(디지털 컨버전스) 스마트 콘텐츠와 웹 융합 응용 SW개발자 양성과정

훈련기간 : 2021.05.07 ~ 2021.12.08

KH정보교육원 | 전승리

INDEX

slack 환경 설정

업무 메시징 툴 설정

1. slack.com 에 접속한다.
2. 회원가입을 진행한다.
3. 회원가입 이후 인증을 진행합니다.
4. 인증 이후 가입한 메일 주소를 채팅창에 적어주세요.
   (그럼 초대 메시지가 날아갈 것입니다 - 휴대폰 연동 가능)
5. 향후 git이나 trello와 연동이 가능하며 원격으로 떨어져 있어도 작업 사항들에 대한 체크가 매우 용이해짐

GitHub 설정

Organization 초대를 위한 기본 절차

1. github(소스 코드 관리) 회원 가입
2. github.com 접속
3. Sign Up 진행
   (등록할때 사용한 이메일과 아이디를 잘 기억해둔다)
4. 이메일로 인증 메일이 날아올텐데 인증 진행
5. 회원가입 완료후 대기
6. 초대자가 볼 수 있도록 이메일주소 공유

Java 설치

기본 환경 설정법
- 구글에서 jdk 1.8을 검색한다.
- Java SE Development Kit 8 - Downloads 를 누른다.
- 눌러서 들어오면 Windows x64를 찾아서 다운받는다.
- 최근 로그인을 요구하고 있으므로 회원가입하고 다운을 받도록 한다.
- 로그인을 해서 최종적으로 jdk-8을 다운받도록 한다.

(현재 java 13 버전까지는 8과의 차이점이 없다)
추가된 것이 있다면 parallel library 정도이기 때문에
아마도 15 버전도 별 차이는 없을 것이라 생각됨
그러므로 이미 15버전을 깔아놨다면 번거롭게 또 설치할 필요는 없다 생각됨

- 다운 받은 내용을 클릭해서 설치하도록 한다.
- 만일에 대비하기 위해 설치되는 위치를 기록해두도록 한다.

C:\Program Files\Java\jre1.8.0_291

Next 신공으로 설치를 했다면 C -> Program Files -> Java -> jdk1.8.0_291 -> bin에
java(자바 가상 머신)과 javac(바이트 코드 생성기 - 속칭 자바 컴파일러)가 보일 것임
(mac이나 리눅스의 경우엔 javac, java 명령어로 확인하면 됨)
linux의 경우엔 vi ~/.bashrc에 환경 변수 설정이 추가로 필요할 수도 있음

Java 설치가 잘 되었는지 확인하는 방법

1. Window + R키를 눌러서 cmd를 입력하여 터미널을 띄운다.
2. java 이런거 없습니다 하지 않는지 확인한다.
3. javac 도 마찬가지로 확인한다.
4. 둘 다 잘 나온다면 문제가 없지만 하나라도 안나온다면 환경설정을 새롭게 할 필요가 있다.

java or javac가 동작하지 않을 경우 환경 변수 설정이 필요하다.

1. 컴퓨터 화면상의 윈도우 버튼을 우클릭한다.
2. 시스템을 누른다.
3. 하단에 고급 시스템 설정을 누른다.
4. 환경 변수를 누른다.
5. 시스템 변수를 보면 path라는 부분이 보일텐데 더블클릭!
6. C:\Program Files\Java\jdk1.8.0_291\bin 를 새롭게 기입
   (앞서 설치한 경로 기억하기가 필요한 이유)
7. 모든 절차를 완벽하게 수행했는지 체크하고 확인을 누른다.
8. cmd를 껏다가 다시 켜서 javac와 java의 동작을 확인한다.

IntelliJ 환경 설정

인텔리제이 설치하기

1. Jetbrains toolbox를 구글에서 검색한다.
2. 가장 상단의 링크를 클릭해서 들어간다.
   (https://www.jetbrains.com/ko-kr/toolbox-app/)
3. 다운로드를 눌러서 프로그램을 다운받는다.
4. 다운 받은 프로그램을 클릭하여 설치를 누른다.
5. JetBrains Toolbox 실행하기를 체크한 상태로 마침을 누른다. (기본값)
6. 설치가 완료되면 작업표시줄에 큐브가 생성된 것을 볼 수 있다.
7. 큐브를 우클릭해서 Open Toolbox를 클릭한다.
8. 별도의 체크 없이 Accept만 누른다. (체크를 해도 별 상관은 없다)
9. 체크 이후 여러가지 설치할 수 있는 툴들이 보일 것이다.
   (여기까지 오면 일단 성공!)
10. IntelliJ IDEA Community Edition을 선택해서 설치(Install)한다.
11. 설치가 완료되면 Installed에 IntelliJ IDEA Community Edition이 보일 것이다.
    이것을 클릭해서 실행하면 라이센스 관련 사항이 올라온다.
12. 라이센스 체크하고 continue를 누른다.
13. 통계를 보낸다고 하는데 역시 네트워크 비용을 차지하여
    속도가 느려질 수 있으니 Don't send를 선택한다.
14. 그러면 Welcome to IntelliJ IDEA 하면서 창이 나타날 것이다.
    (여기까지 나오면 오늘 구축할 환경 구성은 사실상 전부 완료된 것이다)

