Spring, DI, moduly

- dependency injection
- convention over configuration
- hodně komponentů, modulární
- open-source, POJO-Based
- EJB je částí Java EE, Spring ne

DI using XML

SchoolInformationSystem.java

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

public class SchoolInformationSystem {
  private CourseRepository repository;

  public static void main(String[] args) {
    final String C
      = "classpath*:application-config.xml";
    final ApplicationContext ac
      = new ClassPathXmlApplicationContext(C);
    SchoolInformationSystem s = ac.getBean(
      SchoolInformationSystem.class
    );

    System.out.println(s.repository.getName());
  }
}
```

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 ...

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

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>
```

- **Dependency Injection** – aplikační life-cycle kontroluje container odpovědný za dodání té správné implementace dané beany
- **Inversion of Control** – naprogramovaná aplikace je „knihovna“ pro generický framework, který kontroluje aplikační lifecycle
- **Hollywood Principle** - Don’t call us, we’ll call you.

DI using Annotations

SchoolInformationSystem.java

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

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

  public static void main(String[] args) {
    final String C
      = "classpath*:application-config.xml";
    final ApplicationContext ac
      = new ClassPathXmlApplicationContext(C);
    SchoolInformationSystem s = ac.getBean(
      SchoolInformationSystem.class
    );

    System.out.println(s.repository.getName());
  }
}
```

InMemoryCourseRepository.java

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

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

application-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans ... >
  <context:annotation-config/>
  <context:component-scan base-package
    ="cz.cvut.kbss.ear.spring_example"/>
</beans>
```

CourseRepository.java

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

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;

  ...
}
```

InMemoryCourseRepository.java

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

@Named
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; }
}
```

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"; }
}
```

- bean scopes:
  - singleton – jedna instance beany na Spring IoC kontejner [výchozí]
  - prototype – nová instance při každém požadavku
  - request – jedna isntance na HTTP request

- session – jedna instance na HTTP session
- globalSession – jedna instance na global HTTP session

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
    );
  }
}
```

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"; }
}
```

prints "false"

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
    );
  }
}
```

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"; }
}
```

prints "true"

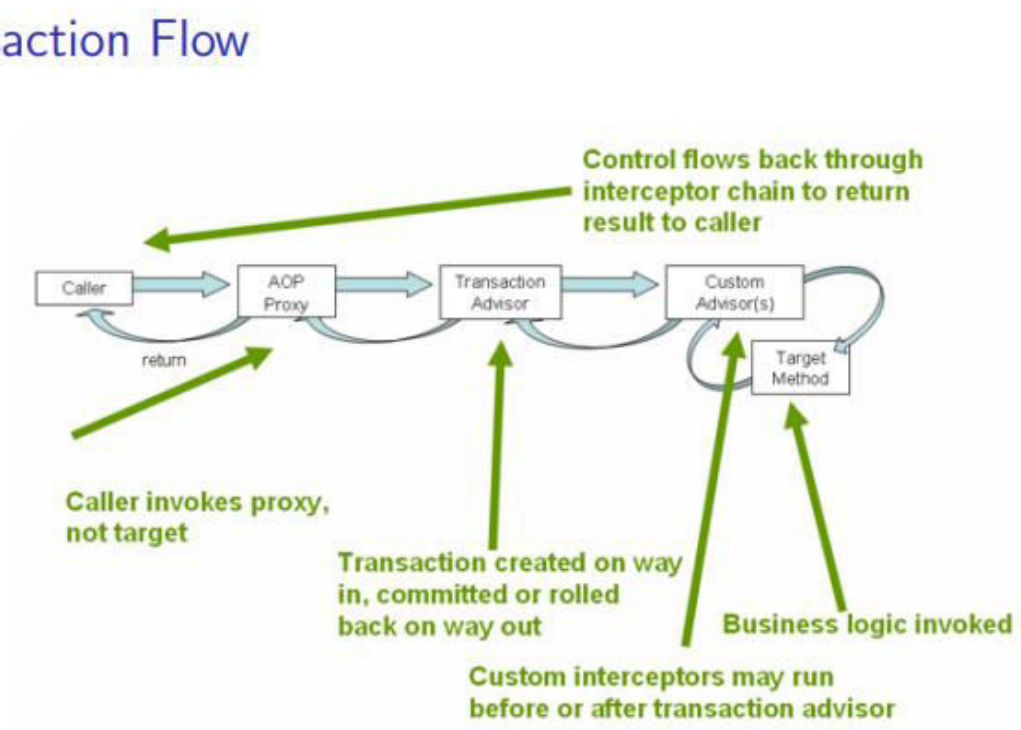
-

- např. JPA entity spravuje JPA framework, Spring o jejich existenci neví
- @Configurable – označuje třídy, jejichž objekty nejsou spravovány Springem, ale mohou injectovat objekty Springem spravované

