Nodejs 보고서 천경재

【문항1】교수 데이터 생성

`SQL Mapper`

```
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="ProfessorMapper">
    <insert id="insertItem">
        INSERT INTO professor (name, userid, position, sal, hiredate, comm,
        deptno) VALUES (#{name}, #{userid}, #{position}, #{sal}, #{hiredate}, #
        {comm}, #{deptno});
    </insert>
</mapper>
```

`Service Layer`

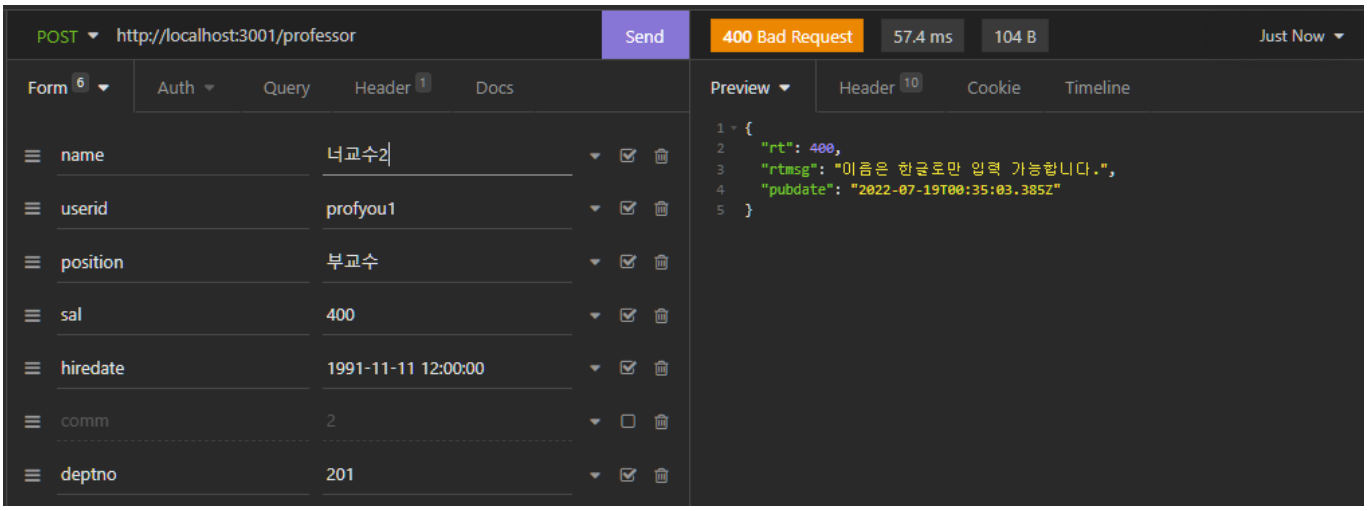
```
import mybatisMapper from "mybatis-mapper";
import DBPool from "../helper/DBPool.js";
import RuntimeException from "../exceptions/RuntimeException.js";

class ProfessorService {
    constructor() {
        mybatisMapper.createMapper([
            "./mappers/StudentMapper.xml",
            "./mappers/ProfessorMapper.xml",
        ]);
    }
    async addItem(params) {
        let dbcon = null;
        let data = null;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement(
                "ProfessorMapper",
                "insertItem",
                params
            );
            let [{ insertId, affectedRows }] = await dbcon.query(sql);

            if (affectedRows === 0) {
                throw new RuntimeException("저장된 데이터가 없습니다.");
            } else {
                console.log("------ 저장완료 -------");
                console.log(insertId);
            }

            sql = mybatisMapper.getStatement("ProfessorMapper", "selectItem", {
                profno: insertId,
            });
            let [result] = await dbcon.query(sql);

            if (result.length === 0) {
                throw new RuntimeException(
                    "저장된 데이터를 조회할 수 없습니다."
                );
            }

            data = result[0];
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }

        return data;
    }
}
```

Controller

```
import express from "express";
import regexHelper from "../helper/RegexHelper.js";
import professorService from "../services/ProfessorService.js";

const ProfessorController = () => {
    const url = "/professor";
    const router = express.Router();

    router.post("/professor", async (req, res, next) => {
        const name = req.post("name");
        const userid = req.post("userid");
        const position = req.post("position");
        const sal = req.post("sal");
        const hiredate = req.post("hiredate");
        const comm = req.post("comm");
        const deptno = req.post("deptno");

        try {
            regexHelper.value(name, "이름을 입력해주세요.");
                        regexHelper.kor(name, "이름은 한글로만 입력 가능합니다.");
            regexHelper.maxLength(name, 20, "이름은 최대 20자 까지 입력 가능합니
다.");
```

```
            regexHelper.value(userid, "아이디를 입력해주세요.");
            regexHelper.engNum(userid, "아이디는 영문과 숫자로만 입력 가능합니다.");
            regexHelper.maxLength(userid, 20, "아이디는 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(position, "직급을 입력해주세요.");
            regexHelper.maxLength(position, 20, "직급은 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(sal, "급여를 입력해주세요.");
            regexHelper.num(sal, "급여는 숫자로만 입력 가능합니다.");
            regexHelper.maxLength(sal, 20, "급여는 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(hiredate, "입사일을 입력해주세요.");
            regexHelper.date(hiredate, "입사일은 yyyy-MM-dd hh:mm:ss 형식에 맞게 입
력해주세요.");
            regexHelper.nullNum(comm, "보직수당은 미입력 또는 숫자로만 입력 가능합니
다.");
            regexHelper.value(deptno, "학과번호를 입력해주세요.");
        } catch (err) {
            return next(err);
        }

        let json = null;

        try {
            json = await professorService.addItem({
                name: name,
                userid: userid,
                position: position,
                sal: sal,
                hiredate: hiredate,
                comm: comm,
                deptno: deptno,
            });
        } catch (err) {
            return next(err);
        }

        res.sendResult({ item: json });
    });
    return router;
};
```

실행결과

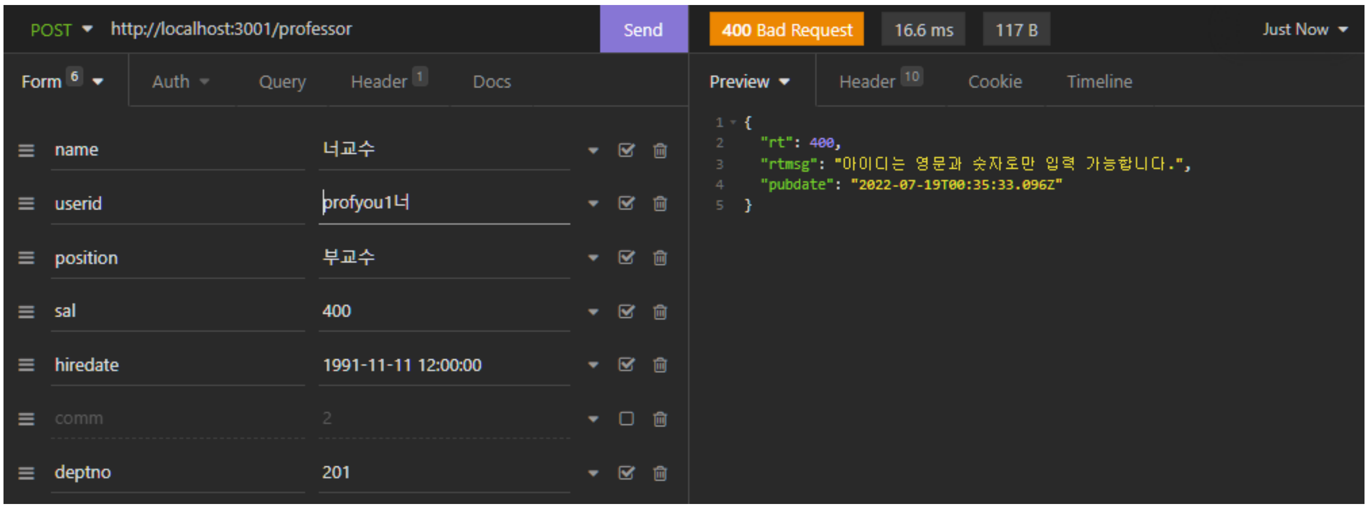


POST http://localhost:3001/professor Send | 400 Bad Request 57.4 ms 104 B Just Now

Form 6 | Auth | Query | Header 1 | Docs — Preview | Header 10 | Cookie | Timeline

name	너교수2
userid	profyou1
position	부교수
sal	400
hiredate	1991-11-11 12:00:00
comm	2
deptno	201

```
{
    "rt": 400,
    "rtmsg": "이름은 한글로만 입력 가능합니다.",
    "pubdate": "2022-07-19T00:35:03.385Z"
}
```

