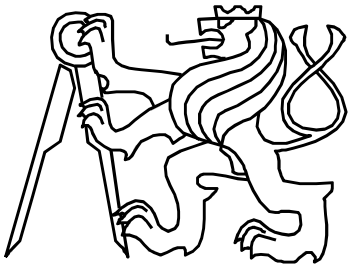
Enterprise Java (BI-EJA)
Technologie programování v jazyku Java (X36TJV)

Ing. Zdeněk Troníček, Ph.D.

Katedra softwarového inženýrství

Fakulta informačních technologií ČVUT v Praze



Letní semestr 2010/2011, přednáška č. 11
https://edux.fit.cvut.cz/courses/BI-EJA
https://edux.feld.cvut.cz/courses/X36TJV

Agenda

- Framework Spring
- Kontejner, injekce závislostí
- Aspektově orientované programování

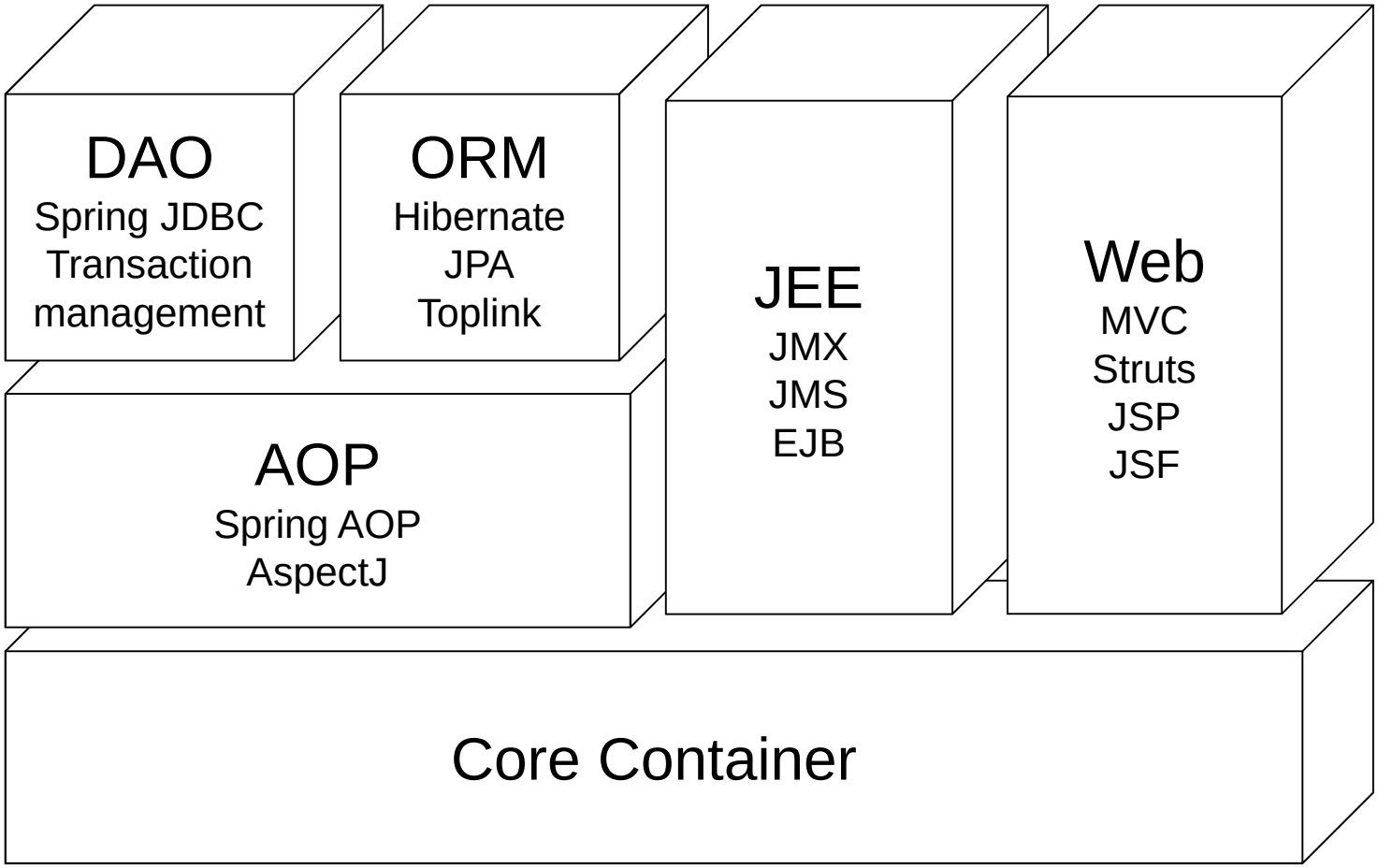
Framework Spring

- lightweight container
- dependency injection (inversion of control)
- aspect-oriented
- non-invasive
- Spring bean = Spring managed object

Hollywood Principle

"Don't call me, I'll call you."

