에는 스크롤이 이동될 값을 입력해야 하는데 각 문항의 스크롤 위치는 모두 다릅니다. 이때 각 문항의 스크롤 위치의 변화를 생각하면 스크롤 위치값을 지정할 수 있습니다.

예를 들어 `.test` 의 높이가 700px일 때, 첫번째 문항에서 다음 문항으로 넘어가기 위해서는 700px 만큼 스크롤을 아래로 이동시키면 됩니다. 그리고 그 다음 문항도 마찬가지입니다.

처음 스크롤 위치는 0이고 그 다음은 700px, 1400px, ..., 이렇게 스크롤은 `.test` 의 높이의 배수로 이동하게 됩니다.

`vheight`의 배수로 이동시키기 위해 `$(window).scrollTop()` 을 `vheight`로 나누고, 그 나눈값을 `Math.floor()` 함수를 사용하여 내림합니다. 그 결과값에 1을 더하고 마지막으로 `vheight`를 곱합니다.

```
function scrollDown() {  
    const vheight = $('.test').height();  
    $('html, body').animate({  
        scrollTop: ((Math.floor($(window).scrollTop() / vheight)+1) * vheight)  
    }, 500);  
}
```

html과 body의 스크롤의 위치를 `animate`로 이동시켜 전체화면이 이동하도록 합니다. `animate`의 속성인 지속 시간을 500으로 설정하여 기본 설정(400)보다 느리게 움직이게 합니다.

이제 "다음" 버튼을 클릭하면 `scrollDown()` 함수를 호출해서 스크롤이 아래로 이동될 수 있도록 합니다. `$(function(){})` 안에서 함수를 호출하여 페이지가 로드되면 이벤트들이 실행될 수 있도록 합니다.

```
$(function(){
});
```

"다음" 버튼에 클릭 이벤트를 부여하기 위해 버튼의 클래스명인 `.next_btn` 을 선택합니다. 우선 `preventDefault()` 함수를 사용하여 버튼 클릭 시 발생할 수 있는 기본 동작이 중단되도록 하고, `scrollDown()` 함수를 호출합니다.

```
$(function(){
    $('.next_btn').click(function(e){
        e.preventDefault();
        scrollDown();
    });
});
```

4.2.2 scrollUp()

"이전" 버튼을 클릭하면 `scrollDown()` 함수와는 반대로 스크롤의 위치를 `.test`의 높이만큼 위로 이동시켜야 합니다.

위로 이동시키기 위해 `$(window).scrollTop()` 을 `vheight`로 나누고, 그 나눈값을 `Math.ceil()` 함수를 사용하여 올림합니다. 그리고 1을 빼준 후 `vheight`를 곱합니다.

```
function scrollUp() {
    const vheight = $('.test').height();
    $('html, body').animate({
        scrollTop: ((Math.ceil($(window).scrollTop() / vheight)-1) * vheight)
    }, 500);
};
```

앞에서 `.next_btn` 에 클릭 이벤트를 주었던 것과 마찬가지로 이번에는 "이전" 버튼인 `.prev_btn`에 클릭 이벤트를 추가하고 `scrollUp()` 함수를 호출합니다.

```
$(function(){
    $('.next_btn').click(function(e){
        e.preventDefault();
        scrollDown();
    });

    $('.prev_btn').click(function(e){
        e.preventDefault();
        scrollUp();
    });
});
```

4.3 유효성 검사

테스트에서 답안을 선택하지 않은 문항이 있다면 다음 문항으로 이동하거나 제출되지 않도록 유효성 검사 기능을 추가합니다.

4.3.1 다음 문항으로 넘어가기

"다음" 버튼을 클릭하면 답안이 선택되었는지 검사를 진행하기 위해

`$('.next_btn').click(function(e){})`에서 코드를 작성합니다. 우선 문제의 답안들을 배열로 가져와 변수 `divs`에 저장하겠습니다.

`$(this)`는 클릭 이벤트가 발생한 요소들의 정보인 object로 여기서는 클릭된 버튼을 의미합니다. `$(this).parent()`는 버튼의 부모인 버튼을 감쌌던 `div` 태그이고, `$(this).parent().prev()`는 `div`의 이전 태그를 선택하는데 이것은 답안들을 감싼 또 다른 `div` 태그입니다. 이때 선택된 `div` 태그의 자식들이 바로 우리가 가져올 답안 배열입니다.

따라서 `$(this).parent().prev().children()`는 문제의 답안들을 모두 배열로 가져올 수 있습니다.

```
let divs = $(this).parent().prev().children();
```

`find()` 함수를 통해 `divs` 배열에서 체크된 `input` 태그를 찾은 후 변수 `inputs`에 저장합니다. 그리고 모든 문항이 선택되지 않았을 경우는 `inputs` 변수의 길이가 1보다 작을 경우와 동일하므로 이때 "문항이 선택되지 않았습니다."라는 문구가 나오는 경고창을 띄웁니다.

`return false`를 추가하여 다음 문항으로 넘어가는 것을 방지합니다.

```
$('.next_btn').click(function(e){
    let divs = $(this).parent().prev().children();
    let inputs = divs.find('input:checked');
    if(inputs.length < 1) {
        alert('문항이 선택되지 않았습니다.');
        return false;
    }
    e.preventDefault();
    scrollDown();
});
```

4.3.2 제출하기

모든 문항을 풀고 "제출" 버튼을 누르면 테스트지가 서버로 전송됩니다. 이때도 앞에서 유효성 검사를 진행했던 것과 같이 답안이 선택되지 않은 문항이 있을 경우가 있는지 검사를 진행해야 합니다.

체크된 모든 `input` 태그를 `radios`라는 변수에 저장하고 그 변수의 길이가 3보다 작을 경우 경고창을 띄우도록 합니다.

```
$("#form").submit(function() {
    let radios = $('input[type=radio]:checked');
    if(radios.length < 3) {
        alert("문항이 선택되지 않았습니다.");
        return false;
    }
    return true;
});
```



이때 `radios.length < 3`의 3은 문항 수를 의미합니다. 그렇기 때문에 백엔드 실습 진행시에는 10으로 변경시켜 주어야 합니다.

4.4 새로고침

새로고침(F5) 시 현재 화면에서 이동되지 않습니다. 이럴 경우 다시 1번부터 문제를 풀어야 하기 때문에 계속 "이전" 버튼을 클릭하여 이전 문항으로 이동해야 하는 문제점이 있습니다. 이를 해결하기 위해 `$(function(){})`; 내부에 아래 코드를 추가하여 **로드 시 스크롤 위치가 0으로 이동할 수 있도록** 합니다.

```
$("html, body").animate({
  scrollTop: 0
}, 500);
```

5. 전체 코드

5.1 form.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" type="text/css" href="css/reset.css">
  <link rel="stylesheet" type="text/css" href="css/form.css">
  <script src="https://code.jquery.com/jquery-3.5.1.js"></script>
  <script type="text/javascript" src="js/form.js"></script>
  <title>나의 개발 유형 찾기</title>
</head>
<body>
  <section id="survey">
    <div class="wrapper">
      <form id="form">
        <div class="test">
          <div class="question_container">
            <h3 class="number">1/3</h3>
            <h3 class="question">당신이 가장 재밌었던 수업은?</h3>
          </div>
          <div class="answer">
            <div>
              <input type="radio" name="answer_1" />
              <label>게임 개발(유니티, 언리얼, Pygame)</label>
            </div>
            <div>
              <input type="radio" name="answer_1" />
              <label>백엔드 또는 인프라(C계열-닷넷, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)</label>
            </div>
          </div>
        </div>
      </form>
    </div>
  </section>
</body>
```

```

<div>
    <input type="radio" name="answer_1" />
    <label>정보보안(해킹과 방어)</label>
</div>
<div>
    <input type="radio" name="answer_1" />
    <label>프론트엔드(HTML, CSS, Javascript, etc)</label>
</div>
<div>
    <input type="radio" name="answer_1" />
    <label>데이터 분석과 인공지능(전처리, 분석, 시각화, 머신러닝, 딥
러닝)</label>
</div>
<div>
    <input type="radio" name="answer_1" />
    <label>이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.
</label>
</div>
<div>
    <input type="radio" name="answer_1" />
    <label>이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.
</label>
</div>
<div class="btn_wrap">
    <button class="next_btn">다 음</button>
</div>
</div>
<div class="test">
    <div class="question_container">
        <h3 class="number">2/3</h3>
        <h3 class="question">여러분의 서비스가 성공하였다면, 서비스를 성공시
킨 요소는?</h3>
        </div>
        <div class="answer">
            <div>
                <input type="radio" name="answer_2" />
                <label>데이터 분석을 통한 효과적인 개인 맞춤형 서비스</label>
            </div>
            <div>
                <input type="radio" name="answer_2" />
                <label>화려하거나 품격있는 디자인, 흥미로운 콘텐츠 제작</label>
            </div>
            <div>
                <input type="radio" name="answer_2" />
                <label>게임적인(Gamification, 게이미피케이션) 요소 도입</label>
            </div>
            <div>
                <input type="radio" name="answer_2" />
                <label>안정적인 서비스 운영</label>
            </div>
            <div>
                <input type="radio" name="answer_2" />
                <label>믿을 수 있는 정보 관리</label>
            </div>
        </div>
    </div>

```

```

<div class="btn_wrap">
    <button class="prev_btn">이 전</button>
    <button class="next_btn">다 음</button>
</div>
</div>
<div class="test">
    <div class="question_container">
        <h3 class="number">3/3</h3>
        <h3 class="question">당신은 친구의 집들이에 가려고 합니다. 당신이 선택한 선물은?</h3>
        </div>
        <div class="answer">
            <div>
                <input type="radio" name="answer_10" />
                <label>집 꾸미기에 도움이 되는 예쁜 인테리어 소품을 골라볼까?</label>
            </div>
            <div>
                <input type="radio" name="answer_10" />
                <label>실용성이 같아지! 화장지랑 수건 세트랑...</label>
            </div>
            <div>
                <input type="radio" name="answer_10" />
                <label>음... 우선 센스있는 집들이 선물 먼저 검색해봐야지!</label>
            </div>
            </div>
            <div>
                <input type="radio" name="answer_10" />
                <label>역시 집은 안전이 최고!! 현관문 보안장치 사줘야겠다.</label>
            </div>
        </div>
        <div class="btn_wrap">
            <button class="prev_btn">이 전</button>
            <input type="submit" value="제출" class="submit_btn"/>
        </div>
    </div>
</form>
</div>
</section>
</body>
</html>

```

5.2 form.css

```

@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
        format("woff");
    font-weight: normal;
    font-style: normal;
}

body {
    background-color: #FAF1BE;
    color: #000;
    overflow: hidden;
    font-family: "NEXON Lv1 Gothic OTF";
}

#survey{
    display: flex;
    justify-content: center;
    width: 100vw;
    height: 100vh;
    margin-top: 50px;
}

.test {
    max-width: 600px;
    height: 100vh;
}

.question_container {
    display: flex;
    flex-direction: column;
    align-items: center;
    font-size: 30px;
    line-height: 1.3;
    padding-bottom: 30px;
}

```

```
.question_container .number {  
    padding-bottom: 30px;  
}  
  
.question_container .question {  
    font-family: "GmarketSansBold";  
}  
  
.answer {  
    font-size: 17px;  
    line-height: 1.3;  
    padding-bottom: 30px;  
}  
  
.answer div {  
    padding-bottom: 15px;  
}  
  
.answer input {  
    cursor: pointer;  
}  
  
.btn_wrap {  
    display: flex;  
    justify-content: center;  
}  
  
.btn_sort{  
    justify-content: space-between;  
}  
  
button, .submit_btn {  
    font-family: "NEXON Lv1 Gothic OTF";  
    font-size: 20px;  
    width: 100px;  
    height: 40px;  
    padding: 5px;  
    border-style: none;  
    border-radius: 10px;  
    background-color: #7F47DD;  
    color: #fff;  
    cursor: pointer;  
}  
  
button:hover, .submit_btn:hover{  
    color: #7F47DD;  
    background-color: #fff;  
}
```

5.3 form.js

```

function scrollUp() {
    const vheight = $('.test').height();
    $('html, body').animate({
        scrollTop: ((Math.ceil($(window).scrollTop() / vheight)-1) * vheight)
    }, 500);
}

function scrollDown() {
    const vheight = $('.test').height();
    $('html, body').animate({
        scrollTop: ((Math.floor($(window).scrollTop() / vheight)+1) * vheight)
    }, 500);
}

$(function(){
    $('.next_btn').click(function(e){
        let divs = $(this).parent().prev().children();
        let inputs = divs.find('input:checked');
        if(inputs.length < 1) {
            alert('문항이 선택되지 않았습니다.');
            return false;
        }
        e.preventDefault();
        scrollDown();
    });

    $('.prev_btn').click(function(e){
        e.preventDefault();
        scrollUp();
    });

    $("#form").submit(function() {
        let radios = $('input[type=radio]:checked');
        if(radios.length < 10) {
            alert("문항이 선택되지 않았습니다.");
            return false;
        }
        return true;
    });

    $("html, body").animate({
        scrollTop: 0
    }, 500);
});

```



003. 결과페이지 만들기

1. 화면 레이아웃 구성하기

1.1 우리가 만든 화면

1.2 화면 나누기

2. HTML

2.1 기본 구조

2.2 태그 입력하기

2.3 id, class 입력하기

2.4 콘텐츠 넣기

2.4.1 결과

2.4.2 공유

2.4.1 버튼

2.4.1 로고

3. CSS 적용하기

3.1 CSS 파일 만들기

3.2 스타일 적용하기

3.2.1 기본 설정

3.2.2 결과

3.2.3 SNS 공유

3.2.4 버튼

3.2.5 로고

3.3 실행화면

4. 기능 추가하기

4.1 js 파일 만들기

4.2 Open graph(오픈 그래프, og 태그) 설정하기

4.3 URL 복사하기

4.4 페이스북 공유하기

4.5 카카오톡 공유하기

4.5.1 카카오 API 사용하기

4.5.2 SDK 포함 및 초기화하기

4.5.3 기능 추가하기

5. 전체 코드

5.1 result.html

5.2 result.css

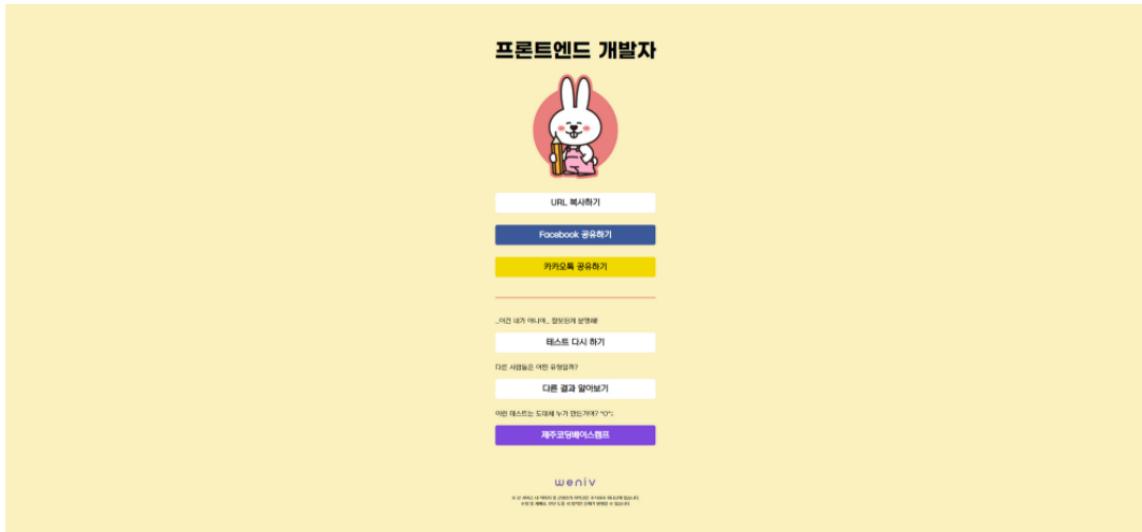
5.3 result.js

6. Update된 최종 Code (UI 업데이트)



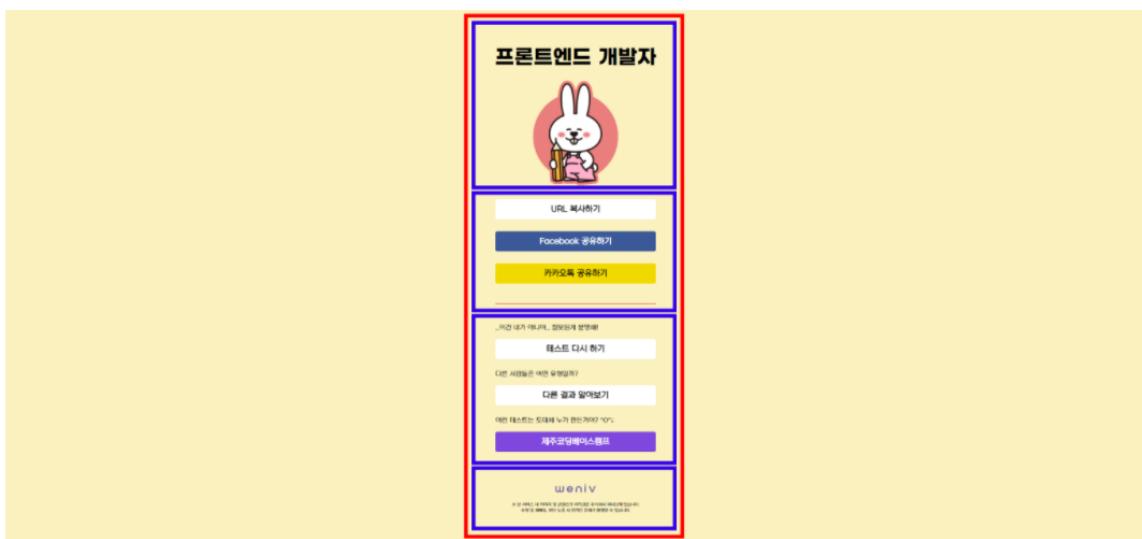
1. 화면 레이아웃 구성하기

1.1 우리가 만들 화면



1.2 화면 나누기

화면 구성을 아래와 같이 나눕니다.



2. HTML

2.1 기본 구조

결과페이지를 만들기 위해 `result.html` 파일을 생성합니다. 기본 구조는 앞서 만들었던 메인페이지 및 설문페이지와 동일합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>나의 개발 유형찾기</title>
</head>
<body>

</body>
</html>
```

2.2 태그 입력하기

콘텐츠들을 넣을 `section` 과 콘텐츠들을 감쌀 `div` 하나, 그리고 실제 콘텐츠들을 넣을 `div` 4개로 구성됩니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section>
        <div>
            <div></div>
            <div></div>
            <div></div>
            <div></div>
        </div>
    </section>
</body>
</html>
```

2.3 id, class 입력하기

CSS 적용을 위해 id와 class명을 각 태그에 입력합니다.

```
<section id="main_contents">
  <div class="wrapper">
    <div class="result"></div>
    <div class="share"></div>
    <div class="buttons"></div>
    <div class="weniv"></div>
  </div>
</section>
```

2.4 콘텐츠 넣기

2.4.1 결과

테스트 결과는 개발 유형과 함께 이미지로 구성되어 있으며 `<div class="result"></div>` 안에 배치합니다. 개발 유형은 `h1` 태그, 이미지는 `img` 태그를 사용합니다.

```
<div class="result">
  <div class="titles">
    <h1>프론트엔드 개발자</h1>
  </div>
  <div class="result_img">
    
  </div>
</div>
```

2.4.2 공유

테스트 결과를 공유하기 위해 3개의 공유 버튼을 만들어 줍니다. "URL 복사하기", "Facebook 공유하기", 그리고 "카카오톡 공유하기"입니다. 3개의 버튼을 각 `div` 안에 `button` 태그를 사용하여 넣습니다.

```
<div class="share">
  <div class="url">
    <button class="copy_btn" type="button">URL 복사하기</button>
  </div>
  <div class="facebook">
    <button class="facebook_share" type="button">Facebook 공유하기</button>
  </div>
  <div class="kakao">
    <button class="kakao_share" type="button">카카오톡 공유하기</button>
  </div>
</div>
```

2.4.1 버튼

공유 버튼 하단에는 3개의 또 다른 버튼들이 있습니다. 이 버튼들은 `ul` 과 `li` 를 사용하여 배치하겠습니다.

각 `li` 안에는 문구를 넣어줄 `h3` 태그와 버튼에 링크를 연결시킬 `a` 태그를 배치합니다. 그리고 `a` 태그 안에는 `button` 태그를 통해 버튼을 만듭니다.

```
<div class="buttons">
  <ul>
    <li>
      <h3>...이건 내가 아니야... 잘못된게 분명해!</h3>
      <a href="#">
        <button type="button">테스트 다시 하기</button>
      </a>
    </li>
    <li>
      <h3>다른 사람들은 어떤 유형일까?</h3>
      <a href="#">
        <button type="button">다른 결과 알아보기</button>
      </a>
    </li>
    <li>
      <h3>이런 테스트는 도대체 누가 만든거야? ^0^;;</h3>
      <a href="#">
        <button type="button">제주코딩베이스캠프</button>
      </a>
    </li>
  </ul>
</div>
```

2.4.1 로고

결과페이지에서도 빠질 수 없는 위니브 로고를 배치합니다. 이전에 해왔던 방법과 마찬가지로 `a` 태그 안에 `img` 태그를 사용하여 위니브 로고 이미지를 넣어줍니다. 그리고 이미지 하단에는 `p` 태그를 사용하여 저작권 문구를 넣습니다.

```
<div class="weniv">
  <a href="http://www.paullab.co.kr">
    
  </a>
  <p>
    ※ 본 서비스 내 이미지 및 콘텐츠의 저작권은 주식회사 위니브에 있습니다.<br>
    수정 및 재배포, 무단 도용 시 법적인 문제가 발생할 수 있습니다.
  </p>
</div>
```



실제 여러분의 테스트 페이지를 제작할 때는 로고와 문구 등을 수정하여 제작해 보시길 바랍니다.

3. CSS 적용하기

3.1 CSS 파일 만들기

결과페이지를 꾸며주기 위한 `result.css` 파일을 css 폴더 내에 만듭니다. 앞서 만들었던 `reset.css` 와 `result.css` 파일을 `result.html`에 아래와 같이 적용시킵니다.

```
<link rel="stylesheet" type="text/css" href="css/reset.css">
<link rel="stylesheet" type="text/css" href="css/form.css">
```

3.2 스타일 적용하기

3.2.1 기본 설정

이전 챕터에서 기본 스타일 설정을 했던 것과 같이 `@font-face` 를 통해 폰트를 적용합니다.

```
@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
    format("woff");
    font-weight: normal;
    font-style: normal;
}
```

결과페이지의 배경색, 글자색, 그리고 기본 폰트 설정을 아래와 같이 합니다.

```
body {
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}
```

모든 콘텐츠들을 담고 있는 `#main_contents` 의 스타일을 설정합니다. 우선 `display` 를 flex로 설정하고, `justify-content` 를 center로 설정하여 콘텐츠들을 중앙으로 배치합니다.

`width` 와 `height` 는 모두 100%로 설정합니다. 그리고 `margin-top` 을 통해 위에 100px만큼의 여백을 만들어 줍니다.

```
#main_contents {
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100%;
    margin-top: 100px;
}
```

콘텐츠를 다시 한 번 감싸주는 `<div class="wrapper"></div>` 를 `display` 를 flex로 설정합니다. 그리고 `flex-direction` 속성을 column으로 설정하여 요소들의 배치 방향을 수직으로 변경시킵니다.

`width` 와 `height` 를 모두 100%로 설정하고 `max-width` 를 사용하여 600px 만큼의 최대 너비를 설정합니다.

```
#main_contents .wrapper {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
    height: 100%;
    max-width: 600px;
    padding-bottom: 50px;
}
```

3.2.2 결과

테스트 결과로 나온 개발 유형과 이미지를 중앙 정렬시킵니다. `display` 를 `flex`로 설정하고 `flex-direction` 을 `column`으로 설정하여 요소들을 수직으로 배치합니다. 그리고 `align-items` 를 `center`로 설정하여 콘텐츠들이 중앙으로 오게 합니다.

```
.result {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

결과로 나온 개발 유형은 폰트 사이즈와 글꼴을 변경시켜 강조합니다.

```
.result h1 {
    font-size: 50px;
    font-family: "GmarketSansBold";
}
```

결과 이미지를 감싼 `div` 의 `width` 와 `height` 를 각각 300px, 324px로 설정합니다. `img` 태그의 `width` 와 `height` 는 모두 100%로 설정하여 앞서 설정한 `div` 의 너비와 높이만큼 차지하도록 합니다.

```
.result_img {
    width: 300px;
    height: 324px;
}

.result_img img {
    width: 100%;
    height: 100%;
}
```

3.2.3 SNS 공유

이제 테스트 결과를 공유할 수 있는 버튼들의 스타일을 설정합시다. 공유 버튼뿐만 아니라 3.2.4 버튼에서 지정할 모든 버튼들의 기본 스타일은 모두 같기 때문에 `button` 태그의 스타일 설정을 먼저 하겠습니다.

아래 코드와 같이 글꼴, 폰트 사이즈, 너비와 높이 등 스타일을 설정합니다.

```
button {
    font-family: "NEXON Lv1 Gothic OTF";
    font-size: 20px;
    font-weight: bold;
    width: 400px;
    height: 50px;
    padding: 5px;
    border-style: none;
    border-radius: 5px;
    cursor: pointer;
    background-color: #fff;
    color: #000;
}
```

이어서 버튼에 마우스 커서를 가져갔을 때의 스타일도 함께 설정합니다.

```
button:hover {
    text-decoration: underline;
}
```

이제 공유 버튼들을 감싼 `div` 의 스타일을 설정합니다. `padding-bottom` 과 `margin-bottom` 을 각각 20px, 50px로 설정하여 하단에 여백을 줍니다.