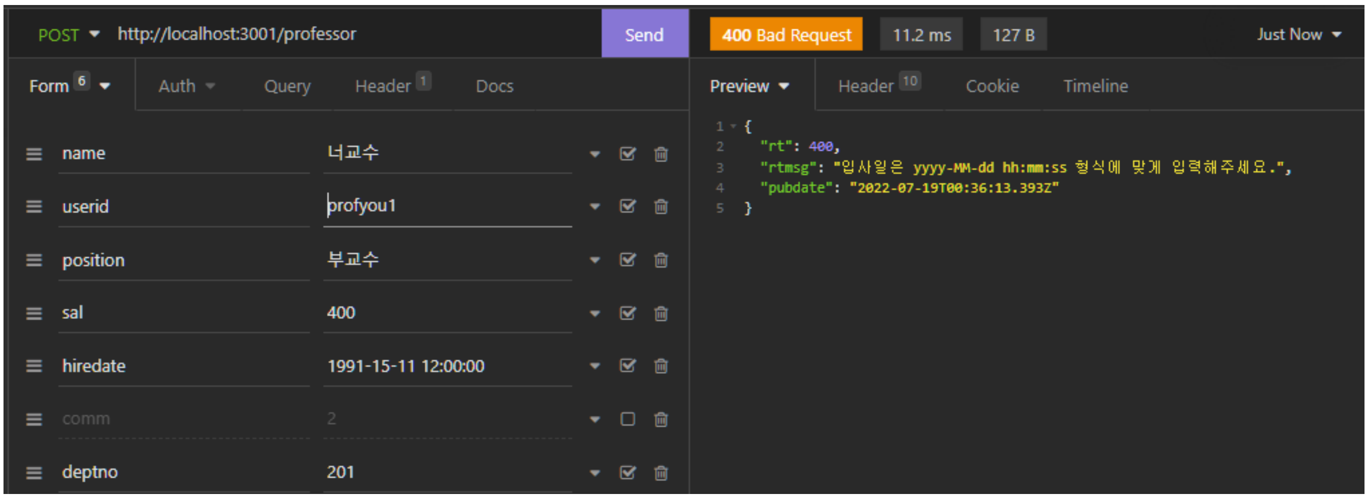


POST http://localhost:3001/professor Send | 400 Bad Request 16.6 ms 117 B Just Now

Form 6 | Auth | Query | Header 1 | Docs — Preview | Header 10 | Cookie | Timeline

name	너교수
userid	profyou1너
position	부교수
sal	400
hiredate	1991-11-11 12:00:00
comm	2
deptno	201

```
{
    "rt": 400,
    "rtmsg": "아이디는 영문과 숫자로만 입력 가능합니다.",
    "pubdate": "2022-07-19T00:35:33.096Z"
}
```

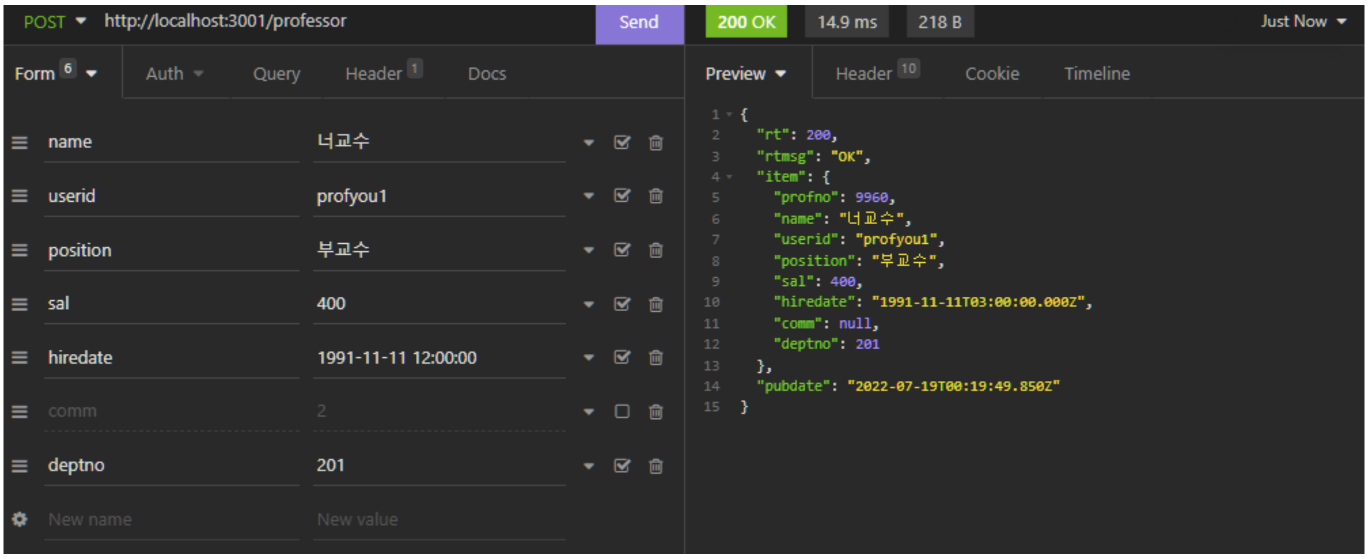


POST http://localhost:3001/professor Send | 400 Bad Request 11.2 ms 127 B Just Now

Form 6 | Auth | Query | Header 1 | Docs — Preview | Header 10 | Cookie | Timeline

name	너교수
userid	profyou1
position	부교수
sal	400
hiredate	1991-15-11 12:00:00
comm	2
deptno	201

```
{
    "rt": 400,
    "rtmsg": "입사일은 yyyy-MM-dd hh:mm:ss 형식에 맞게 입력해주세요.",
    "pubdate": "2022-07-19T00:36:13.393Z"
}
```

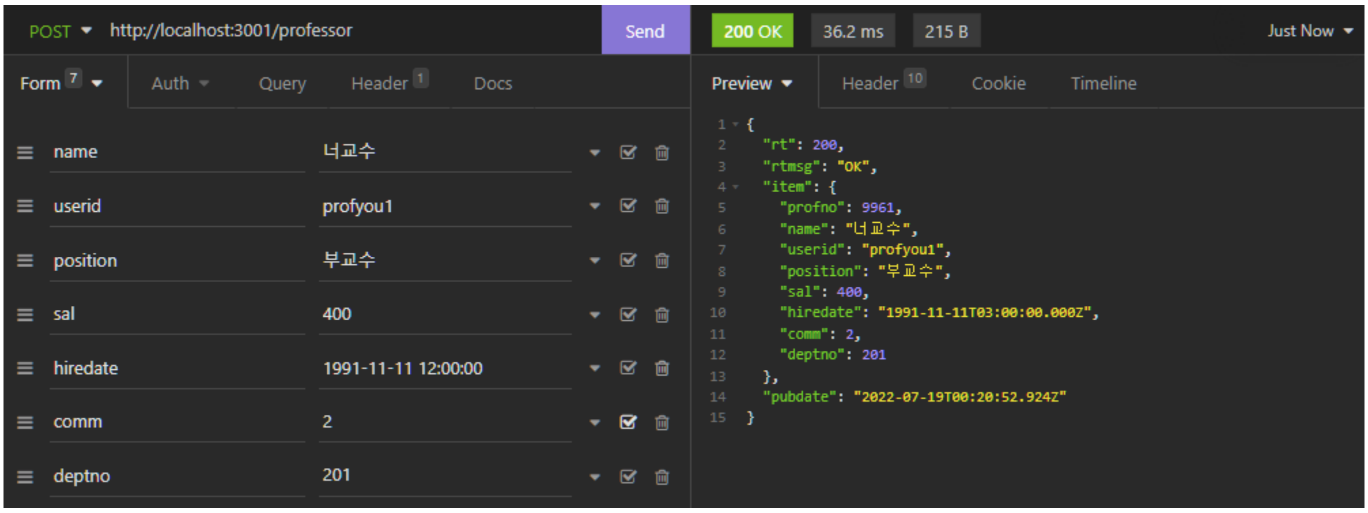


POST http://localhost:3001/professor

200 OK 14.9 ms 218 B

name	너교수
userid	profyou1
position	부교수
sal	400
hiredate	1991-11-11 12:00:00
comm	2
deptno	201
New name	New value

```
{
  "rt": 200,
  "rtmsg": "OK",
  "item": {
    "profno": 9960,
    "name": "너교수",
    "userid": "profyou1",
    "position": "부교수",
    "sal": 400,
    "hiredate": "1991-11-11T03:00:00.000Z",
    "comm": null,
    "deptno": 201
  },
  "pubdate": "2022-07-19T00:19:49.850Z"
}
```



POST http://localhost:3001/professor

200 OK 36.2 ms 215 B

name	너교수
userid	profyou1
position	부교수
sal	400
hiredate	1991-11-11 12:00:00
comm	2
deptno	201

```
{
  "rt": 200,
  "rtmsg": "OK",
  "item": {
    "profno": 9961,
    "name": "너교수",
    "userid": "profyou1",
    "position": "부교수",
    "sal": 400,
    "hiredate": "1991-11-11T03:00:00.000Z",
    "comm": 2,
    "deptno": 201
  },
  "pubdate": "2022-07-19T00:20:52.924Z"
}
```