Spring Modules



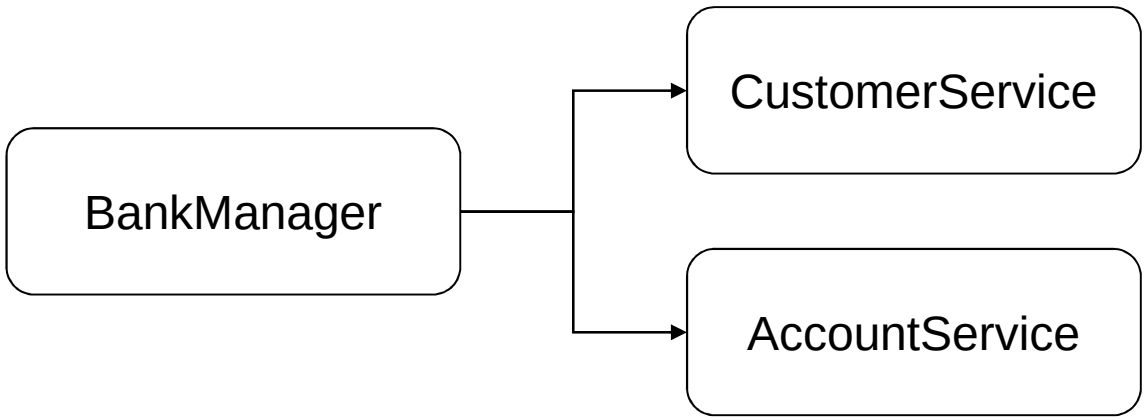
Core Container

BeanFactory

- generic factory
- implements dependency injection

Service

- POJO
- EJB
- RMI object
- web service



Příklad (1)

```
public interface CustomerService { … }
```

```
public interface AccountService { … }
```

```
public class CustomerServiceBean implements CustomerService { … }
```

```
public class AccountServiceBean implements AccountService { … }
```

```
public interface BankManager {
    void addNewCustomer( String name, String email, String currency );
}
```

Příklad (2)

```
public class BankManagerBean implements BankManager {
    private CustomerService customerService;
    private AccountService accountService;

    public void addNewCustomer( String name, String email,
                                String currency ) { … }

    public void setCustomerService( CustomerService customerService ) {
        this.customerService = customerService;
    }
    public void setAccountService( AccountService accountService ) {
        this.accountService = accountService;
    }
}
```

Příklad (3)

```
<beans>
   <bean id="bankManager" class="bank.BankManagerBean">
      <property name="customerService" ref="customerService"/>
      <property name="accountService" ref="accountService"/>
   </bean>
   <bean id="customerService" class="bank.CustomerServiceBean"/>
   <bean id="accountService" class="bank.AccountServiceBean"/>
</beans>
```

Příklad (4)

```
ApplicationContext context =
    new ClassPathXmlApplicationContext( "config.xml" );

BankManager manager = (BankManager)
    context.getBean( "bankManager" );

manager.addNewCustomer( "Rumcajs", "rum@email.cz", "CZK" );
```

Vytváření objektů

Voláním konstruktoru

```
<bean id="exampleBean" class="examples.ExampleBean"/>
```

Pomocí statické tovární metody

```
<bean id="exampleBean" class="examples.StaticFactory"
factory-method="createInstance"/>
```

Pomocí instanční tovární metody

```
<bean id="exampleBean" factory-bean="serviceLocator"
factory-method="createService"/>
```

Injekce závislostí (1)

Constructor Injection

```
public class MovieManager {
    private MovieFinder movieFinder;

    public MovieManager( MovieFinder movieFinder ) {
        this.movieFinder = movieFinder;
    }
    …
}
```

Injekce závislostí (2)

Setter Injection

```
public class MovieManager {
    private MovieFinder movieFinder;

    public void setMovieFinder( MovieFinder movieFinder ) {
        this.movieFinder = movieFinder;
    }
    …
}
```

Autowiring

- byName:

  set**LeagueService**(LeagueService leagueService)
- byType:

  setLeagueService(**LeagueService** leagueService)
- constructor:

  RegisterServiceBean(**LeagueService** leagueService)

```
<bean id="registerService" class="league.RegisterServiceBean"
autowire="byType"/>
```

Bean Scopes

- singleton: one instance per container
- prototype (non-singleton)
- web-based (request, session, global session)
- custom scope

```
<bean id="…" class="…">
  <property name="accountDao"
    ref="accountDao"/>
</bean>
```

```
<bean id="…" class="…">
  <property name="accountDao"
    ref="accountDao"/>
</bean>
```

