에이콘 아카데미

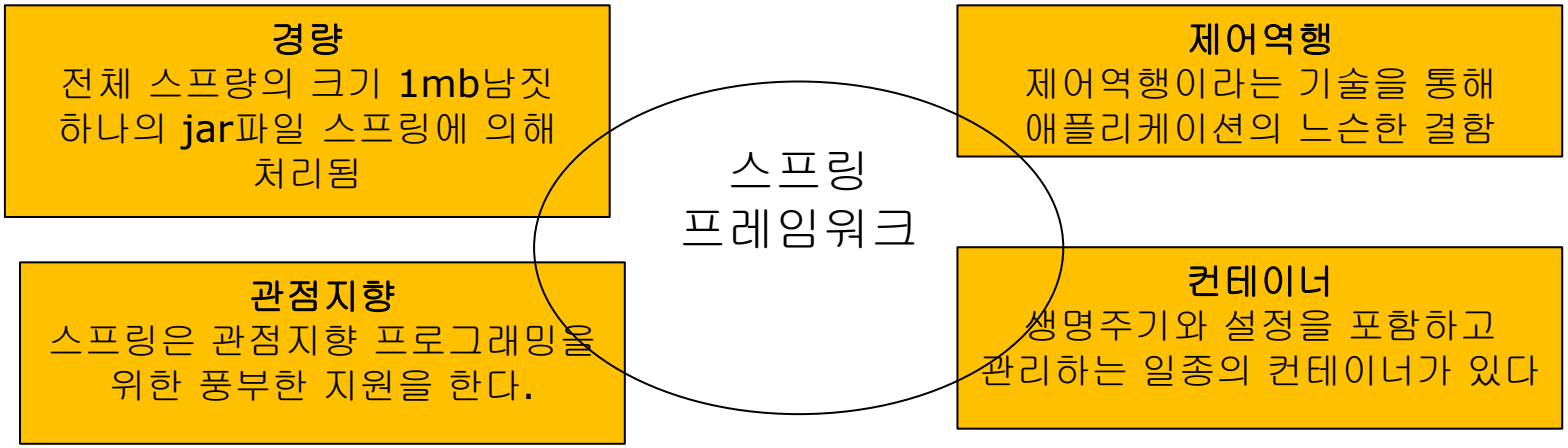
spring시작

zelratole@hanmail.net

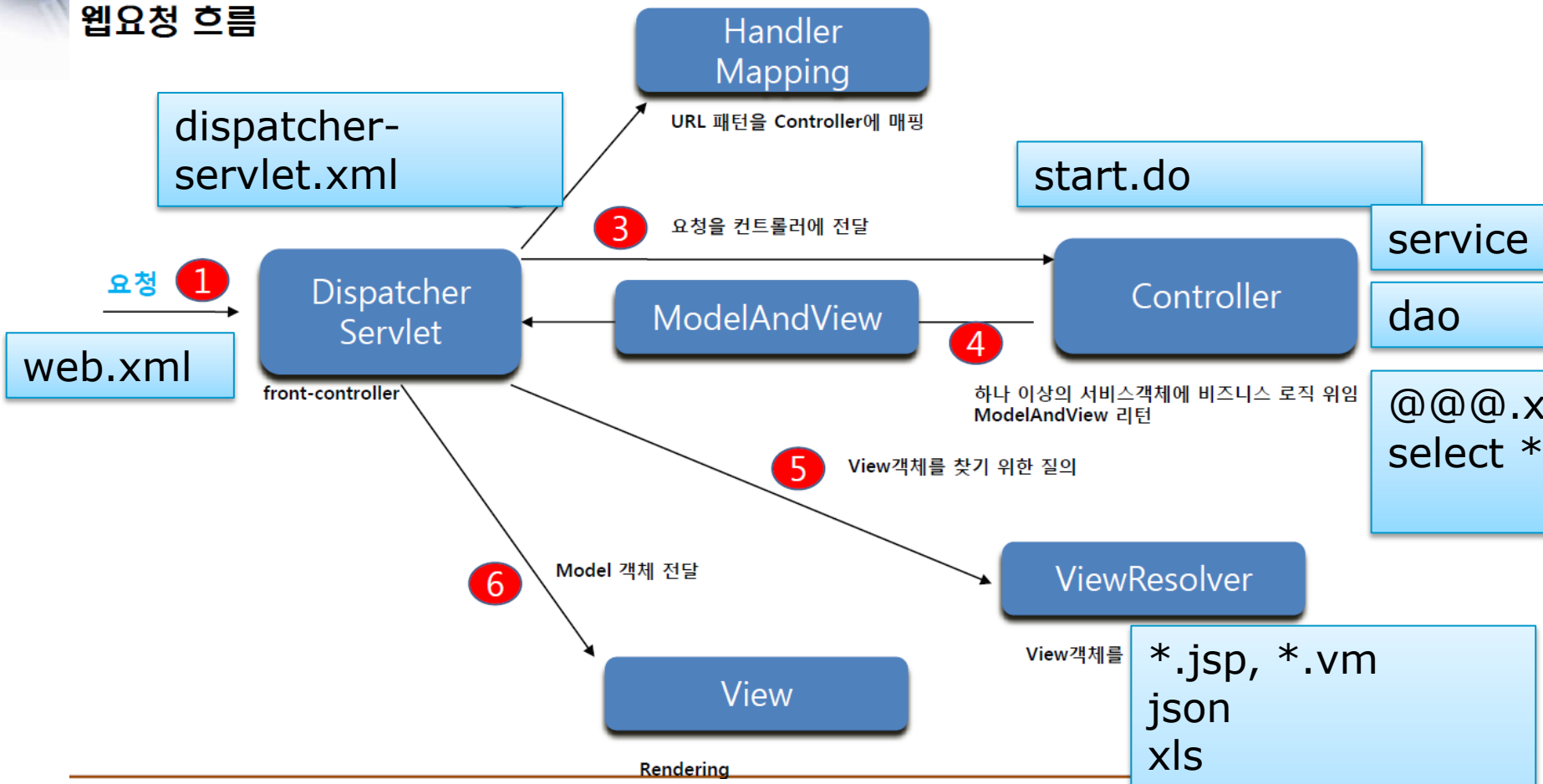
스프링 프레임워크

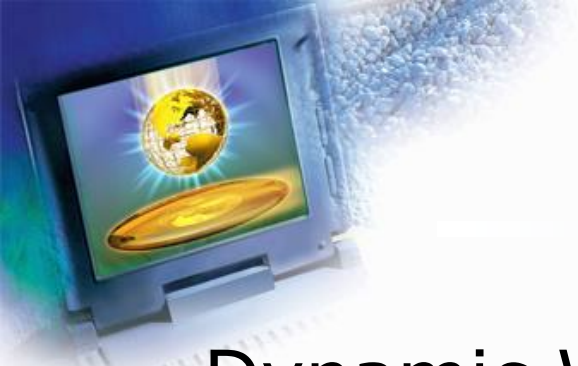
- Rod Johnson이 만든 오픈 소스 프레임워크
- 복잡한 엔터프라이즈 애플리케이션 개발을 겨냥
- 단순성, 테스트 용이성, 느슨한 결합성의 측면이 스프링의 이점을 제시하고 있다.

경량의 제어 역행(DI)과 **관점지향(AOP) 컨테이너** 를 구성되어 있는 프레임워



웹요청 흐름


Handler Mapping
URL 패턴을 Controller에 매핑
dispatcher-servlet.xml
start.do
service
dao
@@@.xm
select * fr
3
요청을 컨트롤러에 전달
요청
1
Dispatcher Servlet
ModelAndView
Controller
4
web.xml
front-controller
하나 이상의 서비스객체에 비즈니스 로직 위임
ModelAndView 리턴
5
View객체를 찾기 위한 질의
6
Model 객체 전달
ViewResolver
View객체를
*.jsp, *.vm
json
xls
이미지, *.zip
View
Rendering



확인예제

- Dynamic Web Project
  - springweb2를 만들고, spring에서 처리하는
    - lib
    - web.xml
    - dispatcher-servlet.xml
- Controller 선언.
  - springweb2.a01_start.A01_CallController.java
- View
  - WebContent
    - a01_start/a01_call.jsp

스프링 모듈

- core
  - 프레임워이가장 기본적인 부분이고, 의존성 삽입 (**D**ependency **I**njection)기능을 제공한다.
- DAO
  - jdbc 코딩과 데이터베이스 업체별 특정 처리할 필요없는 jdbc 추상화 레이어를 제공
- ORM
  - 객체 관계 매핑 API를 위한 통합 레이어 제공한다. Mybatis를 활용해서 DB 처리를 효율화 한다.
- Web
  - 화면 View뿐만아니라, 웹에서 파일업로드, 다운로드
- MVC
  - 웹 애프리케시션의 모델2 패턴을 스프링에서 지원

스프링 개발환경처리

- 개발환경구성방식
  - maven 자동처리
  - lib 추가 처리
- 프레임워구조
  - web.xml
    - spring프레임워을 사용할 수 있게 끔. 호출 처리
    - config : 스프링 프레임워크의 설정 위치
    - 한글처리 filter
  - dispatcher-servlet.xml
    - 스프링의 환경 설정 구체적인 내용
      - mapping(model, view, controller), lib, database등 처리
