Advanced Java technologies: JBoss

Časť 2.

Contexts and Dependency Injection (CDI)

Enterprise JavaBeans 3.1

Jozef Hartinger

December 2012

redhat

Contexts and Dependency Injection for the Java EE platform

- Java EE Špecifikácia (JSR-299)
- Niekoľko implementácií
  - Weld, Resin, OpenWebBeans
- Definuje komponentový model
  - komponenty spravované servrom
  - komponentám sú poskytovné služby
    - správa životného cyklu
    - správa závislostí
    - ďalšie

redhat

Motivácia

- Voľné prepojenie komponent
  - Jednoduchý komponentový model
  - Silne typovaný
- Integrácia vrámci platformy
- Rozšíriteľnosť

redhat

Loose coupling (Voľné prepojenie)

- Rozšíriteľnost a zmeny aplikácie
- Testovanie
- Inversion of control

Strong typing

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

redhat

Spôsoby zabalenia CDI aplikácie

- WAR
  - WEB-INF/beans.xml
  - WEB-INF/classes
  - WEB-INF/lib
- JAR
  - META-INF/beans.xml
- EAR

- Bean descriptor (beans.xml)

redhat

Charakteristika CDI Beany

- Trieda
- Nie je abstraktná
- Má vhodný (public) konštruktor
  - Bezparametrický
  - @Inject (viď. Ďalej)
- Nie je final a nemá final metódy
- Spĺňa ďalšie (menej podstatné) podmienky [1] [2]

- Triedy ktoré spĺňajú tieto kritéria sú implicitne CDI Beany (**nie je nutné ich nijak označiť**)
- 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
- Meno (@Name)
  - voliteľné
- Množina kvalifikátorov (Qualifiers – viď. ďalej)
- Množina stereotypov (Stereotypes)
- Množina interceptor väzieb (Interceptor bindings)

redhat

Príklad CDI Beany

```
public class Registrator {

    @Inject
    private User user;

    public void register() {
        // ...
    }
}
```

