

Hello,

KDT 웹 개발자 양성 프로젝트

1기! 43<sup>th</sup>

with





# Redux 세팅!



```
import { configureStore } from '@reduxjs/toolkit';
import rootReducer from './store';
import { Provider } from 'react-redux';

const reduxDevTool =
  window.__REDUX_DEVTOOLS_EXTENSION__ && window.__REDUX_DEVTOOLS_EXTENSION__();

const store = configureStore({ reducer: rootReducer }, reduxDevTool);
console.log(store.getState());

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <Provider store={store}>
    <App />
  </Provider>
);
```

Src/index.js

- Provider 컴포넌트로 App 컴포넌트 감싸기 + store 설정



# rootReducer

## 설정

```
import { combineReducers } from 'redux';
import mbti from './modules/mbti';

export default combineReducers({
  mbti,
});
```

Src/store/index.js



- 추후 모듈이 추가 되었을 때, 이런 구조를 만들어 놓으면 편리 합니다!
- 새로운 SPA 가 추가될 경우 보통 Redux 모듈이 추가되고, 해당 SPA 는 라우팅으로 구현을 합니다!



# mbti store

## 설정

```
// 초기 상태 설정
const initState = {
  mbtiResult: '',
  page: 0, // 0: 인트로 페이지, 1 ~ n: 선택 페이지, n+1: 결과 페이지
  survey: '질문 목록',
  explanation: '결과에 대한 설명'
};
```

Src/store/modules/mbti.js





```
survey: [
  {
    question:
      '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n퇴근 시간에 나는?',
    answer: [
      {
        text: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다',
        result: 'E',
      },
      {
        text: '1년 전에 알려줬어도 안갔을 건데 뭐... 더 빠르게 집으로 간다',
        result: 'I',
      },
    ],
  },
  {
    question:
      '새로운 서비스 개발 중에, 동료가 새로 나온 신기술을 쓰는게 더 편할거라고 추천을 해준다!\n\n나의 선택은!?',
    answer: [
      {
        text: '원소리며, 그냥 하던 대로 개발하면 되는거지! 기존 생각대로 개발한다',
        result: 'S',
      },
      {
        text: '오호? 그런게 있어? 일단 구글을 찾아본다',
        result: 'N',
      },
    ],
  },
  {
    question:
      '서비스 출시 이틀 전 야근 시간, 갑자기 동료가 어!? 를 외쳤다!\n\n나의 선택은?',
    answer: [
      {
        text: '무슨 버그가 발생한 거지? 아마 DB 관련 버그가 아닐까? 빠르게 동료의 자리로 달려간다',
        result: 'T',
      },
      {
        text: '아... 내일도 야근 각이구나 ㅠㅠ! 일단 동료의 자리로 가 본다',
        result: 'F',
      },
    ],
  },
  {
    question:
      '팀장님이 xx씨 그전에 말한 기능 내일 오후까지 완료 부탁드립니다라고 말했다!\n\n나의 선택은?',
    answer: [
      {
        text: '일단 빠르게 개발 완료하고, 나머지 시간에 논다',
        result: 'J',
      },
      {
        text: '그거 내일 아침에 와서 개발해도 충분 하겠는데? 일단 논다',
        result: 'P',
      },
    ],
  },
],
```

Src/store/modules/mbti.js 중  
질문 항목





```
explanation: {
  ESTJ: {
    text: '무리한 개발 일정만 아니라면 일정을 철저하게 지킬 당신의 MBTI 는!',
    img: '/images/estj.jpg',
  },
  ISTJ: {
    text: '스스로 하고싶은 분야를 끝까지 파고 들어서 끝내 성공 시킬 당신의 MBTI 는!',
    img: '/images/istj.jpg',
  },
  ENTJ: {
    text: '미래의 능력 켜는 개발 팀장님으로 개발팀을 이끌 당신의 MBTI 는!',
    img: '/images/entj.jpg',
  },
  INTJ: {
    text: '혼자서 모든 것을 다 해내는 원맨 캐리의 표본! 당신의 MBTI 는!',
    img: '/images/intj.jpg',
  },
  ESFJ: {
    text: '개발팀의 분위기 메이커이자 아이디어 뱅크가 될 당신의 MBTI 는!',
    img: '/images/esfj.jpg',
  },
  ISFJ: {
    text: '개발팀의 마더 테레사, 고민 상담소 역할을 자처하는 당신의 MBTI 는!',
    img: '/images/isfj.jpg',
  },
  ENFJ: {
    text: '당신이 있는 팀은 언제나 올바른 곳을 향하고 있습니다! 팀원은 물론 팀의 방향을 챙기는 당신의 MBTI 는!',
    img: '/images/enfj.jpg',
  },
  INFJ: {
    text: '예리한 통찰력으로 모든 것을 내다보면서 완벽하게 개발을 할 당신의 MBTI 는!',
    img: '/images/infj.jpg',
  },
  ESTP: {
    text: '쿨하게 자신이 할 것을 하면서 논리적인 개발을 할 당신의 MBTI 는!',
    img: '/images/estp.jpg',
  },
  ISTP: {
    text: '단시간에도 효율적으로 개발하여 모든 것을 완성할 당신의 MBTI 는!',
    img: '/images/istp.jpg',
  },
  ENTP: {
    text: '스스로 흥미만 생긴다면 당장에 페이스북도 만들어 버릴 당신의 MBTI 는!',
    img: '/images/entp.jpg',
  },
  INTP: {
    text: '확실한 주관과 뛰어난 지능을 바탕으로 논리적 개발을 할 당신의 MBTI 는!',
    img: '/images/intp.jpg',
  },
  ESFP: {
    text: '개발팀의 에너지아저! 개발팀 특유의 서먹함을 깨는 당신! 당신의 MBTI 는!',
    img: '/images/esfp.jpg',
  },
  ISFP: {
    text: '뛰어난 호기심과 예술적 감각으로 개발팀의 부족함을 채워갈 당신! 당신의 MBTI 는!',
    img: '/images/isfp.jpg',
  },
  ENFP: {
    text: '자유로운 영혼으로 개발팀의 윤활유 및 활력소가 되어줄 당신의 MBTI 는!',
    img: '/images/enfp.jpg',
  },
  INFP: {
    text: '개발팀의 그 어떤 트러블도 당신 앞에서는 사르르 녹을뿐, 팀의 근간을 다져주는 당신의 MBTI 는!',
    img: '/images/infp.jpg',
  },
},
```

Src/store/modules/mbti.js 중  
결과 설명 항목



# Action Type 설정



```
// 액션 타입(문자열)
const CHECK = 'mbti/CHECK';
const NEXT = 'mbti/NEXT';
const RESET = 'mbti/RESET';
```

Src/store/modules/mbti.js



# Action

# 생성 함수 설정



```
// 액션 생성 함수
// payload -> 선택에 다른 결과 값 result 전달 필요
export function check(result) {
  return {
    type: CHECK,
    payload: { result },
  };
}

export function next() {
  return {
    type: NEXT,
  };
}

export function reset() {
  return {
    type: RESET,
  };
}
```

- 외부에서 사용해야 하므로 export 설정
- Type은 반드시 전달 필요
- 데이터가 필요한 경우 payload 에 담아서 전달

Src/store/modules/mbti.js



# Reducer

# 만들기



// 리듀서

```
export default function mbti(state = initState, action) {  
  switch (action.type) {  
    case CHECK:  
      return {  
        ...state,  
        mbtiResult: state.mbtiResult + action.payload.result,  
      };  
    case NEXT:  
      return {  
        ...state,  
        page: state.page + 1,  
      };  
    case RESET:  
      return {  
        ...state,  
        page: 0,  
        mbtiResult: '',  
      };  
    default:  
      return state;  
  }  
}
```

Src/store/modules/mbti.js



# Button

# 컴포넌트 제작





```
import styled from 'styled-components';

export default function Button({
  text,
  clickEvent,
  mainColor,
  subColor,
  hoverColor,
}) {
  return (
    <MyButton
      onClick={clickEvent}
      mainColor={mainColor}
      subColor={subColor}
      hoverColor={hoverColor}
    >
      {text}
    </MyButton>
  );
}
```

- Styled-components 에 props 를 전달하기 위한 props 전달!

Src/component/Button.js



```
const MyButton = styled.a`
  position: relative;
  display: inline-block;
  cursor: pointer;
  vertical-align: middle;
  text-decoration: none;
  line-height: 1.6em;
  font-size: 1.2em;
  padding: 1.25em 2em;
  background-color: ${(props) => props.mainColor};
  border: 2px solid ${(props) => props.subColor};
  border-radius: 0.75em;
  user-select: none;
  transition: transform 0.15s ease-out;
  transform-style: preserve-3d;
  margin-top: 1em;
```

- 전달 받은 props 의 사용
- Props 를 인자로 받아서 Styled 에 적용이 가능!

