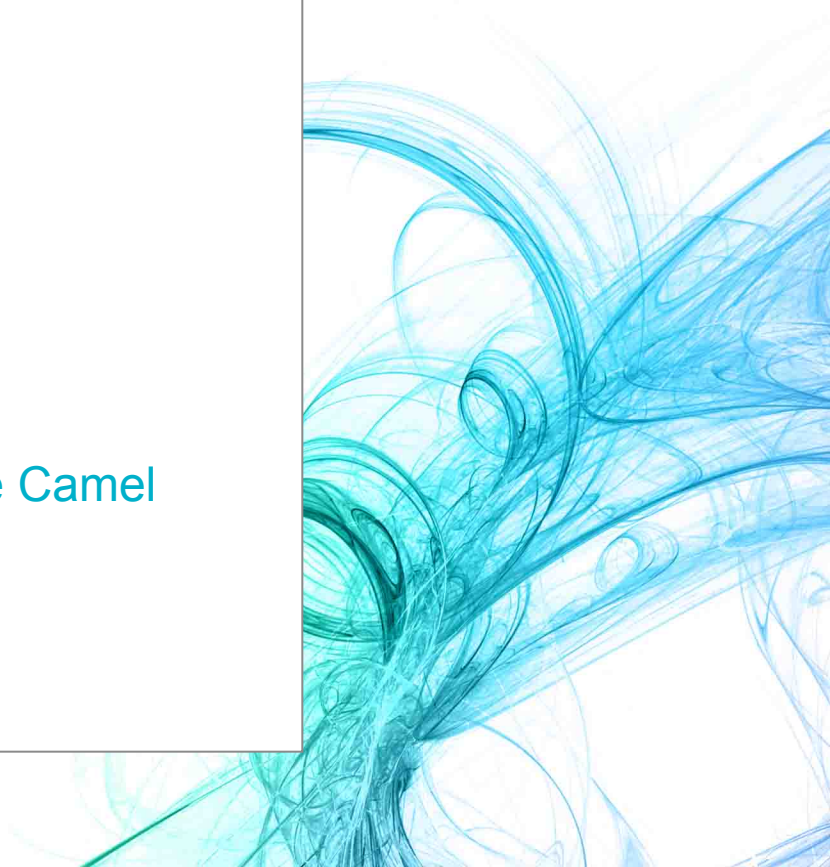
indra

Apache Camel

Úvod do EIP a použití frameworku Apache Camel
Prezentace pro CZJUG



Témata

- Úvod do problému
- EIP, nástroje, které se používají
- Apache Camel
  - Architektura
  - Spring
  - Testování – Junit
  - Optimalizace
- Postřehy

Kdy mluvíme o integraci aplikací?

- Aplikace na sobě nezávisí
- Každá aplikace se zaměřuje na specifickou funkcionalitu
- Delegace činností mezi aplikacemi
- Komunikace je asynchronní, aplikace nečekají na odpověď
- Než je k dispozici odpověď, pracuje aplikace na dalších požadavcích

Příklad: “Messagingové řešení”

Pomocí čeho integrovat?

- Přenos souborů
- Sdílená databáze
- RMI – vzdálené volání procedur
- Messaging

Messaging – některé pojmy z EIP

- Channels – kanály zpráv (fronta, téma)
- Messages – zprávy
- Pipes and Filters – trubky a filtry
- Routing – směrování
- Transformation – transformace
- Endpoints – napojení na externí zdroje, protokoly

EIP

- Enterprise Integration Patterns
- Hlavní úkol:
  - Sjednocení terminologie
  - Popis jednotlivých částí integračního řešení, které se často opakují
  - Pomoc při návrhu
  - Vzorů je hodně, nemá smysl je procházet…

Nástroje k implementaci EIP

- JMS/Java
  - Apache Camel – podpora Springu, příznivá křivka učení
  - Spring Integration – těží z rozšíření a funkčnosti Springu
- MSMQ/.NET
- TIBCO
- MS BizTalk