IntelliJ에서 Java 프로젝트 만들기

1. New Project를 누르고 java를 선택한다.
2. Project SDK에 1.8 등등 버전 정보가 표시되면 괜찮으니 Next
3. Next
4. 실제 작업할 위치를 지정하도록 한다.

Project Name: first
Project Location: D:\java_work\first
(여기서 D 드라이브가 없다면 C에 해도 무방하다)

5. Finish를 누르면 프로젝트 생성이 완료된다.

IntelliJ 테마 변경법

1. IntelliJ를 킨다.
2. File -> Settings ...
3. Appearance & Behavior -> Appearance
4. Theme: 원하는 형식으로 변경

Git 환경 구축

1. 구글에서 git을 검색한다.
2. 첫 번째 나타나는 페이지로 진입한다.
   (https://git-scm.com/)
3. 페이지에 보이는 Download를 선택한다.
4. 운영체제(Windows)를 자신에 맞게 선택한다.
5. 자동으로 다운이 된 프로그램을 클릭하여 설치한다.
   Mac의 경우 - brew install git
   Linux의 경우 - sudo apt-get install git
6. Next -> Next -> Next -> Next -> Next ->
   Next -> Next -> Next -> Next -> Next ->
   Next -> Next -> Next -> Install
7. 설치가 완료되면 Launch Git Bash에만 체크를 하고 Finish를 누른다.
8. git 프롬프트가 나타나면 git --version을 입력해서 2.31.1 버전인지 확인한다.
   (Mac이나 Linux의 경우 2.x 버전이면 OK)
9. pwd 명령어를 통해서 디폴트 디렉토리가 어디로 잡혀있는지 체크한다.
   (학원컴의 경우 /c/Users/user2 에 잡혀있음)
10. cd 드라이브에 작업하고자 한다면 cd 명령어를 써서 위치를 이동한다.
    [cd(Change Directory)의 약자임]
    cd d:
    cd e:
    cd c:

* KHWeb18의 경우 우리가 모두 함께 공용으로 활용하는 저장소에 해당한다.
* fork를 눌러서 자기 자신의 계정으로 사본을 생성해왔음
* 시나리오: 팀 프로젝트를 진행하고 있었음
* 1. 작업을 하다보니 팀원이 작업한 내용을 내가 작업하고 있는 저장소에 땡겨와야함
* 2. 내가 작업한 내용을 원본 저장소에 업로드 해야함
*   2-1. 내가 작업한 내용을 사본 저장소에 저장한다.
*   2-2. 내가 작업한 내용을 원본 저장소에 저장한다.
* 3. 내가 작업한 내용을 팀원이 적용하여 잘 동작하는지 확인을 해야함

11. 깃 페이지에 접속해서 KHWeb18/LectureContents의 우측 상단에 fork를 누른다.
12. 자신의 계정에 fork된 LectureContents쪽의 초록 버튼(Code)를 누른다.
     클릭하면 전체 선택이 될 것이고 이 내용을 Ctrl + C로 복사한다.
13. d 드라이브에 있는 java_work로 이동한다.
     cd java_work
14. pwd 명령어로 현재 d/java_work인지 확인한다.
15. git clone https://github.com/silenc3502/LectureContents.git
     (여기서도 silenc3502는 자신의 계정명으로 대체 되어야 할 것이다)
16. cd LectureContents
17. git remote -v를 통해 자신의 계정명이 있는지 확인한다.

17. git config --global user.email "gcccompil3r@gmail.com"
18. git config --global user.name "silenc3502"

17번, 18번은 각자 자신의 깃 계정 이메일과 깃 아이디를 기입해야 한다.

