효율적인 프로그래밍 방법

-기본에 충실한 개발자-

대덕인재 개발원
담당자: 임영수

진행 순서

- 강사 소개
- 개발자 가져야할 기본적인 자세
- IT기술(JAVA) 기본에 충실한 개발자
  - 자바 기초를 확실히 알자
  - 자바 기본을 통해 프로그램 흐름을 알자
  - 기본적인 오픈 소스에 익숙해 지자
- 효율적인 개발 일정을 위한 준비
- 전문개발자를 위한 자신의 길 선택
- '늦었다고 생각될 때가 가장 빠르다'
- 현실은 초급개발자 마음은 고급개발자

강사 소개

- 2000.03 ~ 2002.06 통신 소대장 근무
- 2002.09 ~ 2003.03 JAVA전문가 교육과정 수료
- 2003.05 ~ 2003.12 아이탑시스템 근무
- 2003.12 ~ 2007.01 연구관리시스템 개발
  (식품의약품안전청 외 2)
- 2007.02 ~ 2007.06 CRM 시스템 구축
- 2007.06 ~ 2008.01 NTIS 시스템 구축
- 2008.04 ~ 2010.12 KT SODE 시스템 구축
- 2011.01 ~ 2012.03 사업관리시스템 구축
- 2012.04 ~ 2012.07 테크인모션 근무
- 2012.08 ~ 2013.07 3세대 특허넷 3차년도 사업참여
- 2013.11 ~ 2014.01 기획재정부 용역사업통합보안관리시스템 구축
- 2014.03 ~ 2016.04 한국철도시설공단 PQ/적격심사 시스템 개선 개발
- 2016.04 ~ 2016.10 피플맥 모바일 조문 서비스
- 2016.10 ~ 2020.06 중소기업기술정보진흥원 사업관리시스템 유지보수
- 2020.06 ~ 창업진흥원 PMS시스템 유지보수

개발자가 가져야할 기본적인 자세

- 개발을 위한 기본 기술을 충실히 학습(업계 중심 기술)
- 맡은 업무에 대해 전문가와 협의 가능한 수준의 업무 지식 학습 (입사후)
- 개발 관련 전문 지식 학습(툴 및 이론)
- 주위 동료와 협업할 수 있는 열린 생각 소유
- 직장인이 아닌 직업인이라는 인식
- 평생 직업에 맞는 평생 학습 이행
- 자기계발 및 기타 교양서적 1년에 1권정도
- '하고자 하는 강한 의지'

IT기술(JAVA)기본에 충실한 개발자

- 자바기초(데이터형, 객체지향입문)
- 자바API 활용
- 자바기본(객체지향, IO, Exception, Thread)
- 웹기본(Servlet, JSP)
- Spring(AOP, DI, IoC)
- Spring MVC
- 오픈소스 활용

기본에 충실한 개발자 - 기초

- 기본데이터형 예제

```
private int intValue;
private short shortValue;
private long longValue;

private float floatValue;
private double doubleValue;

private byte byateValue;
private char charValue;

private boolean booleanValue;

private String stringValue;
private Boolean BooleanValue;
```

결과

```
DefaultType [
intValue=0,
shortValue=0,
longValue=0,
floatValue=0.0,
doubleValue=0.0,
byteValue=0,
charValue= ,
booleanValue=false,
stringValue=null,
BooleanValue=null]
```

기본에 충실한 개발자 - 기초

- String ref val

```
String nameA = "홍길동";
String nameB = "홍길동";
String nameC = new String("홍길동");
String nameD = new String("홍길동");

System.out.println("nameA == nameB: "+(nameA == nameB));
System.out.println("nameA == nameC: "+(nameA == nameC));
System.out.println("nameC == nameD: "+(nameC == nameD));
```

결과

```
nameA == nameB: true
nameA == nameC: false
nameC == nameD: false
```

기본에 충실한 개발자 - 기초

- String VS StringBuffer

String stringValue = "1234567890";
StringBuffer stringBuffer = **new StringBuffer("1234567890");**

