Advanced Java technologies: JBoss

Časť 2.

Contexts and Dependency Injection (CDI)

Enterprise JavaBeans

Jozef Hartinger

November 2014

redhat

@Inject

@SessionScoped

redhat

Contexts and Dependency Injection for the Java EE platform (CDI)

- Java EE Špecifikácia
  - Od Java EE 6 (2009)
- Niekoľko implementácií
  - **Weld** (JBoss)
  - **OpenWebBeans** (Apache)
  - **CanDI** (Resin)
- Aktuálna verzia 1.2

redhat

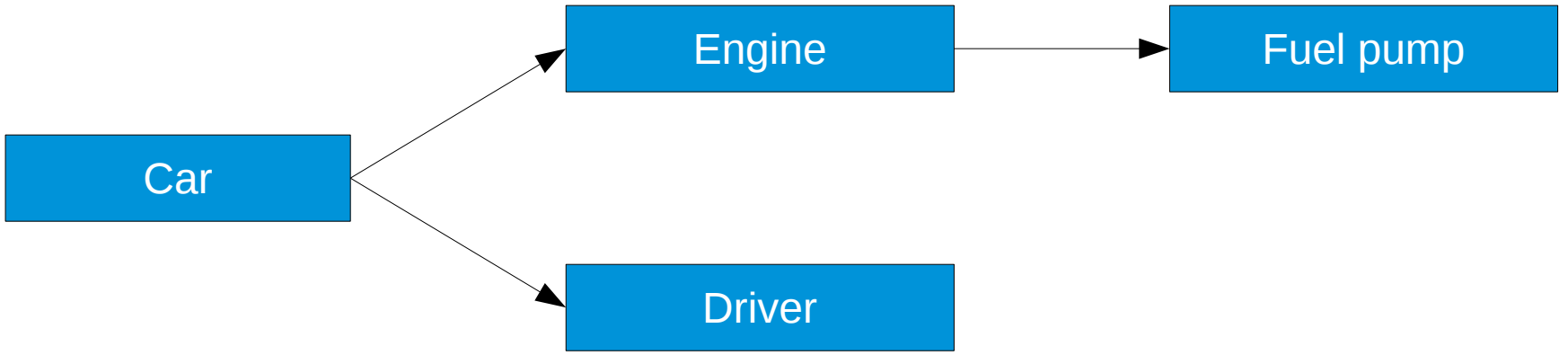
Contexts and Dependency Injection for the Java EE platform (CDI)

- Definuje komponentový model
  - komponenty spravované servrom
  - komponentám sú poskytovné služby
    - správa životného cyklu
    - správa závislostí
    - etc.
- Integrácia vrámci Java EE

redhat

Dependency Injection (@Inject)

- Inversion of control



redhat

Loose coupling (Voľné prepojenie)

- Rozšíriteľnost a zmeny aplikácie
- Testovanie

Strong typing

- Java (rozhrania, triedy, anotácie)
- Refactoring
- Kompilátor

redhat

Charakteristika CDI Beany

- Trieda
- Nie je abstraktná
- Má vhodný konštruktor
  - Bezparametrický
  - @Inject (viď. Ďalej)
- Obsahuje **“bean defining annotation”**
- V prípade, že využíva proxy
  - Nie je final
  - Nemá final metódy

- Session Beany (EJB) sú zároveň CDI Beany

redhat

Vlastnosti CDI Beany

- Scope (viď. ďalej)
- Množina typov (Bean type closure)
  - Všetky nadtypy a všetky (aj nepriamo) implementované rozhrania
- Množina kvalifikátorov (Qualifiers – viď. Ďalej)
- Voliteľne
  - Meno (@Named)
  - Množina stereotypov (Stereotypes)
  - Množina interceptor väzieb (Interceptor bindings)

redhat

Príklad CDI Beany

```
@RequestScoped
public class Car {

    @Inject
    private Driver driver;
    @Inject
    private Engine engine;

    public void driveTo(Location location) {
        engine.start();
        // TODO
    }
```

redhat

Príklad CDI Beany

```
@SessionScoped
@Named
@Stateful
public class LoginManager implements Serializable {

    private static final long serialVersionUID = 1L;

    @Inject
    private EntityManager em;

    @Inject
    private UserManager userManager;

    private User currentUser;
```

redhat

Životný cyklus

- Inversion of control – o životný cyklus sa stará kontainer

- **Vytvorenie novej inštancie** (viď. constructor injection)
- Field injection (naplnenie atribútov objektu)
- @PostConstruct / @Inject initializer
- **Uloženie do kontextu** (scope)
- @PreDestroy / @Disposer method (viď. ďalej)