【문항2】교수 데이터 수정

SQL Mapper

```
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="ProfessorMapper">
    <update id="updateItem">
        UPDATE professor SET name=#{name}, userid=#{userid}, position=#{position},
        sal=#{sal}, hiredate=#{hiredate}, comm=#{comm}, deptno=#{deptno}
        WHERE profno=#{profno};
    </update>

    <select id="selectJoin">
    SELECT
      p.profno, p.name, p.userid, p.position,
      p.sal, p.hiredate, p.comm, p.deptno, d.dname
    FROM professor AS `p`
    INNER JOIN department AS `d`
    ON p.deptno = d.deptno
    WHERE profno=#{profno}
    </select>
</mapper>
```

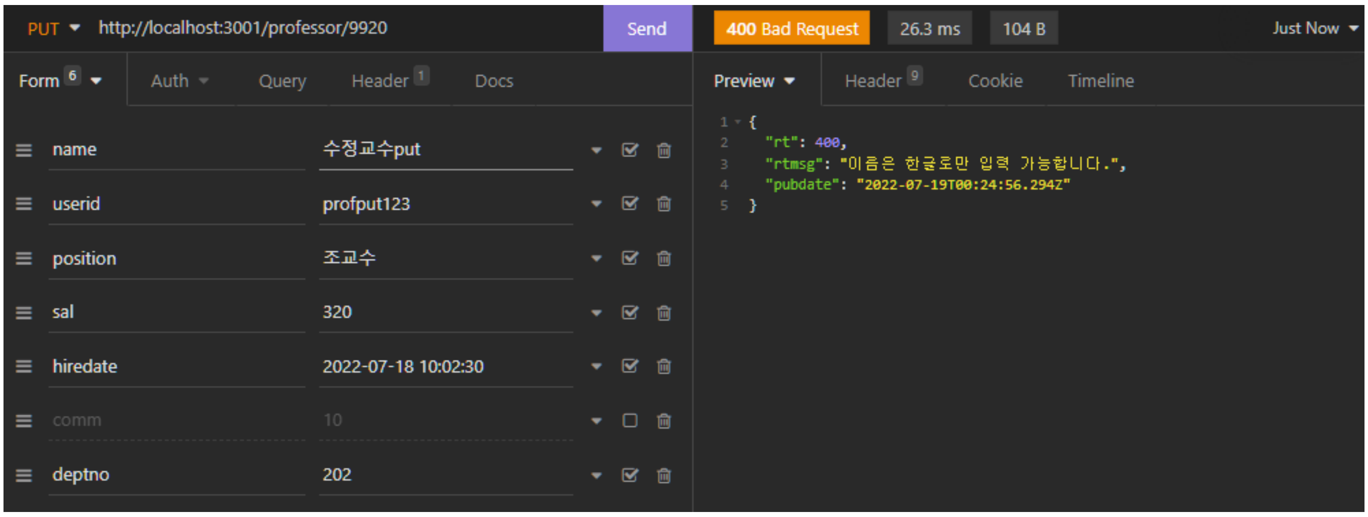
`Service Layer`

```
import mybatisMapper from "mybatis-mapper";
import DBPool from "../helper/DBPool.js";
import RuntimeException from "../exceptions/RuntimeException.js";

class ProfessorService {
    constructor() {
        mybatisMapper.createMapper([
            "./mappers/StudentMapper.xml",
            "./mappers/ProfessorMapper.xml",
        ]);
    }
    async editItem(params) {
        let dbcon = null;
        let data = null;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement("ProfessorMapper", "updateItem",
params);
            let [{ affectedRows }] = await dbcon.query(sql);

            if (affectedRows === 0) {
                throw new RuntimeException("수정된 데이터가 없습니다.");
            }

            sql = mybatisMapper.getStatement("ProfessorMapper", "selectJoin", {
profno: params.profno });
            let [result] = await dbcon.query(sql);

            if (affectedRows === 0) {
                throw new RuntimeException(
                    "수정된 데이터를 조회할 수 없습니다."
                );
            }

            data = result[0];
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }

        return data;
    }
}
```

Controller

```
import express from "express";
import regexHelper from "../helper/RegexHelper.js";
import professorService from "../services/ProfessorService.js";

const ProfessorController = () => {
    const url = "/professor";
    const router = express.Router();

    router.put(`${url}/:profno`, async (req, res, next) => {
        const profno = req.get('profno');
        const name = req.put("name");
        const userid = req.put("userid");
        const position = req.put("position");
        const sal = req.put("sal");
        const hiredate = req.put("hiredate");
        const comm = req.put("comm");
        const deptno = req.put("deptno");

        try {
            regexHelper.value(profno, "교수번호를 입력해주세요.");
            regexHelper.num(profno, "교수번호가 잘못되었습니다.");
            regexHelper.value(name, "이름을 입력해주세요.");
            regexHelper.kor(name, "이름은 한글로만 입력 가능합니다.");
            regexHelper.maxLength(name, 20, "이름은 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(userid, "아이디를 입력해주세요.");
            regexHelper.engNum(userid, "아이디는 영문과 숫자로만 입력 가능합니다.");
            regexHelper.maxLength(userid, 20, "아이디는 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(position, "직급을 입력해주세요.");
            regexHelper.maxLength(position, 20, "직급은 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(sal, "급여를 입력해주세요.");
            regexHelper.num(sal, "급여는 숫자로만 입력 가능합니다.");
            regexHelper.maxLength(sal, 20, "급여는 최대 20자 까지 입력 가능합니
다.");
            regexHelper.value(hiredate, "입사일을 입력해주세요.");
            regexHelper.date(hiredate, "입사일은 yyyy-MM-dd hh:mm:ss 형식에 맞게 입
력해주세요.");
            regexHelper.nullNum(comm, "보직수당은 미입력 또는 숫자로만 입력 가능합니
다.");
            regexHelper.value(deptno, "학과번호를 입력해주세요.");
        } catch (err) {
            return next(err);
        }

        let json = null;
        try {
            json = await professorService.editItem({
                profno: profno,
                name: name,
                userid: userid,
                position: position,
                sal: sal,
                hiredate: hiredate,
                comm: comm,
                deptno: deptno
            });
        } catch (err) {
            return next(err);
        }

        res.sendResult({ item: json });
    });
    return router;
};
```

실행결과

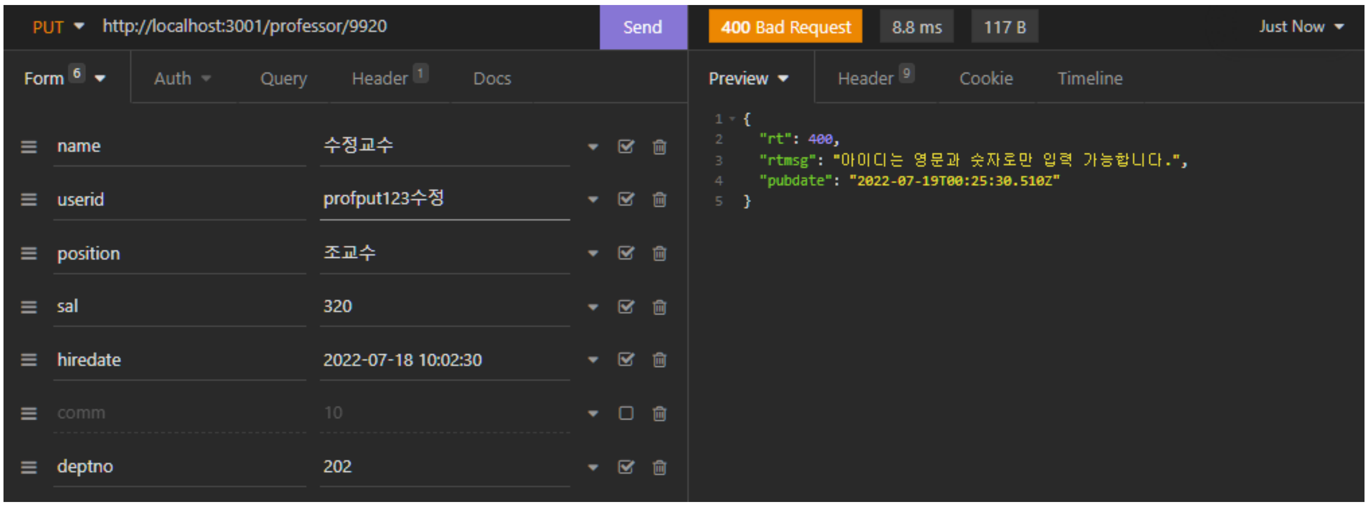


PUT http://localhost:3001/professor/9920 | Send | 400 Bad Request | 26.3 ms | 104 B | Just Now

Form 6 | Auth | Query | Header 1 | Docs | Preview | Header 9 | Cookie | Timeline

name	수정교수put
userid	profput123
position	조교수
sal	320
hiredate	2022-07-18 10:02:30
comm	10
deptno	202

```
{
    "rt": 400,
    "rtmsg": "이름은 한글로만 입력 가능합니다.",
    "pubdate": "2022-07-19T00:24:56.294Z"
}
```

