72 Services - The Software Renovators

72 Services LLC
2028 E Ben White Blvd #240-7272
Austin TX 78741

Phone 512-333-4412
Email info@72.services
Web 72.services
About me

1995
1995
COBOL
IBM Mainframe

2000
2000
Java

2007
2007
Adjunct Professor
Tech Univ Bern

2009
2009
simas GmbH

2011
2011
Board JUG.CH
Leader Bern

2012
2012
JSR-352
Java Batch

2013
2013
JSR-354
Java
Money and Currency

2017
2017
72 Services LLC

Twitter @simas_ch | Mail simon@72.services
About the Customer

Cash & Carry

Wholesale Supplies
Architecture Before

3rd Party

Business Logic

Data

SYNC

SAP

Business Logic

Data

SYNC

Fat Client

Business Logic

Data

SYNC

Web Shop

ERP

Customer Center
Multi Channel Architecture

Web Shop | Field Staff | ERP | Truck Driver | Customer Center

ProlixPro

Business Logic

Data

Common business logic and data for all channels

ProlixPro is based on a service oriented architecture and uses Java EE standard and open source components
Major Concerns

• Requirements
• Performance
• Integration
• Security
• Distribution
Requirements
Replacing an Existing System
Minimal Viable Product (MVP)

Problem
• End users need to see the product to talk about requirements

What helps
• Involve end users from beginning
• Release often to get immediate feedback
The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; premature optimization is the root of all evil in programming. Donald Knuth, 1974

Performance

Data Access
Simplified Model

Sample code: https://github.com/simasch/hpa
How to Find Performance Issues?

• **Load Testing**
  • **Open Source Tools**
    • Gatling - [http://gatling.io/](http://gatling.io/)
    • JMeter - [http://jmeter.apache.org/](http://jmeter.apache.org/)
  • **Do it yourself**
    • Integrate load generator in Client application

• **Turn SQL logging on**
  • Hibernate: `<logger name="org.hibernate.SQL" level="debug"/>`
  • Log4jdbc - [http://log4jdbc.brunorozendo.com/](http://log4jdbc.brunorozendo.com/)
  • Datasource-proxy: [https://github.com/ttddyy/datasource-proxy](https://github.com/ttddyy/datasource-proxy)
**N+1 Select Problem**

- **Mapping** of Orders und OrderItems
  Default: `FetchType.LAZY`

  ```java
  Customer
  @OneToMany(mappedBy="customer")
  private Set<Order> orders;

  Order
  @OneToMany(mappedBy="order")
  private List<OrderItem> items;
  ```

- **Result**
  - 1 query to get all Customers
  - 1 query per Customer to get n Orders
  - 1 query per Order to get n OrderItems
Solutions

• FetchType.**EAGER**
  • Caution: Only a hint for JPA
  • != SQL JOIN
  • Hibernate allows only one List with FetchType.EAGER per Entity

    org.hibernate.HibernateException: cannot simultaneously fetch multiple bags

• JPA **JOIN FETCH**
  • Be aware of Cartesian product when join fetching more than one collection!

• **EntityGraph** (JPA 2.1)
DTO to the Rescue

https://martinfowler.com/eaaCatalog/dataTransferObject.html
public class CustomerInfoDTO {

    private final Long id;
    private final String lastname;
    private final String firstname;
    private final double revenue;

    public CustomerInfoDTO(Long id, String lastname, String firstname, double revenue) {
        this.id = id;
        this.lastname = lastname;
        this.firstname = firstname;
        this.revenue = revenue;
    }

    // getters

}
JPA Constructor Expression

Query q = em.createQuery("SELECT NEW jpa.customer.entity.CustomerInfoDTO(
c.id, c.lastname, c.firstname, SUM(i.product.price))
FROM Customer c
JOIN c.orders o
JOIN o.items i
WHERE lower(c.lastname) LIKE lower(:term)
GROUP BY c.id, c.lastname, c.firstname
ORDER BY c.lastname, c.firstname
");

List<CustomerInfoDTO> list = q.getResultList();
Entity Model != Data Model

Customer
- id
- lastname
- firstname

Order
- 0..*

OrderItem
- *

Product
- price

Do we really need this relationship?
But what about the JPA Query?