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

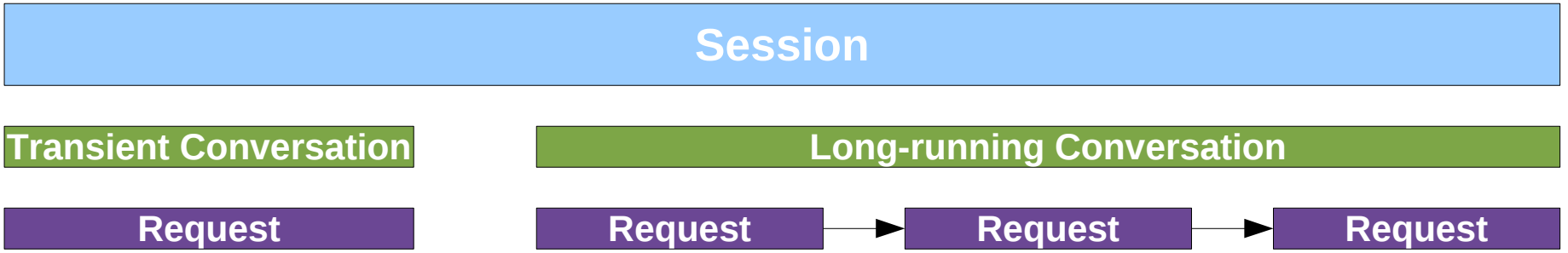
Kontexty - Scopes

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

redhat

Conversation scope

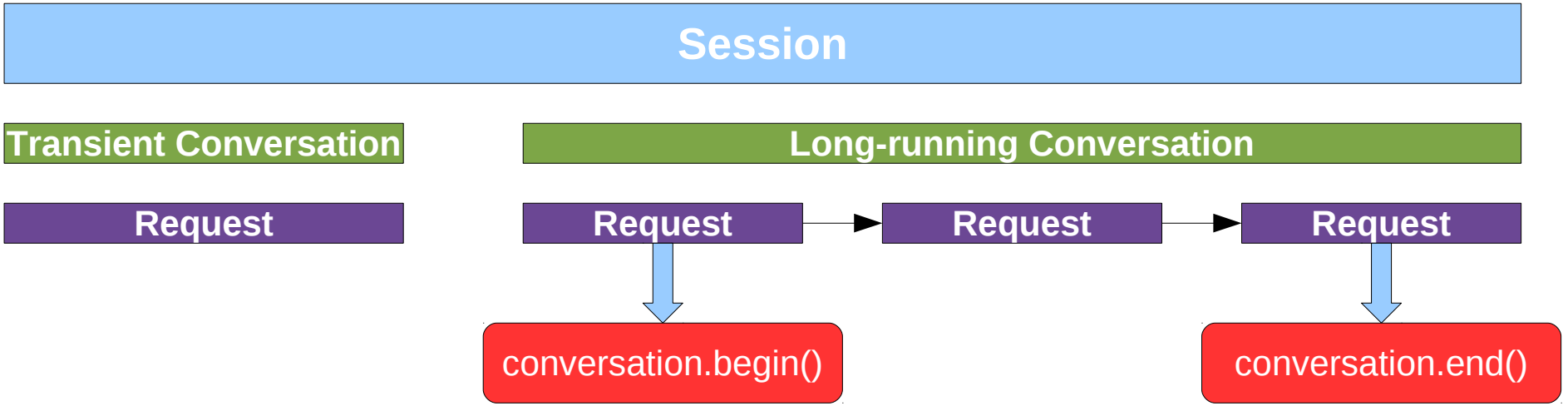
- Uchováva stav sekvencie po sebe idúcich logicky súvisiacich requestov
- Samotná konverzácia má 2 stavy
  - **Transient** – jeden JSF request
  - **Long-running** – sekvencia JSF requestov