또한 하단 다른 버튼들과 구분을 주기 위해 `border-bottom` 을 사용하여 구분선을 넣습니다.

```
.share {
    padding-bottom: 20px;
    margin-bottom: 50px;
    border-bottom: 3px solid #ea7e7c;
}
```

여기서 `padding-bottom` 은 구분선의 위쪽 여백을, `margin-bottom` 은 구분선의 아래 여백을 설정합니다.

각 버튼들은 `margin-bottom` 을 사용하여 30px만큼의 여백을 줍니다.

```
.share_button {
    margin-bottom: 30px;
}
```

"Facebook 공유하기" 버튼은 페이스북의 색상(#3B5998)을 가져와 배경색으로 지정하고 글자색은 흰색으로 설정합니다.

```
.facebook_share {
    background-color: #3b5998;
    color: #fff;
}
```

"카카오톡 공유하기" 버튼도 마찬가지로 카카오톡의 색상(#F1D900)을 가져와 배경색으로 지정합니다. 이때 글자색은 변경하지 않습니다.

```
.kakao_share {
    background-color: #f1d900;
}
```

3.2.4 버튼

3개의 버튼을 감싼 `div` 의 `padding-bottom` 을 50px로 설정하여 아래 로고 이미지와 간격을 줍니다.

```
.buttons {
    padding-bottom: 50px;
}
```

각 버튼 위 문구의 `padding-bottom` 을 20px로 설정하여 아래 버튼과 간격을 줍니다.

```
.buttons h3 {
    padding-bottom: 20px;
}
```

각 버튼들은 `margin-bottom` 을 사용하여 30px만큼 간격을 띄워줍니다.

```
.buttons button {
    margin-bottom: 30px;
}
```

3개의 버튼 중 마지막 버튼에만 배경색과 글자색을 변경시키고 싶습니다. 아래와 같은 코드를 작성하고 마지막 버튼에는 `class="color"` 를 추가합니다.

```
.color {
    background-color: #7F47DD;
    color: #fff;
}
```

```
<button class="color" type="button">제주코딩베이스캠프</button>
```

3.2.5 로고

`display` 를 flex로 설정하고 `flex-direction` 을 column으로 설정하여 요소들을 수직으로 배치시킵니다. 그리고 `align-items` 를 center로 설정하여 콘텐츠들이 중앙으로 오게 합니다.

```
.weniv {
    display: flex;
    flex-direction: column;
    align-items: center;
}
```

로고 이미지에 링크를 연결하고 크기를 조절하기 위해 이미지를 감싸고 있는 `a` 태그의 `width` 를 100px로 설정합니다. 그리고 `img` 의 `width` 와 `height` 를 각각 100%, auto로 설정합니다.

```
.weniv a {
    width: 100px;
}

.weniv img{
    width: 100%;
    height: auto;
}
```

로고 이미지 하단에 들어갈 문구는 아래와 같이 스타일을 설정합니다.

```
.weniv p {
    padding-top: 20px;
    font-size: 10px;
    text-align: center;
    line-height: 1.5;
}
```

3.3 실행화면

프론트엔드 개발자



URL 복사하기

Facebook 공유하기

카카오북 공유하기

...이건 내가 아니라... 잘못된거 문제예

테스트 다시 하기

다른 사람들은 어떤 유형인가?

다른 결과 알아보기

이런 테스트는 도대체 누가 만드거야? "도

제주도당메이스晨报

weniv

A front-end developer's blog about web development.

4. 기능 추가하기

4.1 js 파일 만들기

공유 기능을 추가하기 위해 js 폴더 안에 `result.js` 파일을 생성합니다. 그리고 `result.html`에 적용합니다. 아래 코드는 `body` 태그 내 제일 하단에 입력합니다.

```
<script type="text/javascript" src="js/result.js"></script>
```

그리고 jQuery를 사용하기 위해 `result.html`에 아래와 같은 코드를 추가합니다. 이 코드는 `head` 안에 `title` 태그 위에 입력합니다.

```
<script src="https://code.jquery.com/jquery-3.5.1.js"></script>
```

4.2 Open graph(오픈 그래프, og 태그) 설정하기

공유기능을 추가하기 전에 og 태그 설정을 먼저 진행하겠습니다. open graph는 html의 메타 태그 중 하나로 공유한 링크의 제목, 설명, 미리보기 이미지(썸네일) 등을 설정할 수 있습니다.

우선 아래와 같이 `<head></head>` 안에 메타 데이터를 입력합니다. 테스트 결과에 따라 이미지와 주소가 바꿔므로 image와 url의 content는 비워두고 `result.js`에서 넣도록 하겠습니다.

```
<meta property="og:title" content="나의 개발 유형은?" />
<meta property="og:image" content="" />
<meta property="og:url" content="" />
<meta property="og:description" content="나에게 꼭 맞는 개발 유형은 무엇일까?" />
```



og:title - 웹사이트의 제목

og:image - 웹사이트의 미리보기 이미지(썸네일)로 이미지의 경로가 들어감

og:url - 웹사이트의 주소

og:description - 웹사이트의 설명

`result.js` 에서 아래와 같은 코드를 작성합니다. 현재 url을 변수 `url`에 저장하고 결과 이미지 경로를 변수 `img`에 정합니다. 그리고 `meta` 태그를 선택하고 content 속성값으로 url과 img를 삽입합니다.

```
$($.function() {
    let url = window.location.href;
    let img = $('.result_img img').attr('src');

    $("meta[property='og\\:url']").attr("content", url);
    $("meta[property='og\\:image']").attr("content", img);
});
```

4.3 URL 복사하기

"URL 복사하기" 버튼을 클릭하면 현재 url이 클립보드에 복사되는 기능을 추가해야 합니다. 우선 버튼을 `.querySelector()`로 선택하여 변수에 저장합니다.

```
const copyBtn = document.querySelector('.copy_btn');
```

현재 url이 클립보드에 복사되는 기능을 수행할 함수인 `copyUrl()`을 선언합니다. 그리고 함수 내부에 `url`이라는 변수를 생성하여 현재 url을 저장합니다.

```
function copyUrl() {
    let url = window.location.href;
}
```

`url` 을 클립보드에 저장하기 위해 임시로 `input` 태그를 생성하여 `input` 태그의 value 값에 url을 삽입합니다. `input` 태그를 `.select()` 함수로 선택하고 url이 들어있는 `value` 값을 복사합니다. 그리고 복사가 되면 임시로 만들었던 `input` 태그를 제거합니다.

마지막으로 `alert()` 를 사용하여 "URL이 복사되었습니다"라는 문구가 담겨있는 경고창이 나오게 하여 url 복사가 완료되었음을 알립니다.

```
function copyUrl() {
    let tmp = document.createElement('input');
    let url = location.href;

    document.body.appendChild(tmp);
    tmp.value = url;
    tmp.select();
    document.execCommand("copy");
    document.body.removeChild(tmp);

    alert("URL이 복사되었습니다");
}
```

버튼을 클릭하면 `copyUrl()` 함수가 실행되도록 이벤트를 추가합니다.

```
copyBtn.addEventListener('click', copyUrl);
```

4.4 페이스북 공유하기

"Facebook 공유하기" 버튼을 클릭하면 페이스북으로 공유되도록 하기 위해 해당 버튼을 선택하여 변수 `facebookShare` 에 저장합니다.

```
const facebookShare = document.querySelector('.facebook_share');
```

`sharefacebook()` 함수를 정의하고 그 안에 현재 url을 변수 `url`에 저장합니다.

페이스북 공유하기 스크립트인 '`http://www.facebook.com/sharer/sharer.php?u=`' 를 변수 `facebook`에 저장합니다. 그리고 `facebook`과 `url`을 합쳐서 변수 `link`에 저장한 후 `window.open(link)`를 통해 해당 링크를 열어줍니다.

```
function sharefacebook() {
    let url = window.location.href;
    let facebook = 'http://www.facebook.com/sharer/sharer.php?u=';
    let link = facebook + url;
    window.open(link);
}
```

버튼을 클릭하면 `sharefacebook()` 함수가 실행되도록 이벤트를 추가합니다.

```
facebookShare.addEventListener('click', sharefacebook);
```

4.5 카카오톡 공유하기

 카카오톡 공유하기 기능은 URL이 있어야 테스트가 가능하기 때문에 백엔드 실습 이후에 진행하는 것을 권장합니다.

4.5.1 카카오 API 사용하기

카카오톡 공유하기는 이전 공유하기 기능과 달리 "카카오 API"를 사용해야 합니다. 카카오톡 링크 공유하기 기능을 추가하기 위해서는 Kakao Developers(<https://developers.kakao.com>)에서 제공하는 Javascript SDK를 사용해야 합니다.

JavaScript 키가 있어야 카카오 API 사용이 가능하기 때문에 로그인 후 [내 애플리케이션]에서 애플리케이션을 추가하도록 합니다.

애플리케이션 추가하기

앱 아이콘	<input type="button" value="파일 선택"/> 이미지 업로드 JPG, GIF, PNG 권장 사이즈 128px, 최대 250KB
앱 이름	<input type="text" value="내 애플리케이션 이름"/>
사업자명	<input type="text" value="사업자 정보와 동일한 이름"/>

• 입력된 정보는 사용자가 카카오 로그인을 할 때 표시됩니다.
 • 정보가 정확하지 않은 경우 서비스 이용이 제한될 수 있습니다.

애플리케이션을 추가하면 앱 키가 부여된 것을 확인할 수 있습니다. 우리는 "JavaScript 키"를 사용할 것입니다. (키는 사용자마다 다르게 부여됩니다.)

<u>앱 키</u>	
네이티브 앱 키	[REDACTED]
REST API 키	[REDACTED]
<input checked="" type="checkbox"/> JavaScript 키	[REDACTED]
Admin 키	[REDACTED]

[플랫폼]으로 들어가면 자신의 사이트 도메인을 추가할 수 있습니다. "Web 플랫폼 등록"을 눌러 사이트 도메인을 추가합니다.

Web

Web 플랫폼 등록

Kakao Developers에서 [문서] - [메시지]로 들어가면 API 사용법이 상세하게 설명되어 있습니다. 여기서 우리는 Javascript를 사용할 것이니 [JavaScript] 탭으로 이동합니다.

아래로 내리면 "카카오링크 API로 보내기" 방법이 설명되어 있습니다. 이곳에는 두 가지 방법을 설명을 하지만 우리는 서비스 웹페이지에서 자체적으로 띄운 공유 버튼 클릭 시 해당 함수를 호출해 메시지 보내기 요청을 하도록 만든 `Kakao.Link.sendDefault`를 사용합니다.

아래와 같이 제공되는 sample 코드를 사용하도록 하겠습니다.

```
Kakao.Link.createDefaultButton({
  container: '#CONTAINER_ID',
  objectType: 'feed',
  content: {
    title: '디저트 사진',
    description: '아메리카노, 빵, 케익',
    imageUrl:
      'http://mud-kage.kakao.co.kr/dn/NTmhS/btqfEUdFAUF/FjKzkZsnoeE4o19k1TOVI1/openlink_640x640s.jpg',
    link: {
      mobileWebUrl: 'https://developers.kakao.com',
      androidExecParams: 'test',
    },
  },
  social: {
    likeCount: 10,
    commentCount: 20,
    sharedCount: 30,
  },
});
```

```

buttons: [
  {
    title: '웹으로 이동',
    link: {
      mobileWebUrl: 'https://developers.kakao.com',
    },
  },
  {
    title: '앱으로 이동',
    link: {
      mobileWebUrl: 'https://developers.kakao.com',
    },
  },
]
});

```

4.5.2 SDK 포함 및 초기화하기

웹 페이지에 JavaScript SDK를 포함하기 위해 아래 코드를 `result.html`에 추가합니다.

```
<script src="https://developers.kakao.com/sdk/js/kakao.js"></script>
```

SDK 초기화를 위해 함수를 호출하고 초기화가 잘 되었는지 확인합니다. 이때 JavaScript 키는 자신의 앱 키를 입력합니다.

```
Kakao.init('JavaScript 키 입력');
Kakao.isInitialized();
```

`Kakao.isInitialized()` 는 초기화가 잘 되었는지 확인 후 주석처리합니다.

4.5.3 기능 추가하기

"카카오톡 공유하기" 버튼을 클릭하면 카카오톡 메시지로 링크를 공유할 수 있도록 버튼을 선택하고 변수 `kakaoShare`에 저장합니다.

```
const kakaoShare = document.querySelector('.kakao_share');
```

`sendLink()` 함수를 선언하고 4.5.1에서 sample 코드를 가져온 후 아래와 같이 내용을 수정합니다.

```
function sendLink() {
  Kakao.Link.sendDefault({
    objectType: 'feed',
    content: {
      title: '나의 개발 유형은?',
      description: '나에게 꼭 맞는 개발 유형을 알아보자!!',
      imageUrl:
        '웹페이지 url 입력',
      link: {
        mobileWebUrl: '웹페이지 url 입력',
        webUrl: '웹페이지 url 입력',
      },
    },
    social: {
      likeCount: 286,
      commentCount: 45,
      sharedCount: 845,
    },
    buttons: [
      {
        title: '결과 보러가기',
        link: {
          webUrl: '웹페이지 url 입력',
          mobileWebUrl: '웹페이지 url 입력',
        },
      },
      {
        title: '테스트 하러가기',
        link: {
          webUrl: '웹페이지 url 입력',
          mobileWebUrl: '웹페이지 url 입력',
        },
      },
    ],
  });
}
```

"결과 보러가기" 버튼을 클릭하면 결과페이지로 이동될 수 있도록 `result_url` 이라는 변수를 생성하고 현재 url을 저장합니다. 그리고 '결과 보러가기'의 아래 link 값을 `result_url`로 변경합니다.

```
function sendLink() {
  let result_url = window.location.href;
  Kakao.Link.sendDefault({
    objectType: 'feed',
    content: {
      title: '나의 개발 유형은?',
      description: '나에게 꼭 맞는 개발 유형을 알아보자!!',
      imageUrl:
        '웹페이지 url 입력',
      link: {
        mobileWebUrl: '웹페이지 url 입력',
        webUrl: '웹페이지 url 입력',
      },
    },
    social: {
      likeCount: 286,
      commentCount: 45,
      sharedCount: 845,
    },
    buttons: [
      {
        title: '결과 보러가기',
        link: {
          webUrl: result_url,
          mobileWebUrl: result_url,
        },
      },
      {
        title: '테스트 하러가기',
        link: {
          webUrl: '웹페이지 url 입력',
          mobileWebUrl: '웹페이지 url 입력',
        },
      },
    ],
  });
}
```

"카카오톡 공유하기" 버튼을 클릭하면 `sendLink()` 함수가 실행되도록 이벤트를 추가합니다.

```
kakaoShare.addEventListener('click', sendLink);
```

5. 전체 코드

5.1 result.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta property="og:title" content="나의 개발 유형은?" />
    <meta property="og:image" content="" />
    <meta property="og:url" content="" />
    <meta property="og:description" content="나에게 꼭 맞는 개발 유형은 무엇일까?" />
    <script src="https://developers.kakao.com/sdk/js/kakao.js"></script>
    <link rel="stylesheet" type="text/css" href="css/reset.css">
    <link rel="stylesheet" type="text/css" href="css/result.css">
    <script src="https://code.jquery.com/jquery-3.5.1.js"></script>
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section id="main_contents">
        <div class="wrapper">
            <div class="result">
                <div class="titles">
                    <h1>프론트엔드 개발자</h1>
                </div>
                <div class="result_img">
                    
                </div>
            </div>
            <div class="share">
                <div class="url">
                    <button class="copy_btn" type="button">URL 복사하기</button>
                </div>
                <div class="facebook">
                    <button class="facebook_share" type="button">Facebook 공유하기</button>
                </div>
            </div>
        </div>
    </section>
</body>

```

```

<div class="kakao">
    <button class="kakao_share" type="button">카카오톡 공유하기</button>
</div>
<div class="buttons">
    <ul>
        <li>
            <h3>...이건 내가 아니야... 잘못된게 분명해!</h3>
            <a href="#">
                <button type="button">테스트 다시 하기</button>
            </a>
        </li>
        <li>
            <h3>다른 사람들은 어떤 유형일까?</h3>
            <a href="#">
                <button type="button">다른 결과 알아보기</button>
            </a>
        </li>
        <li>
            <h3>이런 테스트는 도대체 누가 만든거야? ^0^;;</h3>
            <a href="#">
                <button class="color" type="button">제주코딩베이스캠프</button>
            </a>
        </li>
    </ul>
</div>
<div class="weniv">
    <a href="http://www.paullab.co.kr">
        
    </a>
    <p>
        ※ 본 서비스 내 이미지 및 콘텐츠의 저작권은 주식회사 위니브에 있습니다.<br>
    </p>
    수정 및 재배포, 무단 도용 시 법적인 문제가 발생할 수 있습니다.
</div>
</section>
<script type="text/javascript" src="js/result.js"></script>
</body>
</html>

```

5.2 result.css

```

@font-face {
    font-family: "GmarketSansBold";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansBo
ld.woff") format('woff');
    font-weight: normal;
    font-style: normal;
}

@font-face {
    font-family: "NEXON Lv1 Gothic OTF";
    src: url("https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_20-04@2.1/NEXON Lv1 Go
thic OTF.woff")
        format("woff");
    font-weight: normal;
    font-style: normal;
}

body {
    background-color: #FAF1BE;
    color: #000;
    font-family: "NEXON Lv1 Gothic OTF";
}

button {
    font-family: "NEXON Lv1 Gothic OTF";
    font-size: 20px;
    width: 400px;
    height: 50px;
    padding: 5px;
    border-style: none;
    border-radius: 5px;
    cursor: pointer;
    font-weight: bold;
    background-color: #fff;
    color: #000;
}

button:hover {
    text-decoration: underline;
}

.color {
    background-color: #7F47DD;
    color: #fff;
}

```

```

#main_contents {
    display: flex;
    justify-content: center;
    width: 100%;
    height: 100%;
    margin-top: 100px;
}

#main_contents .wrapper {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
    height: 100%;
    max-width: 600px;
    padding-bottom: 50px;
}

.result {
    display: flex;
    flex-direction: column;
    align-items: center;
}

.result h1 {
    font-size: 50px;
    font-family: "GmarketSansBold";
}

.result_img {
    width: 300px;
    height: 324px;
}

.result_img img {
    width: 100%;
    height: 100%;
}

.share {
    padding-bottom: 20px;
    margin-bottom: 50px;
    border-bottom: 3px solid #ea7e7c;
}

.share button {
    margin-bottom: 30px;
}

.facebook_share {
    background-color: #3b5998;
    color: #fff;
}

```

```
.kakao_share {  
    background-color: #f1d900;  
}  
  
.buttons {  
    padding-bottom: 50px;  
}  
  
.buttons h3 {  
    padding-bottom: 20px;  
}  
  
.buttons button {  
    margin-bottom: 30px;  
}  
  
.weniv {  
    display: flex;  
    flex-direction: column;  
    align-items: center;  
}  
  
.weniv a {  
    width: 100px;  
}  
  
.weniv img{  
    width: 100%;  
    height: auto;  
}  
  
.weniv p {  
    padding-top: 20px;  
    font-size: 10px;  
    text-align: center;  
    line-height: 1.5;  
}
```

5.3 result.js

```

const facebookShare = document.querySelector('.facebook_share');
const kakaoShare = document.querySelector('.kakao_share');
const copyBtn = document.querySelector('.copy_btn');

$(function() {
    let url = window.location.href;
    let img = $('.result_img img').attr('src');
    $("meta[property='og\\:url']").attr("content", url);
    $("meta[property='og\\:image']").attr("content", img);
});

Kakao.init('JavaScript 키를 입력하세요');
// Kakao.isInitialized();

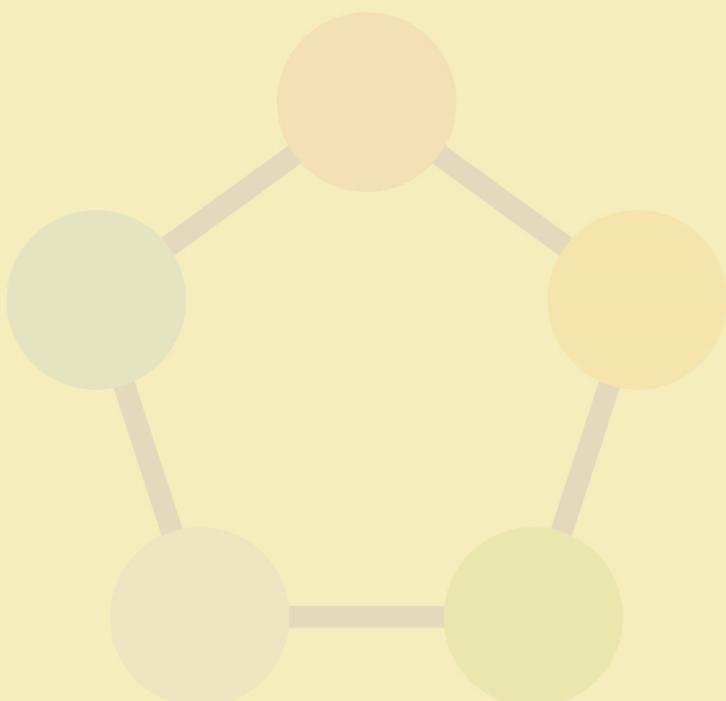
function sendLink() {
    let result_url = window.location.href;
    Kakao.Link.sendDefault({
        objectType: 'feed',
        content: {
            title: '나의 개발 유형은?',
            description: '나에게 꼭 맞는 개발 유형을 알아보자!!',
            imageUrl: 'https://mbit.weniv.co.kr/static/img/mbit_thumbnail.png',
            link: {
                mobileWebUrl: 'https://mbit.weniv.co.kr',
                webUrl: 'https://mbit.weniv.co.kr',
            },
        },
        social: {
            likeCount: 286,
            commentCount: 45,
            sharedCount: 845,
        },
        buttons: [
            {
                title: '결과 보러가기',
                link: {
                    webUrl: result_url,
                    mobileWebUrl: result_url,
                },
            },
            {
                title: '테스트 하러가기',
                link: {
                    webUrl: 'https://mbit.weniv.co.kr',
                    mobileWebUrl: 'https://mbit.weniv.co.kr',
                },
            },
        ],
    });
}

```