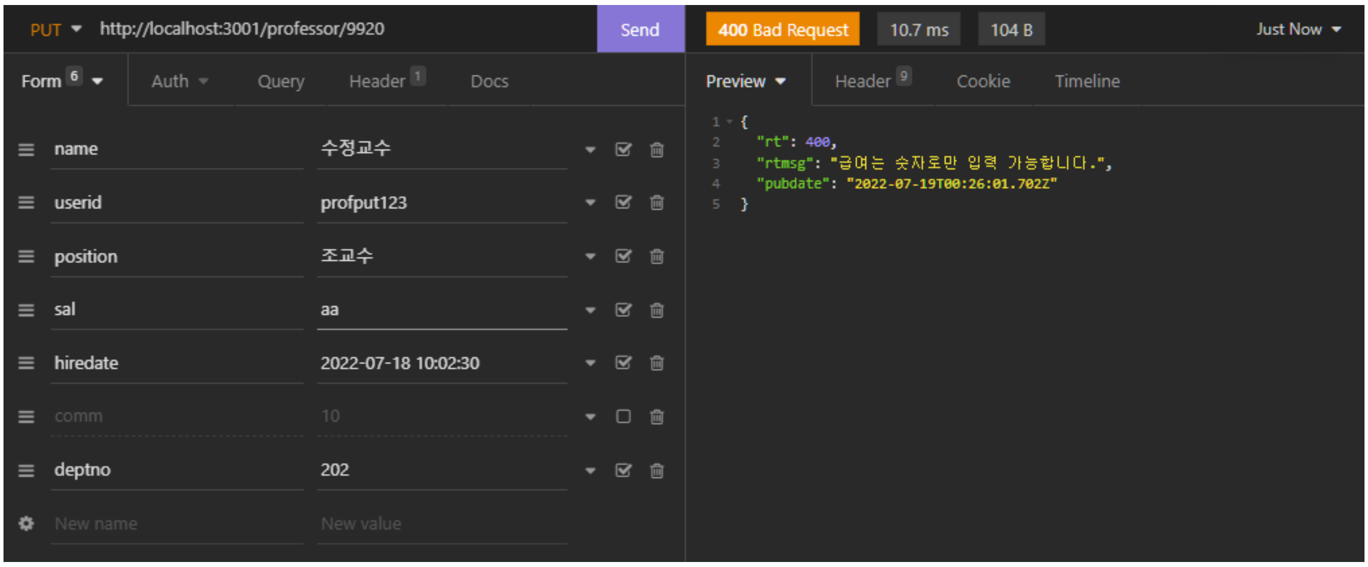


PUT http://localhost:3001/professor/9920 | Send | 400 Bad Request | 8.8 ms | 117 B | Just Now

Form 6 | Auth | Query | Header 1 | Docs | Preview | Header 9 | Cookie | Timeline

name	수정교수
userid	profput123수정
position	조교수
sal	320
hiredate	2022-07-18 10:02:30
comm	10
deptno	202

```
{
    "rt": 400,
    "rtmsg": "아이디는 영문과 숫자로만 입력 가능합니다.",
    "pubdate": "2022-07-19T00:25:30.510Z"
}
```



PUT http://localhost:3001/professor/9920 Send

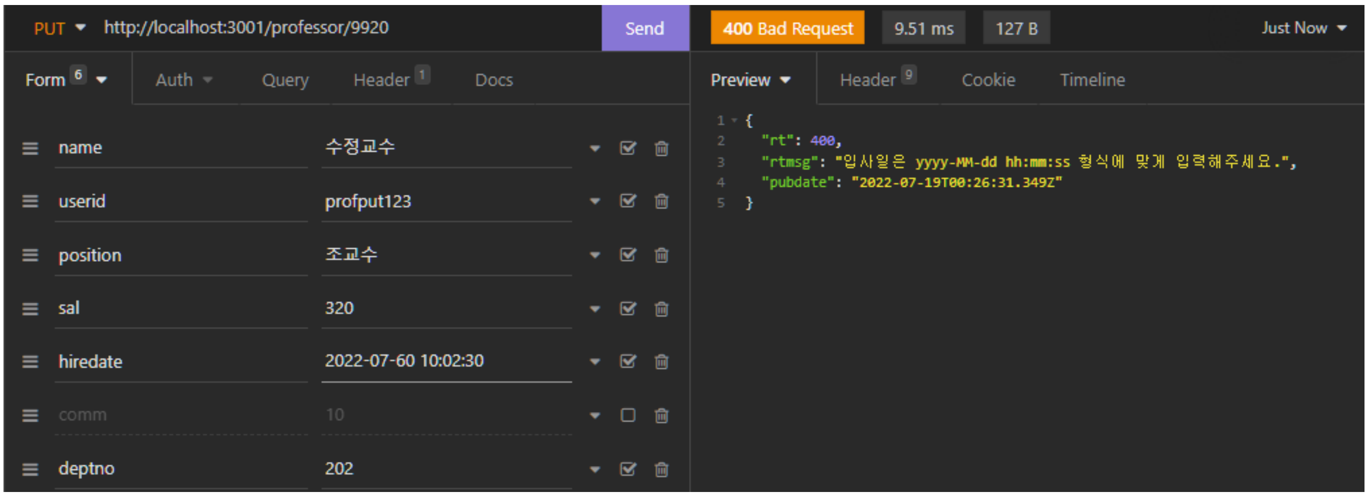
400 Bad Request 10.7 ms 104 B Just Now

Form 6 | Auth | Query | Header 1 | Docs

name	수정교수	☑
userid	profput123	☑
position	조교수	☑
sal	aa	☑
hiredate	2022-07-18 10:02:30	☑
comm	10	☐
deptno	202	☑
New name	New value	

Preview | Header 9 | Cookie | Timeline

```
{
  "rt": 400,
  "rtmsg": "급여는 숫자로만 입력 가능합니다.",
  "pubdate": "2022-07-19T00:26:01.702Z"
}
```



PUT http://localhost:3001/professor/9920 Send

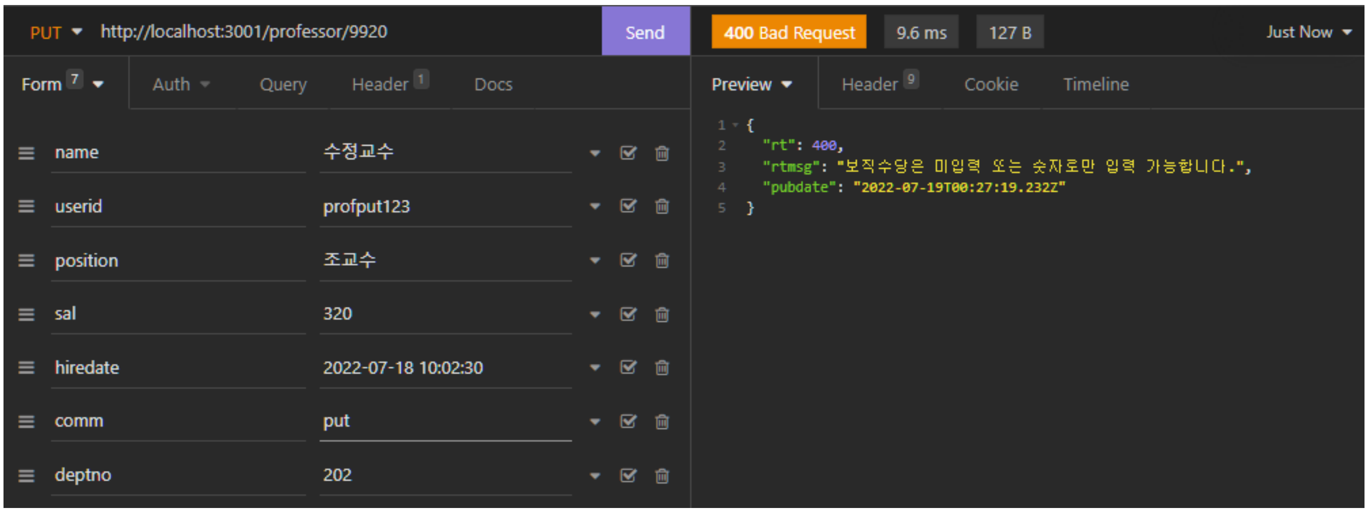
400 Bad Request 9.51 ms 127 B Just Now

Form 6 | Auth | Query | Header 1 | Docs

name	수정교수	☑
userid	profput123	☑
position	조교수	☑
sal	320	☑
hiredate	2022-07-60 10:02:30	☑
comm	10	☐
deptno	202	☑

Preview | Header 9 | Cookie | Timeline

```
{
  "rt": 400,
  "rtmsg": "입사일은 yyyy-MM-dd hh:mm:ss 형식에 맞게 입력해주세요.",
  "pubdate": "2022-07-19T00:26:31.349Z"
}
```