Src/component/Button.js



```
&::before {
  content: '';
  position: absolute;
  width: 100%;
  height: 100%;
  top: 0;
  right: 0;
  left: 0;
  right: 0;
  background: ${({props}) => props.subColor};
  border-radius: inherit;
  box-shadow: 0 0 0 2px ${({props}) => props.subColor};
  transform: translate3d(0, 0.75em, -1em);
}
&:hover {
  background: ${({props}) => props.hoverColor};
  transform: translateY(0.25em);
}
;
```

- SASS 와 마찬가지로 & 를 사용해 스스로 지칭 가능!
- 가상 요소, 클래스 선택자 사용 가능

Src/component/Button.js



# OrangeButton

## 으로 특수화!



```
import Button from './Button';

export default function OrangeButton({ text, clickEvent }) {
  return (
    <Button
      text={text}
      clickEvent={clickEvent}
      mainColor="#fae243"
      subColor="#fa9f1a"
      hoverColor="#faf000"
    />
  );
}
```

Src/component/OrangeButton.js



# Styled-components

## GlobalStyle



```
import { createGlobalStyle } from 'styled-components';

const GlobalStyle = createGlobalStyle`
  @font-face {
    font-family: 'ONE-Mobile-POP';
    src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2105_2@1.0/ONE-Mobile-POP.woff') format('woff');
    font-weight: normal;
    font-style: normal;
  }

  body {
    font-family: 'ONE-Mobile-POP', "Arial", sans-serif;
    padding-top: 1em;
    white-space: pre-wrap;
  }

  ul, ol {
    list-style: none;
    padding-left: 0px;
  }
`;
export default GlobalStyle;
```

Src/component/GlobalStyle.js



```
import GlobalStyle from '../components/GlobalStyle';
```

```
function App() {  
  return (  
    <>  
      <GlobalStyle />  
      <Main>  
        <Start />  
      </Main>  
    </>  
  );  
}
```

```
export default App;
```

Src/App.js





# 페이지 분기 처리



```
import { useSelector } from 'react-redux';
import styled from 'styled-components';
import GlobalStyle from './components/GlobalStyle';
import Start from './components/Start';

function App() {
  const page = useSelector((state) => state.mbti.page);

  return (
    <>
      <GlobalStyle />
      <Main>
        {page === 0 ? <Start /> : <Mbti />}
      </Main>
    </>
  );
}

export default App;
```

- Store 의 page 값이 0 이면 Start 컴포넌트를, 아닐 경우 MbtI 조사를 하는 MbtI 컴포넌트를 보여주기

Src/App.js



# Mbti 컴포넌트

## 제작

```
import { useSelector } from 'react-redux';
import styled from 'styled-components';
import SkyblueButton from './SkyblueButton';
```

```
export default function Mbti() {
  const survey = useSelector((state) => state.mbti.survey);
  const page = useSelector((state) => state.mbti.page);

  return (
    <>
      <SurveyQuestion>{survey[page - 1].question}</SurveyQuestion>
      <ul>
        {survey[page - 1].answer.map((el, index) => {
          return (
            <li key={index}>
              <SkyblueButton
                text={el.text}
              />
              {index === 0 && <Vs>VS</Vs>}
            </li>
          );
        })}
      </ul>
    </>
  );
}
```

- Store 에서 값 받아오기

- 설문 항목의 index 는 0 부터 시작이므로 page-1 의 인덱스로 접근하여 설문 항목 가져오기
- 선택지는 배열에 담겨 있으므로 map 메소드를 이용하여 각각의 버튼을 그려주기
- Vs 는 처음에 한번만 그려지면 되므로 map 의 index 를 이용하여 조건부 렌더링 처리



# 이벤트 핸들러에 액션 생성 함수 지정



```
import { useDispatch } from 'react-redux';
import { next } from '../store/modules/mbti';
```

```
export default function Start() {
  const dispatch = useDispatch();
```

```
  return (
```

```
    <>
```

```
    <Header>개발자 MBTI 조사</Header>
```

```
    <MainImg src="/images/main.jpg" alt="메인 이미지" />
```

```
    <SubHeader>
```

```
      개발자가 흔히 접하는 상황에 따라서 MBTI 를 알아 봅시다!
```

```
    </SubHeader>
```

```
    <OrangeButton text="테스트 시작" clickEvent={() => dispatch(next())} />
```

```
  </>
```

```
);
```

```
}
```

- useDispatch 혹은 dispatch 지정
- 테스트 시작 버튼 클릭 시, next() 액션 생성 함수를 dispatch 로 reducer 에 전달

Src/component/Start.js

```
export default function Mbti() {
  const survey = useSelector((state) => state.mbti.survey);
  const page = useSelector((state) => state.mbti.page);
  const dispatch = useDispatch();

  return (
    <>
      <SurveyQuestion>{survey[page - 1].question}</SurveyQuestion>
      <ul>
        {survey[page - 1].answer.map((el, index) => {
          return (
            <li key={index}>
              <SkyblueButton
                text={el.text}
                clickEvent={() => {
                  dispatch(next());
                }}
              />
              {index === 0 && <Vs>VS</Vs>}
            </li>
          );
        })}
      </ul>
    </>
  );
}
```

Src/component/Mbti.js



- 설문 선택 시, next() 액션 생성 함수를 dispatch 로 reducer 에 전달



# Progress Bar

# 만들기





```
import styled from 'styled-components';

export default function Progress({ page, maxPage }) {
  return (
    <MyProgress>
      <div>
        {page} / {maxPage}
      </div>
      <Fill>
        <Gauge percent={ (page / maxPage) * 100 }></Gauge>
      </Fill>
    </MyProgress>
  );
}
```

- % 로 게이지를 그릴 것이므로 %에 들어갈 숫자 값을 계산하여 전달

Src/component/Progress.js



```
const MyProgress = styled.div`
  margin-top: 3em;
`;

const Fill = styled.div`
  width: 100%;
  height: 10px;
  background-color: #777;
  margin-top: 1em;
  text-align: left;
`;

const Gauge = styled.div`
  background-color: skyblue;
  display: inline-block;
  height: inherit;
  position: relative;
  top: -4px;
  width: ${(props) => props.percent}%;
`;
```

- Props 로 값을 전달 받아 게이지를 그려서 상황에 따라 처리

Src/component/Progress.js



결과를 만드는  
Check() 삽입!



# Check() 액션 생성 함수 삽입!

- 이제 페이지는 잘 넘어 갑니다!
- 그럼 MBTI 조사 결과 값을 만들어야 겠죠!?
- 해당 기능은 CHECK Action 이 담당합니다!



# Check() 액션 생성 함수

```
// payload -> 선택에 다른 결과 값 result 전달 필요
export function check(result) {
  return {
    type: CHECK,
    payload: { result },
  };
}
```

Src/store/modules/mbti.js

- Check() 액션 생성 함수는 결과 값 만을 전달 받네요?

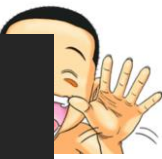


# 리듀서에서의 동작

```
case CHECK:
  return {
    ...state,
    mbtiResult: state.mbtiResult + action.payload.result,
  };
```

Src/store/modules/mbti.js

- 액션 생성 함수로 전달 받은 결과를 mbtiResult 라는 결과 문자열에 추가를 해주는 액션이 끝입니다!
- 그럼, check() 함수를 호출 할 때, 설문 객체에 포함 된 결과 문자열만 전달 하면 되겠군요!



```
survey: [  
  {  
    question:  
      '퇴근 직전에 동료로부터 개발자 모임에 초대를 받은 나!\n\n퇴근 시간에 나는?',  
    answer: [  
      {  
        text: '그런 모임을 왜 이제서야 알려 준거야! 당장 모임으로 출발한다',  
        result: 'E',  
      },  
      {  
        text: '1년 전에 알려줬어도 안갔을 건데 뭘... 더 빠르게 집으로 간다',  
        result: 'I',  
      },  
    ],  
  },  
],  
},
```

- 
- 이 Result 값만 전달 하면 됩니다!



Dispatch 로  
Check() 전달!





```
return (  
  <>  
    <SurveyQuestion>{survey[page - 1].question}</SurveyQuestion>  
    <ul>  
      {survey[page - 1].answer.map((el, index) => {  
        return (  
          <li key={index}>  
            <SkyblueButton  
              text={el.text}  
              clickEvent={() => {  
                dispatch(check(el.result));  
                dispatch(next());  
              }}  
            />  
            {index === 0 && <Vs>VS</Vs>}  
          </li>  
        );  
      })}  
    </ul>  
    <Progress page={page} maxPage={survey.length} />  
  </>  
)
```

- 설문 항목에 있던 결과 문자열의 값을 check()의 인자로 전달!