```
});  
}  
  
function sharefacebook() {  
    let url = window.location.href;  
    let facebook = 'http://www.facebook.com/sharer/sharer.php?u=';  
    let link = facebook + url;  
    window.open(link);  
}  
  
function copyUrl() {  
    let url = window.location.href;  
    let tmp = document.createElement('input');  
  
    document.body.appendChild(tmp);  
    tmp.value = url;  
    tmp.select();  
    document.execCommand("copy");  
    document.body.removeChild(tmp);  
  
    alert("URL이 복사되었습니다");  
}  
  
facebookShare.addEventListener('click', sharefacebook);  
kakaoShare.addEventListener('click', sendLink);  
copyBtn.addEventListener('click', copyUrl);
```

Chapter 2-2

백엔드 (Back-end)





001. 서버 환경 구축(by 구름IDE)

1. 클라우드 서비스란?

2. 구름 IDE

 2.1. 컨테이너 생성하기

 2.2. 컨테이너 살펴보기

 2.3. 간단히 사용해보기

 2.4. 프로젝트 URL

1. 클라우드 서비스란?

서비스를 운영하기 위해서는 서버가 필요합니다. (여기서 서버는 네트워크에서 정보를 제공하기 위한 컴퓨터를 의미합니다.) 그러나 서버를 직접 사기에는 너무 고가이고, 또 배송이 온다고 해서 끌이 아니라 설치를 하는 것도 매우 큰 리소스가 드는 작업입니다. 우리는 그래서 이 서버를 빌려서 사용합니다. 빌리는 것을 클라우드 서비스라 하고, 이 서비스는 크게 3가지로 나눌 수 있습니다.

1. **SaaS(Software as a Service)**

SaaS는 가장 사용자 단에 친밀한 서비스이며 네트워크를 통해 애플리케이션 기능을 이용할 수 있는 서비스입니다.

2. **PaaS(Platform as a Service)**

PaaS는 윈도우와 리눅스 같은 운영체제를 제공하고 개발 가능한 플랫폼도 함께 제공되는 클라우드 서비스입니다.

3. **IaaS(Infrastructure as a Service)**

IaaS는 인프라를 제공하는 클라우드 서비스입니다. 기업에서 특히 많이 쓰입니다.

2. 구름 IDE

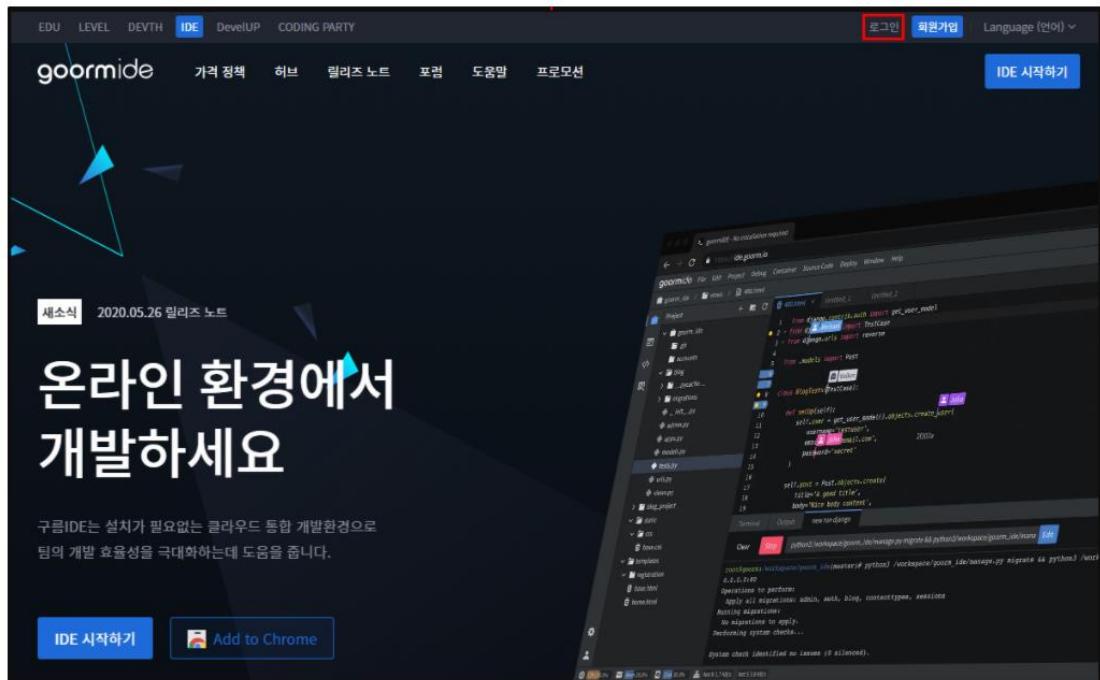
구름 IDE는 설치가 필요없는 클라우드 통합 개발환경(IDE)입니다.

우리가 구름을 사용하는 이유는 3가지가 있습니다.

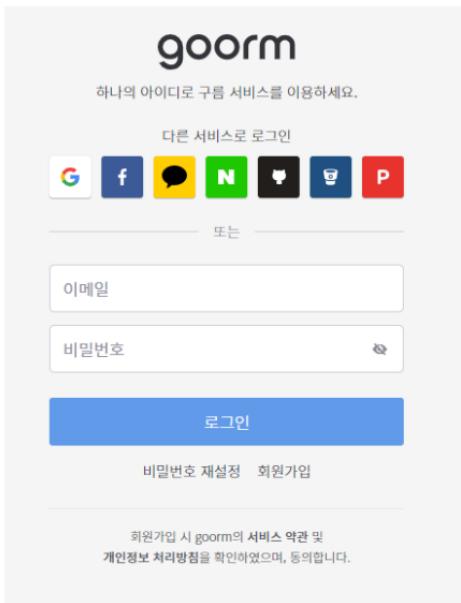
- 보통 개발을 할 때 OS Version, Python Version, Django Version 별 에러를 잡는데 시간이 많이 소요됩니다. 또한 '개발은 배포 환경과 동일하게!'라는 말을 잊지 마세요. 우리의 리소스를 아껴줄 것입니다.
- 집에 서버와 환경을 구축하는 것은 비용(시간 비용, 금전적 비용, 기회 비용)이 상대적으로 큅니다.
- 배포를 명령어 한 번으로 진행할 수 있어요! 또는 구름에서도 서비스를 런칭할 수도 있답니다.

2.1. 컨테이너 생성하기

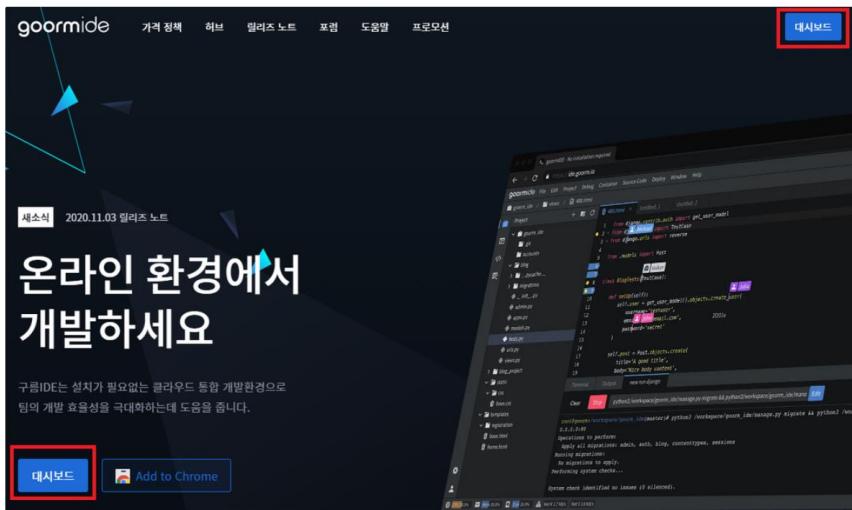
- 구름IDE 메인페이지로 가서 로그인을 합니다.



2. 회원가입을 해야 합니다. 무료계정도 컨테이너 5개를 사용할 수 있는 강력한 서비스입니다.



3. 회원가입을 하면 자동적으로 대시보드에 들어갑니다. 만약 메인화면으로 나오셨다면 대시보드를 눌러주세요.



4. 다음은 대시보드 화면입니다. 새 컨테이너 생성을 누르셔서 컨테이너를 생성해 주세요. 하나의 컨테이너는 하나의 컴퓨터를 세팅하는 것과 같습니다. Python을 선택하신 다음 생성하기를 눌러 주세요.

Containers

새 컨테이너 생성 (0/5)

돌아가기 컨테이너 생성 생성 (Ctrl + M)

이름 *

알파벳, 숫자, _ 만 포함해야합니다. 0/20

설명

컨테이너 설명을 입력해주세요. 0/100

지역

오리건 (미국) 서울 (한국) 프랑크푸르트 (독일) 뮤바이 (인도)

공개 범위

Public Private

Public으로 설정 시 컨테이너 결과에 공개되어 누구나 이 컨테이너에 접속할 수 있습니다.
민감한 정보(서버 비밀번호, 개인 정보,...)를 다룰 경우 노출될 수 있음을 주의바랍니다.

템플릿

Template ZIP / TAR Github Bitbucket Git / SVN Kakao Open Now

배포

Not used Heroku AWS Elastic Beanstalk

소프트웨어 스택

C/C++	HTML/CSS/JS	Python	Django	Flask	PyTorch	Jupyter	Tensorflow
Caffe	Qt	Java	Maven	Gradle	Spring	Spring Boot	JSP
React	React Native	Vue	Node.js	Express	Polymer	Ruby	Rails
PHP	Go	Swift	Arduino	CF	.NET	R	Scala
Kotlin	Hadoop	Spark	Blank				

Template: Python 프로젝트

OS: Ubuntu 18.04 LTS

문의

5. 컨테이너가 만들어지고 있다는 화면이 나오고 곧 컨테이너 생성이 완료되었다고 뜹니다. 컨테이너 실행을 누르지 마시고 대시보드로 이동해주세요.

곧 멋진 작업공간이 생성됩니다.
잠시만 기다려주세요.



컨테이너를 준비중입니다. [1/3]

6. 대시보드로 이동한 다음 해당 컨테이너의 실행 버튼을 눌러주세요.



컨테이너가 성공적으로 생성되었습니다.

컨테이너 실행 대시보드 이동

container

Python | 10GB

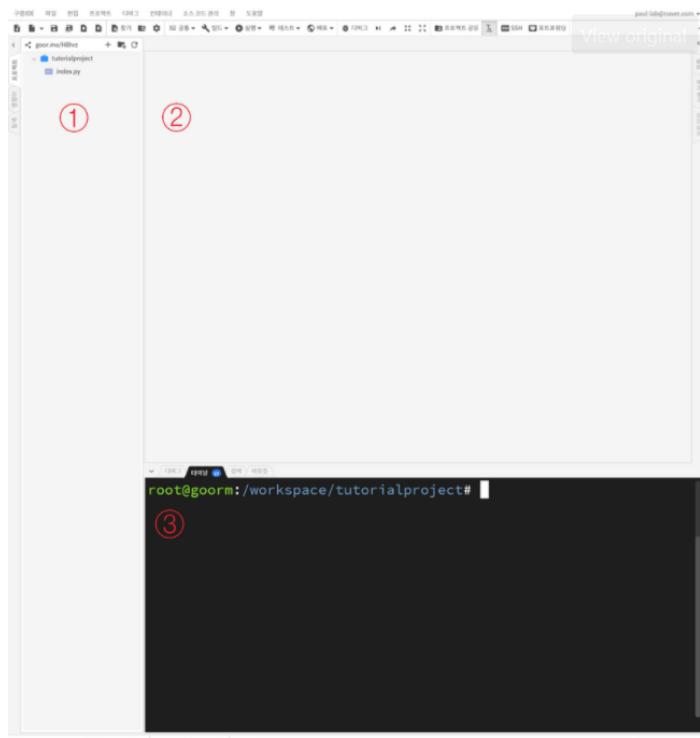
컨테이너

▶ 실행

This screenshot shows a user interface for managing containers. At the top, it displays the container name 'container' and its configuration: 'Python' and '10GB'. Below this, there's a section labeled '컨테이너' (Container). At the bottom, there are three main buttons: a play button with a '+' sign, a stop button with a square, and a large blue '▶ 실행' (Run) button. There are also small icons for settings and other actions.

- 로딩이 완료되었습니다. 여러분만의 클라우드 컴퓨터입니다!

2.2. 컨테이너 살펴보기



컨테이너 실행화면

- 폴더 구조를 볼 수 있는 공간입니다. 우측 상단에 보시면 새로고침 버튼이 있어요. 콘솔창에서 뭔가 작업을 했는데 보이지 않는다면 새로고침을 해보시기 바랍니다.
- txt, html, py파일 등 다양한 파일을 edit 할 수 있는 공간입니다.
- 콘솔창입니다. 우리는 대부분의 명령어를 이곳에 입력하게 됩니다.



오른쪽에는 협업을 위한 공간이 하나 열렸을 텐데, 우리는 사용하지 않을 것이니 접어둡시다! 이 공간은 채팅 등 협업을 위한 다양한 기능을 제공합니다.

2.3. 간단히 사용해보기

자, 이제 간단한 실습을 해보도록 합시다.

- 원쪽 프로젝트 바에서 `index.py` 를 클릭합니다. 그러면 오른쪽 창에 해당 파일의 소스코드가 보입니다. 안에 있는 내용을 다음과 같이 수정해보세요.

```
print('Welcome to jejucodingbasecamp!! :))')
```

- 안에 있는 내용을 수정하면 해당 탭에 `*` 표시가 생깁니다. 해당 표시는 저장이 안되었다는 표시입니다. `Ctrl + S` 를 눌러서 해당 파일의 변경사항을 저장합니다.

- 저장이 된 다음 아래 콘솔창에서 `python index.py` 를 입력해보세요. 작성하신 파이썬 파일이 실행됩니다!

```
python index.py
```

```
index.py (tutorialproject) - 구문 | +
```

구름IDE 파일 편집 프로젝트 디버그 컨테이너 소스 코드 관리 창 도움말 paul-lab@naver.com

goor.me/kuTj tutorialproject index.py

```
1 | print('Welcome to jejucodingcamp!! :))')
```

디버그 터미널 김해 미온듯

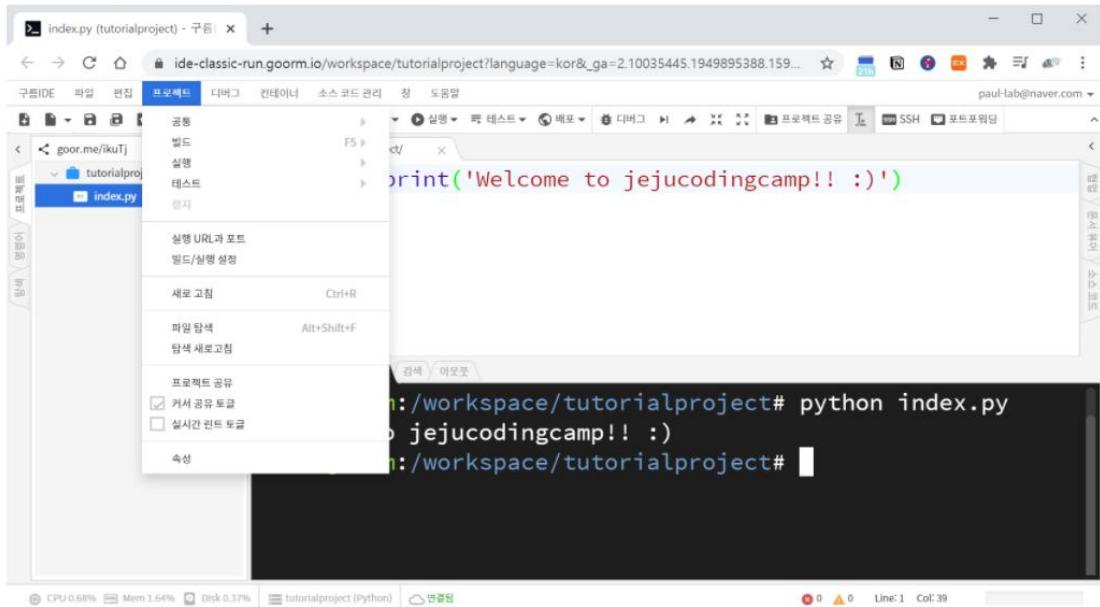
```
root@goorm:/workspace/tutorialproject# python index.py
Welcome to jejucodingcamp!! :)
root@goorm:/workspace/tutorialproject#
```

CPU 0.05% Mem 1.64% Disk 0.37% tutorialproject [Python] 연경설 0 0 Line:1 Col:39

실행 화면

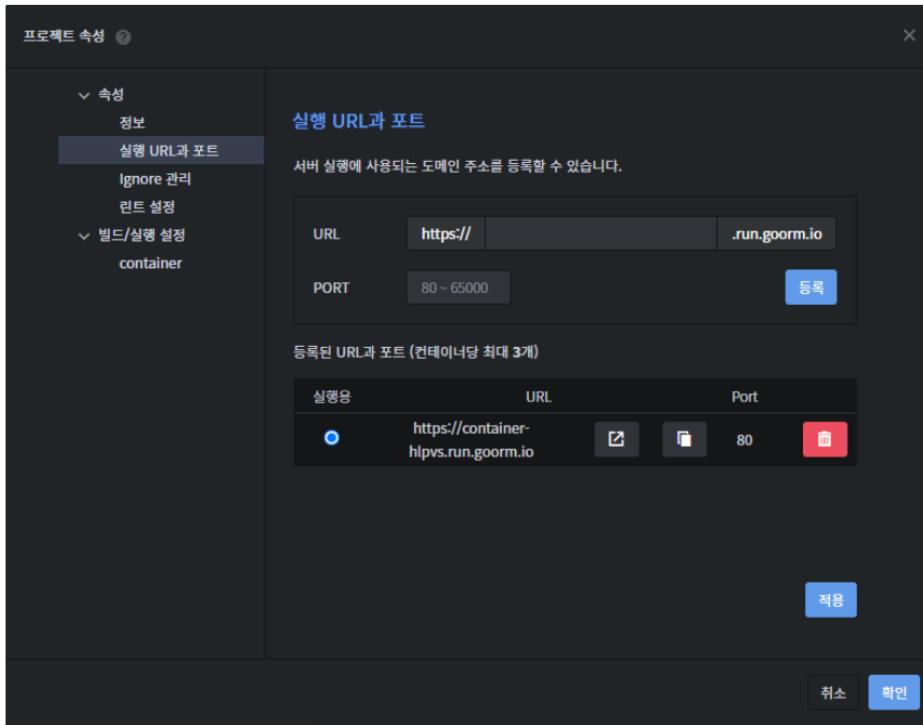
2.4. 프로젝트 URL

- 메뉴에서 **프로젝트** > **실행 URL과 포트** 를 눌러보세요.



💡 프로젝트를 클릭한 화면인 위 화면에서 **실시간 릴트 토글** 체크박스 버튼이 빠져있죠? 이렇게 빼주시면 Error를 잡지 않습니다. 구름IDE에서 아직은 제대로 Error를 잡지 못하니 이걸 빼줍시다!

2. 앞으로 프로젝트에서 사용하게 될 URL을 볼 수 있습니다. 만약 등록이 되어 있지 않다면 위에서 원하는 url을 입력하고 PORT는 80으로 하여 등록버튼을 눌러주세요.





002. 장고 프로젝트 생성하기

1. 가상환경 생성

1.1. 파이썬 설치 확인

1.2. venv 모듈을 통해 가상 환경 생성

1.3. 가상환경 실행

2. django 프로젝트 생성

2.1. django 패키지 설치

2.2. django 프로젝트 생성

2.3. django 프로젝트 실행

2.4. 앱 생성

1. 가상환경 생성



가상환경을 왜 만들어야 할까요?

예를 들어 A 프로젝트에서는 python 2 버전이 필요하고, B 프로젝트에서 python 3 버전이 필요하다면 버전이 충돌하는 문제가 발생합니다.

가상환경을 통해 독립적인 환경을 만들어 주고, 여기서 python뿐만 아니라 다른 다양한 패키지를 설치하여 각 프로젝트들을 관리할 수 있게 됩니다.

1.1. 파이썬 설치 확인

구름IDE에서 컨테이너 생성 시 '소프트웨어 스택'을 `python`으로 선택해서 생성했다면, 기본적으로 python 3.x 버전이 설치가 되어 있습니다. 터미널에 `python -V` 또는 `python --version` 명령어를 통해 python이 설치되었는지와 동시에 버전을 확인 할 수 있습니다.

```
root@goorm:/workspace/MBIT# python -V
Python 3.7.4
```

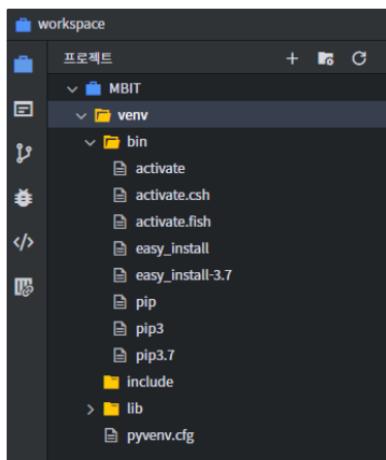
1.2. venv 모듈을 통해 가상 환경 생성

python 3 버전에서는 python 가상환경을 생성할 수 있는 `venv` 모듈이 기본으로 내장되어 있습니다.

`python -m venv [가상환경이름]` 명령어를 통해 가상 환경을 생성합니다.

```
root@goorm:/workspace/MBIT# python -m venv venv
```

가상환경을 생성하면 다음 사진과 같이 해당 이름의 가상환경 폴더가 생성됩니다.



1.3. 가상환경 실행

다음 명령어를 통해 가상환경을 실행합니다.

```
$ source venv/bin/activate
```

그러면 명령줄 앞에 가상환경의 이름이 붙게 됩니다.

```
root@goorm:/workspace/MBIT# source venv/bin/activate
(venv) root@goorm:/workspace/MBIT#
```

가상환경이 잘 실행되고 있는지 확인하려면 `pip list` 명령어를 사용할 수 있습니다. `pip list` 명령어는 현재 실행되고 있는 환경에 설치된 python 모듈들의 리스트를 보여주는데, 다음 사진에서처럼 글로벌 환경일 때와 가상 환경일 때 `pip list`의 결과가 다른 것을 확인할 수 있습니다.

```
root@goorm:/workspace/MBIT# pip list
Package           Version
-----
absl-py          0.11.0
argon2-cffi      20.1.0
astroid          2.4.2
astunparse       1.6.3
async-generator   1.10
attrs            20.2.0
backcall         0.2.0
bleach           3.2.1
bokeh             2.2.3
cached-property  1.5.2
cachetools       4.1.1
certifi          2020.6.20
cffi              1.14.3
chardet          3.0.4
click             7.1.2
cycler            0.10.0
decorator        4.4.2
defusedxml       0.6.0
Django           2.2.4
entrypoints      0.3
flake8          3.8.4
Flask            1.1.2
gast              0.3.3
google-auth      1.23.0
google-auth-oauthlib 0.4.2
google-pasta     0.2.0
graphviz         0.8.4
```

글로벌 환경

```
(venv) root@goorm:/workspace/MBIT# pip list
Package           Version
-----
pip               19.0.3
setuptools       40.8.0
```

가상환경



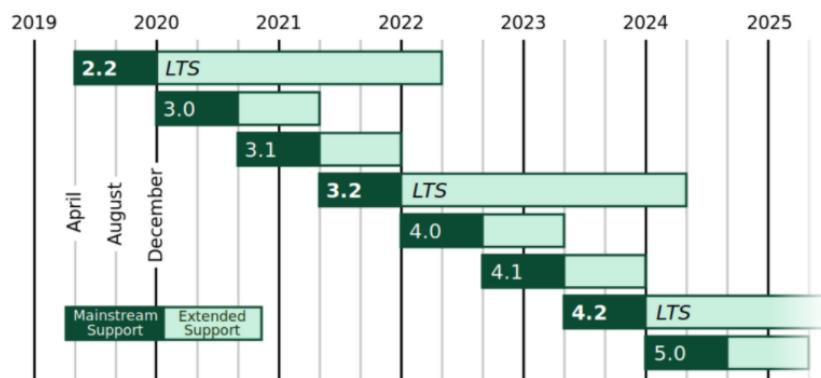
`pip` 사용 중 다음과 같은 경고 메시지가 출력될 수도 있는데, 해당 메시지는 python 패키지 관리를 하는 `pip`의 버전에 대한 경고 메시지입니다. 해당 메시지는 무시해도 무방합니다. 말 그대로 에러가 아닌 경고입니다.

You are using pip version 19.0.3, however version 20.3.3 is available.

You should consider upgrading via the 'pip install --upgrade pip' command.

2. django 프로젝트 생성

Django 업데이트 로드맵입니다. 현재 Django는 3.x를 쓰는 것이 일반적이며, 저희 책에서는 3.1 Version을 사용하여 기술하도록 하겠습니다.



Release Series	Latest Release	End of mainstream support ¹	End of extended support ²
3.0	3.0.8	August, 2020	April, 2021
2.2 LTS	2.2.14	December 2, 2019	April 2022
2.1	2.1.15	April 1, 2019	December 2, 2019
2.0	2.0.13	August 1, 2018	April 1, 2019
1.11 LTS ³	1.11.29	December 2, 2017	April 1, 2020
1.10	1.10.8	April 4, 2017	December 2, 2017
1.9	1.9.13	August 1, 2016	April 4, 2017
1.8 LTS	1.8.19	December 1, 2015	April 1, 2018
1.7	1.7.11	April 1, 2015	December 1, 2015
1.6	1.6.11	September 2, 2014	April 1, 2015
1.5	1.5.12	November 6, 2013	September 2, 2014
1.4 LTS	1.4.22	February 26, 2013	October 1, 2015
1.3	1.3.7	March 23, 2012	February 26, 2013

Release Series	Release Date	End of mainstream support ¹	End of extended support ²
3.1	August 2020	April 2021	December 2021
3.2 LTS	April 2021	December 2021	April 2024
4.0	December 2021	August 2022	April 2023
4.1	August 2022	April 2023	December 2023
4.2 LTS	April 2023	December 2023	April 2026

<https://www.djangoproject.com/download/>

2.1. django 패키지 설치

`pip install` 명령어를 통해 현재 환경에 Django 패키지를 설치합니다.

```
$ pip install django
```

```
(venv) root@goorm:/workspace/MBIT# pip install django
Collecting django
  Downloading https://files.pythonhosted.org/packages/08/c7/7ce40e5a5cb47ede081b9f
a8a3dd93d101c884882ae34927967b0792f5fb/Django-3.1.4-py3-none-any.whl (7.8MB)
    100% |██████████| 7.8MB 4.7MB/s
Collecting sqlparse>=0.2.2 (from django)
  Using cached https://files.pythonhosted.org/packages/14/05/6e8eb62ca685b10e34051
a80d7ea94b7137369d8c0be5c3b9d9b6e3f5dae/sqlparse-0.4.1-py3-none-any.whl
Collecting pytz (from django)
  Downloading https://files.pythonhosted.org/packages/89/06/2c2d3034b4d6bf22f2a4ae
546d16925898658a33b4400cfb7e2cle2871a3/pytz-2020.5-py2.py3-none-any.whl (510kB)
    100% |██████████| 512kB 3.6MB/s
Collecting asgiref<4,>=3.2.10 (from django)
  Downloading https://files.pythonhosted.org/packages/89/49/5531992efc62f9c6d08a71
99dc31176c8c60f7b2548c6ef245f96f29d0d9/asgiref-3.3.1-py3-none-any.whl
Installing collected packages: sqlparse, pytz, asgiref, django
Successfully installed asgiref-3.3.1 django-3.1.4 pytz-2020.5 sqlparse-0.4.1
```

사진처럼 마지막에 `Successfully installed` 메시지가 뜨면 성공적으로 설치된 것입니다.



`pip install django` 명령어는 django의 최신 버전이 설치됩니다.

만약 특정 버전을 설치하고 싶은 경우에 `pip install django==[버전]` 명령어를 사용합니다.

ex) `pip install django==3.1`

이제 `pip list` 명령어를 통해 현재 환경에 django가 설치된 것을 확인할 수 있습니다.

```
(venv) root@goorm:/workspace/MBIT# pip list
Package      Version
-----
asgiref     3.3.1
Django       3.1.4
pip          19.0.3
pytz         2020.5
setuptools   40.8.0
sqlparse     0.4.1
```

django 외에도 추가적으로 설치되는 패키지들이 있는데, 이들은 django에 필요한 패키지들이 자동으로 설치된 것입니다.

2.2. django 프로젝트 생성

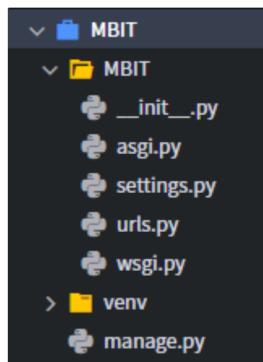
django를 설치하면 이제 `django-admin` 명령어를 사용할 수 있게 됩니다.

가장 대표적으로 `django-admin startproject [프로젝트 명] [경로]` 으로, 해당 `[프로젝트 명]` 으로 해당 `[경로]` 에 프로젝트를 생성하는 명령어입니다.

다음 명령어를 통해 `MBIT` 라는 이름으로 현재 경로(`.`)에 프로젝트를 생성합니다. 명령어 마지막에 점(.)을 꼭 넣어주셔야 합니다!

