Spring

Petr Křemen, Martin Ledvinka

KBSS FEL ČVUT

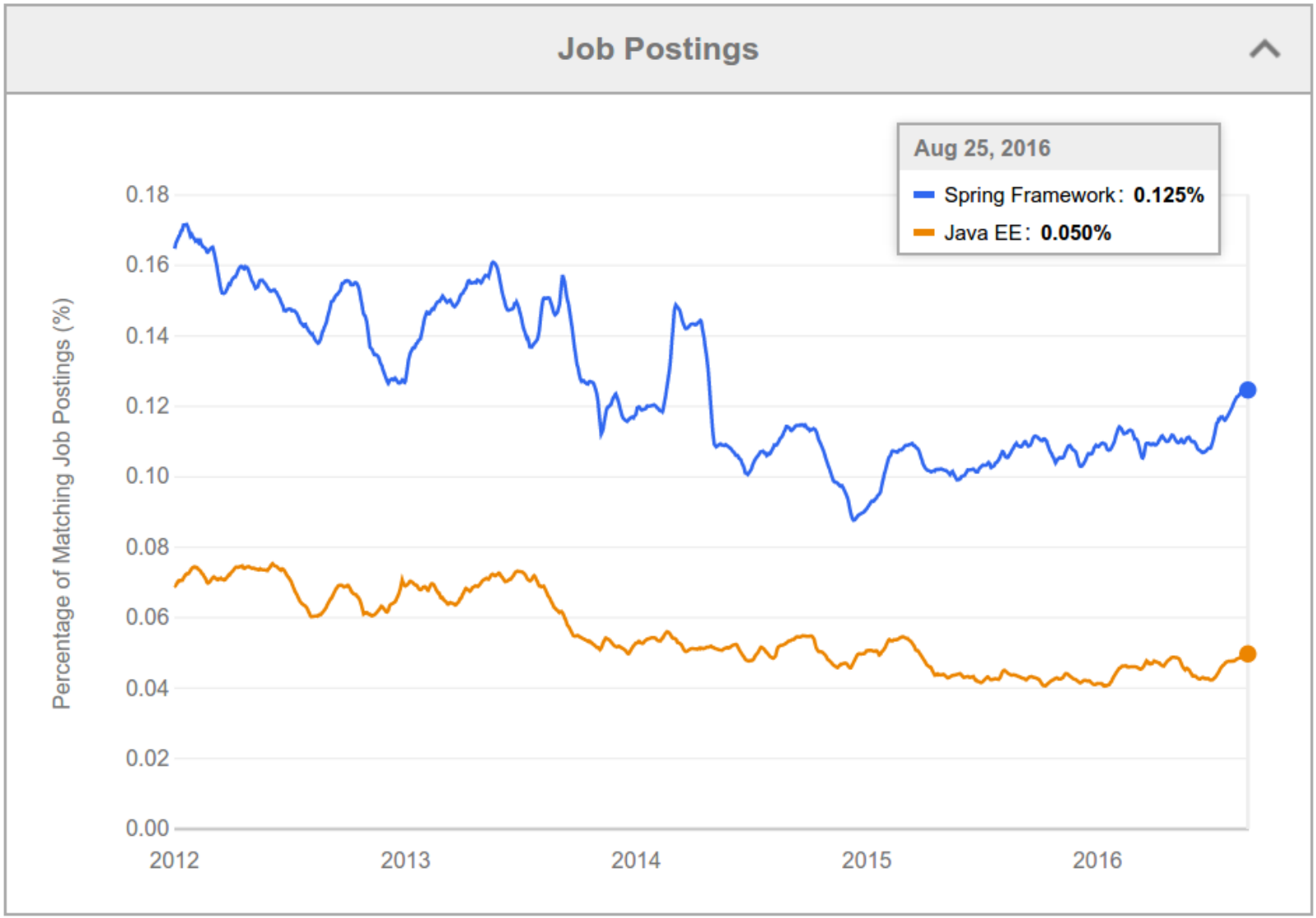
Winter Term 2019

Contents

Business Logic

Spring and Java EE

Job Trends by `indeed.com`



Spring Framework Highlights

pros

- Dependency Injection
- Convention over Configuration
- Many Components for desktop/web/enterprise application development
  - Modular, i.e., individual Spring components can be used and combined with other frameworks
- Open-Source, POJO-Based

cons

- Not part of the Java EE stack

Examples

Examples from this lecture can be found at `https://gitlab.fel.cvut.cz/ear/spring-example`.

