UVC
Ubiquitous Via Connect

웹 개발의 역사

웹 개발자가 겪은 현장의 역사

History of web development

History of web development

UVC
Ubiquitous Via Connect

왜 개발역사를 알아야 하는가?

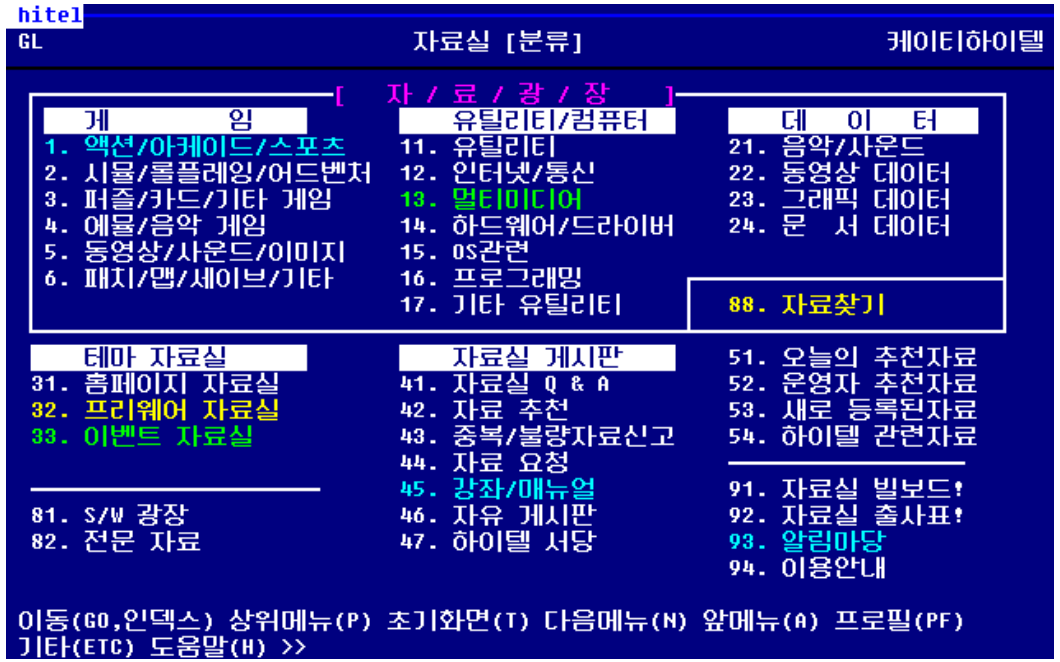
- 프론트와 백엔드가 왜 분리 되었는가? (그럼 합쳐져 있었다는 것인가?)
- 다음 중 (커리어의 건강을 위해) 달리기를 해야 할 상황에 처한 개발자는?
  - 회사에 취업했는데 java spring 유지보수를 맡기네요??
  - nodejs에서 프로시저 호출해서 쓰는데요?? (대부분의 비즈니스 로직이 프로시저에서 돌아요..)
  - 우리 회사는 vue를 쓰기는 하는데 jsp에서 사용하네요??
  - 야 우리 회사는 python에서 vue 돌리던데?? 최신 기술임?
  - vue배웠더니 react 시키는데요???
  - vue배웠더니 angular 시키는데요??
  - 나는 지금 nodejs에서 jQuery하느라 미칠 지경입니다.
  - 회사에서 php, asp를 맡아주면 잘릴 걱정이 없다는 데요?
  - 클라우드만 관리하고 있는데 이거 개발자가 하는 거 맞나요??

- 초보 개발자의 경우 해당 기술이 어느 패러다임에 속하는지 알기 어렵다. → 역사를 통해 알아보자

History of web development

1990년대의 PC통신 (선사시대)

- (한국에서)web의 대중화 이전에 PC통신의 시절
  - 하이텔, 나우누리, 천리안 (go xxx)
- 모뎀을 달아서 유선 전화선에 연결 후 실제로 전화를 거는 시스템.
  - 전화가 걸리는 비프 음 발생(01410)
  - 전화요금 폭탄(불금 밤에는 6시간 통화, 전화요금 10~20만원)
  - 집전화가 계속 통화중인 상태라 다른 전화를 받을 수 없음.
- 리눅스와 비슷한 사용 방식 (명령어(단축키)를 통해 제어 됨)
- 채팅으로 인한 신조어의 탄생
  - 안냐세요, 하이루, 하이, 방가방가
  - 고등학생 → 고등어 → 고딩, 대딩, 직딩, 중딩("너희가 중딩을 아느냐")
  - 번개



언어학자들은 "신세대 은어는 그들만의 사고방식과 유대감의 표현이지만 대부분이 기존의 언어체계를 무시한 것들이어서 정상적인 언어생활에 악영향을 미칠 수 있다"고 우려하고 있다.