19. 올바르게 입력했는지 확인하기 위해 git config --list 를 입력한다.
    위의 17번, 18번에서 입력한 email과 name이 보이면 성공!
20. 테스트를 위해 IntelliJ에서 새로운 프로젝트를 만들도록 한다.
    (우선 내 사본 저장소에 저장하는 방법부터 진행한다)
    File -> New -> Project 클릭

21. Java 설정을 하고 Next 신공을 펼친다.
    중간에 경로 설정과 이름 설정이 나오는 부분만 주의하자!
    경로===> D:\java_work\LectureContents\java\SanghoonLee 와 같이 자신의 이름을 선택한다.
    (여기서 경로는 LectureContents로 앞서서 clone 받은 위치여야 한다)
    프로젝트명을 설정할때 경로에 프로젝트명과 동일한 글자를 한 번 더 적어준다.
    예: 경로===> D:\java_work\LectureContents\java\SanghoonLee\SecondJava
    이름===> SecondJava
22. 프로젝트 생성 이후 src 폴더를 우클릭 -> New -> Java Class
    PrintTest 라고 이름을 지어준다.
23. 아래와 같이 코드를 작성한다.

```
public class PrintTest {
    public static void main(String[] args) {
        // 각자 자신의 이름을 작성하도록 한다.
        System.out.println("이상훈");
    }
}
```

24. File -> Settings
25. Version Control -> Git
26. Test를 눌러서 버전이 잘 나타나는지 확인한다.
27. GitHub 탭에서 Add Account를 누른다.
28. Authorize를 진행해주세요.
29. You have been successfully authorized in GitHub. You can close the page. 메시지가 나오면 성공!
30. 작업을 전부 진행하고 Apply -> OK로 닫는다.
31. 이후 상단에 있는 Git -> commit을 누른다.
32. 그러면 어떤 작업을 했는지 기록하는 창이 나타난다.
33. '[이상훈]두 번째 자바 작업 + 깃(Git) 사본 저장 연습' 형태로 메시지를 적어주세요.
    실제 이 부분에는 여러분들이 작업한 내용에 대해 간략하게 적어야 나중에 프로젝트가 수월해집니다.
    (팀원들간에 서로 어떤 작업을 했는지 파악하는데 활용하곤 합니다)
    작성 이후 commit을 눌러 작업을 진행한다.
34. commit 이후 상단의 Git -> push를 눌러서 사본 저장소에 저장을 완료한다.
35. push를 눌렀을때 메시지가 나올텐데 merge를 누르면

주의할 부분

윈도우에서 git을 사용하는 경우 작업표시줄에 등록한 상태로 띄우면 경로 설정이 이상하게 적용된다.
그러므로 올바른 경로를 사용하기 위해서 윈도우 시작 버튼(MS사 로고 - 좌측 하단)을 눌러서
Git Bash로 실행을 시켜야 올바른 경로가 적용된다.

이번에는 원본의 내용을 사본으로 다운로드 해보자!

1. 먼저 자신의 사본 저장소로 이동한다(cd, ls, pwd 등의 명령어를 활용하여 이동)
2. git remote -v 로 사본이 맞는지 확인한다.
3. git remote add upstream (원본저장소주소)
   ex) git remote add upstream https://github.com/KHWeb18/LectureContents.git
4. git remote -v 를 입력하였을때 origin, upstream 두 개가 같이 보일 것이다.
5. git fetch upstream 를 통해 원격 저장소의 원본 내용을 받아올 준비를 갖춘다.
6. git merge upstream/main 을 통해 원본의 내용을 전부 땡겨온다.

* 실수로 오타를 입력하여 삭제를 해야 하는 경우엔 아래와 같이 한다.

git remote rm 오타낸이름

이번에는 사본의 내용을 원본으로 업로드 해보자!

1. 웹상에서 편하게 작업할 수 있음
   웹상에서 사본 저장소에 진입하여 Contribute -> Open pull request 를 누른다.
   (IntelliJ를 통해 사본의 내용을 commit, push 완료한 상태여야한다)
2. Create pull request 를 누른다.
3. Title에 '[이상훈]사본 내용을 원본에 업로드합니다.' 과 같은 형태로 메시지를 작성한다.
4. 메시지 작성 완료후 Create pull request를 누른다.

참고 사항

upstream/main -> main

위에서 upstream을 git remote add 명령을 통해 KHWeb18(원본)으로 지정하였다.
그러므로 원본에 있는 branch(브랜치) main의 내용을 main(다른 표기가 없으므로 origin을 뜻함)으로 꽂아넣는다.
그러므로 결국 원본 내용이 사본에도 적용되게 된다.

VSCode 환경 설정

일단 우리는 IntelliJ 라이센스가 없어서 Ultimate 를 사용할 수 없다.
이런 경우 대안으로 사용할 수 있는 것이 VSCode에 해당한다.
물론 제일 좋은것은 Ultimate 지만
Community 버전은 기능에 제한이 걸려서 Frontend 개발시 불편함이 꽤 많다.
반면 VSCode는 무료임에도 Ultimate 보다 조금 못미치는 정도라 꽤 쓸만하다.