Spring and EJB

- Both technologies provide enterprise container with DI, IoC, transactions and other relevant features
- EJB is a part of Java EE stack, it is a standard, supporting high-availability, clustering
- Spring is a feature-rich alternative to EJB with many extensions cf. EJB, e.g. `@Configurable`
- A comparison is at `https://zeroturnaround.com/rebellabs/spring-and-java-ee-head-to-head`

Dependency Injection

Dependency Injection Reminder

```
package cz.cvut.kbss.ear.spring_example;
import ...

public class SchoolInformationSystem {

  private CourseRepository repository
    = new InMemoryCourseRepository();

  public static void main(String[] args) {
    SchoolInformationSystem main = new SchoolInformationSystem();
    System.out.println(main.repository.getName());
  }
}
```

The client code (`SchoolInformationSystem`) itself decides which repository implementation to use

- change in **implementation** requires *client code* change.
- change in **configuration** requires *client code* change.

DI using XML

SchoolInformationSystem.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

public class SchoolInformationSystem {
  private CourseRepository repository;
}
```

InMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

public class InMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return
    "In-memory course repository"; }
}
```

CourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
public interface CourseRepository {
  public String getName() { return name; }
}
```

application-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans ...>
  <bean id="SchoolInformationSystem"
        class="cz.cvut.kbss.ear.spring_example.SchoolInformationSystem"
        scope="singleton">
     <property name="repository" ref="CourseRepository"/>
  </bean>
  <bean id="CourseRepository"
        class="cz.cvut.kbss.ear.spring_example.InMemoryCourseRepository">
  </bean>
</beans>
```

DI using Annotations

SchoolInformationSystem.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
public class SchoolInformationSystem {
  @Autowired
  private CourseRepository repository;
}
```

CourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
public interface CourseRepository {
  public String getName() { return name; }
}
```

InMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
public class InMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return
    "In-memory course repository"; }
}
```

Dependency Injection (DI) and Inversion of Control (IoC)

Dependency Injection

Component lifecycle is controlled by the *container* which is responsible for delivering correct implementation of the given bean.

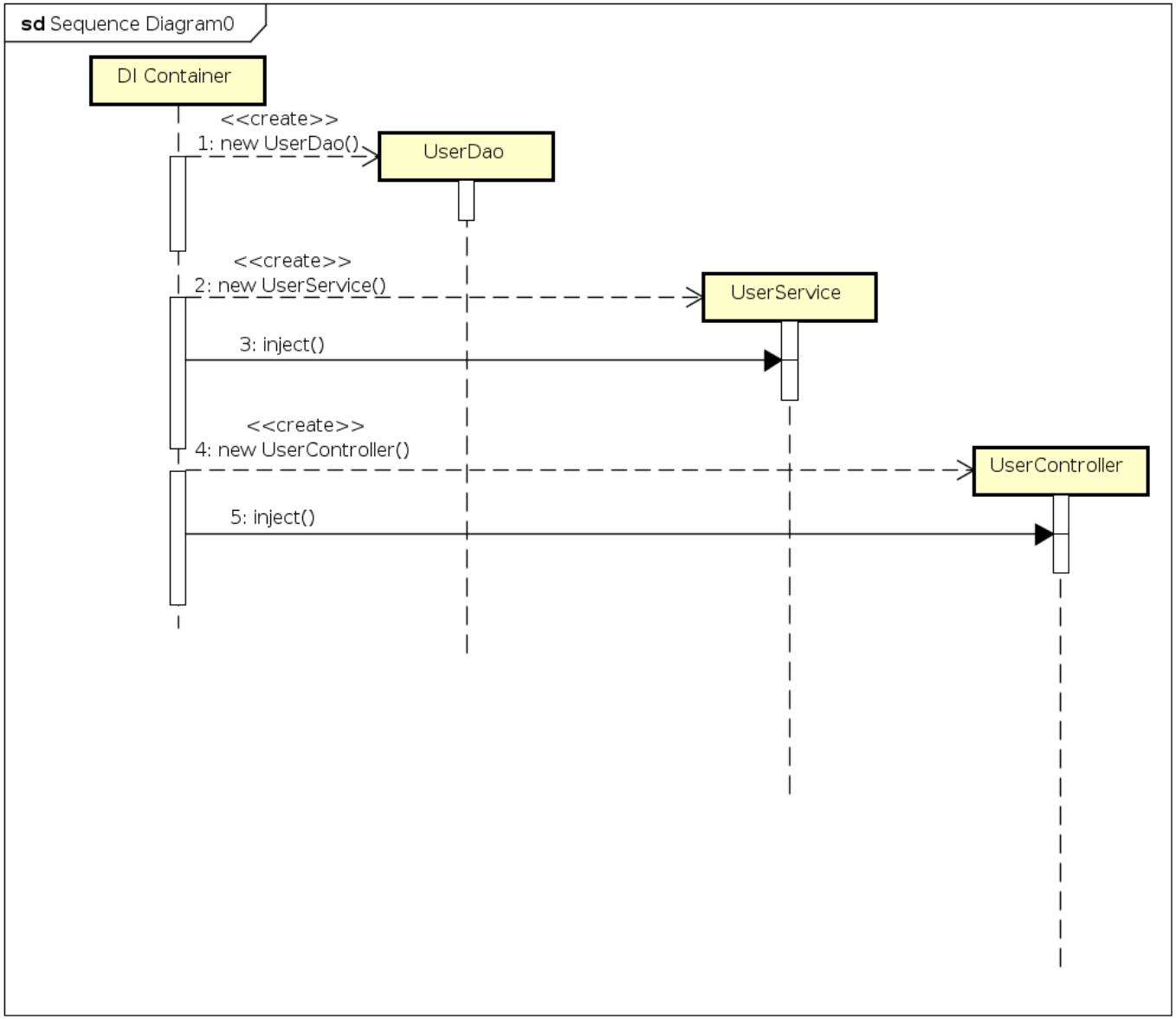
Inversion of Control

The programmed application is a "library" for the generic framework that controls the application lifecycle.

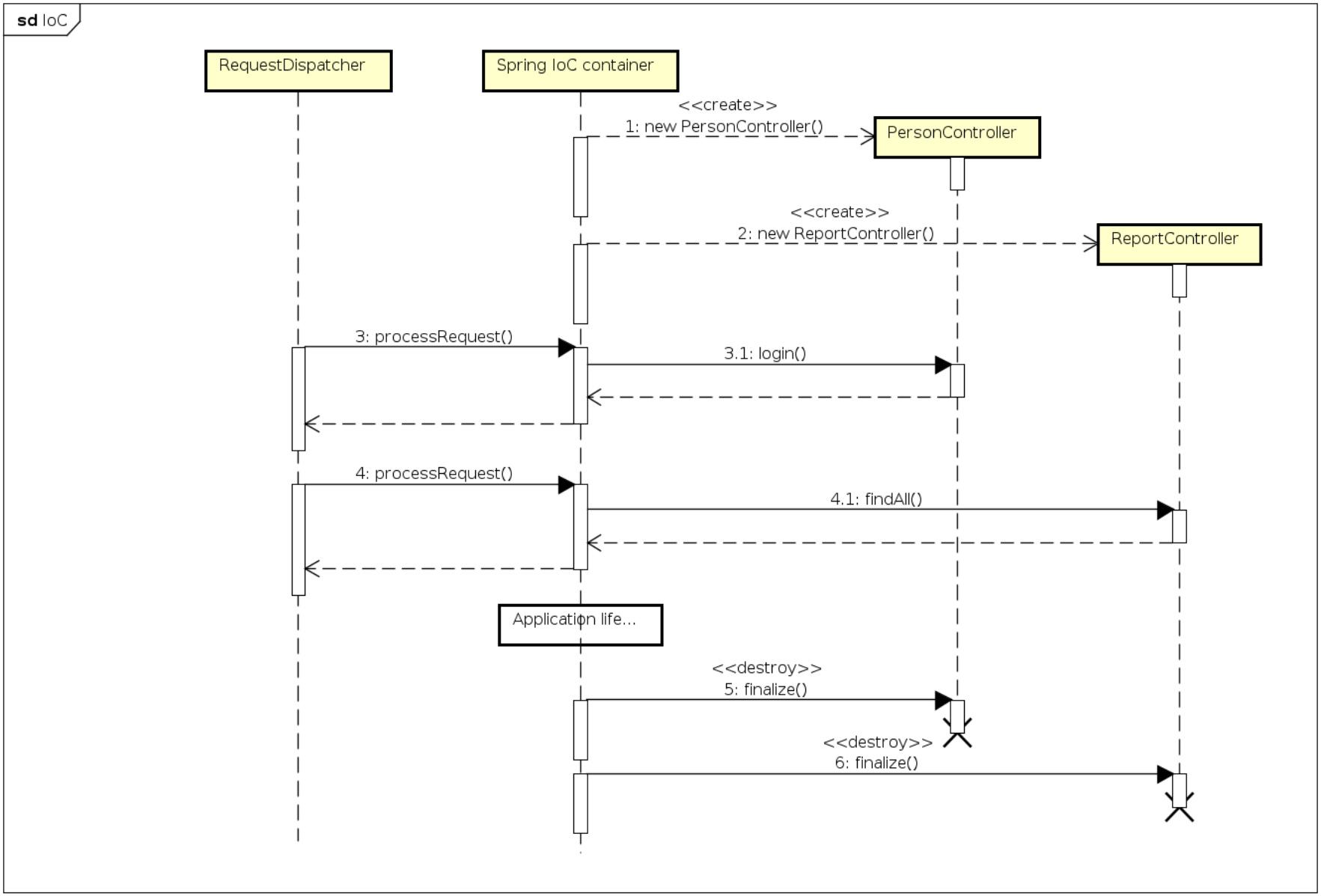
Hollywood Principle

Don't call us, we'll call you.

Dependency Injection



Inversion of Control



Related Dependency Technologies

Dependency Injection for Java (JSR 330)

- Dependency mechanism
- (partially) implemented in Spring
- $\in$ Java EE Web Profile

Context Dependency Injection (CDI) (JSR 299)

- Definition of bean scopes
- Not implemented in Spring
- $\in$ Java EE Web Profile

DI with JSR 330 annotations and bean disambiguation

JSR 330: Dependency Injection for Java

is a part of Java EE Web Profile. Spring supports JSR 330 annotations.

SchoolInformationSystem.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Named
public class SchoolInformationSystem {
  @Inject
  private CourseRepository repository;

  ...
}
```

