

# Spring Roo와 함께 하는 쾌속 웹개발

정상혁 , KSUG ([www.ksug.org](http://www.ksug.org))



# 목차

1. Tool

2. Demo

3. Application

---

# 1. Tool

1.1 개요

1.2 Command line shell

1.3 Round-trip

1.4 익숙한 도우미들

# 1.1 개요

---

## Text Based RAD Tool for Java

- Real Object Oriented의 첫 글자들
- Spring, Maven, JUnit 등 바탕의 코드 자동생성
- Round-trip 지원
- No Lock in, No runtime dependency
  - Roo를 위한 인터페이스, 슈퍼클래스 없음
  - Roo의 특성을 제거하기 쉬움



# 1.2 Command Line Shell

---

## 똑똑한 Shell 제공

- Tab completion
- Hint, Help
- Context-aware, Command hiding
- Script 기록, 실행
- 명령수행이 Transactional
  - 작업수행 중 이상하면 undo
- Backup

# 1.3 Round Trip

---

## 양방향 코드 자동생성

- 도구와 사람이 건드리는 파일 분리
  - 사람은 \*.java, \*.xml만 신경쓰면 됨
- File system monitoring
  - Shell 이 떠 있는 동안에 자동 감지
  - .java , .xml 파일의 변화를 .aj에 반영
- UI
  - 자동생성 여부는 옵션

# 1.4 익숙한 도우미들

---

## Java환경의 지원 도구들

- Maven
  - Library 관리. Build lifecycle
  - AspectJ 컴파일
- JUnit
- Selenium
- Eclipse
  - Spring Tools Suite
  - M2Eclipse
  - AJDT (Aspect J 지원 Plug-in)



## 2. Demo

2.1 Demo 진행

2.2 소스 분석

## 2.1 Demo 진행

---

### 작업 내용

- 프로젝트 생성
- Entity 추가
- Controller 추가
- Selenium 테스트 추가
- 테스트 코드 실행
- Tomcat에서 실행
- Selenium 테스트 실행

## 2.1 Demo 진행

---

### 작업 내용

- Roundtrip 확인
  - Getter 추가, field 삭제
- Script 저장, Script 실행
- Backup
- Roo 제거



## 2.2 소스 분석

---

### 생성결과

- 모든 소스는 생략된 것이 아니고, 기능을 다 하는 클래스
- Entity class
  - 소스에는 getter/setter가 없지만 Mixin으로 생성됨

```
@Entity
@Entity
@RojavaBean
@RotoString
public class Guest {
    @Size(max = 30)
    private String name;
    private Integer price;
    private Boolean speical;
}
```

## 2.2 소스 분석

---

### 생성결과

- Controller class
  - REST의 CoC

```
@RooWebScaffold(automaticallyMaintainView = true,  
formBackingObject = Guest.class)  
@RequestMapping("/guest/**")  
@Controller  
public class GuestController {  
}
```

## 2.2 소스 분석

---

### 생성결과

- Test class
  - Assert 로직도 Mixin으로 삽입됨

```
@RooinTEGRATIONTest(entity = Guest.class)
public class GuestIntegrationTest {
    @Test
    public void testMarkerMethod() {
    }
}
```

---

# 3. Application

3.1 구성 기술과 구조

3.2 Spring 3.0

3.3 AOP

3.4 ORM

3.5 확장

## 3.1 구성기술과 구조

---

### Full Stack, CoC

- 모든 Layer 기술을 다 제공
- Seam, Ruby on Rails처럼 강한 주장이 있는 프레임워크
- Convention Over Configuration
  - Best practice를 Convention으로 심음