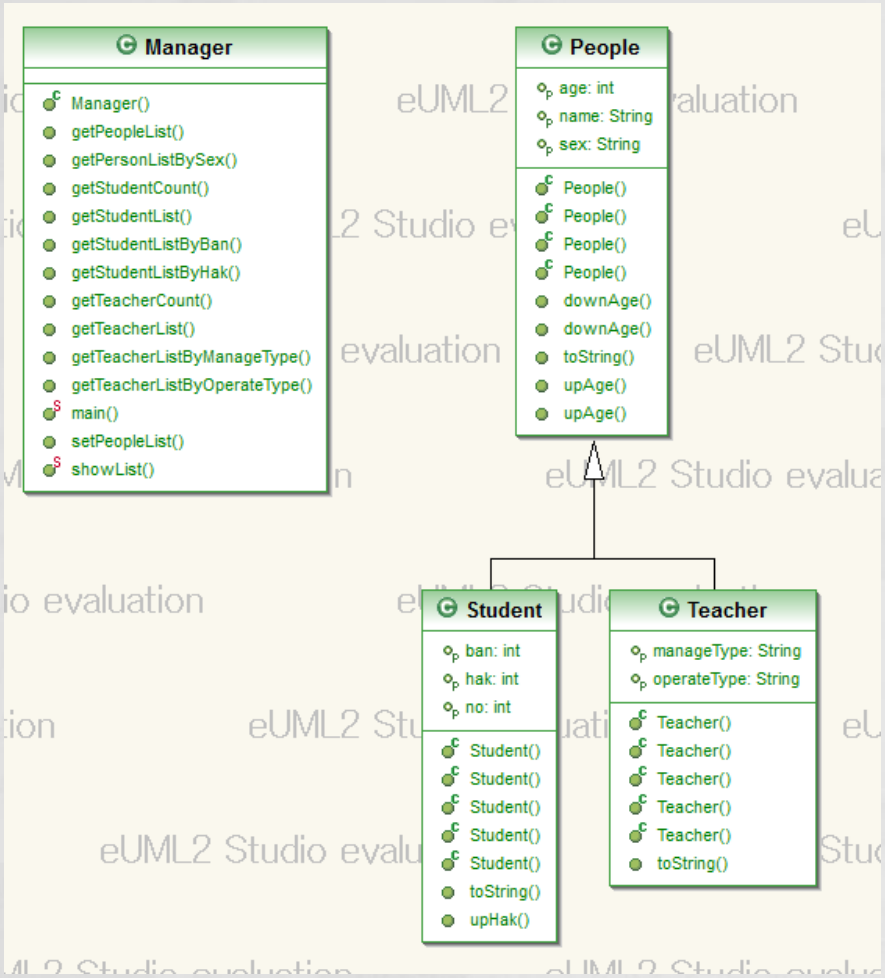
System.*out.println("concat: "+stringValue.concat("abcd"));*
System.*out.println("append: "+stringBuffer.append("abcd"));*

System.*out.println("stringValue: "+stringValue);*
System.*out.println("stringBuffer: "+stringBuffer);*

결과
concat: 1234567890abcd
append: 1234567890abcd
stringValue: 1234567890
stringBuffer: 1234567890abcd

기본에 충실한 개발자 - 기초

- 상속



People people5 = **new Student("김장철", "남자", 11, 4, 4, 11);**
People people6 = **new Student("한지연", "여자", 8, 1, 2, 12);**

People people7 = **new Teacher("최홍만", "남자", 32, "정교사", "담임교사");**
People people8 = **new Teacher("이수학", "남자", 30, "정교사", "담임교사");**

기본에 충실한 개발자 – 기초

- 자바 api 활용
  - java.lang
    - Object
    - String, StringBuffer
    - Integer, Long, Short, Double, Float
  - java.util
    - Collection
    - List
    - Set
    - Map
  - java.io
    - InputStream, OutputStream
    - Reader, Writer