    - 필요한 lib의 설정 위치를 지정 - mybatis
    - resource에서 상세 처리 지정

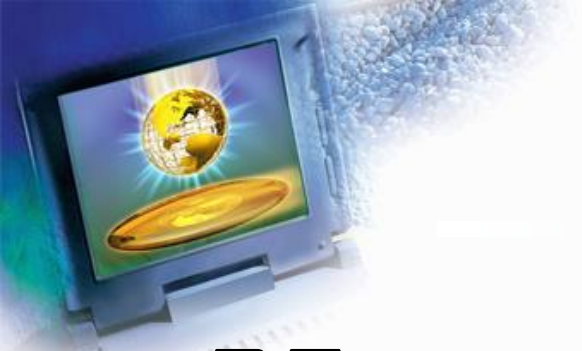
web.xml

- url을 통해서 웹서버를 호출할 때, 가장 먼저 호출되는 곳
  - http://localhost:8080/springweb
- front단 controller 선언(DispatcherServlet)
  - 공통으로 스프링에서 지원하는 servlet 선언
  - 어떤 url패턴으로 스프링의 servlet이 호출되는 지도 선언..
    - *.do ==> 스프링에 사용하는 servlet을 선언
    - default 스프링 컨테이너 설정파일
      - <servlet-name>으로 설정한 이름 dispatcher.xml
- 스프링에서 지원한는 한글 encodingFilter 선언(CharacterEncodingFilter)

- dispatcher.xml에서 controller 등록
- controller
  - 스프링의 공통 servlet인 DispatcherServlet의 다음 단계의 내용을 처리..
  - 요청된 내용 받고, model단으로 요청처리에 필요한 business logic을 처리
    - Service, Dao
  - view단에 넘겨줄 model를 매핑시킨 후,
  - view단 호출
- View단 jsp호출

스프링프레임웍 프로세서

- 시작
  - http://localhost:8080/springweb2/start.do
- WEB-INF 하위에 web.xml 호출
  - 공통 스프링 모듈 dispatcher Servlet
  - servlet name으로 등록된 이름을 기준으로
  - servlet이름.xml을 WEB-INF에서 설정파일로 찾음
- dispatcher-servlet.xml (스프링관련설정)
  - 웹 컨테이너 로딩 시, controller를 메모리에 등록
  - 필요한 모듈 선언해서 메모리에 loading
    - viewresolver: 화면단 호출에 대한 방식 처리
- mapping handler를 통해 start.do를 controller 내에서 찾음.
  - @RequestMapping("start.do")



Controller

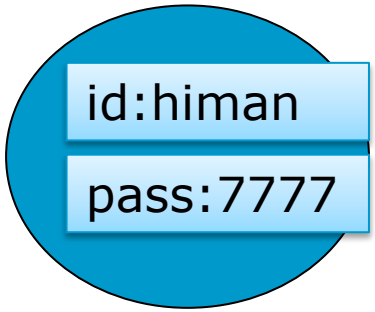
- 구조
  - @Controller 클래스명 위에 선언
  - @RequestMapping("mapping이름")
    - http://localhost:8080/springweb/mapping이름
    - 컨트롤러에 바로 선언, 메서드에 선언
  - 메서드 선언규칙
    - public String 메서드명(외부전달값, Model)
      - 모델단 처리
      - return View단 호출
      - ex) start.do?id=홍길동&pass=7777