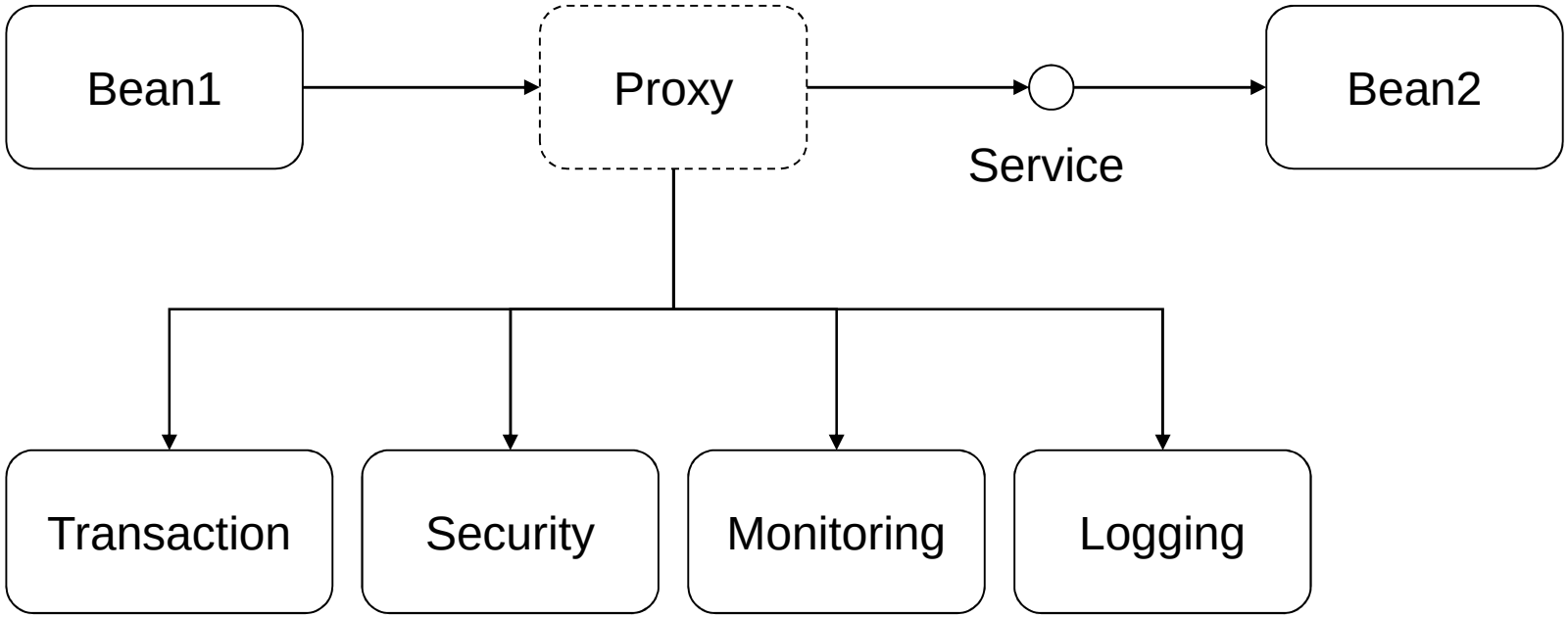
```
<bean id="accountDao" class="…" scope="singleton"/>
```

Konfigurace

```
<context:property-placeholder location="app.properties"/>
```

```
<bean id="registerService" class="RegisterServiceBean">
    <property name="dataDir">
        <value>${application.dataDir}</value>
    </property>
</bean>
```

Aspect Oriented Programming



AOP Concepts

- Aspect: crosscutting concern, e.g. transaction management
- Join point: a point during the execution of a program – method execution
- Advice: action taken by an aspect at a particular join point
- Pointcut: a predicate that matches join points

Types of Advice

- Before advice (@Before)
- After returning advice (@AfterReturning)
- After throwing advice (@AfterThrowing)
- After (finally) advice (@After)
- Around advice (@Around)

Příklad Before

```
@Aspect
public class RegisterAspect {
    private int count;

    @Before( "execution (* springapp.RegisterService+.*(..))" )
    public void used() {
        count++;
    }

    public int getCount() {
        return count;
    }
}
```

Příklad Around

```
@Aspect
public class AroundAspect {

    @Around( "execution (* springapp.RegisterService+.add*(..))" )
    public Object profile( ProceedingJoinPoint pjp ) throws Throwable {
        long start = System.nanoTime();
        try {
            return pjp.proceed();
        } finally {
            long t = System.nanoTime() - start;
            System.out.println( "time: " + t );
        }
    }
}
```

Otázky & odpovědi

tronicek@fit.cvut.cz