redhat

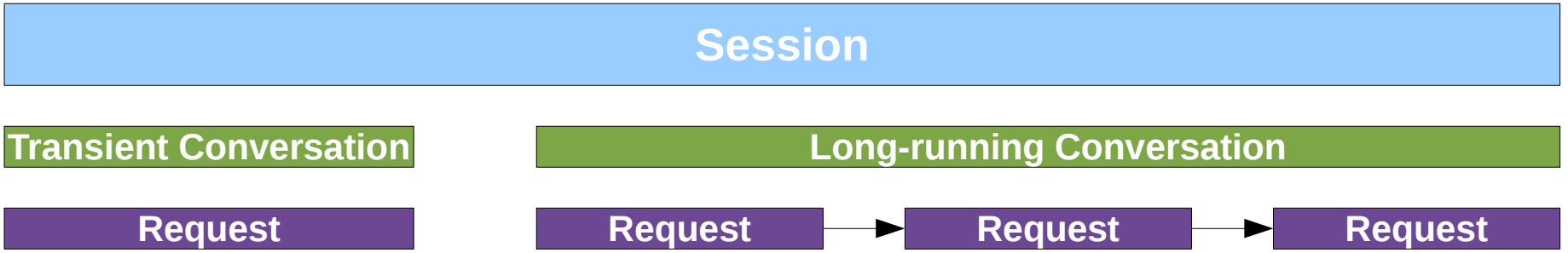
Kontexty - Scopes

- Servlet
  - Request Scope- @RequestScoped
  - Conversation Scope - @ConversationScoped
  - Session Scope - @SessionScoped
  - Application Scope - @ApplicationScoped
- Dependent Scope - @Dependent
  - Vždy sa vytvára nová inštancia
  - Jej životný cyklus je zviazaný s Beanou ktorá ju požaduje
- JTA
  - @TransactionScoped

redhat

Conversation scope

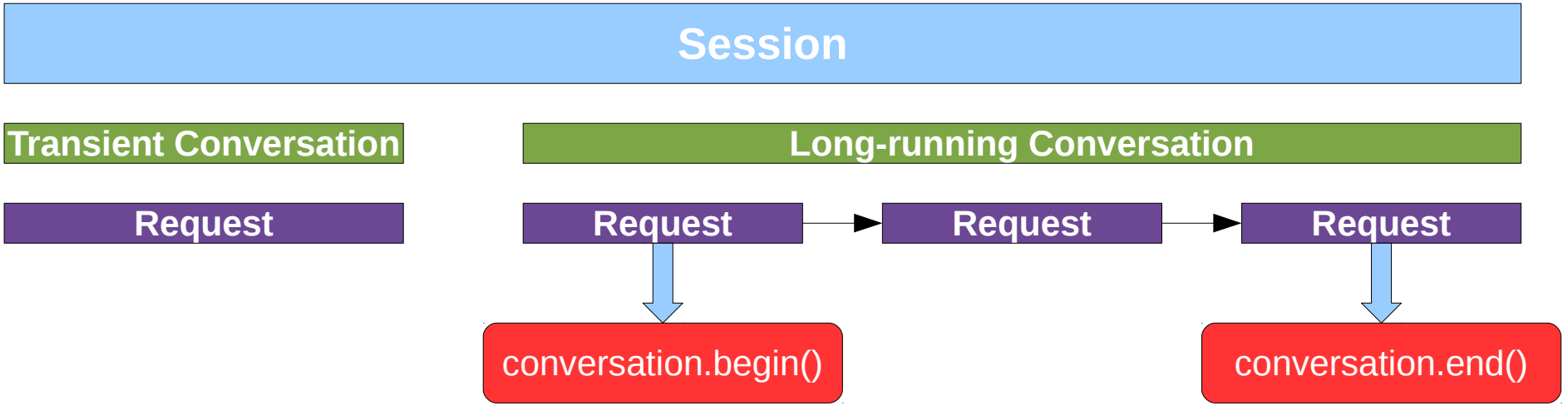
- Uchováva stav sekvencie po sebe idúcich logicky súvisiacich requestov
- Samotná konverzácia má 2 stavy
  - **Transient** – jeden Servlet request
  - **Long-running** – sekvencia Servlet requestov



redhat

Ovládanie konverzácií

```
public class NewAuctionWizzard {

    @Inject
    private Conversation conversation;
```



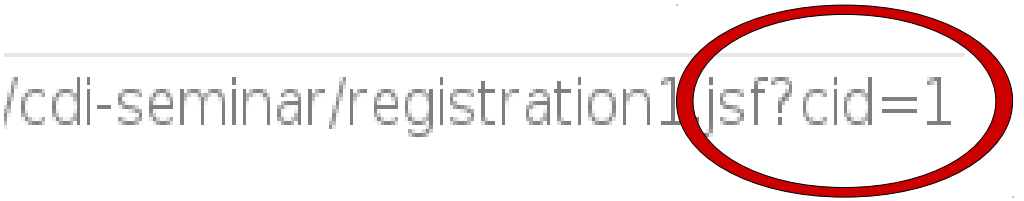
redhat

Conversation API

```
public interface Conversation
{
   public void begin();

   public void begin(String id);

   public void end();

   public String getId();

   public long getTimeout();

   public void setTimeout(long milliseconds);

   public boolean isTransient();
}
```