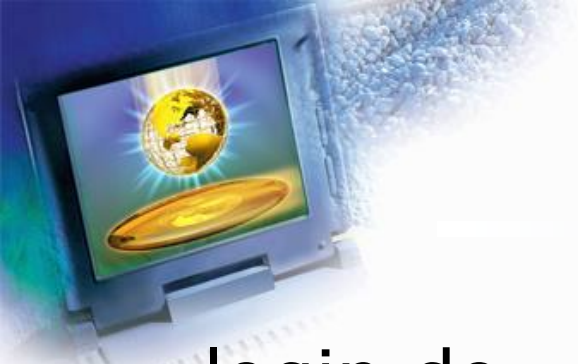
요청 처리 및 모델 처리

- http://localhost:8080/springweb/start.do
  - 요청값 처리(query string)
    - ?id=himan&pass=7777
  - 받는 데이터 : 모델
- @requestMapping("/start.do")
- public String start(요청값전달객체, 뷰단 보내는 객체){
  - return "뷰단 jsp"
- }
  - public String start(Member m, Model d)

요청값 받는 클래스 선언

- ?id=himan&pass=7777
- public String start(Member m, Model d) <<Member>>
  - Member class에서 property id, pass
    - setId(String name)
    - setPass(String pass)
  - 뷰단에 모델 데이터..
    - d.addAttribute("call", "call me");
    - view ==> ${call}
- return "view단 jsp호출";





post 방식 id, pass 체크

- login.do
  - ID: @@@@
  - PASS: @@@@
  - [로그인]
- 맞으면 msg 로그인 성공
- 틀리면 msg 로그인 다시 하세요..

확인예제

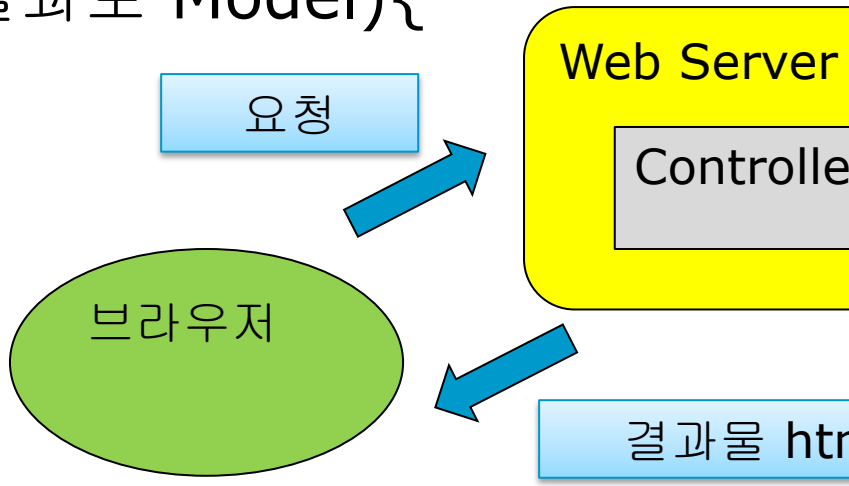
- 초기 화면
  - @@ + @@
  - 정답입력:[ ]
  - [확인]
- 처리 내용
  - 정답 일 때, 페이지 전환 ==> pass!!!
  - 오답 일 때, 다시 문제 페이지로 alert("오답입니다!")

숙제:

- 구매 물건 계산
- 물건명 가격 갯수
- 사과 @@@ [ ]
- 바나나 @@@ [ ]
- 딸기 @@@ [ ]
- [총비용확인]
- ==> 다음 page
  - 총비용 @@@ 입니다.

Controller를 통한 요청처리

- 요청
  - http://localhost:8080/springweb/**callctrl.do?id=himan&pass=7777**
- 자바 Controller
  - @RequestMapping("**callctrl.do**")
  - public String call(요청, 결과로 Model){
  - 요청 DTO
  - 결과 Model : jsp단
    - return "/@@@.jsp";
  - }



요청 처리를 해주는 여러 객체들

- public String call(@@@@, Model d){}
  - 1. 클래스 선언
    - Member m : ?id=himan&pass=7777
  - 2. 한개 단위 입력값 처리
    - @RequestParam(value="id", defaultValue="입력없음") String id
      - value : query string key를 지칭
      - defaultValue : 실제 string key가 없거나, 데이터를 할당하지 않을 경우, 초기치 설정
        - start.do?XXXXX

@RequestParam활용 물건구매

- 초기화면 요청 Controller, View

물건명	[ ]
가격	[ ]
갯수	[ ]
[구매요청]	

- 처리 Controller 및 화면 View
  - 요청하신 물건은 @@@ 이고, 단가는 @@@ 이면, 갯수는 @@@ 이어서 총 비용이 @@@ 입니다.

연습 확인예제(숙제)

- 회원등록(초기화면)

id	
password	
이름	
이메일주소	
포인트	
[회원 등록]	

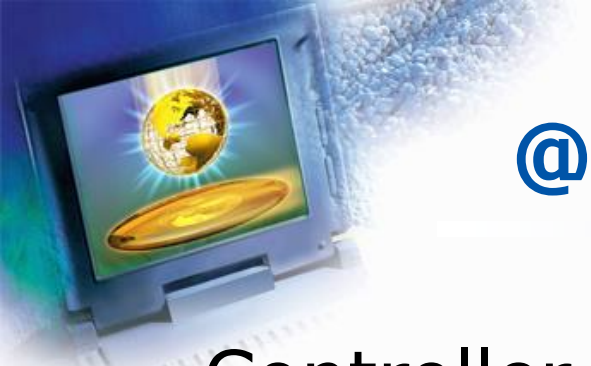
- 등록 결과
  - @@@(이름)님 등록 결과입니다.
  - id: @@@, 패드워드 :@@@, 이메일 주소:@@@
  - 초기가입 포인트+1000 드립니다. 포인트 :@@@

get/post 처리

- Controller에 공통 RequestMapping annotation 설정, get/post 방식 처리..
- @RequestMapping(value="요청 주소",
  - method=**RequestMethod**.GET)
    - 해당 요청 주소로, get방식을 처리했을 때
- @RequestMapping(value="요청주소",
  - method=**RequestMethod**.POST)
    - 해당 요청 주소로, post방식을 처리했을 때.

