**실습 #1**

|  |
| --- |
| <?php      include('./dbinit.php');      $name = $\_POST['name'];      $age = $\_POST['age'];      $query = "insert into test\_table (name, age) values ('$name', '$age')";      if($result = mysqli\_query($conn, $query)){          while($row = mysqli\_fetch\_array ($result)){              print\_r($row);              echo "<br>";          }      }  ?>  <html>  <body>      <form method="post" action="">          <input type="text" name="name" >          <input type="text" name="age" >          <input type="submit">      </form>  </html> |

****

**텍스트이(가) 표시된 사진

자동 생성된 설명**

**실습 #2**

|  |
| --- |
| <html>  <body>  <table border="1" width="1000px" height="800px">  <?php  include('./dbinit.php');  $query = "select \* from Old\_Member union select \* from New\_Member";  echo '<tr>' .  '<th>이름</th><th>학교</th><th>학과</th><th>성별</th>  <th>입학년도</th><th>이메일</th><th>전화번호</th><th>생일</th>  <th>학년</th><th>학점</th>' .  '</tr>';    if($result = mysqli\_query($conn,$query)){  while($row = mysqli\_fetch\_array($result)){  echo '<tr><td>'. $row['Name'].'</td>'.  '<td>'.$row['School'].'</td>'.  '<td>'.$row['Major'].'</td>'.  '<td>'.$row['Gender'].'</td>'.  '<td>'.$row['EnterYear'].'</td>'.  '<td>'.$row['Email'].'</td>'.  '<td>'.$row['Phone'].'</td>'.  '<td>'.$row['Birth'].'</td>'.  '<td>'.$row['Grade'].'</td>'.  '<td>'.$row['Score'].'</td></tr>';    }  echo '</table>';  }  ?>  </body>  </html> |

**테이블이(가) 표시된 사진

자동 생성된 설명**

**실습3**

**Index.ejs**

|  |
| --- |
| <html>  <head>  <title> CAT-Security Node.js & MySQL </title>  <style>  table {      width: 100%;      text-align: center;      border-collapse: collapse;      border-top: 2px solid #ed0606;  }  th,  td {      border-bottom: 1px dashed #001763;  }  </style>  </head>  <body>  <h2> CAT-Security Node.js & MySQL</h2>  <div>  <table>      <thead>      <th> 이름 </th>      <th> 학교 </th>      <th> 전공 </th>      <th> 성별 </th>      <th> 입학년도 </th>      <th> 이메일 </th>      <th> 전화번호 </th>      <th> 생일 </th>      <th> 학년 </th>      </thead>  <tbody> <% for(var i of list) { %> <tr>      <td> <%= i.Name %></td>      <td> <%= i.School %></td>      <td> <%= i.Major %></td>      <td> <%= i.Gender %></td>      <td> <%= i.EnterYear %></td>      <td> <%= i.Email %></td>      <td> <%= i.Phone %></td>      <td> <%= i.Birth %></td>      <td> <%= i.Grade %></td>      <td> <%= i.Score %></td>      </tr> <% } %> </tbody>  </table>  </div>  </body>  </html> |

**App.js**

|  |
| --- |
| const express = require("express");  const ejs = require("ejs");  const path = require("path");  const app = express();  const conn = require("./db.js");  app.listen(8000, function() {      console.log("listening on port 8002");  });  app.use(express.urlencoded({ extended: false }));  app.set("views", path.join(\_\_dirname, "views"));  app.set("view engine", "ejs");  app.get("/", function (req, res) {      var sql = "SELECT \* FROM Old\_Member UNION SELECT \* FROM New\_Member";      conn.query(sql, function(err, rows) {          if (err) console.log("query is not excuted. select fail!\n" + err);          else res.render("index.ejs", {list: rows });      });  }); |

**Db.js**

|  |
| --- |
| const mysql = require("mysql");  const connection = mysql.createConnection( {      host: 'localhost',      user: 'minyeong3',      password: '2022Wnsldj!',      port: 3306,      database: 'minyeong3'  });  connection.connect(function(err){      if (err) console.log(err);      else console.log('Connected!');  });  module.exports = connection; |

