Von µServices zu Microservices

20 Jahre Evolution eines OpenSource Projekts

Heiko Scherrer, JUGS.ch, 05. Mai 2022

Heiko Scherrer

Interface21.io

spring

selbstständiger Berater, Engineer, Coach & Dozent
OCEA, iSAQB & TOGAF
hscherrer@interface21.io
@openwms

open logistics foundation

Expertise
- seit 20 Jahren in unterschiedlichen Rollen und Branchen innerhalb der **Software Produkt & Projekt Entwicklung** tätig
- seit 17 Jahren mit der Lagerverwaltung **OpenWMS.org** unterwegs

Themengebiete
- Cloud Computing - insbesondere **Cloud Transformation**
- Moderne IT Architekturen - **Microservices**
- **Innovative Lösungen** - Assessment, Architektur, Development
- Seit 2007 bin ich überzeugter SpringFramework Anwender

Fokus auf
- Spring Ökosystem
- API Security
- Cloud Architekturen

Abstract

Interessierte Teilnehmer erfahren in dieser Session wie ein OpenSource Projekt von einer anfänglichen monolithischen Architektur über technische **µServices** bis hin zu einer modernen heterogenen **Microservice** Architektur gereift ist. Neben den Beweggründen und **Anforderungen** werden Architektur- und Entwurfentscheide aufgezeigt, ebenso werden die verwendeten **Frameworks**, **Tools** und **Patterns** vorgestellt die für den reibungslosen Betrieb der Software in Produktion sorgen. Garantiert kommen auch die Themen wie **Security**, **Testing** und automatisierte **Dokumentation** nicht zu kurz. Alles in allem ein praktischer Erfahrungsbericht aus mehr als sechs Jahren Einsatz der Microservice Architektur der anhand von Schaubildern, Code und einer Demo hoffentlich wertvollen Input für eigene Projekte liefert.

Angesprochene Frameworks (Auszug): **Spring Boot, Spring Cloud, Spring Dynamic Modules, OSGi, OAuth2, Cloud Provider, Docker**

Agenda

Business Domain Intralogistic

Context and Environment

Systemrequirements

History

Microservice Architecture & Design

What’s next? SaaS

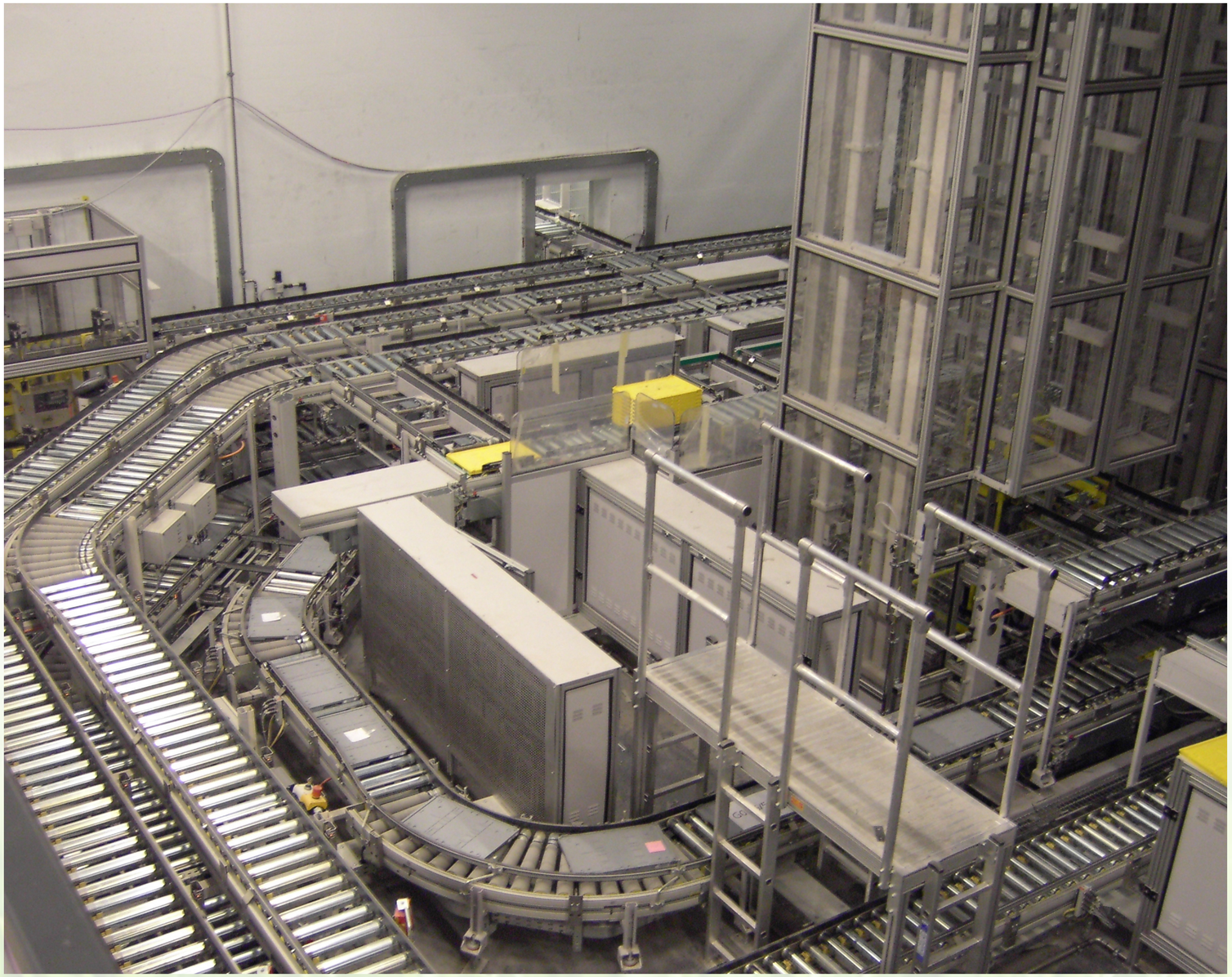
“Als **Intralogistik** *bezeichnet man die logistischen* **Material- und Warenflüsse**, *die sich* **innerhalb** *eines Betriebsgeländes abspielen.”*

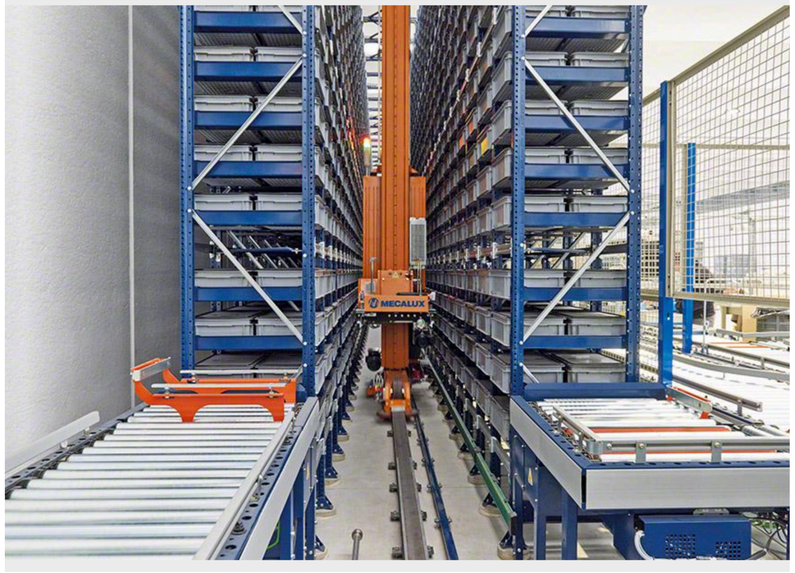
Wikipedia

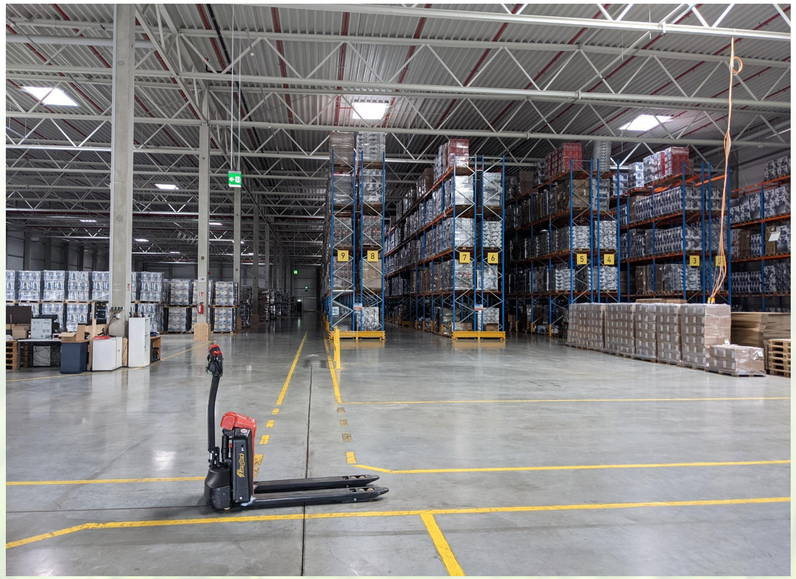
Businessdomain Intralogistic

Internal Movements of Goods

- Goods are moved between Locations
- Often a carrier (TransportUnit) is used for that
- Types of Warehouses:
  - Automatic
  - Manual
  - Mixed

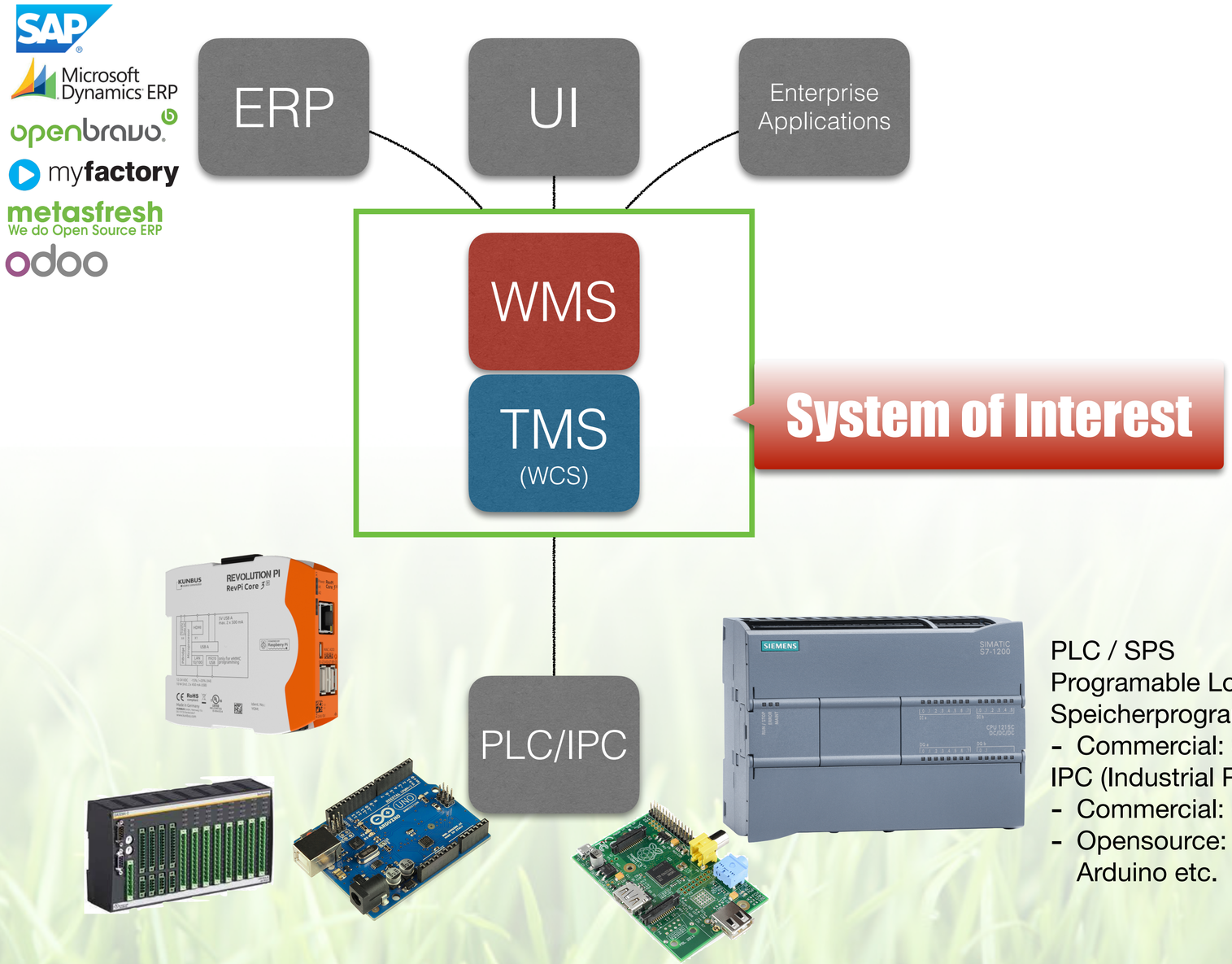






Context & Environment

Business

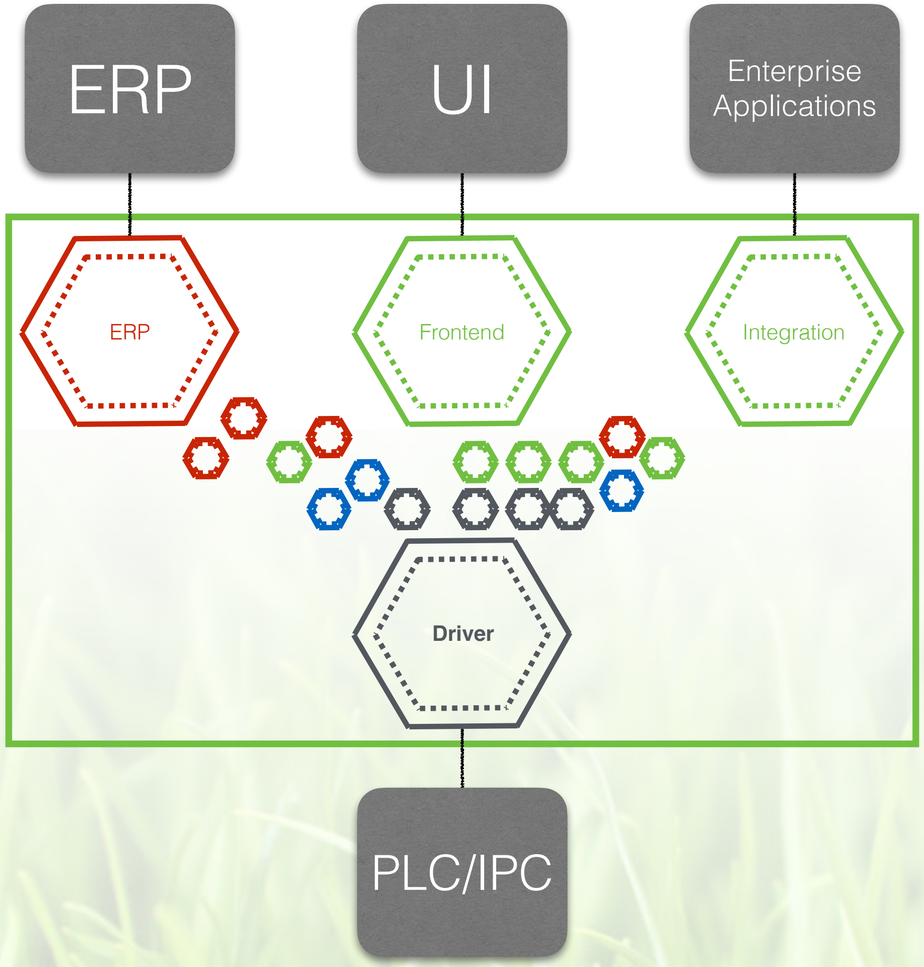


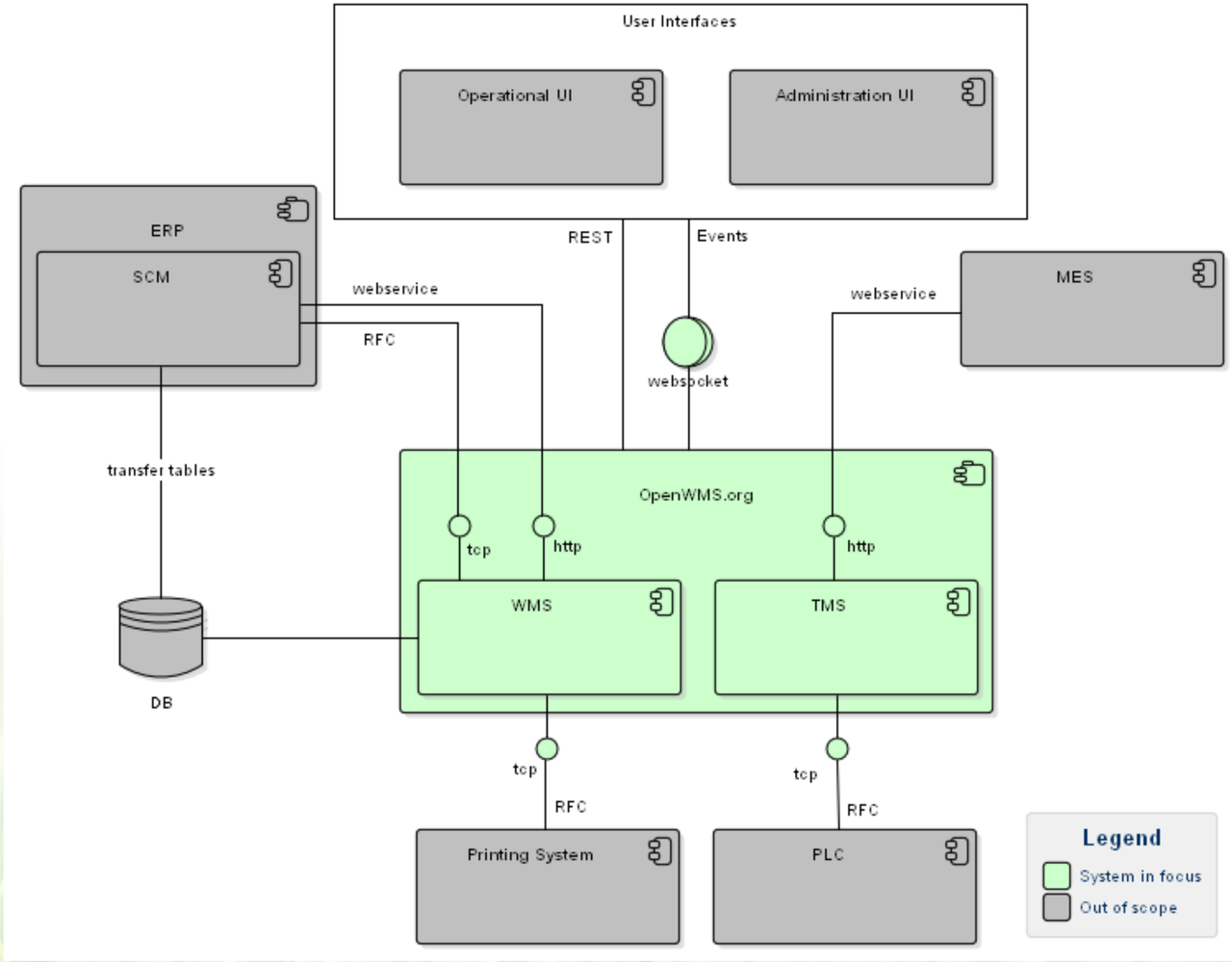
PLC / SPS
Programable Logic Controller /
Speicherprogrammierbare Steuerung
- Commercial: Siemens, Allen Bradley, Fanuc

IPC (Industrial PC)
- Commercial: Bachmann, Beckhoff
- Opensource: Kunbus RevPi, Raspberry Pi, Arduino etc.