Propagácia konverzácie

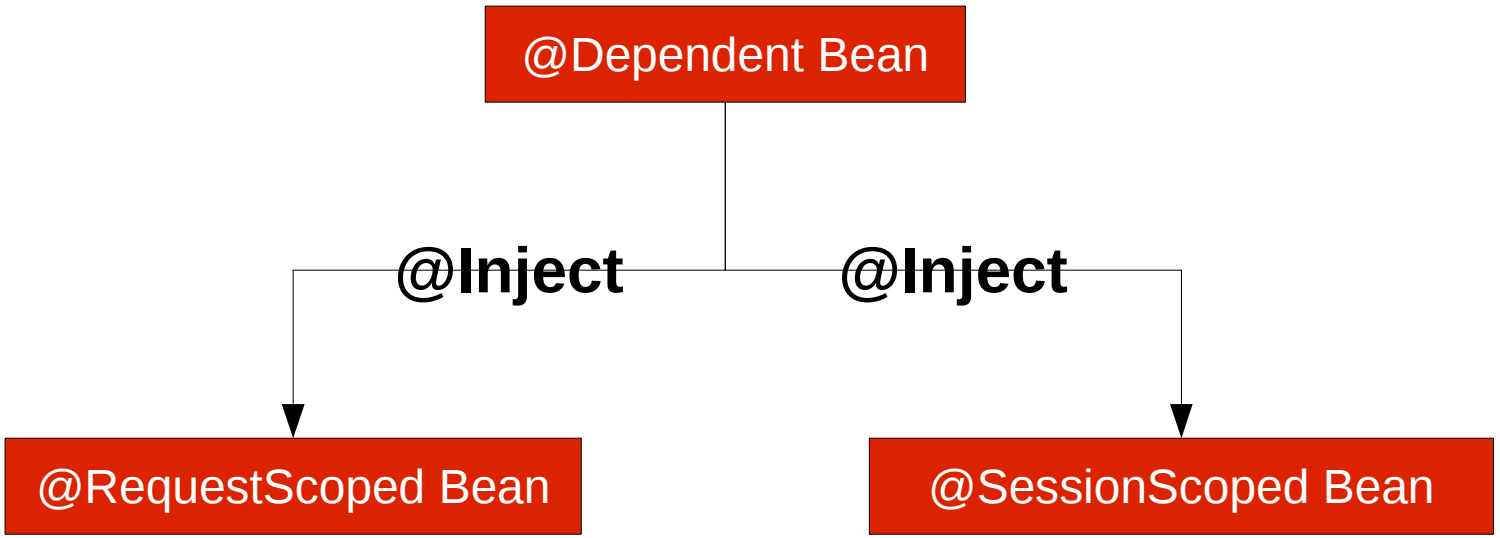
- String identifier



  - Set by application
  - Generated by container
- Vrámci session
- Propagácia pomocou **cid** parametru
  - Automatická propagácia pri odosielaní JSF formuláru
  - Manuálna pri plain Servlet request

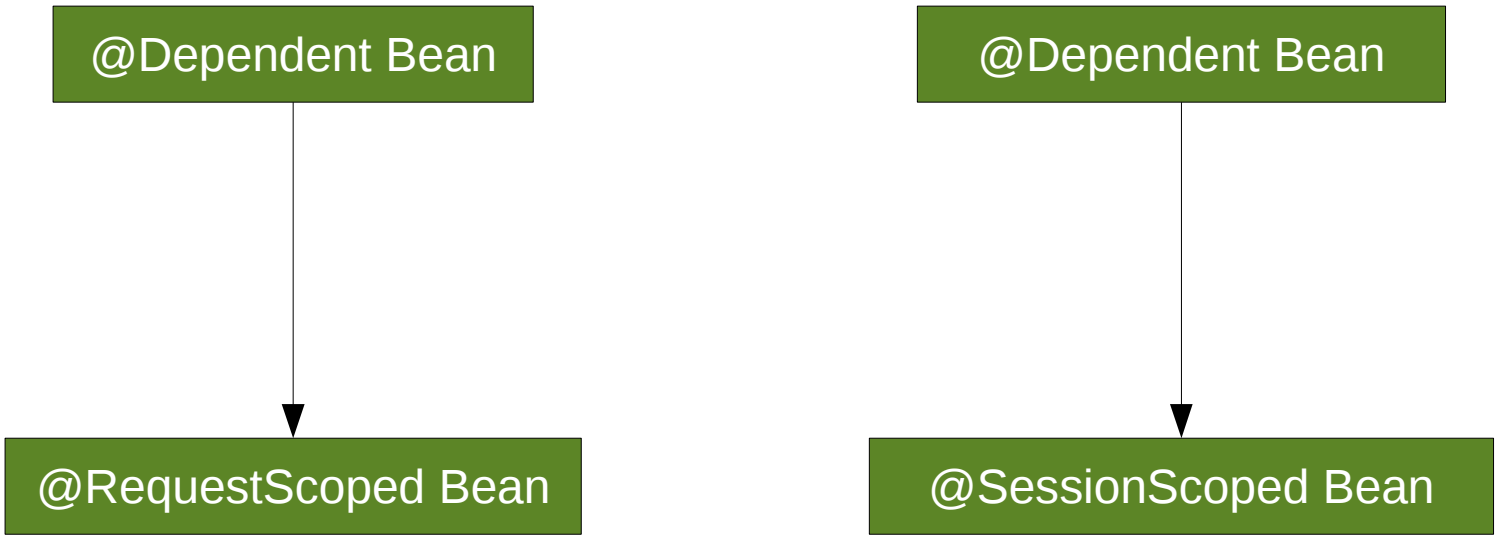
Dependent Pseudo Scope

- Na úrovni tried



Dependent Pseudo Scope

- Na úrovni objektov



redhat

Passivation

- Veľkosť pamäte potrebnej pre uloženie sessions rastie lineárne s počtom existujúcich sessions
- Session a Conversation – passivating scopes
- Odkladanie neaktívnych sessions na sekundárne úložisko
- Replikácia sessions
  - Load balancing
  - Failover
- Každá @SessionScoped alebo @ConversationScoped komponenta musí byť serializovateľná
  - java.io.Serializable
  - java.io.Externalizable

redhat

Dependency Injection

- Mechanizmus získavania závislostí
- Spúšťa sa explicitne pomocou anotácie @Inject

```
@RequestScoped
public class Car {

    @Inject
    private Driver driver;
```