**테이블이(가) 표시된 사진

자동 생성된 설명**

**방명록 만들기**

**App.js**

|  |
| --- |
| const express = require("express");  const ejs = require("ejs");  const path = require("path");  const app = express();  const conn = require("./db.js");  const bodyParser = require('body-parser');  app.listen(8002, function() {      console.log("listening on port 8002");  });  app.use(express.urlencoded({ extended: false }));  app.set("views", path.join(\_\_dirname, "views"));  app.set("view engine", "ejs");  app.get("/", function (req, res) {      var sql = "SELECT \* FROM memo";      conn.query(sql, function(err, rows) {          if (err) console.log("query is not excuted. select fail!\n" + err);          else res.render("memo.ejs", {list: rows });      });  });  app.get("/write", function(req, res){      res.sendFile(\_\_dirname + "/views/write.html");  });  app.post( "/write", function ( req, res ) {      console.log( req.body );        const ti = req.body.title;      console.log(ti);      const con = req.body.content;      console.log(con);      const na = req.body.name;      console.log(na);        var sql = "INSERT INTO memo(title, content, name) VALUES (?,?,?)";      var params =[ti, con, na];        conn.query(sql, params, function(err, rows){          if (err) console.log("query is not excuted. modify fail!\n" + err);          else res.redirect("/");      })  });  app.get("/modify/:seq", function(req, res){      var num = req.params.seq;      console.log(num);      res.sendFile(\_\_dirname + "/views/write.html");  });  app.post( "/modify/:seq", function ( req, res ) {      console.log( req.body );        var num = req.params.seq;      console.log(num);      const ti = req.body.title;      console.log(ti);      const con = req.body.content;      console.log(con);      const na = req.body.name;      console.log(na);        var sql= "update memo set title=?, content=?, name=? where seq=?";      var params =[ti, con, na, num];        conn.query(sql, params, function(err, rows){          if (err) console.log("query is not excuted. modify fail!\n" + err);          else res.redirect("/");      })  });  app.get("/delete/:seq", function(req, res){      var num = req.params.seq;      console.log(num);      var sql = "delete from memo where seq=?";      var params=[num];      conn.query(sql, params, function(err, rows){          if (err) console.log("query is not excuted. modify fail!\n" + err);          else res.redirect("/");      })  }); |

**Memo.ejs**

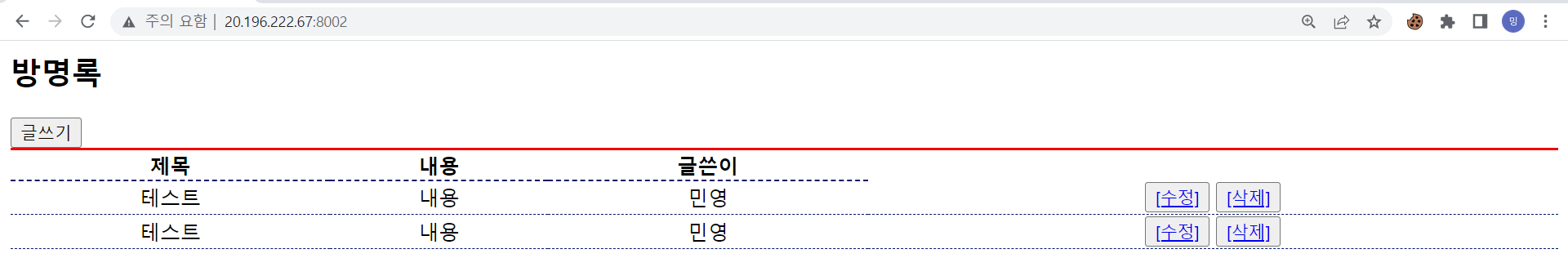
|  |
| --- |
| <html>  <head>  <title> 방명록 </title>  <style>  table {     width: 100%;     text-align: center;     border-collapse: collapse;     border-top: 2px solid #ed0606;  }  th,  td {     border-bottom: 1px dashed #001763;  }  </style>  </head>  <body>  <h2> 방명록 </h2>  <div>  <table>     <thead>     <th> 제목 </th>      <th> 내용 </th>      <th> 글쓴이 </th>     </thead>  <tbody> <% for(var i of list) { %> <tr>     <td> <%= i.title %></td>     <td> <%= i.content %></td>     <td> <%= i.name %></td>     <td><a href="/modify/<%=i.seq %>"><button>[수정]</a></button>        <a href="/delete/<%=i.seq %>"><button>[삭제]</a></button></td>     </tr> <% } %> </tbody>     <a href="/write"><button>글쓰기</button></a>  </table>  </div>  </body>  </html> |