```
$ django-admin startproject MBIT .
```

그러면 현재 경로에 프로젝트 메인 폴더와 `manage.py` 파일이 생성됩니다.

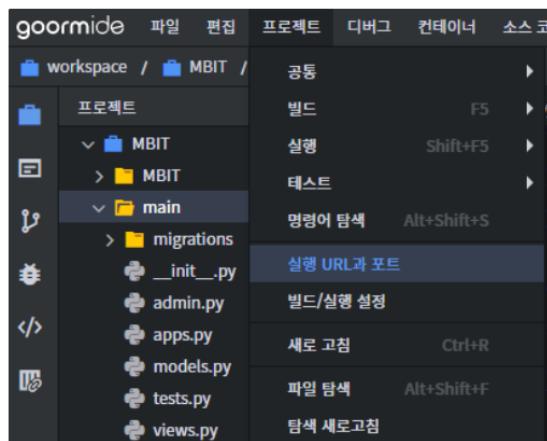


2.3. django 프로젝트 실행

프로젝트가 정상적으로 잘 생성되었는지 확인하기 위해 다음 명령어를 통해 프로젝트를 실행시킵니다.

```
$ python manage.py runserver 0:80
```

프로젝트 웹 페이지로 접속하려면 `프로젝트 > 실행 URL과 포트` 에 등록된 URL로 접속합니다.





접속하면 다음의 오류 메시지의 페이지를 볼 수 있는데, 이는 django 프로젝트가 잘 실행되고 있지만 현재 호스트의 접속을 허가하고 있지 않다는 뜻입니다.

DisallowedHost at /

Invalid HTTP_HOST header: 'mbit-pugwe.run.goorm.io'. You may need to add 'mbit-pugwe.run.goorm.io' to ALLOWED_HOSTS.

```

Request Method: GET
Request URL: http://mbit-pugwe.run.goorm.io/
Django Version: 3.1.4
Exception Type: DisallowedHost
Exception Value: Invalid HTTP_HOST header: 'mbit-pugwe.run.goorm.io'. You may need to add 'mbit-pugwe.run.goorm.io' to ALLOWED_HOSTS.
Exception Location: /workspace/MBIT/venv/lib/python3.7/site-packages/django/http/request.py, line 137, in get_host
Python Executable: /workspace/MBIT/venv/bin/python3
Python Version: 3.7.4
Python Path: ['/workspace/MBIT',
             '/goormservice/cafe/python',
             '/workspace/MBIT',
             '/usr/local/lib/python3.7.zip',
             '/usr/local/lib/python3.7',
             '/usr/local/lib/python3.7/lib-dynload',
             '/workspace/MBIT/venv/lib/python3.7/site-packages']
Server time: Sun, 03 Jan 2021 02:26:51 +0000
  
```

모든 호스트에서 프로젝트 URL에 접근을 허용할 수 있도록 [프로젝트 명]/`settings.py` 에서 `ALLOWED_HOSTS` 를 다음과 같이 수정합니다.

```
ALLOWED_HOSTS = ['*']
```

이제 다시 프로젝트 URL로 접속하면 장고의 메인화면인 로켓을 볼 수 있습니다.

[django](#)

[View release notes for Django 3.1](#)



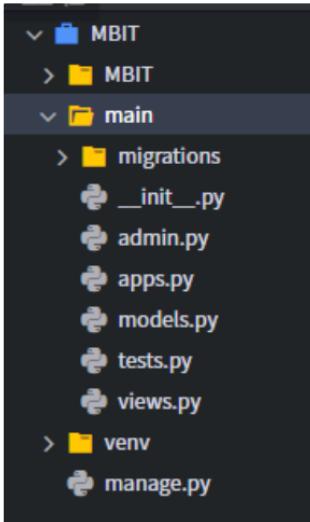
2.4. 앱 생성

이제 해당 프로젝트에서 내가 원하는 기능을 수행하는 앱을 만들어야 합니다. 장고 프로젝트에서 앱을 만드는 명령어는 `python manage.py startapp [앱이름]` 입니다.

다음 명령어를 통해 `main` 이라는 이름의 앱을 생성합니다.

```
$ python manage.py startapp main
```

그러면 해당 이름으로 앱 폴더가 생성됩니다.



앱을 생성한 뒤에는 반드시 해당 앱을 프로젝트에 등록을 해주어야 합니다. 그렇지 않으면 프로젝트는 해당 앱의 존재를 인식하지 못하고 사용할 수 없습니다.

[프로젝트 명]/`settings.py`에서 `INSTALLED_APPS` 리스트에 해당 앱 이름을 추가해줍니다.

```
INSTALLED_APPS = [
    '앱이름',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]
```

 장고에서는 리스트, 튜플, 딕셔너리 등에서 마지막 요소에도 `,`를 붙이는 것을 권장합니다. 붙이지 않아도 에러가 나지는 않으나, 이는 추후에 요소를 뒤에 추가할 때 종종 `,`를 빠트려서 나는 에러를 방지하기 위함입니다.



003. 모델 작성하기

1. 모델(model) 이란?

2. 모델 작성

2.1. Developer : 개발자 유형

2.2. Question (문항)

2.3. Choice : 선택지

3. 마이그레이션

3.1. makemigrations

3.2. migrate

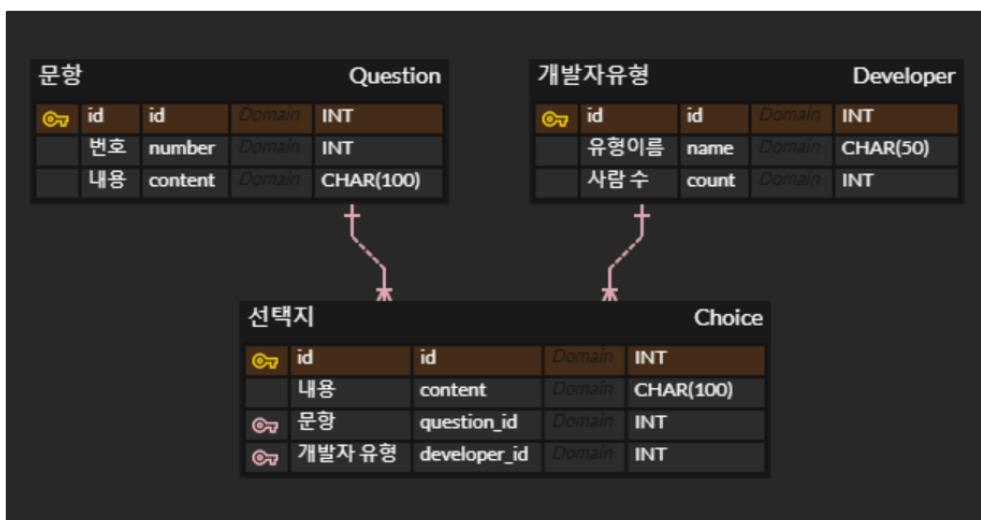
1. 모델(model) 이란?

Django에서 모델은 데이터베이스에 저장될 테이블을 정의하는 클래스입니다. django에서는 SQL을 직접 작성하지 않아도 모델을 저장하면 DB에 저장됩니다. (여기서 SQL이란 관계형 데이터베이스 관리 시스템으로, 데이터를 관리하기 위해 설계된 특수목적의 프로그래밍 언어입니다.)

그렇다면 데이터베이스에는 어떤것이 들어있을까요? 사용자에 대한 ID, PW 등 개인 정보나, 우리가 만들 테스트지의 문항, 그 결과값 등 다양한 데이터들이 저장되어 있습니다.

2. 모델 작성

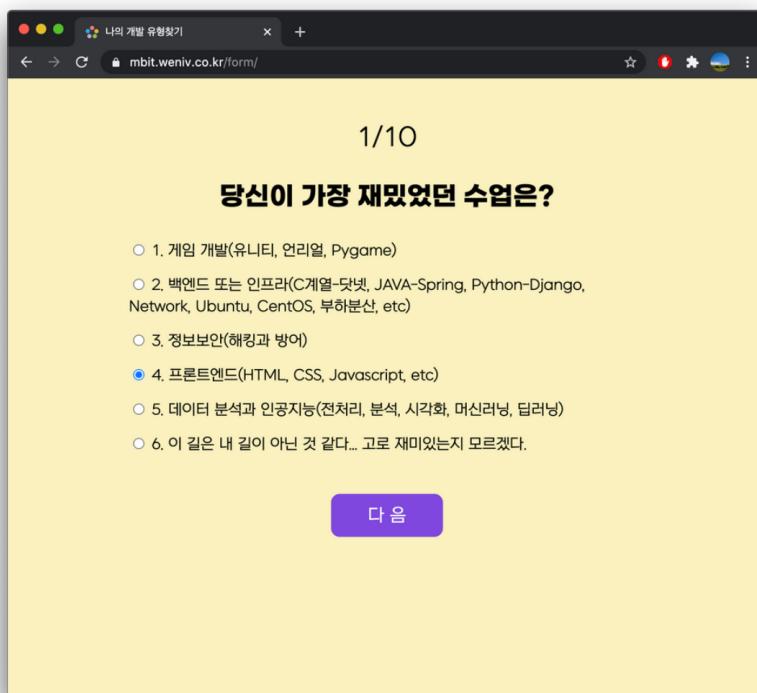
우리가 작성하고자 하는 모델의 설계는 다음과 같습니다.



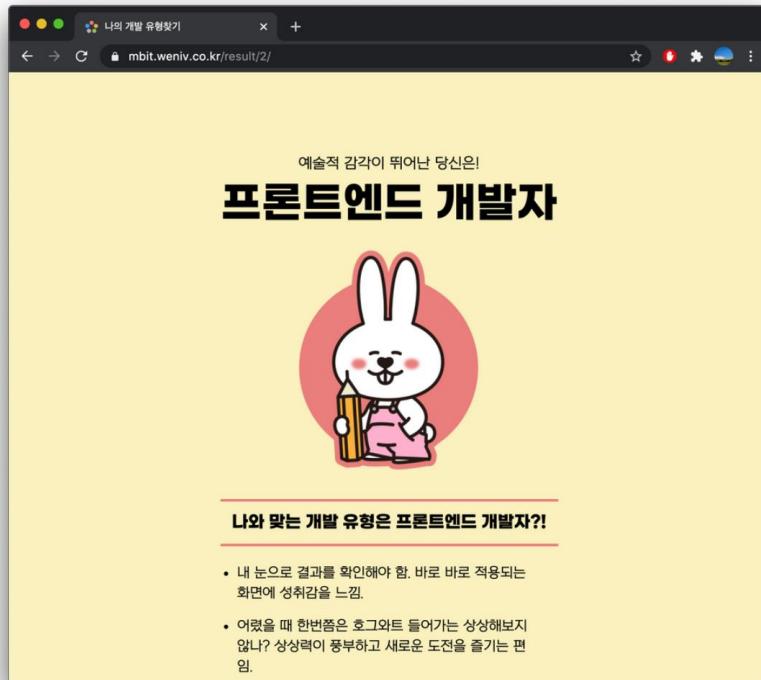
모델 클래스는 [해당 앱 폴더]/`models.py`에 작성합니다.

본격적으로 모델링을 하기 전, 우리가 어떤 서비스를 만들지 다시 한번 생각해 봅시다.

우리는 테스트지를 만들 것이고, 테스트지의 각 항목에 점수를 매겨 점수가 가장 높은 항목을 결과값으로 보여줄 것입니다.



질문지



결과페이지

2.1. Developer : 개발자 유형

개발자 유형은 결과값으로 보여 줄 모델입니다.

개발자 유형에는 두 가지 변수가 들어갑니다. `name` 은 개발자 유형(프론트엔드, 백엔드 등)을 의미하며 `count` 는 테스트 결과에 대한 카운트 값을 의미합니다. (`count` 값은 메인 페이지에서 보여지는 결과값이에요!)

```
class Developer(models.Model):
    name = models.CharField(max_length=50)
    count = models.IntegerField(default=0)
```

2.2. Question (문항)

문항은 테스트지의 질문 목록들을 의미합니다.

`number` 는 문항 번호를 의미하며, `content` 는 질문 내용을 뜻합니다.

문항 번호는 겹치면 안 되기 때문에 `unique=True` 옵션을 줍니다.

```
class Question(models.Model):
    number = models.IntegerField(unique=True)
    content = models.CharField(max_length=100)
```

2.3. Choice : 선택지

`content` 는 해당 선택지의 내용을, `question` 은 해당 선택지가 속한 문항을, `developer` 는 해당 선택지가 어느 개발 유형의 선택지인지를 나타냅니다.

각 `question` 과 `developer` 는 여러 개의 선택지를 가질 수 있지만, 각 선택지는 하나의 `question` 과 하나의 `developer` 만 가질 수 있습니다. 이러한 관계를 1:N 이라고 표현합니다. 그리고 이러한 경우 `models.ForeignKey` 필드를 사용합니다. (DB에서 외래키를 뜻합니다.)

선택지는 속해있는 문항이 없어지거나, 나타내는 개발자 유형이 삭제되면 같이 삭제되어야 된다는 의미로 `on_delete=models.CASCADE` 옵션을 줍니다.

그리고 선택지 중에서는 어느 개발 유형에도 속하지 않는 경우도 있기 때문에 `developer` 에는 `null` 값이 저장될 수 있도록 `null=True` 옵션도 추가합니다.

```
class Choice(models.Model):
    content = models.CharField(max_length=100)
    question = models.ForeignKey(to='main.Question', on_delete=models.CASCADE)
    developer = models.ForeignKey(to='main.developer', on_delete=models.CASCADE, null=True)
```

3. 마이그레이션

마이그레이션(migration)이란 모델에서 추가되거나 수정된 내용들을 DB에 옮기는 작업을 의미합니다. Django에서는 SQL 없이 모델에 작성하면 DB에 저장된다고 했었죠? 이 과정을 수행해주는 명령어입니다.

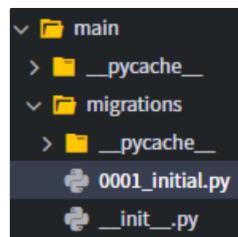
3.1. makemigrations

일단 `makemigrations` 명령어를 통해 DB의 설계도를 작성합니다. 이 과정은 migration을 준비하는 과정이며, `migrate`를 하기 전 반드시 수행해주세요!

```
$ python manage.py makemigrations
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py makemigrations
Migrations for 'main':
  main/migrations/0001_initial.py
    - Create model Developer
    - Create model Question
    - Create model Choice
```

사진처럼 나온다면 `makemigration` 명령이 정상적으로 수행된 것이고 `main/migrations/` 경로에 `001_initial.py`라는 설계도 파일이 만들어졌을 겁니다.



3.2. migrate

이번에는 해당 설계도를 통해 DB에 테이블을 만드는 명령어 `migrate`를 사용합니다.

```
$ python manage.py migrate
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying contenttypes.0001_initial... OK
  Applying auth.0001_initial... OK
  Applying admin.0001_initial... OK
  Applying admin.0002_logentry_remove_auto_add... OK
  Applying admin.0003_logentry_add_action_flag_choices... OK
  Applying contenttypes.0002_remove_content_type_name... OK
  Applying auth.0002_alter_permission_name_max_length... OK
  Applying auth.0003_alter_user_email_max_length... OK
  Applying auth.0004_alter_user_username_opts... OK
  Applying auth.0005_alter_user_last_login_null... OK
  Applying auth.0006_require_contenttypes_0002... OK
  Applying auth.0007_alter_validators_add_error_messages... OK
  Applying auth.0008_alter_user_username_max_length... OK
  Applying auth.0009_alter_user_last_name_max_length... OK
  Applying auth.0010_alter_group_name_max_length... OK
  Applying auth.0011_update_proxy_permissions... OK
  Applying auth.0012_alter_user_first_name_max_length... OK
  Applying main.0001_initial... OK
  Applying sessions.0001_initial... OK
```

`migrate`가 성공하면 위 사진처럼 나올 것입니다.

결과 메시지를 보면, 우리가 만든 `main` 앱 뿐만 아니라 다른 앱들 `admin`, `auth` 등이 보이는데 장고 프로젝트를 처음 생성하면 기본적으로 내장되는 앱들입니다.

처음 `migrate` 시에는 이러한 기본 앱들이 가지고 있는 모델들도 `migrate` 되는 것입니다.



004. 관리자 페이지

1. 관리자 페이지란?
2. 관리자 계정 만들기
3. 관리자 페이지에서 모델 요소 추가하기
 - 3.1. 모델 등록하기
 - 3.2. 모델 요소 추가하기
 - 3.2.1 Developer(개발자 유형) 추가하기
 - 3.2.2. Question(문항) 추가하기
 - 3.2.3. Choice(선택지) 추가하기
 - 3.3. 모델 요소 표현방식 정의하기

1. 관리자 페이지란?

관리자 페이지는 해당 프로젝트를 개발자들이 쉽게 관리할 수 있도록 GUI를 활용한 페이지입니다. 원래는 개발자가 직접 만들어야 하지만 장고는 프로젝트를 생성하면 기본적으로 관리자 페이지 앱을 만들어주는데 그것이 바로 `admin` 앱입니다.

관리자 페이지에서 우리가 만든 모델들의 요소를 쉽게 추가할 수 있습니다.

2. 관리자 계정 만들기

관리자 페이지로 들어가려면 `프로젝트URL/admin/` 으로 접속하면 됩니다. 그러면 관리자 로그인 화면이 뜰 것입니다.

하지만 우리는 아직 관리자 계정을 만들지 않았습니다. 관리자 계정을 만들려면 `createsuperuser` 명령어를 사용하면 됩니다.

```
$ python manage.py createsuperuser
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py createsuperuser
Username (leave blank to use 'root'):
Email address:
Password:
Password (again):
```

그러면 그림처럼 관리자 계정의 정보를 입력하는 프롬프트가 나타납니다. 필요한 정보를 입력하고 `Superuser created successfully` 메시지가 뜨면 관리자 계정을 만드는 것에 성공한 것입니다.
(비밀번호는 `username`이나 이메일과 같이 화면에 나타나지 않으니, 감으로 쳐주셔야 합니다.)

3. 관리자 페이지에서 모델 요소 추가하기

3.1. 모델 등록하기

이제 만들어진 관리자 계정으로 관리자 페이지에 로그인하면 해당 화면을 볼 수 있습니다.

The screenshot shows the Django administration interface. At the top, it says "Django administration" and "WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT". Below that is a "Site administration" header. On the left, there's a sidebar titled "AUTHENTICATION AND AUTHORIZATION" with "Groups" and "Users" listed, each with "Add" and "Change" buttons. To the right, there are two sections: "Recent actions" (empty) and "My actions" (also empty). The main content area is currently empty, showing the message "None available".

원래는 여기서 해당 프로젝트의 모델들을 볼 수 있는데, 우리가 만든 모델들이 보이지 않습니다. 그 이유는 admin 앱에 해당 모델들을 등록을 하지 않았기 때문입니다. 구름 IDE로 돌아가서 `main/admin.py` 파일을 다음과 같이 수정합니다.

```
from django.contrib import admin
from .models import Developer, Question, Choice

admin.site.register(Developer)
admin.site.register(Question)
admin.site.register(Choice)
```

그러면 이제 관리자 페이지에서 우리가 만든 모델들을 볼 수 있습니다.

Django administration

WELCOME, ROOT. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

AUTHENTICATION AND AUTHORIZATION

- Groups [+ Add](#) [Change](#)
- Users [+ Add](#) [Change](#)

MAIN

- Choices [+ Add](#) [Change](#)
- Developers [+ Add](#) [Change](#)
- Questions [+ Add](#) [Change](#)

Recent actions

My actions

None available

3.2. 모델 요소 추가하기

3.2.1 Developer(개발자 유형) 추가하기

이제 `Developers` 의 `Add` 버튼을 눌러서 `프론트 엔드`라는 개발 유형을 추가해봅니다.

Add developer

Name:

[Save and add another](#) [Save and continue editing](#) **SAVE**

`name` 필드 값으로 `프론트 엔드`를 추가하고 `SAVE` 버튼을 누릅니다.

The developer "Developer object (1)" was added successfully.

Select developer to change

[ADD DEVELOPER](#)

Action: 0 of 1 selected

DEVELOPER

[Developer object \(1\)](#)

1 developer

그러면 **Developer object (1)** 이 추가된 것을 볼 수 있습니다. 여기서 숫자 **(1)**은 요소의 **id** 값입니다.

같은 방식으로 **백엔드 개발**, **게임 개발**, **데이터 분석과 인공지능**, **정보보안**을 추가해봅니다.

3.2.2. Question(문항) 추가하기

이번에는 **Questions**의 **Add** 버튼을 눌러서 **당신이 가장 재밌었던 수업은?**이라는 문항을 **1** 번 문항으로 만들어봅시다.

Add question

Number:	<input type="text" value="1"/>
Content:	<input type="text" value="당신이 가장 재밌었던 수업은?"/>
<input type="button" value="Save and add another"/> <input type="button" value="Save and continue editing"/> <input type="button" value="SAVE"/>	

 The question "Question object (1)" was added successfully.

Select question to change

ADD QUESTION +

Action: 0 of 1 selected

QUESTION

Question object (1)

1 question

Question object (1)이 만들어졌습니다.

3.2.3. Choice(선택지) 추가하기

이번에는 `Choices` 의 `Add` 버튼을 눌러서 `프론트엔드(HTML, CSS, Javascript, etc)` 라는 선택지를 `1번` 문항에 추가해봅니다. 해당 선택지는 `프론트 엔드` 개발 유형에 해당하므로 연관 개발자 유형을 `프론트 엔드`로 등록합니다.

Add choice

Content: `프론트엔드(HTML, CSS, Javascript, etc)`

Question: `Question object (1)`

Developer: `Developer object (1)`

`Save and add another` `Save and continue editing` `SAVE`

The choice "Choice object (1)" was added successfully.

Select choice to change

`ADD CHOICE +`

Action: `-----` `Go` 0 of 1 selected

CHOICE

`Choice object (1)`

1 choice

3.3. 모델 요소 표현방식 정의하기

각 모델 요소들을 한 번씩 만들어 보았습니다. 하지만 불편한 점이 있습니다. 각 모델 요소들은 `[모델 명] object (id값)`으로 표현된다는 것입니다. 이것만 봐서는 몇 번 문항인지, 어떤 개발자 유형인지, 어떤 선택지인지 알 수가 없습니다.

몇 번째 문항인지, 어떤 개발자 유형인지 등 이러한 항목들을 데이터 구별을 위해 제목으로 표시하고 싶다면 모델 요소들의 표현 방식을 모델 클래스에서 `__str__()` 메소드를 정의함으로써 바꿀 수 있습니다.

`main/models.py`에서 각 모델의 `__str__()` 메소드를 정의합니다.

```
from django.db import models

class Developer(models.Model):
    name = models.CharField(max_length=50)

    def __str__(self):
        return self.name


class Question(models.Model):
    number = models.IntegerField(unique=True)
    content = models.CharField(max_length=100)

    def __str__(self):
        return f'{self.number}. {self.content}'


class Choice(models.Model):
    content = models.CharField(max_length=100)
    question = models.ForeignKey(to='main.Question', on_delete=models.CASCADE)
    developer = models.ForeignKey(to='main.developer', on_delete=models.CASCADE, null=True)

    def __str__(self):
        return self.content
```

그러면 다음 그림처럼 모델 요소들이 사람이 알아 볼 수 있게 표현됩니다.

- | | | |
|---------------------------------------|--|--|
| <input type="checkbox"/> DEVELOPER | <input type="checkbox"/> QUESTION | <input type="checkbox"/> CHOICE |
| <input type="checkbox"/> 정보보안 | <input type="checkbox"/> 1. 당신이 가장 재밌었던 수업은? | <input type="checkbox"/> 프론트엔드(HTML, CSS, Javascript, etc) |
| <input type="checkbox"/> 데이터 분석과 인공지능 | | |
| <input type="checkbox"/> 게임 개발 | | |
| <input type="checkbox"/> 백엔드 개발 | | |
| <input type="checkbox"/> 프론트 엔드 | | |



005. 페이지 작성하기

1. 메인 페이지

1.1. URL 설정

URL 라우팅

1.2. 뷰 함수 정의

1.3. 템플릿 정의

1.3.1 장고 템플릿 언어

2. 설문 페이지

2.1 URL 설정

2.2 뷰 함수 정의

2.3 템플릿 정의

3. 결과 페이지

3.1. URL 설정

3.2. 뷰 함수 정의

3.3. 템플릿 정의

4. Django 개발 루틴

1. 메인 페이지

1.1. URL 설정

URL 라우팅

URL 라우팅이란 사용자의 요청 주소, 즉 URL에 따라 어떤 앱의 어떤 페이지를 보여줄지를 설정하는 것을 말합니다. 예를 들어 [프로젝트 URL]/admin/ 주소로 요청이 오면 admin 앱의 페이지를 보여주는 식입니다.

이러한 설정이 MBIT/urls.py에 정의되어 있습니다.

```
from django.contrib import admin
from django.urls import path

urlpatterns = [
    path('admin/', admin.site.urls),
]
```

MBIT/urls.py

[프로젝트 URL]/로 요청이 오면 우리가 만들 메인 페이지가 보이도록 해봅시다.

urls.py를 다음과 같이 수정합니다.

```
from django.contrib import admin
from django.urls import path
from main import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index)
]
```

여기서 views.index는 main/views.py에 작성할 함수입니다. 이 설정은 [프로젝트 경로]로 요청이 오면 main/views.py에 있는 index라는 함수를 실행하겠다는 의미입니다.

1.2. 뷰 함수 정의

`main/views.py`에 `index` 함수를 다음과 같이 정의합니다.

