Spring Framework 4 on Java 8

Juergen Hoeller
Spring Framework Lead
Pivotal

The State of the Art: Component Classes

```
@Service
@Lazy
public class MyBookAdminService implements BookAdminService {

    @Autowired
    public MyBookAdminService(AccountRepository repo) {
        ...
    }

    @Transactional
    public BookUpdate updateBook(Addendum addendum) {
        ...
    }
}
```

The State of the Art: Configuration Classes

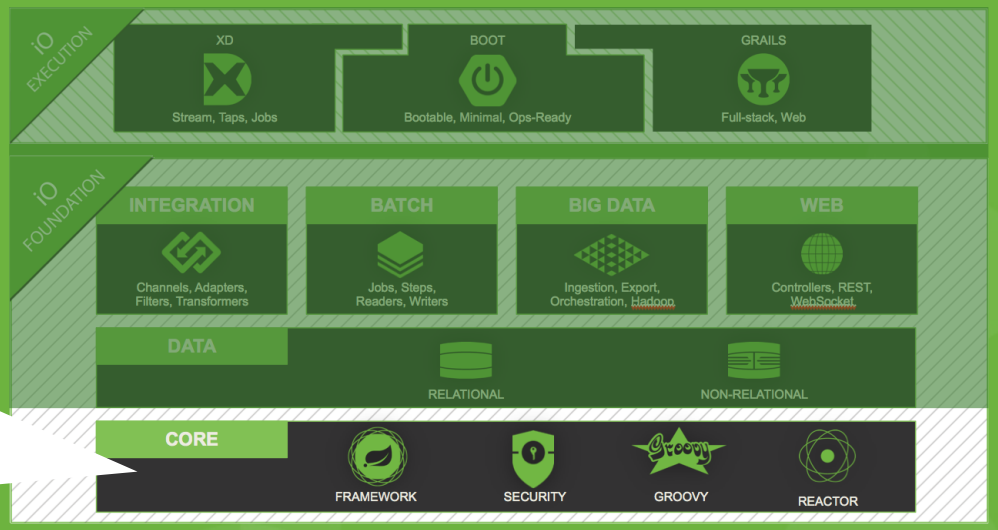
```
@Configuration
@Profile("standalone")
@EnableTransactionManagement
public class MyBookAdminConfig {

    @Bean
    @Scope("session")
    public BookAdminService myBookAdminService() {
        MyBookAdminService service = new MyBookAdminService();
        service.setDataSource(bookAdminDataSource());
        return service;
    }

    ...
}
```

SPRING IO CORE:

Introducing Spring Framework 4.0



Introducing Spring Framework 4.0

- **Ready for new application architectures**
  - embedded web servers and non-traditional datastores
  - lightweight messaging and WebSocket-style architectures
  - custom asynchronous processing with convenient APIs

- **A new baseline**
  - Java SE 6+ (minimum API level: JDK 6 update 18, ~ early 2010)
  - Java EE 6+ (Servlet 3.0 focused, Servlet 2.5 compatible at runtime)
  - comprehensive support for Java SE 8 (language features and APIs)
  - explicit support for Java EE 7 level specifications
    (JMS 2.0, JTA 1.2, JPA 2.1, Bean Validation 1.1, JSR-236 Concurrency)

spring by Pivotal

Generics-based Injection Matching

```
@Service
public class MyBookAdminService implements BookAdminService {

    @Autowired
    public MyBookAdminService(MyRepository<Account> repo) {
        ...
    }
}


@Bean
public MyRepository<Account> myAccountRepository() {
    return new MyAccountRepositoryImpl();
}
```

Many Further Container Refinements

- **Composable annotations with overridable attributes**
  - e.g. custom scope annotation with proxyMode attribute
- **A generalized model for conditional bean definitions**
  - based on @Conditional; see Spring Boot (projects.spring.io/spring-boot)
- **@Autowired @Lazy on injection points**
  - requesting a lazy-initialization proxy individually per injection point
- **Target-class proxies for classes with arbitrary constructors**
  - creating CGLIB proxies using Objenesis, not invoking any constructor

spring by Pivotal

Messaging & WebSocket

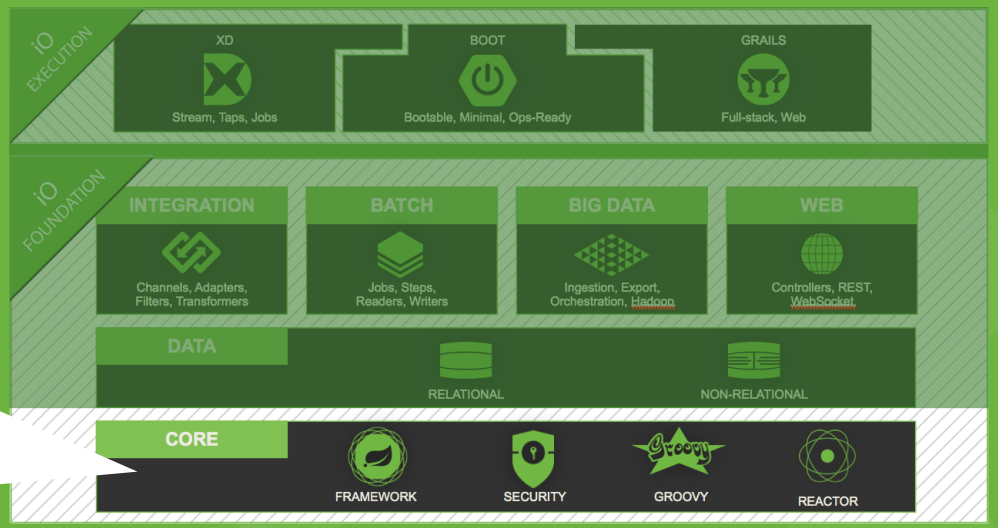
- **General org.springframework.messaging module**
  - core message and channel abstractions
  - endpoints using generic messaging patterns
  - @MessageMapping and co for annotated endpoints