@ModelAttribute

- 요청 처리 객체의 내용을 모델로 데이터를 넘길 때, 요청+model을 기능을 하는 객체를 말한다.
- 요청처리
  - public String call(Member m, Model d){
    - Member : 요청 데이터 입력.
    - Model :View단 넘기는 처리..
    - ==> @ModelAttribute("이름") 클래스명 으로 요청과 model 한꺼번에..
  - }



@ModelAttribute를 통한 수학

- Controller 생성 (공통 RequestMapping)
  - get방식 메서드(@ModelAttribute선언)
    - 랜덤으로 문제 처리
  - post방식 메서드
    - 랜덤으로 문제 결과 처리..
- View단 구성

ModelAttribute
확인예제

- 구매 금액 누적 확인
- 물건 가격 :[ ]
- 물건 갯수 :[ ]
- [구매완료]
- 현재 누적 금액: [ ]

@ModelAttribute

- 해당 클래스를 controller에서 공통을 활용할 수 있다..
- Controller
  - @ModelAttribute("모델이름선언")
  - public Product getProduct(){
    - return new Product();
  - }

공통 ModelAttribute 설정

- 계산기
  - [ ] select[사칙연산] [ ] = [ ]
  - [계산 결과]

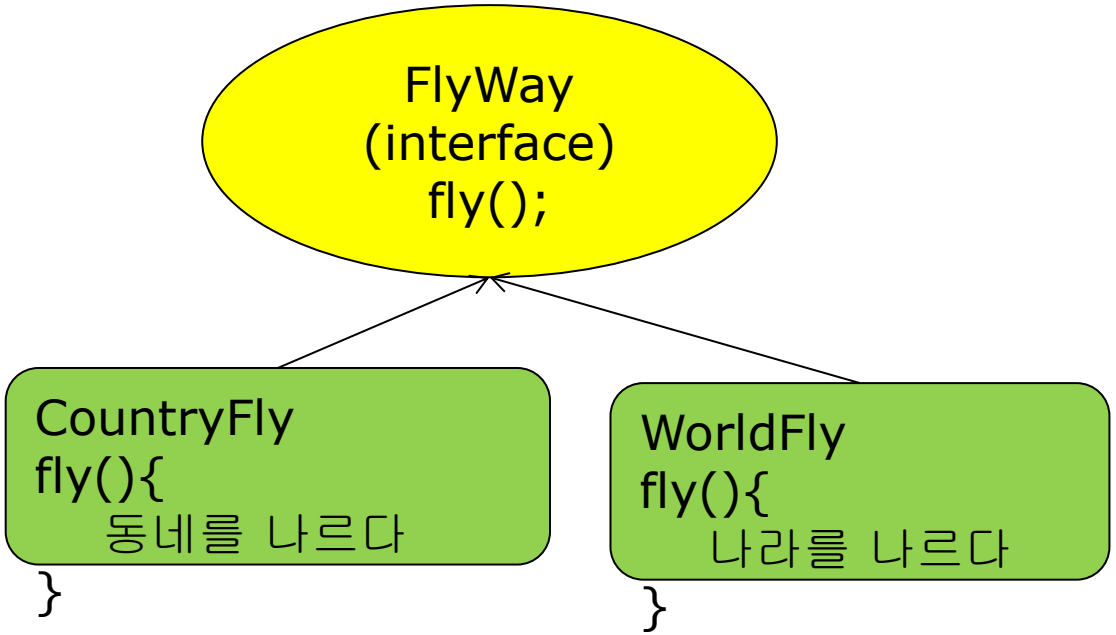
화면단..

@@@.jsp

${person.name}	

DI(Dependency Injection)

- 의존 주입
- 조립기를 사용하여, 기존 클래스에 영향이 없이 서로 간에 **type**을 확인하여 해당 객체를 삽입하는 **code** 처리 방식을 말한다.




```
조립기 클래스

FlyWay f

void setFlying(){
    f = new CountryFly();
    f = new WorldFly();

}
```

spring 조립기

- dispatcher-servlet.xml에서 정의된 클래스는 WAS 로딩 시, 사용자 정의 클래스나 프레워 기반을 클래스가 실제 객체가 메모리에 로딩되는 처리
- 조립기에 객체 생성으로 할당 하기에, 실제 클래스 코드에서는 new XXX 코드가 사라지게 된다.

조립기에서 객체 생성 데이터 할당

- main객체 생성
  - Person **p** = new Person();
  - **p**.setName("홍길동");
  - **p**.setAge(25);
- 조립기
  - <bean id="**p**" class="패키지명...Person">
  - <property name="name" value="홍길동" />
  - <property name="age" value="25" />
  - </bean>

컨테이너 내용 확인.

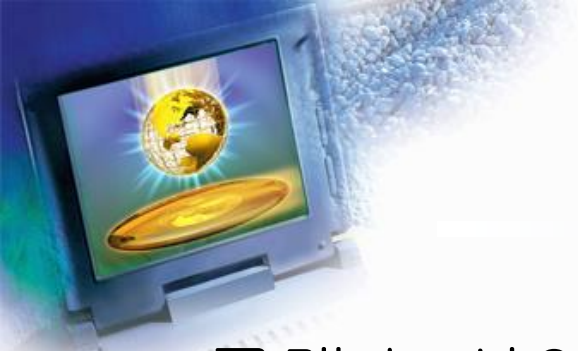
- xml 로딩
  - Resource rs = new ClassPathResource("경로/diexp01.xml");
- xml에 있는 bean을 호출.
  - BeanFactory : bean객체 호출
  - XmlBeanFactory : Resource 에 있는 xml bean 객체 호출
    - .getBean("xml에 선언된 id")
    - Person p = (Person)bean.getBean("**p**");
    - p.getName(), p.getAge()

확인예제

- DTO :Weather
  - 날짜 : 2017.6.1
  - 지역 : 서울 강남
  - 정보 : 오전한때 비
- A02_DI_Exp.java
  - 상단 내용 출력 처리
- diexp02.xml
  - DTO bean 등록