```
@Configurable(preConstruction=true)
@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 */");
  }
}
```

Transakce

- konfigurovatelné přes XML/anotace
- globální/lokální
- wrapuje vícero transakčních API – JDBC, JTA, JPA,…



- transaction propagation – můžeme kontrolovat, jestli a jak by mělo být transakční zpracování metody podporováno … @Transactional(propagation=...

    - REQUIRED
    - SUPPORTS
    - MANDATORY
    - REQUIRES_NEW
    - NOT_SUPPORTED
    - NEVER
    - NESTED

- můžeme řídit úroveň izolace transakce… @Transactional(isolation=...
    - DEFAULT
    - READ_UNCOMMITED
    - READ_COMMITED
    - REPEATABLE_READ
    - SERIALIZABLE

- další atributy transakcí:
    - rollbackFor – které výjimky spustí rollback
    - readOnly – optimalizace v runtime
    - timeout
    - transactionManager

Spring Persistence

1.) použít standardní JPA konfiguraci přes persistence.xml a načíst ji Springem
    - znovupoužití existující konfigurace
    - dva typy XML konfigurací

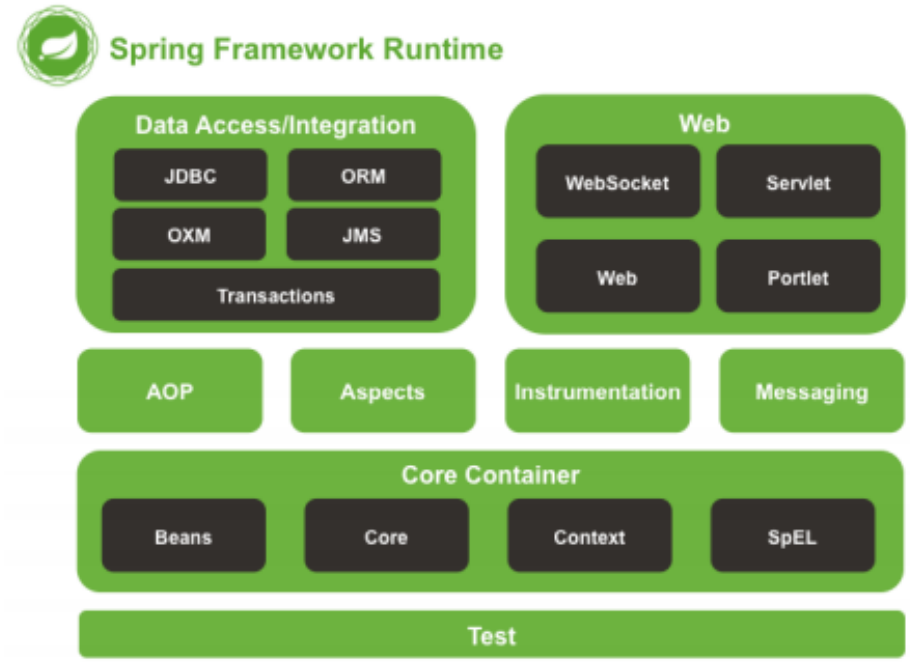
2.) konfigurovat JPA přes Spring
    - jeden typ XML konfigurace/anotací
    - 1+ závislostí na Spring

Spring 5

- založen na Java SE 8, Java EE 7
- @Nullable a @NotNull – compile-time validace null hodnot
- podpora Kotlinu
- reactive programming

Spring moduly

- Spring Core – jádro frameworku
- Spring ORM – JPA integrace a ORM
- Spring MVC
- Spring Test
- Spring Security
- Spring Social – podpora sociálních sítí
- Spring Integration


Spring Framework Runtime
Data Access/Integration
JDBC
ORM
OXM
JMS
Transactions
Web
WebSocket
Servlet
Web
Portlet
AOP
Aspects
Instrumentation
Messaging
Core Container
Beans
Core
Context
SpEL
Test