CourseRepository.java

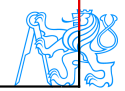
```
package cz.cvut.kbss.ear.spring_example;
public interface CourseRepository {
  public String getName() { return name; }
}
```

InMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Named
public class InMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return "In-memory
    course repository"; }
}
```

AnotherInMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Named("repository")
public class AnotherInMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return "Another
    In-memory course repository"; }
}
```



Spring Bean Scopes

singleton	a single bean instance per Spring IoC container
prototype	a new bean instance each time when requested
request	a single bean instance per HTTP request
session	a single bean instance per HTTP session
globalSession	a single bean instance per global HTTP session

global HTTP session

A session shared accross multiple portlets in a portlet application.

Spring allows custom scope definition (e.g. JSF 2 Flash scope)

Spring Bean Scopes – Prototype

SchoolInformationSystem.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
@Scope("singleton")
public class SchoolInformationSystem {
  @Autowired
  private CourseRepository repository;

  @Autowired
  private CourseRepository
    secondRepository;
  ...

  public static void main(String[] args) {
    ...
    // injected SchoolInformationSystem s;
    System.out.println(
      s.repository == s.secondRepository
    );
  }
}
```

prints “false”

CourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
public interface CourseRepository {
  public String getName() { return name; }
}
```

AnotherInMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component("repository")
@Scope("prototype")
public class AnotherInMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return "Another
    In-memory course repository"; }
}
```

Spring Bean Scopes – Singleton

SchoolInformationSystem.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
@Scope("singleton")
public class SchoolInformationSystem {
  @Autowired
  private CourseRepository repository;

  @Autowired
  private CourseRepository
    secondRepository;
  ...

  public static void main(String[] args) {
    ...
    // injected SchoolInformationSystem s;
    System.out.println(
      s.repository == s.secondRepository
    );
  }
}
```

prints “true”

CourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
public interface CourseRepository {
  public String getName() { return name; }
}
```

AnotherInMemoryCourseRepository.java

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component("repository")
@Scope("singleton")
public class AnotherInMemoryCourseRepository
  implements CourseRepository {
  public String getName() { return "Another
    In-memory course repository"; }
}
```

Dependency Injection Mechanisms

Constructor injection

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
public class SchoolInformationSystem {

  private CourseRepository repository;

  @Autowired
  public
    SchoolInformationSystem(CourseRepository
    repository) {
   this.repository = repository;
  }
}
```

Setter injection

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
public class SchoolInformationSystem {

  private CourseRepository repository;

  @Autowired
  public void setRepository(CourseRepository
    repository) {
   this.repository = repository;
  }
}
```