## 3.1 구성기술과 구조

---

### 단순 2단 구조

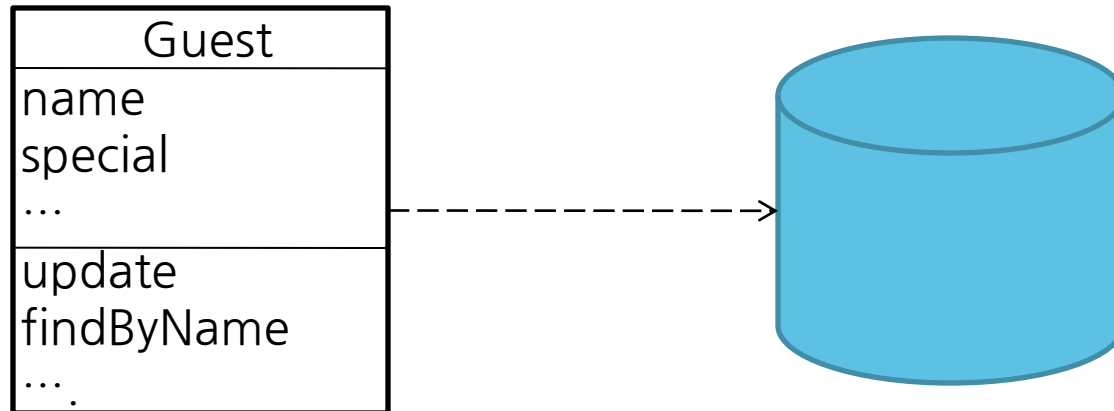
- Entity + Controller
  - Java 파일 단 2개
  - Active Record 패턴 적용
- DAO(Data Access Object)와의 이별
  - Aspect로 관심사의 분리
  - Static 메소드 테스트 기능 제공

## 3.1 구성기술과 구조

### Active Record Pattern

- Domain object에 Data access logic이 들어감

```
Guest guest = Guest.findByName("박성철");  
guest.setSpecial(true);  
guest.update();
```



## 3.2 Spring 3.0

---

### Spring 3.0 기능활용

- REST convention support

```
http://shopping.com/shop/product/  
http://shopping.com/shop/product/1  
http://shopping.com/shop/product/1/form
```

- Beans Valididation (JSR 303)

```
public class Employee{  
    @NotNull @Size(min=1, max=25)  
    private String name;  
  
    @NotNull @NumberFormat(style=Style.CURRENCY)  
    private BigDecimal income = new BigDecimal("1000");
```

## 3.2 Spring 3.0

---

### Spring 3.0 기능활용

- MVC custom name space

```
<mvc:view-controller path="/login"/>  
  
<mvc:interceptors>  
  <bean class="...ThemeChangeInterceptor"/>  
</mvc:interceptors>
```

## 3.3 AOP

---

### Bytecode weaving 적극 활용

- Roo의 자동 생성 부분을 담당
- Aspect J의 Inter-type Declaration을 이용한 Mixin
  - Abstract subclassing, static crosscutting
  - Compile time에서 코드 삽입
  - 성능 손해가 없음
  - Maven plugin과 IDE의 도움으로 별도 컴파일 필요 없음
- @Configurable을 이용한 투명한 DI 기술
  - new로 생성되는 객체에도 DI 적용 가능

## 3.3 AOP

---

### Bytecode weaving 적극 활용

- Inter-type Declaration 으로 해결한 코드들
  - @Configurable 선언
  - Getter, Setter
  - 기본 CRUD기능
  - Finder
    - 검색용 메소드 (findByName 류)
  - Entity의 복수형
    - CoC를 위해서 (Guest->Guests)
  - toString 메소드

# 3.4 ORM

---

## ORM에 대한 걱정들

- Object Relational Mapping
- 성능이 안 좋다?
  - 예측 가능한 SQL로 대응 용이
  - 캐쉬, 분산환경 적용에 유리
    - 예) Tangosol Coherence Cache
- 복잡한 쿼리는 불가능하다?
  - 다양한 매핑 방식 지원 (Lazy loading, Join)
  - 필요하면 Native SQL로 호출가능