Src/component/Mbti.js

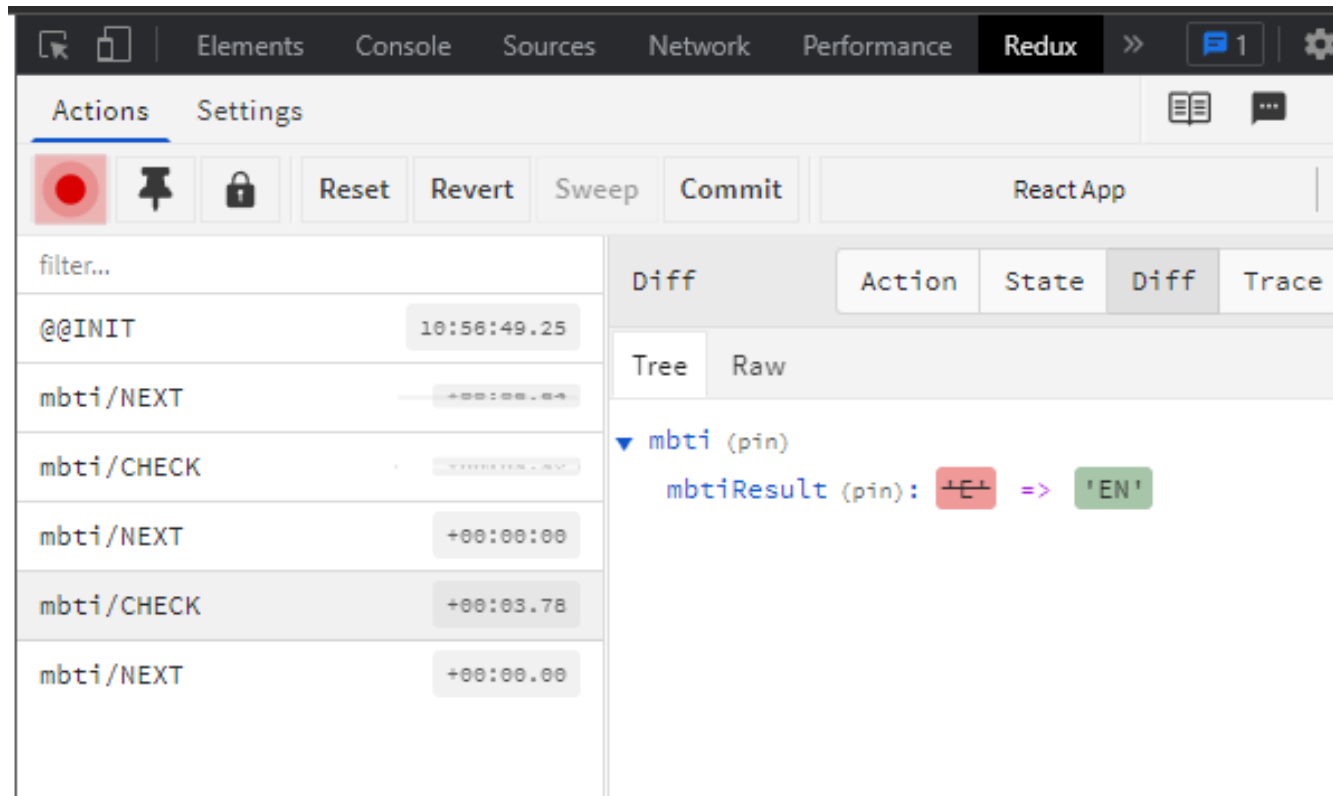


# 동작 확인



# 실제 동작이 잘 되는지 확인해 봅시다!

- 이럴 때 저번에 설치 해 두었던 Redux Devtools 가 역할을 합니다!



```
▼ mbti (pin)  
  mbtiResult (pin): 'EN' => 'ENFJ'
```



# 결과 출력 컴포넌트

## 작성



# 이제 redux 에 모인 결과를 출력

- MBTI 결과가 잘 반영 되는 것을 확인 하였으니, 해당 결과를 보여줄 결과 컴포넌트인 Show.js 를 만들어 봅시다!
- **Show 컴포넌트**는 Mbti 최종 결과가 들어있는 mbtiResult 값과, Mbti 결과 값에 맞는 설명 + 이미지를 출력해 주면 됩니다!
- 먼저 텍스트 부터 입력해서 디자인 부터 하고 결과 값 출력을 해봅시다!



```
import styled from 'styled-components';

export default function Show() {
  return (
    <>
      <Header>당신의 개발자 MBTI 결과는?</Header>
      <Explanation>결과 설명 출력</Explanation>
      <Result>결과 출력</Result>
      <Additional>이건 재미로 읽어 보세요!</Additional>
      <AdditionalImg src={} alt="팩폭" />
      <OrangeButton text="다시 검사하기" clickEvent={} />
    </>
  );
}
```

Src/component/Show.js

```
const Header = styled.p`  
  font-size: 3em;  
`;  
;
```

```
const Explanation = styled.p`  
  font-size: 1.5em;  
  color: #777;  
`;  
;
```

```
const Result = styled.p`  
  font-size: 3em;  
  color: dodgerblue;  
`;  
;
```

```
const Additional = styled.p`  
  font-size: 2em;  
  color: orange;  
`;  
;
```

```
const AdditionalImg = styled.img`  
  width: 500px;  
  transform: translateX(-35px);  
`;  
;
```

Src/component/Show.js





# Redux 에서 결과 값 출력하기!





# Redux 에서 결과 값 받아서 출력하기!

- 컴포넌트 디자인은 마쳤으니 이제 Redux 에서 결과 값을 받아서 출력해 봅시다!
- MBTI 결과는 Store 의 mbtiResult 에 있으므로 해당 값을 받아오기
- 그리고 설명은 각각의 MBTI 결과 값을 Key 로 가지는 객체로 선언을 하였기 때문에 편리하게 접근이 가능합니다!



```
export default function Show() {
  const result = useSelector((state) => state.mbti.mbtiResult);
  const explanation = useSelector((state) => state.mbti.explanation[result]);
  const dispatch = useDispatch();

  return (
    <>
      <Header>당신의 개발자 MBTI 결과는?</Header>
      <Explanation>{explanation.text}</Explanation>
      <Result>{result}</Result>
      <Additional>이건 재미로 읽어 보세요!</Additional>
      <AdditionalImg src={explanation.img} alt="팩폭" />
      <PinkButton text="다시 검사하기" clickEvent={} />
    </>
  );
}
```

Src/component/Show.js



# Reset()

## 액션 생성 함수 전달



# 다시하기 버튼 기능 추가!

- 이제 다시하기 버튼을 눌렀을 때, 페이지가 최초로 돌아가는 기능을 추가해 주면 됩니다!
- 이미 RESET 기능(page 를 0 으로 만들고, mbtiResult 를 초기화)은 구현이 되었으니 dispatch 를 통해 전달만 합시다!



```
export default function Show() {
  const result = useSelector((state) => state.mbti.mbtiResult);
  const explanation = useSelector((state) => state.mbti.explanation[result]);
  const dispatch = useDispatch();

  return (
    <>
      <Header>당신의 개발자 MBTI 결과는?</Header>
      <Explanation>{explanation.text}</Explanation>
      <Result>{result}</Result>
      <Additional>이건 재미로 읽어 보세요!</Additional>
      <AdditionalImg src={explanation.img} alt="팩폭" />
      <PinkButton text="다시 검사하기" clickEvent={() => dispatch(reset())} />
    </>
  );
}
```

Src/component/Show.js



# App.js

## 분기 처리!



# App.js 분기 처리

- 이제 결과 페이지도 보여줘야 하기 때문에, 결과 페이지에 대한 분기 처리도 해봅시다!
- Page 가 0 → Start 컴포넌트
- Page 가 1 ~ n → Mbti 컴포넌트
- Page 가 n + 1 → Show 컴포넌트
- If 문을 쓰는 것 보다는 간단하게 3항 연산자를 2중으로 써서 처리 해봅시다!



```
function App() {  
  const page = useSelector((state) => state.mbti.page);  
  const survey = useSelector((state) => state.mbti.survey);  
  
  return (  
    <>  
      <GlobalStyle />  
      <Main>  
        {page === 0 ? (  
          <Start />  
        ) : page !== survey.length + 1 ? (  
          <Mbti />  
        ) : (  
          <Show />  
        )}  
      </Main>  
    </>  
  );  
}
```

Src/App.js









# SQL

(Structured Query Language)



# 관계형, Relational DBMS(RDBMS)

- RDBMS 는 SQL 을 사용하는 DB 입니다
- 키와 값의 관계를 테이블화 시킨 원칙을 토대로 DB 를 구성합니다

아이디	이름	연락처
flower	화사	010-1111-1111
finetree	솔라	010-2222-2222
moon	문별	010-3333-3333
whee	휘인	010-4444-4444



# 관계형, Relational DBMS(RDBMS)



- 장점

- 구조화가 명확하게 되어 있어서 예외가 없음
- 데이터 입, 출력 속도가 매우 빠릅니다
- 신뢰성이 매우 높음

- 단점

- DB의 구조 변경이 매우 어려움 → 빅데이터 등에는 사용이 어려움(새로운 키가 추가 되면 전체 스키마 변경이 필요)