```
from .models import Question, Developer, Choice

def index(request):
    developers = Developer.objects.all()

    context = {
        'developers': developers,
    }

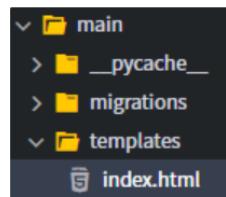
    return render(request, 'index.html', context=context)
```

- `index` 함수는 `return` 값으로 '`index.html`'을 `render` 해주며 이 코드는 `index.html`을 페이지로 보여주겠다는 뜻입니다.
- `Developer.objects.all()`은 모든 `Developer`의 요소들을 리스트같은 형태로 반환합니다.

 정확히는 `QuerySet`이라는 자료구조지만 `List`처럼 활용이 가능합니다.
- `context` 딕셔너리에 담은 요소들은 `index.html`에서 변수처럼 사용할 수 있습니다. 자세한 것은 다음 3) 템플릿 정의에서 설명합니다.

1.3. 템플릿 정의

장고에서 html 파일들을 별개로 **장고 템플릿**이라고 부릅니다. 장고는 기본적으로 템플릿 파일들을 각 앱 폴더 안의 `templates`라는 폴더 안에서 찾습니다. 하지만 기본적으로 생성되어 있지는 않으므로 직접 `templates` 폴더를 생성해야합니다. 우리의 `main` 앱 폴더 안에 `templates` 폴더를 생성하고 그 안에 `index.html` 템플릿 파일을 만듭니다.



1.3.1 장고 템플릿 언어

장고 템플릿에서는 장고만의 문법인 장고 템플릿 언어(django template language)를 사용할 수 있습니다. html 파일 안에서 원래 쓸 수 없었던 **변수**, **반복문**, **제어문** 등을 사용할 수 있습니다. 대체로 파이썬 문법과 비슷하므로 쉽게 익힐 수 있습니다.

- **변수** - 장고문서

출력하고자 하는 변수를 `{{ variable }}` 형식으로 출력할 수 있습니다.

```
{{ question }}
```

- **for** - 장고문서

반복문으로 `{% for [변수] in [반복가능한 개체] %}` 형식으로 사용합니다. 끝날 때 반드시 `{% endfor %}` 가 필요합니다.

```
{% for question in questions %}
    ...
{% endfor %}
```

- **if** - 장고문서

제어문으로 `{% if [조건] %}` 형식으로 사용합니다. `{% elif [조건] %}` 또는 `{% else %}` 으로 elif, else 문도 사용이 가능합니다. 끝날 때 반드시 `{% endif %}` 가 필요합니다.

```
{% if 조건1 %}
    ...
{% elif 조건2 %}
    ...
{% else %}
    ...
{% endif %}
```

- `{% csrf_token %}` - 장고문서

사이트 간 요청 위조(Cross-site request forgery)를 방지하는 토큰으로 **form** 태그 안에 반드시 작성해주어야 합니다.

`index.html` 파일을 다음과 같이 작성합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>index</title>
</head>
<body>
    <h1>index 페이지입니다</h1>
    <a href="/form/">설문 하기</a>
    {% for developer in developers %}
        <p>{{ developer.name }} : {{ developer.count }}명</p>
    {% endfor %}
</body>
</html>
```

이제 `[프로젝트 URL]`로 접속하면 해당 페이지가 보여지게 됩니다.

index 페이지입니다

[설문 하기](#)

백엔드 개발자(네트워크와 인프라를 포함) : 0명

프론트엔드 개발자 : 0명

데이터 분석과 인공지능 : 0명

정보보안 : 0명

게임 개발 : 0명

2. 설문 페이지

2.1 URL 설정

[프로젝트 URL]/form/ 으로 요청이 오면 우리가 만들 설문 페이지가 보이도록 해봅시다.

urls.py 를 다음과 같이 수정합니다.

```
from django.contrib import admin
from django.urls import path
from main import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),
    path('form/', views.form),
]
```

여기서 views.form 는 main/views.py 에 작성할 함수입니다. 이 설정은 [프로젝트 URL]/form/ 로 요청이 오면 main/views.py 에 있는 form 이라는 함수를 실행하겠다는 의미입니다.

2.2 뷰 함수 정의

main/views.py 에 form 함수를 다음과 같이 정의합니다.

```
def form(request):
    questions = Question.objects.all()

    context = {
        'questions': questions,
    }

    return render(request, 'form.html', context)
```

- form 함수의 역할은 form.html 을 페이지로 보여주는 역할입니다.
- Question.objects.all() : 모든 Question 요소를 불러옵니다.

2.3 템플릿 정의

`templates` 폴더를 생성하고 그 안에 `form.html` 템플릿 파일을 만들고 다음과 같이 작성합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>form</title>
</head>
<body>
    <form action="/result/" method="post">
        {% csrf_token %}
        {% for question in questions %}
            <h3>{{ question.number }}번 {{ question.content }}</h3>
            {% for choice in question.choice_set.all %}
                <div>
                    <input type="radio" name="question-{{ question.pk }}" id="choice-{{ choice.pk }}"
                        value="{{ choice.developer.pk }}">
                    <label for="choice-{{ choice.pk }}">{{ forloop.counter }}. {{ choice.content }}</label>
                </div>
            {% endfor %}
            <hr>
        {% endfor %}
        <input type="submit" value="제출하기">
    </form>
</body>
</html>
```

이제 `[프로젝트 URL]/form/` 로 접속하면 해당 페이지가 보여지게 됩니다.

1번 당신이 가장 재밌었던 수업은?

- 1. 프론트엔드(HTML, CSS, Javascript, etc)

3. 결과 페이지

3.1. URL 설정

[프로젝트 URL]/result/ 로 요청하면 우리가 만들 결과 페이지가 보이도록 해봅시다.

`urls.py` 를 다음과 같이 수정합니다.

```
from django.contrib import admin
from django.urls import path
from main import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', views.index),
    path('form/', views.form),
    path('result/', views.result),
]
```

여기서 `views.form` 는 `main/views.py` 에 작성할 함수입니다. 이 설정은 [프로젝트 URL]/result/ 로 요청이 오면 `main/views.py` 에 있는 `result` 라는 함수를 실행하겠다는 의미입니다.

3.2. 뷰 함수 정의

`main/views.py` 에 `result` 함수를 다음과 같이 정의합니다.

```
def result(request):

    # 문항 수
    N = Question.objects.count()
    # 개발자 유형 수
    K = Developer.objects.count()

    # 개발유형마다 몇 개인지 저장할 리스트 counter[1] = (1번 유형 점수(개수))
    counter = [0] * (K + 1)

    for n in range(1, N+1):
        developer_id = int(request.POST[f'question-{n}'][0])
        counter[developer_id] += 1

    # 최고점 개발 유형
    best_developer_id = max(range(1, K + 1), key=lambda id: counter[id])
    best_developer = Developer.objects.get(pk=best_developer_id)
    best_developer.count += 1
    best_developer.save()
```

```

context = {
    'developer': best_developer,
    'counter': counter
}

return render(request, 'result.html', context)

```

- `form` 함수의 역할은 `form.html` 을 페이지로 보여주는 역할입니다.
- `Developer.objects.get(pk=best_developer_id)`
`id` 값이 `best_developer_id` 인 요소를 찾습니다.

3.3. 템플릿 정의

`templates` 폴더를 안에 `form.html` 템플릿 파일을 만들고 다음과 같이 작성합니다.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>result</title>
</head>
<body>
    <h1>결과 페이지</h1>
    <p>{{ developer.name }}</p>
    <a href="/">메인화면</a>
    <a href="/form">다시하기</a>
</body>
</html>

```

이제 `[프로젝트 URL]/result/` 로 접속하면 해당 페이지가 보여지게 됩니다.

결과 페이지

프론트 엔드

[메인화면](#) [다시하기](#)

4. Django 개발 루틴

이제 Django를 어떻게 개발해야 하는지 순서가 감이 오시나요?

다시 한번 정리해보자면,

 `urls.py`에서 url을 설정해준다  `views.py`에서 url에 접속 했을 때 실행해 줄 함수를 작성한다  `templates` 안에 `템플릿 파일`을 만들고 사용자가 접근할 수 있도록 코드를 작성한다.



006. 장고 shell과 ORM

1. 장고 shell

1.1. shell

1.2. shell_plus

2. ORM

2.1. 조회

2.2. 생성

2.3. 수정

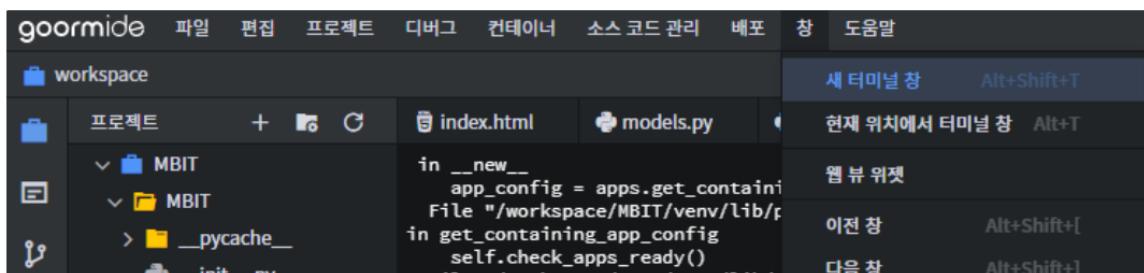
2.4. 삭제

1. 장고 shell

1.1. shell

장고 쉘(shell)은 파이썬 쉘 처럼 한 줄, 한 줄 명령을 통해 장고의 요소를 조작할 수 있는 CLI 환경입니다.

기존 터미널 창에는 서버를 켜두고 새 터미널 창에서 작업해보겠습니다. 메뉴 중 **창>새 터미널 창** 을 누르거나, **Alt+Shift+T** 버튼을 눌러 새 터미널 창을 엽니다.



💡 새 터미널 창에서 작업할 시, 반드시 `source venv/Scripts/activate` 명령어로 가상환경을 켜는 것을 잊지맙시다.

다음 명령을 통해 장고 shell을 실행시킬 수 있습니다.

```
python manage.py shell
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py shell
Python 3.7.4 (default, Nov  4 2020, 10:17:35)
[GCC 7.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 
```

종료하고 싶다면 `exit()` 함수를 호출하면 된다.

```
>> exit()
```

1.2. shell_plus

장고 shell의 불편한 점은 필요한 모델이 있으면 항상 `import` 를 해주어야 한다는 점입니다.

하지만 `django-extensions` 라는 패키지를 설치하면 `shell_plus` 라는 것을 사용할 수 있는데, 기본적으로 장고 shell과 동일하지만, 실행 시 해당 프로젝트 내의 필요한 것들을 자동으로 `import` 해준다는 점입니다.

- `django-extensions` 를 먼저 설치해줍니다.

```
pip install django-extensions
```

```
(venv) root@goorm:/workspace/MBIT# pip install django-extensions
Collecting django-extensions
  Downloading https://files.pythonhosted.org/packages/1a/e1/3e5142f101869b665820bc71d317706eb
labdbb1302f83728b1248258dc49/django_extensions-3.1.0-py3-none-any.whl (222kB)
  100% |██████████| 225kB 4.6MB/s
Installing collected packages: django-extensions
Successfully installed django-extensions-3.1.0
```

- 해당 프로젝트의 `settings.py` 의 `INSTALLED_APPS` 에 `django_extensions` 를 추가합니다. 설치 할 때와 다르게 `-` (dash)가 아니라 `_` (underscore)라는 점에 유의합니다.

```
INSTALLED_APPS = [
    ...,
    'django_extensions',
    ...,
]
```

- 이제 `shell_plus` 를 실행시켜 봅니다.

```
python manage.py shell_plus
```

사진과 같이 프로젝트를 다루는데 필요한 모듈들을 실행만 시켜도 자동으로 `import` 해줍니다.

```
(venv) root@goorm:/workspace/MBIT# python manage.py shell_plus
# Shell Plus Model Imports
from django.contrib.admin.models import LogEntry
from django.contrib.auth.models import Group, Permission, User
from django.contrib.contenttypes.models import ContentType
from django.contrib.sessions.models import Session
from main.models import Choice, Developer, Question
# Shell Plus Django Imports
from django.core.cache import cache
from django.conf import settings
from django.contrib.auth import get_user_model
from django.db import transaction
from django.db.models import Avg, Case, Count, F, Max, Min, Prefetch, Q, Sum, When
from django.utils import timezone
from django.urls import reverse
from django.db.models import Exists, OuterRef, Subquery
Python 3.7.4 (default, Nov  4 2020, 10:17:35)
[GCC 7.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
(InteractiveConsole)
>>> 
```

2. ORM

ORM이란 우리가 만든 모델 클래스와 DB에 생성된 테이블을 자동으로 연관지어 주는 기술로 우리가 DB를 직접 조작할 필요 없이 모델 클래스의 python 문법을 통해서 DB를 조작할 수 있는 편리한 기술입니다.

2.1. 조회

1. `all`

```
>> Question.objects.all()
<QuerySet [Question: 1. 당신이 가장 재밌었던 수업은?]>
```

2. `get`

```
>> Question.objects.get(pk=1)
<Question: 1. 당신이 가장 재밌었던 수업은?>
```

2.2. 생성

```
>> Question.objects.create(number=2, content="여러분의 서비스가 성공하였다면, 서비스를 성공  
시킨 요소는?")  
<Question: 2. 여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?>
```

2.3. 수정

```
>> question = Question.objects.get(pk=2)  
>> question.content = "수정된 문항"  
>> question  
<Question: 2. 수정된 문항>  
>> question.save()
```

 수정 후에는 반드시 `.save()` 를 해주어야 DB에 반영이 된다.

2.4. 삭제

```
>> question = Question.objects.get(pk=2)  
>> question.delete()  
(1, {'main.Question': 1})
```

- `(1, {'main.Question': 1})`

총 1 개의 요소가 삭제되었고, 그 중 `main.Question` 모델의 요소가 1 개 삭제되었다는 의미입니다.



007. Database Seeding

1. 데이터베이스 시딩(database seeding)이란?

2. dumpdata

3. loaddata

1. 데이터베이스 시딩(database seeding)이란?

데이터베이스의 초기 데이터를 설정하는 것을 말한다.

모델에 요소를 추가하는 방법으로 우리는 두 가지 방법을 사용하였습니다.

- 관리자 페이지
- 장고 Shell

그렇지만 아주 많은 데이터를 이러한 방법으로 일일이 저장하기에는 무리가 있습니다. 이럴 때 `loaddata` 와 `dumpdata` 명령어를 활용할 수 있습니다.

2. dumpdata

`dumpdata` 명령어는 현재 DB에 있는 데이터를 텍스트 형식의 파일로 저장하는 것을 말합니다. 추후에 이 파일은 `loaddata` 명령어를 통해 읽어서 원하는 DB에 저장할 수 있습니다.

```
python manage.py dumpdata main --output data.json
```

- `main` : `main` 이라는 앱의 모델들의 데이터를 저장
- `--output data.json` : `data.json` 이라는 파일로 저장

명령 수행 결과로 만들어진 `data.json` 파일은 다음과 같습니다.

```
[{"model": "main.developer", "pk": 1, "fields": {"name": "프론트 엔드"}}, {"model": "main.developer", "pk": 2, "fields": {"name": "백엔드 개발"}}, {"model": "main.developer", "pk": 3, "fields": {"name": "게임 개발"}}, {"model": "main.developer", "pk": 4, "fields": {"name": "데이터 분석과 인공지능"}}, {"model": "main.developer", "pk": 5, "fields": {"name": "정보보안"}}, {"model": "main.question", "pk": 1, "fields": {"number": 1, "content": "당신이 가장 재밌었던 수업은?"}}, {"model": "main.choice", "pk": 1, "fields": {"content": "프론트엔드(HTML, CSS, Javascript, etc)", "question": 1, "developer": 1}}]
```

하지만 한 줄로 출력되어 알아보기가 쉽지 않습니다. 그럴 때에는 `--indent 4` 옵션을 추가합니다.

```
python manage.py dumpdata main --output data.json --indent 4
```

그러면 아래와 같이 알아보기 쉽게 들여쓰기가 되어서 출력됩니다.

```
[
{
    "model": "main.developer",
    "pk": 1,
    "fields": {
        "name": "프론트 엔드"
    }
},
{
    "model": "main.developer",
    "pk": 2,
    "fields": {
        "name": "백엔드 개발"
    }
},
{
    "model": "main.developer",
    "pk": 3,
    "fields": {
        "name": "게임 개발"
    }
},
{
    "model": "main.developer",
    "pk": 4,
```

```

    "fields": {
        "name": "데이터 분석과 인공지능"
    }
},
{
    "model": "main.developer",
    "pk": 5,
    "fields": {
        "name": "정보보안"
    }
},
{
    "model": "main.question",
    "pk": 1,
    "fields": {
        "number": 1,
        "content": "당신이 가장 재밌었던 수업은?"
    }
},
{
    "model": "main.choice",
    "pk": 1,
    "fields": {
        "content": "프론트엔드(HTML, CSS, Javascript, etc)",
        "question": 1,
        "developer": 1
    }
}
]

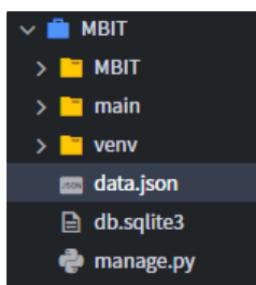
```

3. loaddata

`dumpdata`에서 설명했듯이, `loaddata` 는 `dumpdata`로 생성된 데이터 파일을 읽어서 DB에 저장하는 명령어입니다. `loaddata` 명령어를 이용해 우리가 미리 준비한 설문 데이터를 DB에 저장해봅시다.

④ `data.json` 14.4KB

프로젝트 폴더에 `data.json` 파일을 만들고 해당 내용을 모두 복사해서 붙여넣습니다.



이제 `loaddata` 명령어로 DB에 저장합니다.

```
python manage.py loaddata data.json
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py loaddata data.json
Installed 66 object(s) from 1 fixture(s)
```

사진처럼 나온다면 저장에 성공한 것입니다.

이제 설문 페이지로 접속하면 저장된 데이터들이 모두 잘 나오는 것을 확인할 수 있습니다.

1번 당신이 가장 재밌었던 수업은?

- 1. 게임 개발(유니티, 언리얼, Pygame)
- 2. 백엔드 또는 인프라(C계열-닷넷, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)
- 3. 정보보안(해킹과 방어)
- 4. 프론트엔드(HTML, CSS, Javascript, etc)
- 5. 데이터 분석과 인공지능(전처리, 분석, 시각화, 머신러닝, 딥러닝)
- 6. 이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.

2번 여러분의 서비스가 성공하였다면, 서비스를 성공시킨 요소는?

- 1. 데이터 분석을 통한 효과적인 개인 맞춤형 서비스
- 2. 화려하거나 품격있는 디자인, 흥미로운 콘텐츠 제작
- 3. 게임적인(Gamification, 게이미피케이션) 요소 도입
- 4. 안정적인 서비스 운영
- 5. 믿을 수 있는 정보 관리

3번 오류가 생겼습니다. 모두의 중요도가 동일하다고 할 때 당신이 먼저 처리할 오류는?

- 1. 서브 페이지 화면 로딩 시간이 딜레이되는 현상
- 2. 데이터 분석을 통한 맞춤형 광고 서비스 오류
- 3. 홈페이지 화면의 뒤틀린 이미지, 폰트 깨짐 현상
- 4. (개인정보 보호법에는 저촉되지 않지만) 민감 정보가 노출되는 현상
- 5. 오류가 발생했을 때 할 수 있는 게임

4번 당신이 만약 웹 서비스를 만들었을 때, 가장 듣기 좋을 것 같은 칭찬은?

- 1. 웹 서비스 디자인의 화려함, 유려함, 섬세함 > 친구 개발자 : "네가 만든 사이트 진짜 예쁘다!"
- 2. 웹 서비스의 기능 > 친구 개발자 : "동시 접속이 1천명인데 이렇게 빠르다고?" 또는 "SNS 로그인이 다 들어가 있네, 난 요즘 이런게 편하더라."
- 3. 시큐어 코딩 > 해커 친구 : "보안 취약점이 없는 완벽한 시큐어코딩이야. 너의 사이트는 완벽해."
- 4. 데이터 분석과 서비스의 인공지능 응용 > 친구 개발자 : "서비스를 해보니 내가 원하는 광고가 나오더라고" 또는 "회원수 2만명인 중고사이트에 마약같은 불법 게시물이 하나도 없다고?"
- 5. 흥미유발 및 재미요소 > 친구 게이머 : "사이트에 있는 그 게임 때문에, 2박 3일동안 잠도 자지 못했어."



008. static 파일

1. 정적(static) 파일이란?

2. 장고에서 정적파일 사용하기

2.1. STATIC_URL

2.2. 템플릿에서 static 파일 사용하기

1. 정적(static) 파일이란?

정적(static) 파일은 프로젝트를 서비스하는 동안 절대로 내용물이 변하지 않는 이미지 파일, JavaScript 파일, css 스타일 시트 파일 등을 말합니다. 언제나 내용이 변하지 않기 때문에 정적(static)이라고 부릅니다.

2. 장고에서 정적파일 사용하기

2.1. STATIC_URL

장고에서는 정적 파일을 사용할 때 `../static/image/1.png` 같은 파일 구조의 경로를 사용하지 않습니다. 왜냐하면 이러한 경로가 외부에 노출 될 시 프로젝트의 구조가 드러나기 때문에 보안상 취약하기 때문입니다. 그래서 이러한 프로젝트의 구조를 감추기 위해서 URL을 사용합니다.

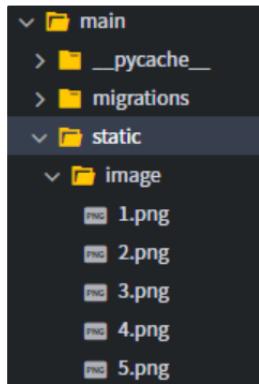
`settings.py` 의 가장 마지막 줄에 `STATIC_URL` 이 정의되어 있습니다.

```
STATIC_URL = '/static/'
```

해당 프로젝트에서 정적 파일에 접근하려면 무조건 `[프로젝트URL]/static/` URL을 사용해야 합니다.

해당 URL은 기본적으로 각 앱의 `static`에 접근합니다. 하지만 기본적으로 만들어지진 않기 때문에 개발자가 직접 static 폴더를 만들어주어야 합니다. `main` 앱 폴더 아래에 `static` 폴더를 만들고 그 안에 `image` 폴더를 만들고 다음의 이미지 파일들을 저장합니다.

📎 image.zip 501.1KB



그리고 서버를 재실행한 뒤 `[프로젝트 URL]/static/image/1.png`로 접속해봅니다. 그러면 해당 이미지 파일을 볼 수 있습니다.



2.2. 템플릿에서 static 파일 사용하기

템플릿에서 static URL을 사용하기 위해서는 `static 태그`를 사용해야 하는데, 이를 사용하기 위해서 먼저 템플릿 파일 초기에 `{% load static %}`을 해주어야 한다. 해당 템플릿 파일에서 static 태그를 사용하겠다는 뜻입니다.

static 태그는 `{% static '하위경로' %}`로 사용합니다. 그러면 settings.py에 설정된 STATIC_URL 값에 `하위경로`가 붙어서 출력됩니다. 예를 들어 `{% static 'images/' %}`이라고 하면 `/static/images/`로 출력됩니다.

`result.html` 템플릿을 다음과 같이 수정합니다.

```
% load static %
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>result</title>
</head>
<body>
    <h1>결과 페이지</h1>
    <p>{{ developer.name }}</p>
    
    <a href="/">메인화면</a>
    <a href="/form">다시하기</a>
</body>
</html>
```

이제 설문 결과 개발자 유형에 따라 사진이 같이 출력됩니다.

결과 페이지

백엔드 개발자(네트워크와 인프라를 포함)



[메인화면](#) [다시하기](#)



009. 템플릿 적용하기

지금까지 프론트엔드(템플릿)과 백엔드(서버)를 따로 개발하였습니다. 왜냐하면 장고 템플릿의 경우 `html/css/javascript` 뿐 아니라 `장고 템플릿 언어(django template language)` 가 섞여 있어서 스타일링과 서버 개발을 동시에 하면 매우 복잡해집니다. 따라서 템플릿은 따로 `html/css/javascript` 으로만 작업을 하여 완성한 뒤 서버가 완성되면 `장고 템플릿 언어` 를 통해 데이터를 뿌려주는 방식으로 완성합니다. 이제 프론트엔드 파트에서 완성한 템플릿을 가져와 장고 템플릿 언어를 추가해 봅시다.

1. static 파일 가져오기

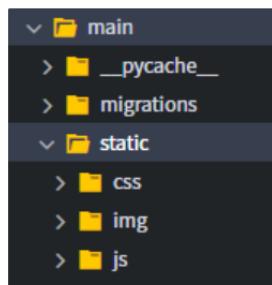
2. 메인 페이지

3. 설문 페이지

4. 결과 페이지

1. static 파일 가져오기

템플릿에 사용될 `js` 파일과 `css` 파일을 앱의 `static` 폴더에 배치합니다.



2. 메인 페이지



```
{% load static %}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section id="main_contents">
        <div class="wrapper">
            <div class="title">
                <h3 class="main_title">나의 MBIT</h3>
                <h3 class="sub_title">My Best IT personalities</h3>
            </div>
        </div>
    </section>
</body>
```

```

<div class="explain">
    <p>
        "코딩 공부를 시작하려는데...<br>
        도대체 뭐 부터 공부해야 할까?"<br><br>
        나의 개발 유형을 테스트 해보자!
    </p>
</div>
<div class="buttons">
    <a href="/form/">
        <button class="start" type="button">시작하기</button>
    </a>
    <a href="http://www.paullab.co.kr/curriculum.html" target="_blank">
        <button class="lecture">강의 보러가기</button>
    </a>
</div>
<div class="result_data">
    <div class="data_wrap">
        <ul>
            {% for developer in developers %}
                <li>{{ developer.name }} : {{ developer.count }}명</li>
            {% endfor %}
        </ul>
    </div>
</div>
<div class="weniv">
    <a href="http://www.paullab.co.kr">
        
    </a>
</div>
</div>
</section>
</body>
</html>

```

- `{% load static %}` : `static` 태그를 쓸 수 있도록 합니다.
- 모든 static 파일의 경로를 `static` 태그로 표현합니다.
- 설문 참여 현황을 `for` 태그를 통해 표현합니다.

```

<ul>
    {% for developer in developers %}
        <li>{{ developer.name }} : {{ developer.count }}명</li>
    {% endfor %}
</ul>

```

3. 설문 페이지

1/10

당신이 가장 재밌었던 수업은?

