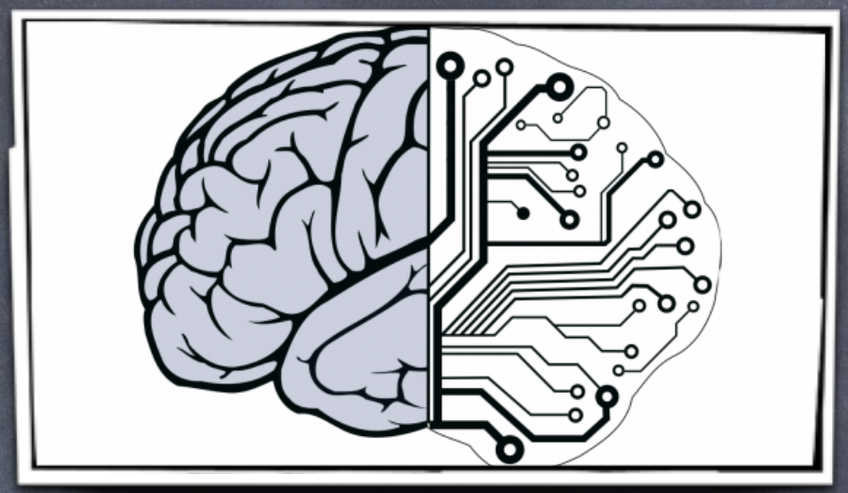
brain.useIt();



= Brain.getEntity(id);

brain.record();