개발자	오라클
발표일	1995년 5월 23일
라이선스	GPL v2 (커뮤니티 에디션) 상용 라이선스 (표준, 엔터프라이즈, 클러스터)
링크	   



[https://www.w3schools.com/sql/trysql.asp?filename=trysql\\_select\\_all](https://www.w3schools.com/sql/trysql.asp?filename=trysql_select_all)

---

### SQL Statement:

```
SELECT * FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

### Result:

Click "**Run SQL**" to execute the SQL statement above.

W3Schools has created an SQL database in your browser.

The menu to the right displays the database, and will reflect any changes.

Feel free to experiment with any SQL statement.

You can restore the database at any time.



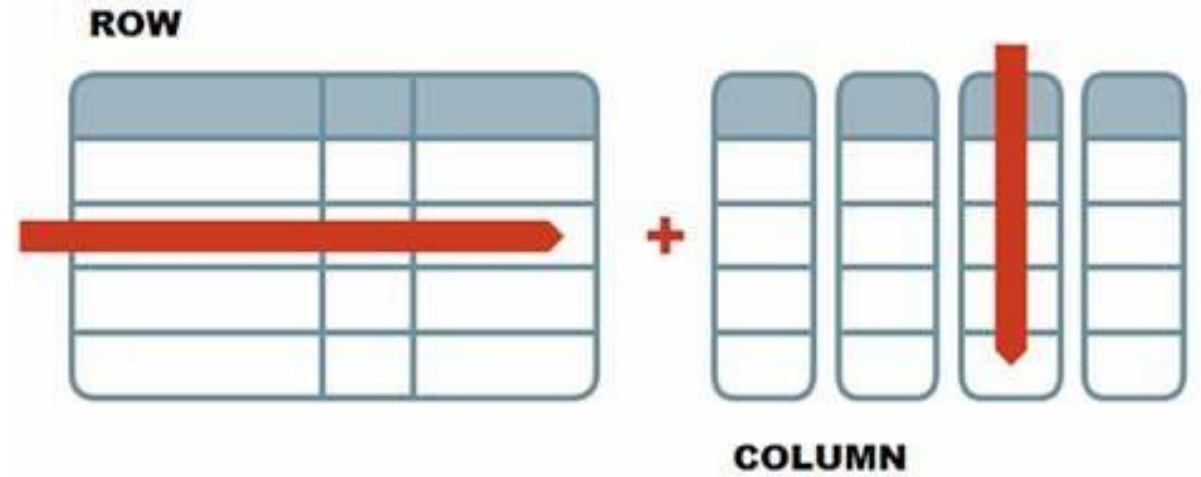




Diagram illustrating a table structure with labels and arrows:

- 행(row)**: Points to the rows of the table.
- 열 이름**: Points to the header row.
- 열(column)**: Points to the columns of the table.

아이디	이름	연락처
flower	화사	010-1111-1111
finetree	솔라	010-2222-2222
moon	문별	010-3333-3333
whee	휘인	010-4444-4444





# SELECT



# SELECT, DB 에서 원하는 데이터 가져오기

- 갓춰진 DB 에서 데이터를 뽑아 올 때 사용하는 구문 입니다!
- 일단
- `SELECT * FROM Customer;`

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada
11	B's Beverages	Victoria Ashworth	Fauntleroy Circus	London	EC2 5NT	UK
12	Cactus Comidas para llevar	Patricio Simpson	Cerrito 333	Buenos Aires	1010	Argentina
13	Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.	05022	Mexico
14	Chop-suey Chinese	Yang Wang	Hauptstr. 29	Bern	3012	Switzerland
15	Comércio Mineiro	Pedro Afonso	Av. dos Lusíadas, 23	São Paulo	05432-043	Brazil



# 원하는 컬럼의 값만 가지고 올 때?

- 모든 컬럼 값이 필요 한게 아니라면!?
- `SELECT 컬럼명 FROM table`
- `SELECT City, Country FROM Customers`

## SQL Statement:



```
SELECT City, Country FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

## Result:

Number of Records: 91

City	Country
Berlin	Germany
México D.F.	Mexico
México D.F.	Mexico
London	UK
Luleå	Sweden
Mannheim	Germany
Strasbourg	France

# DB에는 영향 없이 원하는 데이터 추가 할 때!



- 필요한 값을 임의로 추가해서 가져오고 싶을 땐?
- `SELECT 컬럼명, 원하는 데이터 FROM table`
- `SELECT City, 1, '원하는 문자열', NULL FROM Customers`

# SQL Statement:

SELECT City, 1, '원하는 문자열', NULL FROM Customers

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

# Result:

Number of Records: 91

1	City	'원하는 문자열'	NULL
1	Berlin	원하는 문자열	null
1	México D.F.	원하는 문자열	null
1	México D.F.	원하는 문자열	null
1	London	원하는 문자열	null
1	Luleå	원하는 문자열	null
1	Mannheim	원하는 문자열	null
1	Strasbourg	원하는 문자열	null



# 원하는 조건을 충족하는 Row 만 가져 올 때!



- 원하는 조건을 충족 하는 값을 찾고 싶을 땐? **WHERE**
- **SELECT \* FROM table WHERE 조건**
- **SELECT \* FROM OrderDetails WHERE Quantity > 5;**

SQL Statement:

SELECT \* FROM OrderDetails WHERE Quantity > 5;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 476

OrderDetailID	OrderID	ProductID	Quantity
1	10248	11	12
2	10248	42	10
4	10249	14	9
5	10249	51	40
6	10250	41	10
7	10250	51	35
8	10250	65	15
9	10251	22	6



# Row 를 정렬해서 가져 올 때!

- 선택한 값들을 정렬해서 가지고 올 때는? ORDER BY
- SELECT \* FROM Customers ORDER BY ContactName;
- SELECT \* FROM Customers ORDER BY ContactName DESC;
- SELECT \* FROM Customers ORDER BY CustomerName ASC, ContactName DESC;

SQL Statement:

SELECT \* FROM Customers ORDER BY ContactName;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
69	Romero y tomillo	Alejandra Camino	Gran Vía, 1	Madrid	28001	Spain
52	Morgenstern Gesundkost	Alexander Feuer	Heerstr. 22	Leipzig	04179	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
81	Tradição Hipermercados	Anabela Domingues	Av. Inês de Castro, 414	São Paulo	05634-030	Brazil
31	Gourmet Lanchonetes	André Fonseca	Av. Brasil, 442	Campinas	04876-786	Brazil
19	Eastern Connection	Ann Devon	35 King George	London	WX3 6FW	UK
41	La maison d'Asie	Annette Roulet	1 rue Alsace-Lorraine	Toulouse	31000	France
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
21	Familia Arquibaldo	Aria Cruz	Rua Orós, 92	São Paulo	05442-030	Brazil
75	Split Rail Beer & Ale	Art Braunschweiger	P.O. Box 555	Lander	82520	USA

# SQL Statement:

```
SELECT * FROM Customers ORDER BY ContactName DESC;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

# Result:

Number of Records: 91

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
54	Océano Atlántico Ltda.	Yvonne Moncada	Ing. Gustavo Moncada 8585 Piso 20-A	Buenos Aires	1010	Argentina
42	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver	V3F 2K1	Canada
36	Hungry Coyote Import Store	Yoshi Latimer	City Center Plaza 516 Main St.	Elgin	97827	USA



# Row 의 개수를 지정 하고 싶을 때!

- 가지고 오는 ROW의 수를 지정하고 싶을 때? **LIMIT**
- **SELECT \* FROM Customers LIMIT 10;**
- **SELECT \* FROM Customers LIMIT 30, 10;**
  - 원하는 Row 순번, 자르는 개수

## SQL Statement:

```
SELECT * FROM Customers LIMIT 10;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

## Result:

Number of Records: 10

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
1	Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin	12209	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Avda. de la Constitución 2222	México D.F.	05021	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mataderos 2312	México D.F.	05023	Mexico
4	Around the Horn	Thomas Hardy	120 Hanover Sq.	London	WA1 1DP	UK
5	Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå	S-958 22	Sweden
6	Blauer See Delikatessen	Hanna Moos	Forsterstr. 57	Mannheim	68306	Germany
7	Blondel père et fils	Frédérique Citeaux	24, place Kléber	Strasbourg	67000	France
8	Bólido Comidas preparadas	Martín Sommer	C/ Araquil, 67	Madrid	28023	Spain
9	Bon app'	Laurence Lebihans	12, rue des Bouchers	Marseille	13008	France
10	Bottom-Dollar Marketse	Elizabeth Lincoln	23 Tsawassen Blvd.	Tsawassen	T2F 8M4	Canada

# Column 명을 변경해서 가지고 오고 싶을 때?



- Column 명을 변경해서 가지고 올 때? **AS**
- **SELECT** CustomerId **AS** id, CustomerName **AS** name **FROM**  
Customers;





## SQL Statement:

```
SELECT CustomerId AS id, CustomerName AS name FROM Customers;
```

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

## Result:

Number of Records: 91

id	name
1	Alfreds Futterkiste
2	Ana Trujillo Emparedados y helados
3	Antonio Moreno Taquería
4	Around the Horn
5	Berglunds snabbköp
6	Blauer See Delikatessen



# 테이블 합치기!

# JOIN

# 여러 테이블의 값을 하나로 합치고 싶다면!?



- 여러 테이블을 하나로 붙이고 싶을 땐? JOIN
- `SELECT * FROM Products P JOIN Categories C ON P.CategoryID = P.CategoryID;`

SQL Statement:

SELECT \* FROM Products P JOIN Categories C ON P.CategoryID = P.CategoryID;

Edit the SQL Statement, and click "Run SQL" to see the result.