Context & Environment

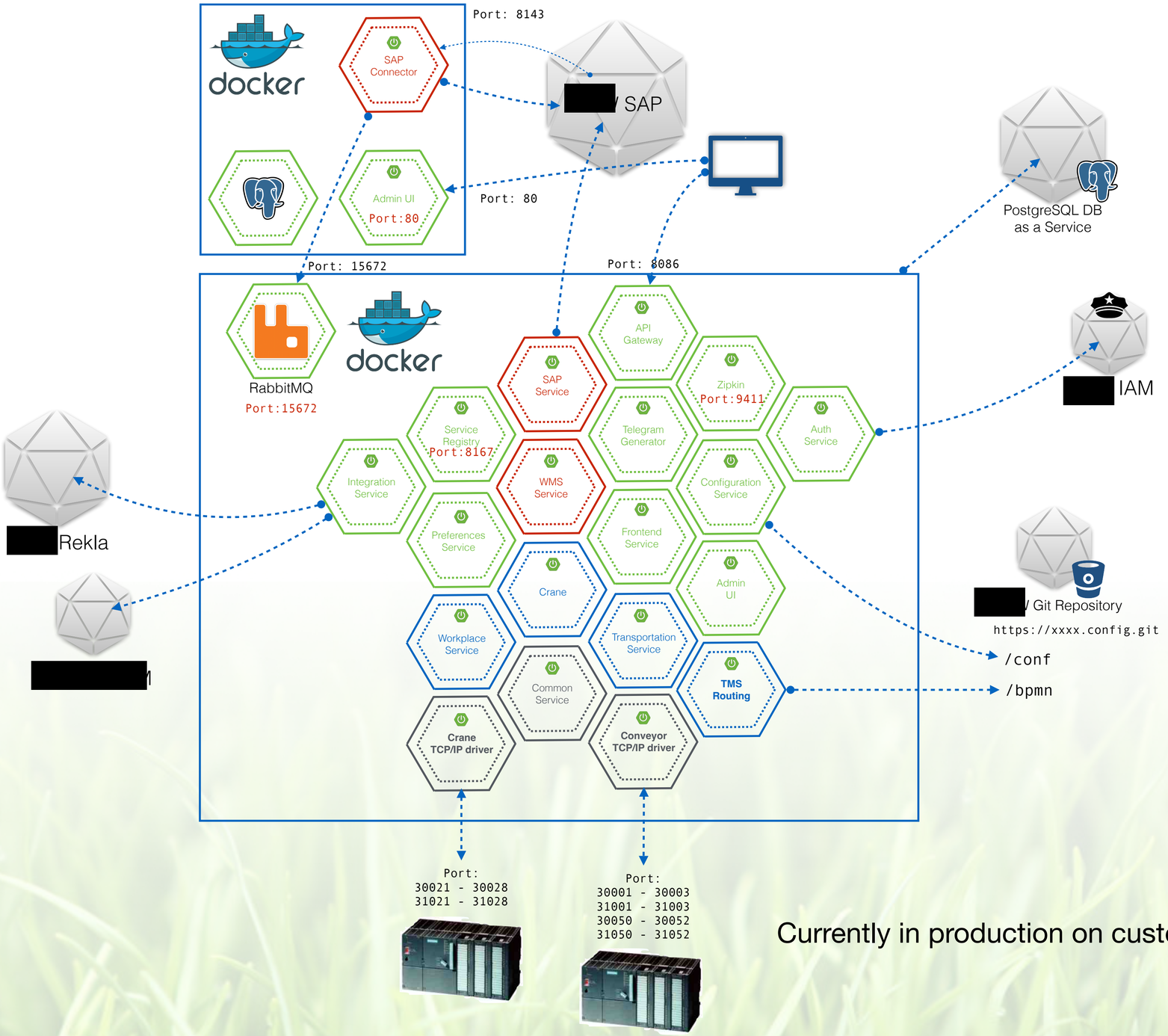
Technical





Context & Environment

System Context



Currently in production on customer side

Systemrequirements

Relevant Subsystems and their Requirements

- Subsystems: ERP, PLC, UI, other Business Applications
- Projectspecific Interfaces and Technologies
  - Technology: Socket, WS*, REST, Kafka, MQTT, DB, File…
  - Semantic: Supplier defines the semantic - not OpenWMS
- Expected response times depend on the type of warehouse (automatic/ manual)
  - Who is my client? Humans don’t have fixed timeouts - A Robot has!

Systemrequirements

Longevity

Security

Interoperability

Portability

Scalability

Reliability

Fault tolerance

Recoverability

Efficiency

Availability

Performance

Throughput

Maintainability

Cost Efficiency

Dev. X

Extendability

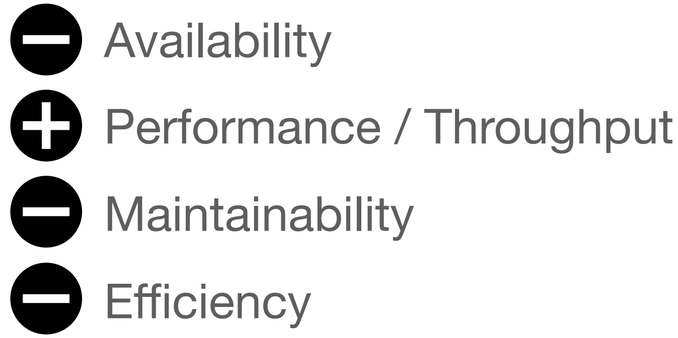
Operationability

History
In 2004

- Logistic software is predominantly written in languages like C/C++
- Less distributed systems exist, most of the time super computers are in place
- 4GL: Oracle PL/SQL and Forms gain popularity
- 2 master thesis build the base of OpenWMS.org

History

In 2004



- **Availability**: JBoss 4.0.3 in parallel HA Cluster
- **Availability**: PLC driver is a HA Singleton, but doesn’t prevent downtimes at deployment time
- **Throughput**: driver component in Java at least performant as the one in C
- **Maintainability**: XDoclet, Code Generation, JSF Websites, EAR/HAR
- **Longlivety**: J2EE Stack promises longevity

History

In 2004

Monolith - JBoss 4.0.3 - XDoclet - J2EE1.4 - Hibernate 2 - JSF 1

```
/*
 * This class was automatically generated with
 * <a href="http://www.castor.org">Castor 0.9.5.4</a>, using an XML
 * Schema.
 * $Id: TransportUnit.java,v 1.1 2005/10/15 15:43:07 hscherrer $
 */

package org.openwms.common.domain;

  //---------------------------------/
 //- Imported classes and packages -/
//---------------------------------/

import java.util.Vector;

import org.exolab.castor.xml.Marshaller;
import org.exolab.castor.xml.Unmarshaller;

/**
 * Class TransportUnit.
 *
 * @version $Revision: 1.1 $ $Date: 2005/10/15 15:43:07 $
 */
public class TransportUnit implements java.io.Serializable {


      //---------------------------/
     //- Class/Member Variables -/
    //---------------------------/

    /**
     * Field _transportUnits
     */
    private java.util.Vector<TransportUnit> _transportUnits;
```

```
/**
 * @ejb.bean
 *  name               = "TransportUnitMgr"
 *  transaction-type   = "Container"
 *  view-type          = "both"
 *  jndi-name          = "TransportUnitMgr"
 *  local-jndi-name    = "TransportUnitMgr"
 *  type               = "Stateless"
 *
 * @jboss.clustered
 *  cluster="true";
 *
 * @jboss.cluster-config
 *  home-policy="org.jboss.ha.framework.interfaces.FirstAvailable"
 *  bean-policy="org.jboss.ha.framework.interfaces.FirstAvailable"
 */
public class TransportUnitMgrBean implements SessionBean {
```

History
In 2005/2006

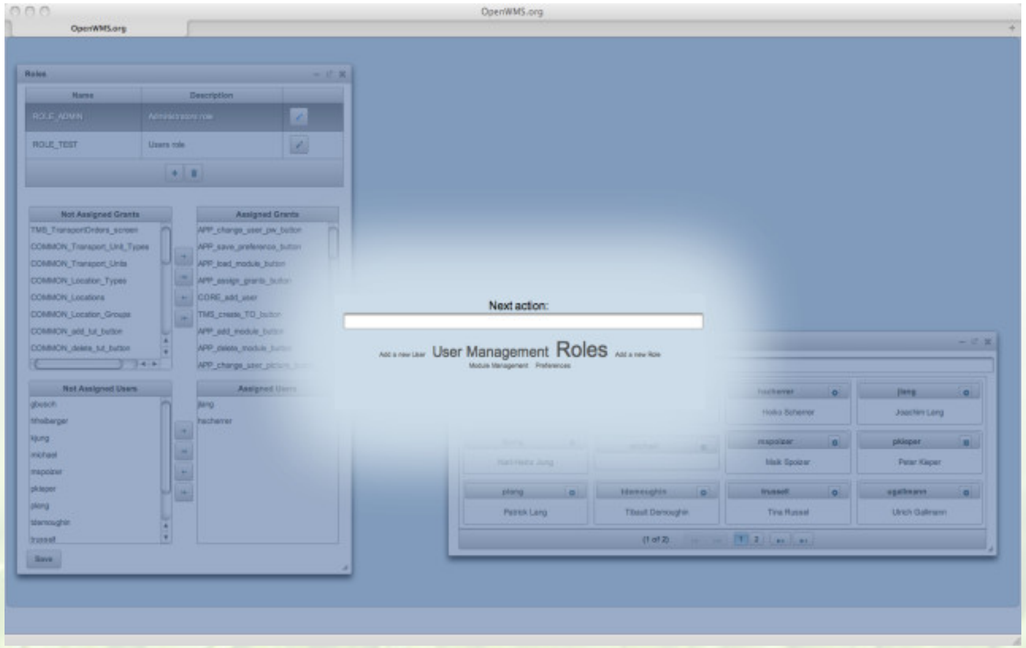
- Move to Java 5, JavaEE, JBoss 5, EJB3.0
- Deployment as EAR, WAR
- Development becomes easier (Annotations)
- Development cycles still clumsy
- JSF as frontend technology very exhausting

- Availability
- Performance / Throughput
- Maintainability
- Efficiency

History
In 2007

- Move to Spring Framework 2.0.3/2.0.5/3
- XML Configuration & Annotationen
- Apache Tomcat Deployments & Hotswap
- Move from Ant & Ivy to Maven 1
- JSF mit a4j / Richfaces

- Availability
- Performance / Throughput
- Maintainability
- Efficiency

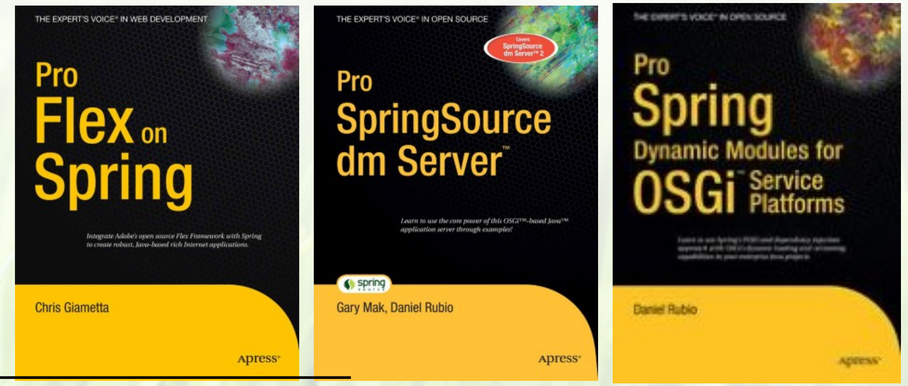


Rise of μServices

History
In 2009 (OSGi)

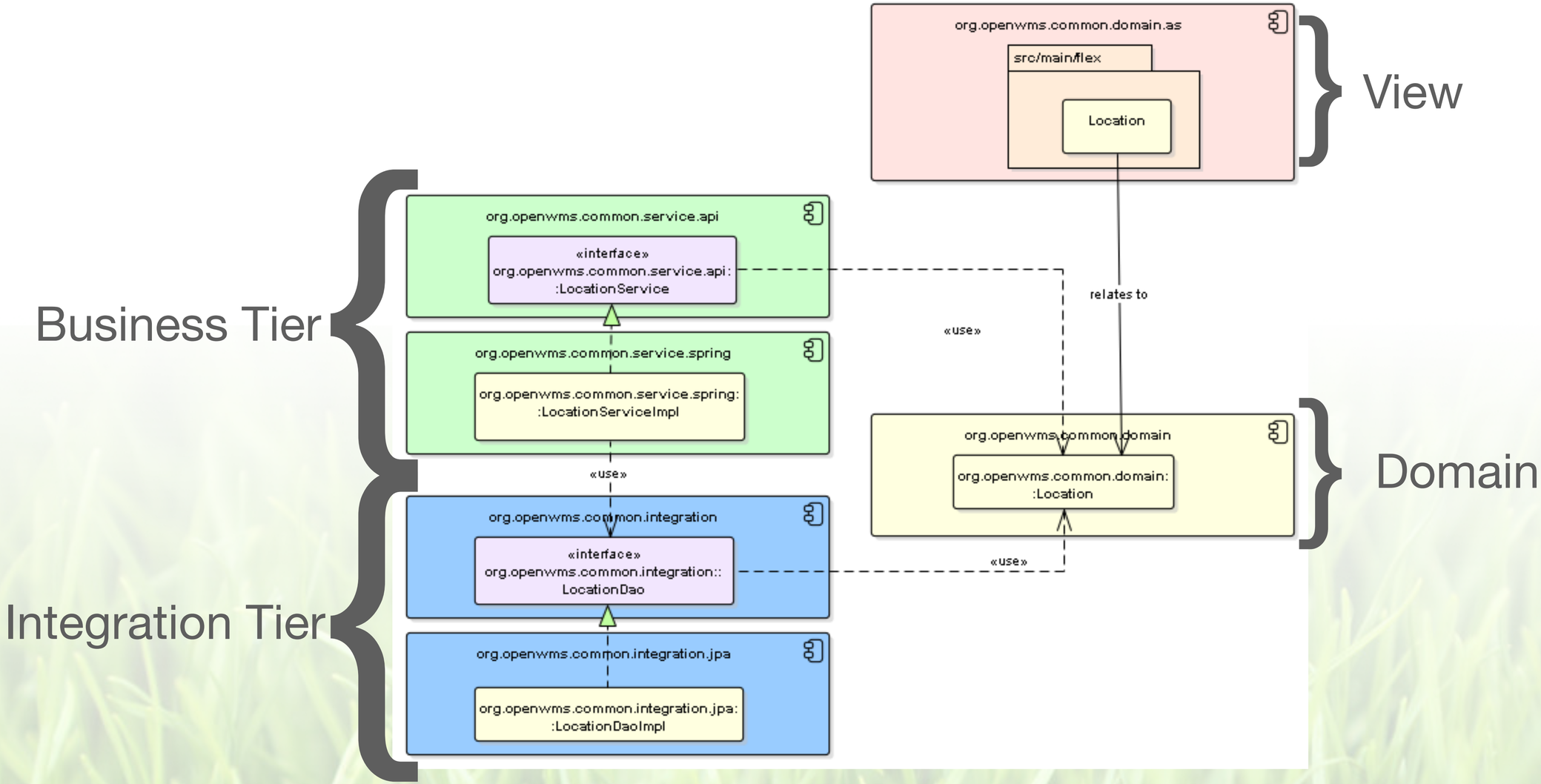
- Spring DM (Dynamic Modules), SpringSource Enterprise Bundle Repository (EBR), Spring Flex
- OSGi Runtime: Spring DM Server with Service Damping

Spring DM automatically creates proxies for OSGi services so that the actual service object may come and go at runtime. If a service disappears, any proxies to the service will wait for the service to re-appear. This effect is known as damping. (https://docs.spring.io/s2-dmserver/2.0.x/getting-started/htmlsingle/getting-started.html)



History

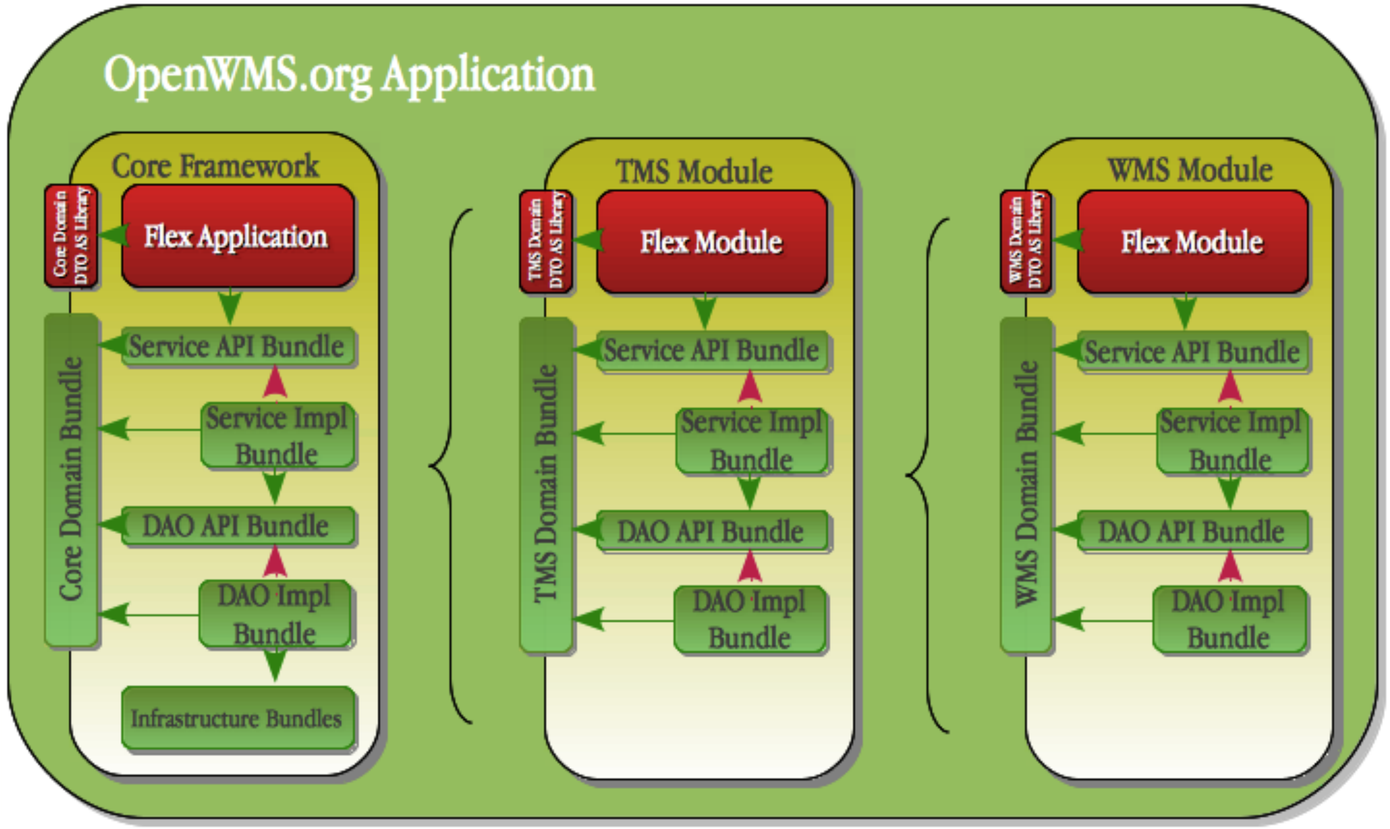
OpenWMS with Spring & Flex on OSGi (till 2014)

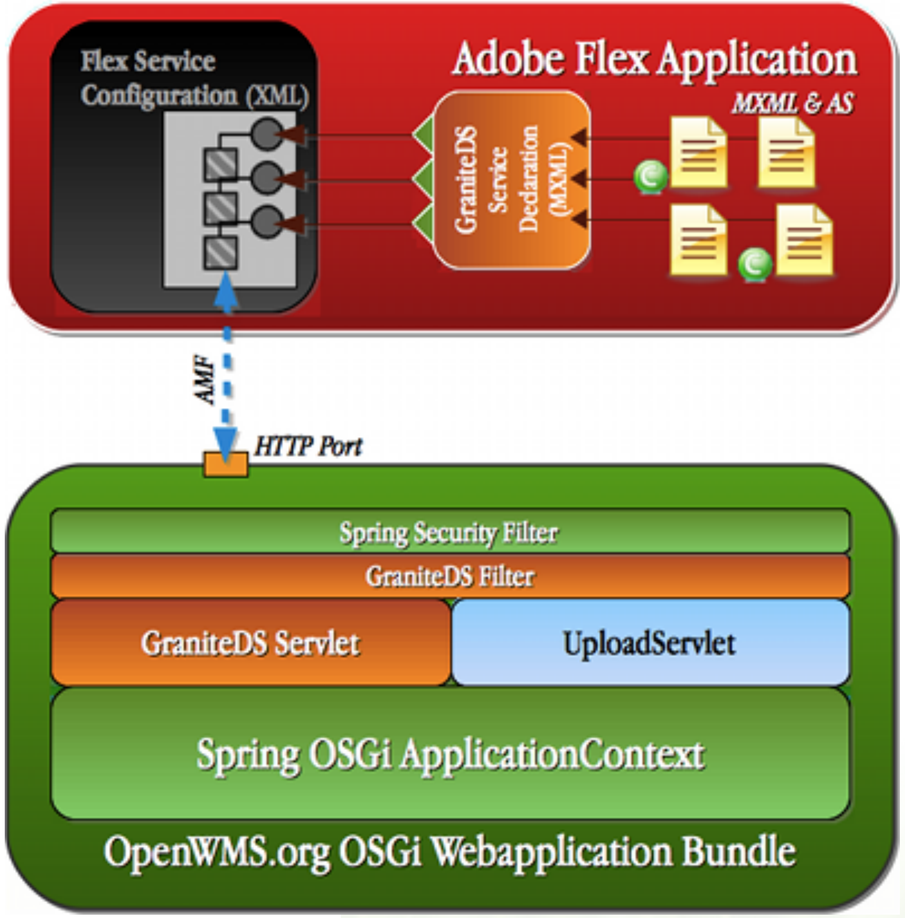


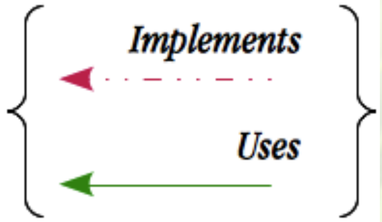
History

OpenWMS with Spring & Flex on OSGi

http://openwms2005.sourceforge.net/user/reference/0.1/html/index/index.html







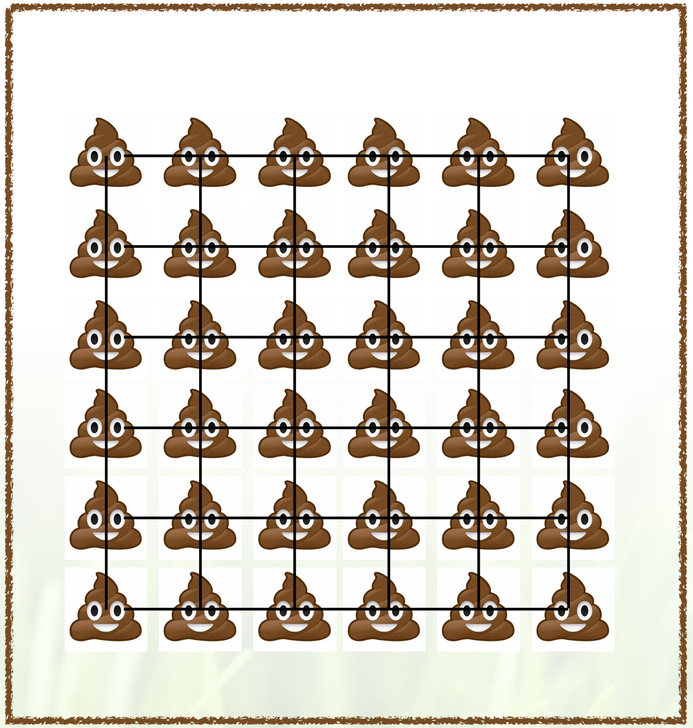
History

OpenWMS with Spring & Flex on OSGi (till 2014)

