

## Spring & Ibatis 프레임워크 과정 (Spring DWR)

 $11.8.27 \sim 11.10.15$ 

한국소프웨어기술진흥협회(KOSTA)



## **Spring DWR**

### ❖Spring Dwr 개요

- DWR(Direct Web Remote)은 Ajax를 이용해서 자바스크립트에서 원격지 웹 컨테이너 서버의 자바객체를 호출할 수 있도록 해주는 자바기반의 RPC라이브러리 이다.
- DWR은 서버 측의 객체를 호출해 주는 자바스크립트 코드를 자동으로 생성해 줄뿐만 아니라 결과 객체를 JSON형식으로 알맞게 변환해 주기 때문에 적은 노력으로 자바스크립트에서 원격지 객체를 쉽게 사용할 수 있는 장점이 있다.



## **Spring DWR**

- ❖ Spring DWR 구현 과정
  - Spring DWR 라이브러리를 추가한다.(dwr.jar)
  - <dwr:controller> 태그 및 URL 매핑설정을 이용해서 DWR 관련 컨트 롤러 설정
  - <dwr:remote>, <dwr:configuration>태그를 이용해서 제공할 빈객체 변화 매핑 설정
  - 자바 스크립트에서 스프링 빈에 해당하는 자바스크립트 호출



❖ Spring Dwr 설정1 DWR관련 스키마 내용을 servlet.xml파일에 추가



- ❖ Spring Dwr 설정2
  - DWR 관련 자바스크립트 요청에 대한 컨트롤러 설정

<dwr:controller id="dwrController" />



- ❖ Spring3 Dwr 설정3
- ❖ DWR자바스크립트 요청에 대한 HandlerMapping 설정



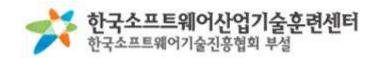
- ❖ Spring Dwr 설정4
  - 스프링 빈객체에 <dwr:remote>를 이용한 설정



## Spring Dwr 구현

- ❖ Spring Dwr 구현
  - 자바스크립트에서 자바객체를 호출





# Spring & Ibatis 프레임워크 과정 (Spring Transaction)

 $11.8.27 \sim 11.10.15$ 

한국소프웨어기술진흥협회(KOSTA)



## **Spring Transaction**

- ❖Spring Transaction 개요
  - 트랜잭션은 성공적으로 처리되거나 또는 하나라도 실패하면 완전히 실패 처리를 해야 하는 경우에 사용
  - 스프링에서 트랜잭션 관리 기능을 지원하고, 간단한 설정으로 트랜잭션 처리 가능함

- 코드기반처리 방식, 선언적 트랜잭션, 어노테이션 기반 트랜잭션



#### JDBC 기반 트랜잭션 관리자 설정

- JDBC와 iBatis와 같은 JDBC를 이용해서 DB연동 처리하는 경우 DataSourceTransactionManager를 트랜잭션 관련자로 설정



## TransactionTemplate을 이용한 트랜잭션처리

- TransactionManager를 통하 TransactionTemplate 설정



## TransactionTemplate을 이용한 트랜잭션처리

❖ DAO클래스 구현

```
return transactionTemplate.execute(
new TransactionCallback<ModelAndView>() {
@Override
 public ModelAndView doInTransaction(TransactionStatus status) {
  Try{
     addOrder(<u>order</u>);//order 추가
   }catch(Exception e){
   status.setRollbackOnly();
  return mav;
}//end doInTransaction
});//end execute
```



### 선언적 트랜잭션 구현

- <tx:advice>태그를 이용한 트랜잭션 처리
- servlet.xml 스키마 추가

xmlns:tx=<u>http://www.springframework.org/schema/tx</u> <u>http://www.springframework.org/schema/tx</u> <u>http://www.springframework.org/schema/tx/</u>spring -tx-3.0.xsd"



### 선언적 트랜잭션 구현

- <tx:advice>태그를 이용한 트랜잭션 처리

```
<tx:advice id= "txAdvice" transaction-manager= "transactionManager">
 <tx:attributes>
   <tx:method name= "order" propagation="REQUIRED"
     rollback-for= "Exception"/>
  </tx:attributes>
</tx:advice>
<aop:config>
  <aop:pointcut expression= "execution(public * kosta.controller..*(..))"
     id= "publicMethod"/>
   <aop:advisor advice-ref= "txAdvice"
     pointcut-ref= "publicMethod"/>
</aop:config>
```



### 선언적 트랜잭션 구현

- 어노테이션 기반 트랜잭션 설정

```
<tx:annotation-driven transaction-manager= "transactionManager"/>
@Transactional(propagation=Propagation.REQUIRED,
rollbackFor={Exception.class})
public ModelAndView order(int amount2, String id)throws
Exception{
 if(item.getAmount() < amount2){</pre>
  throw new Exception("재고부족");
 addOrder(order);//order 추가
 updateItem(of);//item 재고수정
return mav;
```