Field injection

```
package cz.cvut.kbss.ear.spring_example;
import ...

@Component
public class SchoolInformationSystem {

  @Autowired
  private CourseRepository repository;
}
```

Dependency management for non-Spring objects

- Sometimes Spring cannot manage bean lifecycle, but needs to inject into it
  - Objects of other frameworks need not be ready for being managed by Spring
  - JPA entities – based on OO paradigm, objects should encapsulate both state and operations
- Annotation `@Configurable` denotes classes, objects of which are not managed by Spring, yet can inject Spring-managed objects
  - Byte-code instrumentation (aspect weaving)
    - Load-time weaving (java agent)
    - Compile-time weaving (aspect compiler)
  - Same mechanism used for declarative transactions (see later)

@Configurable – Example

```
@Configurable
@Entity
public class User {

  @Column(length=40, nullable=false)
  private String password;

  @Column(length=40, nullable=false)
  private String salt;

  @Autowired
  private transient HashProvider provider;
  ...
  public void setPassword(String password) {
    this.password = provider.computeHash(
    password + salt + "/* long string */");
  }
}
```

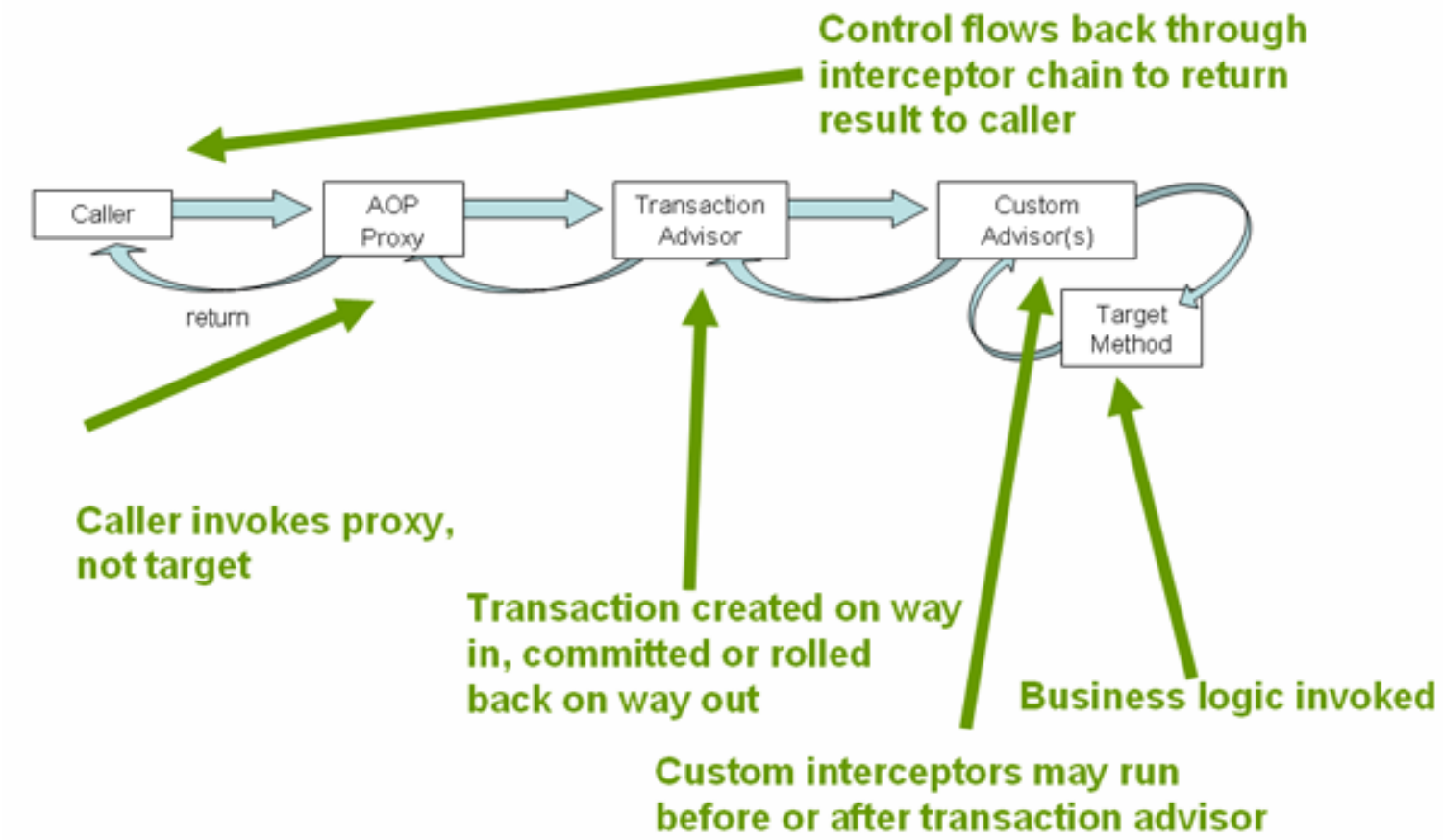
Spring Container Features

Declarative Transactions

```
@Component
public class UserService {

   @Autowired
   private UserDao userDao;

   @Transactional(readOnly=true)
   public List<UserDTO> findAll() {
      // implementation
   }

   @Transactional
   public UserDTO persist(UserDTO user, String password) {
      // implementation
   }

   @Transactional(readOnly=true)
   public UserDTO findByUsername(String name) {
      // implementation
   }

   // Other methods
}
```

- Transactions configurable through XML/annotations
- Global/local transactions
- Wraps multiple transaction APIs – JDBC, JTA, JPA, ...

Transaction Flow



Source:

`docs.spring.io/spring-framework/docs/4.2.x/spring-framework-reference/html/transaction.html`

Transaction Propagation

We can control, whether and how the transactional execution of the method should be supported