Single Node

Monolith

Single Node



OSGi µServices

The OSGi bundle hell

- LICENSE
- org.openwms.app
- org.openwms.common
  - org.openwms.common.comm
  - org.openwms.common.domain
  - org.openwms.common.domain.as
  - org.openwms.common.flex.lib
  - org.openwms.common.flex.module
  - org.openwms.common.integration
  - org.openwms.common.integration.jpa
  - org.openwms.common.service.api
  - org.openwms.common.service.spring
  - org.openwms.common.util
  - pom.xml
  - src
- org.openwms.core
  - org.openwms.client.flex.lib
  - org.openwms.core.client.rest
  - org.openwms.core.domain
  - org.openwms.core.domain.as
  - org.openwms.core.infrastructure.configuration.provider
  - org.openwms.core.infrastructure.hsql
  - org.openwms.core.infrastructure.oracle
  - org.openwms.core.infrastructure.postgres
  - org.openwms.core.integration
  - org.openwms.core.integration.eclipselink
  - org.openwms.core.integration.file
  - org.openwms.core.integration.hibernate
  - org.openwms.core.integration.jpa
  - org.openwms.core.service.api
  - org.openwms.core.service.spring
  - org.openwms.core.test
  - org.openwms.core.util
  - pom.xml
  - src
- org.openwms.tms
  - org.openwms.tms.domain
  - org.openwms.tms.domain.as
  - org.openwms.tms.flex.module
  - org.openwms.tms.integration
  - org.openwms.tms.integration.jpa
  - org.openwms.tms.service.api
  - org.openwms.tms.service.spring
  - org.openwms.tms.util
  - pom.xml
  - src
- org.openwms.wms
- pom.xml
- README.md
- src

History

OSGi Insights - Manifest

Manifest Generation with Bundlor using a **template.mf**

```
Manifest-Version: 1.0
Bundle-Vendor: ${project.groupId}
Bundle-Version: ${bundle.version}
Bundle-Name: ${project.artifactId}
Bundle-ManifestVersion: 2
Bundle-SymbolicName: ${project.artifactId}
Import-Library:
 org.springframework.spring;version="${osgi.spring}"
Import-Bundle:
 com.springsource.javax.persistence;version="1.0.0",
 org.openwms.core.domain;version="0.0.1.SNAPSHOT",
 org.openwms.common.domain;version="0.0.1.SNAPSHOT",
 org.openwms.common.integration;version="0.0.1.SNAPSHOT",
 com.springsource.org.hibernate;version="${osgi.hibernate}"
Require-Bundle:
 com.springsource.org.hibernate,
 com.springsource.javassist
Import-Package:
 org.hibernate.proxy; version="0.0.0",
 javax.sql;version="0",
 org.springframework.context.weaving;version="${osgi.spring}",
 org.springframework.transaction.aspectj;version="${osgi.spring}"
Excluded-Imports:
 org.openwms.common.framework.test.*,
 org.openwms.common.test.*,
 org.springframework.test.*,
 org.junit.*,
 junit.framework.*
Excluded-Exports:
 org.openwms.common.framework.test.*,
 org.openwms.common.test.*,
 org.springframework.test.*,
 org.junit.*
```

History

OSGi Insights - Maven

SpringSource OSGi Plugins

```
<plugin>
    <groupId>org.sonatype.flexmojos</groupId>
    <artifactId>flexmojos-maven-plugin</artifactId>
    <version>3.3.0</version>
</plugin>

<plugin>
    <groupId>com.springsource.bundlor</groupId>
    <artifactId>com.springsource.bundlor.maven</artifactId>
    <version>1.0.0.RELEASE</version>
    <configuration>
        <inputPath>${basedir}/target/classes</inputPath>
    </configuration>
    <executions>
        <execution>
            <goals>
                <goal>bundlor</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-par-plugin</artifactId>
    <version>1.0.0.RELEASE</version>
</plugin>
```

Eases MANIFEST handling

Building PAR Archives

Adobe Flex Support

```
<!-- Flex Framework and stuff -->
<dependency>
    <groupId>com.adobe.flex.framework</groupId>
    <artifactId>flex-framework</artifactId>
    <version>${flex.version}</version>
    <type>pom</type>
</dependency>
<dependency>
    <groupId>com.adobe.flex</groupId>
    <artifactId>compiler</artifactId>
    <version>${flex.version}</version>
    <type>pom</type>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-swc</artifactId>
    <version>${granite.version}</version>
    <type>swc</type>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-essentials-swc</artifactId>
    <version>${granite.version}</version>
    <type>swc</type>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-core</artifactId>
    <version>${granite.version}</version>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-spring</artifactId>
    <version>${granite.version}</version>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-eclipselink</artifactId>
    <version>${granite.version}</version>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-toplink</artifactId>
    <version>${granite.version}</version>
</dependency>
<dependency>
    <groupId>org.graniteds</groupId>
    <artifactId>granite-hibernate</artifactId>
    <version>${granite.version}</version>
</dependency>
<dependency>
    <groupId>com.asunit</groupId>
    <artifactId>asunit</artifactId>
    <version>20071011</version>
    <type>swc</type>
</dependency>
```

Library Definition

```
<!-- Spring Fra
<dependency>
    <groupId>org.                    oupId>
    <artifactId>o                    g.spring-library</artifactId>
    <type>libd</type>
    <scope>provided</scope>
</dependency>
<!-- Spring Security -->
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>org.springframework.security.core</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.springframework.security</groupId>
    <artifactId>org.springframework.security.web</artifactId>
    <scope>provided</scope>
</dependency>

<!-- JEE API -->
<dependency>
    <groupId>javax.persistence</groupId>
    <artifactId>com.springsource.javax.persistence</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>javax.transaction</groupId>
    <artifactId>com.springsource.javax.transaction</artifactId>
    <scope>provided</scope>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>com.springsource.org.apache.commons.lang</artifactId>
    <scope>provided</scope>
</dependency>

<!-- JPA Provider -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>org.hibernate.ejb-library</artifactId>
    <type>libd</type>
    <scope>test</scope>
</dependency>

<!-- Weaver -->
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>org.aspectj-library</artifactId>
    <type>libd</type>
    <scope>test</scope>
</dependency>
```

Repackaged OSGi bundles

Loadtime Weaving

PAR: https://docs.spring.io/s2-dmserver/2.0.x/programmer-guide/html/ch04s02.html

History

OSGi Insights - Imports & Exports

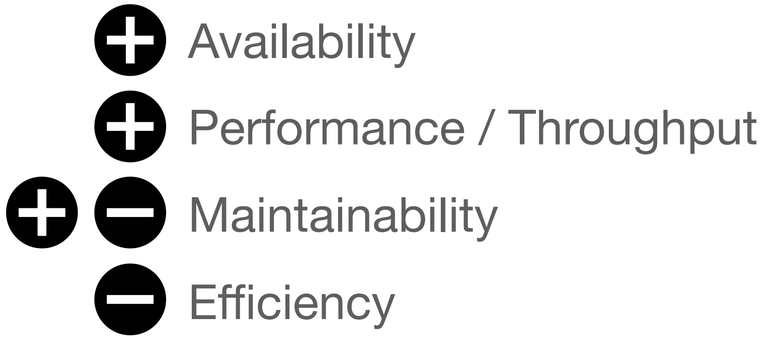
Context activated with Spring Profile

osgi-context.xml

```
<osgix:cm-properties id="globals" persistent-id="org.openwms.core.infrastructure.configuration-${openwms.configuration.version}" />

<context:property-placeholder properties-ref="globals" />

<!-- imports -->
<osgi:reference id="transactionManager" interface="org.springframework.transaction.PlatformTransactionManager" />
<osgi:reference id="locationDao" interface="org.openwms.common.integration.LocationDao" />
<osgi:reference id="locationTypeDao" interface="org.openwms.common.integration.LocationTypeDao" />
<osgi:reference id="locationGroupDao" interface="org.openwms.common.integration.LocationGroupDao" />
<osgi:reference id="transportUnitDao" interface="org.openwms.common.integration.TransportUnitDao" />
<osgi:reference id="transportUnitTypeDao" interface="org.openwms.common.integration.TransportUnitTypeDao" />

    lpers and delegates -->
    reference id="onRemovalListener" interface="org.openwms.core.service.listener.OnRemovalListener" cardinality="0..1" />

<!-- exports -->
<osgi:service ref="locationGroupService" interface="org.openwms.common.service.LocationGroupService" />
<osgi:service ref="locationService" interface="org.openwms.common.service.LocationService" />
<osgi:service ref="transportUnitService" interface="org.openwms.common.service.TransportUnitService" />
```

Spring Exporter Pattern

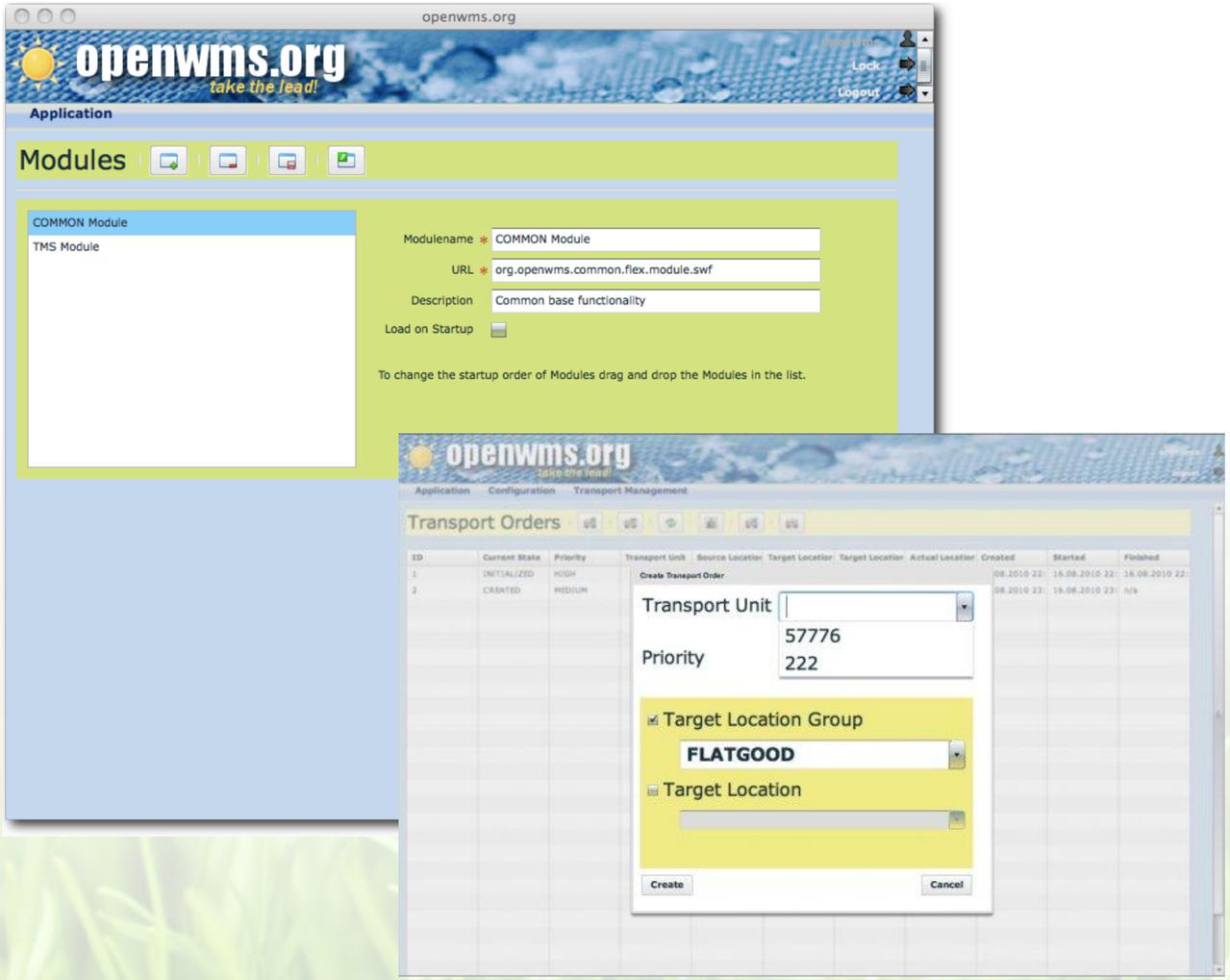
History
Summary OSGi



- Clean technical structuring and modularisation
- Application is vertically and horizontally layered
- Despite the abstraction with **Bundlor** and **SpringDM**, OSGi is still a niche technology compared to popular JavaEE technology
- Very **flat learn curve** and **low Developer Adoption**
- SpringSource has 2013 announced the end of support for OSGi
- Remote OSGi (R-OSGI ETHZ) https://people.inf.ethz.ch/troscoe/pubs/middleware07-rosgi.pdf

History

Adobe Flex & ActionScript v3



History

Adobe Flex & ActionScript v3

- Modular UI with Flex Modules, **ActionScript**, **GraniteDS**, binary **AMF** protocol
- RAD and WYSIWYG with **Adobe Flex Builder**
- High performing in Load & Rendering
- AS was a rich and modern language with **annotations**, **DI**, **event bubbling** etc.
- GraniteDS is for AS what Spring is for Java (https://github.com/graniteds/graniteds)
- GraniteDS generates AS types from Java types with Groovy Script templates

Historiy

ActionScript v3

- Superset of EcmaScript 4
- Like Java/C#
- Namespaces, Classes, Props
- Inheritance
- Annotations
- AsDoc

```
package org.openwms.common.domain {

    import mx.collections.ListCollectionView;

    [Bindable]
    [RemoteClass(alias="org.openwms.common.domain.Location")]
    /**
     * A Location, defines a place within a warehouse.
     * <p>
     * Could be something like a storage location in the stock as well as a location
     * on a conveyer. Also virtual or error locations can be represented with the
     * <code>Location</code> entity.
     * </p>
     * Multiple <code>Location</code>s can be grouped together to a <code>LocationGroup</code>.
     *
     * @version $Revision$
     * @since 0.1
     * @see org.openwms.common.domain.LocationGroup
     */
    public class Location extends LocationBase {

       public function Location(pk:LocationPK = null) {
            super();
            if (pk != null) {
                _locationId = pk;
            }
        }

        /**
         * Returns the locationId in the format area/aisle/x/y/z.
         *
         * @return area/aisle/x/y/z
         */
        public function toString():String {
            return _locationId.area+"/"+_locationId.aisle+"/"+_locationId.x+"/"+_locationId.y+"
        }

        public function set messages(value:ListCollectionView) : void {
            _messages = value;
        }

    }
}
```

History

ActionScript v3


```
[Name("locationDelegate")]
[ManagedEvent(name="LOAD_ALL_LOCATION_TYPES")]
public class LocationDelegate {

        [Inject]
        [Bindable]
        public va                ontext;

        [Inject]
        [Bindable]
        public var commonModelLocator:CommonModelLocator;

        public function LocationDele

        [Observer("CREATE_LOCATION")]
        /**
         * Creates a new Location by calling the corresponding service.
         * Tide event observers : CREATE_LOCATION
         *
         * @param event A LocationEvent that stores the new Location in its data
         */
        public function createLocation(event : LocationEvent) : void {
            tideContext.locationService.addEntity(event.data as Location, onLocationCreated, onFault);
        }
        private function onLocationCreated(event:TideResultEvent):void {
            dispatchEvent(new LocationEvent(LocationEvent.LOAD_ALL_LOCATIONS));
        }
```

Name

IoC

Event Observer

Functional argument

History

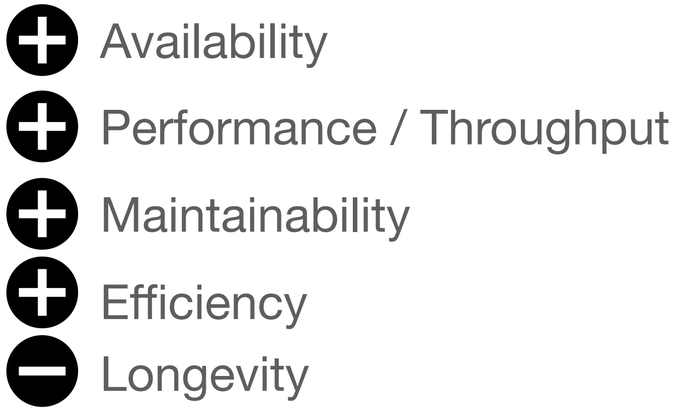
Flash/Flex/Shockwave & MXML

- Components encapsulate view & state
- Kinda React with JSX ?

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Canvas … show="onShow()">

    <mx:Metadata>
        [Name]
        [ManagedEvent(name="SAVE_LOCATION")]
        [ResourceBundle("commonEntity")]
    </mx:Metadata>

    <mx:Script>
        <![CDATA[
            [Inject]
            [Bindable]
            public var commonModelLocator : CommonModelLocator;
            [Inject]
            [Bindable]
            public var identity : Identity;

            [Bindable]
            private var selected : Location;

            private function onShow() : void {
               dispatchEvent(new LocationTypeEvent(LocationTypeEvent.LOAD_ALL_LOCATION_TYPES));
            }

        ]]>
    </mx:Script>

    <mx:HBox height="100%" width="95%" x="10" y="63">
        <mx:DataGrid id="locationsList" allowMultipleSelection="true" sortableColumns="true" dataProvider="{c
            <mx:columns>
                <comm:CoordinateColumn column="area" headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'ent
                <comm:CoordinateColumn column="aisle" headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'en
                <comm:CoordinateColumn column="x" headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity
                <comm:CoordinateColumn column="y" headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity
                <comm:CoordinateColumn column="z" headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity
                <mx:DataGridColumn headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity_location_locat
                <mx:DataGridColumn headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity_location_descr
                <mx:DataGridColumn headerText="{I18nUtil.trans(I18nUtil.COMMON_ENTITY, 'entity_location_count
                    <mx:itemRenderer>
                        <mx:Component>
                            <mx:CheckBox/>
                        </mx:Component>
                    </mx:itemRenderer>
                </mx:DataGridColumn>
            </mx:columns>
        </mx:DataGrid>
    </mx:HBox>
</mx:Canvas>
```

History

Fazit Adobe Flex



- **ActionScript v3** with **GraniteDS** very advanced language and framework
- Scales well for big enterprise application
- Very comprehensive tool support und high efficiency in development thanks to Flex Builder
- No standard, requires **Flash Plugin**
- With the rise of HTML5, Flash has been handed over to the ASF

History

Tools: DocBook

- User docs & developer docs
- XML dialect
- OASIS standard
- Maven Plugin generates
  HTML | HTML-Single | PDF from XML

```
<plugin>
    <groupId>com.agilejava.docbkx</groupId>
    <artifactId>docbkx-maven-plugin</artifactId>
    <version>2.0.13</version>
    <configuration>
        <sourceDirectory>${docbook.source}</sourceDirectory>
        <xincludeSupported>true</xincludeSupported>
       <useExtensions>1</useExtensions>
        <highlightSource>1</highlightSource>
        <highlightDefaultLanguage></highlightDefaultLanguage>
        <calloutsExtension>1</calloutsExtension>
    </configuration>
    <executions>
        <execution>
            <id>PDF</id>
            <goals>
                <goal>generate-pdf</goal>
            </goals>
            ...
        </execution>
       <execution>
            <id>HTML-SINGLE</id>
            <goals>
                <goal>generate-html</goal>
            </goals>
            ...
       </execution>
        <execution>
            <id>HTML</id>
            <goals>
                <goal>generate-html</goal>
            </goals>
           <configuration>
                <chunkedOutput>true</chunkedOutput>
                <htmlCustomization>${docbook.source}/resources/xsl/html-chunk.xsl</h
                <targetDirectory>${docbook.target}/html</targetDirectory>
                <htmlStylesheet>http://openwms2005.sourceforge.net/user/reference/${
                <postProcess>
                    <copy todir="${docbook.target}/html/index/resources">
                        <fileset dir="${docbook.source}/resources">
                           <include name="**/*.css" />
                            <include name="**/*.png" />
                            <include name="**/*.gif" />
                            <include name="**/*.jpg" />
                        </fileset>
                    </copy>
                </postProcess>
            </configuration>
        </execution>
    </executions>
    <dependencies>
        <dependency>
            <groupId>org.docbook</groupId>
            <artifactId>docbook-xml</artifactId>
            <version>4.4</version>
            <scope>runtime</scope>
        </dependency>
```

History

Tools: Customised DocBook Glossary

- **Javadoc Doclet** for @GlossaryTerm
- **Maven Plugin extension** generated **DocBook section** for glossary