기본에 충실한 개발자 – JAVA 기본

- 파일 io와 Exception 처리

```java
public class FileReaderTest {
    public static void main(String[] args){
        File file = new File("d:/temp/sample.txt");
        if(file.canRead()){
            try {
                BufferedReader bufferedReader = new BufferedReader(new FileReader(file));
                while(bufferedReader.read() != -1){
                    System.out.println(bufferedReader.readLine());
                }
                bufferedReader.close();
            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }
}
```

```java
public class FileReaderTest {
    public static void main(String[] args){
        File file = new File("d:/temp/sample.txt");
        if(file.canRead()){
            BufferedReader bufferedReader = null;
            try {
                bufferedReader = new BufferedReader(new FileReader(file));
                while(bufferedReader.read() != -1){
                    System.out.println(bufferedReader.readLine());
                }

            } catch (Exception e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } finally {
                if(bufferedReader != null){
                    try {
                        bufferedReader.close();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        }
    }
}
```

기본에 충실한 개발자 – JAVA 기본

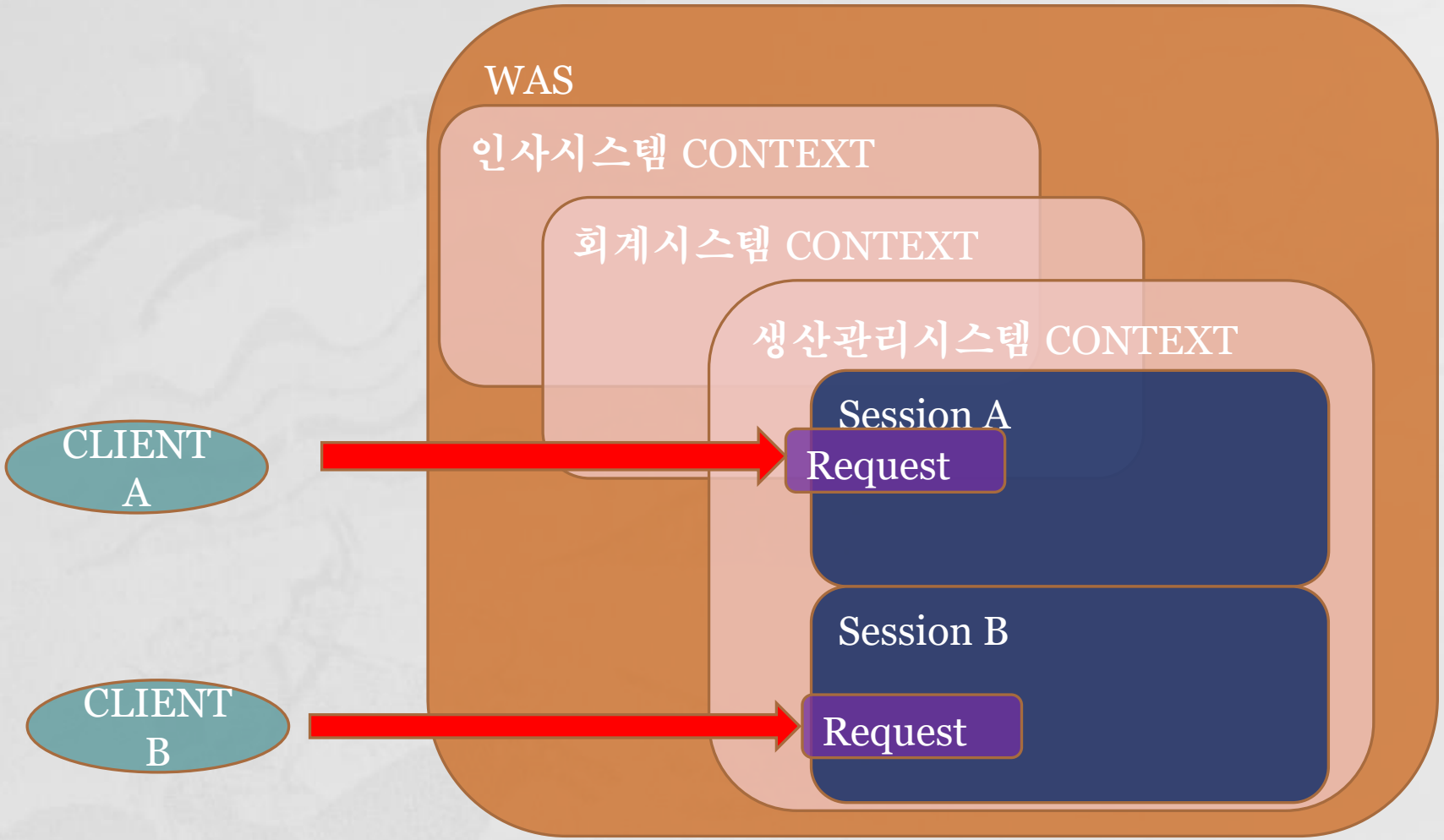
- Thread

```
public class SampleThread extends Thread{
    String name;

    public SampleThread(String name) {
        super();
        this.name = name;
    }

    @Override
    public void run() {
        while(true){//Thread는 여기에 항상 true를 넣어야 할까?
            // TODO Auto-generated method stub
            System.out.println(name+" call!!");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }


    public static void main(String[] args) {
        SampleThread firstSampleThread = new SampleThread("홍길동");
        SampleThread secondSampleThread = new SampleThread("유관순");

        firstSampleThread.start();
        secondSampleThread.start();
    }
}
```

