

Overview of Spring 4.0

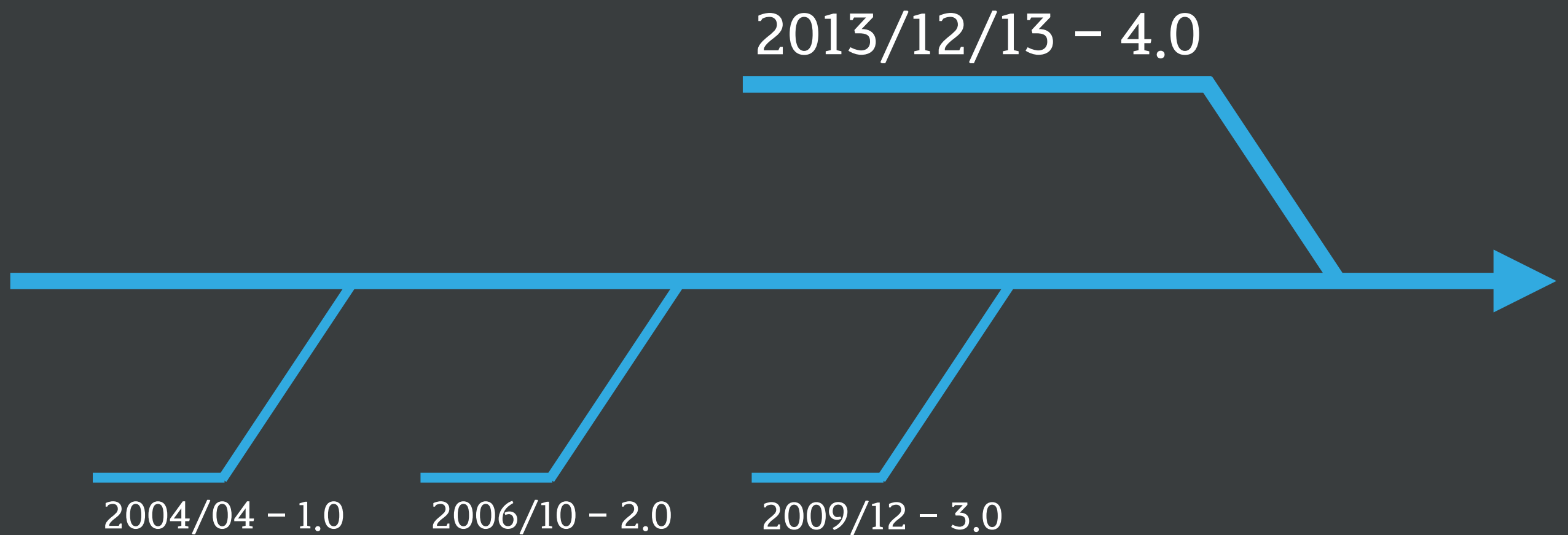
2 RVP100 465 3 RVP100



박용권 SK planet

- : 한국 스프링 사용자 모임(KSUG)
- : 봄싹(SpringSprout)
- : 라 스칼라 코딩단

1.0 배포, 그 후 10년



“스프링 4.0”

살펴보기

- ✓ Java 8 지원
- ✓ Java EE 6 및 7 지원
- ✓ Groovy DSL을 사용한 빈 정의
- ✓ 자바 웹소켓 API 지원
- ✓ REST Client 개발시 비동기 지원
- ✓ 그외 개선사항
- ✓ @Deprecated 클래스 및 메소드 삭제

Java 8

지원 기능

- ✓ lambda expressions
- ✓ method references
- ✓ JSR-310 Date and Time
- ✓ repeatable annotations
- ✓ parameter name discovery
 - : based on the `-parameters` compiler flag