```
/**
 * A Location, represents some physical as well as v
 * <p>
 * Could be something like a storage location in the
 * </p>
 * Multiple Locations can be assigned to a LocationGroup
 *
 * @GlossaryTerm
 * @author <a href="mailto:scherrer@openwms.org">Hei
 * @version $Revision$
 * @since 0.1
 * @see org.openwms.common.domain.LocationGroup
 */
@Entity
@Table(name = "COM_LOCATION")
public class Location { … }
```

Custom Javadoc annotation

```
<plugin>
   <artifactId>maven-javadoc-plugin</artifactId>
   <version>${plugin.javadoc.version}</version>
   <configuration>
       <source>${project.build.sourceEncoding}</source>
       <doclet>org.openwms.doclet.GlossaryGeneratorDoclet</doclet>
       <docletArtifact>
           <groupId>org.openwms</groupId>
           <artifactId>org.openwms.doclet</artifactId>
           <version>0.0.1-SNAPSHOT</version>
       </docletArtifact>
       <aggregate>true</aggregate>
       <charset>${project.build.sourceEncoding}</charset>
       <encoding>${project.build.sourceEncoding}</encoding>
       <docencoding>${project.build.sourceEncoding}</docencoding>
       <breakiterator>true</breakiterator>
       <level>private</level>
       <show>private</show>
       <keywords>true</keywords>
       <useStandardDocletOptions>false</useStandardDocletOptions>
   </configuration>
</plugin>
```

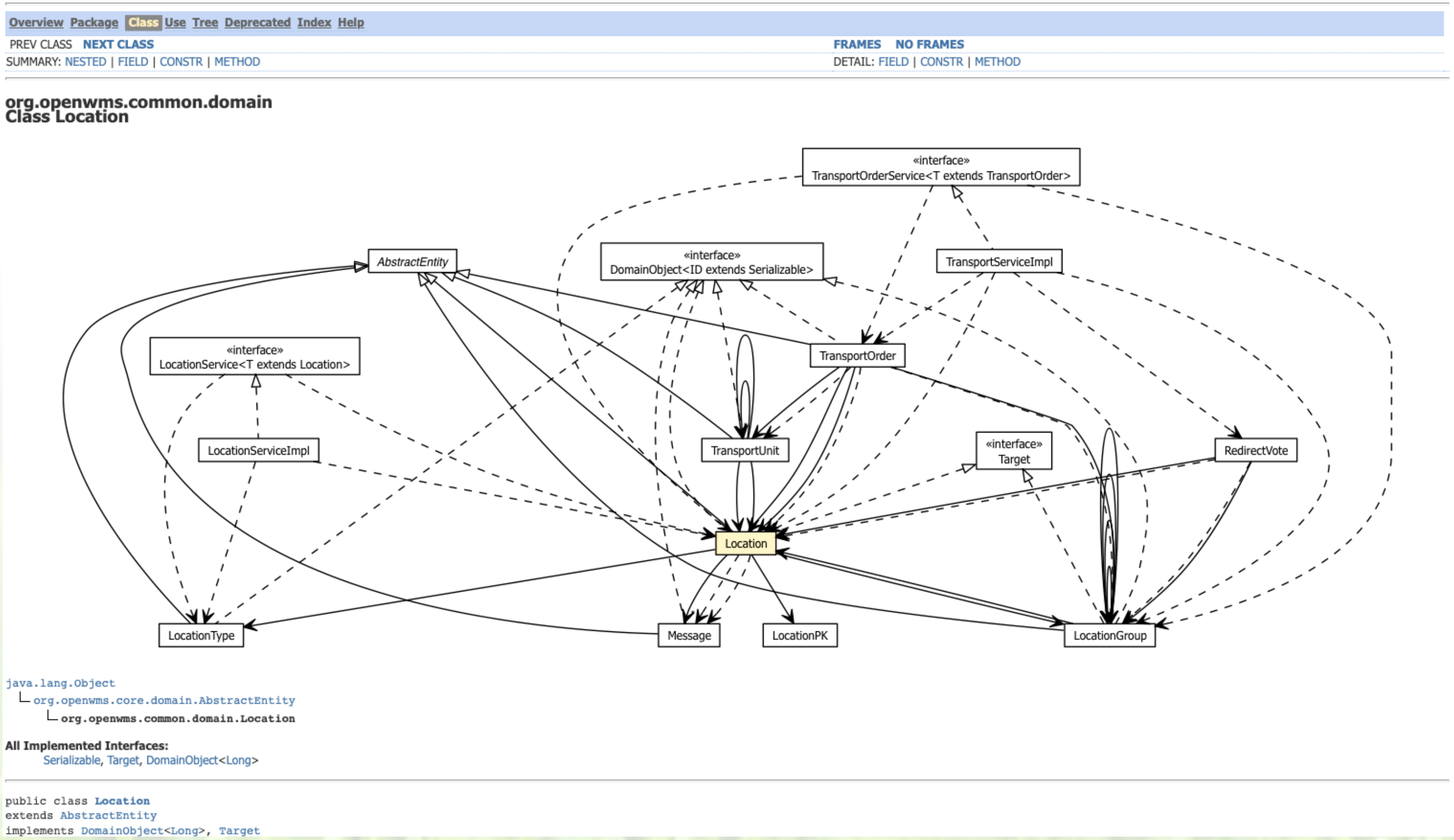
Custom Doclet

History

Tools: Javadoc UML

- Maven Plugin extension generates clickable UML in Javadocs

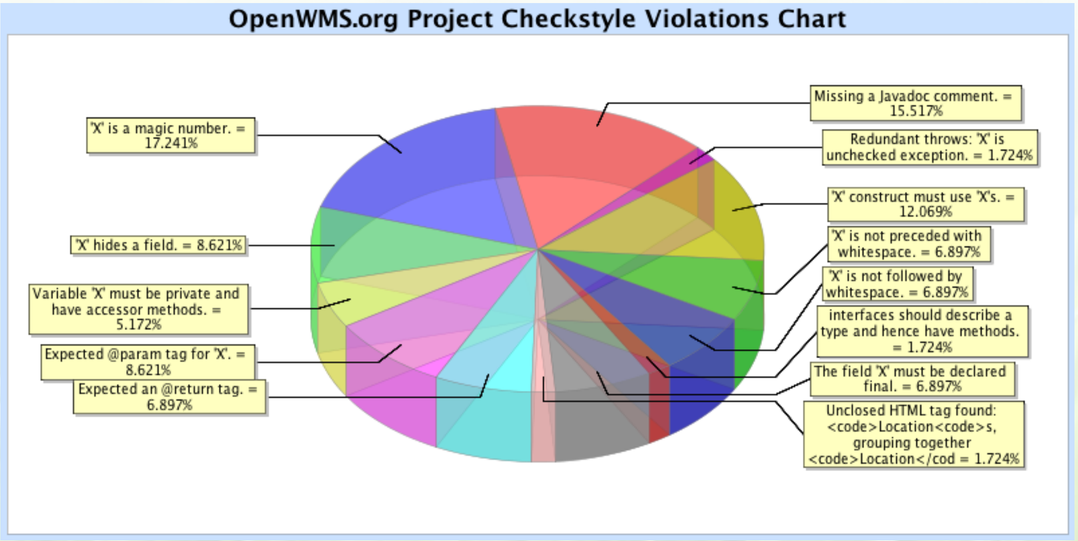
```
<doclet>gr.spinellis.umlgraph.doclet.UmlGraphDoc</doclet>
<docletArtifact>
    <groupId>gr.spinellis</groupId>
    <artifactId>UmlGraph</artifactId>
    <version>4.6</version>
</docletArtifact>
```



History

Tools: Maven Site

- Custom Maven Site Template (**Skin**)
- Taglist, Cobertura, JDepend, Findbugs, Xref, PMD, CPD
- http://openwms2005.sourceforge.net



History

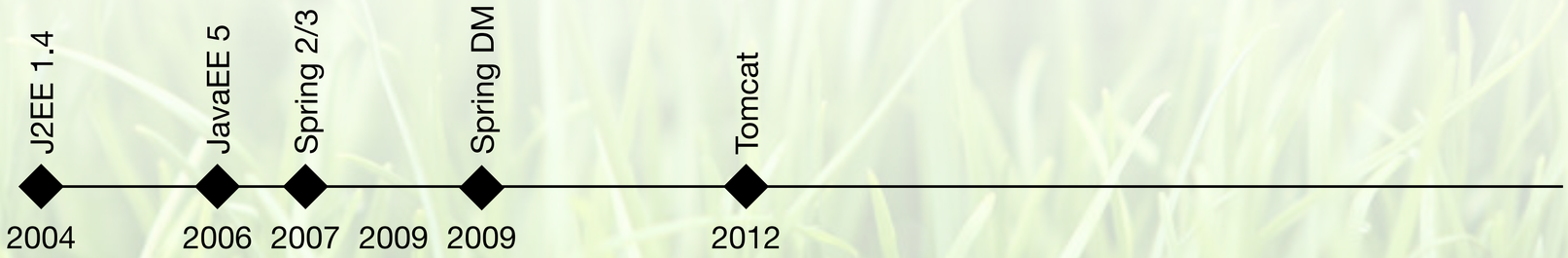
AngularJS (2012) & Tomcat Deployment

- Modular AngularJS UI realised with Maven Overlays
- REST API between UI und backend
- OSGi bundles are packaged in WAR and deployed in Tomcat
- **Spring Profiles** make it possible to activate/deactivate OSGi exporters

Spring Profiles

=> Application can run in dmServer as well as on Tomcat

Spring Exporter Pattern

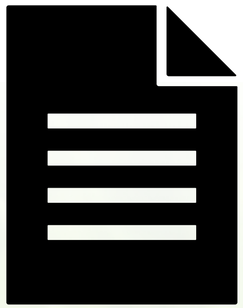


History

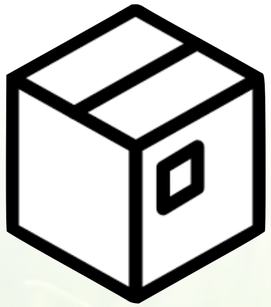
Support dmServer & Tomcat

Build artifacts can be deployed to OSGi server as well in a Servlet Container

Spring Profiles

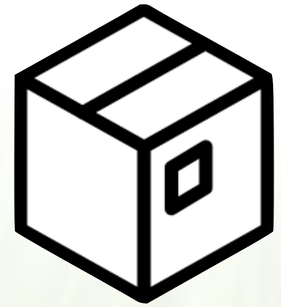


Plan



PAR Archive

org.openwms.common.domain.jar
org.openwms.common.integration.jar
org.openwms.common.integration.jpa.jar
org.openwms.common.service.api.jar
org.openwms.common.service.spring.jar
org.openwms.common.util.jar
org.openwms.core.client.rest.jar
org.openwms.core.domain.jar
org.openwms.core.infrastructure.configuration.provider.jar
org.openwms.core.infrastructure.postgres.jar
org.openwms.core.integration.file.jar
org.openwms.core.integration.hibernate.jar
org.openwms.core.integration.jar
org.openwms.core.integration.jpa.jar
org.openwms.core.service.api.jar
org.openwms.core.service.spring.jar
org.openwms.core.util.jar
org.openwms.tms.domain.jar
org.openwms.tms.integration.jar
org.openwms.tms.integration.jpa.jar
org.openwms.tms.service.api.jar
org.openwms.tms.service.spring.jar
org.openwms.tms.util.jar

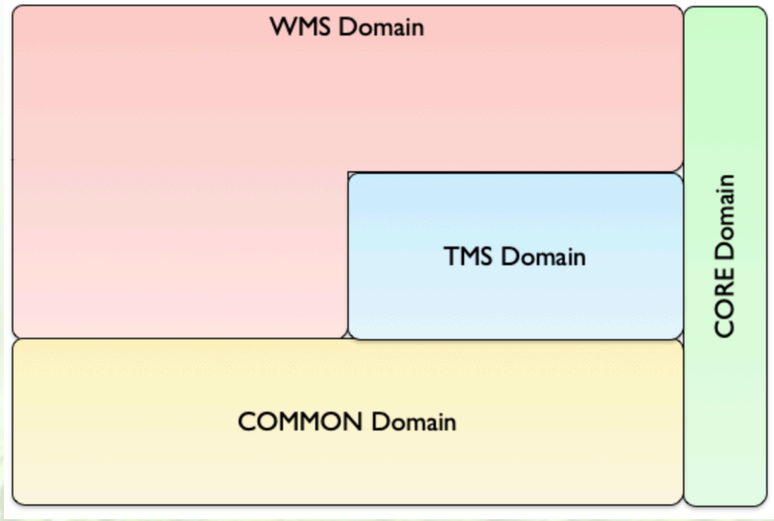


WAR Archive

Microservices

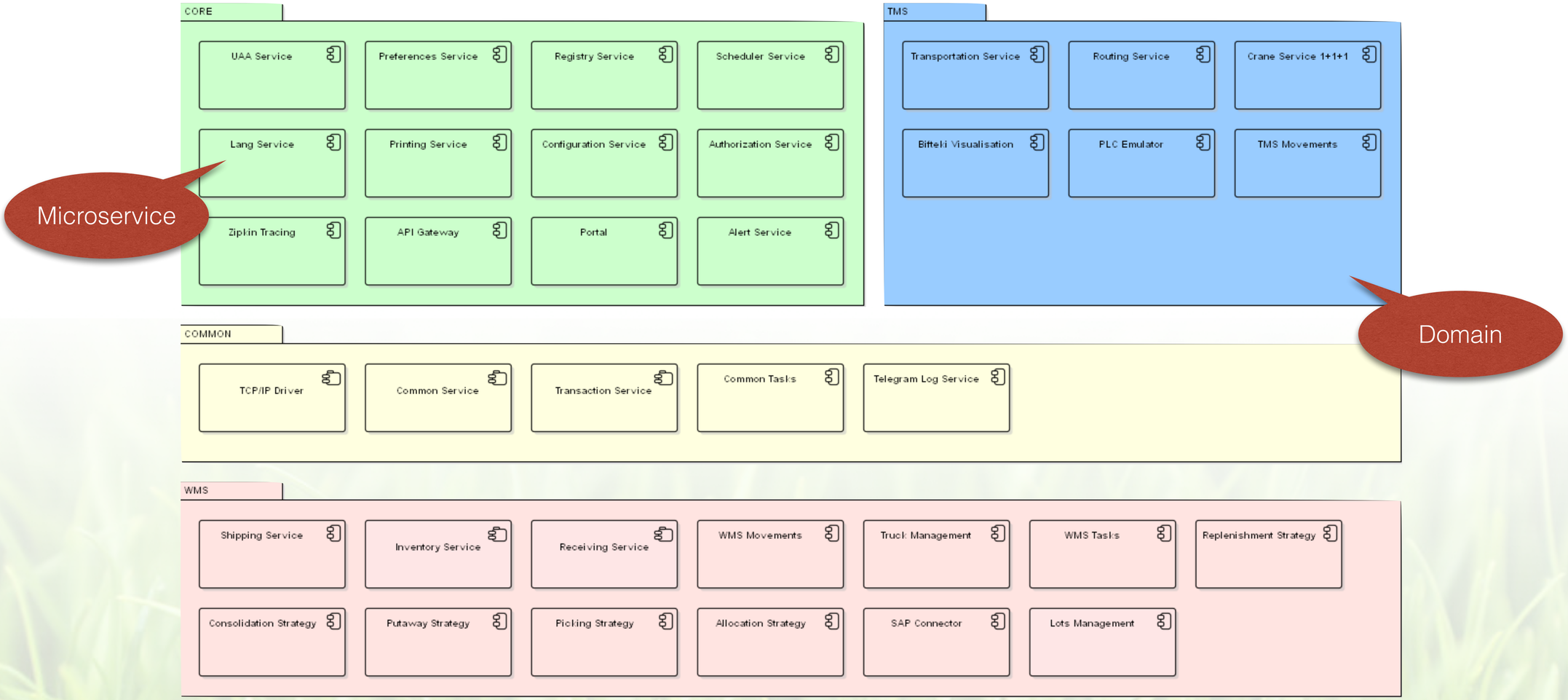
Move to Microservices
2016

- Application was already technically und functionally modularised
- 2016: **SpringBoot**, **Netflix OSS**, **SpringCloud**
- Separation in 4 super domains: **CORE**, **COMMON**, **TMS**, **WMS**
- Infrastructure build with Software Services
  => **No binding to runtime environment**
- Service communication with **REST** & **async. AMQP**



Microservice Architecture

Big Picture

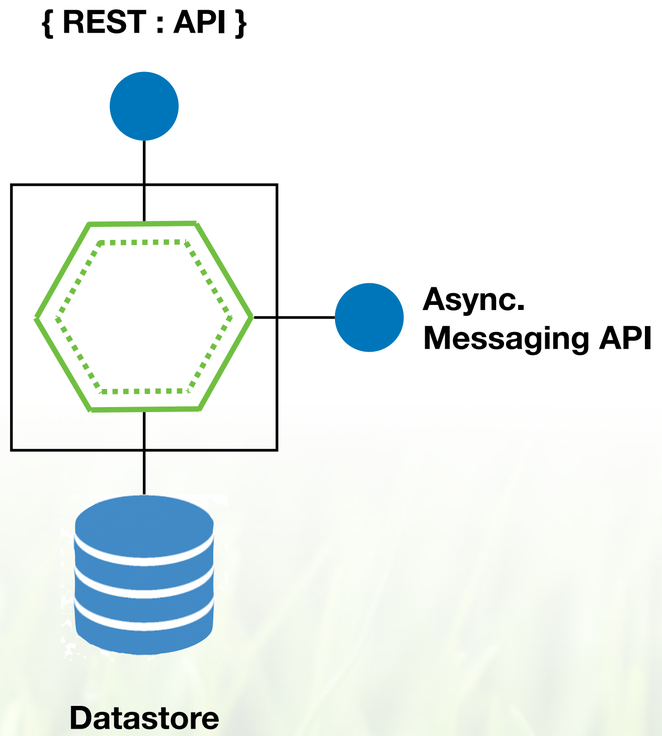


Microservices

Whitebox View

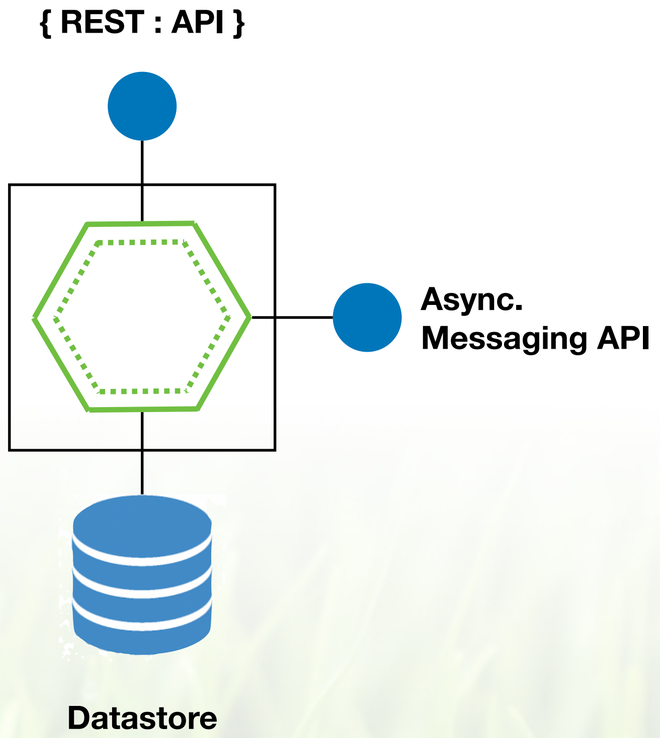
- Users / Clients are interested in
  - API
  - Datastore requirements
  - Logging/Tracing (Monitoring)

=> Documentation is necessary



Microservices
Documentation

- Each microservice has its own website
- Format: **Markdown**, APT, Xdoc etc.
- Built as part of the CI Pipeline
  - **REST API Documentation**
  - Configuration
  - **Database Schema**
  - **Wiki**
  - Javadocs
  - Quality Management
  - Team, CI, VCS etc

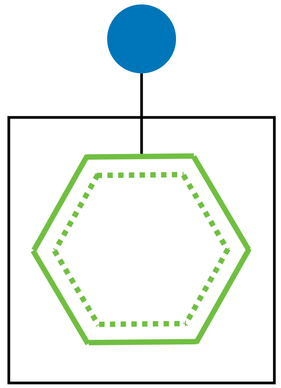


Example: https://openwms.github.io/org.openwms.common.service

Microservices

Doumentation: REST API

{ REST : API }



- **Spring REST Docs** is combined with API-Tests
- API Tests run with **maven-failsafe-plugin**

simple

```
@Test void shall_findby_plccode_404() throws Exception {
    mockMvc.perform(
            get(LocationApiConstants.API_LOCATIONS)
                    .queryParam("plcCode", "NOT EXISTS")
                    .accept(LocationVO.MEDIA_TYPE)
            )
            .andExpect(status().isNotFound())
            .andDo(document("loc-find-plc-404"));
}
```

with JsonPath