VSCode 다운로드

1. 구글에서 VSCode를 검색한다.
2. 자신의 환경에 맞게 VSCode를 다운로드 받는다.
3. 실행해서 확인 누른다.
4. 동의를 누른다.
5. 난 그냥 기본값을 사용
6. 다음
7. 다음
8. 설치
9. 실행
10. 색상 선택을 위해 Pick Theme! (원하는 색상 선택)
11. HTML Snippets 를 선택해서 install 한다.
12. Vue 3 Snippets 를 선택해서 install 한다.
13. JavaScript (ES6) code snippets 를 선택해서 install 한다.

vue 환경 구성
일단 Vue나 React나 Svelte 같은 최신 기능들은 대부분 ECMA 6 스펙을 따른다.

nodejs 설치
1. https://nodejs.org/ko/download/ 에서 자신의 운영체제에 맞게 다운로드한다.
2. 실행
3. Next
4. 라이센스 동의 후 Next
5. Next
6. Next
7. Next
8. nodejs 일부 패키지에 C/C++ 기능이 필요한 것들이 있음
   묻지도 따지지도 말고 그냥 필요하면 다 설치하라고 체크하고 Next
9. Install
10. 설치가 꽤 오래 걸릴 수도 있으니 커피나 한 잔 하면 좋다.
11. node -v 로 nodejs 설치 여부를 체크한다.
12. npm --version 으로 npm 설치 여부를 체크한다.

Vue.js 설치
1. npm install -g @vue/cli
2. 오류나면 npm audit fix
3. MAC에선 권한(permission) 없다고 하면 관리자 모드로 sudo 줘서 실행한다.

vue 프로젝트 생성
1. 적당한 위치를 잡는다(HTML/CSS, JavaScript에선 javascript 폴더로 잡음)
2. vue create frontend
   frontend라는 이름으로 프로젝트가 만들어짐

3. 옵션 설정을 해야함

```
? Please pick a preset:
  Default ([Vue 2] babel, eslint)
  Default (Vue 3) ([Vue 3] babel, eslint)
❯ Manually select features

? Check the features needed for your project:
 (*) Choose Vue version
 (*) Babel
 ( ) TypeScript
 ( ) Progressive Web App (PWA) Support
 (*) Router
❯(*) Vuex
 ( ) CSS Pre-processors
 (*) Linter / Formatter
 ( ) Unit Testing
 ( ) E2E Testing


Choose a version of Vue.js that you want to start the project with 3.x

? Use history mode for router? (Requires proper server setup for index fallback in
production) Yes

? Pick a linter / formatter config: (Use arrow keys)
❯ ESLint with error prevention only
  ESLint + Airbnb config
  ESLint + Standard config
  ESLint + Prettier

? Pick additional lint features: (Press <space> to select, <a> to toggle all, <i> to
invert selection)
❯(*) Lint on save
 ( ) Lint and fix on commit

? Where do you prefer placing config for Babel, ESLint, etc.?
  In dedicated config files
❯ In package.json

? 다음 질문엔 그냥 앤터치면 된다.

그럼 뭔가 프로젝트 설정하기 위해 많은 작업이 일어난다.
vue 프로젝트 구동 -> cd frontend -> npm run serve
```

Spring 환경 설정

HTML/CSS, Vue, React, JavaScript 뭐가 됫건 일단 서버에 붙어야 멋지게 실행가능하다.
기본적으로 서버에 붙을 수 있게 Controller 정도만 만들어놓도록 한다.

Spring 초기 설정

1. 구글에서 spring initializr 를 검색한다.
   https://start.spring.io/
2. Project: Gradle Project
   Language: Java
   Spring Boot: 2.5.0
   Project Metadata: 변경 없음
   Dependencies: Lombok, Spring Web, Thymeleaf,
   JDBC API, Spring Data JPA, MySQL Driver
3. Generate를 누르면 다운이 받아짐
4. 다운 받은 내용을 압축해제한다(내용이 바이러스라면서 사라지면 동작 안되니 주의)
5. 압축 해제한 폴더를 우리의 git 경로상 javascript에 있는 이름 폴더로 이동시킨다.
6. IntelliJ를 킨다.
7. Open을 눌러서 javascript상에 있는 이름 폴더의 demo를 선택한다.
8. 신뢰(Trust)하냐고 묻는데 당연히 Trust를 누르면 된다.
9. IntelliJ가 gradle을 구동시켜 프로젝트를 빌드하기 시작하는데
   시간이 꽤 걸리니 커피 혹은 차나 한 잔 하도록 한다.
10. 준비가 완료되면 Project 쪽 상단의 메뉴가 바뀐다.
11. 진입하자마자 src/main/resources/application.properties의 이름을 Refactor로 변경한다.
    application.yaml로 변경한다.
12. 그리고 내부의 내용을 아래와 같이 작성한다.

```
server:
  port: 7777
```

13. src/main/java/com.example.demo 폴더의 DemoApplication을 실행한다.
14. 콘솔창에 포트 번호 7777로 서버가 구동된다는 것이 나오면 OK!
15. build.gradle 폴더에 sourceCompatibility = '8' 부분을 버전에 맞춰서 작성한다.