PUT http://localhost:3001/professor/9920 Send

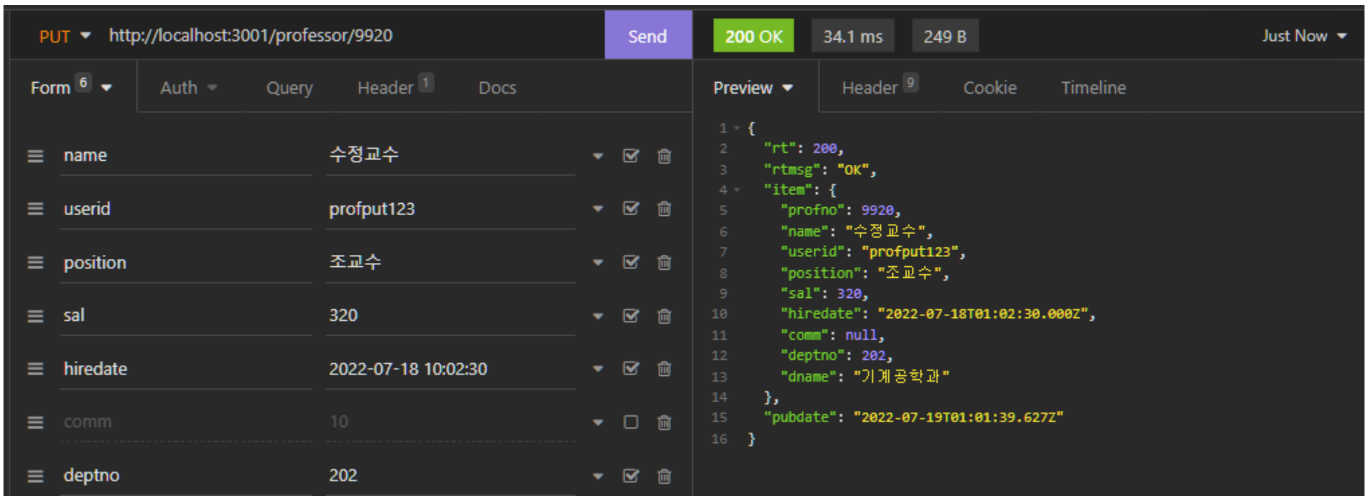
400 Bad Request 9.6 ms 127 B Just Now

Form 7 | Auth | Query | Header 1 | Docs

name	수정교수	☑
userid	profput123	☑
position	조교수	☑
sal	320	☑
hiredate	2022-07-18 10:02:30	☑
comm	put	☑
deptno	202	☑

Preview | Header 9 | Cookie | Timeline

```
{
  "rt": 400,
  "rtmsg": "보직수당은 미입력 또는 숫자로만 입력 가능합니다.",
  "pubdate": "2022-07-19T00:27:19.232Z"
}
```

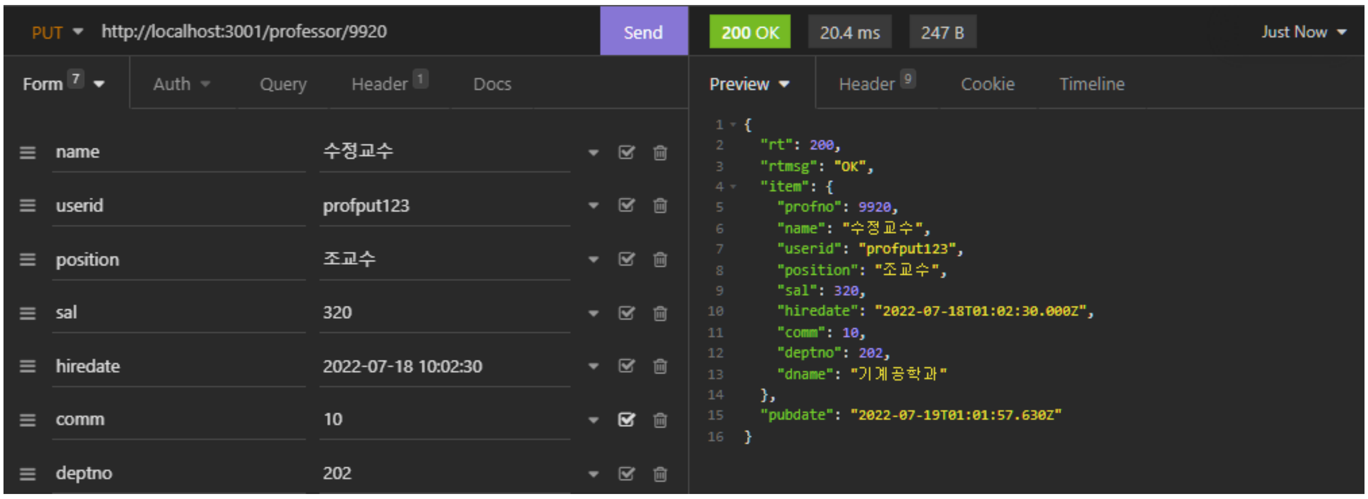


PUT http://localhost:3001/professor/9920

200 OK 34.1 ms 249 B

name	수정교수
userid	profput123
position	조교수
sal	320
hiredate	2022-07-18 10:02:30
comm	10
deptno	202

```
{
  "rt": 200,
  "rtmsg": "OK",
  "item": {
    "profno": 9920,
    "name": "수정교수",
    "userid": "profput123",
    "position": "조교수",
    "sal": 320,
    "hiredate": "2022-07-18T01:02:30.000Z",
    "comm": null,
    "deptno": 202,
    "dname": "기계공학과"
  },
  "pubdate": "2022-07-19T01:01:39.627Z"
}
```

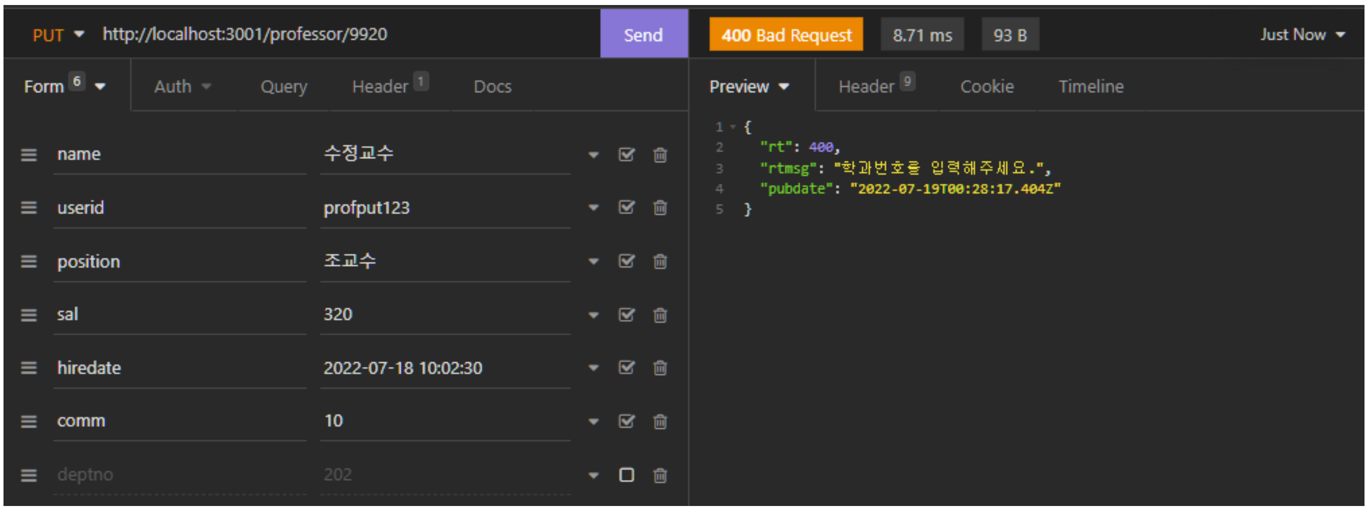


PUT http://localhost:3001/professor/9920

200 OK 20.4 ms 247 B

name	수정교수
userid	profput123
position	조교수
sal	320
hiredate	2022-07-18 10:02:30
comm	10
deptno	202

```
{
  "rt": 200,
  "rtmsg": "OK",
  "item": {
    "profno": 9920,
    "name": "수정교수",
    "userid": "profput123",
    "position": "조교수",
    "sal": 320,
    "hiredate": "2022-07-18T01:02:30.000Z",
    "comm": 10,
    "deptno": 202,
    "dname": "기계공학과"
  },
  "pubdate": "2022-07-19T01:01:57.630Z"
}
```



PUT http://localhost:3001/professor/9920

400 Bad Request 8.71 ms 93 B

name	수정교수
userid	profput123
position	조교수
sal	320
hiredate	2022-07-18 10:02:30
comm	10
deptno	202

```
{
  "rt": 400,
  "rtmsg": "학과번호를 입력해주세요.",
  "pubdate": "2022-07-19T00:28:17.404Z"
}
```

【문항3】교수 데이터 삭제

`SQL Mapper`

```
// 교수
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="ProfessorMapper">
    <delete id="deleteItem">
    DELETE FROM professor WHERE profno=#{profno};
    </delete>
</mapper>
```

```
// 학생
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="StudentMapper">

    <update id="profnoUpdate">
    UPDATE student SET profno=null WHERE profno=#{profno};
    </update>

</mapper>
```