A Pragmatic View on Software Architecture and the Rich Domain Model

with Spring, AspectJ and JPA

by M. Dix

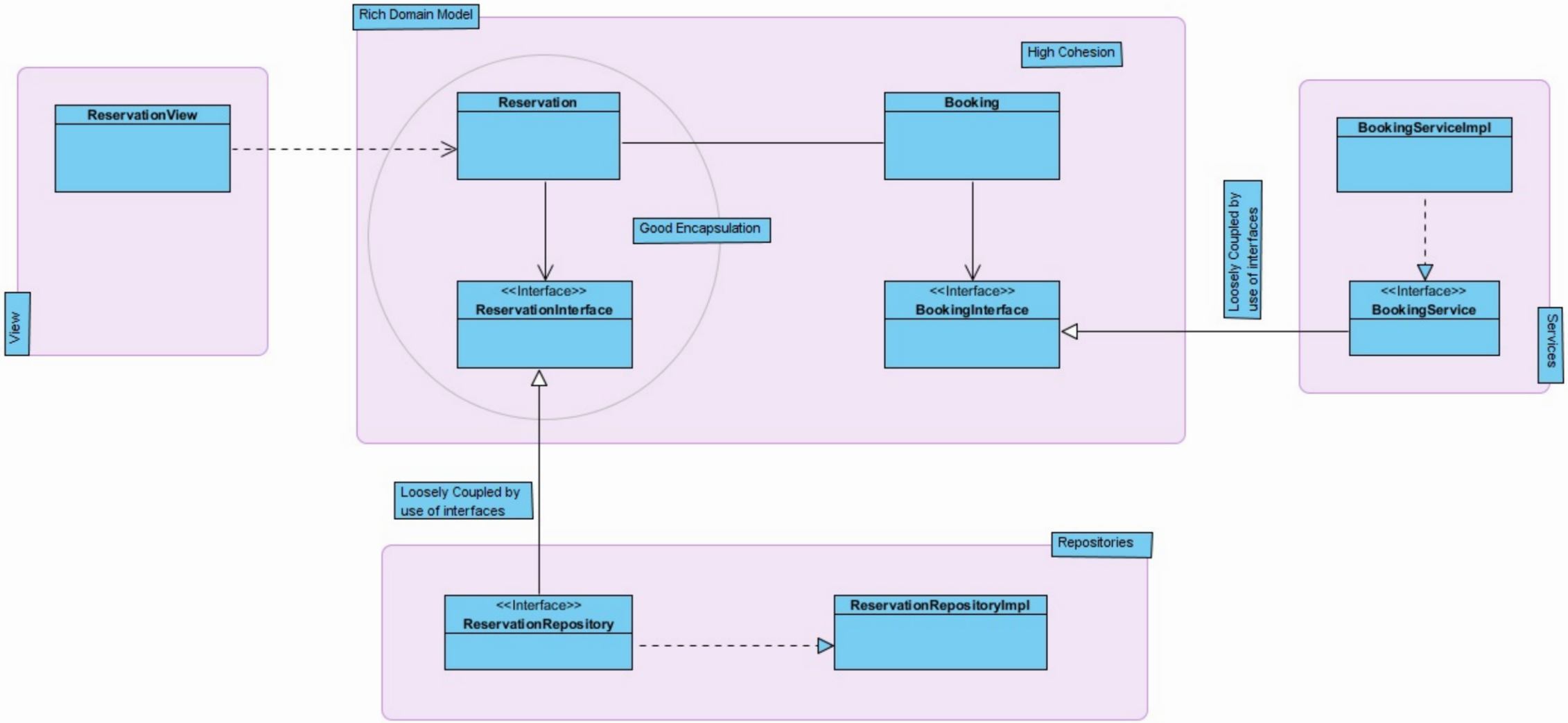
A Short Intro on the Rich Domain Model

- Catch all complexity in the heart of the system
- Defines structure, behavior and handles information
- Responsible for coordination and execution of processes and tasks
- Most processes and tasks need information
- The domain object is responsible for gathering the information it needs to successfully finish a process or task

Object Orientation Basics

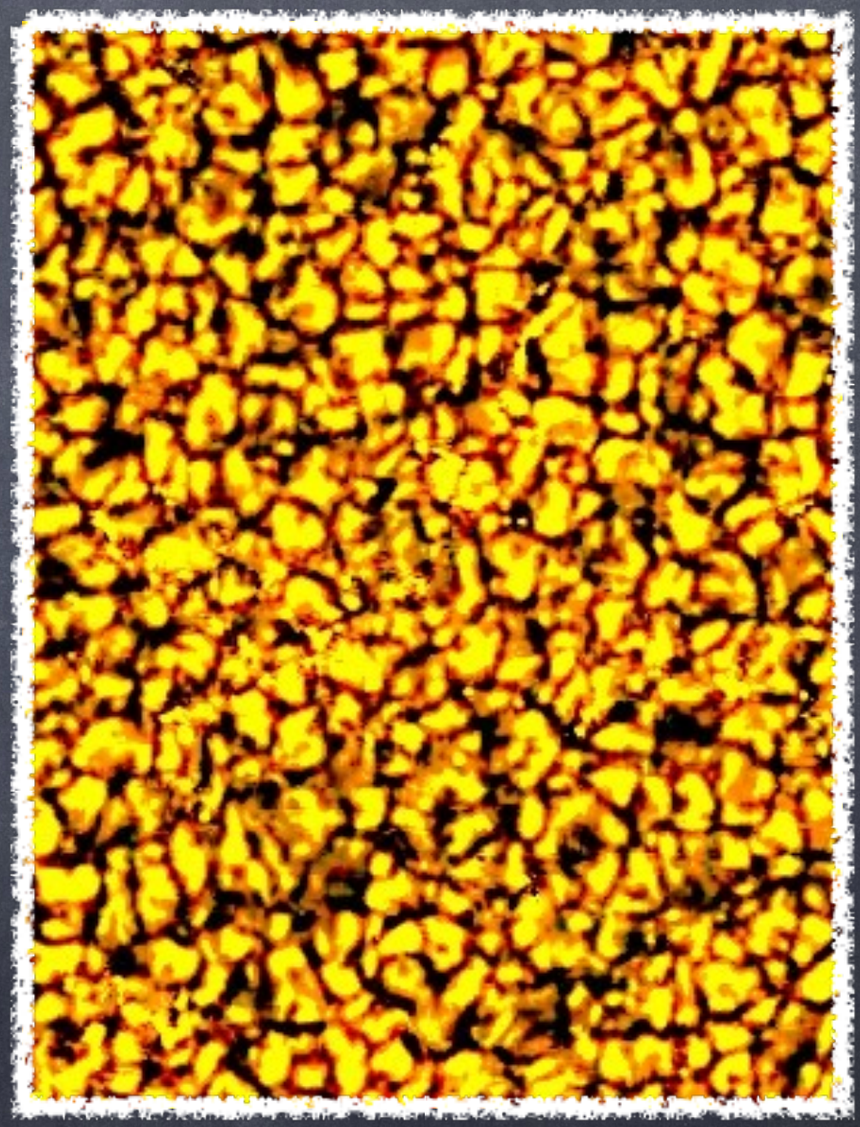
- Loosely Coupling
  - The degree of coupling between modules
  - Loosely Coupling via interfaces results in good traceability of issues
- High Cohesion
  - Group elements in modules that belong to each other
  - Good traceability of issues
- Encapsulation
  - Shield Internal operation by eg. using data hiding
  - Make changes without side effects on client code