redhat

Pravidlá DI – Typesafe resolution

- Výber na základe
  - Typu
  - Kvalifikátorov (Qualifiers)
    - Slúžia primárne na odlíšenie viacerých inštancií daného typu
- Na dependency injection sa použije objekt daného typu
  - Z kontextu (ak objekt existuje v kontexte)
  - Nový
    - vytvorí ju container
    - pri @Dependent sa vytvára vždy nový objekt

redhat

Qualifiers

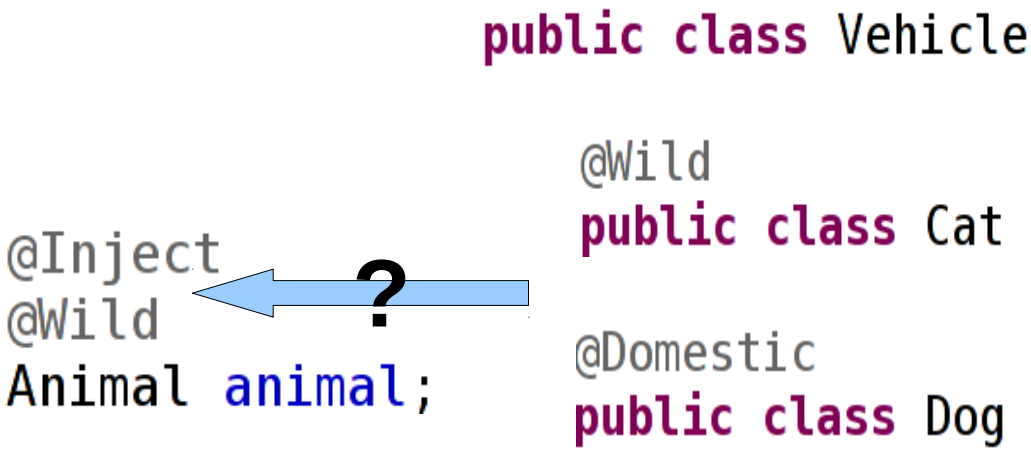
```
@Qualifier
@Retention(RetentionPolicy.RUNTIME)
public @interface LoggedIn {

}
```

```
@Inject
@LoggedIn
private User user;
```

redhat

Pravidlá DI – Typesafe resolution

- Injection point X vyžaduje
  - Typ (Z)
  - Množinu kvalifikátorov
- Bean Y poskytuje
  - Množinu typov
  - Množinu kvalifikátorov
- Bean Y je vhodným kandidátom ak
  - Vyžadovaný typ sa nachádza v množine poskytovaných typov
  - Všetky požadované kvalifikátory sa nachádzajú v množine poskytovaných kvalifikátorov



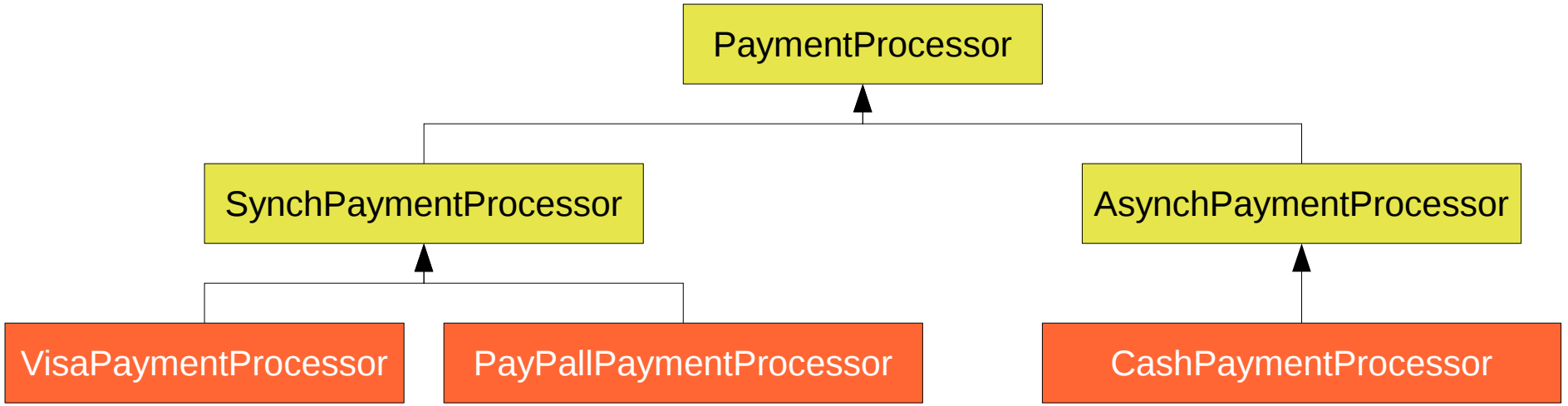
redhat

Poskytované typy (Bean types)

- Tranzitívny uzáver rozširovaných a implementovaných typov
- java.lang.Object je vzdy typom
- java.io.Serializable nie je nikdy typom

redhat

Príklad typovej rezolúcie (typy)

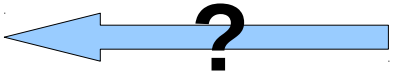



```
@Inject
private PaymentProcessor pp;

@Inject
private SynchPaymentProcessor spp;

@Inject
private VisaPaymentProcessor vpp;
```

redhat

Príklad typovej rezolúcie (kvalifikátory)

```
@Inject
@Any
private Animal animal;

@Inject
@Domestic
private Animal domestic;

@Inject
@Wild @Lazy
private Animal lazyWildAnimal;
```




```
public interface Animal

public class Sheep

@Domestic
public class Dog

@Wild
@Hungry
public class Lion

@Wild
public class Elephant

@Domestic
@Lazy
public class Cat
```

redhat

Špeciálne kvalifikátory

- @Default – implicitný kvalifikátor v prípade, že daná komponenta / injection point nedefinuje iný kvalifikátor
- @Any - implicitný kvalifikátor každej komponenty
- @Named