```
@Test void shall_findby_plccode() throws Exception {
    mockMvc.perform(
            get(LocationApiConstants.API_LOCATIONS)
                .queryParam("plcCode", TestData.LOCATION_PLC_CODE_EXT)
            )
            .andExpect(status().isOk())
            .andExpect(jsonPath("pKey").exists())
            .andExpect(jsonPath("plcCode", is(TestData.LOCATION_PLC_CODE_EXT)))
 ..
            .andDo(document("loc-find-plc"));
}
```

…document request & response fields

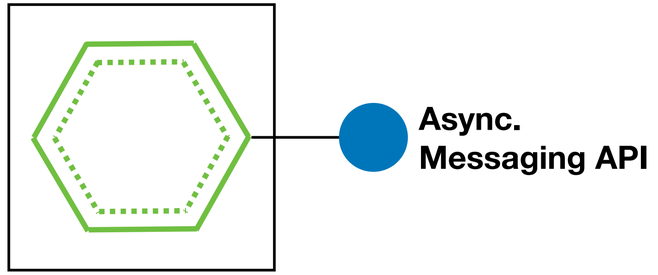
```
.andDo(document("loc-created",
        preprocessResponse(prettyPrint()),
        requestFields(
                fieldWithPath("links[]").ignored(),
                fieldWithPath("locationId").description("Unique natural key"),
                fieldWithPath("plcCode").description("PLC code of the Location")
..
        ),
        responseFields(
                fieldWithPath("links[].*").ignored(),
                fieldWithPath("pKey").description("The persistent technical key of the Location"),
                fieldWithPath("plcCode").description("PLC code of the Location")
..
       )
));
```

Example: https://openwms.github.io/org.openwms.common.service/rest/2.0.0-SNAPSHOT/api.html

Andy Wilkinson Spring I/O 2016: Why REST Docs (https://youtu.be/5XRotk-HQJo)

Microservices

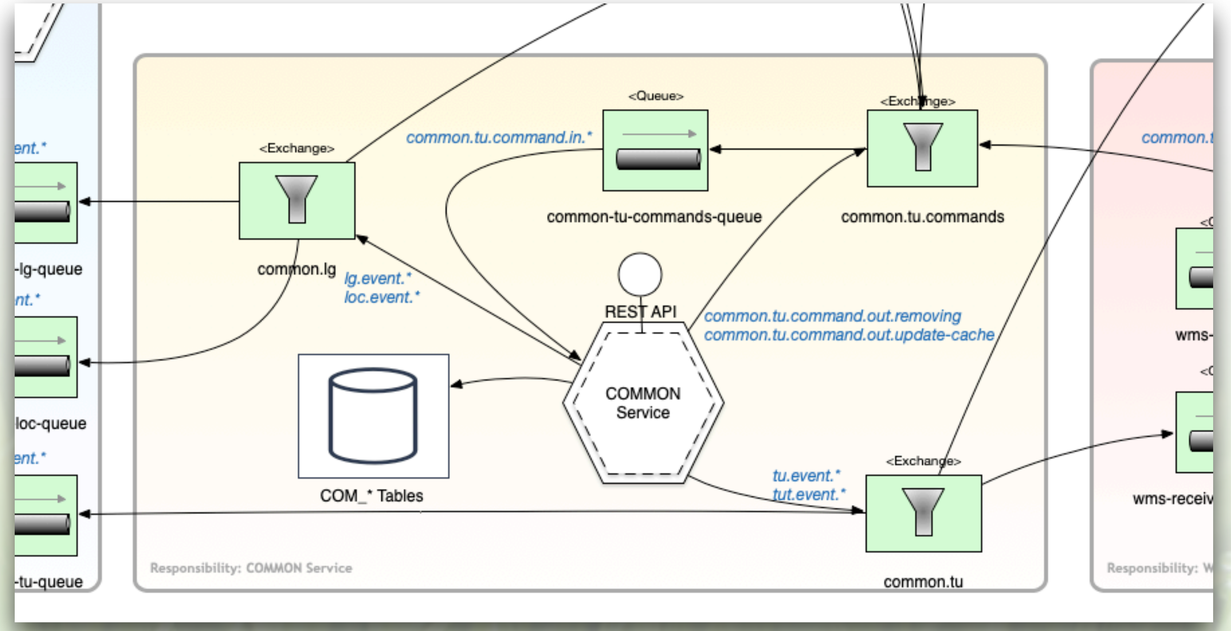
Documentation: Async API



- At best with AsyncAPI (asyncapi.com) - **Not in use yet**
- Current: Documentation with OmniGraffle (always outdated!)
- Best Practises on API Design:

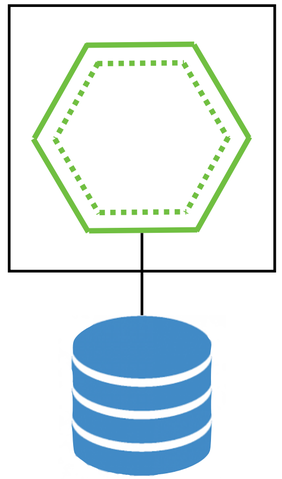
=> SBB@Github

https://schweizerischebundesbahnen.github.io/api-principles/



Microservices

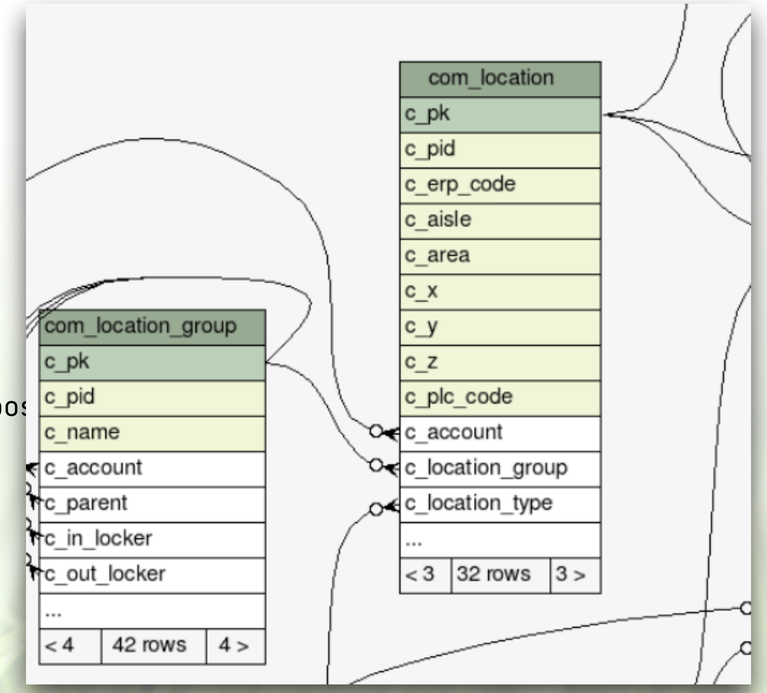
Documentation: Database Schema



Datastore

schemaspy-maven-plugin is part of the Maven Site build

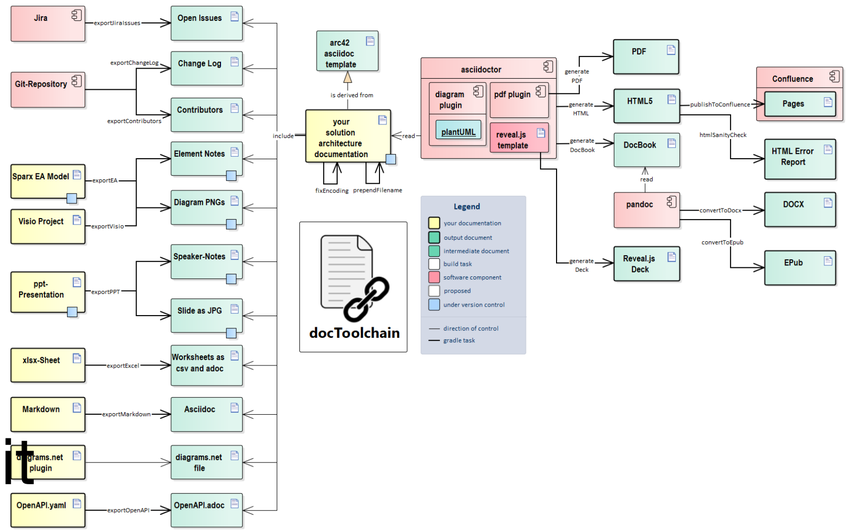
```
<plugin>
    <groupId>nl.geodienstencentrum.maven</groupId>
    <artifactId>schemaspy-maven-plugin</artifactId>
    <configuration>
        <databaseType>pgsql</databaseType>
        <database>${DB_NAME}</database>
        <host>${DB_HOST}</host>
        <port>${DB_PORT}</port>
        <user>${DB_USERNAME}</user>
        <schema>${SCHEMA}</schema>
        <outputDirectory>${schemagen.outputDirectory}</outputDirectory>
        <password>${DB_PASSWORD}</password>
        <pathToDrivers>${M2_REPO_DIR}/org/postgresql/postgresql/${postgresql.version}/postgresql-${pos
    </configuration>
</plugin>
```



Example: https://openwms.github.io/org.openwms.common.service/2.0.0-SNAPSHOT/schemaspy/relationships.html

Microservices

Documentation: DocsAsCode



- Microservice documentation close to source code in Git
- DocToolchain accepts various input/output formats
- **DocToolchain** is part of the Maven Site build
- The **arc42** template is used for architectural documentation
- Upload to **Confluence** in Atlassian Cloud

=> In future upload will happen on custom **OpenProject** server (Markdown in DB)

https://github.com/doctoolchain/doctoolchain

Microservices

Documentation: Customised DocToolchain

- Custom Exporters for Github & JIRA Issues
- Custom Ruby and Groovy site templates

Github Action

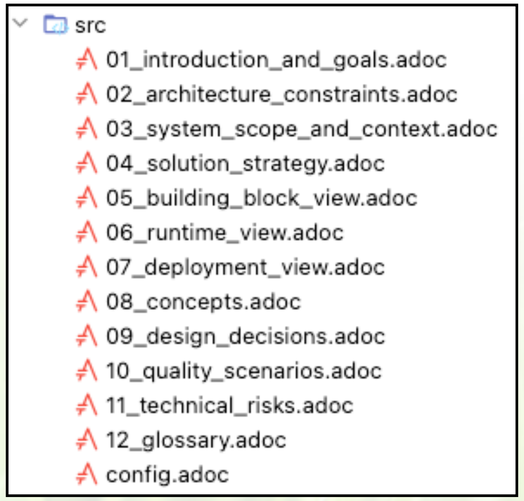
```
- name: Confluence Update
  run: |
    ./scripts/toolsetup
    ./docToolchain-master/bin/doctoolchain doc exportGithubIssues publishToConfluence
```

```
#!/bin/bash
wget https://github.com/openwms/docToolchain/archive/master.zip -P doc/
unzip ./doc/master.zip
```

Exporter

```
task exportGithubIssues() {
..
        gitHub.get(
            path: "repos/${organization}/${repository}/issues",
            query: ['state':'all','filter':'all','per_page': "${resultsPerPage}"],
            headers: headers
        ).data.each {
          if (!it.html_url.contains("/pull/")) {
            if ("open".equals(it.state)) {
              openIssues.append("| ${‘https://www.github.com/..)
            } else {
              closedIssues.append("| ${‘https://www.github.com/..)
            }
            allIssues.append("| ${‘https://www.github.com/'+organization+'/..)
          }
        }
}
```

Docs as Asciidoc



Confluence Output

Example: https://openwms.atlassian.net/wiki/spaces/OPENWMS/overview

Microservice Runtimes

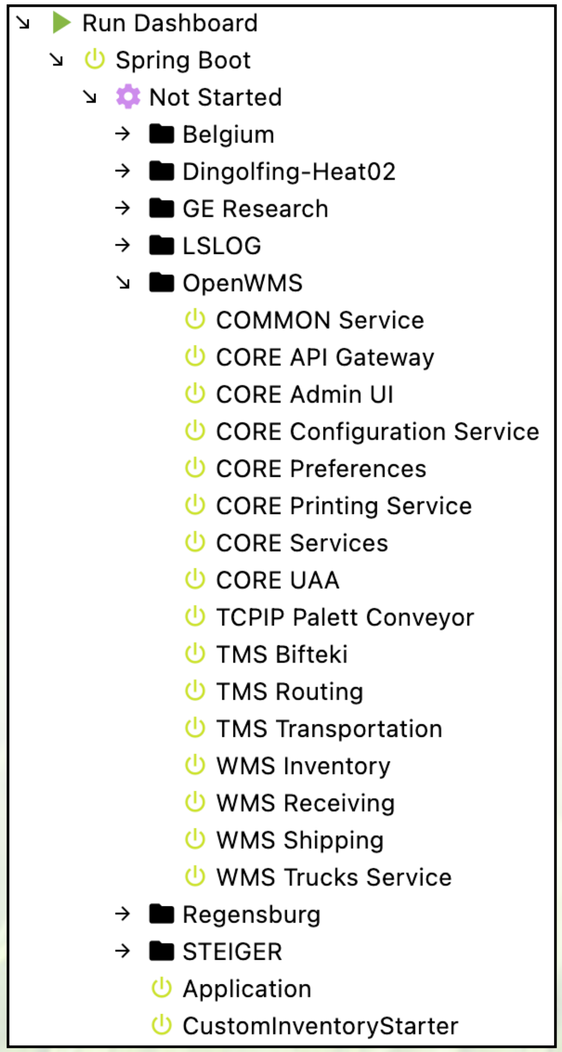
Microservices

Runtimes (with SpringBoot)

- JVM Executable Fat JAR (java -jar)
- SpringBoot Executable JAR (https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#deployment.installing)
- Microsoft Windows Service
- Unix Daemon
- **IDE**
- **Docker-Compose**
- Docker Swarm
- Kubernetes
- Heroku
- **Cloud Provider (AWS EKS, Azure AKS, GCP GKE)**

Portability

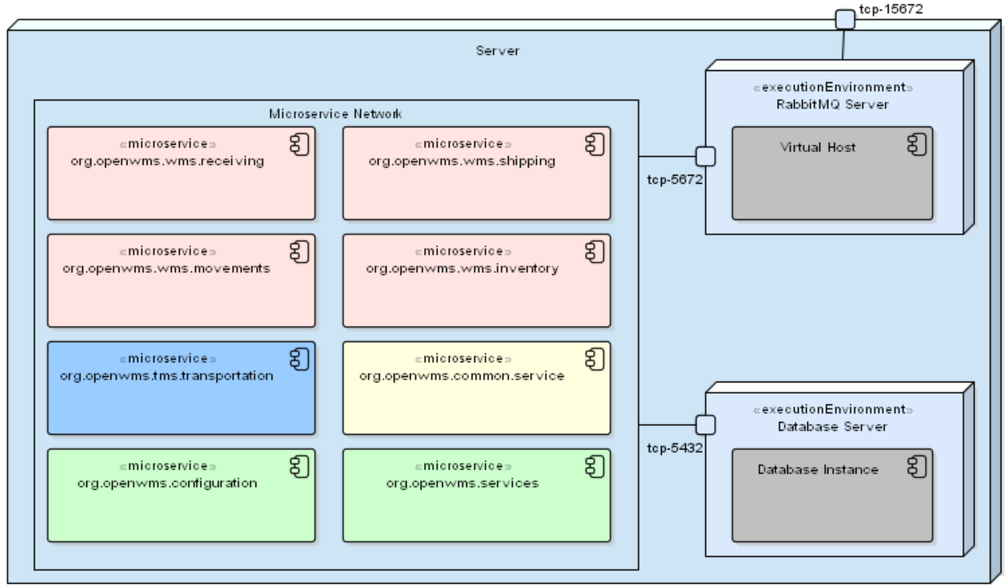
Interoperability



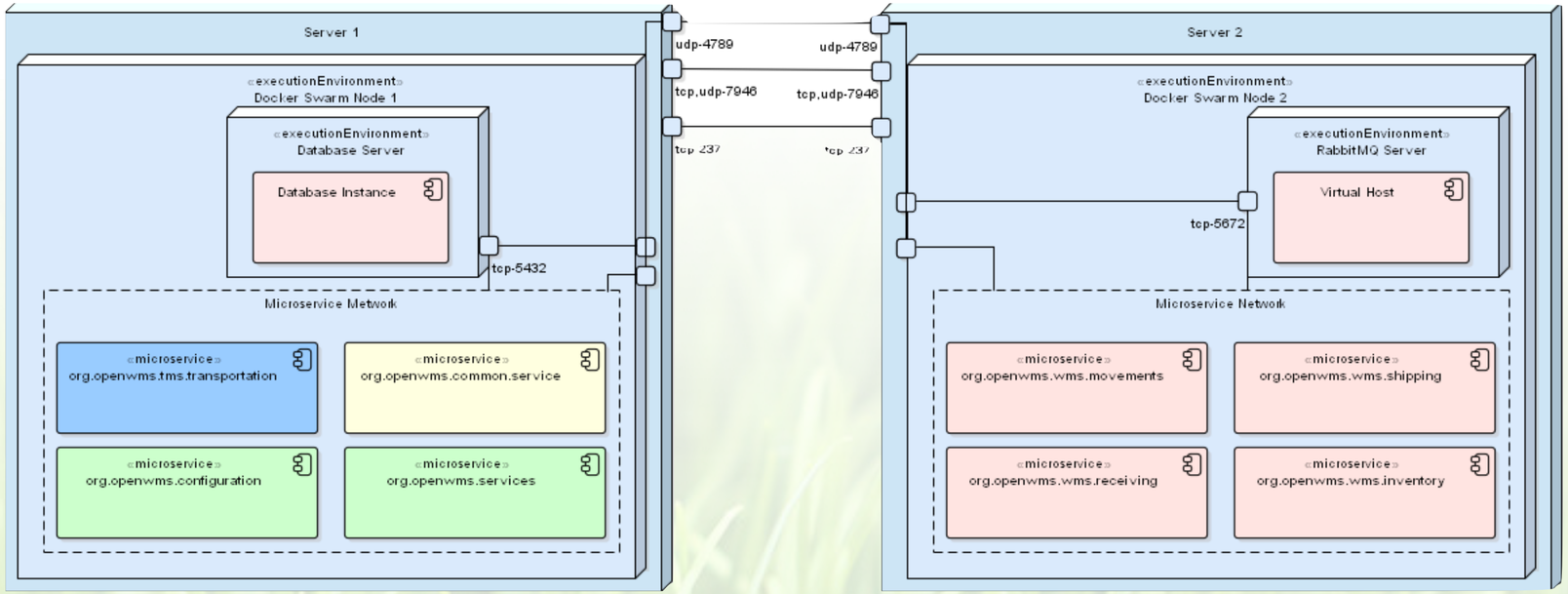
Microservices

Runtimes

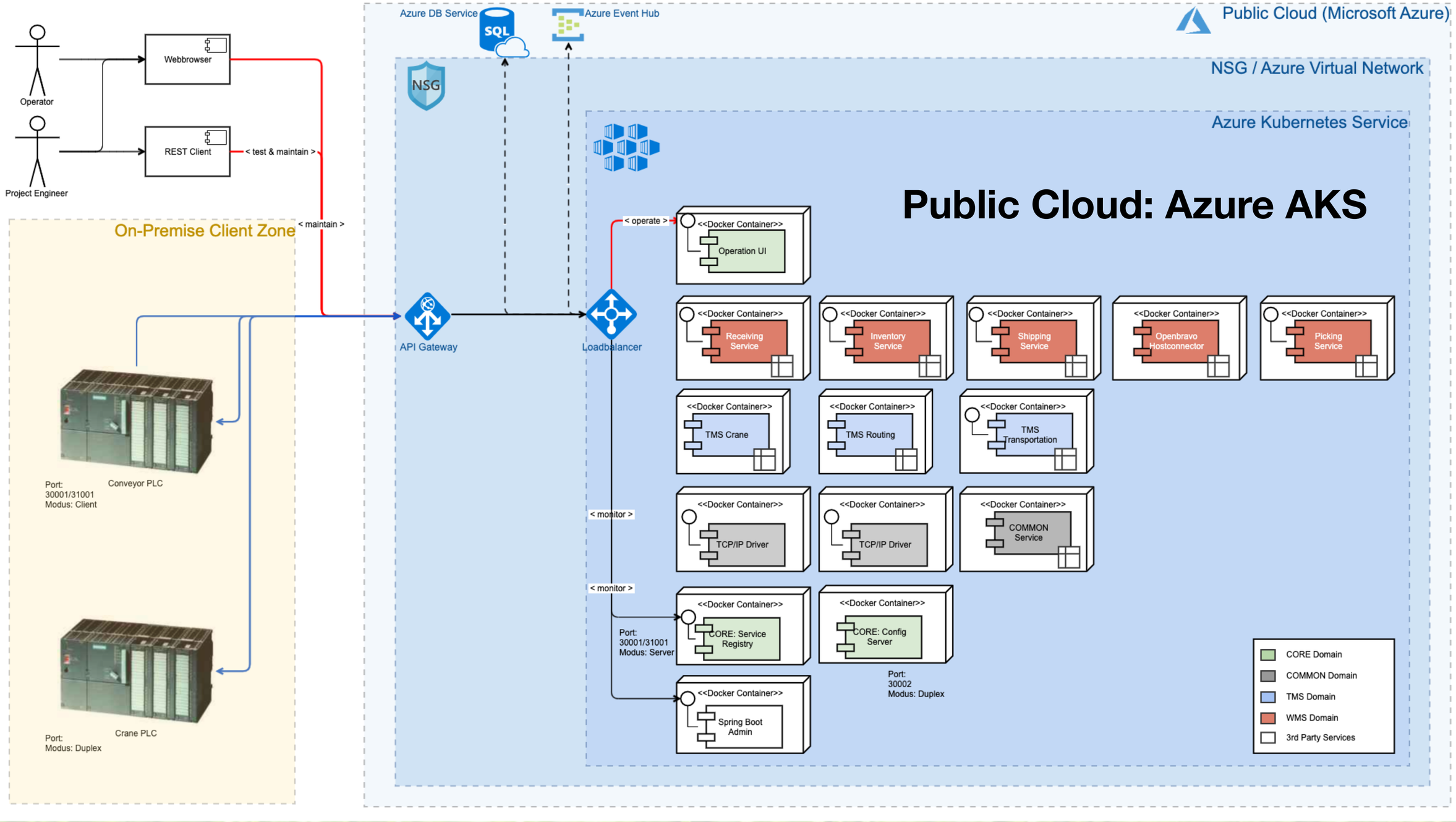
On-Premise: (Virtual) Server in Datacenter



Single Node



Multi Node



Microservices

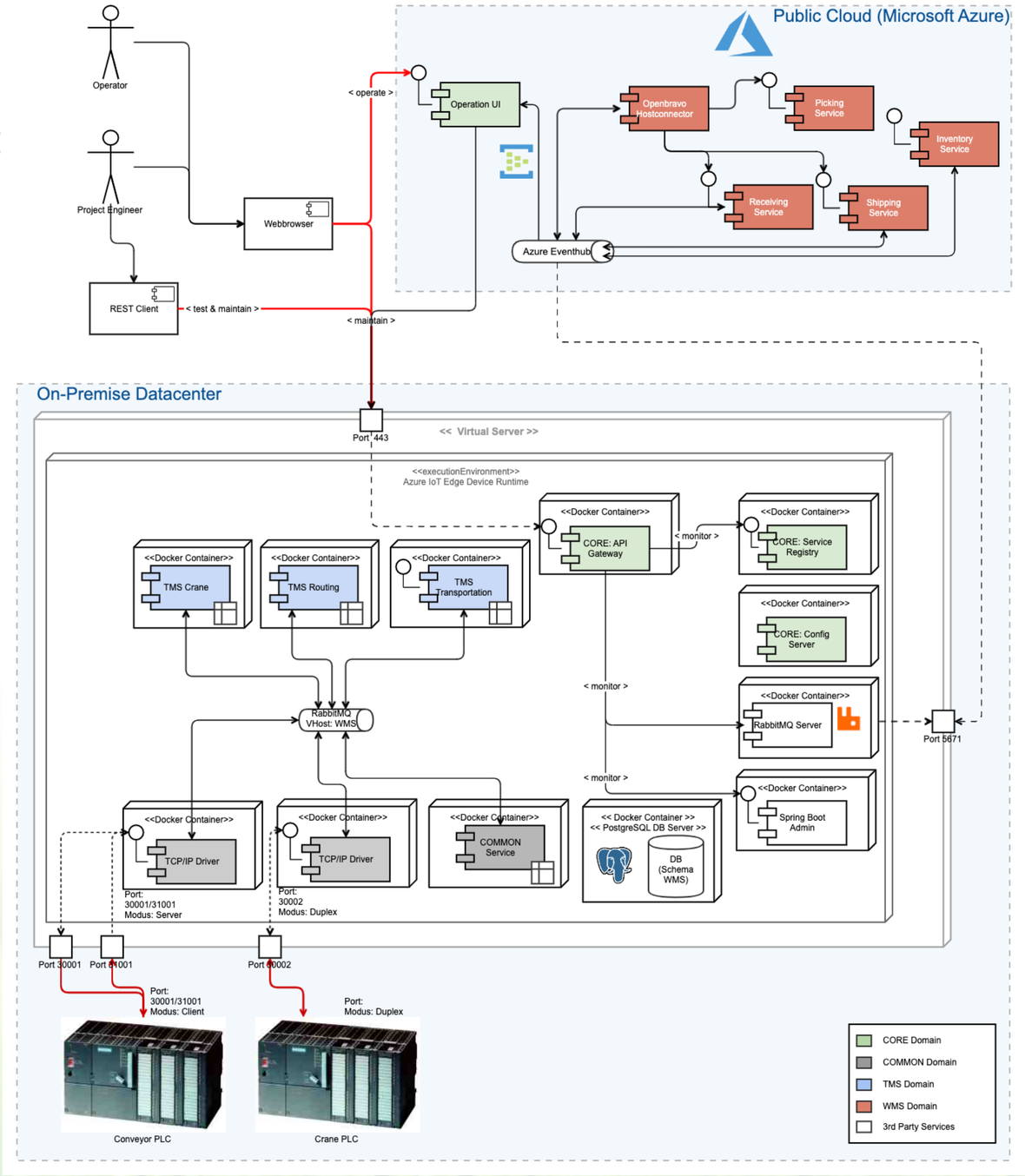
Runtimes

⊕ Throughput

Hybrid Cloud: Azure

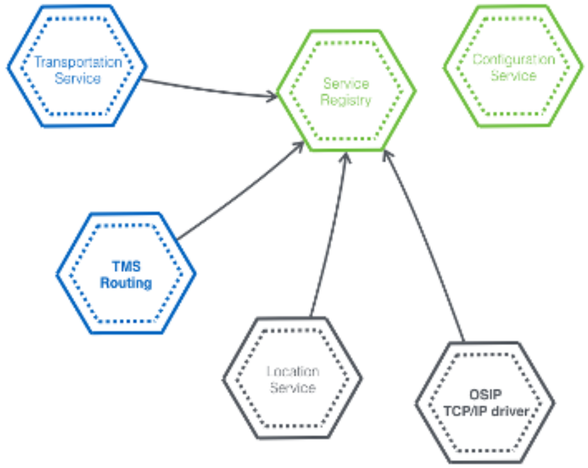
- Azure IoT Edge
- Azure IoT Device Runtime
- Azure Eventhub
- Azure Cosmos DB

Azure IoT Device Runtime is deployed on servers or IPC with less resources.
Workloads and services are managed with **Azure IoT Device Management**



Microservice Architecture

Infrastructure Services



- Service Registry (**Eureka**)
- Central Config Server (**Archaius**)
- Registry-First Approach (https://github.com/openwms/org.openwms/wiki/Secured-Eureka-First-services-on-Heroku)
- **Bootstrap.yml** required to load the initial bootstrap configuration of each service before the central configuration is fetched from Archaius

Microservice Architecture

Eureka

Eureka Server configuration