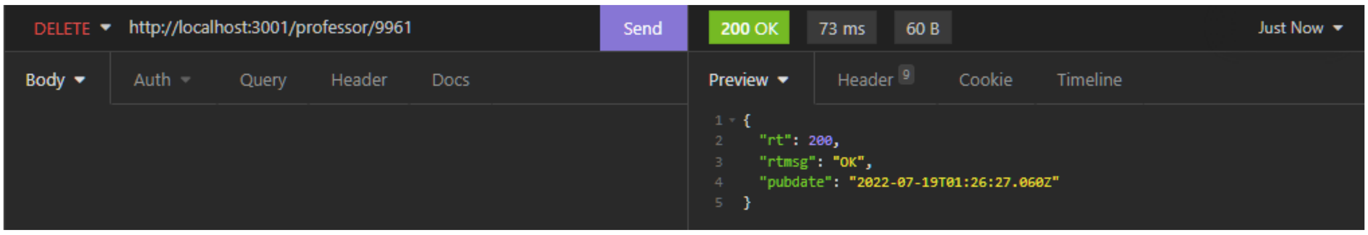
`Service Layer`

```
import mybatisMapper from "mybatis-mapper";
import DBPool from "../helper/DBPool.js";
import RuntimeException from "../exceptions/RuntimeException.js";

class ProfessorService {
    constructor() {
        mybatisMapper.createMapper([
            "./mappers/StudentMapper.xml",
            "./mappers/ProfessorMapper.xml",
        ]);
    }
    async deleteItem(params) {
        let dbcon = null;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement("StudentMapper", "profnoUpdate",
params);
            let [{ affectedRows }] = await dbcon.query(sql);

            sql = mybatisMapper.getStatement("ProfessorMapper", "deleteItem",
params);
            [{ affectedRows }] = await dbcon.query(sql);

            if (affectedRows === 0) {
                throw new RuntimeException("삭제된 데이터가 없습니다.");
            }
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }
    }
}
```

`Controller`

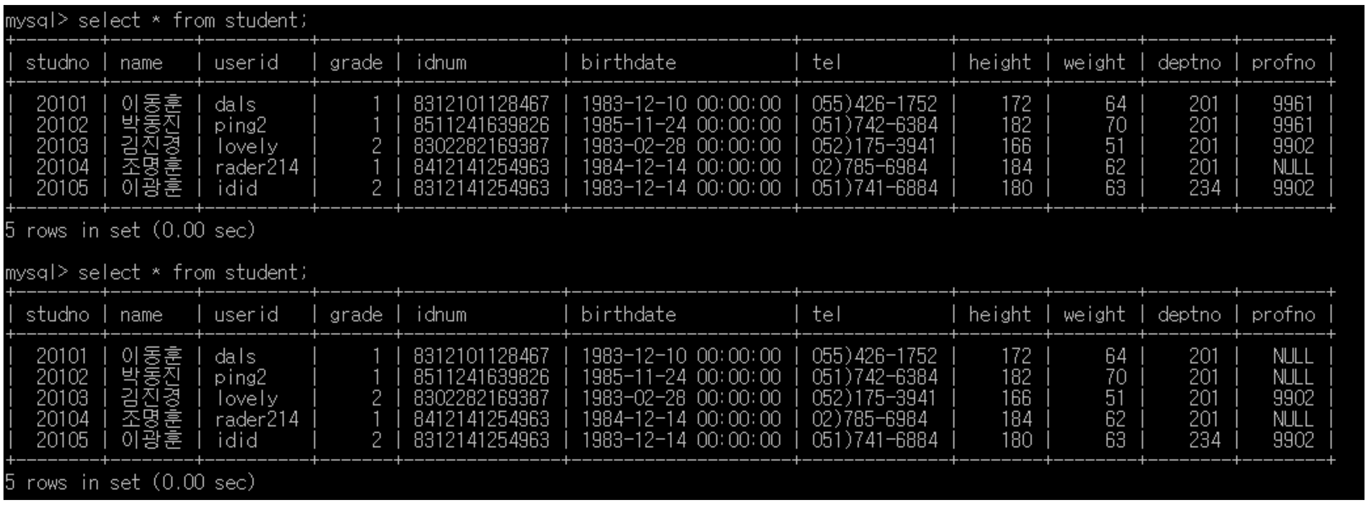
```
import express from "express";
import regexHelper from "../helper/RegexHelper.js";
import professorService from "../services/ProfessorService.js";

const ProfessorController = () => {
    const url = "/professor";
    const router = express.Router();

    router.delete(`${url}/:profno`, async (req, res, next) => {
        const profno = req.get('profno');
        try {
            regexHelper.value(profno, "교수번호를 입력해주세요.");
            regexHelper.num(profno, "교수번호가 잘못되었습니다.");
        } catch (err) {
            return next(err);
        }

        try {
            await professorService.deleteItem({
                profno: profno
            });
        } catch (err) {
            return next(err);
        }

        res.sendResult();
    });

    return router;
};
```

실행결과



http://localhost:3001/professor/9961

```
{
  "rt": 200,
  "rtmsg": "OK",
  "pubdate": "2022-07-19T01:26:27.060Z"
}
```



```
mysql> select * from student;
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
| studno | name   | userid    | grade | idnum         | birthdate           | tel          | height | weight | deptno | profno |
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
|  20101 | 이동훈 | dals      |     1 | 8312101128467 | 1983-12-10 00:00:00 | 055)426-1752 |    172 |     64 |    201 |   9961 |
|  20102 | 박동진 | ping2     |     1 | 8511241639826 | 1985-11-24 00:00:00 | 051)742-6384 |    182 |     70 |    201 |   9961 |
|  20103 | 김진경 | lovely    |     2 | 8302282169387 | 1983-02-28 00:00:00 | 052)175-3941 |    166 |     51 |    201 |   9902 |
|  20104 | 조명훈 | rader214  |     1 | 8412141254963 | 1984-12-14 00:00:00 | 02)785-6984  |    184 |     62 |    201 |   NULL |
|  20105 | 이광훈 | idid      |     2 | 8312141254963 | 1983-12-14 00:00:00 | 051)741-6884 |    180 |     63 |    234 |   9902 |
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
5 rows in set (0.00 sec)

mysql> select * from student;
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
| studno | name   | userid    | grade | idnum         | birthdate           | tel          | height | weight | deptno | profno |
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
|  20101 | 이동훈 | dals      |     1 | 8312101128467 | 1983-12-10 00:00:00 | 055)426-1752 |    172 |     64 |    201 |   NULL |
|  20102 | 박동진 | ping2     |     1 | 8511241639826 | 1985-11-24 00:00:00 | 051)742-6384 |    182 |     70 |    201 |   NULL |
|  20103 | 김진경 | lovely    |     2 | 8302282169387 | 1983-02-28 00:00:00 | 052)175-3941 |    166 |     51 |    201 |   9902 |
|  20104 | 조명훈 | rader214  |     1 | 8412141254963 | 1984-12-14 00:00:00 | 02)785-6984  |    184 |     62 |    201 |   NULL |
|  20105 | 이광훈 | idid      |     2 | 8312141254963 | 1983-12-14 00:00:00 | 051)741-6884 |    180 |     63 |    234 |   9902 |
+--------+--------+-----------+-------+---------------+---------------------+--------------+--------+--------+--------+--------+
5 rows in set (0.00 sec)
```

【문항4】교수 데이터 조회

`SQL Mapper`

```xml
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="ProfessorMapper">
    <select id="selectJoin">
    SELECT
      p.profno, p.name, p.userid, p.position,
      p.sal, p.hiredate, p.comm, p.deptno, d.dname
    FROM professor AS `p`
    INNER JOIN department AS `d`
    ON p.deptno = d.deptno
    WHERE profno=#{profno}
    </select>
</mapper>
```

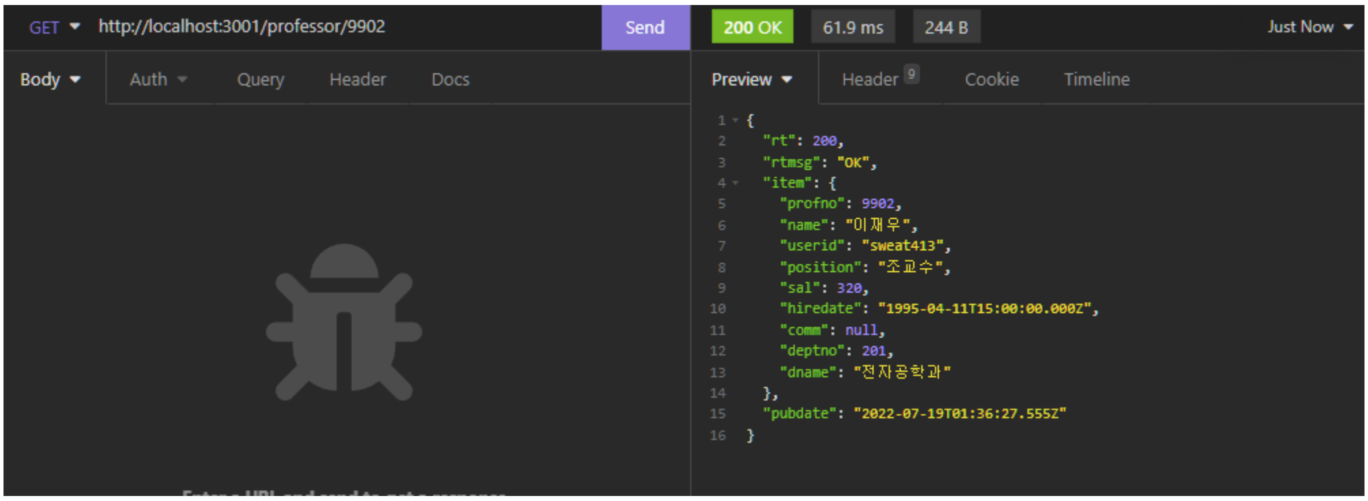
`Service Layer`

```js
import mybatisMapper from "mybatis-mapper";
import DBPool from "../helper/DBPool.js";
import RuntimeException from "../exceptions/RuntimeException.js";

class ProfessorService {
    constructor() {
        mybatisMapper.createMapper([
            "./mappers/StudentMapper.xml",
            "./mappers/ProfessorMapper.xml",
        ]);
    }
    async getItem(params) {
        let dbcon = null;
        let data = null;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement("ProfessorMapper", "selectJoin",
params);
            let [result] = await dbcon.query(sql);

            if (result.length === 0) {
                throw new RuntimeException("조회된 데이터가 없습니다.");
            }

            data = result[0];
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }

        return data;
    }
}
```