생성자를 통한 할당

- DTO : 생성자를 통하여 데이터 할당할 수 있도록 선언
  - Person(String name, int age)
  - Person p = new Person("홍길동", 25);
- xml bean 선언
  - <bean>
    - <constructor-arg value="홍길동"/>
    - <constructor-arg value="25"/>
  - </bean>



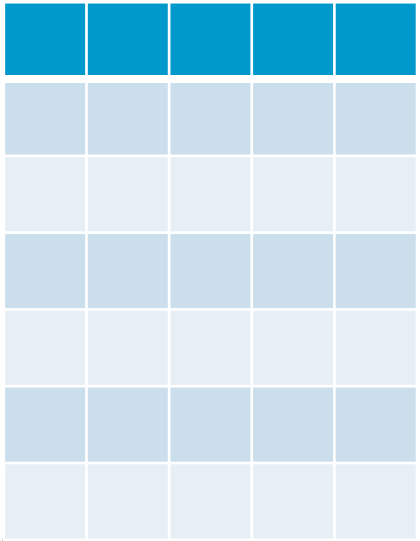
객체에 객체 할당 처리..

- 클래스 선언..
  - class Woman{}
  - class Man{
    - Woman w;
    - public void setWoman(Woman w){
      - this.w = w;

- 조립기(Container) 처리
  - <bean id="woman" class="@@@.Woman"/>
  - <bean id="man"
    - <property name="woman" ref=" woman "/>

객체에 객체 할당 처리..

- Main()나 외부 클래스에서 호출 형태..
  - Woman woman = new Woman();
  - Man man = new Man();
  - man.setWoman(woman);



- 조립기(Container) 처리
  - <bean id="woman" class="@@@.W
  - <bean id="man" class="@@@.Man"/>
    - <property name="woman" ref="woman "/>

확인예제

- Product와 Mart를 autowire로 처리 후,
- 해당 selling()를 호출 후, 출력하세요..
- diexp08_auto.xml
- A08_GenericXml.java

확인예제(byType, constructor)

- BookStore
  - 서점 이름
  - Book b;
  - 생성자(Book display)
  - void displayBook(){
    - @@@ 서점에서
    - 전시하는 도서는 @@@, 가격@@@
- Book
  - 도서명
  - 가격

A11_GenericXml.java
diexp11_auto.xml
BookStore.java
Book.java

autowire="constructor"

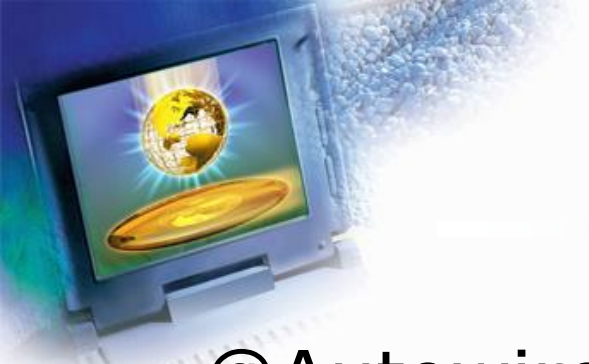
각 java에서 autowire 설정

- 조립기 선언
  - <context:annotation-config/>
- 각 클래스에서 autowire 및 annotation을 설정 처리..

@Autowired 확인예제

- 음식점
  - 가게이름
  - 음식(autowired)
  - void eatLunch(){
    - 오늘 점심은 @@@ 가서
    - @@@를 @@@원으로 먹기로 했다.
- 음식
  - 종류
  - 가격

A19_GenericXml.java
diexp19_auto.xml
Restaurant.java
Food.java

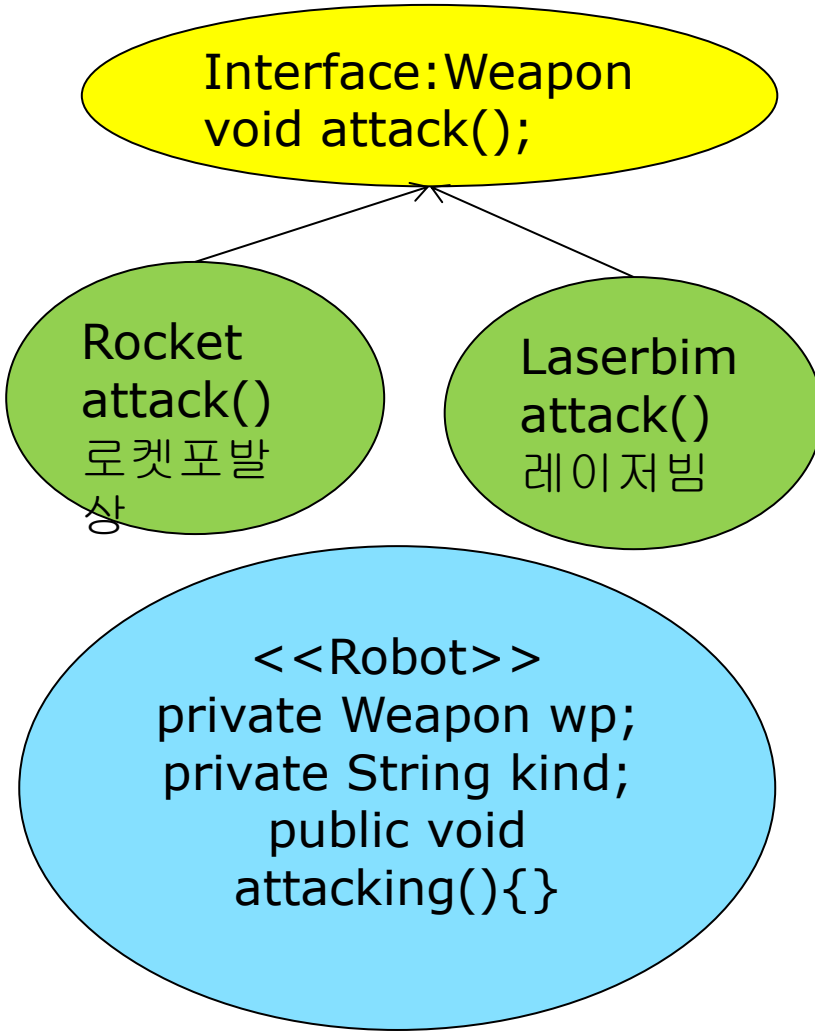


@Qualifier

- @Autowired로 설정된 객체가 두가지 이상 type이 할당 될에 에러발생..
  - 할당할 수 있는 객체를 선택해야 할 필요가 있을 때, id값을 @Qualifier("bean ID")을 선언한다.
- @Autowired에서 해당 객체가 메모리로 할당이 안될 수도 있을 때, 반드시 할당해야 되는 것인 경우에는 에러발생. 이를 방지하지 위해 필수는 아니라는 옵션 설정으로 에러방지..
  - @Autowired(required=false)