동작 환경

- ✓ Java SE 6+ (JDK 6 update 10, ~2008)
- ✓ JDK 8 기반 애플리케이션은 4.0 이상 권장

lambda expressions & method references

콜백 인터페이스(callback interfaces)에 적용 가능

✓ JdbcTemplate

: PreparedStatementSetter

`void setValues(PreparedStatement ps) throws SQLException`

: RowMapper

`Object mapRow(ResultSet rs, int rowNum) throws SQLException`

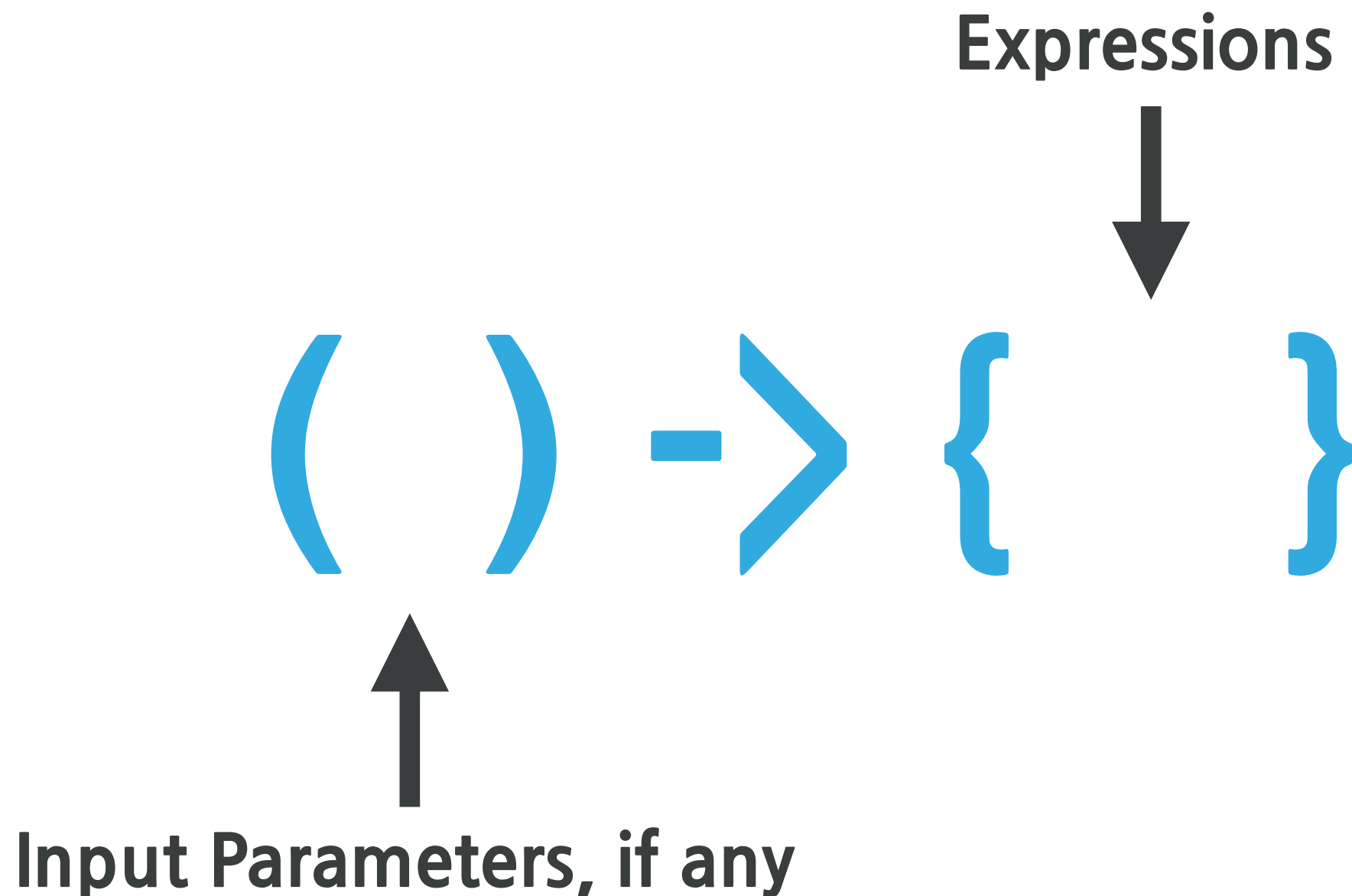
✓ JmsTemplate

: MessageCreator

`Message createMessage(Session session) throws JMSException`

✓ TransactionTemplate, TaskExecutor, etc

lambda expressions **syntax**



Lambdas with JdbcTemplate

```
JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);
```

```
List<Person> persons = jdbcTemplate.query(  
    "SELECT name, age FROM person where age = ?",  
    new PreparedStatementSetter() {  
        @Override  
        public void setValues(PreparedStatement ps) throws SQLException {  
            ps.setInt(1, 35);  
        }  
    },  
    new RowMapper<Person>() {  
        @Override  
        public Person mapRow(ResultSet rs, int rowNum) throws SQLException {  
            return new Person(rs.getString(1), rs.getInt(2));  
        }  
    }  
);
```