## 3.4 ORM

---

### ORM의 확산

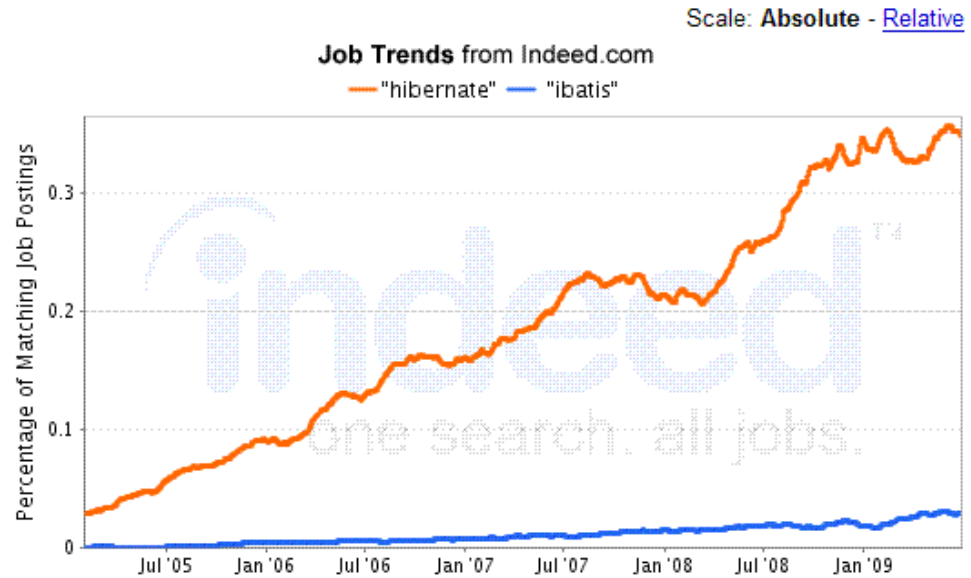
- 다른 언어에서도..
  - PHP(Zend, Codeigniter), Python(Django), Ruby등의
- Java 프레임워크도 ‘생산성’이라는 말을 위해서는..
  - JBoss Seam
  - Play framework
  - AribaWeb
  - Spring ROO

# 3.4 ORM

## ORM의 확산

- It job trend

### "hibernate", "ibatis" Job Trends



Indeed.com searches millions of jobs from thousands of job sites.  
This job trends graph shows the percentage of jobs we find that contain your search terms.

Find ["hibernate" jobs](#), ["ibatis" jobs](#)

출처 : <http://www.indeed.com/jobtrends>

## 3.4 ORM

---

### Roo의 JPA 지원

- Provider 선택 가능
  - Hibernate, OpenJpa, EclipseLink
- “Open EntityManager in View” pattern 지원
  - OpenEntityManagerInViewFilter
  - View에서 lazy loading 지원

출처 : <http://www.indeed.com/jobtrends>

## 3.5 확장

---

### 다양한 기술의 생성 도구로 활용 가능

- 기본 제공 Add-on
  - Spring Security, Spring webflow, JMS, SMTP등
- Add-on 으로 확장가능 구조
  - Custom add-on 개발 가능
- 향후 지원 예정
  - Spring Batch, Spring Integration
  - Spring Blaze DS & Flex
  - GWT
- OSGi bundle 생성 가능

# 소감

---

## 또 한 번 봄의 시작이 되길..

- Dynamic typing 언어 진영의 발전에 대한 Java 진영의 대답
  - Open Class vs AspectJ Mixin
- 더 높은 추상화 수준으로 나아가기
  - 객체지향
  - Domain Specific Language
  - 실세계 언어와 유사한 표현을 목표로
- 과거의 유산 포용하기
- 만드는 즐거움

# 참고자료 URL

---

- <http://forum.springsource.org/showthread.php?t=71985>
- <http://www.ksug.org/101>
- [http://jaoo.dk/aarhus-2009/file?path=/jaoo-aarhus-2009/slides/RodJohnson\\_ExtremeJavaProductivityWithSpringRooAndSpring30.pdf](http://jaoo.dk/aarhus-2009/file?path=/jaoo-aarhus-2009/slides/RodJohnson_ExtremeJavaProductivityWithSpringRooAndSpring30.pdf)