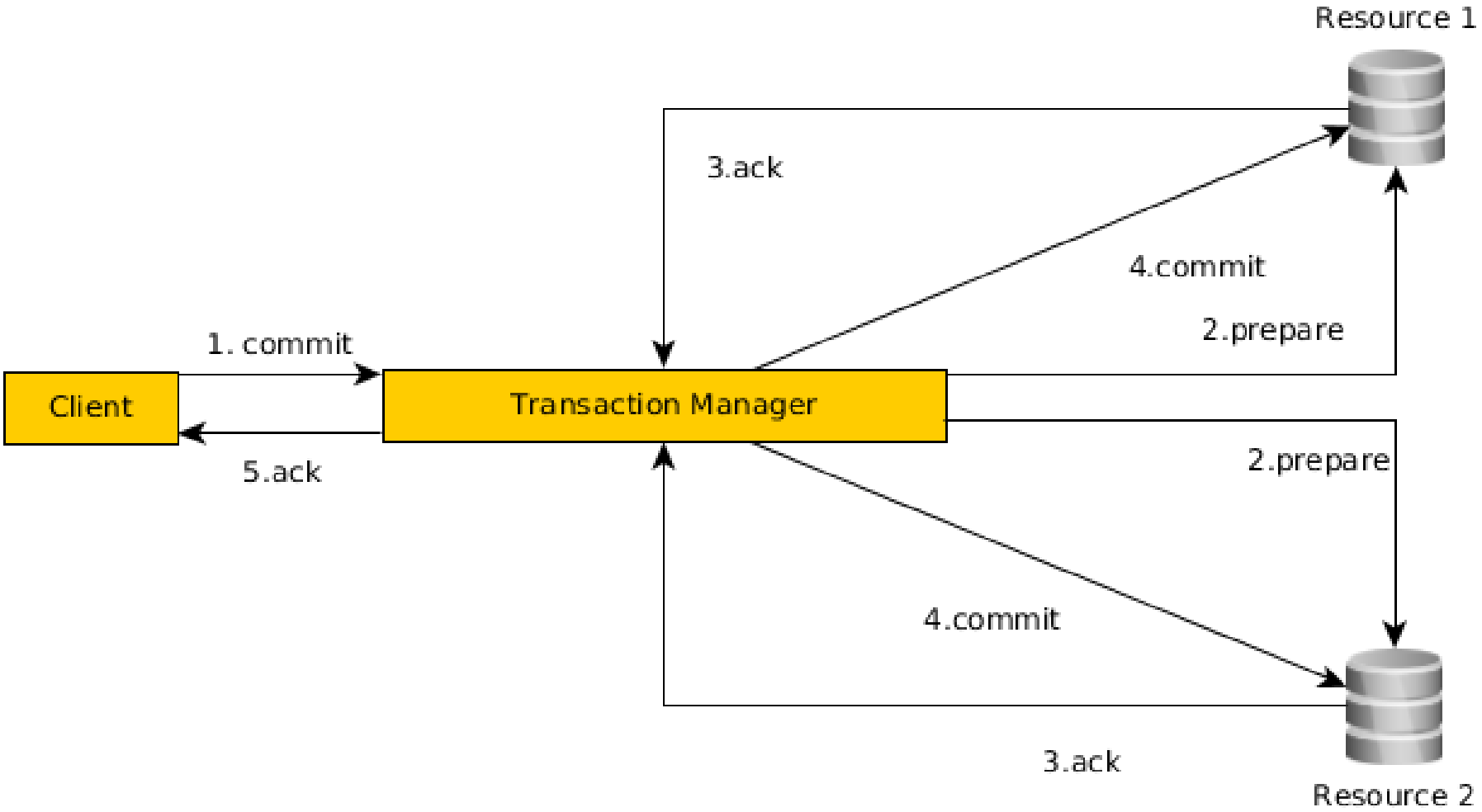
`@Transactional(propagation=...`

- `MANDATORY`
- `NESTED`
- `NEVER`
- `NOT_SUPPORTED`
- `REQUIRED` – default
- `REQUIRES_NEW`
- `SUPPORTS`

Other Transaction Attributes

- `isolation` – transaction isolation level
- `rollbackFor` (and other similar) – which exception hierarchies cause rollback (`RuntimeException` and `Error` by default)
- `readOnly` – true/false – readonly transactions can be optimized in runtime
- `timeout`
- `transactionManager`

Distributed Transactions



Spring and Persistence

1. Use standard JPA configuration through `persistence.xml` and load it by Spring
   - Reuse of existing configuration
   - Two XML configuration types
2. Configure JPA using Spring
   - One type of XML configuration/annotations
   - One more dependency on Spring...

JPA Configuration

```
@Configuration
@PropertySources({@PropertySource("classpath:jpa.properties"),
  @PropertySource("classpath:jdbc.properties")})
@ComponentScan(basePackages = "cz.cvut.kbss.ear.eshop.dao")
public class PersistenceConfig {
    @Autowired
    private final Environment environment;

    @Bean
    public DataSource dataSource() {
        final BoneCPDataSource ds = new BoneCPDataSource();
        ds.setDriverClass(environment.getRequiredProperty("jdbc.driverClassName"));
        ds.setJdbcUrl(environment.getRequiredProperty("jdbc.url"));
        ds.setUsername(environment.getRequiredProperty("jdbc.username"));
        ds.setPassword(environment.getRequiredProperty("jdbc.password"));
        return ds;
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource ds) {
        final LocalContainerEntityManagerFactoryBean emf = new
          LocalContainerEntityManagerFactoryBean();
        emf.setDataSource(ds);
        emf.setJpaVendorAdapter(new EclipseLinkJpaVendorAdapter());
        emf.setPackagesToScan("cz.cvut.kbss.ear.eshop.model");

        final Properties props = new Properties();
        props.setProperty("databasePlatform", environment.getRequiredProperty("jpa.platform"));
        emf.setJpaProperties(props);
        return emf;
    }
```

Security

```
@Transactional
public class UserService {

    @Autowired
    private UserDao dao;

    @Secured("ROLE_ADMIN")
    public void persist(UserDto user, String password, Boolean isAdmin) {
        // Implementation
    }

    @Secured("ROLE_ADMIN")
    public void removeById(Long id) {
        // Implementation
    }
}
```