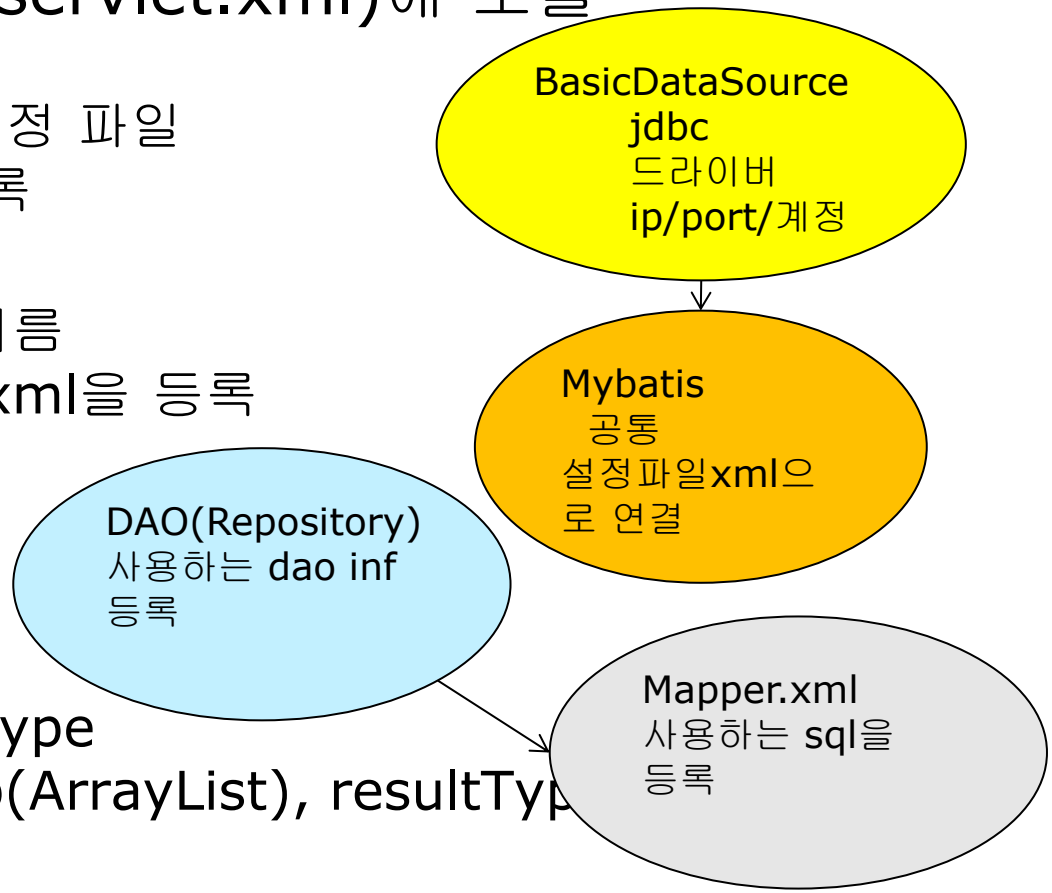
@Qualifier 확인예제

- Robot.java
- Weapon.java
- A20_GenericXml.java
- diexp20_auto.xml
- 출력물
  - @@@ 로봇이 @@@공격을
  - 하다.



Mybatis연결

- 컨테이너(dispatcher-servlet.xml)에 모듈
  - DB 연결정보
  - mybatis 모듈 등록, 설정 파일
  - dao interface 위치 등록
- Mybatis 설정
  - 공통 vo를 등록, alias이름
  - 업무별 XXXXmapper.xml을 등록
- Mapper.xml
  - namespace 설정
  - resultMap 선언..: vo
  - sql 선언 :select
    - 입력값 :paramterType
    - return : resultMap(ArrayList), resultTyp
    - update, insert



dispatch-servlet.xml(조립기)

- 연결:*BasicDataSource : dataSource*
  - 드라이버
  - DB서버 정보(ip, port, sid, 계정, 비밀번호)
- mybatis 설정
  - 연결 정보를 받아서 mybatis와 설정..
    - dataSource ref => *dataSource*
  - mybatis 공통 설정 config xml파일 선언
    - configLocation
- Repository(DAO) 설정
  - *MapperScannerConfigurer를 통해서 dao단의 패키지를 선언.*