- Runnable

```
public class SampleRunner implements Runnable {
    String name;

    public SampleRunner(String name){
        super();
        this.name = name;
    }

    @Override
    public void run() {
        while(true){//Thread는 여기에 항상 true를 넣어야 할까?
            // TODO Auto-generated method stub
            System.out.println(name+" call!!");
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }
    }

    public static void main(String[] args) {
        SampleRunner firstSampleRunner = new SampleRunner("홍길동");
        SampleRunner secondSampleRunner = new SampleRunner("유관순");

        Thread firstSampleThread = new Thread(firstSampleRunner);
        firstSampleThread.start();

        Thread secondSampleThread = new Thread(secondSampleRunner);
        secondSampleThread.start();
    }

}
```

기본에 충실한 개발자 – 웹기본



기본에 충실한 개발자 – 웹기본(계속)

- HttpServlet
  - doGet(HttpServletRequest request, HttpServletResponse response)
  - doPost(HttpServletRequest request, HttpServletResponse response)
- Servlet 과 JSP 주요 객체

Servlet	JSP
HttpServletRequest	request
HttpServletResponse	response
HttpSession	Session
ServletContext	application

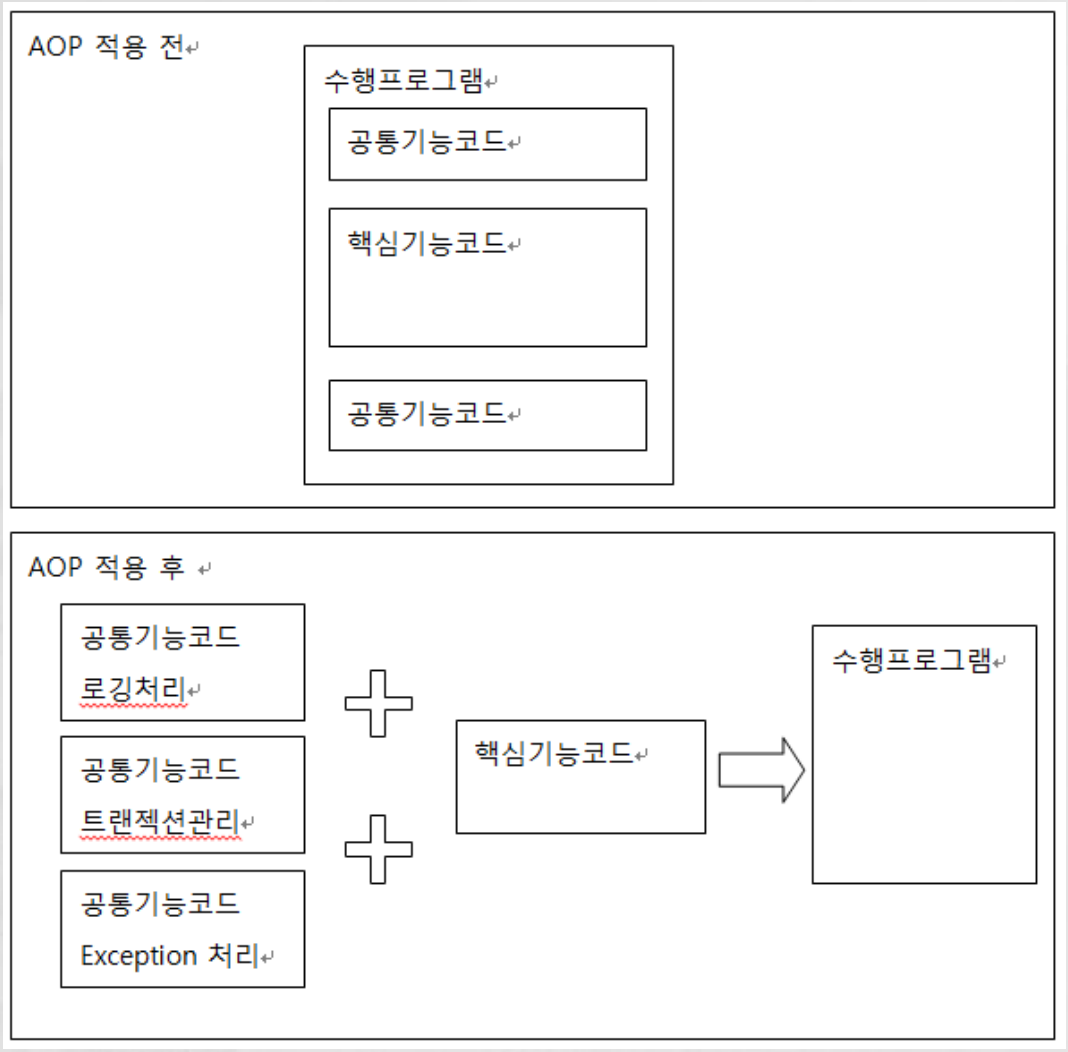
- web.xml

기본에 충실한 개발자 – Spring

- IoC(Inversion of Control)
  - 객체의 생성에서부터 생명주기까지 모든 객체의 제어권이 Spring에게 있다.
- DI(Dependency Injection)
  - Setter Injection
  - Constructor Injection
  - Method Injection
- AOP(Aspect Oriented Programming)
  - 핵심관심사: 구현 대상이 되는 핵심 비즈니스 기능
  - 공통관심사: 로깅, Exception/ Transaction 처리

(labPart2 참고)

기본에 충실한 개발자 – Spring AOP 개념



기본에 충실한 개발자 – Spring MVC

- Spring Framework
  - AOP를 위한 Container
  - Web Tier Integration
  - Data Access Tier Integration
  - Test Integration

- Spring MVC
  - Model2 개발 방식을 위한 MVC 프레임워크
  - DispatcherServlet
  - ViewResolver: jsp, XML, JSON, PDF, POI 등
  - View: JSTL, Spring Form Tag Lib

기본에 충실한 개발자 – 오픈소스활용

- JAVA 개발시 필수활용 LIB
  - POI : excel 관련 API
  - commons-io : FILE 관련 IO API
  - commons-validator: 유효성 검증 API
  - commons-beanutils: 객체 속성 활용 API
  - log4J: 오픈소스 기반 logging API
  - IBATIS: 현장에서 DAO 개발 표준으로 적용
- 형상관리툴
  - CVS, SVN
- Javascript 및 웹개발 관련 기술
  - jQuery
  - HTML
  - CSS

기본에 충실한 개발자 – 오픈소스활용

- 파일 경로 파싱

filename = "d:/upload/board/201406/info.txt";
substring? charAt? 확장자 분리?