Run SQL »

Result:

Number of Records: 616

ProductID	ProductName	SupplierID	CategoryID	Unit	Price	CategoryName	Description
1	Chais	1	1	10 boxes x 20 bags	18	Beverages	Soft drinks, coffees, teas, beers, and ales
1	Chais	1	2	10 boxes x 20 bags	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings
1	Chais	1	3	10 boxes x 20 bags	18	Confections	Desserts, candies, and sweet breads
1	Chais	1	4	10 boxes x 20 bags	18	Dairy Products	Cheeses
1	Chais	1	5	10 boxes x 20 bags	18	Grains/Cereals	Breads, crackers, pasta, and cereal
1	Chais	1	6	10 boxes x 20 bags	18	Meat/Poultry	Prepared meats
1	Chais	1	7	10 boxes x 20 bags	18	Produce	Dried fruit and bean curd
1	Chais	1	8	10 boxes x 20 bags	18	Seafood	Seaweed and fish
2	Chang	1	1	24 - 12 oz bottles	19	Beverages	Soft drinks, coffees, teas, beers, and ales
2	Chang	1	2	24 - 12 oz bottles	18	Condiments	Sweet and savory sauces, relishes, spreads, and seasonings





Local 에서  
연습 시작!

# 이제 Local 에 Mysql DB 를 구축해 봅시다!



- 터미널 실행!

- Mysql -V

```
C:\Users\wtetz>mysql -V  
mysql Ver 8.0.28 for Win64 on x86_64 (MySQL Community Server - GPL)
```

- 이게 안 뜨면 시스템 환경 변수 설정
  - <https://dog-developers.tistory.com/21>



# 이제 Local 에 Mysql DB 를 구축해 봅시다!

- `mysql -u root -p`
- 설정한 비밀 번호를 치고 들어가기

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is 53  
Server version: 8.0.28 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2022, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> _
```



# 이제 Local 에 Mysql DB 를 구축해 봅시다!



- show databases;

```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mydb       |
| mysql      |
| performance_schema |
| sakila     |
| sys        |
| world      |
+-----+
7 rows in set (0.00 sec)
```

# 이제 Local 에 Mysql DB 를 구축해 봅시다!



- use sakila;
- show tables;

```
mysql> use sakila;
Database changed
mysql> show tables;
+-----+
| Tables_in_sakila |
+-----+
| actor              |
| actor_info         |
| address            |
| category           |
| city               |
| country            |
| customer           |
| customer_list      |
| film               |
| film_actor         |
| film_category      |
| film_list          |
| film_text          |
| inventory          |
| language           |
| nicer_but_slower_film_list |
| payment            |
| rental             |
| sales_by_film_category |
| sales_by_store     |
+-----+
```



# 여기서 부터는 자유롭게 Query 를 사용!

- 다만 아무래도 CLI 는 불편하기 때문에 GUI 를 사용하는게 좋겠죠?
- 명령어가 익숙하면 상관이 없지만, 그렇지 않다면 아무래도 Workbench 를 사용하는 편이 좋습니다!
- MySQL Workbench 실행



# MySQL Workbench

# Workbench 사용!



- 먼저 schema 선택 하기

Navigator

MANAGEMENT

- Server Status
- Client Connections
- Users and Privileges
- Status and System Variables
- Data Export
- Data Import/Restore

INSTANCE

- Startup / Shutdown
- Server Logs
- Options File

PERFORMANCE

- Dashboard
- Performance Reports
- Performance Schema Setup

Administration Schemas

Information

Table: **answer**

Columns:

**\_id** int AI PK

Query 1 question answer ans

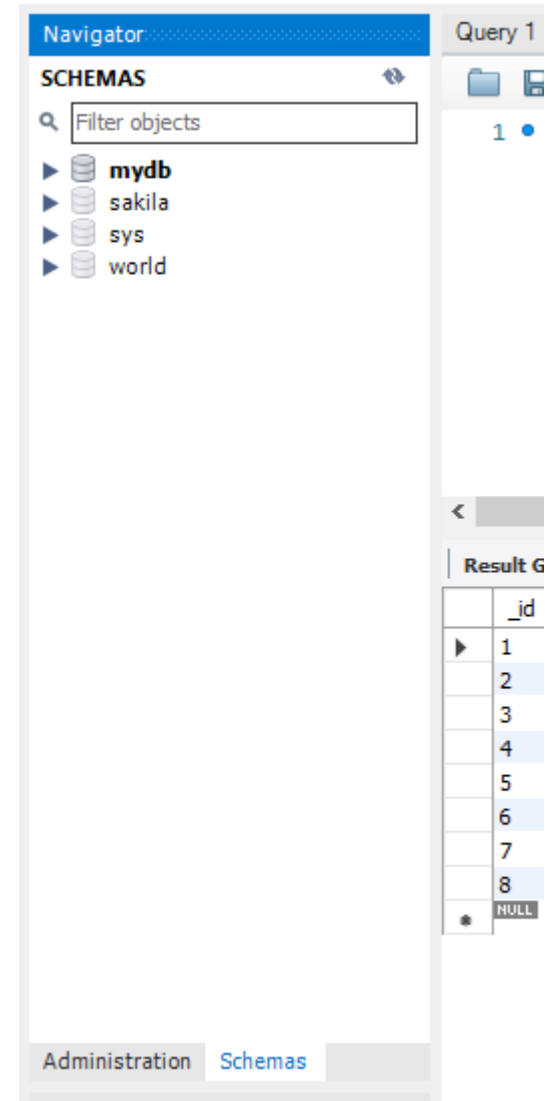
1 • SELECT \* FROM mydb.answer

Result Grid

	_id	question_id	answer_text
▶	1	0	그런 모임을 왜 이제서
	2	0	1년 전에 알려줬어도
	3	1	원소리여, 그냥 하던
	4	1	오호? 그런게 있어? 들
	5	2	무슨 버그가 발생한 거
	6	2	아... 내일도 야근 각
	7	3	일단 빠르게 개발 완
	8	3	그거 내일 아침에 와
*	NULL	NULL	NULL

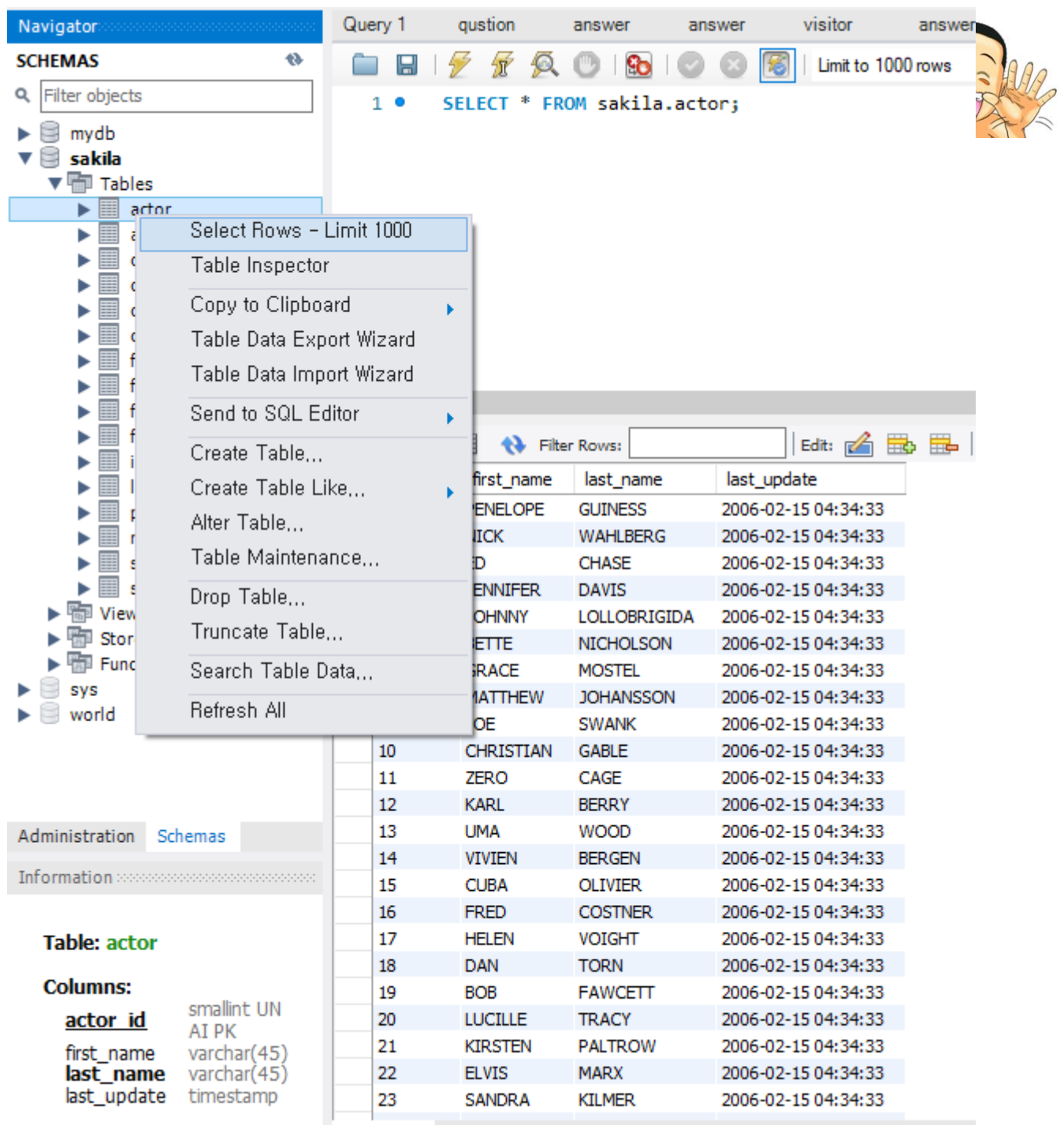
# Schema

- 만들어진 DB 를 확인 할 수 있습니다!



# Schema

- 각각의 DB 에 있는 테이블과 데이터 확인 가능



Query 1 question answer answer visitor answer

Limit to 1000 rows

1 • SELECT \* FROM sakila.actor;

Filter objects

mydb

sakila

Tables

actor

Select Rows - Limit 1000

Table Inspector

Copy to Clipboard

Table Data Export Wizard

Table Data Import Wizard

Send to SQL Editor

Create Table...

Create Table Like...

Alter Table...

Table Maintenance...

Drop Table...

Truncate Table...

Search Table Data...

Refresh All

Filter Rows:

	first_name	last_name	last_update
1	ENVELOPE	GUINNESS	2006-02-15 04:34:33
2	WICK	WAHLBERG	2006-02-15 04:34:33
3	D	CHASE	2006-02-15 04:34:33
4	ENNIFER	DAVIS	2006-02-15 04:34:33
5	JOHNNY	LOLLOBRIGIDA	2006-02-15 04:34:33
6	ETTE	NICHOLSON	2006-02-15 04:34:33
7	GRACE	MOSTEL	2006-02-15 04:34:33
8	MATTHEW	JOHANSSON	2006-02-15 04:34:33
9	OE	SWANK	2006-02-15 04:34:33
10	CHRISTIAN	GABLE	2006-02-15 04:34:33
11	ZERO	CAGE	2006-02-15 04:34:33
12	KARL	BERRY	2006-02-15 04:34:33
13	UMA	WOOD	2006-02-15 04:34:33
14	VIVIEN	BERGEN	2006-02-15 04:34:33
15	CUBA	OLIVIER	2006-02-15 04:34:33
16	FRED	COSTNER	2006-02-15 04:34:33
17	HELEN	VOIGHT	2006-02-15 04:34:33
18	DAN	TORN	2006-02-15 04:34:33
19	BOB	FAWCETT	2006-02-15 04:34:33
20	LUCILLE	TRACY	2006-02-15 04:34:33
21	KIRSTEN	PALTROW	2006-02-15 04:34:33
22	ELVIS	MARX	2006-02-15 04:34:33
23	SANDRA	KILMER	2006-02-15 04:34:33

Administration Schemas

Information

Table: actor

Columns:

actor_id	
first_name	smallint UNSIGNED
last_name	AI PK
last_update	varchar(45)
	varchar(45)
	timestamp



# 수업을 위한 DB 만들기

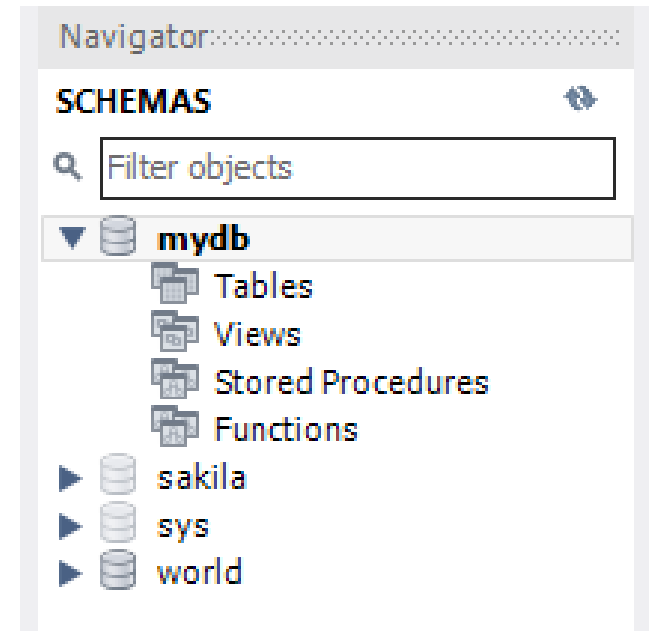


# DB 생성!



- `CREATE SCHEMA `mydb` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;`