Lambdas with JdbcTemplate


```
JdbcTemplate jdbcTemplate = new JdbcTemplate(dataSource);
```

```
List<Person> persons = jdbcTemplate.query(  
    "SELECT name, age FROM person where age = ?",  
    (PreparedStatement ps) -> {  
        ps.setInt(1, 35);  
    },  
    (ResultSet rs, int rowNum) -> {  
        return new Person(rs.getString(1), rs.getInt(2));  
    }  
);
```

```
List<Person> persons = jdbcTemplate.query(  
    "SELECT name, age FROM person where age = ?",  
    ps -> ps.setInt(1, 35),  
    (rs, rowNum) -> new Person(rs.getString(1), rs.getInt(2))  
);
```

Method References with JdbcTemplate

```
public List<Person> findAll() {  
    return jdbcTemplate.query("SELECT name, age FROM person", this::mapPerson);  
}  
  
private Person mapPerson(ResultSet rs, int rowNum) throws SQLException {  
    return new Person(rs.getString("name"), rs.getInt("age"));  
}
```



JSR-310: Date and Time API

```
import org.springframework.format.annotation.DateTimeFormat;  
  
import java.time.*;  
  
public class JSR310Format {  
  
    @DateTimeFormat(iso = DateTimeFormat.ISO.DATE_TIME)  
    private LocalDateTime localDateTime;  
  
    @DateTimeFormat(pattern = "yyyy-MM-dd")  
    private LocalDate localDate;  
  
    @DateTimeFormat(pattern = "HH:mm:ss")  
    private LocalTime localTime;  
  
}
```

Repeatable Annotations

```
import org.springframework.context.annotation.PropertySource;
import org.springframework.context.annotation.PropertySources;

@PropertySources({
    @PropertySource("classpath:META-INF/properties/environment.xml"),
    @PropertySource("classpath:META-INF/properties/environment.properties")
})
class PropertySourceConfig {
    // ...
}

@PropertySource("classpath:META-INF/properties/environment.xml")
@PropertySource("classpath:META-INF/properties/environment.properties")
class PropertySourceConfig {
    // ...
}
```

Java EE 6 and 7

지원 기능

- ✓ JMS 2.0
- ✓ JTA 1.2
- ✓ JPA 2.1
- ✓ Bean Validation 1.1
- ✓ JSR-236 Concurrency Utilities

동작 환경

- ✓ Java EE 6+
- ✓ Servlet 2.5 도 호환되나, 3.0 이상 권장

“Groovy DSL” 빈 정의

Groovy

- ✓ JVM에서 동작하는 동적 언어
- ✓ 컴파일 하지 않고 실행하는 스크립트 언어
- ✓ Java + Python, Ruby, Smalltalk
- ✓ 간결한 문법, 강력한 기능 그리고 DSL 지원

Bean Definition DSL

- ✓ 간결한 문법으로 빈 정의
- ✓ XML NameSpace 기능 지원

Groovy DSL로 빈 정의하기

```
import org.springframework.web.servlet.view.InternalResourceViewResolver
import org.springframework.web.servlet.view.JstlView

beans {
    jstlViewResolver(InternalResourceViewResolver) {
        viewClass = JstlView
        prefix = "/WEB-INF/views"
        suffix = ".jsp"
    }
}
```

↑ Groovy DSL

↓ XML

```
<bean id="jstlViewResolver"
      class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
    <property name="prefix" value="/WEB-INF/views"/>
    <property name="suffix" value=".jsp"/>
</bean>
```

Groovy DSL 로 XML Namespace 사용하기

```
beans {  
    xmlns context: 'http://www.springframework.org/schema/context'  
    context.'component-scan'('base-package': 'jco.conference.oxquiz')  
}
```

↑ Groovy DSL
↓ XML

```
<beans xmlns="http://www.springframework.org/schema/beans"  
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xmlns:context="http://www.springframework.org/schema/context"  
        xsi:schemaLocation="...">  
    <context:component-scan base-package="jco.conference.oxquiz"/>  
</beans>
```


Groovy DSL 로 Spring MVC 설정

```
import org.springframework.web.servlet.view.InternalResourceViewResolver
import org.springframework.web.servlet.view.JstlView

beans {
    xmlns context: 'http://www.springframework.org/schema/context'
    xmlns mvc: 'http://www.springframework.org/schema/mvc'

    context.'component-scan'('base-package': 'jco.conference.oxquiz')

    mvc.'annotation-driven'()
    mvc.'default-servlet-handler'()

    mvc.'resources'('mapping': '/resources/**', 'location': '/resources/')

    jstlViewResolver(InternalResourceViewResolver) {
        viewClass = JstlView
        prefix = '/WEB-INF/views'
        suffix = '.jsp'
    }
}
```