A View on Software Architecture



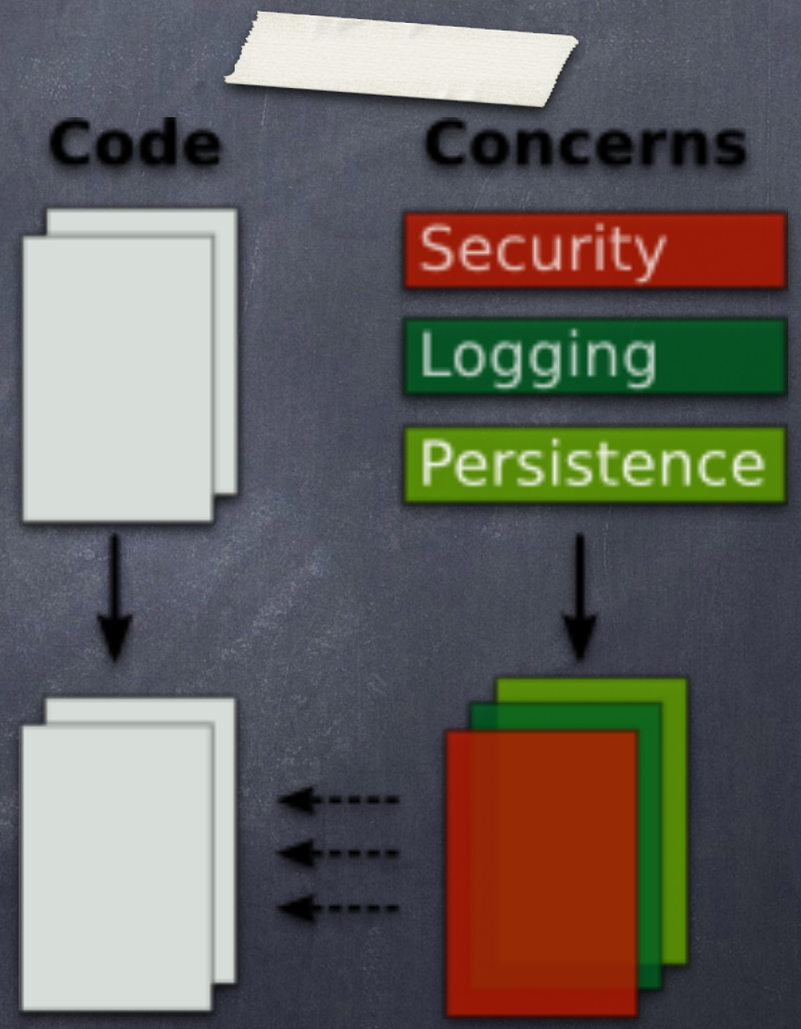
Transaction Demarcation

- Transactions are cross cutting in nature
- High Granularity
  - Definition on Domain
- Low Granularity
  - Definition on Repository



AspectJ LTW to support the Rich Domain

- Load Time Weaving (LTW)
  - No AspectJ compiling necessary
  - Better Maintainable
  - Minimal overhead
- To support autowiring of unmanaged domain objects via @Configurable
- To support transactionality on unmanaged domain objects
- To better support full JUnit integration testing of the Rich Domain Model



The Example Setup

- Spring
- OpenJPA META-INF/Persistence.xml
- OpenJPA Spring configuration
- OpenJPA build time enhancement
  - OpenJPA load time enhancement does not support all features OpenJPA has to offer eg. inheritance mapping fails, with used setup
- AspectJ META-INF/aop.xml
- AspectJ LTW Configuration
  - Autowiring used in unmanaged domain objects
  - Transactionality used in unmanaged domain objects
- Making JUnit integration testing work with AspectJ setup
- JUnit examples