```
Query 1 x
1 CREATE SCHEMA `myDB` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci;
2
```



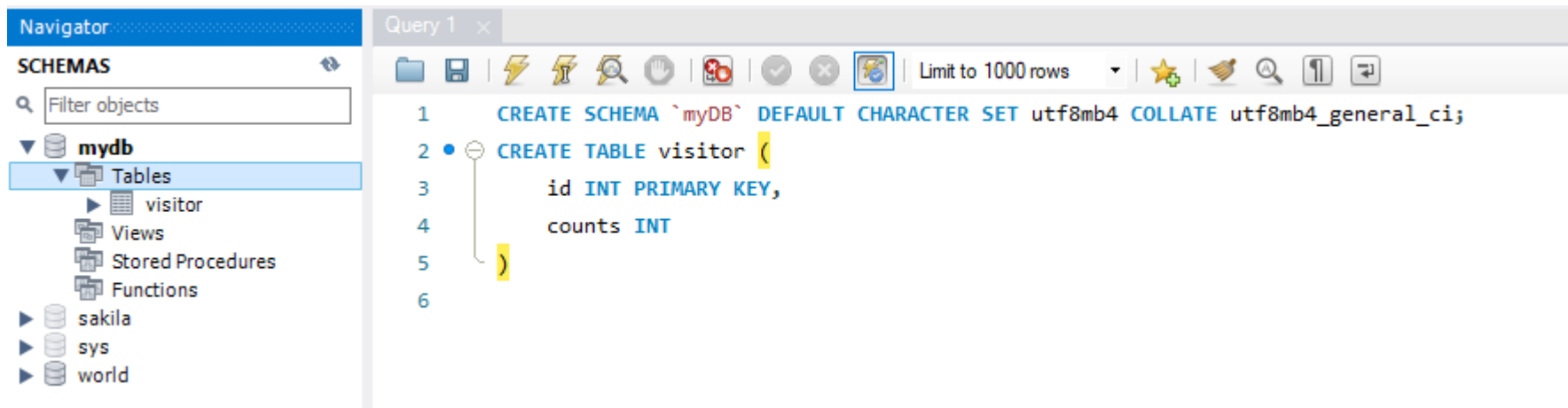


# TABLE 생성!

# DB 의 TABLE 생성!



- CREATE TABLE visitor ( id INT PRIMARY KEY, counts INT );





### 3. 테이블 생성시 제약 넣기

```
CREATE TABLE people (  
  person_id INT AUTO_INCREMENT PRIMARY KEY,  
  person_name VARCHAR(10) NOT NULL,  
  nickname VARCHAR(10) UNIQUE NOT NULL,  
  age TINYINT UNSIGNED,  
  is_married TINYINT DEFAULT 0  
);
```

제약	설명
AUTO_INCREMENT	새 행 생성시마다 자동으로 1씩 증가
PRIMARY KEY	중복 입력 불가, NULL(빈 값) 불가
UNIQUE	중복 입력 불가
NOT NULL	NULL(빈 값) 입력 불가
UNSIGNED	(숫자일시) 양수만 가능
DEFAULT	값 입력이 없을 시 기본값



# DATA 삽입하기



# 생성한 TABLE 에 DATA 삽입!

- **INSERT INTO** visitor ( id, counts ) **VALUES** ( 1, 0 );

A screenshot of a database query editor interface. The top toolbar includes icons for file operations, query execution, and a 'Limit to 1000 rows' dropdown. The query editor contains two lines of SQL code:

```
1 • SELECT * from mydb.visitor;  
2 • insert into visitor (id, counts) VALUES (1, 0);
```

The bottom section shows the 'Result Grid' with a table containing the results of the query. The table has two columns: 'id' and 'counts'. The first row shows the value '1' for 'id' and '0' for 'counts'. Below this row, there are two cells containing 'NULL'. The interface also includes a 'Filter Rows' input field and buttons for 'Edit', 'Export/Import', and 'Wrap'.



Query 1 x question answer answer visitor answer actor

Limit to 1000 rows