WebSocket and Messaging


spring-websocket module

- ✓ JSR 356: Java 웹소켓 API 지원
- ✓ SockJS로 웹소켓 미지원 브라우저 대응

spring-messaging module

- ✓ 메시지, 채널 등의 추상화
- ✓ 고수준 메시지 프로토콜 STOMP 지원

JSR 356: Java WebSocket API

- 
- ✓ HTML5 표준
 - ✓ API: W3C / 프로토콜: IETF
 - ✓ 웹 소켓 프로토콜 사용
 - ✓ 양방향 통신

JSR 356: Java WebSocket API



- ✓ Tomcat 7.0.47+ and 8.0
- ✓ Jetty 9.0 and 9.1
- ✓ WildFly 8.0 (JBoss Application Server)
- ✓ GlassFish 4.0

Can I use WebSockets?

- ✓ Chrome 14.0+ (Current 32.0)
- ✓ Safari 6.0+ (Current 7.0)
- ✓ Firefox 11.0+ (Current 27.0)
- ✓ Internet Explorer 10.0+ (Current 11.0)
- ✓ iOS Safari 6.0+ (Current 7.0)
- ✓ Android Browser 4.4

SockJS 는...

JavaScript로
브라우저 종류에 상관없이
실시간 웹 구현 기술

지원 기능

- ✓ WebSocket
- ✓ AJAX long polling
- ✓ AJAX multipart streaming
- ✓ Forever Iframe
- ✓ JSONP Polling

SockJS family

- ✓ SockJS-client JavaScript client library
- ✓ SockJS-node Node.js server
- ✓ SockJS-erlang Erlang server
- ✓ SockJS-tornado Python server
- ✓ SockJS-twisted Python server
- ✓ vert.x Java/vert.x server
- ✓ SockJS-cyclone Python server

WebSocketHandler 인터페이스 구현

```
public class EchoHandler implements WebSocketHandler {  
  
    @Override  
    public void handleMessage(WebSocketSession session  
                             , WebSocketMessage<?> message) throws Exception {  
  
        sessions.forEach(session -> {  
            try { session.sendMessage(message); }  
            catch (IOException ignore) { }  
        });  
  
    }  
  
    // 생략  
}
```

WebSocket 활성화 및 WebSocketHandler 등록

```
@Configuration
@EnableWebSocket
public class ExampleWebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(echoHandler(), "/websocket/echo");
    }

    @Bean
    public EchoHandler echoHandler() {
        return new EchoHandler();
    }
}
```


SockJS WebSocketHandler 등록하기

```
@Configuration
@EnableWebSocket
public class ExampleWebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(echoHandler(), "/websocket/echo/sockjs").withSockJS();
    }

    @Bean
    public EchoHandler echoHandler() {
        return new EchoHandler();
    }
}
```

STOMP Message 처리 구현

```
import org.springframework.stereotype.Controller;  
import org.springframework.messaging.handler.annotation.MessageMapping;  
import org.springframework.messaging.Message;
```

```
@Controller
```

```
public class EchoController {
```

```
    @Autowired
```

```
    private SimpMessageSendingOperations messagingTemplate;
```

```
    @MessageMapping("/message")
```

```
    public void echo(Message<String> message) {  
        messagingTemplate.send("/topic/echo", message);
```

```
    }
```

```
}
```

STOMP 활성화 및 설정

```
@Configuration
@EnableWebSocketMessageBroker
public class ExampleWebSocketConfig implements WebSocketMessageBrokerConfigurer {

    @Override
    public void registerStompEndpoints(StompEndpointRegistry registry) {
        registry.addEndpoint("/websocket/endpoint/stomp");
    }

    @Override
    public void configureMessageBroker(MessageBrokerRegistry registry) {
        registry.setApplicationDestinationPrefixes("/stomp");
        registry.enableSimpleBroker("/topic/");
    }

    @Bean
    public EchoController echoController() {
        return new EchoController();
    }
}
```

REST 클라이언트 개발시 비동기 지원

AsyncRestTemplate

```
final String GITHUB_API_USERS_REPOS = "https://api.github.com/users/{user}/repos";

ListenableFuture<ResponseEntity<Repository[]>> reposForEntity =
    asyncRestTemplate.getForEntity(GITHUB_API_USERS_REPOS, Repository[].class, uriVariables);

reposForEntity.addCallback(new ListenableFutureCallback<ResponseEntity<Repository[]>>() {
    @Override
    public void onSuccess(ResponseEntity<Repository[]> entity) {
        // 응답 데이터 처리
    }
    @Override
    public void onFailure(Throwable throwable) {
        // 예외 처리
    }
});
```