Query q = em.createQuery("SELECT NEW jpa.customer.entity.CustomerInfoDTO(
c.id, c.lastname, c.firstname, SUM(i.product.price))
FROM Customer c
JOIN c.orders o on o.customerId = c.id
JOIN o.items i
WHERE lower(c.lastname) LIKE lower(:term)
GROUP BY c.id, c.lastname, c.firstname
ORDER BY c.lastname, c.firstname ");

List<CustomerInfoDTO> list = q.getResultList();
Is O/R Mapping Dangerous?

• Write O/R Mapping code like you would write SQL!

• Beware of EntityManager.find()

• Just because you’re using Hibernate, doesn’t mean you have to use it for everything. A point I’ve been making for about ten years now.
  
  *Gavin King, “Father of Hibernate”, 12/10/2013*
SQL

**PRO**
- Declarative and Expressive
- Easy to test with SQL tools/IDE
- Easy to optimization

**But wait!**

**SQL with Java?**
**Plain JDBC?**
JPA 2.1 Constructor Result

Query q = em.createNativeQuery("SELECT C.ID AS ID,
   C.LASTNAME AS LASTNAME,
   C.FIRSTNAME AS FIRSTNAME,
   SUM(P.PRICE) AS REVENUE
FROM CUSTOMERS C
JOIN ORDERS O ON O.CUSTOMER_ID = C.ID
JOIN ORDERITEMS I ON I.ORDER_ID = O.ID
JOIN PRODUCTS P ON P.ID = I.PRODUCT_ID
WHERE LOWER(C.LASTNAME) LIKE LOWER(?)
GROUP BY C.ID, C.LASTNAME, C.FIRSTNAME
ORDER BY C.LASTNAME, C.FIRSTNAME
", "CustomerInfoDTO");

@SqlResultSetMapping(name="CustomerInfoDTO",
classes={
   @ConstructorResult(
      targetClass=CustomerInfoDTO.class,
      columns={
         @ColumnResult(name="ID"),
         @ColumnResult(name="LASTNAME"),
         @ColumnResult(name="FIRSTNAME"),
         @ColumnResult(name="REVENUE"))
   })
Query Language Result Mapper - qlrm.org

Query q = em.createNativeQuery("SELECT C.ID, C.LASTNAME, C.FIRSTNAME, SUM(P.PRICE)
FROM CUSTOMERS C
    JOIN ORDERS O ON O.CUSTOMER_ID = C.ID
    JOIN ORDERITEMS I ON I.ORDER_ID = O.ID
    JOIN PRODUCTS P ON P.ID = I.PRODUCT_ID
WHERE LOWER(C.LASTNAME) LIKE LOWER(?)
GROUP BY C.ID, C.LASTNAME, C.FIRSTNAME
ORDER BY C.LASTNAME, C.FIRSTNAME
");
JpaResultMapper mapper = new JpaResultMapper();
List<CustomerInfoDTO> list = jpaResultMapper.list(q, CustomerInfoDTO.class);
List<CustomerInfoDTO> list = create.

    select(CUSTOMERS.ID, CUSTOMERS.LASTNAME, CUSTOMERS.FIRSTNAME, sum(PRODUCTS.PRICE)).
    from(CUSTOMERS).
    join(ORDERS).on(ORDERS.CUSTOMER_ID.eq(CUSTOMERS.ID)).
    join(ORDERITEMS).on(ORDERITEMS.ORDER_ID.eq(ORDERS.ID)).
    join(PRODUCTS).on(PRODUCTS.ID.eq(ORDERITEMS.PRODUCT_ID)).

where(CUSTOMERS.LASTNAME.like(term + "\%")).

groupBy(CUSTOMERS.ID, CUSTOMERS.LASTNAME, CUSTOMERS.FIRSTNAME).

orderBy(CUSTOMERS.LASTNAME, CUSTOMERS.FIRSTNAME).

fetchInto(CustomerInfoDTO.class);
Single Model

Source: https://martinfowler.com/bliki/CQRS.html
CQRS

query model reads from database

command model updates database

query services update presentations from query model

command model executes validations, and consequential logic

Service Interfaces

user makes a change in the UI

application routes change information to command model

Source: https://martinfowler.com/bliki/CQRS.html
CQRS in Small

• **Command**
  • JPA Entities

• **Query**
  • JPA Constructor Expression
  • JPA ConstructorResult
  • QLRM
  • jOOQ
Pre-Calculated Data