`Controller`

```
import express from "express";
import regexHelper from "../helper/RegexHelper.js";
import professorService from "../services/ProfessorService.js";

const ProfessorController = () => {
    const url = "/professor";
    const router = express.Router();

    router.get(`${url}/:profno`, async (req, res, next) => {
        const profno = req.get("profno");

        try {
            regexHelper.value(profno, "교수번호를 입력해주세요.");
            regexHelper.num(profno, "교수번호가 잘못되었습니다.");
        } catch (err) {
            return next(err);
        }

        let json = null;

        try {
            json = await professorService.getItem({
                profno: profno,
            });
        } catch (err) {
            return next(err);
        }

        res.sendResult({ item: json });
    });

    return router;
};
```

실행결과



```
{
  "rt": 200,
  "rtmsg": "OK",
  "item": {
    "profno": 9902,
    "name": "이재우",
    "userid": "sweat413",
    "position": "조교수",
    "sal": 320,
    "hiredate": "1995-04-11T15:00:00.000Z",
    "comm": null,
    "deptno": 201,
    "dname": "전자공학과"
  },
  "pubdate": "2022-07-19T01:36:27.555Z"
}
```

【문항5】교수 데이터 목록 조회

`SQL Mapper`

```
<?xml version="1.0" encoding="URF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="ProfessorMapper">
    <select id="selectList">
    select
      p.profno, p.name, p.userid, p.position,
      p.sal, p.hiredate, p.comm, p.deptno, d.dname
    FROM professor AS `p`
    INNER JOIN department AS `d`
    ON p.deptno = d.deptno
        <where>
            <if test="name != null and name != ''">
            name LIKE concat('%', #{name}, '%')
            </if>
        </where>
    ORDER BY profno DESC
        <if test="listCount > 0">
        LIMIT ${offset}, ${listCount}
        </if>
    </select>

    <select id="selectCountAll">
    SELECT COUNT(*) AS `cnt` FROM professor
        <where>
            <if test="name != null and name != ''">
            name LIKE concat('%', #{name}, '%')
            </if>
        </where>
    </select>
</mapper>
```

Service Layer

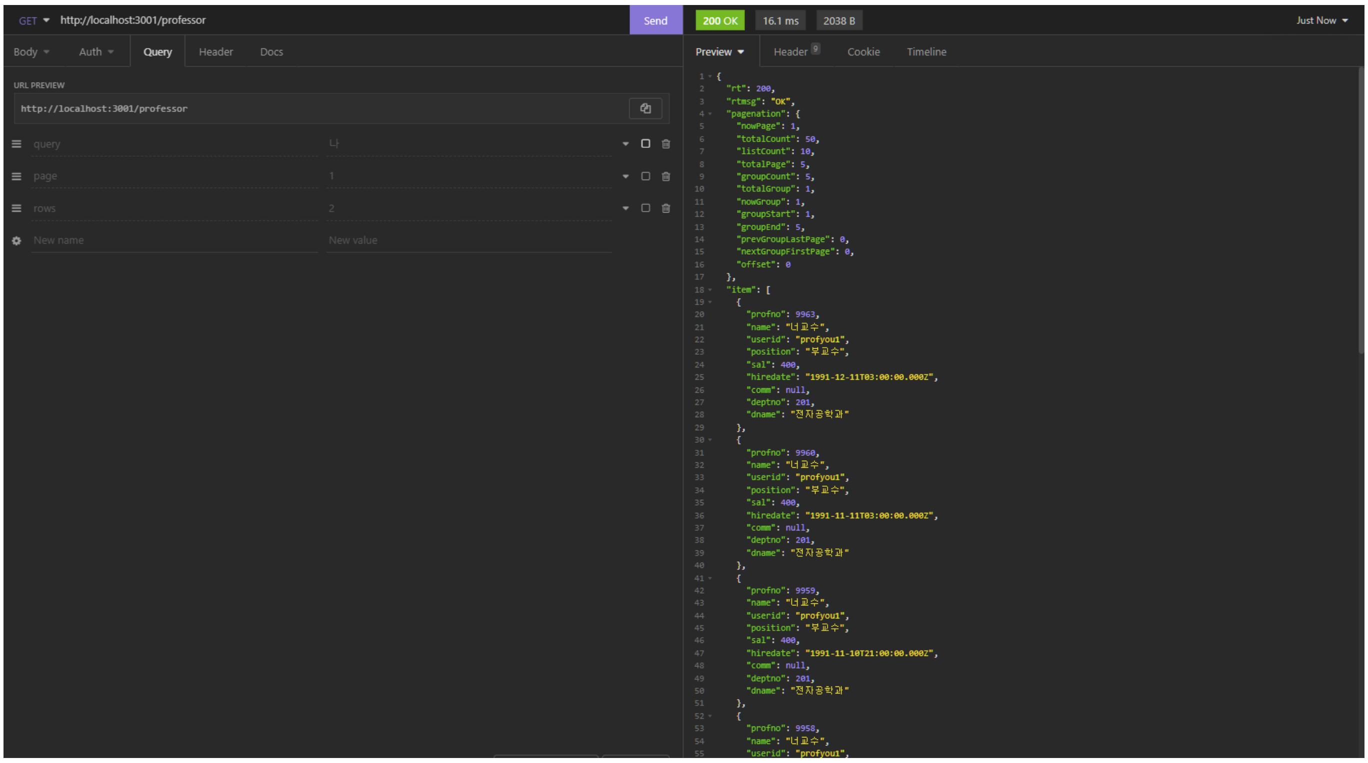
```
import mybatisMapper from "mybatis-mapper";
import DBPool from "../helper/DBPool.js";
import RuntimeException from "../exceptions/RuntimeException.js";

class ProfessorService {
    constructor() {
        mybatisMapper.createMapper([
            "./mappers/StudentMapper.xml",
            "./mappers/ProfessorMapper.xml",
        ]);
    }

    async getCount(params) {
        let dbcon = null;
        let cnt = 0;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement("ProfessorMapper",
"selectCountAll", params);
            let [result] = await dbcon.query(sql);

            if (result.length > 0) {
                cnt = result[0].cnt;
            }
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }

        return cnt;
    }

    async getList(params) {
        let dbcon = null;
        let data = null;

        try {
            dbcon = await DBPool.getConnection();

            let sql = mybatisMapper.getStatement("ProfessorMapper", "selectList",
params);
            let [result] = await dbcon.query(sql);

            if (result.length === 0) {
                throw new RuntimeException("조회된 데이터가 없습니다.");
            }

            data = result;
        } catch (err) {
            throw err;
        } finally {
            if (dbcon) {
                dbcon.release();
            }
        }

        return data;
    }
}
```

Controller

```
import express from "express";
import regexHelper from "../helper/RegexHelper.js";
import { pagenation } from "../helper/UtilHelper.js";
import professorService from "../services/ProfessorService.js";

const ProfessorController = () => {
    const url = "/professor";
    const router = express.Router();

    router.get(url, async (req, res, next) => {

        const query = req.get('query');
        const page = req.get('page', 1);
        const rows = req.get('rows', 5);

        const params = {};
        if (query) {
            params.name = query;
        }

        let json = null;
        let pageInfo = null;
```