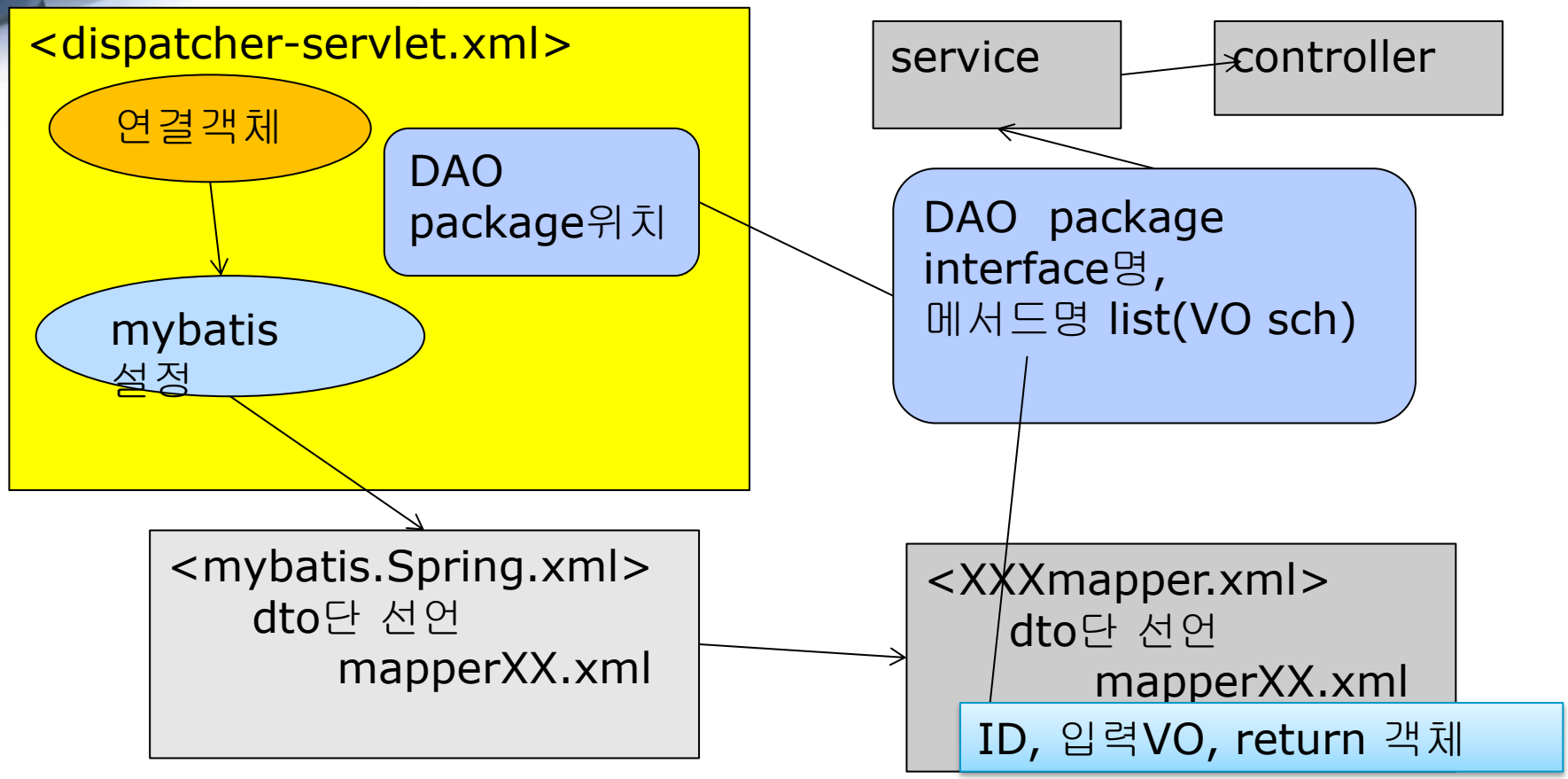
mybatis 공통 설정 config xml

- *classpath:resource/mybatis.Spring.xml*
  - classpath:src를 기준으로 잡힘.
  - resource폴드에 mybatis공통 설정 파일인 mybatis.Spring.xml
- 주로 하는 역할
  - 모듈별로 실제 처리할 sql 파일(xml)을 등록처리
    - <mapper resource="EmpMapper.xml"/>
  - 공통 VO, DTO등을 alias으로 지정, sql파일에서 활용성을 높이기 위해, 간단한 이름으로 설정.
    - *springweb.z01_dto.Emp ==> emp*

XXXMapper.xml

- mybatis 설정 config.xml 에서 모듈별로 sql을 사용할 수 있게끔 처리하는 파일
  - sql 처리 :select * from
  - 인식할 수 있는 id명 : dao단에서 호출 시 필요
    - hello.CallDao.callList
  - 입력관련 데이터 값 처리 parameType="CallDto"
  - return값에 대한 처리 : resultMap =VO가 모인 ArrayList형태
  - namespace : hello.CallDao를 선언하면 id에 namespace명 생략가능
  - DAO단 interface와 밀접한 연관관계
  - dao단 interface 패키지명.인터페이스명.메서드명
    - package hello;
    - public interface CallDao{
    - public ArrayList<Vo> callList(CallDto dto);

호출 상관관계도


<dispatcher-servlet.xml>
연결객체
DAO
package위치
mybatis
설정
service
controller
DAO package
interface명,
메서드명 list(VO sch)
<mybatis.Spring.xml>
dto단 선언
mapperXX.xml
<XXXmapper.xml>
dto단 선언
mapperXX.xml
ID, 입력VO, return 객체

mybatis MVC 패턴

- Controller

- Service
- Repository(Dao)
- mybatis SQL (*.xml)
- 화면단(*.jsp)

- select * from DEPT 결과물을 처리하는 DAO 만들기
- VO : 단위 데이터를 저장할 객체
  - Dept
- DAO단(Repository)
  - interface DeptDao
  - 메서드명 detpList
    - public ArrayList<Dept> deptList();
- mybatis.Spring.xml
  - Dept의 alias이름 설정
  - DeptMapper.xml 생성 및 선언
- DeptMapper.xml
  - namespace 선언
  - <resultMap id="deptRsMap" type=""
  - <select id="" resultMap=""

DAO 만들기 연습

- select * from member로 데이터 가져오기
- DTO
- DAO단
  - 메서드선언
- mybatis.Spring.xml 처리
- memberMapper.xml 처리

autowire

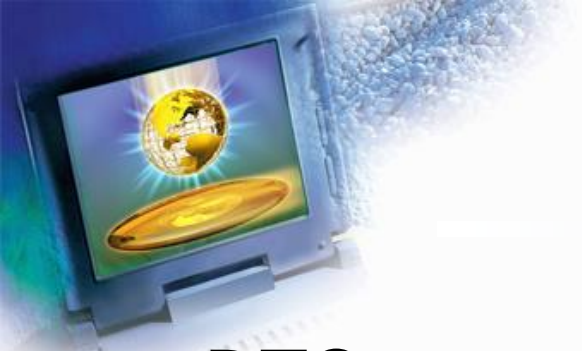
- 자동 객체를 해당 정의된 객체에 할당.

type casting 없이 bean활용

- main()
  - GenericXmlApplicationContext ctx
  - Person p =ctx.getBean("p", Person.class);

생성자와 객체할당 예제

- Product
  - 물건명 가격 : 생성자로 할당..
- Mart
  - 마트이름
  - Product
  - selling()
    - 어서오세요 @@@ 마트 입니다.
    - 오늘 추천 물건..
    - @@@, @@@@원으로 모십니다!!
- A04_GenericXml.java, diexp04.xml



(1:다)에 대한 조립기 처리..