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();
}
```

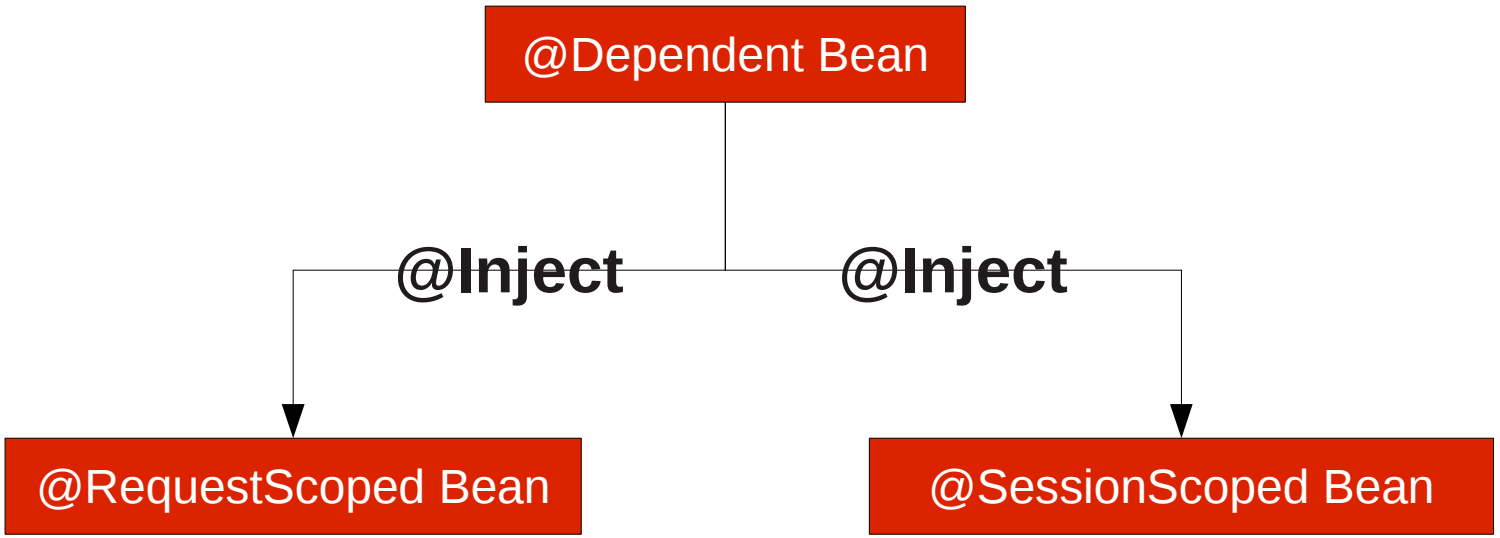
redhat

Propagácia konverzácie

- String identifier
  - Set by application
  - Generated by container
- Vrámci session
- Propagácia pomocou **cid** parametru
- Automatická propagácia pri odosielaní JSF formuláru

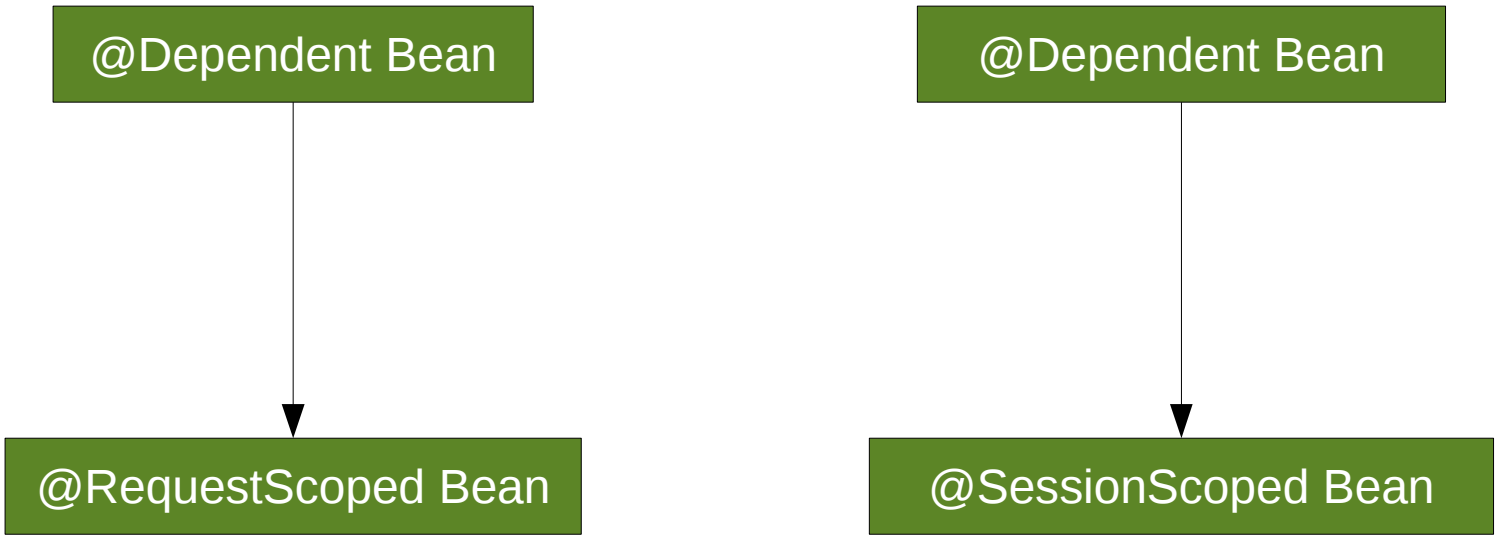
Dependent Pseudo Scope

- 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