- Method access control using annotations
- More on this in weeks 7 and 10

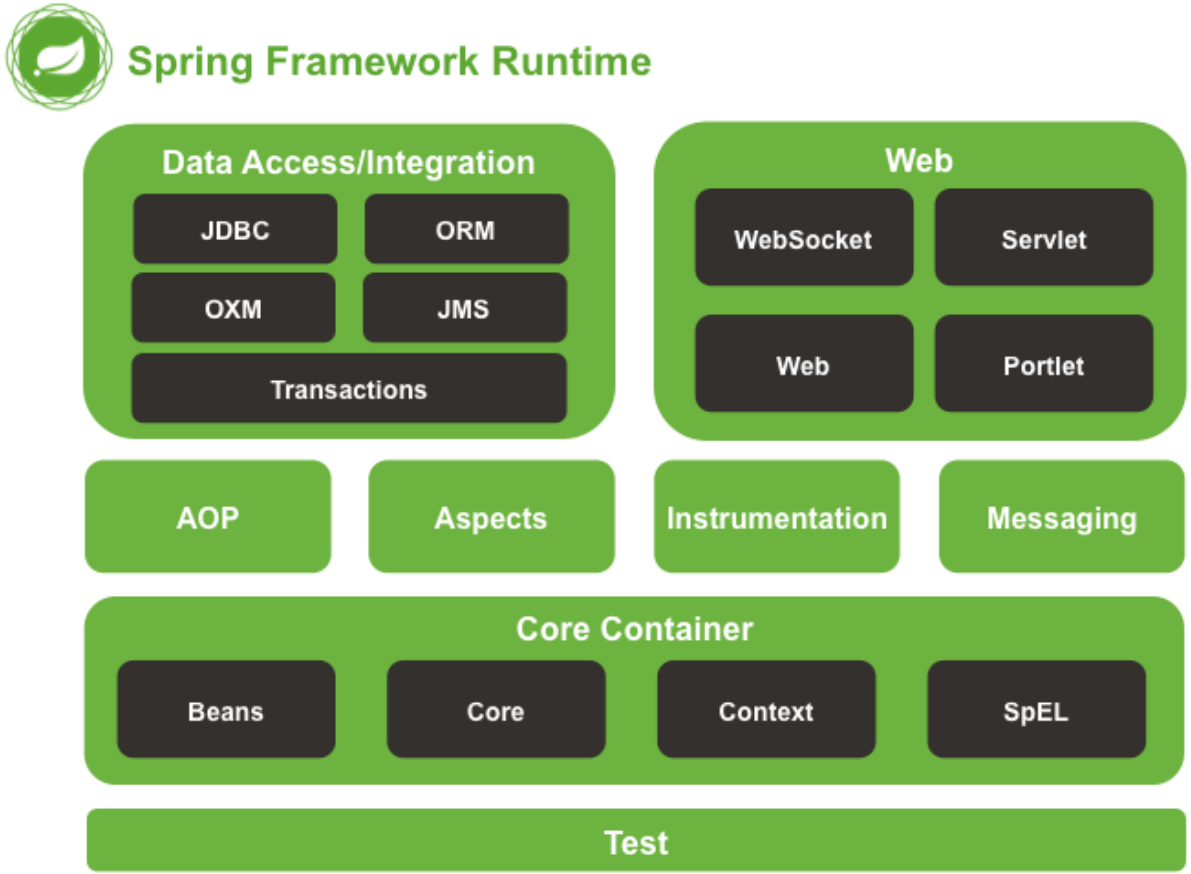
Spring 5

Spring 5 Features

- Built on Java SE 8, Java EE 7
- `@Nullable` and `@NotNull` – compile time validation of null values
- Kotlin support – functional programming (web endpoints/bean registration).
- Reactive programming – "async logic without callbacks" (WebFlux)

Spring Modules

Spring Landscape



source: Spring documentation,
`docs.spring.io/spring/docs/current/spring-framework-reference/html/overview.html`

Selected Spring Modules

Spring Core	framework core
Spring ORM	JPA integration and ORM
Spring MVC	MVC web framework, REST controllers
Spring Test	testing support
Spring Security	application security support
Spring Data	access to data – paging, filtering, map-reduce
Spring Integration	enterprise integration patterns – gateways, channels, adapters
Spring Boot	

Spring Boot

Spring Boot

- Spring module for rapid standalone application development
- Greatly simplifies configuration and deployment
- Taking convention over configuration to the next level
  - Composed annotations group common annotations
  - Sensible configuration defaults
  - Automatic classpath scan for beans
  - Package as jar for simple startup – embedded application server (Tomcat or Jetty) for web applications
- Externalized configuration – `application.properties`

Spring Boot II

- To simplify configuration even more, starter projects containing common dependencies are provided
  - *spring-boot-starter-parent* – parent Maven project
  - *spring-boot-starter-data-jpa*
  - *spring-boot-starter-web*
  - *spring-boot-starter-security*
  - ...
- Test extensions allowing to isolate tested components
  - `@DataJpaTest`, `@SpringBootTest`
- Automatic creation of default beans
  - `@ConditionalOnMissingBean`
  - `ObjectMapper`, `DataSource`, `TransactionManager`
  - `TestRestTemplate` for tests

Resources

- Spring home
  https://spring.io/
- Spring Framework – Documentation
  https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html
- Spring Boot – Documentation
  https://docs.spring.io/spring-boot/docs/current/reference/html/
- Spring (WPA lecture)
  https://cw.fel.cvut.cz/wiki/_media/courses/a7b39wpa/spring1.pdf