```
1 • SELECT * from mydb.visitor;  
2 • insert into visitor (id, counts) VALUES (2, 1);  
3
```

<

Result Grid Filter Rows: Edit: Export/Impo

	id ▲	counts
▶	1	5
	2	1
*	NULL	NULL



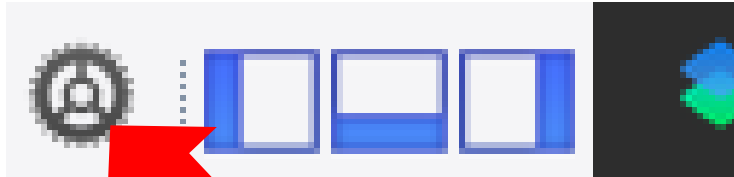
# DATA 수정 및 삭제



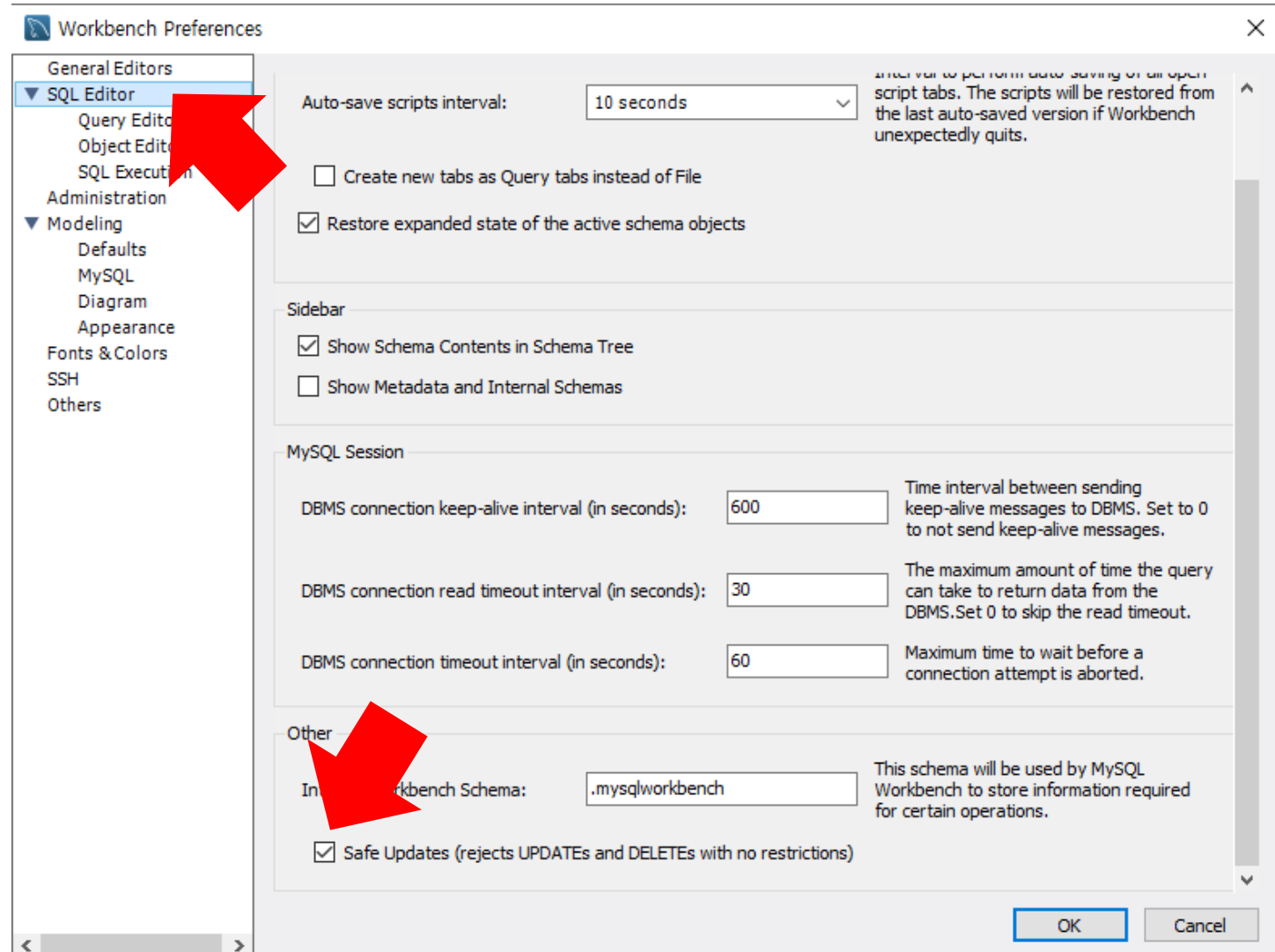


# 먼저 설정 변경이 필요합니다!

- 환경 설정 > SQL Editor



- 체크 박스를 해제







# DATA 삭제

# DATA 삭제!



- DELETE FROM visitor WHERE id = 2;

3 • DELETE FROM visitor WHERE id = 2;

<		
Result Grid		
Filter Rows: <input type="text"/>		
Edit:  		
	id	counts
▶	1	5
✱	NULL	NULL



# DATA 수정



# DATA 수정(Update)!

- **UPDATE** visitor **SET** counts = counts + 1 **WHERE** id = 1;

3 • **DELETE FROM** visitor **WHERE** id = 2;

<

Result Grid | Filter Rows:  | Edit:

	id	counts
▶	1	5
✱	NULL	NULL



# Table 수정 및 삭제하기



## ALTER TABLE - 테이블 변경

```
-- 테이블명 변경
ALTER TABLE people RENAME TO friends,
-- 컬럼 자료형 변경
CHANGE COLUMN person_id person_id TINYINT,
-- 컬럼명 변경
CHANGE COLUMN person_name person_nickname VARCHAR(10),
-- 컬럼 삭제
DROP COLUMN birthday,
-- 컬럼 추가
ADD COLUMN is_married TINYINT AFTER age;
```

## DROP TABLE - 테이블 삭제

```
DROP TABLE friends;
```



WorkBench 로  
수행하기!





# TABLE 생성!



Navigator

SCHEMAS

Filter objects

mydb

Tables

- answer
- explanation
- new\_table
  - Columns
  - Indexes
  - Foreign Keys
  - Triggers
- question
- timestamps
- visitor
- Views
- Stored Procedures
- Functions
- sakila
- sys
- world

Administration Schemas

Information

Schema: mydb

question answer answer visitor answer actor new\_table timestamps new\_table - Table x



Table Name: new\_table

Schema: mydb

Charset/Collation: Default Charset

Default Collation

Engine: InnoDB

Comments:

Column Name	Datatype	PK	NN	UQ	B	UN	ZF	AI	G	Default/Expression
		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Column Name:

Data Type:

Charset/Collation:

Default:

Comments:

Storage:

☐ Virtual

☐ Stored

☐ Primary Key

☐ Not Null

☐ Unique

☐ Binary

☐ Unsigned

☐ Zero Fill

☐ Auto Increment

☐ Generated

Columns

Indexes

Foreign Keys

Triggers

Partitioning

Options

Apply

Revert





# TABLE 수정!



Navigator

**SCHEMAS**

Filter objects

mydb

Tables

- answe
- explai
- new\_t
- Co
- In
- For
- Tri
- questi
- timest
- visitor

Views

Stored Pro

Functions

sakila

sys

world

question

answer

ans

Table Nam

Charset/Collatic

ments:

Administration

Sch

Information

- Select Rows - Limit 1000
- Table Inspector
- Copy to Clipboard
- Table Data Export Wizard
- Table Data Import Wizard
- Send to SQL Editor
- Create Table...
- Create Table Like...
- Alter Table...
- Table Maintenance...
- Drop Table...
- Truncate Table...
- Search Table Data...
- Refresh All



# Data 수정 및 삭제













answer

answer

visitor

×

answer



1 •


SELECT \* FROM mydb.visitor;


2 •

UPDATE mydb.visitor SET counts

<

Result Grid





Filter Rows:

	id	counts
▶	2	6
⊛	NULL	NULL

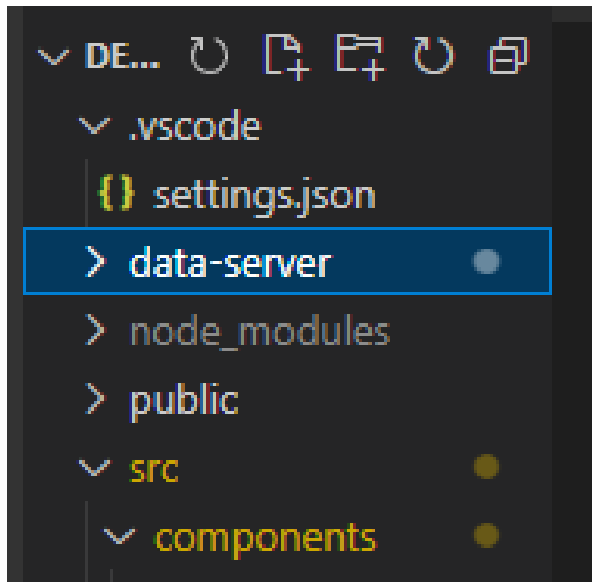


# DB 통신용 서버 구축하기



# DB 통신을 하는 Back 서버를 구축

- 이제 DB 통신을 하는 Backend 서버를 구축하고 해당 서버로 부터 데이터를 받아 봅시다!
- 리액트 프로젝트의 최 상단에 data-server 폴더 만들기!





# Express 서버 구축!



- 먼저 모듈부터 설치 합시다!
- `Npm i express cors mysql`



# Express 서버 코드 작성!