**Write.html**

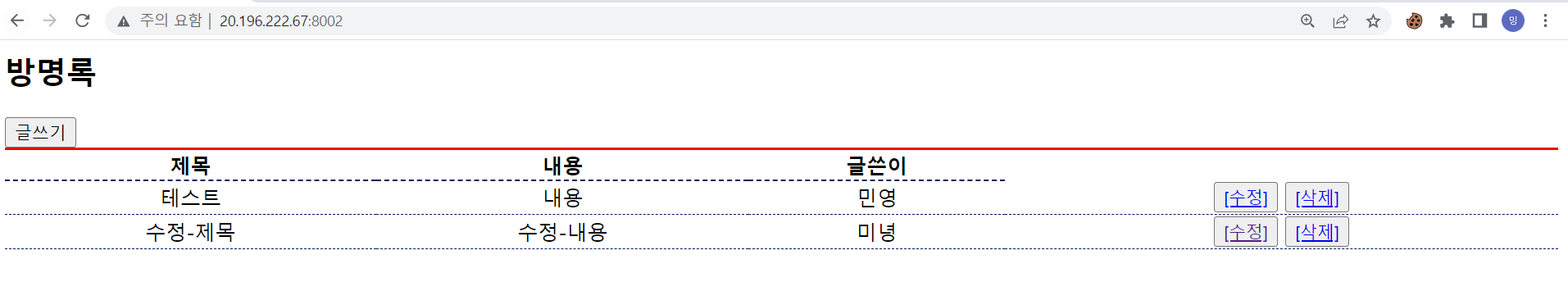
|  |
| --- |
| <!doctype html>  <head>  <meta charset="UTF-8">  <style>          table.table2 {              border-collapse: separate;              border-spacing: 1px;              text-align: left;              line-height: 1.5;              border-top: 1px solid #ccc;              margin: 20px 10px;          }          table.table2 tr {              width: 50px;              padding: 10px;              font-weight: bold;              vertical-align: top;              border-bottom: 1px solid #ccc;          }          table.table2 td {              width: 100px;              padding: 10px;              vertical-align: top;              border-bottom: 1px solid #ccc;          }      </style>  <title>방명록</title>  </head>  <body>          <h1><a href="/">방명록</a></h1>                  <form method="post" action="">                  <table style="padding-top:50px" align=center width=auto border=0 cellpadding=2>              <tr>                  <td style="height:40; float:center; background-color:#3C3C3C">                      <p style="font-size:15px; text-align:center; color:white; margin-top:15px; margin-bottom:15px"><b>글 작성하기</b></p>                  </td>              </tr>              <tr>                  <td bgcolor=white>                      <table class="table2">                         <tr>                          <td> 제목 <input type="text" name="title"/><br></td>                          </tr>                             <tr>                          <td> <textarea rows="10" cols="50" name="content" placeholder='1000자 이상 입력하세요.'></textarea>                          </tr>                          <tr>                              <td> 이름 <input type="text" name="name"/><br></td>                          </tr>                  </table>                  <center>                   <button>작성</button></a>                      </center>                </td>            </tr>       </table>      </form>      </body>  </html> |

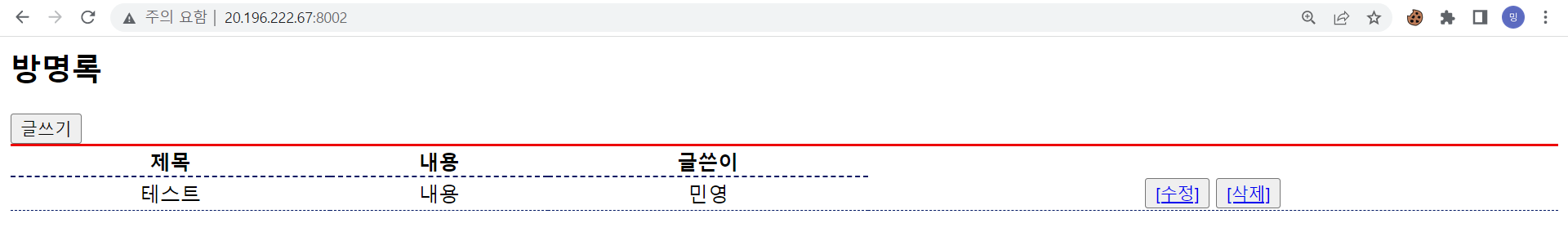
**Db.js**

|  |
| --- |
| const mysql = require("mysql");  const connection = mysql.createConnection( {      host: 'localhost',      user: 'minyeong3',      password: '2022Wnsldj!',      port: 3306,      database: 'minyeong3'  });  connection.connect(function(err){      if (err) console.log(err);      else console.log('Connected!');  });  module.exports = connection; |