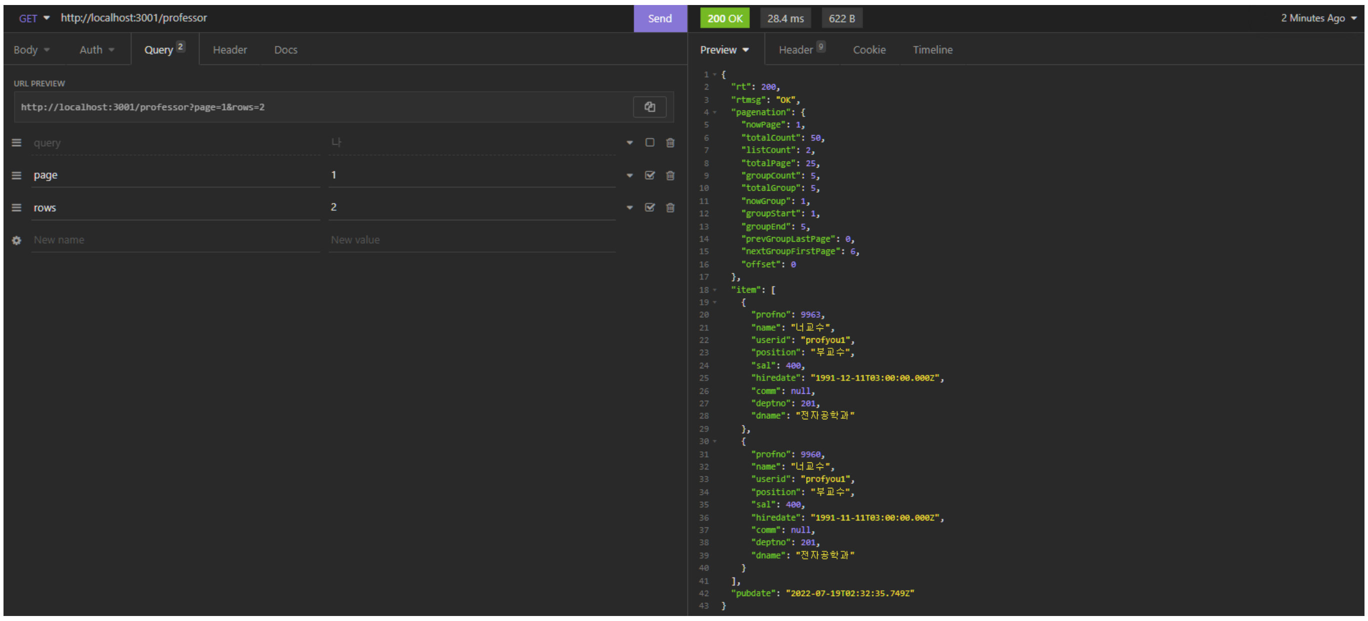
```
            try {
                const totalCount = await professorService.getCount(params);
                pageInfo = pagination(totalCount, page, rows);

                params.offset = pageInfo.offset;
                params.listCount = pageInfo.listCount;
                json = await professorService.getList(params);
            } catch (err) {
                return next(err);
            }

            res.sendResult({ pagination: pageInfo, item: json });
        });

        return router;
    };
```

실행결과



```
http://localhost:3001/professor
query  나
page   1
rows   2
```

```
{
  "rt": 200,
  "rtmsg": "OK",
  "pagenation": {
    "nowPage": 1,
    "totalCount": 50,
    "listCount": 10,
    "totalPage": 5,
    "groupCount": 5,
    "totalGroup": 1,
    "nowGroup": 1,
    "groupStart": 1,
    "groupEnd": 5,
    "prevGroupLastPage": 0,
    "nextGroupFirstPage": 0,
    "offset": 0
  },
  "item": [
    {
      "profno": 9963,
      "name": "너교수",
      "userid": "profyou1",
      "position": "부교수",
      "sal": 400,
      "hiredate": "1991-12-11T03:00:00.000Z",
      "comm": null,
      "deptno": 201,
      "dname": "전자공학과"
    },
    {
      "profno": 9960,
      "name": "너교수",
      "userid": "profyou1",
      "position": "부교수",
      "sal": 400,
      "hiredate": "1991-11-11T03:00:00.000Z",
      "comm": null,
      "deptno": 201,
      "dname": "전자공학과"
    },
    {
      "profno": 9959,
      "name": "너교수",
      "userid": "profyou1",
      "position": "부교수",
      "sal": 400,
      "hiredate": "1991-11-10T21:00:00.000Z",
      "comm": null,
      "deptno": 201,
      "dname": "전자공학과"
    },
    {
      "profno": 9958,
      "name": "너교수",
      "userid": "profyou1",
```

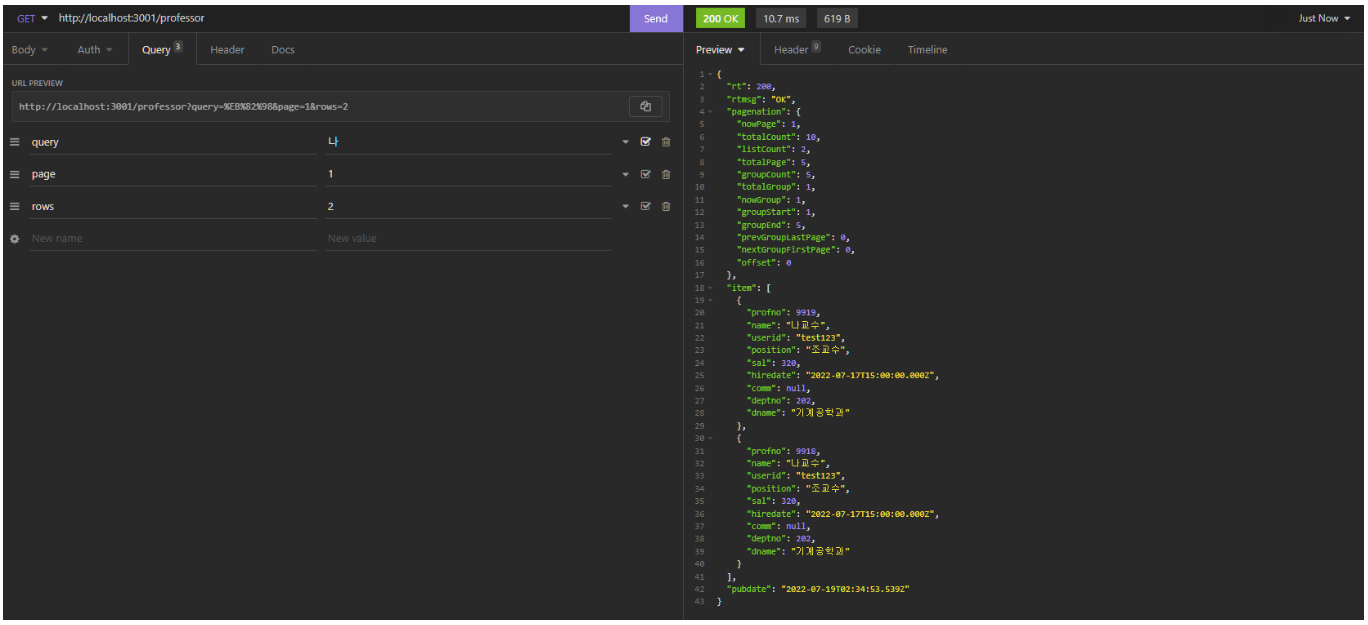


GET http://localhost:3001/professor

```
http://localhost:3001/professor?page=1&rows=2
```

query	나
page	1
rows	2
New name	New value

```
{
  "rt": 200,
  "rtmsg": "OK",
  "pagenation": {
    "nowPage": 1,
    "totalCount": 50,
    "listCount": 2,
    "totalPage": 25,
    "groupCount": 5,
    "totalGroup": 5,
    "nowGroup": 1,
    "groupStart": 1,
    "groupEnd": 5,
    "prevGroupLastPage": 0,
    "nextGroupFirstPage": 6,
    "offset": 0
  },
  "item": [
    {
      "profno": 9963,
      "name": "너교수",
      "userid": "profyou1",
      "position": "부교수",
      "sal": 400,
      "hiredate": "1991-12-11T03:00:00.000Z",
      "comm": null,
      "deptno": 201,
      "dname": "전자공학과"
    },
    {
      "profno": 9960,
      "name": "너교수",
      "userid": "profyou1",
      "position": "부교수",
      "sal": 400,
      "hiredate": "1991-11-11T03:00:00.000Z",
      "comm": null,
      "deptno": 201,
      "dname": "전자공학과"
    }
  ],
  "pubdate": "2022-07-19T02:32:35.749Z"
}
```



GET http://localhost:3001/professor

```
http://localhost:3001/professor?query=%EB%82%98&page=1&rows=2
```

query	나
page	1
rows	2
New name	New value

```
{
  "rt": 200,
  "rtmsg": "OK",
  "pagenation": {
    "nowPage": 1,
    "totalCount": 10,
    "listCount": 2,
    "totalPage": 5,
    "groupCount": 5,
    "totalGroup": 1,
    "nowGroup": 1,
    "groupStart": 1,
    "groupEnd": 5,
    "prevGroupLastPage": 0,
    "nextGroupFirstPage": 0,
    "offset": 0
  },
  "item": [
    {
      "profno": 9919,
      "name": "나교수",
      "userid": "test123",
      "position": "조교수",
      "sal": 320,
      "hiredate": "2022-07-17T15:00:00.000Z",
      "comm": null,
      "deptno": 202,
      "dname": "기계공학과"
    },
    {
      "profno": 9918,
      "name": "나교수",
      "userid": "test123",
      "position": "조교수",
      "sal": 320,
      "hiredate": "2022-07-17T15:00:00.000Z",
      "comm": null,
      "deptno": 202,
      "dname": "기계공학과"
    }
  ],
  "pubdate": "2022-07-19T02:34:53.539Z"
}
```