1. **IoC**란
2. **Spring POJO(Plain Old java Object)**
   1. 참고 URL
      1. http://itewbm.tistory.com/entry/POJOPlain-Old-Java-Object
   2. ex) jUnit version-4
   3. 평범 자바 오브젝트
      1. 누구에게도 종속 받지 않는 평범한 자바 객체, annotation을 사용해서 활용
      2. 자바개발자가 마음대로 정의 할 수 있는 객체이다.
      3. EJB 컨테이너에 의존 하는 객체처럼 복잡하지 않고 개발자가 쓰기 편한 간단한 Object를 만들어 사용하는 간단한 Object 개념이다
   4. POJO의 필수요소
      1. light-weight(possibly): 가볍게
      2. fiexible: 유연성
      3. simple: 간단 명료
      4. supported by separate optional components such as hibernate or spring.
   5. 개발 배경
      1. 2000년 마틴 파울러가 가을에 개최된 어느 컨퍼런스 발표준비를 하면서 처음 만들어낸 말이다. 마틴 파울러는 EJB(Enterprise JavaBean) 보다 단순한 자바 오브젝트에 도메인 로직을 넣어 사용하는 것이 여러 가지 장점이 있지만 사람들이 EJB를 사용하는 이유는 지어진 이름이 없다고 판단하고 POJO라는 이름을 붙였다.
   6. EJB의 강력한 대안으로 EJB가 아닌 자바이상의 특징을 가지고 있는 프로그래밍 모델이다.
3. **값이 저장되는 scope 4가지 (page, request, session, Application)**
   1. 값을 유지하는 범위
   2. 참고 URL
      1. http://hyeonstorage.tistory.com/88
   3. Application
      1. ServletContext는 전체 application의 설정 (application 변수명) -> 전체 scope
   4. 하나의 Servlet마다 하나의 ServletConfig가 추상화 되어 만들어진다.
   5. WAS가 종료될 때 삭제된다.
4. **Cookie & Session**
   1. Http Protocol 자체가 상태유지를 해주는 프로토콜이 아니기 때문에 상태유지를 할 수 있는 기술이 필요하다.
   2. Cookie
      1. Client에 상태유지 할 정보를 저장하는 방법
      2. 요청을 할 때, Cookie를 가지고 들어간다.
      3. Server에서 Cookie를 만들어 준다.
      4. 객체가 아닌 String정보만 저장한다.
      5. 새로운 쿠키를 만들어서 변경 혹은 삭제가 가능하다. 시간: 0으로 해서 삭제
   3. Session
      1. Server에 상태유지 할 정보를 저장하는 방법
      2. Client를 식별하기 위해서 내부적으로 Cookie를 이용 한다.
      3. session에 저장된 정보는 계속 가지고 있는다.
      4. 내부적으로 Cookie를 사용한다.
         1. getSession()함수에서 Cookie를 생성해서 Browser에 전달
         2. Cookie에는 id값만 저장된다.
      5. setMaxAge(-1) -> -1 옵션은 Browser가 유지되는 동안을 의미한다.
   4. 요청이 바뀌어도 정보가 유지되었으면 할 때 사용.
   5. 쿠키는 서버에서 기억할 수 있는 아이디를 사용해야 한다.
5. **Filter, Listener**
6. Dao
   1. Data Access Object
7. Presentation Layer (WEB, Application, 어디에 보여줄 것인가), Service Layer (Service, Servlet) (logic), Data Layer (DAO(interface) <- DAO <- DB)
8. VO(Value Object) 값만 가지고 다니는 Object, DTO(Data Transfer Object) -> 가방으로 비유
9. “kang”.equals(name), name.equals(“kang”)
10. **JDBC Database실행 객체 3개**
    1. Statement
       1. 완벽한 쿼리가 준비된 상태에서 가능
    2. PreparedStatement (Statement 상속)
       1. 쿼리 자체가 완벽하지 않은 상태에서 가능
       2. ?로 표시, 반드시 binding
       3. Statement보다 성능이 더 좋다, 쿼리를 번역한 상태로 가져다 사용한다. 물음표제외한 부분.
    3. CallableStatement (Statement 상속)
       1. 주로 Stored 프로시저를 사용하기 위해서 사용된다.
    4. close
       1. 접속 가능한 수가 정해져 있기 때문에 통로를 제대로 닫아주지 않으면 문제가 생길 수 있다.
11. MVC2는 모든 요청은 Servlet 1개가 다 받는다.
    1. Web application에 진입할 수 있는 통로를 하나로 제한 함으로써 보안을 강화하기 위해서이다.
12. 요청이 바뀌어서 재 요청하는 redirect
13. 요청은 하나지만 다른 요청을 하는 경우 forward를 사용 인자로 request response를 보내준다.