****

**수정**

****

**삭제**

**글쓰기**

****

**Rest api 조사**

**Api란?**

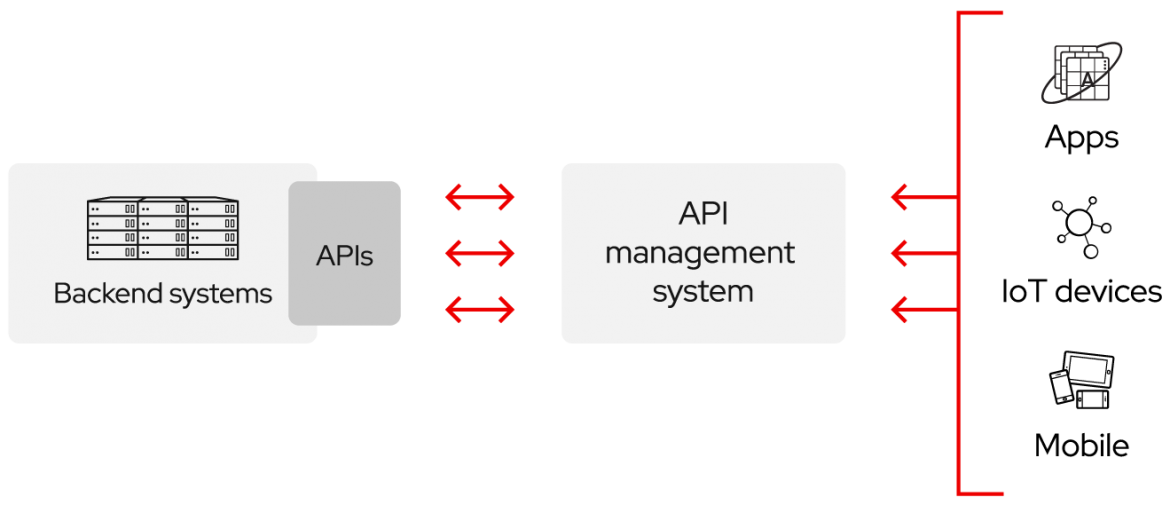
정의 및 프로토콜 집합을 사용하여 두 소프트웨어 구성 요소가 서로 통신할 수 있게 하는 메커니즘

컴퓨터나 시스템과 상호작용하여 정보를 검색하거나, 기능을 수행할 때 API는 사용자가 원하는 것을 시스템에 전달할 수 있게 지원함 → 시스템은 요청을 이해하고 이행함

**작동방식)**

+ 파싱: 어떤 페이지(문서, html 등)에서 내가 원하는 데이터를 특정 패턴이나 순서로 추출해 가공하는 것

한쪽 당사자가 특정한 방식으로 구성된 원격 요청을 보내면 다른 쪽 당사자의 소프트웨어가 이에 응답하는 방식임



자체 인프라를 연결하는 간소화된 방식

▷ 고객 및 다른 외부 사용자와의 데이터 공유를 허용하기도 함

리소스에 대한 액세스 범위를 넓히는 동시에 보안과 제어를 유지함

▷ 액세스 권한을 어떻게, 누구에게 제공할지 여부만 결정

구현방식을 알지 못하는 제품, 서비스와 통신 O

애플리케이션 개발을 간소화하여 시간과 비용을 절약 O

**Rest란?**

자원을 이름(자원의 표현)으로 구분하여 해당 자원의 상태(정보)를 주고 받는 모든 것

**자원** : 해당 소프트웨어가 관리하는 모든 것

**자원의 표현 :** 그 자원을 표현하기 위한 이름

<구성>

1. **자원(Resource) : HTTP URI**
2. **자원에 대한 행위(Verb) : HTTP Method**
3. **자원에 대한 행위의 내용 (Representations) : HTTP Message Pay Load**

HTTP 프로토콜의 인프라를 그대로 사용하므로 REST API 사용을 위한 별도의 인프라 필요 X

HTTP 표준 프로토콜에 따르는 모든 플랫폼에서 사용 O

**Rest api란?**

Rest의 원리를 따르는 API를 의미함

<규칙>

* Url은 동사보다 명사를, 대문자보다는 소문자 사용
* 마지막 슬래시 (/) 포함 X
* 언더바 대신 하이픈 사용
* 파일 확장자는 URL에 포함 X
* 행위를 포함 X

유연성이 뛰어남

개발자는 서버에서 직접 웹 애플리케이션이나 사이트 사용자에게 데이터를 제공해야 하는 모든 곳에서 Rest Api 사용함

<주요 구성요소>

* Customer: 통신을 시작하는 사용자 측(자신의 장치에서)에서 시작된 클라이언트 또는 프로그램.
* Servant : API를 기능 및 데이터에 대한 액세스로 사용하는 서버.
* Resources : 서버가 클라이언트에 전송하는 모든 콘텐츠(비디오, 텍스트, 사진).