- 1. 게임 개발(유니티, 언리얼, Pygame)
- 2. 백엔드 또는 인프라(C계열-닷넷, JAVA-Spring, Python-Django, Network, Ubuntu, CentOS, 부하분산, etc)
- 3. 정보보안(해킹과 방어)
- 4. 프론트엔드(HTML, CSS, Javascript, etc)
- 5. 데이터 분석과 인공지능(전처리, 분석, 시각화, 머신러닝, 딥러닝)
- 6. 이 길은 내 길이 아닌 것 같다... 고로 재미있는지 모르겠다.

다음

```
{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/form.css' %}">
    <script src="https://code.jquery.com/jquery-3.5.1.js"></script>
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section id="survey">
        <div class="wrapper">
            <form id="form" action="/result/" method="post">
                {% csrf_token %}
                {% for question in questions %}
                    <div class="test">
                        <div class="question_container">
                            <h3 class="number">{{ question.number }}/{{ questions.count }}</h3>
                            <h3 class="question">{{ question.content }}</h3>
                        </div>
                    </div>
                {% endfor %}
            </form>
        </div>
    </section>
</body>
```

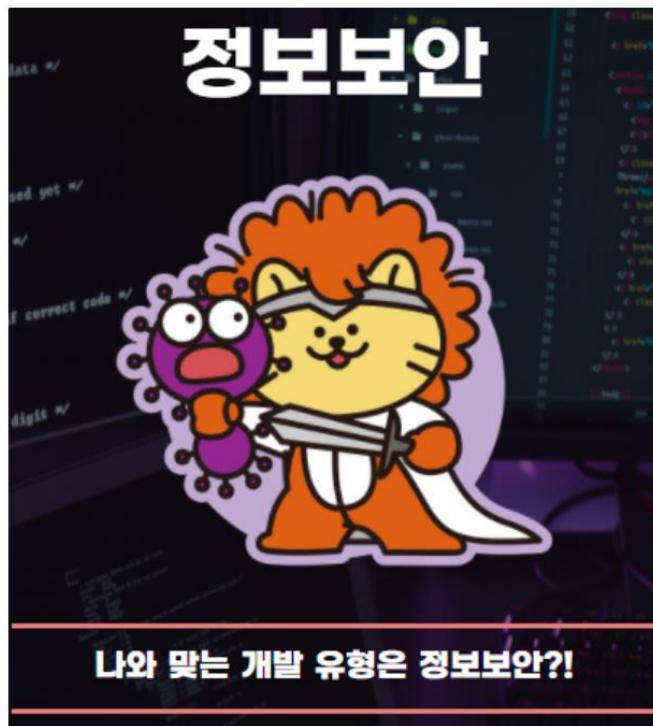
```

<div class="answer">
    {% for choice in question.choice_set.all %}
        <div>
            <input
                type="radio"
                name="question-{{ question.number }}"
                id="choice-{{ choice.pk }}"
                value="{{ choice.developer.pk }}"
            />
            <label for="choice-{{ choice.pk }}">{{ forloop.counter }}.
            {{ choice.content }}</label>
        </div>
    {% endfor %}
</div>
{% if not forloop.first %}
<div class="btn_wrap btn_sort">
    {% else %}
        <div class="btn_wrap">
    {% endif %}
    {% if not forloop.first %}
        <button class="prev_btn">이전</button>
    {% endif %}
    {% if not forloop.last %}
        <button class="next_btn">다음</button>
    {% else %}
        <input type="submit" value="제출" class="submit_btn"/>
    {% endif %}
    </div>
</div>
    {% endfor %}
</form>
</div>
</section>
<script type="text/javascript" src="{% static 'js/form.js' %}"></script>
</body>
</html>

```

- `{% load static %}` : `static` 태그를 사용할 수 있도록 합니다.
- `{% csrf_token %}` : CSRF 보안을 위해 항상 `form` 태그 안에 넣어주어야 합니다.
- 데이터들을 변수와 `for`, `if` 태그를 이용해 채워줍니다.

4. 결과 페이지



```
% load static
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/result.css' %}">
    <title>나의 개발 유형찾기</title>
</head>
<body>
    <section id="main_contents">
        <div class="wrapper">
            <div class="result">
                <div class="titles">
                    <h3>예술적 감각이 뛰어난 당신은!</h3>
                    <h1>{{ developer.name }}</h1>
                </div>
                <div class="result_img">
                    
                </div>
            </div>
            <div class="result_explains">
                <div class="explain">
                    <h3 class="title">나와 맞는 개발 유형은 {{ developer.name }}?!</h3>
                </div>
            </div>
        </div>
    </section>
</body>
```

```

<ul>
  <li>
    프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
  </li>
  <li>
    프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
  </li>
  <li>
    프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
  </li>
</ul>
</div>
<div class="explain">
  <h3 class="title">Front-end 개발자가 뭐지?</h3>
  <ul>
    <li>
      프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
    </li>
    <li>
      프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
    </li>
    <li>
      프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그럴기 때문에 HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
    </li>
  </ul>
  <div class="img_wrap">
    
  </div>
  <h3>HTML</h3>
</li>

```

```

<li>
    <div class="img_wrap">
        
    </div>
    <h3>CSS</h3>
</li>
<li>
    <div class="img_wrap">
        
    </div>
    <h3>JavaScript</h3>
</li>
</ul>
<p>
    프론트엔드는 어쩌고 저쩌고 이기 때문에 기본적으로 무엇을 그릴지 때문에
    HTML,CSS를 공부하고 JS를 어쩌고 하는 것이 좋아!
</p>
</div>
</div>
<div class="lectures">
    <h3 class="title">강의 추천</h3>
    <p>Front-end 공부를 시작하기 좋은 강의를 추천해 드릴게요!</p>
    <ul>
        <li>
            
            <h3>HTML, CSS, JS, Python 30분 요약강좌</h3>
            <a href="https://www.inflearn.com/course/제주코딩-웹개발-30분요약"
            target="_blank">
                <button type="button">강의 보러가기</button>
            </a>
        </li>
        <li>
            
            <h3>코알못에서 웹서비스 런칭까지 : 제주 코딩 베이스캠프(Django)</h3>
            <a href="https://www.inflearn.com/course/web_fullstack" target
            ="_blank">
                <button type="button">강의 보러가기</button>
            </a>
        </li>
    </ul>
</div>
<div class="buttons">
    <ul>
        <li>
            <h3>...이건 내가 아니야... 잘못된게 분명해!</h3>
            <a href="/">
                <button type="button">테스트 다시 하기</button>
            </a>
        </li>
    </ul>

```

```

<il>
    <h3>다른 사람들은 어떤 유형일까?</h3>
    <a href="#">
        <button type="button">다른 결과 알아보기</button>
    </a>
</il>
<il>
    <h3>이런 테스트는 도대체 누가 만든거야? ^0^;;</h3>
    <a href="#">
        <button class="color" type="button">제주코딩베이스캠프</button>
    >
        </a>
    </il>
</ul>
</div>
<div class="weniv">
    <a href="http://www.paullab.co.kr">
        
    </a>
    <p>
        ※ 본 서비스 내 이미지 및 콘텐츠의 저작권은 주식회사 위니브에 있습니다.<br>
    >
        수정 및 재배포, 무단 도용 시 법적인 문제가 발생할 수 있습니다.
    </p>
</div>
</div>
</section>
</body>
</html>

```

- `{% load static %}` : `static` 태그를 사용할 수 있도록 합니다
- 데이터들을 변수와 `for`, `if` 태그를 이용해 채워줍니다.



[심화] 배포하기

1. 배포 전 준비

1.1. 환경 변수 설정

1.2. collectstatic

2. WSGI

2.1. uWSGI

- uWSGI 실행해보기

- ini 파일로 uWSGI 실행

3. Nginx

3.1. 웹서버

3.2. Nginx 설정하기

3.3. uWSGI 소켓 설정

3.4. Nginx 실행하기

4. 구름 IDE 실행 명령 추가

1. 배포 전 준비

1.1. 환경 변수 설정

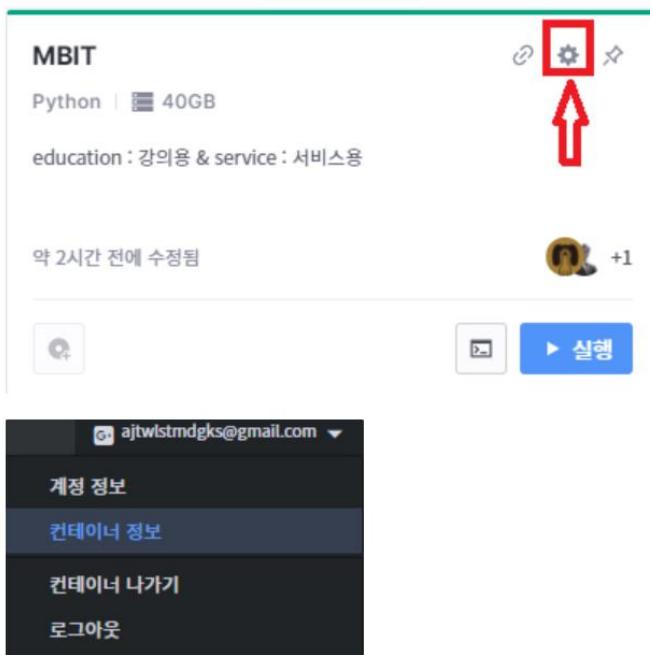
프로젝트의 `settings.py` 파일에 보면 `SECRET_KEY` 와 `DEBUG` 변수가 있습니다.

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = '%1jvk*h2sno=wepa5p)dt+c6gtq44@fn^c1=7&+8ypc)k4xe16'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True
```

- `SECRET_KEY` : 프로젝트의 보안 등에 이용되는 긴 랜덤 문자열 값입니다. 보안과 관련되어 있으므로 외부에 노출되어서는 안 됩니다.
- `DEBUG` : 이 값이 `True` 이면 개발 단계에서 개발자를 위해 도와주는 디버그 정보가 보여지게 됩니다. 하지만 배포 시에는 프로젝트 보안 상 디버그 정보가 노출 되어서는 안 됩니다.

1. 컨테이너의 설정으로 들어갑니다. 컨테이너 목록에서 톱니바퀴 아이콘을 누르거나, IDE 실행 화면에서 오른쪽 상단의 `컨테이너 정보`를 통해 들어갈 수 있습니다.



2. 다음과 같이 환경 변수를 등록합니다.

환경변수		+ 추가
Key	Value	...
SECRET_KEY	%6jvk*h2sno=wepa5p)dt+c6gtq44@fn^cl=7&+8ypc)k4xe16	...
DEBUG	False	...

root 권한 사용자에게만 공개/적용되고 IDE 페이지 새로 고침 후에 추가됩니다.

3. `settings.py`에서 이 환경 변수들을 사용하여 `SECRET_KEY` 와 `DEBUG` 값을 정의합니다. 환경 변수의 값들은 os 모듈의 `environ`을 통해 접근할 수 있습니다.

```
# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = os.environ['SECRET_KEY']

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True if os.environ['DEBUG'] == 'True' else False
```

환경변수 값들은 모두 문자열로 저장되기 때문에 `True` 를 넣더라도 `bool` 타입의 `True` 가 아니라 문자열 `'True'` 인 것에 유의하세요.

1.2. collectstatic

장고의 `runserver` 명령을 통해 프로젝트를 실행하였지만, 이는 개발용으로 임시로 실제 배포할 때에는 적합하지 않습니다. 그래서 뒤에 설명될 웹서버라는 것을 사용합니다. 하지만 웹서버를 사용하면 이제 앱마다 가지고 있는 static 파일에 접근하는 django 서버의 기능을 사용하지 못합니다. 그래서 앱마다 나누어져 있는 static 파일들을 한 곳에 모으는 과정이 필요한데 이를 해주는 명령어가 `collectstatic`입니다.

1. `settings.py`에서 `STATIC_ROOT` 변수에 static 파일들을 모을 경로를 설정합니다.

```
STATIC_ROOT = BASE_DIR / 'staticfiles'
```

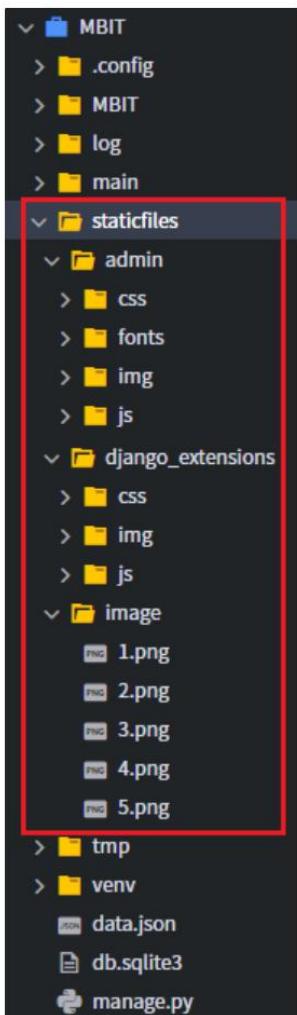
`BASE_DIR` 은 프로젝트 생성 시 자동으로 만들어 준 경로값으로 `manage.py` 가 있는 경로를 가리킵니다. 우리는 `BASE_DIR` 경로의 `staticfiles` 라는 폴더에 static 파일들을 모을 것입니다.

2. `collectstatic` 명령어로 static 파일들을 모읍니다.

```
$ python manage.py collectstatic
```

```
(venv) root@goorm:/workspace/MBIT# python manage.py collectstatic
142 static files copied to '/workspace/MBIT/staticfiles'.
```

그러면 해당 경로에 `staticfiles`라는 폴더가 생성되고 안에 static 파일들이 모인 것을 확인할 수 있습니다.



2. WSGI

WSGI 란 Web Server Gateway Interface 웹서버와 파이썬 앱 사이를 연결해주는 장치입니다. 웹서버와 우리가 만든 장고 웹 애플리케이션은 직접적으로 상호 간의 통신이 불가능하기 때문에 그 사이에서 **WSGI** 가 중재자 역할을 해줍니다.

사용자가 웹서버에 요청을 보내면 **WSGI** 가 django에 넘겨주고, django는 요청을 처리한 뒤 응답을 **WSGI** 를 통해 웹서버로 전달합니다.

사용자 <-> 웹 서버 <-> WSGI <-> 장고

2.1. uWSGI

uWSGI 는 WSGI의 기능을 구현한 파이썬 패키지입니다. **uWSGI** 를 이용하여 장고 프로젝트를 실행시켜보겠습니다.

- uWSGI 실행해보기

1. **uWSGI** 를 설치할 가상환경을 활성화합니다.

```
$ source [가상환경폴더경로]/bin/activate
```

2. **pip** 를 이용해 **uWSGI** 를 설치합니다. (약간의 시간 소요)

```
$ pip install uwsgi
```

3. **uwsgi** 를 실행시켜봅니다.

```
$ uwsgi \
--http :80 \
--home /workspace/MBIT/venv/ \
--chdir /workspace/MBIT/ \
--static-map /static=/workspace/MBIT/staticfiles/ \
-w MBIT.wsgi
```

```
$ uwsgi --http :80 --home /workspace/MBIT/venv --chdir /workspace/MBIT --static-map
/static=/workspace/MBIT/staticfiles/ -w MBIT.wsgi
```

```
(venv) root@goorm:/workspace/MBIT# uwsgi --http :80 --home /workspace/MBIT/venv --chdir /workspace/MBIT -w MBIT.wsgi
*** Starting uWSGI 2.0.19.1 (64bit) on [Fri Jan  8 01:21:59 2021] ***
compiled with version: 7.5.0 on 08 January 2021 01:14:31
os: Linux-4.4.0-1119-aws #133-Ubuntu SMP Tue Dec 1 19:04:22 UTC 2020
nodename: goorm
machine: x86_64
clock source: unix
pcre jit disabled
detected number of CPU cores: 2
current working directory: /workspace/MBIT
detected binary path: /workspace/MBIT/venv/bin/uwsgi
uWSGI running as root, you can use --uid/--gid/--chroot options
*** WARNING: you are running uWSGI as root !!! (use the --uid flag) ***
chdir() to /workspace/MBIT
```

이제 **runserver** 명령 없어도 80번 포트로 접속하면 프로젝트 페이지로 접속이 가능합니다.

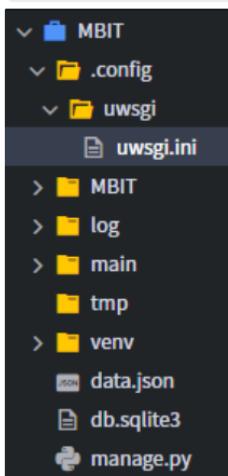
index 페이지입니다

[설문 하기](#)

- ini 파일로 uWSGI 실행

uWSGI를 실행하기 위해서 옵션이 많이 필요합니다. 또한 해당 옵션들을 절대 경로로 적어주다보니 명령어 하나의 양이 적지 않습니다. 이를 위해 `.ini` 파일에 해당 설정들을 저장하여 실행할 수 있습니다.

1. `/workspace/MBIT/.config/uwsgi/uwsgi.ini` 경로에 `wsgi.ini` 파일을 만듭니다.



2. `uwsgi.ini` 파일을 다음과 같이 수정해줍니다.

```
[uwsgi]
chdir = /workspace/MBIT/
module = MBIT.wsgi:application
home = /workspace/MBIT/venv/

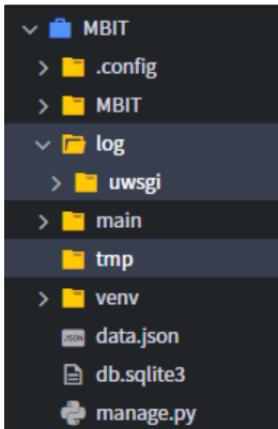
uid = root
gid = root

http = :80

enable-threads = true
master = true
vacuum = true
pidfile = /workspace/MBIT/tmp/MBIT.pid
logto = /workspace/MBIT/log/uwsgi/@(exec://date +%%Y-%%m-%%d).log
log-reopen = true
static-map = /static=/workspace/MBIT/staticfiles/
```

- `chdir` : django project의 manage.py가 존재하는 경로를 지정해줍니다.
- `module` : wsgi 파일을 지정해줍니다.
- `home` : 가상환경의 경로를 지정해줍니다.
- `uid`, `gid` : uWSGI를 실행할 사용자 및 사용자그룹을 지정해줍니다.
- `http` : http 프로토콜을 통해서 요청을 받으며 포트 번호를 정해줍니다.
- `enable-threads` : 스레드 사용 여부를 결정합니다.
- `master` : 마스터 프로세스 사용 여부를 결정합니다.
- `vacuum` : 실행 시 자동 생성되는 파일들을 삭제해줍니다.
- `pidfile` : 실행되는 프로세스의 id 값을 담고 있는 파일, pidfile의 경로를 지정해줍니다.
- `logto` : 로그파일을 작성할 위치를 설정합니다.
- `log-reopen` : 재시작할 시 로그를 다시 열어줍니다.

3. 프로젝트 경로에 `tmp` 폴더와 `log/uwsgi` 폴더를 만듭니다.



4. `uwsgi` 명령어로 `uwsgi.ini` 파일을 실행시킵니다.

```
$ uwsgi -i .config/uwsgi/uwsgi.ini
```

```
(venv) root@goorm:/workspace/MBIT# uwsgi -i .config/uwsgi/uwsgi.ini
[uWSGI] getting INI configuration from .config/uwsgi/uwsgi.ini
```

5. 이제 80번 포트의 URL로 접속합니다.

index 페이지입니다

[설문 하기](#)

3. Nginx

`uwsgi` 를 사용하여 django를 웹서버와 연결시킬 준비를 마쳤습니다.
이번에는 본격적으로 웹서버에 대해서 알아보도록 하겠습니다.

3.1. 웹서버

웹서버란 HTTP를 통해 웹 브라우저에서 요청하는 HTML 문서나 오브젝트(이미지 파일 등)을 전송해 주는 서비스 프로그램을 말합니다.

웹서버도 여러 종류가 있어 웹서버마다 차이가 존재하겠지만 대체로 다음과 같은 기능들을 제공합니다.

- 인증
- 정적 콘텐츠 관리
- HTTPS 지원
- 콘텐츠 압축
- 가상 호스팅
- 대용량 파일 지원
- 대역폭 스로틀링

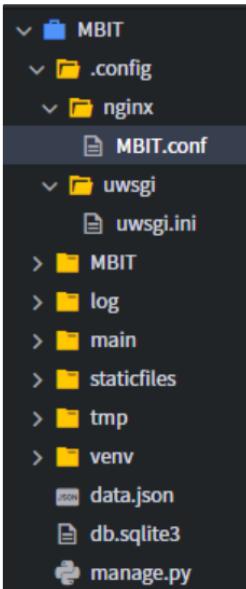
아파치, microsoft의 IIS, nginx 등 여러 웹서버가 있지만 여기서는 nginx를 사용해보도록 하겠습니다.

3.2. Nginx 설정하기

1. 구름 ide에서는 `nginx` 가 기본적으로 설치되어 있습니다. 다음 명령어를 통해 `nginx` 설치여부와 버전을 확인할 수 있습니다.

```
$ nginx -v
nginx version: nginx/1.14.0 (Ubuntu)
```

2. `.config/nginx/MBIT.conf` 파일을 만듭니다. 내용을 다음과 같이 수정합니다.



3. `MBIT.conf` 파일을 다음과 같이 수정합니다.

```

server {
    listen 80;
    server_name *.run.goorm.io;
    charset utf-8;
    client_max_body_size 128M;

    location / {
        uwsgi_pass unix:///workspace/MBIT/tmp/MBIT.sock;
        include uwsgi_params;
    }

    location /static/ {
        alias /workspace/MBIT/staticfiles/;
    }
}
  
```

4. `MBIT.conf` 파일을 `/etc/nginx/sites-available/` 폴더에 복사합니다.

```
$ cp -f /workspace/MBIT/.config/nginx/MBIT.conf /etc/nginx/sites-available/
```

`sites-available` 폴더는 가상 서버 환경들에 대한 설정 파일들이 위치하는 부분입니다. 가상 서버를 사용하거나 사용하지 않던간에 그에 대한 설정 파일들이 위치하는 곳입니다.

5. `sites-available` 로 이동한 설정 파일을 `sites-enabled`에 링크해줍니다.

```
$ ln -sf /etc/nginx/sites-available/MBIT.conf /etc/nginx/sites-enabled/
```

`sites-enabled` 는 `sites-available` 에 있는 가상 서버 파일들중에서 실행시키고 싶은 파일을 symlink로 연결한 폴더입니다. 실제로 이 폴더에 위치한 가상서버 환경 파일들을 읽어서 서버를 세팅합니다.

6. `sites-enabled` 폴더에 있던 `default` 파일은 삭제해줍니다.

```
$ rm /etc/nginx/sites-enabled/default
```

3.3. uWSGI 소켓 설정

`nginx` 를 설치하고 설정을 완료했으니 기존의 `uWSGI` 가 `nginx`와 연결되도록 설정을 해주겠습니다. 웹서버인 `nginx` 와 `uWSGI` 사이의 통신을 매개 HTTP 요청을 사용할 수도 있지만 서버 안쪽에서의 통신이기 때문에 소켓 방식이 overhead가 적어서 더 효율적입니다.

따라서 이를 위한 설정을 해주도록 하겠습니다.

다시 로컬로 넘어와 `uwsgi.ini` 파일을 수정해줍시다.

```
[uwsgi]
chdir = /workspace/MBIT/
module = MBIT.wsgi:application
home = /workspace/MBIT/venv/

uid = root
gid = root

##### 0| 부분 #####
socket = /workspace/MBIT/tmp/MBIT.sock
chmod-socket = 666
chown-socket = root:root
#####

enable-threads = true
master = true
vacuum = true
pidfile = /workspace/MBIT/tmp/MBIT.pid
logto = /workspace/MBIT/log/uwsgi/@(exec://date +%%Y-%%m-%%d).log
log-reopen = true
static-map = /static=/workspace/MBIT/staticfiles/
```

3.4. Nginx 실행하기

- 다음 명령어로 `nginx`를 실행합니다.

```
$ service nginx start
```

```
root@goorm:/workspace/MBIT# service nginx start
* Starting nginx nginx
...done.
```

- `80` 번 포트의 URL로 접속하면 사진처럼 nginx가 작동하는 것을 확인할 수 있습니다.

502 Bad Gateway

nginx/1.14.0 (Ubuntu)

아직 페이지가 정상적으로 보이지 않는 것은 `uWSGI` 가 실행 되지 않아 소켓이 없는 상태이기 때문입니다.

3. 이제 uwsgi 를 실행시켜 소켓을 활성화합니다.