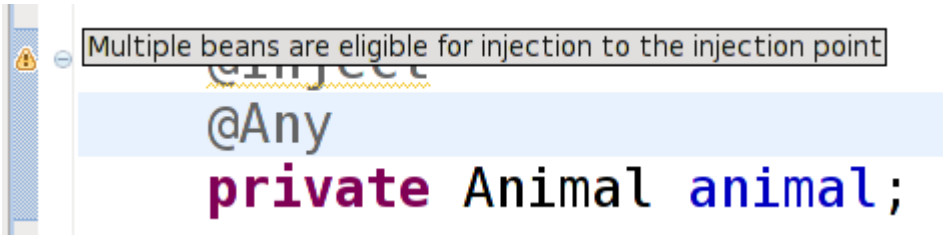
```
public class Shelter {

    @Inject
    Animal animal1;

    @Inject @Default
    Animal animal2;

    @Inject @Any
    Animal animal3;
}
```

```
public class Dog
    implements Animal {
}
```

```
@Domestic
public class Cat
    implements Animal {
}
```

Pravidlá DI – Typesafe resolution

- 1 injection point = práve jeden vhodný Bean
- Kontrolované pri deploy (alebo tooling)



redhat

Typy DI

```
@Inject
private PaymentProcessor pp;
```

- Field injection
- Constructor
  - Immutable objects
- Initializer methods
- Producer method/Disposer/Observer

```
@Inject
public PaymentManager(PaymentProcessor pp) {
    this.pp = pp;
}
```

```
@Inject
public void init(PaymentProcessor pp) {
    this.pp = pp;
}
```

redhat

Producer method

- Factory - pre prípady keď volanie konštruktoru trieda nestačí na vytvorenie objektu
- Rozšírená kontrola
  - Výsledok JPA dotazu
  - Náhodné číslo
  - ...
- Konkrétna metóda (nie je abstraktná)
  - Umiestnená na CDI Bean
  - Tranzitívny uzáver typu návratovej hodnoty = typ objektu
  - Kvalifikátory, Scope a meno objektu sa definujú na metóde

redhat

Producer method

```
@ApplicationScoped
public class RandomNumberGenerator {

    private final java.util.Random random = new java.util.Random();

    @Produces
    @Random
    public int generateRandomInt() {
        return random.nextInt();
    }
}
```

redhat

Producer method

```
@Produces
@Named
@CurrentAuction
public Auction getCurrentAuction() {
    if (currentAuction != null &&
            !em.contains(currentAuction)) {
        currentAuction = em.merge(currentAuction);
    }
    return currentAuction;
}
```

redhat

Producer field

- Zjednodušenie producer metód

```
@Produces
@Zero
private final int zero = 0;
```

redhat

Disposer method

- Niektoré objekty vyžadujú explicitné zničenie
- @PreDestroy nefunguje na produkovaných objektoch

```
public void close(@Disposes Connection connection) {
    try
    {
        connection.close();
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
}
```

redhat

Alternatives

- @Alternative
- Nahradenie komponenty inou na základe prostredia
  - Konfigurácia
  - Testovanie
- Alternative musí byť explicitne aktivovaná (enabled) pomocou **@Priority**
- Rozšírenie **Typesafe resolution** algoritmu
  1. Výber vhodných kandidátov na základe typu a kvalifikátorov
  2. V prípade, že existuje viac ako jeden kandidát sú eliminovaní všetci kandidáti okrem aktivovaných alternatives
  3. V prípade, že všetci kandidáti majú definovanú prioritu, je vybraný ten s najvyššou prioritou

redhat

Integrácia s prezentačnou vrstvou

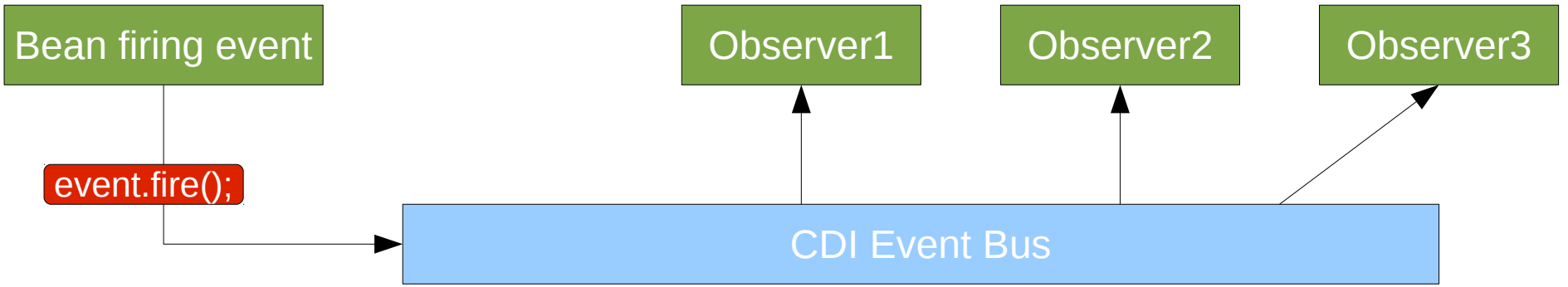
- Prístup k CDI komponentám pomocou mena
- @Named

```
<h:form>
    <h:inputText value="#{student.firstName}" />
    <h:inputText value="#{student.surname}" />
    <h:commandButton id="register"
        action="#{registrator.register}" value="Register" />
</h:form>
```

redhat

Udalosti (Events)