• **Search** Engine

• **Materialized** Views

• **Prefilled** Tables

• Add **Columns** to hold pre-calculated data

```java
@Access(AccessType.PROPERTY)
public double getTotalAmount() {
    return items.stream()
        .mapToDouble(item ->
            item.getProduct().getPrice()
        ).sum();
}
```
Apache Solr

Pro
• Java API
• SQL Data Import

Goodies
• Facets
• Suggest
• Spell checking
• Boosting
• Language Support
**Conclusion**

- **Entities** just for **data manipulation**

- **DTO** for **read** access

- Turn **SQL logging** on during development!
Performance
Response Time
Price Calculation

Problem

• Custom price per product and customer
  • Calculation of price: ~ 300 ms
  • Results on page: 20
  • Total processing time: ~ 6 sec

Solution Parallelization

• Caution: Leads to 20 times more requests and much more load on the server!
• But - where to parallelize?
Parallelization

- **Client-side**
  - Control on clients
  - Network latency

- **Server-side**
  - No network latency
  - App server tuning

```
@Stateless
public class PriceCalculator {

@Asynchronous
public Future<Price> calculatePrice(
  Customer c, Product p) {

  Price price = ...
  return newAsyncResult<Price>(price);
}
}
```
Integration
Requirements

- ~80 Interfaces to
  - Customers
  - Suppliers
  - 3rd Party Systems (Transport, Logistics, Finance)

- Integration code **should not contain business logic!**
Apache Camel

• Implementation of Enterprise Integration Patterns
  http://www.enterpriseintegrationpatterns.com/

• Camel empowers you to define routing and mediation rules in a variety of domain-specific languages
Integration Platform
public class EdiD01bAperakToSupplierProvisionConfirmationRouteTest extends CamelTestSupport {
    @Test
    public void testTransformEdiD01bAperakToSupplierProvisionConfirmation() throws Exception {
        String expectedResult = CamelTestHelper.getFileAsString(getClass(), filePathIncoming);
        mockOut.expectedBodiesReceived(expectedResult);
        template.sendBody(CamelTestHelper.DIRECT_IN, CamelTestHelper.getRemoteFileFromClasspathResource(getClass(), filePathIncoming));
        assertMockEndpointsSatisfied();
    }
}
Batch Processing

• **Requirements**
  • Scheduling
  • Restartable
  • Parallel Execution

• **Is Camel the right tool for this job?**
JSR-352

- Spring Batch with **JSR-352**
Security
Concern - Expose ERP system to the Internet

Internet

Intranet

ProlixPro

Integration

- Customers
- Suppliers
- Logistics
- Transport

Sales Office

Field Staff

Customer

Truck Driver
Web Application Firewall (WAF)

- Internet
- DMZ
- Intranet
  - ERP
  - Sales Office
  - Integration
    - Customers
    - Suppliers
    - Logistics
    - Transport
  - CC
- Customer
- Field Staff
- Truck Driver
Distribution
Runtime

- Rich Client Eclipse RCP
- Web Shop ReactJS
- Mobile Shop Andorid
- Delivery App Android

JRNJP

REST/JSON

Apache Solr
- JBoss AS

Oracle DB

ActiveMQ

Spring Batch

Camel WildFly
Centralized Logging

- JBoss AS
- WildFly
- Apache Solr
- ActiveMQ
- Spring Batch

Filebeat

Logstash

Elasticsearch

Kibana

Support
Logging in JavaScript Client?

**Problem**
- WAF may block requests
- Client catches Exception

**Solution**
- Logging to a cloud provider
Configuration

- **Centralized** Database
- **Default** and **Staging**

```java
@Inject
@ConfigProperty("compass.route.from")
private String compassRouteFrom;
```
Conclusion
Lessons Learned

• **Workshops** are the best way to get first requirements but then immediately start developing (MVP)

• Do **load testing** from beginning

• Turn on **SQL logging** during development

• **Protect** your system with Web Application Firewall

• **Separate** integration and batch processing from online system

• **Centralized** logging and configuration for distributed systems
Thank you