```
public class PaymentManager {

    @Inject
    private PaymentProcessor pp;
```

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

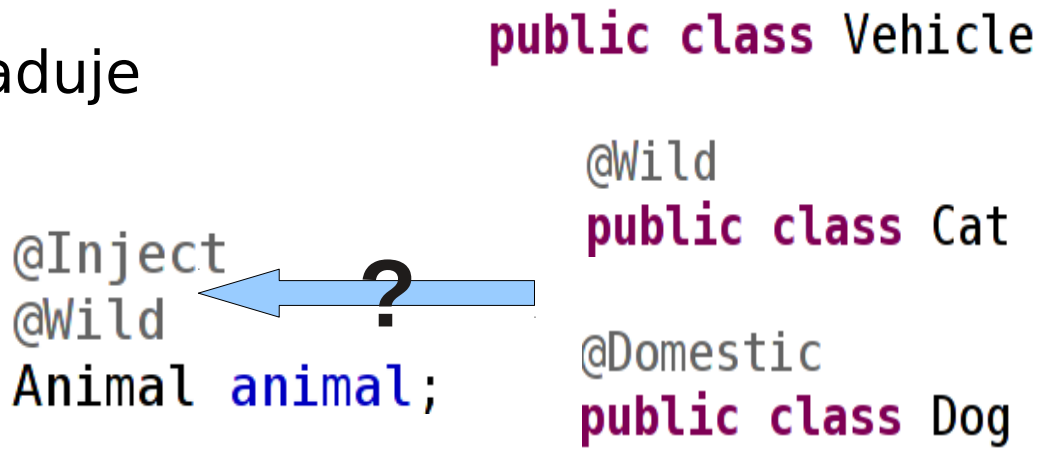
```
@Qualifier
@Target({ TYPE, METHOD, PARAMETER, FIELD })
@Retention(RUNTIME)
public @interface LoggedIn {
}
```

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

redhat

Pravidlá DI – Typesafe resolution

- Injection point Beanu 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



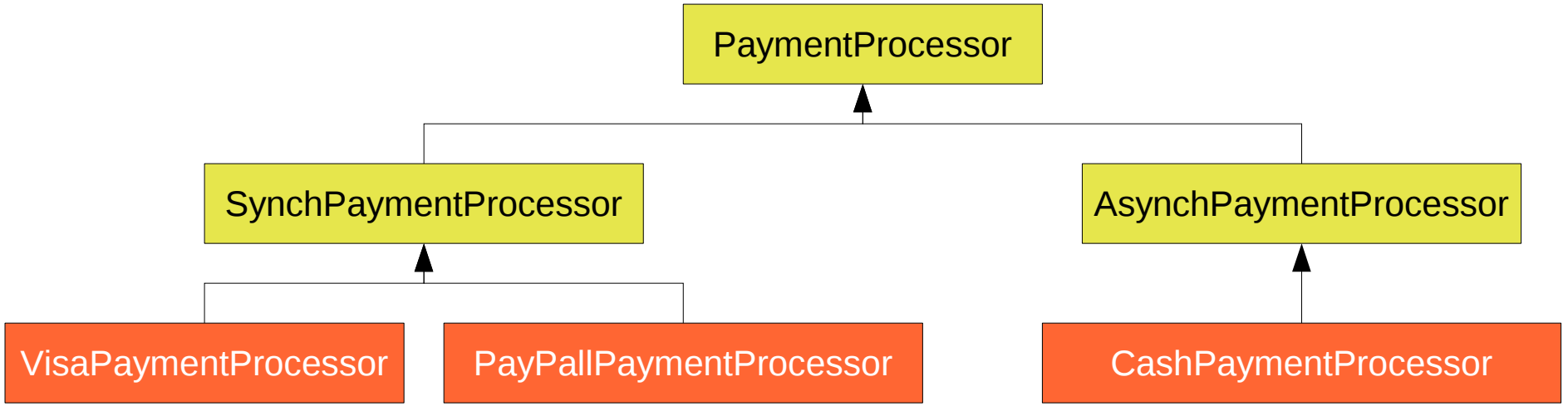
redhat

Poskytované typy

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

redhat

Príklad typovej rezolúcie (typy)

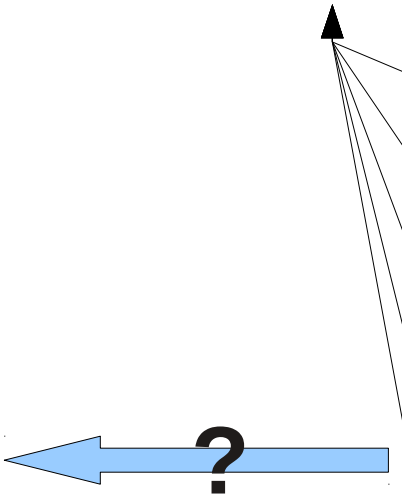



```
@Inject
private PaymentProcessor pp;