Forward & Redirect

* 현재 작업중인 페이지에서 다른 페이지로 이동하는 페이지 전환 방식

Forward

* Web Container 차원에서 페이지 이동이 일어난다. 웹 브라우저에서는 다른 페이지로 이동했는지 확인 할 수 없다. 웹 브라우저에는 최초에 호출한 URL이 표시되고 이동된 페이지의 URL정보를 확인 할 수 없다. 또한, 동일한 웹 컨테이너에 있는 페이지로만 이동 가능하다.
* 최초 호출한 URL에 대한 request와 response객체는 이동된 페이지에서의 request, response와 공유한다.
* Forward할 URL로 요청정보를 말 그대로 forward(건네주기), 그대로 전달한다. 따라서, 사용자가 최초로 요청한 요청정보가 다음 URL에서도 유효하다.

Redirect

* Web Container 는 Redirect 명령을 받으면 웹 브라우저를 다른 페이지로 이동시킨다. 웹 브라우저는 지시된 URL주소로 요청을 해서 페이지를 이동한다. 또한, Forward와 다르게 다른 웹 컨테이너에 있는 주소로 이동이 가능하다.
* Forward와 다르게 Redirect에서는 새로운 페이지로 이동하면서 request와 response객체를 새롭게 생성한다.
* 최초 요청을 받은 URL1 에서 클라이언트로 redirect할 URL2를 리턴하고, 클라이언트는 이를 받아서 새로운 요청을 생성해서 URL2로 요청을 한다. 따라서, 최초로 요청한 요청정보는 더 이상 유효하지 않는다.

Forward와 Redirect 사용 예

* Forward의 경우 단순 조회 요청(글 목록 보기, 검색 등)에 사용하는 것이 바람직하다.
* Redirect의 경우 시스템(Session, Database)에 변화가 생기는 요청(로그인, 회원가입, 글쓰기)에 사용하는 것이 바람직하다.
* 예를 들어 게시판에 글을 작성하고 ‘작성을 완료했습니다.’ 라는 응답 페이지에서 새로 고침을 누를 수 있는데 이 때, Forward로 구현되어 있는 경우 이전의 request와 response를 공유하기 때문에 중복된 글이 작성 될 수 있다. 따라서 이러한 경우에는 Redirect를 해야 한다.

**요청하나 당 URL하나**

웹 컨테이너

* 웹 서버의 컴포넌트 중 하나로 Apache Tomcat이 웹 컨테이너 중 하나의 예이다.

웹 서버

* 웹 브라우저에게 정적인 파일을 제공하는 서버로 HTML, jpeg, gif와 같은 파일들을 HTTP 프로토콜을 통해 웹 브라우저에 제공한다.
* 최근에는 웹 서버에서도 내부 Application을 동작시킬 수 있는 컨테이너를 내장하고 있다.

웹 컨테이너 혹은 Servlet 컨테이너

* 서버단에서 Application을 동작할 수 있도록 지원한다.
* 초창기에는 CGI, 그 이후에는 Servlet, ASP, JSP, PHP 등의 프로그램으로 사용되고 있다.

웹 어플리케이션 서버

* 동적 페이지를 보여준다.
* 웹 서버 + 웹 컨테이너?

ex) init-param, param-name, param-value

JDBC: <http://hyeonstorage.tistory.com/111>

Spring POJO: <http://egloos.zum.com/kaludin/v/2267379>

Assert: <http://junit.sourceforge.net/javadoc/org/junit/Assert.html>

Redirect Forward:

<http://ysssb.tistory.com/entry/forward%EC%99%80-redirect%EC%9D%98-%EC%B0%A8%EC%9D%B4>

MVC구조: <http://hsp1116.tistory.com/9>

MVC위키: <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

화요일 오후2시 전용우 튜터님(대기실)

다음주 Event loop 강의 + 리뷰

Front-End (github reservation-system/Issues)

* Template의 활용
  + 구조가 한눈에 들어오지 않는다.
  + 하나의 함수에서 너무 많은 처리를 하고 있다.
  + 유지보수 측면에서 좋지 않다.
  + handlebarsjs -> template 엔진
* attr의 다양한 활용
  + prop/attr/data 용도가 다르다
  + 학습 해볼 것
* dom탐색
  + markup구조가 변경되면 코드를 계속 수정해 주어야 한다.
  + css선택자 처음부터 ~ **E[foo\*=”bar”] 까지는 알아야한다.**
* 콜백에 지나친 익명 함수 사용
  + Module Pattern 모듈화
* 비동기의 이해
  + 비동기적으로 코드가 수행되기 때문에 원하는 결과를 항상 얻는다는 보장을 할 수 없다.
  + event loop

1. 위의 개념들 정리
2. Front-End 공부할 자료 찾아볼 것