common-io.jar FilenameUtils 클래스 사용
FilenameUtils.getPrefix(filename); // d:/
FilenameUtils.getPath(filename); // upload/board/201406/
FilenameUtils.getBaseName(filename); //info
FilenameUtils.getExtension(filename); //txt

- text 파일 파싱/ 생성

FileInputStream? FileOutputStream?

common-io.jar FileUtils 클래스 사용
File file = new File("d:/temp/writesample.txt");
FileUtils.writeLines(file, lines); //파일생성

List<String> lines =FileUtils.readLines(file); //파일파싱

기본에 충실한 개발자 – 오픈소스활용

- 데이터 유효성 검사
  - 날짜 유효성 검사(yyyy.MM.dd)
  - 이메일 유효성 검사
  - url 유효성 검사

String 관련 메소드로 개발?

commons-validator.jar 사용

```
GenericValidator.isDate("2014.03.12", "yyyy.MM.dd", true);
GenericValidator.isEmail("test@naver.com");
GenericValidator.isUrl("http://www.ddit.or.kr");
```

- 숫자형식 검사

commons-lang.jar 사용

```
NumberUtils.isDigits("123456");    //true
NumberUtils.isDigits("123.456");   //false
NumberUtils.isNumber("123456");   //true
NumberUtils.isNumber("123.456");   //true
NumberUtils.isNumber("12.34.56");  //false
```

기본에 충실한 개발자 – 오픈소스활용

- 객체 복사

  객체생성 후 속성값 설정? 반복문을 활용한 데이터 복사?

  commons-lang.jar 사용

  List<String> targetList = (List<String>) ObjectUtils.*clone(sourceList);*

  GenericValidator.isEmail("test@naver.com");

  GenericValidator.isUrl("http://www.ddit.or.kr");