Camel Framework

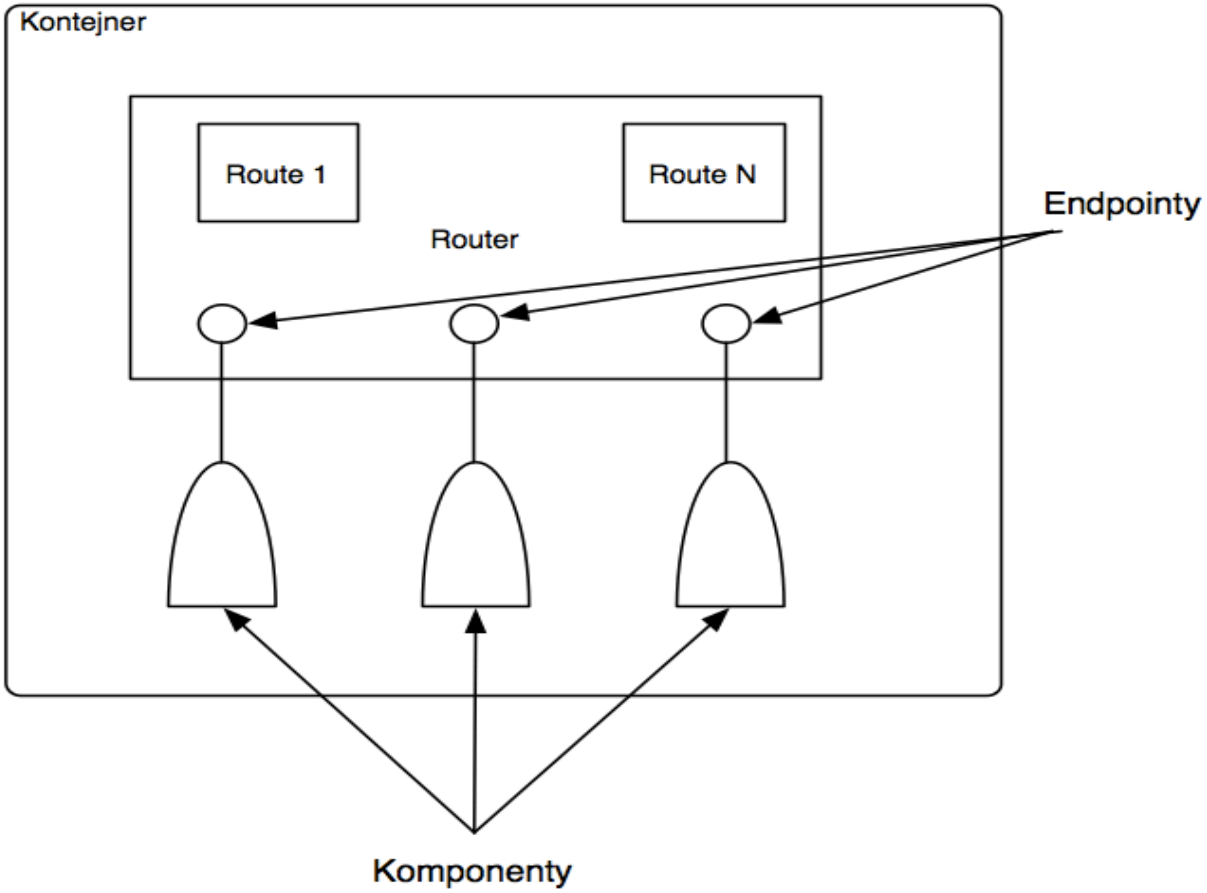
- Režie frameworku v použitém řešení je velmi malá
- Nezávislost na protokolu/zdroji
- Implementován velký počet integračních vzorů
- Velmi dobře zdokumentovaný
- Velmi jednoduše rozšiřitelný
- Relativně jednoduchý – příznivá křivka učení

Camel Framework

Architektura:

- Komponenta
  - Integruje protokol nebo externí zdroj
  - Programátorsky: factory pro určitý typ endpointu
- Endpoint – produkuje/konzumuje data
- Router – obaluje pravidla pro směrování zpráv, endpointy

Camel Framework



Camel deployment

- Možnosti nasazení:
  - Spring IoC kontejner
  - Samostatně
  - OSGi kontejner

Camel a Spring

- Transakce
- XML konfigurace
- Bean Integration
- Dependency Injection – camel context/spring beany
- JMS komponenta – postavena na JMS Template