- Komunikácia pomocou odoslania správ
- Oddelenie producentov správ od ich konzumentov
- Loose coupling – ďalšie observery sa môžu pridať neskôr bez zásahu do producenta udalostí
- Synchrónne
- Event = message (payload)



redhat

Udalosti (Events) – Odoslanie správy

- Vstavaný Event bean

```
public class RegistrationManager {
    @Inject
    @Registered
    private Event<User> userEvent;
```

- Vyvolanie udalosti

```
public void register()
{
    em.persist(user);
    userEvent.fire(user);
}
```

redhat

Vlastnosti správy

- Typ správy
- Qualifiers

```
public class RegistrationManager {
    @Inject
    @Registered
    private Event<User> userEvent;
```

redhat

Udalosti (Events) – Doručenie správy

- Observer method
- Pravidlá typovej rezolúcie rovnaké ako pre DI

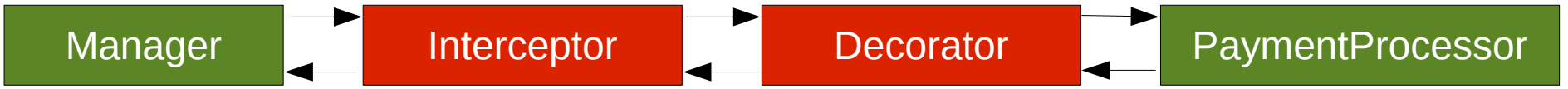
```
public void log(@Observes User user) {
    System.out.println("Hello " + user.getName());
}
```

```
public void log(@Observes @Registered User user) {
    System.out.println("Hello " + user.getName());
}
```

redhat

Dekorátory a Interceptory

- Aspect-oriented programming
- Odchytávanie volaní metód
- Izolovanie kódu, ktorý by sme inak opakovali v metódach
- Zmena správania v závislosti na prostredí



redhat

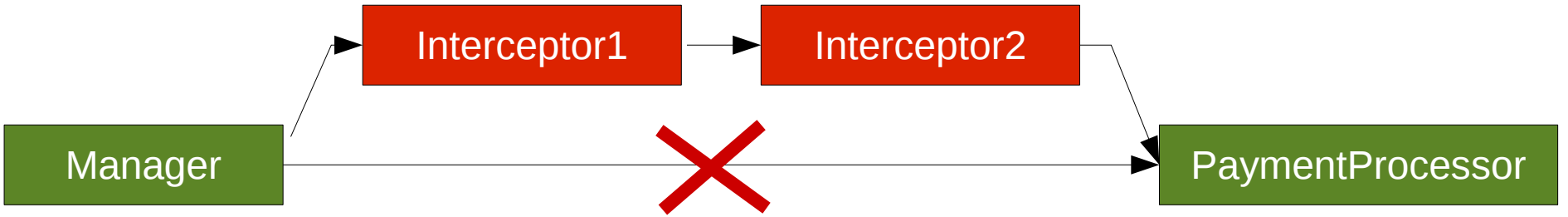
Interceptory

- Oddelenie technických detailov od logiky aplikácie
- Cross-cutting concenrs (logovanie, zabezpečenie, caching)
- Môžu odchytávať
  - Volania metód
  - Životný cyklus
    - @AroundConstruct
    - @PostConstruct
    - @PreDestroy

redhat

Interceptory

```
public class PaymentManager {

    @Inject
    private PaymentProcessor pp;

    pp.foo();
```



redhat

Interceptory

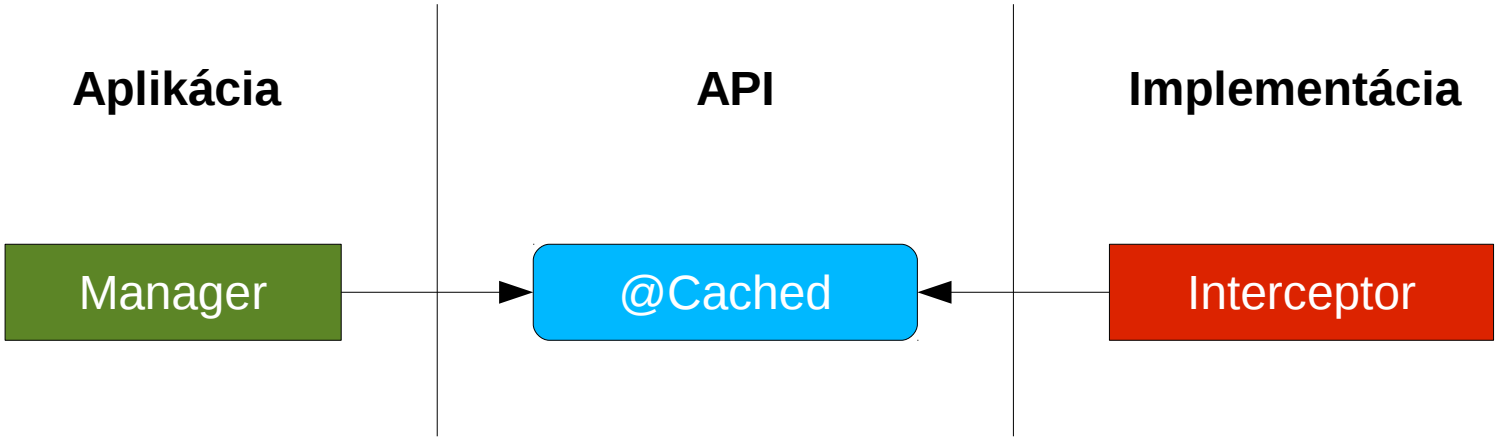
```
@Interceptor
@Dependent
@Cached
@Priority(Interceptor.Priority.LIBRARY_BEFORE)
public class CachingInterceptor {

    @AroundInvoke
    public Object intercept(InvocationContext ctx) throws Exception {
        // do something before the call
        Object result = ctx.proceed();
        // do something after the call
        return result;
    }
}
```