매일신문 입력 1999-06-11 14:15:00

History of web development

2000년대 웹 개발 (고대)

- 초고속 인터넷 보급사업으로 인해 PC통신 대신 web으로 서비스가 이동하기 시작.
- asp, jsp, php같은 "스크립트언어"를 통해 동적으로 html을 만드는 방식 사용 (내 인생을 바꾼 html, 비전공자 파이팅!)
- 웹 개발자: 스크립트 언어를 통해 비즈니스 로직을 만들고 DB 처리를 하는 사람
- 웹 디자이너: 화면 구성을 만들고 이미지를 만드는 사람(html로의 변환은 하지 않음)
- DBA: DB를 설치하고 설계/운영 하는 사람
- SE: 시스템을 구축하고 운영 하는 사람(유닉스/리눅스 잘 다루는 사람, 네트워크 구축 가능)
- "웹마스터"라는 것이 유행했음.
- 찰떡궁합(과 개발자의 인식)
    - Unix(sun Solaris), Oracle, Apache, WebLogic, JSP (최고의 성능, 최고의 비용, 어깨가 으쓱)
    - Windows NT, MS-SQL, IIS, ASP (해킹에 가장 취약, 코딩은 쉬운데 서버가 뻗어)
    - Linux, MySQL, Apache, PHP (가장 저렴한 조합, 연봉도 제일 저렴해)

DB의 연결 방법

- 주로 RDB를 사용함
- 직접 쿼리 작성 방식: Database connector인 jdbc나 odbc를 사용함

History of web development

2010년대 웹 개발 (중세)

- java spring의 천하 통일. OOP 광풍. 프레임워크 도입의 시작
- 웹개발로 대동단결: C/S프로그램은 지양하고 주요 시스템들이 웹으로 개발되고 있음.
- 웹 개발자: 웹 개발 이라기 보다는 그냥 자바 개발자로 퉁 쳤음.
- 웹 디자이너: jQuery를 잘 다루어야 좋은 대접. (퍼블리셔 개념의 출현). 점점 사라짐(개발자가 땜빵)
- ~~DBA~~: DBA는 거의 사라지는 중. (비싸서 안 뽑고 개발자가 땜빵)
- SE: 아직 존재하긴 하나 클라우드때문에 사라지는 중 (역시 개발자가 땜빵)
- 주요 구성 방식
    - Linux, MySQL, Apache, Tomcat, java servlet(java spring), jQuery

DB의 연결 방법

- 간접 쿼리 방식(ORM으로 가기 위한 중간 단계) – 쿼리도 아니고 ORM도 아닌 어중간한 상태.
- SQL Mapper사용: MyBatis (쿼리 같은 문법을 사용해서 쿼리를 생성함)

History of web development

2020년대 웹 개발 (현대)

- nodejs, python의 대 약진, jQuery는 역사의 뒤안길로 (공교롭게도 둘 다 interpreter언어)
- Back-end, Front-end의 개념으로 개발이 나뉘어짐
- 클라우드를 이용한 확장성 중요(scale in/out) – (일년내내 아무도 접속 안 하다가 단 며칠동안 전세계 트래픽이 몰리는 곳은? (링크))
- 웹 개발자: Back-end개발과 Front-end개발이 나뉨(물론 Full stack으로 묶여서 혼자 다 함)
- ~~웹 디자이너~~: 퍼블리셔로 진화됨. 하지만 안 뽑아 줌(개발자가 대신 함)
- ~~DBA~~: 볼 수도 없지만 안 뽑아 줌. (대신 빅데이터, 데이터 분석가 등 완전 박사 급 전문화..)
- ~~SE~~: DevOps라는 이름으로 개발자가 대신 함. 클라우드도 잘 다루면 됨(그게 쉽냐?)
- 결론: 개발자 니가 다 해라 (풀스택 + 데브옵스) (시장에서 살아 남았지만 사라진 사람들의 몫은 전부 다 차지 함)
    - 다행히 데이터 분석까지는 개발자한테 맡기지는 않음 (최소 석사 이상은 해야...)
- 주요 구성 방식
    - Linux(Cloud or VM), MySQL, Nginx, Front-end(react, vuejs), Back-end(nodejs, python)