- DTO
  - class Mart
    - private ArrayList<Product> prodlist;
    - public void setProdlist(ArrayList<Product> li){
      - this.prodlist = li;

```
main()
    ArrayList<Product> buyList = new XXXX();
    buyList.add(new Product("사과",3000));
    buyList.add(new Product("바나나",4000));
    mart.setProductlist(buyList);
```

조립기에서 처리..

- <bean id="**prod01**" class="@@@">
  - <property name="name" value="사과"/>
- <bean id="**prod02**" class="@@@">
  - <property name="name" value="바나나"/>
- ....
- <bean id="mart01" class="@@@">
  - <property name="prodlist">
    - <list>
      - <ref bean="**prod01**"/>
      - <ref bean="**prod02**"/>

(1:다) 조립기 처리

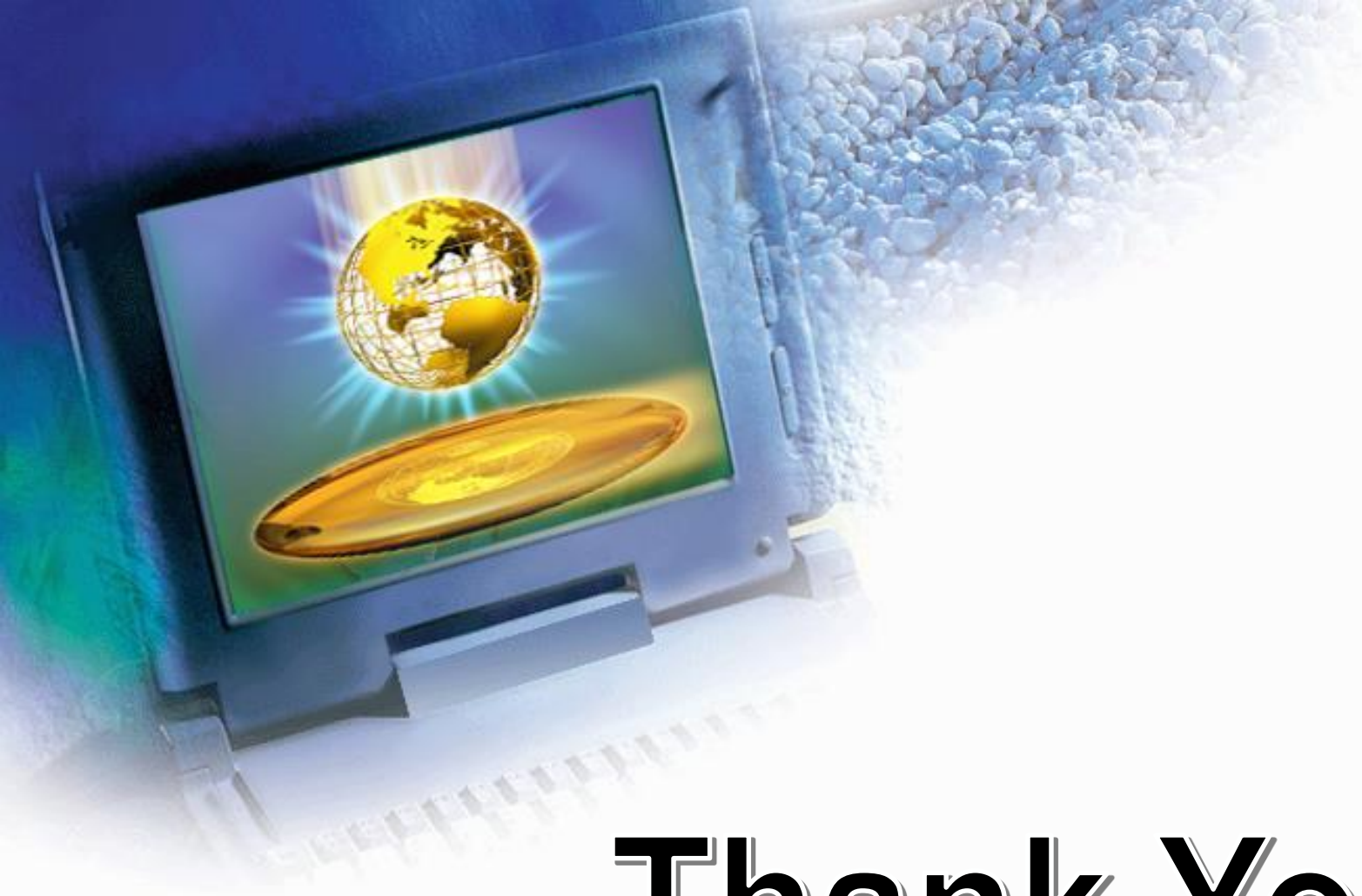
- Schedule
  - 시간, 내용
- DailySchedule
  - 날짜(문자열)
  - ArrayList<Schedule> schList;
  - showSchedule()
    - @@월 @@일 일과계획
    - 6:00 기상
    - 7:00 일과계획
    - 8:00 @@@ 출발..

controller단 입력과 호출처리

- /springweb2/start.do?id=홍길동&pass=7777
- @RequestMapping("호출하는 mapping이름")
- public String form(입력값 형식)
- 입력값형식
  - @RequestParam("key값") 할당받는 데이터 형식 변수명
    - ex) @RequestParam("id") String name
    - @RequestParam("pass") int pass
  - VO :Member에 setXXX,getXXX 있으면 자동으로 객체에 입력처리 됨..
    - Member mem
      - mem.getId(), mem.getPass() 데이터가 입력되어 있음.
    - ex) public String form(Member mem){
    -
      - }

확인예제

- controller
- A03_ParamExp.java
- view a03_ParamExp.jsp
- url
  http://localhost:8080/springweb2/param2.do
- * ?procname=사과&price=3000&cnt=3
- 출력물 물건명 :@@@ 가격 :@@@ 갯수:@@@ 총계:@@@



Thank You !

zelratole@hanmail.net