Camel komponenty

- Integrují do jádra routeru určitý protokol nebo zdroj
- Z pohledu programátora – factory pro vytvoření endpointu
- Camel obsahuje mnoho hotových komponent
  - JMS komponenta (messaging)
  - CXF komponenta (web services)
  - File komponenta (soubory, adresáře)
  - Bean komponenta (beany nebo POJO)
  - Mock komponenta (testování)
  - a další…

Camel endpoint

- Vytvářeny komponentami
- Konverze vstupu na vnitřní formát frameworku
- Konverze vnitřního formátu frameworku na výstupní
- Odkazuje se na ně pomocí URI, charakteristický prefix
  - jms – JMS koponenta (JMS messaging)
  - cxf – CXF komponenta (web services)
  - file – komponenta pro práci se soubory
  - mock – testování

Camel endpoint

- Varianta definice endpointu v xml
- Odkaz na endpoint pomocí ref=id

```
<camel:endpoint id="input” uri="jms:input@z3smq_3001" />
```

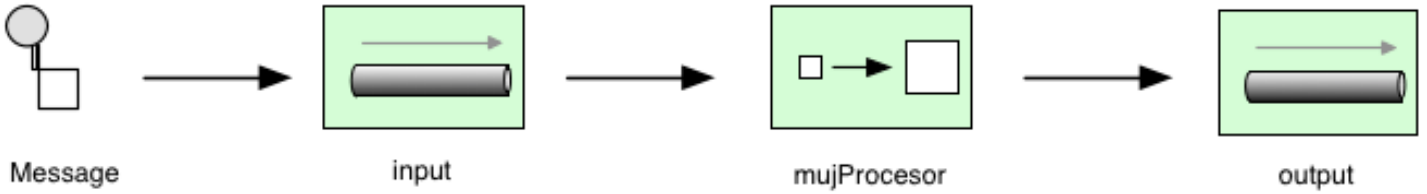
Camel router

- Router je reprezentován CamelContext (kontext) objektem
- Kontext obsahuje:
  - Definice chování routeru
  - Definice směrování zpráv
  - Definice endpointů
- Kontext může být definován pomocí
  - XML
  - Java kódu
  - XML a Java kódu

Camel router – XML konifgurace

- Příklad kontextu definovaného pomocí XML:

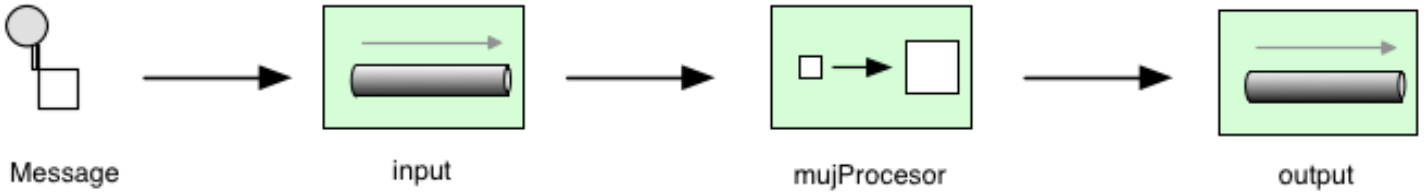
```
<camel:camelContext id="camel”>
        <camel:route>
                <camel:from uri="jms:input@z3smq_3001" />
                <camel:bean ref=”MujProcesor" />
                <camel:to uri="jms:output@z3smq_3001" />
        </camel:route>
</camel:camelContext>
```



Camel router – konfigurace pomocí Java kódu

- Příklad cesty definované pomocí Java kódu

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
       from("jms:input@z3smq_3001")
              .bean(mujProcesor)
              .to("jms:output@z3smq_3001");
    }
};
```



Camel procesor

```
@Override
public void process(Exchange exchange) throws Exception {
        Message message = exchange.getIn();
        MRPayload mrPayload = (MRPayload) message.getBody();
        BAPayload baPayload = new BAPayload();
        ...
        message.setBody(baPayload);
        exchange.setOut(message);

        if (exchange.isFailed()) {
                Exception exception = exchange.getException();
                logger.error(exception);
        }
}
```

Camel – směrování zpráv

- Filtrování – filtr zahodí zprávu, která nevyhovuje podmínce

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
        from(”jms:input")
            .filter(header("foo").isEqualTo("bar"))
                .to(”jms:output");
    }
};
```