```yaml
eureka:
  client:
    register-with-eureka: false # default is true
    fetch-registry: false # default is true
    instance-info-replication-interval-seconds: 30 # default is 30
    service-url:
      defaultZone: ${owms.eureka.zone} # Must be camelCase
  server:
    wait-time-in-ms-when-sync-empty: 0 # default is 60000

    # The interval in which the instance eviction task scans for instances with expired leases.
    # Given in milliseconds.
    eviction-interval-timer-in-ms: 60000 # default is 60000

    # Switch off self-preservation. Will turn lease expiration on and evict all instances which no longer sent a heartbeat and whose lease has expired.
    # Self-preservation is desirable for Eureka clusters and where network outages (e.g. between data centers) could be possible.
    # Note: the lease validity / expiration is configured in the Eureka _client_ instances (see eureka.instance.lease-expiration-duration-in-seconds).
    enable-self-preservation: false # default is true

    # Make sure this is set to the same value as the lease renewal interval in Eureka _client_ instances (or slightly higher)
    # This value is relevant for Eureka's calculation of the 'current renewal threshold'.
    # Specifically, the following equation is used: current renewal threshold = (60s / expected-client-renewal-interval-seconds) * renewal-percent-threshold *
    # In this case:
    # - for one registered client: 60 / 3 * 0.5 * 1 = 10.
    # - for two registered clients: 60 / 3 * 0,5 * 2 = 20.
    # As soon as two clients are connected:
    expected-client-renewal-interval-seconds:  3

    renewal-percent-threshold: 0.49 # default is 0.85
    peer-node-read-timeout-ms: 1000 # default is 200
  instance:
    secure-port-enabled: false
    non-secure-port-enabled: true
    metadata-map:
      username: ${spring.security.user.name}
      password: ${spring.security.user.password}
      protocol: ${owms.srv.protocol}
      zone: ${owms.eureka.zone}
```

Custom metadata shared between services

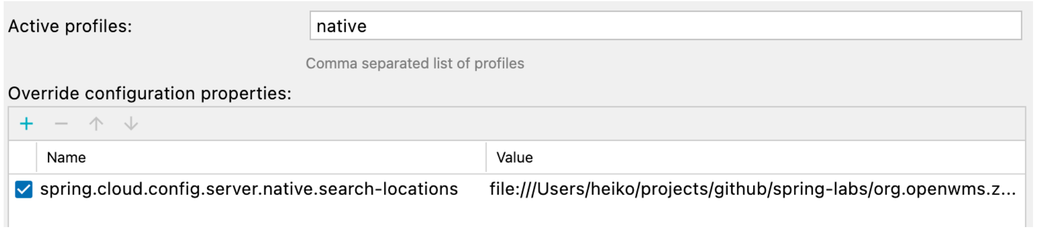
Microservice Architecture

Archaius

application.yml

```yaml
---
spring:
  config:
    activate:
      on-profile: native
  cloud:
    config:
      server:
        native:
          search-locations: classpath:config/
        bootstrap: true
  security:
    user:
      password: '{cipher}AQAVmwzi/bMLqH8bop5DYYZaVZM……='
```

By default: Pick up service configuration files from the config folder on the classpath



Override config location at runtime

docker-compose.yml

```yaml
cfg:
  image: "interface21/openwms-configuration:1.2.0"
  environment:
    spring.profiles.active: "native"
    spring.cloud.config.server.native.search-locations: file:///openwms/conf
    spring.rabbitmq.host: rabbitmq
    spring.zipkin.base-url: http://tracing:9411/
    owms.eureka.url: http://user:sa@srv:8761
    owms.srv.hostname: cfg
  volumes:
    - ./:/openwms
    - /etc/localtime:/etc/localtime:ro
  networks:
    - owms
```

Override in production. Mount the config folder into the Docker image and use a file reference
Also here: Override Eureka location, RabbitMQ hostname etc.

Microservice Security

Microservice Architecture

Infrastructure Security Services

- Access only permitted through API Gateway
- From Zuul 1 to new **Spring Cloud Gateway**
- Routes / URI are **centrally** managed and protected
- User Identity is extracted from token and propagated
- Authorization is centrally managed


```
spring:
  application:
 cloud:
    discovery:
      enabled: true
    gateway:
      globalcors:
        add-to-simple-url-handler-mapping: true
        corsConfigurations:
          '[/**]':
            allowCredentials: true
            allowedOrigins: "*"
            allowed-methods: "*"
            allowed-headers: "*"
      default-filters:
        - DedupeResponseHeader=Access-Control-Allow-Origin Ac
      httpclient:
        websocket:
          max-frame-payload-length: 2000000
        wiretap: true
      httpserver:
        wiretap: true
      discovery:
        locator:
          enabled: false  #creates default/factory routes bas
          lower-case-service-id: true
      routes:
      - id: common-service
        uri: lb://common-service
        predicates:
        - Path=/common/**
        filters:
        - StripPrefix=1
        - IdentityHeader
        - AuthorizationMatrix
```

Custom

Custom

Load-balanced

Custom
Authorisation
GatewayFilters

Microservice

API Gateway - Custom Grant / Path mapping

application.yml

```
owms:
  auth:
    mappings:
      - id: common-service
        paths:
          - path: /v1/locations/reset-plc-state
            verbs: post
            grants: access.stockmanagement.action.blockbins
          - path: /v1/targets/**
            verbs: post
            grants: access.resources.action.cranes, access.resources.action.workplace
          - path: /v1/ui/transport-units/filtered
            verbs: post
            grants: access.transportunits.screen
            log-action-exclude: true
          - path: /v1/transport-unit/inactivate
            verbs: patch
            grants: access.transportunit.actions.delete
          - path: /v1/transport-unit/inactivate/**
            verbs: patch
            grants: access.transportunit.actions.delete
```

Protected service name

Protected path with comma separated list of verbs and grants

Microservice

API Gateway - How to customise GatewayFilterFactory

```
public class AuthorizationMatrixGatewayFilterFactory extends
AbstractGatewayFilterFactory<AbstractGatewayFilterFactory.NameConfig>
{

}
```

Extend Springs AbstractGatewayFilterFactory

Microservice

API Gateway - How to customise GatewayFilterFactory

```
public class AuthorizationMatrixGatewayFilterFactory extends
AbstractGatewayFilterFactory<AbstractGatewayFilterFactory.NameConfig> {

private final AuthorizationMatrix matrix; // Spring Configuration
private Map<String, List<String>> grantMapping;


}
```

Inject the authorisation mappings from the configuration file (matrix)

Microservice

API Gateway - How to customise GatewayFilterFactory

```
public class AuthorizationMatrixGatewayFilterFactory extends
AbstractGatewayFilterFactory<AbstractGatewayFilterFactory.NameConfig> {

private final AuthorizationMatrix matrix; // Spring Configuration
private Map<String, List<String>> grantMapping;

@PostConstruct
void onPostConstruct() {
    grantMapping = new HashMap<>();
    for (Mapping m : matrix.getMappings()) {
        grantMapping.putAll(m.getPaths().stream()
                .flatMap(p -> p.getVerbs().stream()
                        .map(v -> v.toLowerCase() + p.getPath().toLowerCase())
                        .map(pv -> new Aggregate(m + pv, p.getGrants())))
                ).collect(Collectors.toMap(a -> a.id, a -> a.val)));
    }
}
}
```

Optional: Build an map from the configuration for convenience

Microservice

API Gateway - Custom GatewayFilterFactory

```
public class AuthorizationMatrixGatewayFilterFactory extends
AbstractGatewayFilterFactory<AbstractGatewayFilterFactory.NameConfig> {


…

@Override
public GatewayFilter apply(NameConfig config) {
    return new GatewayFilter() {
        @Override
        public Mono<Void> filter(ServerWebExchange exchange, GatewayFilterChain chain) {
    }};
}
```

Implement apply of GatewayFilterFactory that actually does the work and checks incoming requests

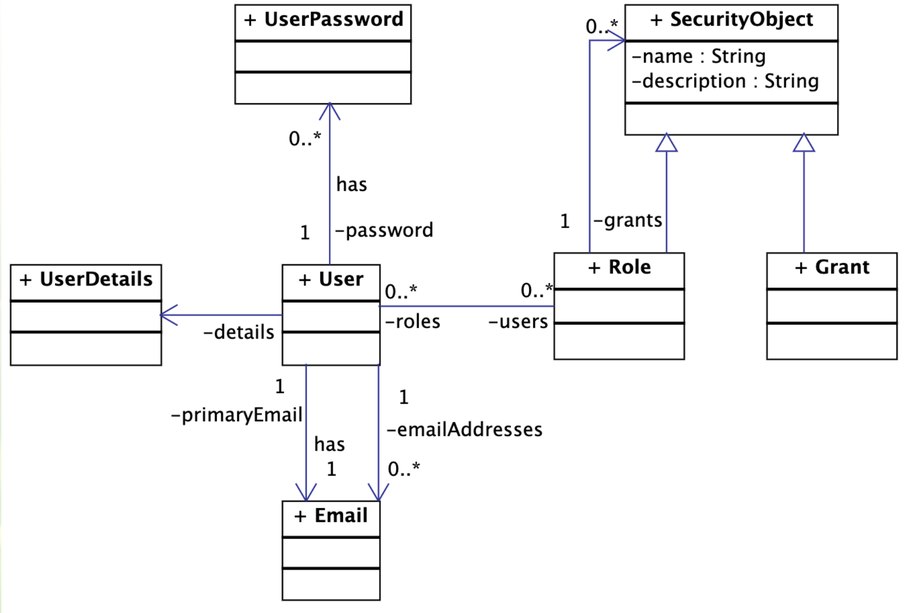
DEMO

Microservice Architecture

Infrastructure Services

- Custom **UAA Service** (User Authentication and Administration). Subset of **OAuth2** and **OIDC** endpoints to obtain tokens from plus administration endpoints for **Users**, **Roles**, **Clients** and **Grants**

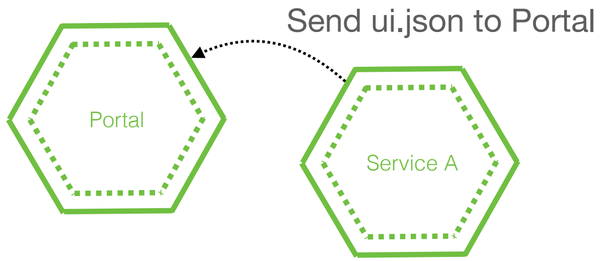
- **Portal** (EOL). Microservice with UI based on **single-spa**. Starting services connect to the Portal and announce their own UI Widgets and menu entry points



https://single-spa.js.org

Microservice Architecture

Infrastructure Services



- **Portal** (EOL). Microservice with UI based on **single-spa**. Starting services connect to the Portal and announce their own UI widgets and menu entry points
- The Portal populates the menu bar and routes
- Advantage: Microfrontend approach. Each micro service contains its own set of screens
- Downside: The microservice UI projects must be ***ejected*** to work with single-spa

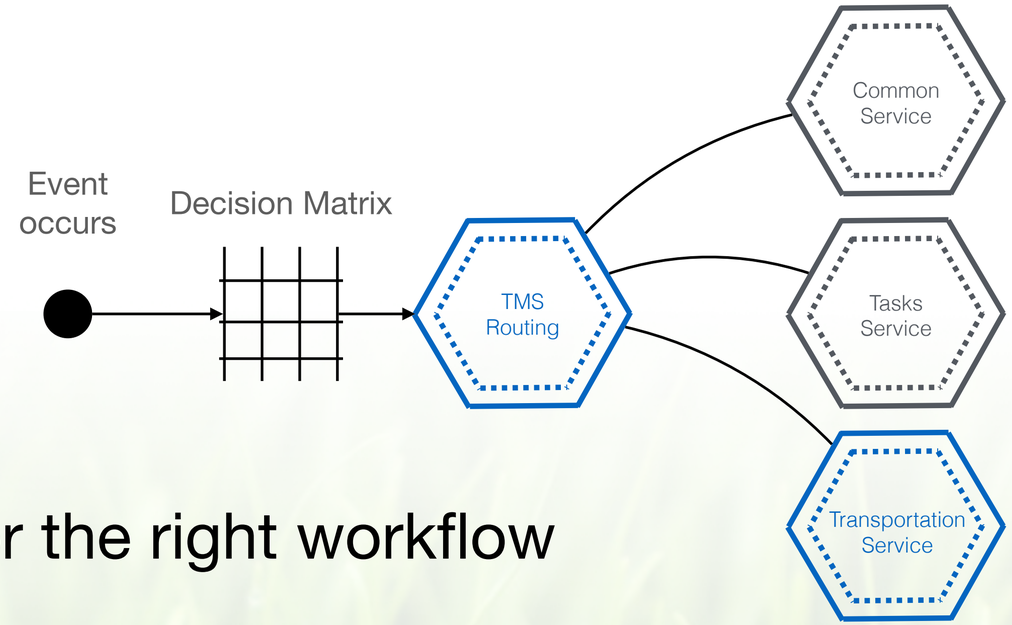
```
[
  {
    "module": "TMS",
    "name": "Transport Management",
    "desc": "Transport Management",
    "version": "0.1",
    "active": true,
    "thumb": “../g
    "hash": "#/tms",
    "path": "tms/static/js/singleSpaEntry.js",
    "storeUrl": "tms/static/js/store.js",
    "menuItems": [
      {
        "name": "Routes",
        "route": "/tms/routes",
        "exact": true
      },
      {
        "name": "Actions",
        "route": "/tms/actions",
        "exact": true
      }
    ]
  }
]
```

https://single-spa.js.org

Microservice Architecture

Workflow Integration to orchestrate microservices

- Activity, Flowable, Camunda Workflow Engines bootstrapped in-memory
- Standard BPMN & DMN



  - Event occurs
  - A CallContext is created
  - Decision Matrix is asked for the right workflow
  - Workflow is chosen and parameterised
  - Workflow is executed

Efficiency

Microservice Architecture

Workflow

- WFE Bootstrapping (camunda-bpmn-spring-boot-starter, flowable-spring-boot-starter)
- Microservice Execution
- Designer

DEMO

Huge benefit:

Be fast and efficient in project realisation with reusable functional building blocks and workflows without coding if-else logic

Efficiency

Microservice Architecture

Workflow - Microservice execution

- Find the workflow definition and start execution