redhat

Interceptor binding

- Anotácia reprezentujúca funkcionalitu poskytovanú interceptorom (loose coupling)

```
@InterceptorBinding
@Retention(RetentionPolicy.RUNTIME)
public @interface Cached {

}
```



redhat

Decorator

- “Typovaný” interceptor – “pozná” sémantiku volania ktoré obaľuje
- Nie je nutné anotovať odchytávaný objekt
- Implementuje priamo metódy rozhrania ktoré obaľuje
  - Nemusí implementovať všetky metódy rozhrania (abstraktná trieda)
- Špeciálny injection point (**@Delegate**)

redhat

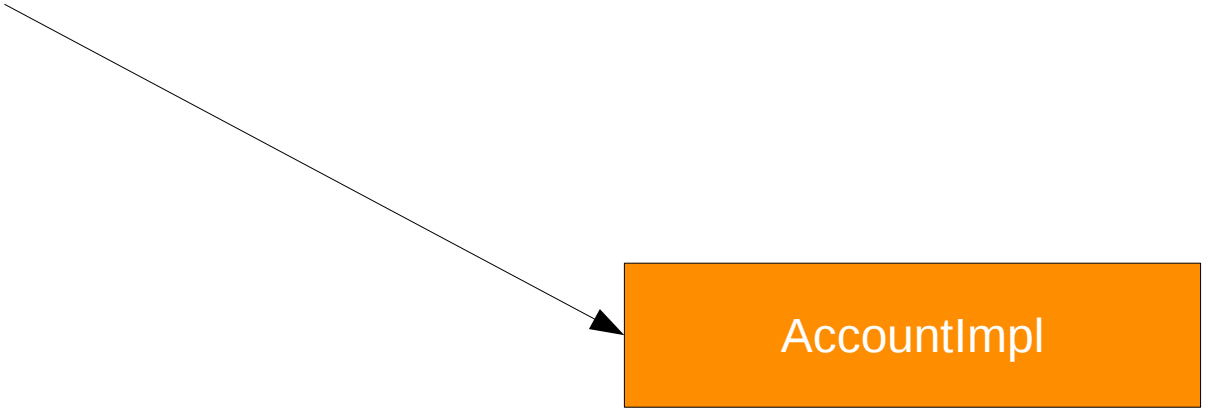
Decorator

```
public interface Account {
    void withdraw(BigDecimal amount);
}


@Dependent
public class AccountImpl implements Account {

    private BigDecimal balance;

    public AccountImpl(BigDecimal initialBalance) {
        this.balance = initialBalance;
    }

    public void withdraw(BigDecimal amount) {
        balance.subtract(amount);
    }
}
```

redhat

Decorator

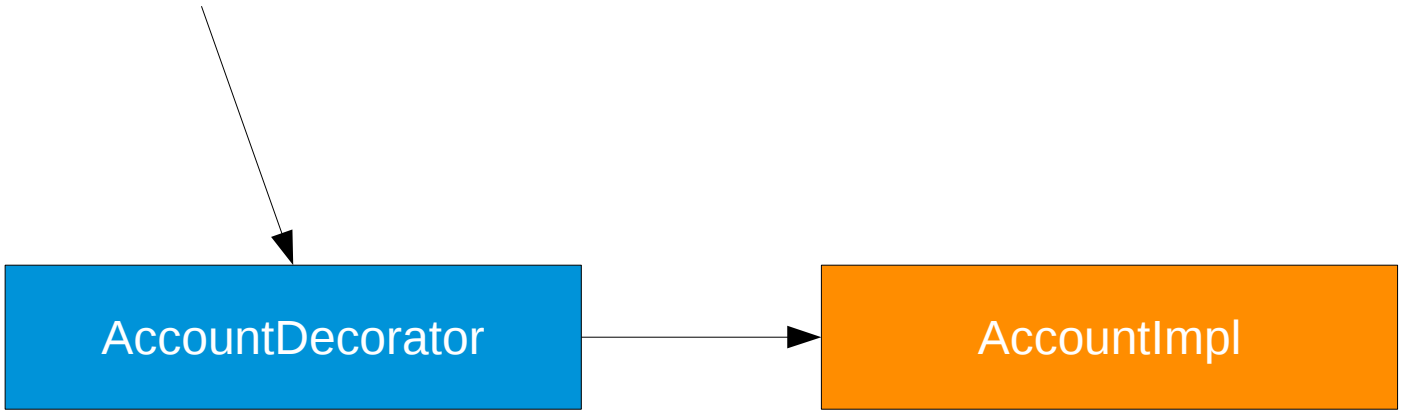
```
@Inject
private Account account;

account.withdraw(BigDecimal.TEN);
```