- 문자열관련 처리

  commons-lang.jar 사용

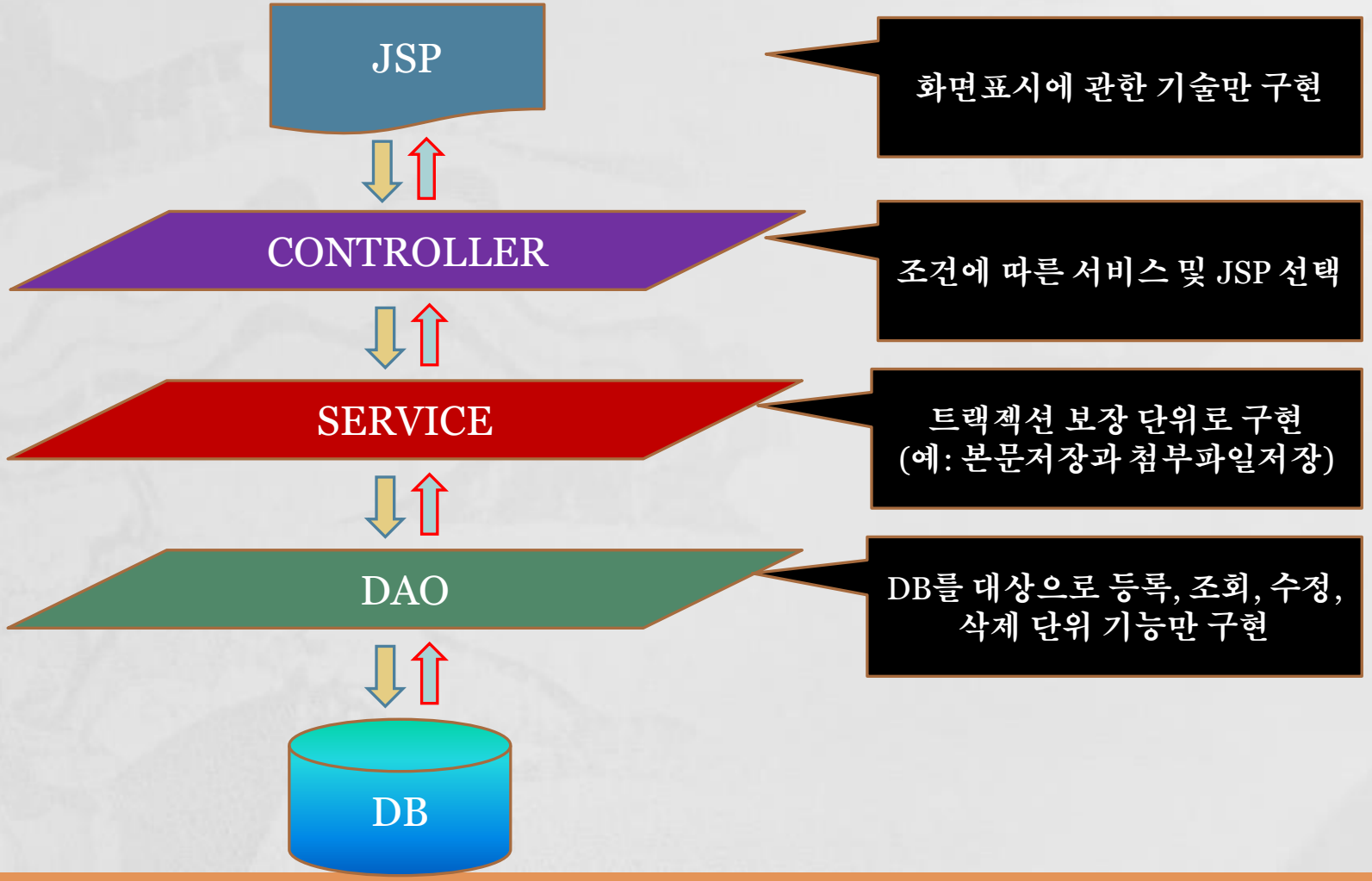
  StringUtils.*leftPad("테스트", 40, '-')*

  ----------테스트

  StringUtils.*repeat("테스트", "-",5)*

  테스트-테스트-테스트-테스트-테스트

효율적인 개발일정을 위한 준비
- 레이어 기반 구현 방식 이해

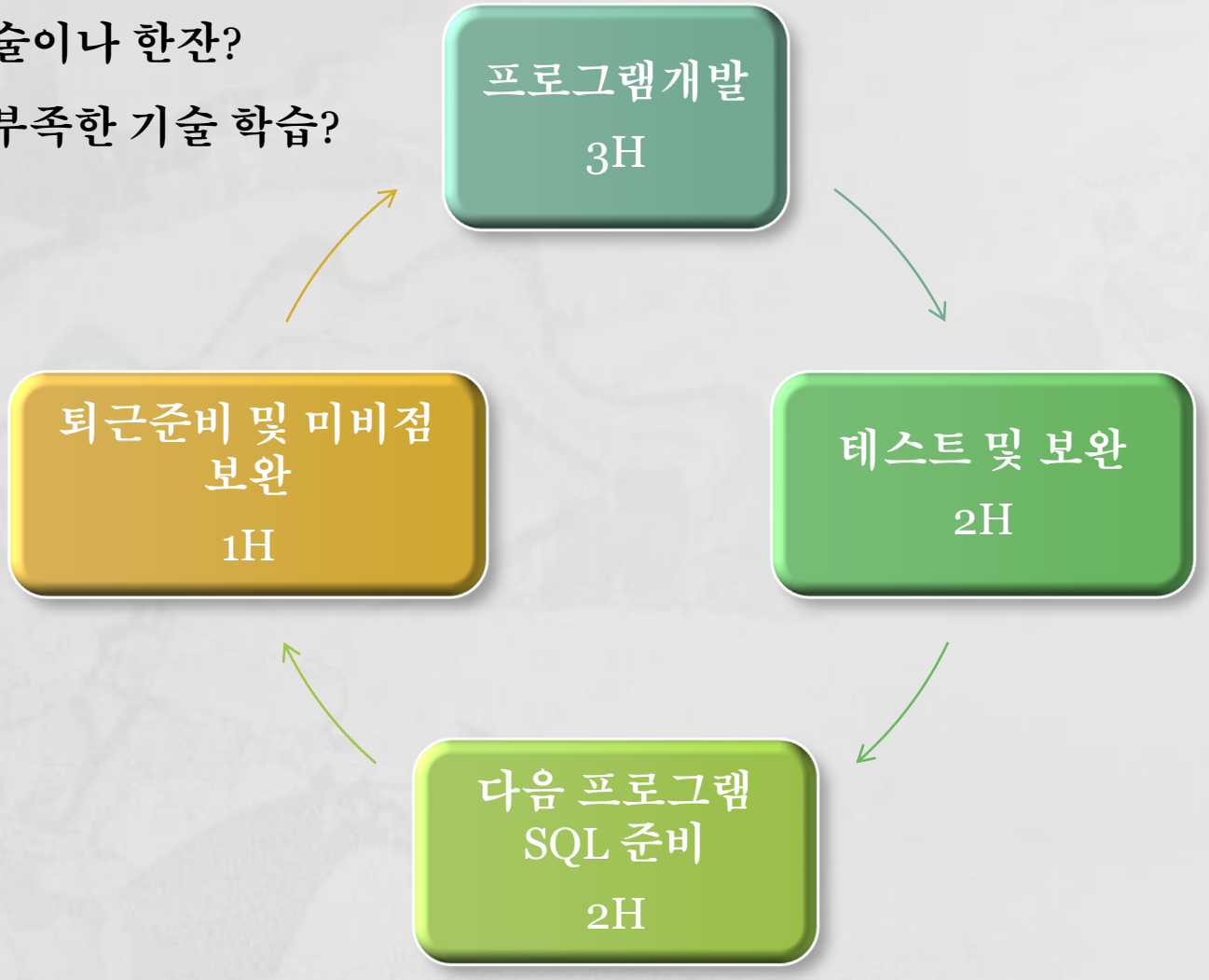


효율적인 개발일정을 위한 준비

- 개발타임스케줄에 따른 실무 연습

퇴근 후 술이나 한잔?

퇴근 후 부족한 기술 학습?



전문개발자를 위한 자신의 길 선택

- 기술전문 분야를 선정해 볼 것.
  - 자바 웹개발(SI)
  - .NET 웹개발(SI)
  - 임베디드
  - 안드로이드/IOS 모바일
  - 플렉서
- 하나 이상의 분야에 대한 업무지식을 습득.
- 5년 후의 자신의 모습을 상상 해 볼 것.
- 개발자에서 향후 성장 방향 선택
  - 개발PM/PL