```
$ uwsgi -i .config/uwsgi/uwsgi.ini
```

```
(venv) root@goorm:/workspace/MBIT# uwsgi -i .config/uwsgi/uwsgi.ini
[uWSGI] getting INI configuration from .config/uwsgi/uwsgi.ini
[uwsgi-static] added mapping for /static => /workspace/MBIT/staticfiles/
```

이제 다시 접속하면 프로젝트에 정상적으로 접속할 수 있습니다.

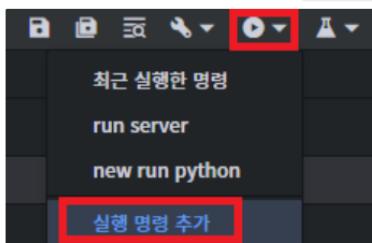
index 페이지입니다

[설문 하기](#)

4. 구름 IDE 실행 명령 추가

매번 프로젝트를 실행할 때마다 nginx 또는 uwsgi 를 실행시키는 명령어를 작성하는 것은 번거로운 일입니다. 구름IDE에서는 이러한 명령어들을 미리 저장해서 실행 시킬 수 있습니다.

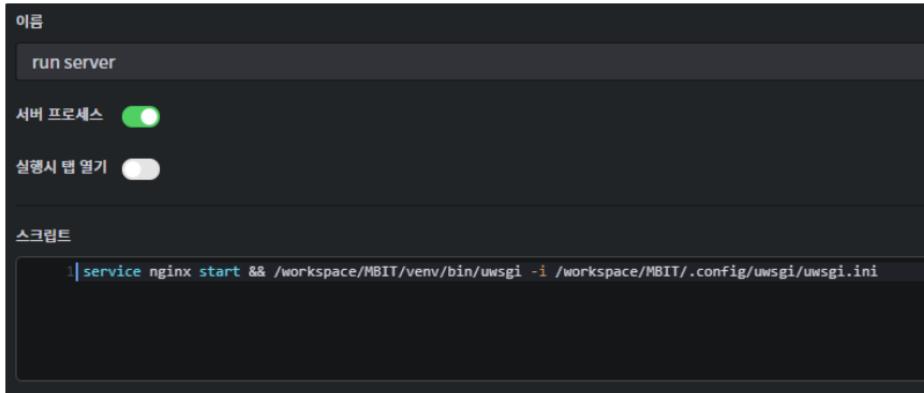
1. 구름IDE 오른쪽 상단에서 실행 명령 추가 를 클릭합니다.



2. ① 템에 본인이 원하는 이름을 짓고 해당 명령의 스크립트를 다음과 같이 작성합니다.

```
service nginx start && /workspace/MBIT/venv/bin/uwsgi -i /workspace/MBIT/.config/uwsgi/uwsgi.ini
```

- `service nginx start` : nginx를 실행합니다.
- `&&` : 앞의 명령어가 성공했을 시, 다음 명령어를 실행합니다.
- `/workspace/MBIT/venv/bin/uwsgi -i /workspace/MBIT/.config/uwsgi/uwsgi.ini` : uwsgi를 실행합니다.



3. 서버 프로세스는 ON으로 활성화 하면 실행시 프로젝트 URL을 보여줍니다.

```
Croot@goorm:/workspace/MBIT# service nginx start && /workspace/MBIT/venv/bin/uwsgi -i /workspace/MBIT/.config/uwsgi/uwsgi.ini
* Starting nginx nginx
...done.
[uWSGI] getting INI configuration from /workspace/MBIT/.config/uwsgi/uwsgi.ini
[uwsgi-static] added mapping for /static => /workspace/MBIT/staticfiles/
```

`종료` 버튼으로 해당 명령을 종료할 수 있고, `편집` 버튼으로 수정이 가능합니다.



[심화] 프로젝트 개선

1. Redirect

1.1. 결과 페이지의 문제점

1.2. redirect란?

1.3. views.py 수정

1.4. URL 인자

2. URL 설정 개선

2.1. URL 분리

2.2. URL name

3. 템플릿 개선

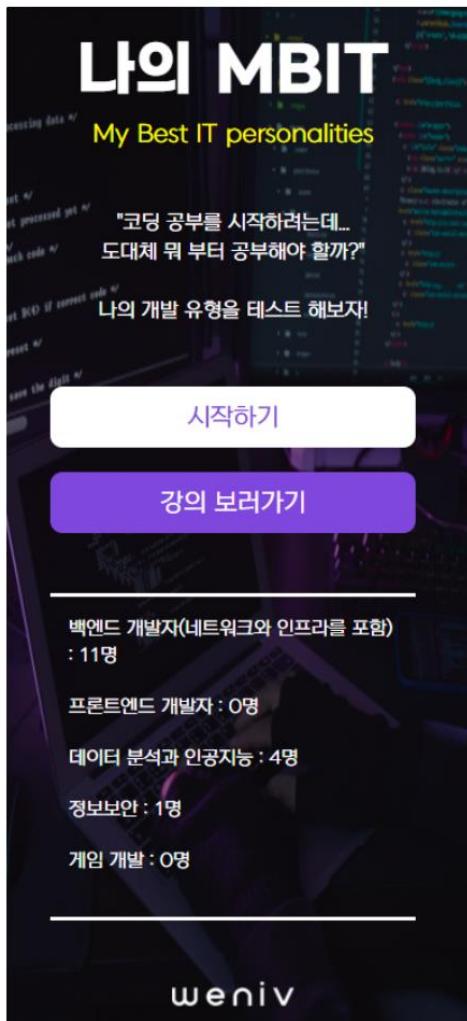
3.1. 템플릿 이름 구별

3.2. 템플릿 상속

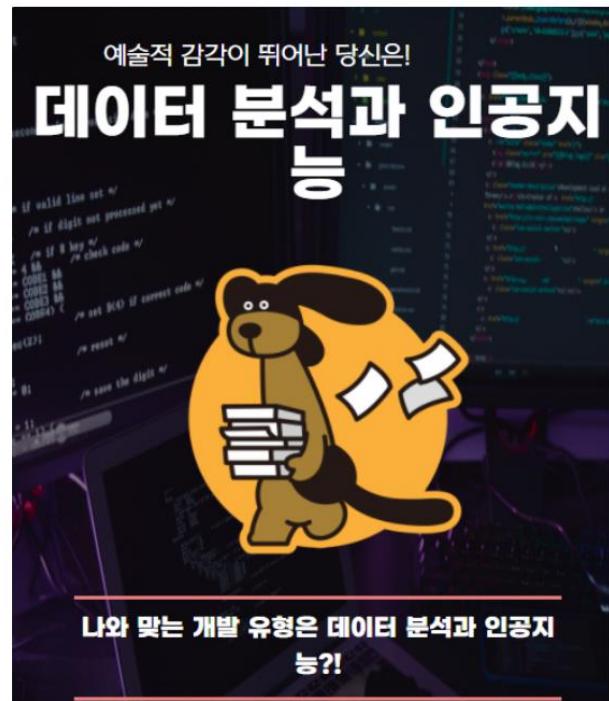
1. Redirect

1.1. 결과 페이지의 문제점

지금까지 만든 MBIT 프로젝트의 결과 페이지에는 치명적인 문제점이 있습니다. 웹 브라우저의 탭을 두 개 열어서 하나는 메인 페이지를 하나는 설문을 마친 결과 페이지로 접속합니다.

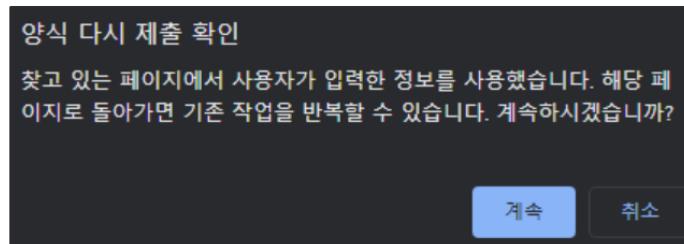


메인 페이지

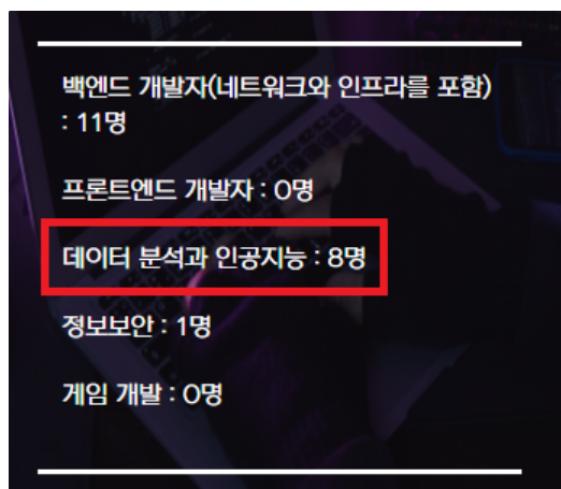


결과 페이지

현재 메인 페이지를 보면 데이터 분석과 인공지능이 결과로 나온 인원은 4명입니다. 이제 결과 페이지를 새로고침 해봅시다. 그러면 사진처럼 양식 다시 제출 확인 메시지 창이 뜨는데 계속 버튼을 눌러봅니다.



이 새로고침 과정을 여러 번 반복해봅니다. 그 후 메인 페이지 텁을 새로고침 해봅니다. 그러면 결과 페이지를 새로고침한 횟수만큼 데이터 분석과 인공지능 인원이 늘어나 있는 것을 확인 할 수 있습니다.



이는 당연히 우리가 원하는 결과가 아닙니다. 사용자가 결과 페이지를 새로고침해도 페이지만 새로 로딩되기만 원하지 설문 결과가 다시 제출되는 것은 원치 않습니다.

웹 브라우저는 요청을 보내고 응답을 받은 뒤에도 보낸 요청이 무엇이었는지 기억하고 있습니다. 유저가 새로고침을 하면 기억하고 있던 요청을 다시 보내는 것입니다. 즉, 결과 페이지에서 새로고침을 하면 설문 데이터가 서버로 다시 전달되는 것입니다. 이를 방지하기 위해서 `redirect` 를 사용해야 합니다.

1.2. redirect란?

`redirect` 는 브라우저에게 어떤 특정한 url로 재요청을 하라고 명령하는 것입니다. 설문 제출 요청을 처리를 한 뒤, 결과 페이지를 렌더링하기만 하는 요청으로 리다이렉트하여 문제점을 해결할 수 있습니다. 설문 제출 요청 후에 결과 페이지를 받기만 하는 요청이 한 번 더 이루어지기 때문에 더 이상 처음의 설문 제출 요청은 웹 브라우저에 남지 않게 됩니다. 왜냐하면 웹 브라우저는 이전 상태는 전혀 기억하지 못하게 되어있기 때문입니다.

[이전] : 설문제출 요청 → 처리 후 결과페이지 렌더링

[이후] : 설문 제출 요청 → 처리 → 결과 페이지 요청 → 결과 페이지 렌더링

1.3. views.py 설정

이제 `views.py` 에서 설문 처리와 결과 페이지를 보여주는 `result` 함수를 두 개로 나누어야 합니다. 하나는 설문 처리만 하는 `submit` 함수, 하나는 결과 페이지를 렌더링 하는 `result` 함수로 나눌 것입니다.

1. 기존의 `result` 함수의 이름을 `submit` 으로 바꿉니다.

```
def submit(request):
    ...
```

2. `urls.py` 에서 `submit` 을 요청할 URL로 `/submit/` 을 추가합니다.

```
urlpatterns = [
    ...
    path('submit/', views.submit), # 여기
]
```

3. `form.html` 에서 `<form>` 태그의 `action` 속성의 URL을 `/submit/` 으로 바꿉니다.

```
<form id="form" action="/submit/" method="post">
```

4. 결과 페이지 `result.html` 을 렌더링할 `result` 함수를 새로 정의합니다.

```
def result(request):
    return render(request, 'result.html')
```

5. `submit` 함수의 return 값을 다음과 같이 `redirect`로 `/result/`로 재요청을 보내도록 합니다.
(`redirect`를 반드시 import 해줍니다.)

```
from django.shortcuts import render, redirect

def submit(request):
    ...
    return redirect('/result/')
```

1.4. URL 인자

여기서 문제가 있습니다. 결과페이지에는 결과 개발유형에 대한 정보를 `context`로 넘겨주었습니다.
하지만 `redirect` 시에는 `context`로 넘겨줄 수 없습니다. 대신에 `URL argument`를 이용합니다.

1. `urls.py` 에서 결과 페이지를 보여주는 URL을 다음과 같이 수정합니다.

```
urlpatterns = [
    ...
    path('result/<int:developer_id>', views.result),
]
```

이는 URL 패턴으로 int 타입의 `developer_id`라는 인자를 받을 수 있다는 뜻입니다.

2. `submit` 함수의 `redirect`를 다음과 같이 수정합니다.

```
def submit(request):
    ...
    return redirect(f'/result/{best_developer_id}')
```

그러면 URL 패턴의 `<int:developer_id>` 자리에 `best_developer_id` 값이 들어가게 됩니다. 예를 들어, `best_developer_id` 값이 `2`라면 `/result/2/` 라는 URL로 요청이 가게 됩니다.

3. `result` 함수에서 `<int:developer_id>` 인자 값을 받으려면 똑같은 이름의 매개변수를 설정해주세요 됩니다.

```
def result(request, developer_id):
    ...
```

4. 이제 `developer_id` 를 이용해 개발자 유형을 조회하여 `context` 로 넘겨주면 됩니다.

```
def result(request, developer_id):
    developer = Developer.objects.get(pk=developer_id)
    context = {
        'developer': developer,
    }
    return render(request, 'result.html', context=context)
```

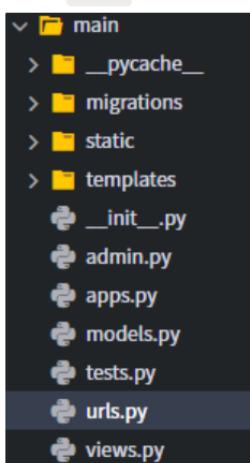
이제 아무리 결과 페이지에서 새로고침을 해도 `result` 함수만 호출되기 때문에 설문 참여자 수는 늘어나지 않습니다.

2. URL 설정 개선

2.1. URL 분리

현재 모든 URL 라우팅 설정을 `MBIT/urls.py` 에 설정하였습니다. 하지만 프로젝트에 앱이 많아지면 URL 설정을 한곳에 모으면 관리하기 힘들어집니다. 그래서 URL 설정을 앱별로 나누어서 관리 하는 것이 좋습니다. `main` 앱과 관련된 URL을 `main/urls.py` 에 설정해보겠습니다.

1. 먼저 `main` 앱 폴더 아래에 `urls.py` 파일을 만듭니다.



2. `urls.py` 의 내용을 다음과 같이 작성합니다.

```
from django.urls import path
from . import views

url_patterns = [
    path('', views.index),
    path('form/', views.form),
    path('submit/', views.submit),
    path('result/<int:developer_id>/', views.result),
]
```

3. `MBIT/urls.py` 를 다음과 같이 수정합니다.

```
from django.contrib import admin
from django.urls import path, include
from main import views

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
]
```

[프로젝트 URL]/~ 로 요청이 오면 `main` 앱의 `urls.py` 에서 URL 매칭을 찾으라는 설정입니다.

2.2. URL name

여기에서 리다이렉트를 사용하기 위해서 `/result/` URL을 `/submit/` 과 `/result/` 로 분리 시키기 위해서 기존의 `/result/` URL을 `/submit/` 으로 바꾸었습니다. 이때 `/result/` URL을 사용하고 있는 모든 곳에서 바꾸어주어야 했습니다. 예를 들어, `form.html` 에서 `<form>` 태그의 `action` 속성을 `/result/` 에서 `/submit/` 으로 바꾸어 주어야 했습니다. 만약 기존에 `/result/` URL을 사용하던 곳이 수 없이 많다고 가정해 봅시다. 그 전부를 `/submit/` 으로 바꾸는 것은 정말로 비효율적입니다. 이와 같은 상황을 우리는 '하드코딩'이라고 표현합니다. 이를 해결할 수 있는 방법이 URL name입니다.

간단히 말하면 URL을 변수(name)에 넣어서 사용하는 겁니다. 특정 URL에 우리가 알 수 있는 변수 (name)을 지정하고, 이 URL을 사용해야 하는 곳에서 URL을 직접 사용하는 대신에 이 변수(name)를 사용합니다.

1. `main/urls.py` 에서 각 path에 `name` 인자를 추가합니다.

```
urlpatterns = [
    path('/', views.index, name='index'),
    path('form/', views.form, name='form'),
    path('submit/', views.submit, name='submit'),
    path('result/<int:developer_id>/', views.result, name='result'),
]
```

2. `main/urls.py` 에 `app_name` 이라는 변수를 정의합니다.

```
app_name = 'main'

urlpatterns = [
    ...
]
```

`app_name` 은 URL namespace(이름공간)으로 여러 앱들마다 동일한 URL name을 가질 때 구별하기 위함입니다. 예를 들어, main 앱 뿐만 아니라 다른 앱에도 `submit` 이라는 URL name이 존재하면 `app_name:submit` 형식으로 URL name 앞에 namespace를 붙여서 구별할 수 있습니다.

3. 템플릿에서 URL을 사용한 부분들을 URL 태그 `{% url '[URL name]' %}` 를 사용합니다.

- index.html의 '시작하기' 버튼

```
<a href="{% url 'main:form' %}">
    <button class="start" type="button">시작하기</button>
</a>
```

- form.html의 `<form>` 태그의 `action` 속성

```
<form id="form" action="{% url 'main:submit' %}" method="post">
```

- result.html의 '테스트 다시 하기' 버튼

```
<a href="{% url 'main:index' %}">
    <button type="button">테스트 다시 하기</button>
</a>
```

4. `views.py`에서 `submit` 함수의 경우 `redirect`를 사용하기 위해 URL을 사용한 적이 있습니다. 이 또한 URL name으로 표현할 수 있습니다.

Python ▾

```
def form(request):
    ...
    return redirect('main:result', developer_id=best_developer_id)
```

`redirect`의 `developer_id`라는 키워드 인자가 URL 인자 값으로 들어갑니다.

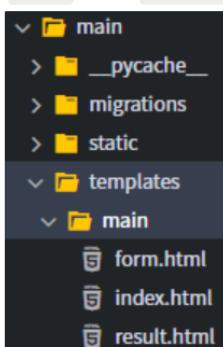
이제 URL을 변경하고 싶을 때 `urls.py`의 URL만 변경시키면 해당 URL name을 사용하는 곳은 자동으로 반영이 됩니다.

3. 템플릿 개선

3.1. 템플릿 이름 구별

지금은 앱이 `main` 하나지만, 여러개의 앱이 있을 때는 템플릿의 이름이 겹칠 수 있는 문제가 발생합니다. 예를 들어, `main` 외의 다른 앱들도 `form.html`이라는 이름의 템플릿을 가지고 싶을 수 있습니다. 하지만 똑같이 `form.html`이라는 이름으로 만들면 개발자들의 입장에서는 어느 앱의 `form.html`을 가리키는 것인지 헷갈리게 됩니다. 그래서 각 앱의 `templates` 폴더 아래에 각 앱의 이름의 폴더를 하나 더 만들고 그 안에 템플릿 파일들을 저장합니다. 그러면 각 템플릿의 이름은 `앱이름/템플릿명.html` 형태가 되어서 똑같은 템플릿 명이더라도 `앱이름`으로 구별할 수 있게 됩니다.

1. `main` 앱의 `templates` 폴더 아래에 `main` 폴더를 만들고 템플릿 파일들을 이 폴더에 옮깁니다.



2. `views.py`에서 `render` 함수에 쓰인 템플릿 이름들을 수정합니다.

- `index` 함수

```

def index(request):
    ...
    return render(request, 'main/index.html', context=context)
  
```

- `form` 함수

```

def form(request):
    ...
    return render(request, 'main/form.html', context)
  
```

- `result` 함수

```
def result(request, developer_id):  
    ...  
    return render(request, 'main/result.html', context=context)
```

3.2. 템플릿 상속

지금까지 작성한 템플릿들의 문제점은 반복적인 코드가 많다는 것입니다. 각 템플릿마다 다음과 같은 공통의 코드를 가지고 있습니다.

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    ...  
    <title>나의 개발 유형찾기</title>  
</head>  
<body>  
    ...  
</body>  
</html>
```

이러한 경우 공통의 틀이 되는 코드는 하나의 파일에 한 번만 작성하고, 이를 필요로 하는 템플릿에서 가져다 쓸 수 있도록 만들 수 있습니다. 이를 **템플릿 상속**이라고 합니다.

1. `main/templates/main/` 폴더에 공통의 템플릿을 작성할 `base.html` 파일을 만들고, 다음과 같이 작성합니다.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    {% block head %}
    {% endblock head %}
    <title>Document</title>
</head>
<body>
    {% block body %}
    {% endblock body %}

    {% block js %}
    {% endblock js %}
</body>
</html>
```

base.html

이 템플릿은 가장 큰 틀을 제공하고, 이 틀의 각 `{% block [block이름] %}`이라는 공간에 각 템플릿들의 고유한 코드가 들어갑니다.

2. `index.html`을 다음과 같이 수정합니다.

```
base.html
{% extends 'main/base.html' %}
{% load static %}

{% block head %}
    <link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/style.css' %}">
{% endblock head %}

{% block body %}
    <section id="main_contents">
        <div class="wrapper">
            ...
        </div>
    </section>
{% endblock body %}
```

index.html

- `{% extends 'base.html' %}` : 부모 템플릿으로 `base.html` 을 사용하겠다는 뜻입니다.
- `{% block [blockname] %} ~ {% endblock [blockname] %}`
`base.html` 의 각 `block` 부분에 삽입될 코드를 작성합니다.

3. 동일한 방식으로 `form.html`, `result.html` 도 수정합니다.

```

{% extends 'main/base.html' %}

{% load static %}

{% block head %}
<link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'css/form.css' %}">
<script src="https://code.jquery.com/jquery-3.5.1.js"></script>
{% endblock head %}

{% block body %}
<section id="survey">
<div class="wrapper">
    ...
</div>
</section>
{% endblock body %}

{% block js %}
<script type="text/javascript" src="{% static 'js/form.js' %}"></script>
{% endblock js %}

```

form.html

HTML ▾

```
{% extends 'main/base.html' %}

{% load static %}

{% block head %}
<meta property="og:title" content="나의 개발 유형은?" />
<meta property="og:image" content="" />
<meta property="og:url" content="" />
<meta property="og:description" content="나에게 꼭 맞는 개발 유형은 무엇일까?" />

<link rel="stylesheet" href="{% static 'css/reset.css' %}">
<link rel="stylesheet" href="{% static 'css/result.css' %}">
<script src="https://code.jquery.com/jquery-3.6.0.js"></script>
{% endblock head %}

{% block body %}
<section id="main_contents">
  <div class="wrapper">
    ...
  </div>
</section>
{% endblock body %}

{% block js %}
<script type="text/javascript" src="{% static 'js/result.js' %}"></script>
<script src="https://developers.kakao.com/sdk/js/kakao.js"></script>
{% endblock js %}
```

result.html



[심화] 결과 페이지 완성하기

1. 개발자 유형별 데이터 추가

1.1 JSON

2.1 JSONField

2. 결과 페이지 템플릿에 데이터 추가

2.1 title

2.2 features

2.3 descriptions

2.4 languages

2.5 lectures

3. static 폴더 설정

4. 다른 결과 알아보기

4.1 URL

4.2 View

4.3 Template

5. 카카오톡 공유하기 기능 구현

1. 개발자 유형별 데이터 추가

현재까지 만든 결과 페이지에는 개발자 유형별 설명이 포함되어 있지 않습니다. 이제 각 개발자 별로 상세 설명을 추가하겠습니다.

1.1 JSON

각 개발자 유형별 설명 데이터를 DB에 JSON 형식으로 넣을 것입니다. JSON(JavaScript Object Notation)이란 자바스크립트 언어에서의 객체 표기법으로 자세한 설명은 부록(용어집, 교양을 위한
얇은 지식)의 JSON 파트에서 확인하실 수 있습니다.

다음은 백엔드 개발자에 대한 설명 데이터의 JSON입니다.