redhat

Decorator

```
@Inject
private Account account;

account.withdraw(BigDecimal.TEN);
```



redhat

Decorator

```
public class AccountDecorator implements Account {


    private final Account delegate;

    public AccountDecorator(Account delegate) {
        this.delegate = delegate;
    }

    @Override
    public void withdraw(BigDecimal amount) {
        if (amount.compareTo(LIMIT) > 0 &&
                !isAuthorizedForLargeTransactions()) {
            throw new SecurityException("Amount too high!");
        }
        delegate.withdraw(amount);
    }
```

redhat

Decorator

```
@Dependent
@Decorator
@Priority(Interceptor.Priority.APPLICATION)
public class AccountDecorator implements Account {

    private final Account delegate;

    @Inject
    public AccountDecorator(@Delegate @Any Account delegate) {
        this.delegate = delegate;
    }

    @Override
    public void withdraw(BigDecimal amount) {
        if (amount.compareTo(LIMIT) > 0 &&
                !isAuthorizedForLargeTransactions()) {
            throw new SecurityException("Amount too high!");
        }
        delegate.withdraw(amount);
    }
```

redhat

Aktivácia a poradie

- @Priority(2000) – magic number
- Interceptors, Decorators, Alternatives

Interceptor.PRIORITY.

0 - 999	PLATFORM_BEFORE
1000 - 1999	LIBRARY_BEFORE
2000 - 2999	APPLICATION
3000 - 3999	LIBRARY_AFTER
4000 - 4999	PLATFORM_AFTER

- Transactional interceptor (200)
- Bean Validation interceptor (4800)

redhat

Stereotypy

- Obrana pred “annotation hell”
- Anotácia združujúca
  - Scope
  - Interceptor bindings
  - @Name
- Vstavaný interceptor **@Model**

```
@Named
@RequestScoped
@Documented
@Stereotype
@Target( { TYPE, METHOD, FIELD })
@Retention(RUNTIME)
public @interface Model
{
}
```

redhat

Integrácia

- Beans
  - @Inject **HttpServletRequest**
  - @Inject **UserTransaction**
  - @Inject **javax.security.Principal**
- Events
  - **@Initialized(SessionScoped.class) HttpSession**
- Scopes
  - **@TransactionScoped**

redhat

Integrácia – non-contextual components

- Servlets / Filters
- JPA Entity listeners
- JSF converters / validators
- JAX-WS / JAX-RS / WebSocket endpoints

- @Inject (the component itself cannot be injected anywhere)
- Firing events (no observing support)
- Interceptors

redhat

Contexts and Dependency Injection

- Vlastnosti CDI komponent
- Životný cyklus (Scopes)
- Dependency Injection
- Udalosti (Events)
- Dekorátory a Interceptory (Decorators and Interceptors)
- Integrácia

redhat

Tipy na záver

- Nezabúdať **scope annotation**
- Pozor na správny FQCN!
  - javax.inject.Inject
  - javax.enterprise.inject.Produces
  - javax.enterprise.context.*Scoped
  - javax.enterprise.event.Event
  - javax.enterprise.event.Observes
  - javax.enterprise.inject.spi.Extension;

redhat

Enterprise JavaBeans

- Komponentový model poskytujúci služby:
  - Resource injection
  - Transactions
  - **Security** (separátna prednáška)
  - Remote method invocation (RMI) - @Remote
  - Naming and directory services - JNDI
  - **Messaging** (JMS)
  - **Asynchronous invocations**
  - **Scheduling** (Timer service)
  - Concurrency control

redhat

Enterprise JavaBeans

- Session beans
  - Stateless
  - Stateful
  - Singleton
- Message-driven beans

redhat

Enterprise JavaBeans 3.1

- Zjednodušenie existujúceho komponentového modelu
  - Rozhrania (views) nie sú povinné
  - Session beans sú automaticky CDI komponentami
    - Scopes (Stateful session beany)
    - Dependency injection
    - Events
    - Interceptory / Dekorátory
  - Možnosť používať v jednoduchej webovej aplikácii (.war)
- **@Singleton** session bean
- Asynchrónne volanie metód

redhat

Singleton session bean

- Jediná inštancia zdieľaná vrámci celej aplikácie
  - Vrámci jednej JVM
- V porovnaní s **@ApplicationScoped** navyše
  - Inicializovaná pri štarte aplikácie (**@Startup**)
  - Podporuje paralelný prístup
    - @Lock(LockType.READ)
    - @Lock(LockType.WRITE)
    - @AccessTimeout
    - @ConcurrencyManagement
- DEMO na cvičení

redhat

Asynchrónne volania metód

- Abstrakcia od priameho použitia vlákien, exekútorov a thread pools
- Použitie
  - Asynchrónne vykonávanie dlhotrvajúcej úlohy
    - Odosielanie potvrdzujúceho e-mailu
    - Návratová hodnota **void**
  - Paralelizácia výpočtu
    - Návratová hodnota **Future<V>** - **AsyncResult<V>**
- **@Asynchronous**
- DEMO na cvičení

redhat

Enterprise JavaBeans 3.2

- Prevažne kozmetické zmeny
  - @Stateful(passivationCapable = false)
  - @PostConstruct/@PreDestroy volitelne transakčné
  - TimerService.getAllTimers()
    - Všetky Timers pre daný modul

redhat

Odkazy

- http://cdi-spec.org/
- http://docs.jboss.org/cdi/spec/1.2/
- http://docs.oracle.com/javaee/7/tutorial/doc/

jharting@redhat.com | www.redhat.com