```
var processDefinition = repositoryService.createProcessDefinitionQuery().processDefinitionKey(program.getProgramKey()).active().latestVersion().singleResult();
..
runtimeService.startProcessInstanceById(processDefinition.getId(), runtimeVariables);
```

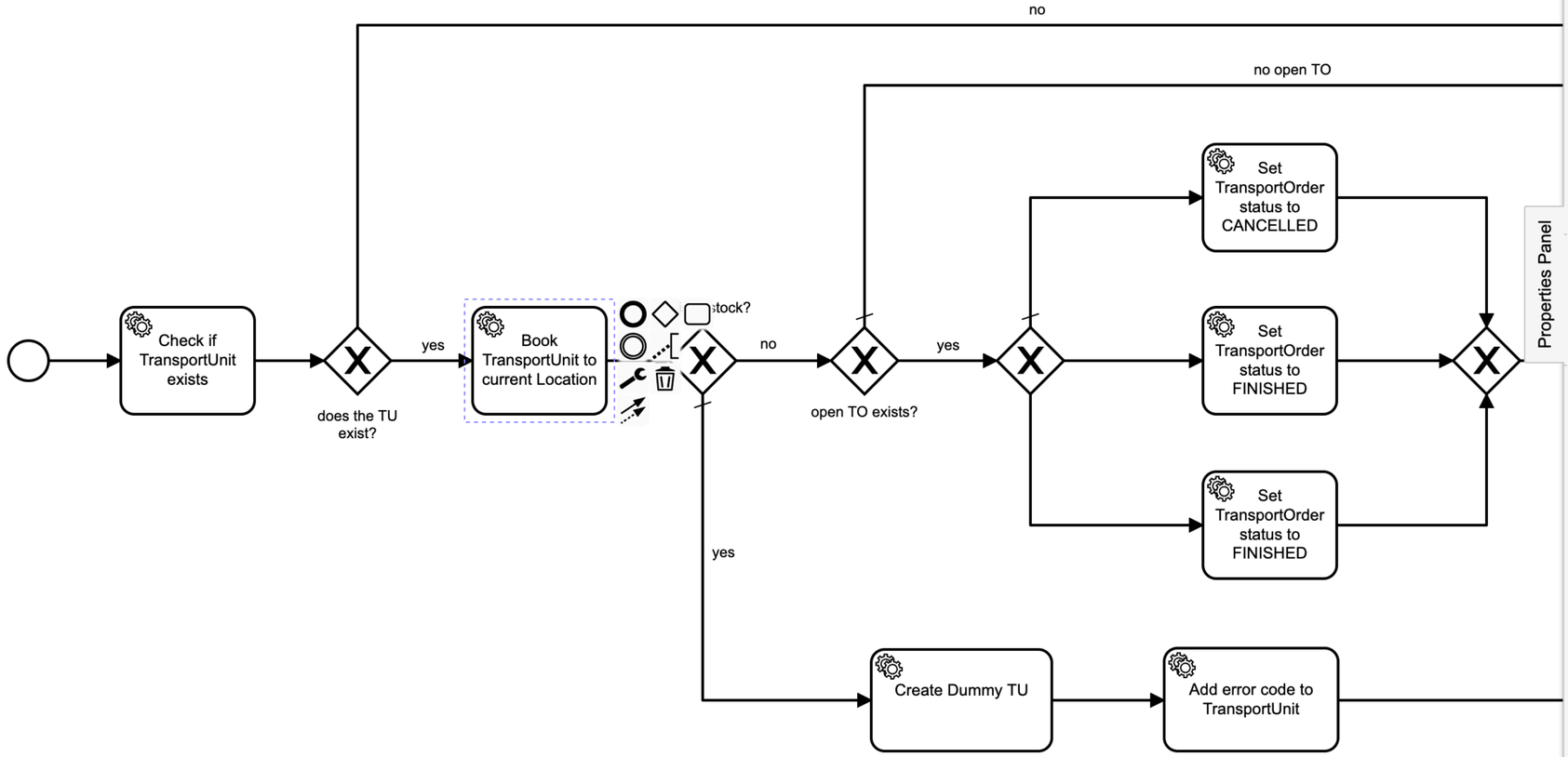
- A workflow consists of multiple BPMN Service Tasks. Each one points to a Feign client execution with JUEL syntax:

```
#{transportUnitApi.createTU(in.barcode, in.actualLocation, "PALLET", false)}
```

Feign client bean

Variable taken from the CallContext

WF023-GearControl.bpmn



Properties Panel

ServiceTask_1h382ks

General | Listeners | Input/Output | Field Injections | Extensions

General

Id

ServiceTask_1h382ks

Name

Book TransportUnit to current Location

Details

Implementation

Expression

Expression

#{transportUnitService.moveTU(in.msg.barcode, in.msg.actualLocationId)}

Result Variable

isMoved

Asynchronous Continuations

- [ ] Asynchronous Before
- [ ] Asynchronous After

Documentation

Element Documentation

Microservice Architecture

Commands & Events

- **Event**: Something has happened, inform others about it
- **Command**: Request to do something for me
- Part of the asynchronous messaging, realised with RabbitMQ
- RabbitMQ **Exporters** (AmqpTemplate) can be easily configured with **Spring Profiles**

Spring
Exporter Pattern

Spring Profiles

- Makes it possible to use also other technologies like Kafka, MQTT etc.

Microservice Architecture

Commands & Events: Propagating Events

```java
@Measured
public void changeGroupState(@NotBlank String pKey, @NotNull LocationGroupState stateIn, @NotNull LocationGroupState stateOut) {
    var locationGroup = repository.findBypKey(pKey).orElseThrow(() -> new NotFoundException(
            translator, LOCATION_GROUP_NOT_FOUND_BY_PKEY, new String[]{pKey}, pKey
    ));
    locationGroup.changeState(stateIn, stateOut);
    ctx.publishEvent(LocationGroupEvent.of(locationGroup, LocationGroupEvent.LocationGroupEventType.STATE_CHANGE));
}
```

Sending an **internal** Event as part of the current business operation

```java
@TransactionalEventListener(fallbackExecution = true)
public void onEvent(LocationGroupEvent event) {
    switch (event.getType()) {
        case CREATED -> amqpTemplate.convertAndSend(exchangeName, "lg.event.created", locationGroupMapper.convertToMO((LocationGroup) event.getSource()));
        case CHANGED -> amqpTemplate.convertAndSend(exchangeName, "lg.event.changed", locationGroupMapper.convertToMO((LocationGroup) event.getSource()));
        case DELETED -> amqpTemplate.convertAndSend(exchangeName, "lg.event.deleted", locationGroupMapper.convertToMO((LocationGroup) event.getSource()));
        case STATE_CHANGE -> {
            LocationGroupMO msg = locationGroupMapper.convertToMO((LocationGroup) event.getSource());
            validate(validator, msg);
            amqpTemplate.convertAndSend(exchangeName, "lg.event.state-changed", msg);
        }
        default -> throw new UnsupportedOperationException(format("LocationGroupEvent [%s] currently not supported", event.getType()));
    }
}
```

Executed when the database transaction succeeds

Lightweight variant of : https://microservices.io/patterns/data/transactional-outbox.html

Microservice Architecture

Commands & Events: Sending Commands

```
@Measured
@RabbitListener(queues = "${owms.commands.common.loc.queue-name}")
public void onCommand(@Payload LocationCommand command) {
   if (LocationCommand.Type.SET_LOCATION_EMPTY == command.getType()) {
        validate(validator, command, ValidationGroups.SetLocationEmpty.class);
        var errorCode = ErrorCodeVO.LOCK_STATE_IN_AND_OUT;
        errorCode.setPlcState(LOCATION_EMPTY);
        locationService.changeState(command.getLocation().pKey(), errorCode);
    }
}
```

Receiver validates and handles the command

What if event/command handling fails on receiving?

- Retry execution
- Dead Letter Queuing -> Reprocessing with **RabbitMQ Shovel** plugin
- Client-side retry

Microservice Architecture

Async. Configuration

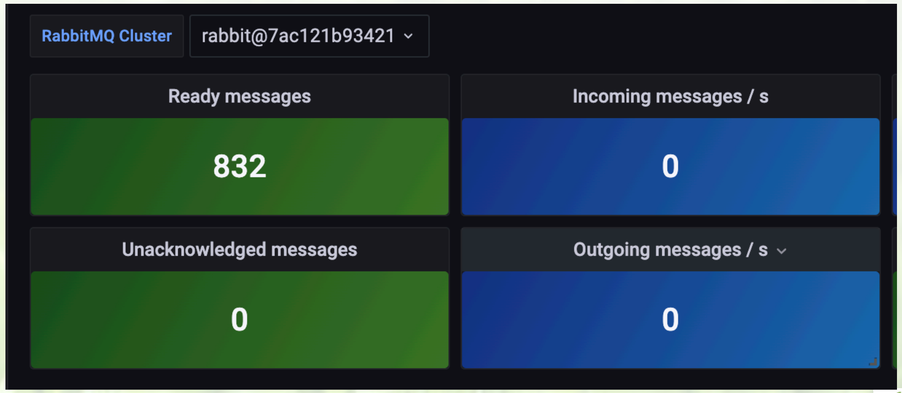
```
@RefreshScope
@Bean
Queue commandsQueue(@Value("${owms.commands.common.tu.queue-name}") String queueName,
        @Value("${owms.common.dead-letter.exchange-name}") String exchangeName) {
    return QueueBuilder.durable(queueName)
            .withArgument("x-dead-letter-exchange", exchangeName)
            .withArgument("x-dead-letter-routing-key", POISON_MESSAGE)
            .build();
}
```

Typical AMQP Queue configuration with a dead-letter binding

```
@RefreshScope
@Bean Binding dlBinding(
        @Value("${owms.common.dead-letter.queue-name}") String queueName,
        @Value("${owms.common.dead-letter.exchange-name}") String exchangeName) {
    return BindingBuilder
            .bind(dlQueue(queueName))
            .to(dlExchange(exchangeName))
            .with(POISON_MESSAGE);
}
```

Dead-letter binding

Dead-letter must be monitored!




```
---
spring:
  config:
    activate:
      on-profile: ASYNCHRONOUS
  rabbitmq:
    listener:
      simple:
        retry:
          max-attempts: 3
          enabled: true
          initial-interval: 1s
          max-interval: 1s
```

Retry Configuration on receiving side

Microservice Architecture

Async. Configuration

Retry configuration on sender side

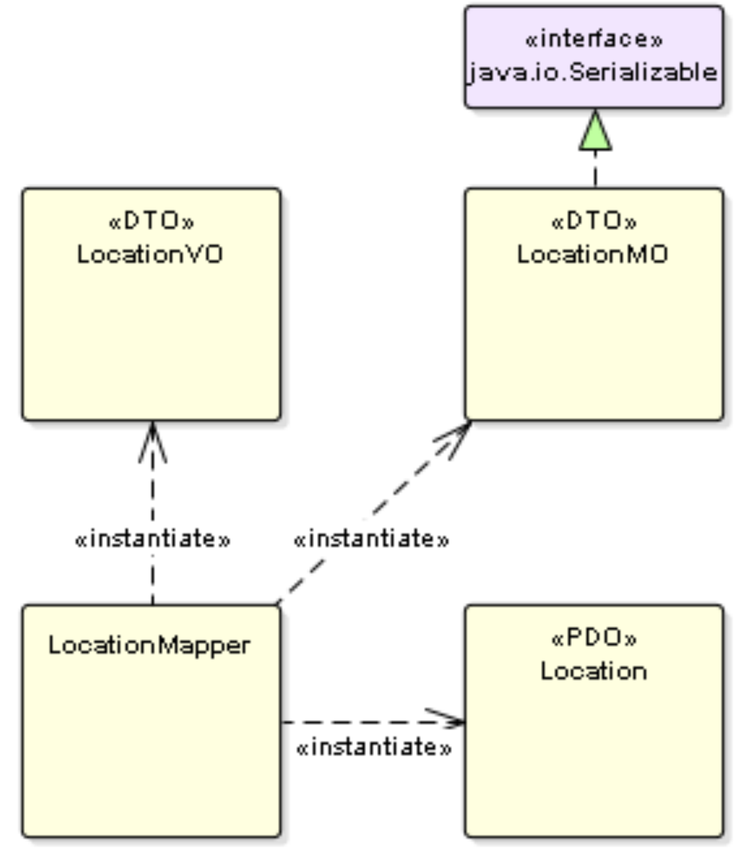
```
@Primary
@Bean(name = "amqpTemplate")
public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory,
        ObjectProvider<MessageConverter> messageConverter,
        @Autowired(required = false) RabbitTemplateConfigurable rabbitTemplateConfigurable) {
    var rabbitTemplate = new RabbitTemplate(connectionFactory);
    var backOffPolicy = new ExponentialBackOffPolicy();
    backOffPolicy.setMultiplier(2);
    backOffPolicy.setMaxInterval(15000);
    backOffPolicy.setInitialInterval(500);
    RetryTemplate retryTemplate = new RetryTemplate();
    retryTemplate.setBackOffPolicy(backOffPolicy);
    rabbitTemplate.setRetryTemplate(retryTemplate);
    rabbitTemplate.setMessageConverter(messageConverter.getIfUnique());
    if (rabbitTemplateConfigurable =≠ null) {
        rabbitTemplateConfigurable.configure(rabbitTemplate);
    }
    return rabbitTemplate;
}
```

Attach a RetryTemplate to the RabbitTemplate

A custom RabbitTemplateConfigurable propagates the CallContext with the outgoing message

Microservice Design

Mapping & Models



- In use: **Apache Dozer** & **MapStruct**
- 2-3 models exist
- Persistent Domain Object: **Location**
- View Object: **LocationVO** built for the REST API
- Message Object: **LocationMO** built for serialisation with AMQP
- Jackson/JPA annotations are not mixed! **Lombok** is never used, but Java Records. Lombok had issues together with AspectJ

Microservice Design

Mapping & Models

	Pros	Cons
Apache Dozer	• Doesn't require accessors at all • Took over by the community and pushes new releases • Custom mappers can be implemented • Independent from the used application framework (Spring, CDI, …)	• Reflection is used, less performant • Issues with AOT compiling • Most is done with XML • Small community
MapStruct	• Spring managed - leverages DI • Annotation driven • Ready for AOT	• Requires accessors • Another Maven plugin. May bite with QueryDSL, Lombok and other pre-compiler plugins

Microservice Architecture

Logging & Tracing

- Slf4j & Logback with logstash-logback-encoder
- In Production: **Push-to-ELK** vs. **Filebeat**
- @Measured + @MeasuredRestController. A Spring AOP advice measures execution time with a StopWatch and prints to TSL logs

```
/** Map methods with Measured. */
@Pointcut("@within(org.ameba.http.MeasuredRestController) || @annotation(org.ameba.annotation.Measured)")
public void isMeasured() {}
```

Logging & Tracing

Tenant-aware logs

Appenders for **tenant-aware** trace files, exception logs & TSL (technical service logs) using Logbacks SiftingAppender

WebFilter populates the CallContext with the Tenant, TenantDiscriminator populates the MDC

```
<appender name="LOGFILE" class="ch.qos.logback.classic.sift.SiftingAppender">
    <discriminator class="org.ameba.logging.TenantDiscriminator" />
    <sift>
        <appender name="FILE-${Tenant}" class="ch.qos.logback.core.rolling.RollingFileAppender">
            <file>${LOG_PATH}/${Tenant}-${MODULE_NAME}.log</file>
            <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
                <fileNamePattern>${LOG_PATH}/${Tenant}-${MODULE_NAME}.%i.log.gz</fileNamePattern>
                <minIndex>1</minIndex>
                <maxIndex>10</maxIndex>
            </rollingPolicy>

            <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
                <maxFileSize>10MB</maxFileSize>
            </triggeringPolicy>

            <encoder>
                <pattern>${Tenant} ${MODULE_NAME} %d{YYYY-MM-dd HH:mm:ss.SSS} %X{X-RequestID} %-5level %clr([${PID:- }/%tid]){magenta} %clr(-){faint}
[%-40.40logger{39}]${mdcPattern} : %msg%n</pattern>
            </encoder>
        </appender>
    </sift>
</appender>
```

Logging & Tracing

Appender to push logs to ELK

```
<appender name="ELK" class="net.logstash.logback.appender.LogstashTcpSocketAppender">
    <destination>elk:5000</destination>
    <encoder class="net.logstash.logback.encoder.LogstashEncoder"/>
</appender>
```

- **Downside of Push-to-ELK** approach: If ELK is down, the logs remain in the micro service Docker container and the container file grows steadily

Logging & Tracing

Spring Profile dependent log configuration

```
<springProfile name="PROD">
    <logger name="PRESENTATION_LAYER_EXCEPTION" level="ERROR"/>
    <logger name="SERVICE_LAYER_EXCEPTION" level="ERROR"/>
    <logger name="INTEGRATION_LAYER_EXCEPTION" level="ERROR"/>
    <logger name="CALLCONTEXT" level="OFF"/>
    <logger name="MEASURED" level="INFO"/>
    <root level="INFO">
        <appender-ref ref="ELK"/>
    </root>
</springProfile>
<springProfile name="!PROD">
<logger name="PRESENTATION_LAYER_EXCEPTION" level="ERROR" additivity="false">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="EXCFILE"/>
    <appender-ref ref="LOGFILE"/>
</logger>
..
<logger name="CALLCONTEXT" level="DEBUG"/>
    <logger name="MEASURED" level="INFO">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="TSL"/>
</logger>
<root level="INFO">
    <appender-ref ref="STDOUT"/>
    <appender-ref ref="LOGFILE"/>
</root>
</springProfile>
```

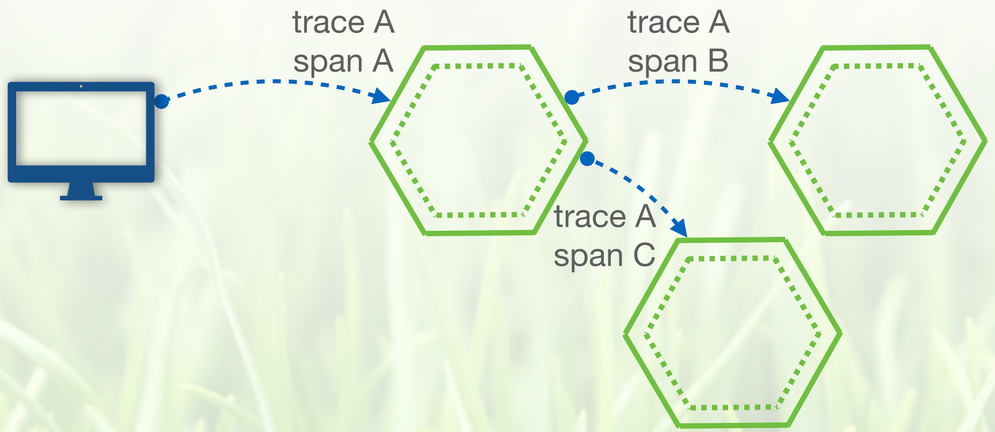
Spring Profiles

Microservice Architecture

CallContext Propagation

```
FROM openzipkin/zipkin:2.21.5
ENV TZ=CET
ENV STORAGE_TYPE=elasticsearch
ENV ES_HOSTS=elk:9200
ENV ES_USERNAME=foo
ENV ES_PASSWORD=bar
```

- Callgraph tracing with Sleuth, visualised with Zipkin
- Additional Interceptors for **Feign**, **RestTemplate** & **RabbitTemplate** to propagate **CallContext**, **User identity** etc.



DEMO

Microservice Architecture

CallContext Propagation

- Populate Tenant & Identity via Feign (same for CallContext in different class)

```
public class FeignClientInterceptor implements RequestInterceptor {

   @Override
    public void apply(RequestTemplate template) {
        TenantHolder.setCurrentTenant((d) -> template.header(Constants.HEADER_VALUE_X_TENANT, d));
        IdentityContextHolder.setCurrentIdentity((d) ->  template.header(Constants.HEADER_VALUE_X_IDENTITY, d));
    }
}
```

Microservice Architecture

CallContext Propagation

- Populate Identity via RestTemplate (same for CallContext in different class)

```
public class IdentityClientRequestInterceptor implements BaseClientHttpRequestInterceptor {

    @Override
    public ClientHttpResponse intercept(HttpRequest request, byte[] body, ClientHttpRequestExecution execution) throws IOException {
        IdentityContextHolder.currentIdentity().ifPresent(s -> request.getHeaders().add(Constants.HEADER_VALUE_X_IDENTITY, s));
        return execution.execute(request, body);
    }
}
```

- Configure it in MVC

```
@ExcludeFromScan
@ConditionalOnWebApplication
@Configuration
public class WebMvcConfiguration implements WebMvcConfigurer {

   public @Bean
    List<BaseClientHttpRequestInterceptor> baseRestTemplateInterceptors() {
        return new ArrayList<>(asList(
                new CallContextClientRequestInterceptor(),
                new IdentityClientRequestInterceptor()
        ));
    }
}
```

Microservice Architecture

CallContext Propagation

- Populate CallContext, Identity, Tenant via RabbitTemplate

```
@Primary
@Bean(name = "amqpTemplate")
public RabbitTemplate rabbitTemplate(ConnectionFactory connectionFactory,
        ObjectProvider<MessageConverter> messageConverter,
        @Autowired(required = false) RabbitTemplateConfigurable rabbitTemplateConfigurable) {
    var rabbitTemplate = new RabbitTemplate(connectionFactory);
..
    if (rabbitTemplateConfigurable =≠ null) {
        rabbitTemplateConfigurable.configure(rabbitTemplate);
    }
    return rabbitTemplate;
}
```

- The RabbitTemplateConfigurable has a List of MessageHeaderEnhancers, each one add its own headers to outgoing messages

```
class IdentityEnhancer implements MessageHeaderEnhancer {

    @Override
    public void enhance(final RabbitTemplate rabbitTemplate) {
        rabbitTemplate.addBeforePublishPostProcessors(
            m → {
                if (IdentityContextHolder.currentIdentity().isPresent()) {
                    m.getMessageProperties().getHeaders().put("owms_identity", IdentityContextHolder.getCurrentIdentity());
                }
                return m;
            }
        );
    }
}
```

Microservice Architecture

Caching

- Feign client interfaces with **@Cacheable** annotations