그외 개선사항

Core Container

- ✓ Generics 기반 의존성 주입
- ✓ 메타 애노테이션 속성 재정의
- ✓ 조건부 빈 정의
- ✓ 기타

Web

- ✓ @RestController 추가
- ✓ Spring MVC: TimeZone 사용 지원

Testing

- ✓ 테스트용 메타 애노테이션 정의
- ✓ mock.web.* 이하 Servlet 3.0 지원

Generics 기반 의존성 주입

```
public interface Repository<T> {  
    ...  
}
```

```
public interface PlayerRepository extends Repository<Player> {  
    ...  
}
```

```
@Controller  
public class PlayerController {  
    @Autowired  
    private Repository<Player> playerRepository;  
}
```

메타 애노테이션 속성 재정의

```
@Transactional("indexDataSource")
@Retention(RetentionPolicy.RUNTIME)
public @interface IndexDataSourceTransactional {
    boolean readOnly() default false;
}

@Transactional("metaDataSource")
@Retention(RetentionPolicy.RUNTIME)
public @interface MetaDataSourceTransactional {
    boolean readOnly() default false;
}

@Service
public class ExampleServiceImpl implements ExampleService {

    @IndexDataSourceTransactional(readOnly = true)
    public IndexEntity findIndex() { ... }

    @MetaDataSourceTransactional(readOnly = true)
    public MetaEntity findMeta() { ... }
}
```

조건부 빈 정의

- ✓ 조건에 따라 빈 정의를 전략적으로 선택하는 일반적인 방법 추가
 - : Spring 3.1 @Profile 보다 더욱 유연하고 동적으로 지원
 - : Spring Boot 프로젝트에서 적극적으로 사용
- ✓ @Conditional을 통한 조건부 처리 구현
 - : 다양한 조건에 따라 대응 가능(시스템 OS, 자바 버전 등...)
 - : @Profile도 ProfileCondition을 통해 동작되도록 구현
 - : 사용자 정의 조건부 빈 정의 지원

Core Container 개선사항: 기타

- ✓ 기본 생성자 없이 CGLIB 프록시 생성
: 오픈소스 objenesis를 사용한 프록시 대상 객체 생성
- ✓ 순서를 보장하는 의존성 주입
: Ordered 인터페이스 또는 @Ordered 에 따른 정렬 후 의존성 주입
- ✓ 의존성 주입시 @Lazy 지원
- ✓ @Description: 빈 정의시 주석 작성

테스트용 메타 애노테이션 정의

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration({"app-config.xml", "test-data-access-config.xml"})
@ActiveProfiles("dev")
@Transactional
public class OrderServiceTests {
    ...
}
```

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration({"app-config.xml", "test-data-access-config.xml"})
@ActiveProfiles("dev")
@Transactional
public class UserServiceTests {
    ...
}
```

테스트용 메타 애노테이션 정의

```
@RunWith(SpringJUnit4ClassRunner.class)
@TransactionalDevTest
public class OrderServiceTests {
    ...
}
```

```
@RunWith(SpringJUnit4ClassRunner.class)
@TransactionalDevTest
public class UserServiceTests {
    ...
}
```

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@ContextConfiguration({"app-config.xml", "test-data-access-config.xml"})
@ActiveProfiles("dev")
@Transactional
public @interface TransactionalDevTest { }
```

@Deprecated 클래스 및 메소드 삭제

✓ @Deprecated 클래스 및 메소드 모두 삭제

: API Differences Report에서 확인 가능

✓ Third-Party 라이브러리 호환

: 버전 4.0 기준 2010/2011년도 이후 버전을 사용

: Hibernate 3.6+, EhCache 2.1+, Quartz 1.8+, Groovy 1.8+, Joda-Time 2.0+

: 예외적으로 Hibernate Validator는 4.3+ 버전이 필요

: Jackson 1.8/1.9는 @Deprecated 선언, 2.0+ 이상을 지원하는데 집중

궁금하세요?

Special Thanks to

"Designed for Sails.js, an MVC framework for Node by Balderdash"

한국 스프링 사용자 그룹 (KSUG)

<http://www.ksug.org/>

<http://groups.google.com/group/ksug>

<https://www.facebook.com/groups/springkorea/>

고맙습니다!

@arawnkr

arawn.kr@gmail.com