- **WebSocket endpoint model along the lines of Spring MVC**
  - JSR-356 support for raw WebSocket handling
  - flexible endpoints through native server support (Tomcat 7/8, Jetty 9, GlassFish 4, WildFly 8)
  - transparent SockJS fallback option
  - STOMP for higher-level messaging on top of a WebSocket channel

spring by Pivotal

SPRING IO CORE:

Spring Framework 4 and Java 8



Spring Framework 4 and Java 8

- **First-class support for Java 8 language and API features**
  - lambda expressions
  - method references
  - JSR-310 Date and Time
  - repeatable annotations
  - parameter name discovery

- **Full runtime compatibility with JDK 8**
  - for Spring apps built against JDK 6/7 but running against JDK 8
  - when moving existing apps to a JDK 8 based deployment platform

Lambda Conventions in Spring APIs

- 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
  - **TransactionCallback:**
    Object doInTransaction(TransactionStatus status)

spring by Pivotal

Lambdas with Spring's JdbcTemplate (v1)

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

jt.query("SELECT name, age FROM person WHERE dep = ?",
    ps -> ps.setString(1, "Sales"),
    (rs, rowNum) -> new Person(rs.getString(1), rs.getInt(2)));
```

Lambdas with Spring's JdbcTemplate (v2)

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

jt.query("SELECT name, age FROM person WHERE dep = ?",
    ps -> {
        ps.setString(1, "Sales");
    },
    (rs, rowNum) -> {
        return new Person(rs.getString(1), rs.getInt(2));
    });
```

Method References with Spring's JdbcTemplate

```
public List<Person> getPersonList(String department) {
    JdbcTemplate jt = new JdbcTemplate(this.dataSource);
    return jt.query("SELECT name, age FROM person WHERE dep = ?",
        ps -> ps.setString(1, "Sales"),
        this::mapPerson);
}

private Person mapPerson(ResultSet rs, int rowNum)
        throws SQLException {
    return new Person(rs.getString(1), rs.getInt(2));
}
```

JSR-310 Date and Time

```
import java.time.*;
import org.springframework.format.annotation.*;

public class Customer {

    // @DateTimeFormat(iso=ISO.DATE)
    private LocalDate birthDate;

    @DateTimeFormat(pattern="M/d/yy h:mm")
    private LocalDateTime lastContact;

    ...
}
```

Repeatable Annotations

```
@Scheduled(cron = "0 0 12 * * ?")
@Scheduled(cron = "0 0 18 * * ?")
public void performTempFileCleanup() {
    ...
}

@Schedules({
    @Scheduled(cron = "0 0 12 * * ?"),
    @Scheduled(cron = "0 0 18 * * ?")
})
public void performTempFileCleanup() {
    ...
}
```

Parameter Name Discovery

- **Spring's DefaultParameterNameDiscoverer**
  - as of Spring Framework 4.0: aware of Java 8's parameter reflection
  - now checking Java 8 first (-parameters)
  - ASM-based reading of debug symbols next (-debug)

```
@Controller
public class MyMvcController {

    @RequestMapping(value="/books/{id}", method=GET)
    public Book findBook(@PathVariable long id) {
        return this.bookAdminService.findBook(id);
    }
}
```

SPRING IO CORE:

Outlook: Spring Framework 4.1

Key Themes for Spring Framework 4.1

- **Comprehensive web resource handling**
  - cache control refinements, pluggable resource handler strategies
- **Caching support revisited**
  - alignment with JCache 1.0 annotations, user-requested enhancements
- **JMS support overhaul**
  - alignment with messaging module, annotation-driven endpoints
- **Spring Framework 4.1 GA scheduled for July 2014**

spring by Pivotal

Learn More. Stay Connected.

- **Core framework:**
  projects.spring.io/spring-framework
- **Check out Spring Boot:**
  projects.spring.io/spring-boot
- **Current and upcoming releases:**
  Spring Framework 4.0.5 on May 20th
  Spring Framework 4.1 RC1 on July 1st

Twitter: twitter.com/springcentral

YouTube: spring.io/video

LinkedIn: spring.io/linkedin

Google Plus: spring.io/gplus

spring by Pivotal