```
@GetMapping(value = API_LOCATIONS, params = {"locationGroupNames"}, produces = LocationVO.MEDIA_TYPE)
@Cacheable("locations")
List<LocationVO> findForLocationGroups(@RequestParam("locationGroupNames") List<String> locationGroupNames);
```

- Change Listener

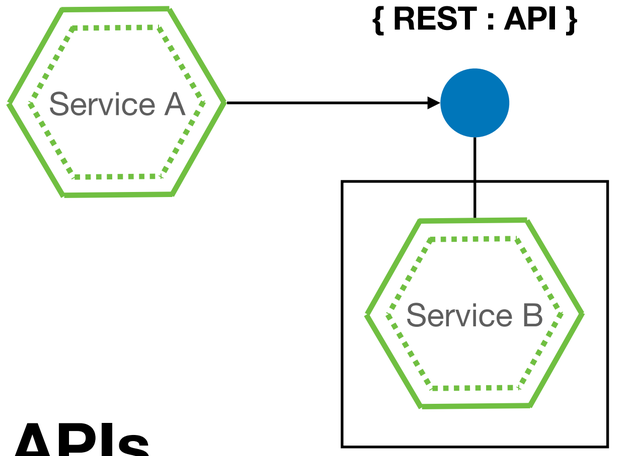
```
@Measured
@RabbitListener(queues = "${owms.events.common.lg.queue-name}")
public void onLocationGroupEvent(@Payload LocationGroupMO msg, @Header("amqp_receivedRoutingKey") String routingKey) {
..
    switch (routingKey) {
        case "lg.event.boot" -> {
            LOGGER.debug("[Cache] Location(Group) master data source has restarted -> evicting cache");
            janitor.evictLocationGroupCache();
            janitor.evictLocationCache();
        }
        case "lg.event.changed" -> {
            LOGGER.debug("[Cache] LocationGroup has changed -> evicting cache");
            janitor.evictLocationGroupCache();
        }
..
    }
}
```

Janitor

```
@CacheEvict(cacheNames = "locationGroups", allEntries = true)
public void evictLocationGroupCache() { }
```

Microservice Architecture

Client-JARs

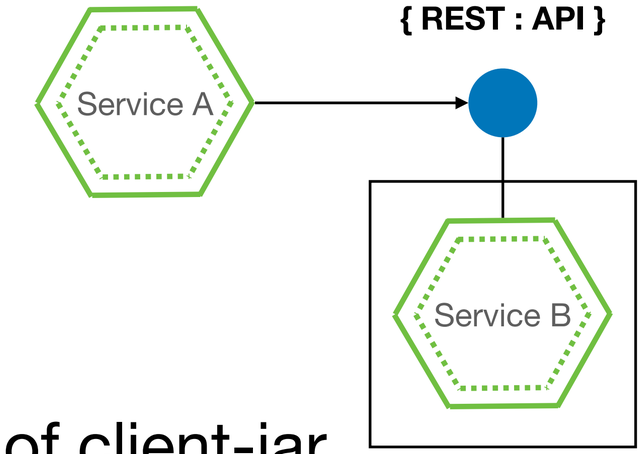


- Loose coupling between Microservices with **evolutionary APIs**
- Service A has a **limited view** of interest of Service B API
- Service A uses a **client-side Feign API** with its own data classes to access Service B API
- Changes in Service B API could reflect a bunch of Feign clients
- Why does Service B not provide the client-side stub to all clients?
- Service B could build and ship this stub with every successful build

To be discussed

Microservice Architecture

Client-JARs



Service B: Provider of client-jar

```
<plugin>
    <artifactId>maven-jar-plugin</artifactId>
    <executions>
        <execution>
            <id>client-build</id>
            <phase>package</phase>
            <goals>
                <goal>jar</goal>
            </goals>
            <configuration>
                <classifier>client</classifier>
                <includes>
                    <include>**/api/**/*.class</include>
                </includes>
            </configuration>
        </execution>
        <execution>
            <id>full-build</id>
            <phase>package</phase>
            <goals>
                <goal>jar</goal>
            </goals>
            <configuration>
                <classifier>all</classifier>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Service A: Consumer of client-jar

```
<dependency>
    <groupId>org.openwms</groupId>
    <artifactId>org.openwms.common.service</artifactId>
    <classifier>client</classifier>
</dependency>
```

Microservice Design

Extending SpringBoot services

*<u>Scenario:</u> "A SpringBoot application that is kind of a **Product** needs to be reused in projects as a **"base" application** and **must be extended**"*

Microservice Design

Extending SpringBoot services

- Extending the **classpath** at **runtime**? https://github.com/openwms/org.openwms/wiki/Extend-Base-Services-in-Projects

```
java -cp openwms-common-service-exec.jar
-Dloader.main=org.openwms.common.CommonStarter
-Dloader.path=project.jar
org.springframework.boot.loader.PropertiesLauncher
```

Different Loader

Microservice Design

Extending SpringBoot services

- Extending the **Docker** image at **build time**? https://github.com/openwms/org.openwms/wiki/Extend-Base-Services-in-Projects---Part-II-with-Spring-Boot-2.3

base image

```
FROM adoptopenjdk/openjdk11-openj9:jre-11.0.7_10_openj9-0.20.0-alpine as builder
WORKDIR application
ARG JAR_FILE=target/openwms-common-service-exec.jar
COPY ${JAR_FILE} application.jar
RUN java -Djarmode=layertools -jar application.jar extract

FROM adoptopenjdk/openjdk11-openj9:jre-11.0.7_10_openj9-0.20.0-alpine
WORKDIR application
COPY --from=builder application/dependencies/ ./
COPY --from=builder application/spring-boot-loader/ ./
COPY --from=builder application/snapshot-dependencies/ ./
COPY --from=builder application/application/ ./
ENTRYPOINT ["java", "-Xshareclasses -Xquickstart -noverify", "o.s.b.l.JarLauncher"]
```

extended image

```
FROM openwms/org.openwms.common.service:latest
ARG JAR_FILE=target/openwms-wms-putaway.jar
COPY ${JAR_FILE} BOOT-INF/lib/openwms-wms-putaway.jar
ENTRYPOINT ["java", "-Xshareclasses -Xquickstart -noverify", "o.s.b.l.JarLauncher"]
```

Microservice Design

Extending SpringBoot services

=> Leads to issues during development. The customised application must already work at development time in the IDE

- Solution: Work with additional build artefacts: **-exec.jar, -all.jar**
- The wrapping and extending application has a dependency to the base JAR (all)

Plain JAR

```
<dependency>
   <groupId>org.openwms</groupId>
   <artifactId>org.openwms.common.service</artifactId>
   <classifier>all</classifier>
</dependency>
```

SpringBoot executable

```
<dependency>
   <groupId>org.openwms</groupId>
   <artifactId>org.openwms.common.service</artifactId>
   <classifier>exec</classifier>
</dependency>
```

Microservice Design

Extending SpringBoot services

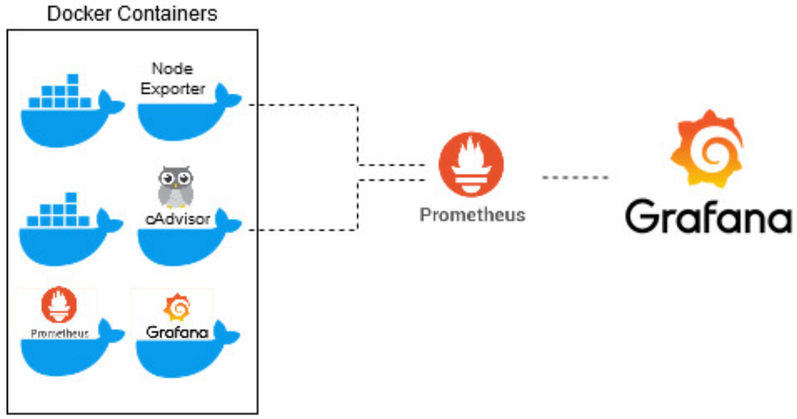
```
<plugin>
    <artifactId>maven-jar-plugin</artifactId>
    <executions>
        <execution>
            <id>client-build</id>
            <phase>package</phase>
            <goals>
                <goal>jar</goal>
            </goals>
            <configuration>
                <classifier>client</classifier>
                <includes>
                    <include>**/api/**/*.class</include>
                </includes>
            </configuration>
        </execution>
        <execution>
            <id>full-build</id>
            <phase>package</phase>
            <goals>
                <goal>jar</goal>
            </goals>
            <configuration>
                <classifier>all</classifier>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Building the **all** and **exec** jars

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <executions>
        <execution>
            <id>repackage</id>
            <goals>
                <goal>build-info</goal>
                <goal>repackage</goal>
            </goals>
            <configuration>
                <classifier>exec</classifier>
            </configuration>
        </execution>
    </executions>
    <configuration>
        <layers>
            <configuration>${project.basedir}/src/layers.xml</configuration
            <enabled>true</enabled>
        </layers>
    </configuration>
</plugin>
```

Microservice Architecture

Monitoring



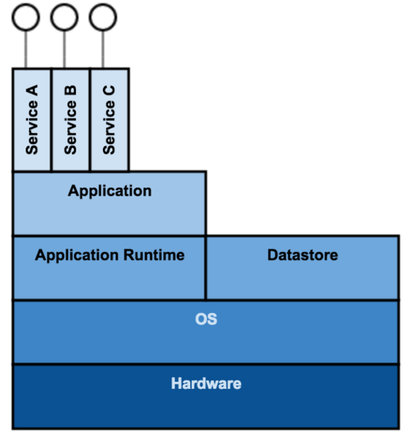
- Monitoring a distributed system in production is essential and should be considered early in a project
- In OpenWMS.org we use
  - **ELK** for log aggregation and business monitoring
  - **Prometheus**, **Grafana**, **cAdvisor**, **AlertManager**, **Node Exporter**, **Postgres Exporter** for technical monitoring
  - **Zipkin** for trace analysis
  - **SpringBoot Admin** for basic service observability and log management
  - **Custom Transaction Service** that is asynchronously fed with business transactions

DEMO

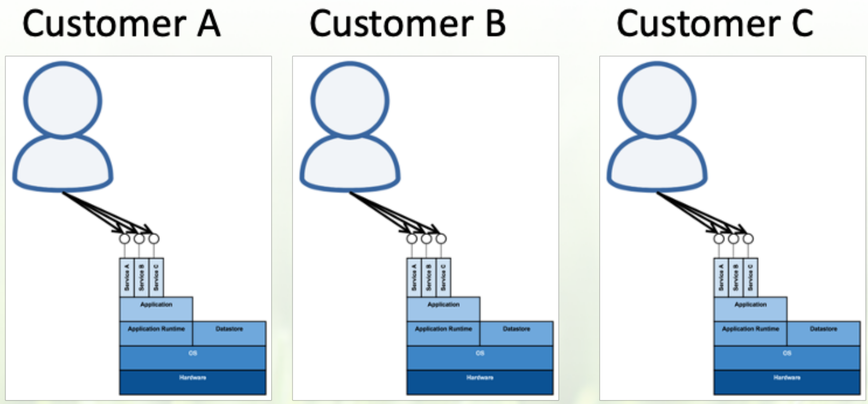
Microservice Architecture

Multitenancy

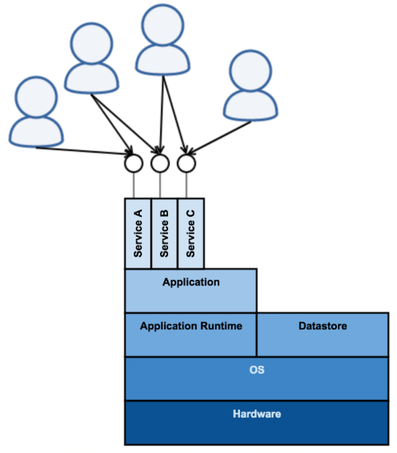
Traditional Stack



Dedicated instances per customer



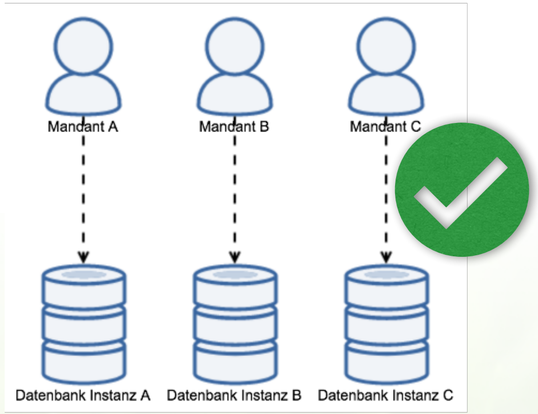
Shared instance



Microservice Architecture

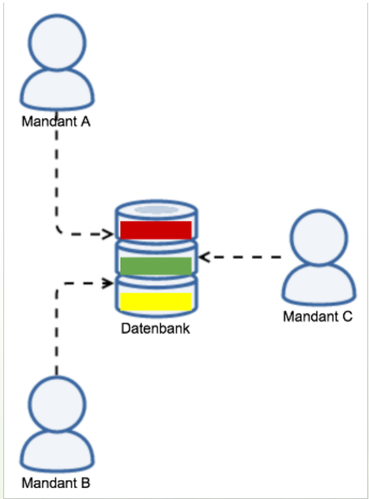
Multitenancy

- In a shared instance data must be strictly separated by tenant
- Log separation is done with tenant-aware log appenders
- Datastore separation:



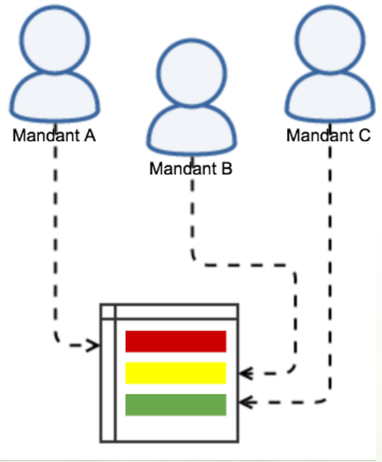
<u>DB instance per tenant</u>

Each client has its dedicated service that bootstraps fast enough does the work and shuts down



<u>DB schema per tenant</u>

One service may run for a longer time but must manage several DataSources (resource consumption increases!)



<u>Table separation with discriminator</u>

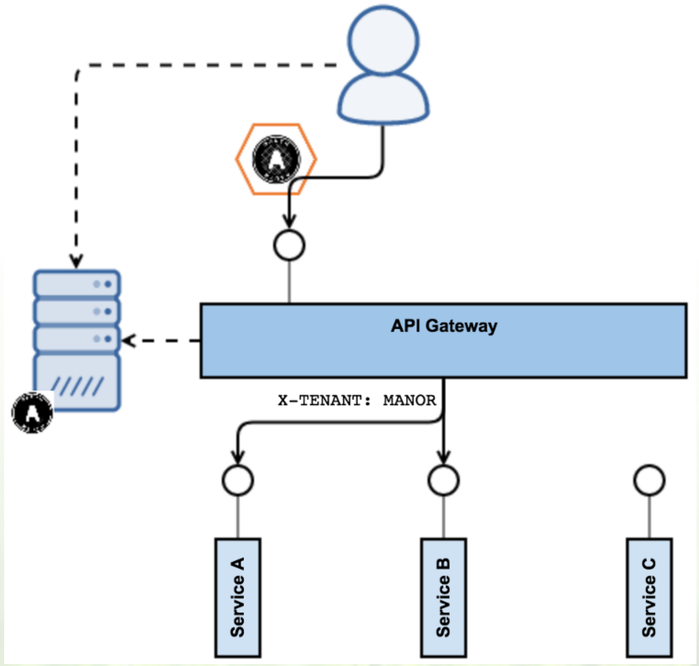
One service operates on the same DataSource (data grows steadily and impacts performance)

Microservice Architecture

Multitenancy - How to identify the tenant on service side?

Solution with OAuth2

- An authorised client accesses the API Gateway with an OAuth2 Access Token and an OIDC ID Token
- The API Gateway either reads tenant information from the **signed ID Token** or requests additional information from the OAuth2 server
- All downstream calls have the **X-Tenant** header set with the tenant name

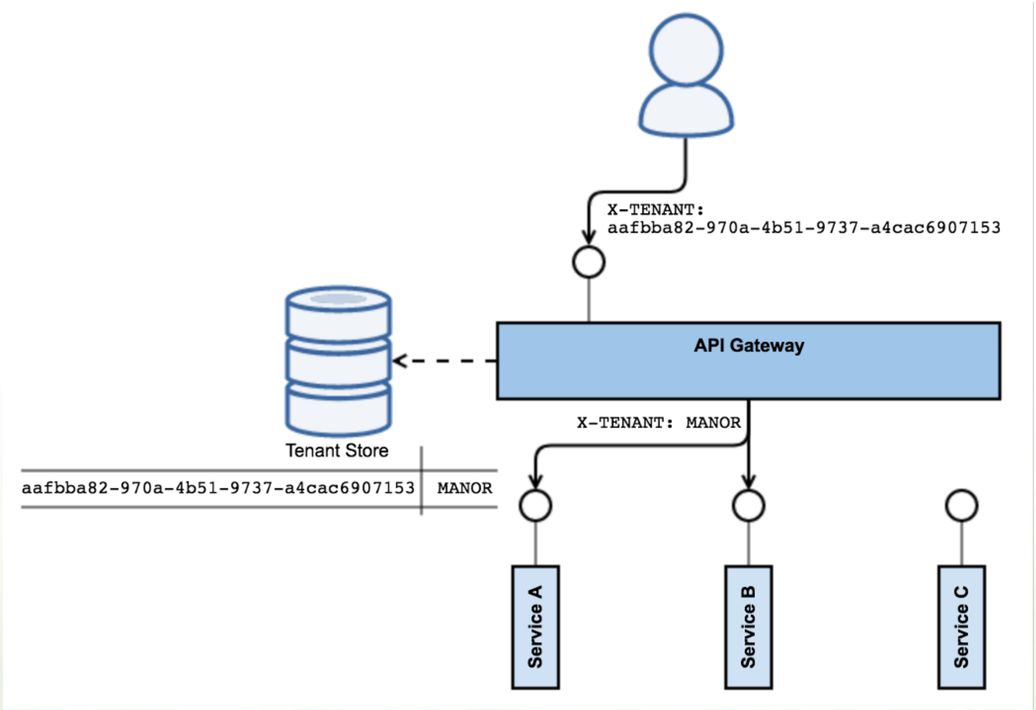


Microservice Architecture

Multitenancy - How to identify the tenant on service side?

Solution without OAuth2

- The API Gateway is responsible to translate the incoming fixed X-Tenant header with an internal tenant representation
- More simple but insecure solution. Shared keys



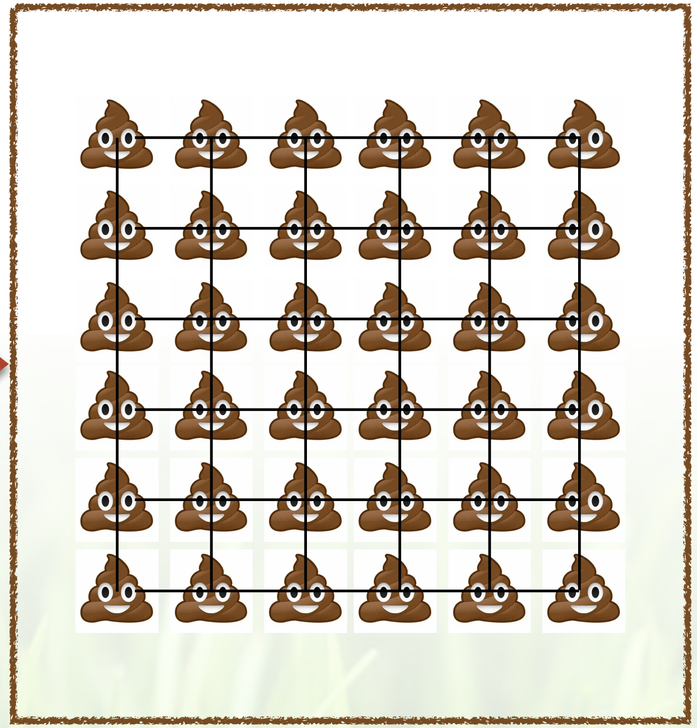
Evolution

Monolith -> µServices -> Microservices

Single Node

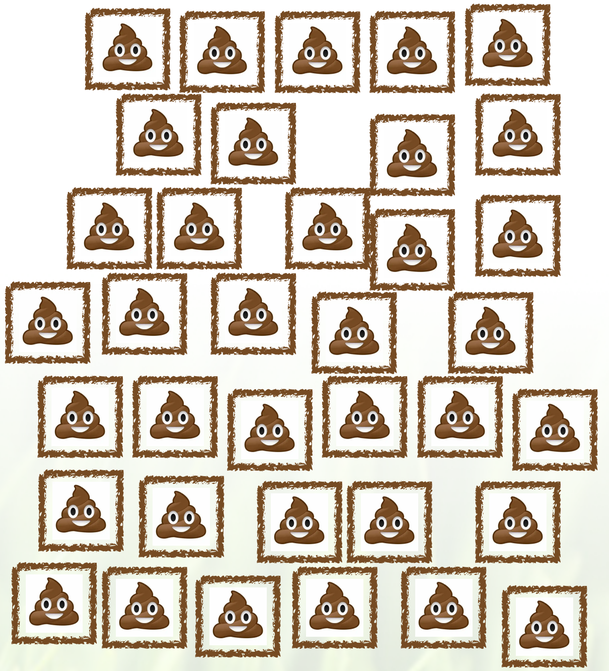
Monolith

Single Node



OSGi µServices

Multiple Nodes



Microservices

What’s next? SaaS

Future Topics

SaaS Options

- SaaS in two different models:
- **Shared SaaS Instances**. Cloud infrastructure is provided and SLA is guaranteed by interface21.io => Risk on my own
- Services on the **cloud provider's marketplace**. Can be installed on cloud resources the companies pay and care for => Customer pays for service usage and subscription

- The last option requires less effort on the software side. Each service is solely used by one company
- The first option requires the software to be tenant-aware

Future Topics
SaaS requirements

- Scale to 0
- Less resource consumption

=> Spring Native and AOT compiling

- Requires a revised architecture into smaller chunks aka well-defined functions
- With Spring Native many traditional features are not supported anymore (like the config server)
- Perhaps RedHat Quarkus is the better choice for this new architecture?

https://docs.spring.io/spring-native/docs/current/reference/htmlsingle/#support-spring-cloud

Thank you!