@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

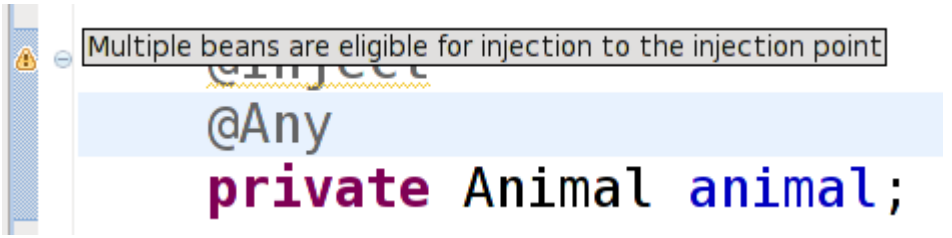
```
public class Shelter {

    @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

- Pre prípady keď Java 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) v component descriptore (beans.xml)
- 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

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

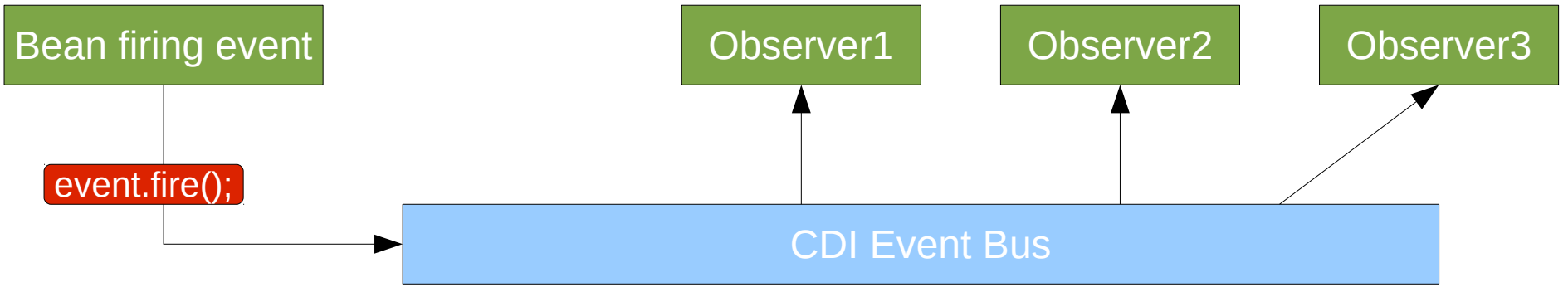
Použitie DI v non-CDI beans

- Servlet / Filter / ServletListener
- Message-Driven bean
- JAX-RS resources / providers
- JAX-WS endpoints

redhat

Udalosti (Events)

- Založené na návrhovom vzore “Observer”
- Oddelenie producentov udalostí od ich konzumentov
- Loose coupling – ďalšie observery sa môžu pridať neskôr bez zásahu do producenta udalostí
- Synchrónne
- Eventy môžu niesť “náklad”



redhat

Udalosti (Events) – Vyvolanie udalosti

- Vstavaný Event bean

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

- Vyvolanie udalosti

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

redhat

Udalosti (Events) – Odchytenie udalosti

- Observer method
- Pravidlá typovej rezolúcie rovnaké ako pre DI

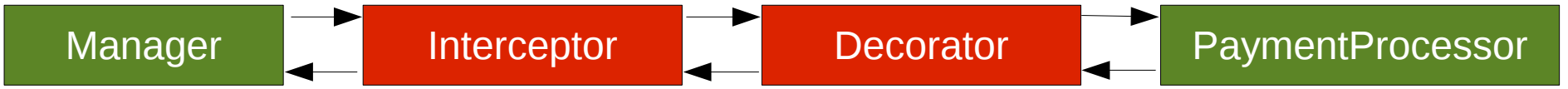
```
public void log(@Observes User user) {
    System.out.println("Hello " + user.getName());
}
```

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

redhat

Dekorátory a Interceptory

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



redhat

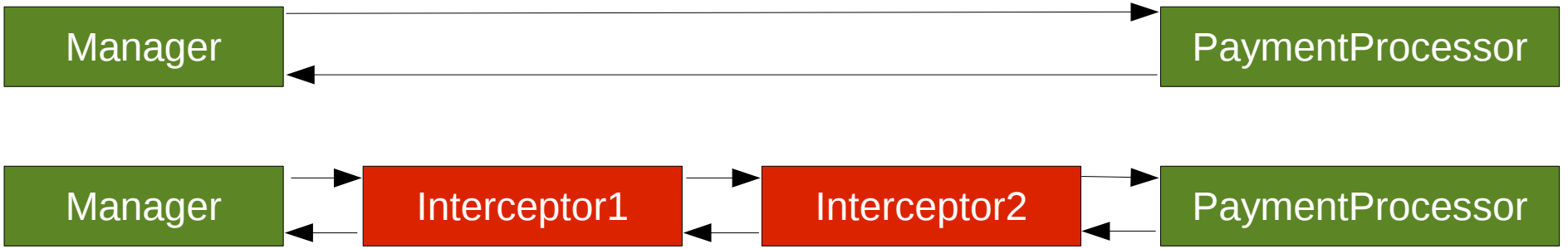