Camel – směrování zpráv

- Podmínky

```
RouteBuilder builder = new RouteBuilder() {
    public void configure() {
        ((ChoiceDefinition) from(”jms:input")
                .choice()
                .when(header("mode").isEqualTo("NORMAL"))
                        .to(jms:normalChannel))
                .when(header("mode").isEqualTo("PRIORITY"))
                        .to(jms:priorityChannel);
    }
};
```

Camel Exception Handling

- Možnost definovat, co se stane při výskytu výjimky
- Příjemce zprávy, která způsobila výjimku, může být procesor i fronta
- Lze zakázat znovuposlání zprávy (redelivery policy)
- Znovuposlání začíná tam, kde se stala výjimka (zpráva neputuje od začátku cesty)
- Použít lze jak XML deklaraci, tak i Java kód

Camel Exception Handling

- Příklad definice v XML :

```
<camel:onException>
        <camel:exception>java.lang.Exception</camel:exception>
        <camel:redeliveryPolicy disableRedelivery="true" />
        <camel:process ref="exceptionHandler" />
        <camel:rollback markRollbackOnly="true" />
</camel:onException>
```

- Příklad definice v Java kódu

```
onException(IOException.class).maximumRedeliveries(3);
```

Testování

- Vyplatí se použít Spring
- Deklarace

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations =
        "classpath:testErrorHandlerContext.xml”)
public class RedeliveryProcessorTest {
. . .
```

- Definice proměnných

```
@EndpointInject(ref = "delayer”)
protected MockEndpoint delayer;
```

Testování

```
@Test
public void testMsgProcessed () throws Exception {
        delayer.reset();
        Message msg = new DefaultMessage();
        Exchange exchange = delayer
                    .createExchange(ExchangePattern.InOnly);
        exchange.setIn(msg);
        redeliveryProcessor.process(exchange);
        delayer.expectedMessageCount(1);
        delayer.assertIsSatisfied();
}
```

Camel – možnosti optimalizace

- Agregace
  - Přenos informací je “drahý”, informace se seskupí a přenáší v “balících”
  - Poměr cena/výkon – velké zrychlení chodu aplikace
  - Dvě varianty:
    - **Bez perzistence**
    - **S perzistencí**
  - Nevýhoda – nefunguje dobře s XA transakcemi

Camel – možnosti optimalizace

- Zpracování ve více vláknech

```
Processor checker = (Processor) (new BACheckerMR());

from("jms:input@z3smq_3001”)
        .process(checker)
        .threads(5)
        .to(jms:output@z3smq_3001);
```

Camel – možnosti optimalizace

- Load balancing – distribuce zátěže na více endpointů
  - Round Robin
  - Random
  - Sticky (podle jsessionId, JMSXGroupId…)
  - Failover
  - Weighted Round Robin
  - Weighted Random

Camel – možnosti optimalizace

- Příklad definice v XML – rozhazuje zprávy do dvou procesorů

```
<camel:multicast>
        <camel:loadBalance>
                <camel:roundRobin />
                <camel:process ref="journalingCore1" />
                <camel:process ref="journalingCore2" />
        </camel:loadBalance>
</camel:multicast>
```

Postřehy

- Procesor – metoda process() – I když běží ve vláknech, instance je jen jedna – pozor na proměnné
- Agregátor – XA transakce – nefunguje úplně jak by měl
- InOnly typ JMS endpointu – rychlost zapisování, obsah je stejný objekt
- JBOSS – bez jboss-camel knihovny nefunguje správně, classloader resolving
- JBOSS – transakční manažer/XA transakce a JMS
- Spring backend aplikace – použití RAR

Odkazy na další informace

- http://www.enterpriseintegrationpatterns.com/
- http://camel.apache.org/
- http://fusesource.com

Martin Polovinčák
Telco
mpolovincak@indracompany.com

INDRA Czech Repuiblic s.r.o.
Karolinská 1
186 00, Prague
Czech Republic
T +420 246 085 700
F +420 246 085 701
www.indra.cz