```
{
  "title": "보이지 않는 것을 보는 당신은!",

  "features": [
    "어떤 일이든 안정적인 것이 우선임. 밴드 활동을 한다면, 화려한 보컬보다는 안정적인 연주를 이끌어나가는 베이스와 드럼을 맡는 편.",
    "주로 데이터를 다루는 데에 관심이 있음. 시각적으로 디자인을 하는 것에는 관심이 없고 귀찮음.",
    "논리적인 사고를 하는 것을 즐기는 스타일으로 추리소설, 방탈출 게임 등을 좋아하고 어려운 문제를 해결하는데 성취감을 느낌.",
    "효율적인 거 너무 좋아 최고야...! 목적지에 갈 때도 최단 경로 알아내서 그 길로 가는 편. 지하철 탈 때 아무 데서나 서 있지 않음. 빠른 환승 구역 앞에 서 있어야 마음이 안정됨."
  ],

  "descriptions": [
    "프론트 개발자가 사용자에게 보여지는 화면을 담당하는 개발자라면, 백엔드 개발자는 로그인, 추천 기능 등 서비스의 뒷부분 즉, 서버 설계를 담당하는 개발자입니다.",
    "클라이언트(사용자)와 데이터베이스(데이터가 모여있는 곳) 사이를 중계하는 서버를 개발합니다. 클라이언트가 요청하는 데이터를 가공하여 제공(예를 들어 넷플릭스에서 내가 원하는 영상 리스트업)하는 프로그램을 개발합니다."
  ],

  "languages": {
    "list": [
      {
        "name": "python",
        "img": "img/lang/python.png"
      },
      {
        "name": "Java",
        "img": "img/lang/java.png"
      },
      {
        "name": "JavaScript",
        "img": "img/lang/js.png"
      }
    ],
    "comments": [
      "python을 배우신 후에는, Django 또는 Flask 웹 프레임워크를 배우시면 됩니다.",
      "Java를 배우신 후에는, Spring과 Spring Boot 웹 프레임워크를 공부하시면 좋습니다.",
      "JavaScript를 배우신 경우 Node.js라는 런타임 환경과 그 위에서 동작하는 Express.js 등의 웹 프레임워크를 공부하시면 됩니다."
    ]
  }
}
```

```
"lectures": [
  {
    "name": "Python 부트캠프",
    "img": "img/lec/lec_python_bootcamp.png",
    "url": "https://www.inflearn.com/course/파이썬-부트캠프"
  },
  {
    "name": "코알못에서 웹서비스 런칭까지 : 제주 코딩 베이스캠프(Django)",
    "img": "img/lec/lec_web_fullstack.png",
    "url": "https://www.inflearn.com/course/web_fullstack"
  },
  {
    "name": "Django Mini project BEST 3",
    "img": "img/lec/lec_django_mini.png",
    "url": "https://www.inflearn.com/course/Django-미니프로젝트강좌"
  }
]
```

- **title**

다음 사진처럼 개발자 유형 이름 위에 붙는 제목입니다.

보이지 않는 것을 보는 당신은!

백엔드 개발자

- **features**

개발자 유형의 특징에 대한 설명입니다.

나와 맞는 개발 유형은 백엔드 개발자?!

- 어떤 일이든 안정적인 것이 우선임. 밴드 활동을 한다면, 화려한 보컬보다는 안정적인 연주를 이끌어나가는 베이스와 드럼을 맡는 편.
- 주로 데이터를 다루는 데에 관심이 있음. 시각적으로 디자인을 하는 것에는 관심이 없고 귀찮음.
- 논리적인 사고를 하는 것을 즐기는 스타일으로 추리 소설, 방탈출 게임 등을 좋아하고 어려운 문제를 해결하는데 성취감을 느낌.
- 효율적인 거 너무 좋아 최고야...! 목적지에 갈 때도 최단 경로 알아내서 그 길로 가는 편. 지하철 탈 때 아무 데서나 서 있지 않음. 빠른 환승 구역 앞에 서 있어야 마음이 안정됨.

- **descriptions**

개발자 유형마다 하는 일을 설명합니다.

백엔드 개발자(이)가 뭐지?

- 프론트 개발자가 사용자에게 보여지는 화면을 담당하는 개발자라면, 백엔드 개발자는 로그인, 추천 기능 등 서비스의 뒷부분 즉, 서버 설계를 담당하는 개발자입니다.
- 클라이언트(사용자)와 데이터베이스(데이터가 모여있는 곳) 사이를 중계하는 서버를 개발합니다. 클라이언트가 요청하는 데이터를 가공하여 제공(예를 들어 넷플릭스에서 내가 원하는 영상 리스트업)하는 프로그램을 개발합니다.

- **languages**

해당 개발자가 되기 위해 공부해야 할 프로그래밍 언어에 대한 설명입니다.

그래서 어떤 언어부터 공부해야 할까?



python



Java



JavaScript

- python을 배우신 후에는, Django 또는 Flask 웹 프레임워크를 배우시면 됩니다.
- Java를 배우신 후에는, Spring과 Spring Boot 웹 프레임워크를 공부하시면 좋습니다.
- JavaScript를 배우신 경우 Node.js라는 런타임 환경과 그 위에서 동작하는 Express.js 등의 웹 프레임워크를 공부하시면 됩니다.

- lectures

해당 개발자가 되기 위해 수강할만한 강의들의 목록입니다.

강의 추천

백엔드 개발자 공부를 시작하기 좋은 강의를 추천해 드릴게요!



Python 부트캠프

강의 보러가기

2.1 JSONField

이러한 JSON 데이터를 DB에 넣기 위해서 `Developer` 모델에 `JSONField` 를 추가합니다.

```
class Developer(models.Model):
    name = models.CharField(max_length=50)
    count = models.IntegerField(default=0)
    data = models.JSONField()

    def __str__(self):
        return self.name
```

모델이 변경되었으니 `makemigrations`, `migrate` 명령을 수행합니다. 중간에 data 필드가 다른 데이터에 비어있다는 문구가 뜰텐데 1을 입력하고, 다시 1을 입력해서 모든 값을 다 1로 채워줍니다.

```
$ python manage.py makemigrations
```

```
$ python manage.py migrate
```

```
root@goorm:/workspace/testmbit# source venv/bin/activate
(venv) root@goorm:/workspace/testmbit# python manage.py makemigrations
You are trying to add a non-nullable field 'data' to developer without a default; we can't do that (the database needs something to populate existing rows).
Please select a fix:
 1) Provide a one-off default now (will be set on all existing rows with a null value for this column)
 2) Quit, and let me add a default in models.py
Select an option: 1
Please enter the default value now, as valid Python
The datetime and django.utils.timezone modules are available, so you can do e.g. timezone.now
Type 'exit' to exit this prompt
>>> 1
Migrations for 'main':
  main/migrations/0002_developer_data.py
    - Add field data to developer
(venv) root@goorm:/workspace/testmbit# python manage.py migrate
(venv) root@goorm:/workspace/testmbit# python manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, main, sessions
Running migrations:
  Applying main.0002_developer_data... OK
```

데이터를 일일이 타이핑해서 집어 넣기는 힘들겠죠? 그래서 업데이트된 새로운 `data.json`을 준비했습니다.

 `data.json` 28.4KB

내용을 복사 붙여넣기 하시거나, 파일 자체를 교체하신 뒤에 Database Seeding 파트에서 배운 `loaddata` 명령을 수행합니다.

```
$ python manage.py loaddata data.json
```

shell_plus를 이용해 데이터가 잘 들어갔는지 확인 할 수 있습니다.

```
$ python manage.py shell_plus
```

```
>>> backend = Developer.objects.get(pk=1)
>>> backend
<Developer: 백엔드 개발자>
>>> backend.data
{'title': '보이지 않는 것을 보는 당신은!', 'features': ['어떤 일이든 안정적인 것이 우선임. 밴드 활동을 한다면, 화려한 보컬보다는 안정적인 연주를 이끌어나가는 베이스와 드럼을 맡는 편.', '주로 데이터를 다루는 데에 관심이 있음. 시각적으로 디자인을 하는 것에는 관심이 없고 커玷음.', '논리적인 사고를 하는 것을 즐기는 스타일으로 추리소설, 방탈출 게임 등을 좋아하고 어려운 문제를 해결하는데 성취감을 느낌.', '효율적인 거 너무 좋아 최고야...! 목적지에 갈 때도 최단 경로 알아내서 그 길로 가는 편. 지하철 탈 때 아무 데서나 서 있지 않음. 빠른 환승 구역 앞에 서 있어야 마음이 안정됨.'], 'descriptions': ['프론트 개발자가 사용자에게 보여지는 화면을 담당하는 개발자라면, 백엔드 개발자는 로그인, 추천 기능 등 서비스의 뒷부분 즉, 서버 설계를 담당하는 개발자입니다.', '클라이언트(사용자)와 데이터베이스(데이터가 모여있는 곳) 사이를 중계하는 서버를 개발합니다. 클라이언트가 요청하는 데이터를 가공하여 제공(예를 들어 넷플릭스에서 내가 원하는 영상 리스트 얻)하는 프로그램을 개발합니다.'], 'languages': {'list': [{'name': 'python', 'img': 'img/lang/python.png'}, {'name': 'Java', 'img': 'img/lang/java.png'}, {'name': 'JavaScript', 'img': 'img/lang/js.png'}]}, 'comments': ['python을 배우신 후에는, Django 또는 Flask 웹 프레임워크를 배우시면 됩니다.', 'Java를 배우신 후에는, Spring과 Spring Boot 웹 프레임워크를 공부하시면 좋습니다.', 'JavaScript를 배우신 경우 Node.js라는 런타임 환경과 그 위에서 동작하는 Express.js 등의 웹 프레임워크를 공부하시면 됩니다.']},
[{'name': 'Python 부트캠프', 'img': 'img/lec/lec_python_bootcamp.png', 'url': 'https://www.inflearn.com/course/파이썬-부트캠프'}, {'name': '코알못에서 웹서비스 런칭까지 : 제주 코딩 베이스캠프(Django)', 'img': 'img/lec/lec_web_fullstack.png', 'url': 'https://www.inflearn.com/course/web_fullstack'}, {'name': 'Django Mini project BEST 3', 'img': 'img/lec/lec_django_mini.png', 'url': 'https://www.inflearn.com/course/Django-미니프로젝트강좌'}]}
```

`backend.data` 의 타입을 확인해보면 딕셔너리임을 알 수 있습니다.

```
>>> type(backend.data)
<class 'dict'>
```

이렇게 Python에서는 주로 JSON 데이터를 dictionary로 변환하여 표현합니다.

2. 결과 페이지 템플릿에 데이터 추가

이제 결과 페이지 템플릿 `result.html`을 추가된 데이터들이 보여지도록 수정하겠습니다.

2.1 title

```
...
<div class="titles">
    <h3>{{ developer.data.title }}</h3>
    <h1>{{ developer.name }}</h1>
</div>
....
```

2.2 features

```
<div class="explain">
    <h3 class="title">나와 맞는 개발 유형은 {{ developer.name }}?!</h3>
    <ul>
        {% for feature in developer.data.features %}
        <li>
            {{ feature }}
        </li>
        {% endfor %}
    </ul>
</div>
```

2.3 descriptions

```
<div class="explain">
    <h3 class="title">{{ developer.name }}(이)가 뭐지?</h3>
    <ul>
        {% for description in developer.data.descriptions %}
        <li>
            {{ description }}
        </li>
        {% endfor %}
    </ul>
</div>
```

2.4 languages

```
<div class="explain">
    <h3 class="title">그래서 어떤 언어부터 공부해야 할까?</h3>
    <ul class="language_lists">
        {% for language in developer.data.languages.list %}
        <li>
            <div class="img_wrap">
                
            </div>
            <h3>{{ language.name }}</h3>
        </li>
        {% endfor %}
    </ul>
    {% if data.languages.comments %}
    <ul>
        {% for comment in developer.data.languages.comments %}
        <li>
            {{ comment }}
        </li>
        {% endfor %}
    </ul>
    {% endif %}
</div>
```

2.5 lectures

```
<div class="lectures">
    <h3 class="title">강의 추천</h3>
    <p>{{ developer.name }} 공부를 시작하기 좋은 강의를 추천해 드릴게요!</p>
    <ul>
        {% for lecture in data.lectures %}
        <li>
            <a href="{{ lecture.url }}" target="_blank">
                
                <h3>{{ lecture.name }}</h3>
                <button type="button">강의 보러가기</button>
            </a>
        </li>
        {% endfor %}
    </ul>
</div>
```

3. static 폴더 설정

위와 같이 템플릿을 올바르게 수정하여도 아직 노출 되지 않는 이미지들이 있을 것입니다. 왜냐하면 필요한 이미지들을 static 폴더에 아직 업데이트를 하지 않았기 때문입니다.

⑩ <http://paullab.co.kr/static.zip>

위의 파일을 다운 받아서 **staticfiles** 폴더를 교체해주세요. 필요한 이미지가 많아졌기 때문에 **img** 폴더 하위에 추가적으로 분류하였습니다.

- **lang** 폴더(프로그래밍 언어 이미지)
- **lec** 폴더(강의 이미지)

이제 문제 없이 이미지들이 잘 노출될 것입니다.

 영상에도 static 폴더에 업로드를 하였다가 작동이 되지 않아 staticfiles에 업로드를 하였는데요. static 폴더가 아니라 staticfiles에 이미지를 업로드 해주셔야 합니다.

4. 다른 결과 알아보기

마지막으로 아직 구현되지 않은 부분이 있습니다. 바로 **다른 결과 알아보기** 입니다.

다른 사람들은 어떤 유형일까?

다른 결과 알아보기

이를 구현하기 위해서 `/all_results/` 라는 경로에 원하는 유형을 골라 볼 수 있는 페이지를 만들 것입니다.

여기서도 마찬가지로 URL → View → Template 순으로 개발합니다.

4.1 URL

```
urlpatterns = [
    path('', views.index, name='index'),
    path('form/', views.form, name='form'),
    path('submit/', views.submit, name='submit'),
    path('result/<int:developer_id>/', views.result, name='result'),
    path('all_results/', views.all_results, name='all_results'),
]
```

main/urls.py

4.2 View

views.py에 `all_results` 뷰 함수를 정의합니다. `all_results.html`은 다음에 만들 템플릿 파일입니다.

```
def all_results(request):
    return render(request, 'main/all_results.html')
```

main/views.py

4.3 Template

`all_results.html` 템플릿을 다음과 같이 작성합니다.

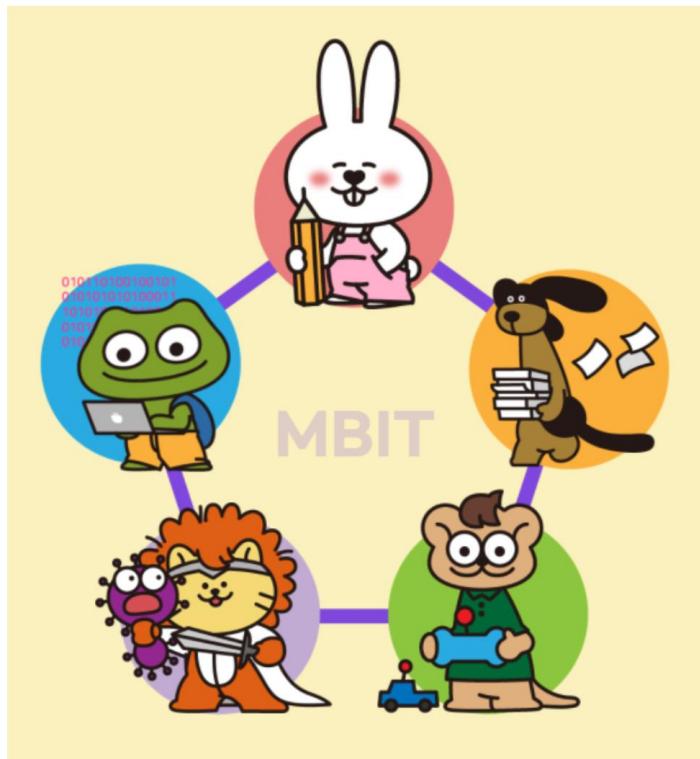
```
{% extends 'main/base.html' %}
{% load static %}

{% block head %}
    <link rel="shortcut icon" href="{% static 'img/favicon.ico' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/reset.css' %}">
    <link rel="stylesheet" type="text/css" href="{% static 'css/all_results.css' %}">
    <title>나의 개발 유형찾기</title>
{% endblock head %}

{% block body %}
    <section id="main_contents">
        <div class="wrapper">
            <h1>MBIT</h1>
            <div class="mbit">
                <div class="frontend">
                    <a href="/result/2/"></a>
                </div>
                <div class="backend">
                    <a href="/result/1/"></a>
                </div>
                <div class="security">
                    <a href="/result/4/"></a>
                </div>
                <div class="game">
                    <a href="/result/5/"></a>
                </div>
            </div>
        </div>
    </section>
{% endblock %}
```

```
<div class="data">
    <a href="/result/3/"></a>
</div>
</div>
</div>
</section>
{% endblock body %}
```

이제 `/all_results/` 경로로 접속하면 개발 유형을 선택할 수 있는 페이지가 보입니다. 각 캐릭터를 클릭하면 해당 개발 유형의 페이지로 이동됩니다.



5. 카카오톡 공유하기 기능 구현

- result.html에 js 순서를 아래와 같이 바꿔주세요.

```
137  {% block js %}  
138  <script src="https://developers.kakao.com/sdk/js/kakao.js"></script>  
139  <script src="https://code.jquery.com/jquery-3.6.0.js"></script>  
140  <script type="text/javascript" src="{% static 'js/result.js' %}"></script>  
141  {% endblock js %}
```

- result.js 파일 안에 있는 url을 모두 현재 자신의 url로 바꿔주세요.

```
Kakao.Link.sendDefault({  
    objectType: 'feed',  
    content: {  
        title: '나의 개발 유형은?',  
        description: '나에게 꼭 맞는 개발 유형을 알아보자!!!',  
        imageUrl: 'https://testmbit-abrgy.run.goorm.io/static/im  
        link: {  
            mobileWebUrl: 'https://testmbit-abrgy.run.goorm.io',  
            webUrl: 'https://testmbit-abrgy.run.goorm.io',  
        },  
    },  
    social: {
```

- 카카오 developers에서 여러분의 web을 등록해주세요. (강의 영상 참고하세요.)

The screenshot shows the Kakao Developers platform interface. At the top, there's a navigation bar with 'kakao developers' on the left and '내 애플리케이션' (My Application) on the right. Below the navigation, a sidebar on the left lists various settings like 팀 관리, 제품 설정, 카카오 로그인, etc. The main content area shows an application named 'mbit' (ID: 550400, OWNER: Web). It has sections for Android, iOS, and Web. Under the Web section, the '사이트 도메인' field is set to 'https://testmbit-abrgy.run.goorm.io'. There are '삭제' and '수정' buttons at the bottom of this section.

4. kakao 키가 아래 제대로 입력되어 있는지 확인해주세요.

kakao developers

내 애플리케이션 > 앱 설정 > 요약 정보

The screenshot shows the Kakao Developers platform interface. On the left, there's a sidebar with various settings like App Settings, General, Business, API Key, Platforms, Team Management, Product Settings, Kakao Login, and Accessibility. The 'API Key' section is highlighted. At the top right, it shows an APP named 'mbit' with ID 550400, OWNER, and Web. Below the header, there's a table with four rows: 'Native App Key' (empty), 'REST API Key' (highlighted with a red box), 'JavaScript Key' (empty), and 'Admin Key' (empty).

```
4 ////////////////////////////////////////////////////////////////// 카카오 공유 코드 //////////////////////////////////////////////////////////////////  
5  
6 const kakaoShare = document.querySelector('.kakao_share');  
7 Kakao.init('6d [REDACTED]');  
8 Kakao.isInitialized();
```



[심화] 커스텀 도메인

1. 커스텀 도메인

1.1. 구름 IDE에서 커스텀 도메인 설정 방법

1. 커스텀 도메인

구름 IDE에서는 `*.run.goorm.io` 형태의 무료 도메인을 제공하지만, 실제로 서비스를 하기 위해서는 개발자가 원하는 브랜드명의 도메인을 사용하는 것이 좋습니다. 이번 장에서는 cafe24에서 구입한 커스텀 도메인, `weniv.co.kr`을 구름 IDE에 적용시켜보겠습니다. 단, 구름 IDE에서 커스텀 도메인은 Premium에서만 제공합니다.

1개월마다 6개월마다 -10% 1년마다 -20%			
FREE	STUDENT	BASIC	Premium
₩0 /월 TOTAL ₩0 goormIDE를 마음껏 이용해보세요.	₩7,000 /월 TOTAL ₩7,000 1개월 요금제 결제 시	₩10,000 /월 TOTAL ₩10,000 1개월 요금제 결제 시	₩18,000 /월 TOTAL ₩18,000 1개월 요금제 결제 시
Includes: <ul style="list-style-type: none"> <input checked="" type="checkbox"/> CPU 낮음 <input checked="" type="checkbox"/> 메모리 1GB <input checked="" type="checkbox"/> 저장공간 10GB <input checked="" type="checkbox"/> 컨테이너 생성 5개 <input checked="" type="checkbox"/> 공유 인원 5명/컨테이너 <input checked="" type="checkbox"/> 게스트 3명/컨테이너 <input checked="" type="checkbox"/> 도메인 생성 3개/컨테이너 <input checked="" type="checkbox"/> 컨테이너 동시 실행 1개 <input checked="" type="checkbox"/> 이미지 기능 <input checked="" type="checkbox"/> 항상 켜두기 기능 <input checked="" type="checkbox"/> 사용자 도메인 연결 	Includes: <ul style="list-style-type: none"> <input checked="" type="checkbox"/> CPU 높음 <input checked="" type="checkbox"/> 메모리 2GB <input checked="" type="checkbox"/> 저장공간 10GB <input checked="" type="checkbox"/> 컨테이너 생성 10개 <input checked="" type="checkbox"/> 공유 인원 10명/컨테이너 <input checked="" type="checkbox"/> 게스트 5명/컨테이너 <input checked="" type="checkbox"/> 도메인 생성 5개/컨테이너 <input checked="" type="checkbox"/> 컨테이너 동시 실행 2개 <input checked="" type="checkbox"/> 이미지 기능 3개 <input checked="" type="checkbox"/> 항상 켜두기 기능 <input checked="" type="checkbox"/> 사용자 도메인 연결 	Includes: <ul style="list-style-type: none"> <input checked="" type="checkbox"/> CPU 매우 높음 <input checked="" type="checkbox"/> 메모리 4GB <input checked="" type="checkbox"/> 저장공간 20GB <input checked="" type="checkbox"/> 컨테이너 생성 10개 <input checked="" type="checkbox"/> 공유 인원 10명/컨테이너 <input checked="" type="checkbox"/> 게스트 5명/컨테이너 <input checked="" type="checkbox"/> 도메인 생성 5개/컨테이너 <input checked="" type="checkbox"/> 컨테이너 동시 실행 2개 <input checked="" type="checkbox"/> 이미지 기능 3개 <input checked="" type="checkbox"/> 항상 켜두기 기능 <input checked="" type="checkbox"/> 사용자 도메인 연결 	Includes: <ul style="list-style-type: none"> <input checked="" type="checkbox"/> CPU 매우 높음 <input checked="" type="checkbox"/> 메모리 4GB <input checked="" type="checkbox"/> 저장공간 40GB <input checked="" type="checkbox"/> 컨테이너 생성 20개 <input checked="" type="checkbox"/> 공유 인원 20명/컨테이너 <input checked="" type="checkbox"/> 게스트 10명/컨테이너 <input checked="" type="checkbox"/> 도메인 생성 10개/컨테이너 <input checked="" type="checkbox"/> 컨테이너 동시 실행 4개 <input checked="" type="checkbox"/> 이미지 기능 5개 <input checked="" type="checkbox"/> 항상 켜두기 기능 1개 <input checked="" type="checkbox"/> 사용자 도메인 연결 2개
다운그레이드	인증하기	정기결제	✓ 이용중인 플랜

구름IDE 가격정책 (2021.01.13 기준)

1.1. 구름 IDE에서 커스텀 도메인 설정 방법

1. 별칭(CNAME) 설정

커스텀 도메인 `weniv.co.kr` 아래에 서브 도메인으로 `mbit.weniv.co.kr`을 설정하고 실제 값은 `domain.run.goorm.io`로 설정합니다. 여기서 `domain`은 사용자가 설정한 도메인 명이 아니라 그냥 `domain` 그대로 작성하면 됩니다.

The screenshot shows the 'CNAME' configuration section. At the top, there's a note: '선택한 도메인의 실제 별칭을' with buttons for '수정' (Edit) and '삭제' (Delete). To the right is a 'CNAME 추가' (Add CNAME) button. Below this, there's a table with two rows:

선택	도메인 별칭	실제 도메인명
<input checked="" type="radio"/>	<code>*.weniv.co.kr</code>	<code>weniv.co.kr</code>
<input type="radio"/>	<code>mbit.weniv.co.kr</code>	<code>domain.run.goorm.io</code>

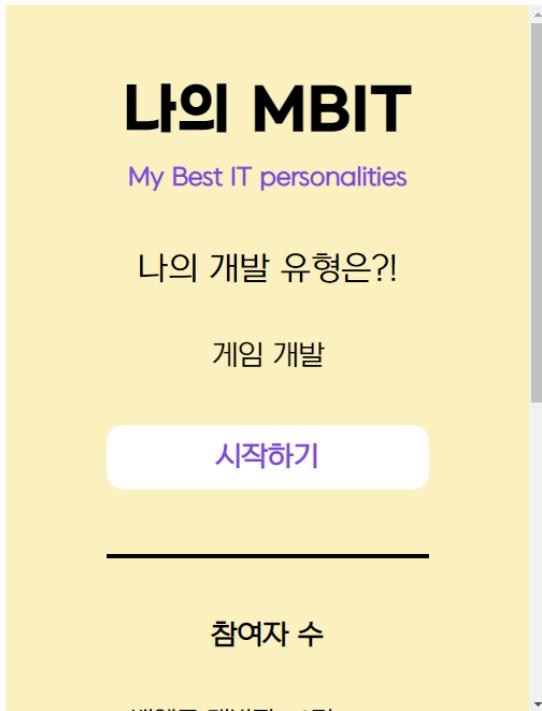
2. 구름 IDE에 해당 서브 도메인 `mbit.weniv.co.kr`을 등록합니다. 커스텀 도메인 등록은 IDE 내에서 불가능하고, 컨테이너 설정 페이지에서 가능합니다.

The screenshot shows the 'URL/Port' configuration section. At the top, there's a 'URL/Port' label and a '+ 추가' (Add) button. Below this, there's a table with two rows:

URL	Port	...
<code>https://jejucodingcamp.run.goorm.io</code>	81	...
<code>https://mbit.weniv.co.kr</code>	80	...

[참고] 구름 IDE 설명서

3. 이제 등록한 도메인의 해당 포트로 애플리케이션을 실행하고 해당 도메인으로 접속할 수 있습니다.





영상 강의 압축 파일

영상 강의의 html, css, js, img, py 파일을 압축하여 올려드립니다.

allfile.zip

Last modified by 제주코딩베이스캠프 4 days ago

[drive.google.com](#)

초판 1쇄 발행 | 2021년 2월 19일

지은이 | 이호준 김혜원 김유진 차경림 김진 현지연 정승한

편집 | 이호준 차경림 현지연

총괄 | 이호준

펴낸곳 | 사도출판

주소 | 제주특별자치도 제주시 동광로 137 대동빌딩 4층

표지디자인 | 차경림

홈페이지 | <http://www.paullab.co.kr>

E-mail | paul-lab@naver.com

Copy right © 2021 by. 사도출판

이 책의 저작권은 사도출판에 있습니다. 저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금합니다.

이 책에 대한 의견을 주시거나 오탈자 및 잘못된 내용의 수정 정보는 사도출판의 이메일로 연락을 주시기 바랍니다.