DB의 연결 방법

- RDB를 객체로 사용함 (객체 지향이 DB로 확장 됨)
- ORM을 사용: JPA, sequelize, Django ORM 등

2000년대 JDBC를 이용한 DB 사용 방법

```
try{
  sQuery  = " insert into member_ext ";
  sQuery += " (ucode, mobile_code, birth_time, salary, education, mobile_check, agree_global) ";
  sQuery += " values(?,?,?,?,?,?,?) ";
  con = dbm.getConnection();
  pstmt = con.prepareStatement(sQuery);
  pstmt.setInt(1, ucode);
  pstmt.setString(2, mobileCode);
  ...
  pstmt.setString(7, agreeGlobal);
  iDBReturn = pstmt.executeUpdate();
}catch(Exception e){}
finally{
  try{ if(rs != null) rs.close(); rs = null; } catch(Exception x) { }
  try{ if(pstmt != null) pstmt.close(); pstmt = null; } catch(Exception x) { }
  try { if ( con != null ) dbm.freeConnection(con); } catch (Exception x) { }
}
```

물음표 개수 틀리면 안됨

일반 statement 사용하면 안됨

pstmt 순서 틀리면 안됨

커넥션은 반드시 반납해야 함
(커넥션 관리의 부담은 오로지 개발자가...
실수하면 끔찍한 결과가 초래됨)

2010년대 MyBatis를 이용한 DB 사용 방법

```
<select id="selectUserAdminList" resultType="UserAdmin">
    SELECT
        <include refid="allColumn"/>
    FROM tb_user_admin u
    LEFT OUTER JOIN tb_user_organization o
    ON u."orgSeq" = o."seq"
    <where>
        <if test="userAuth != null">
            and u."userAuth" = #{userAuth}
        </if>
        <if test="applyState != null">
            and u."applyState" = #{applyState}
        </if>
        <if test='useDayCheck != null and useDayCheck.equals("Y")'>
            <![CDATA[
            and (length(u."useStartDay") = 0 or u."useStartDay" <= (select to_char(current_timestamp, 'YYYYMMDD')))
            and (length(u."expireDay") = 0 or u."expireDay" >= (select to_char(current_timestamp, 'YYYYMMDD')))
            ]]>
        </if>
    </where>
    ORDER BY "seq" desc
</select>
```

XML에 쿼리를 코딩함
(객체"처럼" 호출 해서 사용)

쿼리를 조건에 따라
동적으로 생성 가능함

#{foo}과 ${foo}는 차이가 있음
(구분을 잘 해야 함)

2020년대 sequelize를 이용한 DB 사용 방법

```
selectOne(params: UserSelectOneParams): Promise<UserAttributes | null> {
  return new Promise((resolve, reject) => {
    User.findOne({
      attributes: ['id', 'companyId', 'userid', 'password', 'name', 'auth'],
      where: { id: params.id },
    })
      .then((selectedOne) => {
        resolve(selectedOne);
      })
      .catch((err) => {
        reject(err);
      });
  });
},
```

모든 모델을 객체로 취급
(간결해 진 것 같은 느낌 적인 느낌)

사라지는 기술들

사라지는 기술들

UVC
Ubiquitous Via Connect

자꾸 사라지는 기술들. 새로 생기는 기술들

- 배워서 익숙해질 만 하면 바뀌는 패러다임
- 쫓아가지 않으면 뒤쳐지는 분야. 마치 흐르는 강물에서 상류를 향해 헤엄치는 느낌.
- 10년을 일해도 안 해본 것이 더 많은 분야. (~~모르는 데요~~. 안 해봤는데요)

어떡하지?

- 완전히 새로운 것은 없다. 기존 기술을 기반으로 나아갈 뿐.
- 기존 기술을 모르면 기존 기술부터 배워야 하는 부담도 있다.
- 미리 올라탈 필요가 없다. 대세가 된 후 올라타도 늦지 않는다.
    - 신 기술에 대한 맹신 NO! (검증 비용은 감당할 수 없을 만큼 클 수 있다.)

프로토콜은 사라지지 않는다.

- 프로토콜은 기술의 근본이다.