```
const express = require('express');
const cors = require('cors');
const PORT = 4000;

const app = express();

app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cors());

const dataRouter = require('./routes/data');
app.use('/data', dataRouter);

app.listen(PORT, () => {
  console.log(`데이터 통신 서버가 ${PORT}에서 작동 중`);
});
```

Data-server/server.js



# MySQL 서버 통신용 모듈 작성

- dbConnect.js 라는 DB 통신용 모듈을 작성해 봅시다!

```
const mysql = require('mysql');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '1234',
  port: '3306',
  database: 'mydb',
});
```

```
connection.connect();
```

```
module.exports = connection;
```

Data-server/dbConnect.js



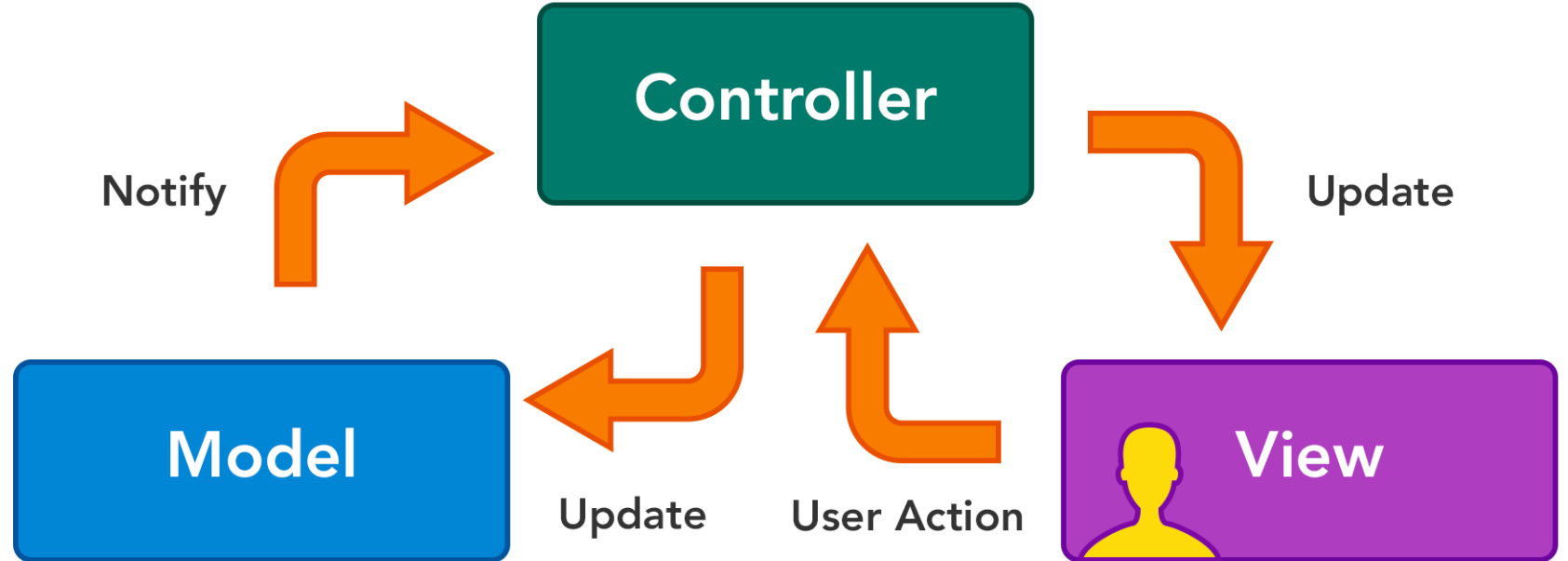
# DB 통신용

# 컨트롤러 작성



# DB 통신 컨트롤러 작성

- DB와 통신을 해서 우리가 원하는 데이터를 만들어 주는 Controller 를 작성해 봅시다





# DB 통신 컨트롤러 작성

- Constrollers 폴더를 작성하고 dataController.js 파일 생성



```
const connection = require('../dbConnect');

const db = {
  getCounts: (cb) => {
    connection.query('SELECT counts FROM mydb.visitor;', (err, data) => {
      if (err) throw err;
      cb(data);
    });
  },
  incCounts: (cb) => {
    connection.query(
      'UPDATE mydb.visitor SET counts = counts + 1 WHERE id = 1;',
      (err) => {
        if (err) throw err;
        cb(JSON.stringify('업데이트 성공'));
      }
    );
  },
};

module.exports = db;
```

Data-server/controlers/dataController.js



# 주소별

# Routing 처리



# 주소 요청에 따른 데이터 처리를 위한 라우팅



- Routes 폴더를 작성하고, data.js 모듈 만들기!



```
const express = require('express');
const router = express.Router();

const db = require('../controllers/dataController');

router.get('/count', (req, res) => {
  db.getCounts((data) => {
    res.send(data);
  });
});

router.post('/inccount', (req, res) => {
  db.incCounts((msg) => {
    res.send(msg);
  });
});

module.exports = router;
```

Data-server/routes/data.js



# React 앱에서 데이터 요청!



# 시작 페이지에서 데이터 받기

```
export default function Start() {
  const [counts, setCounts] = useState(0);
  const dispatch = useDispatch();

  useEffect(() => {
    async function fetchData() {
      const resCount = await fetch('http://localhost:4000/data/count');
      if (resCount.status === 200) {
        const num = await resCount.json();
        if (num[0].counts !== 0) setCounts(num[0].counts);
      } else {
        throw new Error('통신 이상');
      }
    }
    fetchData();
  }, [counts]);
```

Src/component/Start.js





```
return (  
  <>  
    <Header>개발자 MBTI 조사</Header>  
    <MainImg src="/images/main.jpg" alt="메인 이미지" />  
    <SubHeader>  
      개발자가 흔히 접하는 상황에 따라서 MBTI 를 알아 봅시다! 지금까지{'\n\n'}  
      {counts} 명이 참여해 주셨습니다!  
    </SubHeader>  
  
    <OrangeButton text="테스트 시작" clickEvent={() => dispatch(next())} />  
  </>  
>);  
}
```

Src/component/Start.js



# 통신 에러 시 대처 방법



```
D:\git\developer-mbti\node_modules\mysql\lib\protocol\Parser.js:437
  throw err; // Rethrow non-MySQL errors
  ^
```

```
Error: ER_NOT_SUPPORTED_AUTH_MODE: Client does not support authentication protocol requested by server;
consider upgrading MySQL client
    at Handshake.Sequence._packetToError (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Sequence.js:47:14)
    at Handshake.ErrorPacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\sequences\Handshake.js:123:18)
    at Protocol.parsePacket (D:\git\developer-mbti\node_modules\mysql\lib\protocol\Protocol.js:281:23)
```

<https://1mini2.tistory.com/88>

3



조운호

2018.07.13 오후 4:33

<https://stackoverflow.com/questions/50093144/mysql-8-0-client-does-not-support-authentication-protocol-requested-by-server>

ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '사용할패스워드'

뭔가 이렇게 하니까 됐어요~~

3 • ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY '1234';





마지막 페이지에서

Count + 1 요청



```
export default function Show() {
  const result = useSelector((state) => state.mbti.mbtiResult);
  const explanation = useSelector((state) => state.mbti.explanation[result]);
  const dispatch = useDispatch();

  const incCount = async () => {
    const resInc = await fetch('http://localhost:4000/data/inccount', {
      method: 'POST',
    });
    if (resInc.status === 200) {
      console.log(await resInc.json());
    } else {
      throw new Error('통신 이상');
    }
  };
};
```

Src/component/Show.js



```
return (  
  <>  
    <Header>당신의 개발자 MBTI 결과는?</Header>  
    <Explanation>{explanation.text}</Explanation>  
    <Result>{result}</Result>  
    <Additional>이건 재미로 읽어 보세요!</Additional>  
    <AdditionalImg src={explanation.img} alt="팩폭" />  
    <PinkButton  
      text="다시 검사하기"  
      clickEvent={() => {  
        incCount();  
        dispatch(reset());  
      }}  
    />  
  </>  
>);  
}
```

Src/component/Show.js



수고하셨습니다!