  - 사업관리
  - DBA
  - 시스템 아키텍처등

늦었다고 생각될 때가 가장 빠르다

- 6개월간의 인재개발원에서의 시간
  - 적성에 맞지 않는다고 고민한 시간
  - 3D 업종, 신기술 습득에 대한 두려움
- 천리길도 한걸음부터(셋 중 하나라도 자신있게)
  - DB SQL 작성능력
  - JAVA 기본 기술
  - HTML, CSS, javascript, jQuery등 화면단 기술

현실은 초급개발자 마음은 고급개발자

- 신입사원도 사원이다.
- 지금의 위치와 능력은 앞으로 자신의 개발자 생활의 위치와 능력과 크게 달라지지 않는다.
- 능력 없고 착한 사람? 능력 있고 거친 사람? 누가 좋을까?
- 함께 일하고 있는 사람은 오너가 아닌 그냥 오래 다닌 사원이다.
- 직장상사는 질문에 답을 주는 사람이 아니라 업무를 지시하고 확인하는 또 다른 직원이다.
- “고객과 시연한다” 생각하고 테스트는 꼭 한다.
- 프로젝트는 기간이 아닌 결과가 끝나야 끝이다.
- 퇴근 이후의 시간 활용이 3년 후의 자신의 모습을 결정할 것 입니다.