Interceptory

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

redhat

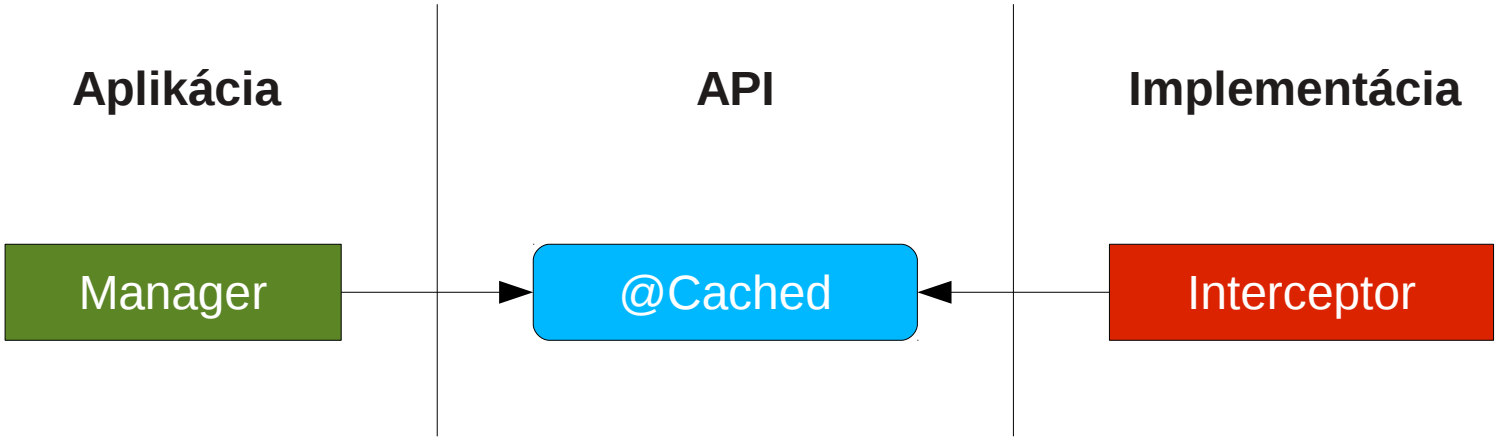
Interceptory

```
public class PaymentManager {

    @Inject
    private PaymentProcessor pp;

    pp.foo();
```



Interceptor binding

- Anotácia reprezentujúca funkcionalitu poskytovanú interceptorom (loose coupling)



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)

```java
@Decorator
public abstract class LargeTransactionDecorator implements Account {

    @Inject
    @Delegate
    private Account account;
    @Inject
    @LoggedIn
    private User user;
    @Inject
    @SecurityLimit
    private BigDecimal limit;

    public void withdraw(BigDecimal amount) {
        // security check
        if (amount.compareTo(limit) > 0) {
            if (!user.isAuthorizedForLargeTransactions()) {
                throw new SecurityException("Amount too high.");
            }
        }
        // proceed
        account.withdraw(amount);
    }
}
```

redhat

Aktivácia a poradie

```
<beans>
    <decorators>
        <class>cz.muni.fi.pv243.FooDecorator</class>
        <class>cz.muni.fi.pv243.BarDecorator</class>
    </decorators>
    <interceptors>
       <class>cz.muni.fi.pv243.FooInterceptor</class>
    </interceptors>
</beans>
```

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

Contexts and Dependency Injection

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

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

jharting@redhat.com | www.redhat.com