**CHAPTER1 스프링 개발 툴**

mavne이나 gradle(<https://gradle.org/releases/>) 을 이용하여 환경구성

기존 manve project에서 gradle로 변경 하려면 해당 명령어 사용하면 된다.

**gradle init --type pom**

**CHAPTER4 스프링 REST**

-REST는 HTTP를 기본으로 웹 서비스를 구현하는 아키텍처로 각광받고 있다.

**레시피 4-1 REST 서비스로 XML 발행하기**

-스프링 mvc에서 애플리케이션 데이터를 REST 서비스로 발행하는 주역은 @RequestMapping과 @PathVariable 두 애너테이션이다.

**MarshallingVeiw로 XML 만들기**

@RequestMapping("/members")  
public String getRestMembers(Model model) {  
 Members members = new Members();  
 members.addMembers(memberService.findAll());  
 model.addAttribute("members", members);  
 return "membertemplate";  
}

🡺host\_name/애플리케이션명/members URL로 접근 가능

@Bean  
public View membertemplate() {  
 return new MarshallingView(jaxb2Marshaller());  
}

🡺위의 뷰는 MarshallingView 형으로 정의 (마샬러를 이용해 렌더링 하는 범용 클래스)

-마샬링은 메모리에 있는 객체를 특정한 데이터 형식으로 변환하는 과정

@Bean  
public Marshaller jaxb2Marshaller() {  
 Jaxb2Marshaller marshaller = new Jaxb2Marshaller();  
 marshaller.setClassesToBeBound(Members.class, Member.class);  
 return marshaller;  
}

🡺classesToBeBound는 XML로 변환할 대상 클래스

@XmlRootElement  
public class Member {  
 private String name;  
 private String phone;  
 private String email;

}

🡺@XmlRootElement는 Jaxb2Marshaller가 클래스 필드를 자동으로 감지해서 XML 데이터로 변경

**콘텐트 협상**

-http://[호스트명]/[애플리케이션명]/memebrs 처럼 확장 없이 URL을 요청할 경우 MVC 뷰의 해석 로직에 달려 있다. 이런 경우 URL 확장자 또는 HTTP 헤더를 보고 뷰를 결정한다.

ex) Accept: application/xml

**@ResponseBody로 XML 만들기**

-여러 뷰로 보여주기 위해 뷰를 하나씩 추가하는건 너무 번거롭기 때문에 이런 경우 HttpMessageConverter를 이용하면 된다.

@RequestMapping("/members")  
@ResponseBody  
public Members getRestMembers(Model model) {  
 Members members = new Members();  
 members.addMembers(memberService.findAll());  
 model.addAttribute("members", members);  
 return members;  
}

🡺@ResponseBody를 사용해서 응답 본문으로 취급

**결 과**

<http://localhost:8080/members.xml>

<members>  
 <member>  
 <email>marten@deinum.biz</email>  
 <name>Marten Deinum</name>  
 <phone>00-31-1234567890</phone>  
 </member>  
 <member>  
 <email>john@doe.com</email>  
 <name>John Doe</name>  
 <phone>1-800-800-800</phone>  
 </member>  
 <member>  
 <email>jane@doe.com</email>  
 <name>Jane Doe</name>  
 <phone>1-801-802-803</phone>  
 </member>  
</members>

<http://localhost:8080/members>

marten@deinum.bizMarten Deinum00-31-1234567890john@doe.comJohn Doe1-800-800-800jane@doe.comJane Doe1-801-802-803

**@PathVariable로 결과 거르기**

@RequestMapping("/member/{memberid}")  
@ResponseBody  
public Member getMember(@PathVariable("memberid") long memberID) {  
 return memberService.find(memberID);  
}

🡺@PathVariable을 이용해서 매개변수 사용하기

@RequestMapping("/member/\*/{memberid}")

🡺와일드카드 사용할 경우 /member/심준보/1 이런 url 호출도 가능

**ResponseEntity로 클라이언트에게 알려주기**

@RequestMapping("/member/{memberid}")  
@ResponseBody  
public ResponseEntity<Member> getMember(@PathVariable("memberid") long memberID) {  
 Member member = memberService.find(memberID);  
 if (member != null) {  
 return new ResponseEntity<Member>(member, HttpStatus.*OK*);  
 }  
 return new ResponseEntity(HttpStatus.*NOT\_FOUND*);  
}

🡺ResponseEntity는 결과 본문을 HTTP 상태 코드와 함께 집어넣은 래퍼 클래스이다.

**레시피 4-2 REST 서비스로 JSON 발행하기**