Spring Configuration

Root Context Spring

- <context:spring-configured/>
- <context:annotation-config/>

Configuration OpenJPA - Part I

"META-INF/Persistence.xml

Persistence.xml for WebSphere

```
<persistence-unit name="the-persistence-unit">
    <description>Persistence Unit</description>
    <provider>com.ibm.websphere.persistence.PersistenceProviderImpl</provider>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <validation-mode>AUTO</validation-mode>
    <properties>
        <property name="openjpa.jdbc.DBDictionary"
value="org.apache.openjpa.jdbc.DB2Dictionary" />
        <property name="openjpa.RuntimeUnenhancedClasses" value="unsupported" />
        <property name="openjpa.Compatibility" value="QuotedNumbersInQueries=true" />
        <property name="openjpa.Log" value="DefaultLevel=WARN, Tool=TRACE"/>
    </properties>
  </persistence-unit>
</persistence>
```

Spring Configuration OpenJPA - Part II

Entity Manager Factory

```
<bean id="entityManagerFactory
class="org.springframework.orm.jpa.
LocalContainerEntityManagerFactoryB
ean">

  <property name="dataSource"
ref="testDataSource" />

  <property name="jpaVendorAdapter"
ref="openJPAVendorAdapter" />

  <property
name="persistenceUnitName"
value="the-persistence-unit" />

</bean>
```

OpenJPA vendor adapter & test datasource

```
<bean id="openJPAVendorAdapter"
class="org.springframework.orm.jpa.vendor.OpenJpaVen
dorAdapter">

  <property name="showSql" value="false" />

  <property name="generateDdl" value="false" />

  <property name="database" value="DB2" />

</bean>

<!-- Test datasource -->

<bean id="testDataSource"
class="org.springframework.jdbc.datasource.DriverMana
gerDataSource">

  <property name="driverClassName"
value="com.ibm.db2.jcc.DB2Driver" />

  <property name="username" value="testuser" />

  <property name="password" value="testpassword" />

  <property name="url value="url" />

</bean>
```

Build Configuration OpenJPA - Part III

Build time Enhancement

Ant start script

```
<project name="jpa_enhance_builder">
  <target name="jpaBuild" description="Build Domain Project">
    <exec dir="" executable="cmd">
      <arg value="/c" />
      <arg value="mvn.bat" />
      <arg line="process-classes" />
    </exec>
  </target>
</project>
```

Maven Build Time Enhancement

```
<plugin>
  <groupId>org.apache.openjpa</groupId>
  <artifactId>openjpa-maven-plugin</artifactId>
  <version>2.2.0</version>
  <configuration>
    <includes>nl/company/project/domain/**/*.class</includes>
  </configuration>
  <executions>
    <execution>
      <id>enhancer</id>
      <phase>process-classes</phase>
      <goals>
        <goal>enhance</goal>
      </goals>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.apache.openjpa</groupId>
      <artifactId>openjpa</artifactId>
      <version>${version.openjpa.must.be.same.as.runtime}</version>
    </dependency>
  </dependencies>
</plugin>
```

AspectJ Configuration

"META-INF/aop.xml

```
<aspectj>
    <weaver>
        <!-- Package of aspects -->
        <include within="nl.company.project.aspect.*" />
        <!-- Weaving for @Configurable and @Transactional -->
        <include within="nl.company.project.domain.*" />
        <include within="nl.company.project.domain.model.*" />
        <include within="nl.company.project.dao.impl.*" />
        <!-- excludes   -->
        <exclude within="org.apache.openjpa..*" />
        <!-- abstracts / base classes quirk necessary for JUnit aop.xml only. -->
        <exclude within="nl.company.project.domain.model.Entity" />
    </weaver>
    <aspects>
        <aspect name="nl.company.project.aspect.SomeAspect" />
    </aspects>
</aspectj>
```

AspectJ LTW Configuration

- Spring configuration Weaver for JUnit integration testing
  - <context:load-time-weaver aspectj-weaving="on" weaver-class="org.springframework.instrument.classloading.InstrumentationLoadTimeWeaver"/>
- Spring configuration Weaver for WebSphere Production
  - <context:load-time-weaver aspectj-weaving="on" weaver-class="org.springframework.instrument.classloading.websphere.WebSphereLoadTimeWeaver"/>
- Spring configuration for Transaction manager
  - <bean id="txManager" class="org.springframework.orm.jpa.JpaTransactionManager">
  - <tx:annotation-driven transaction-manager="txManager" mode="aspectj" />
- Domain Object
  - @Configurable configures an AspectJ aspect (AnnotationBeanConfigurerAspect) supported by Spring with a joinpoint on constructor creation
  - @Transactional configures an AspectJ aspect supported by Spring for transactionality

For Autowiring on unmanaged domain objects

For Transactionality on unmanaged domain objects

Configuration
JUnit integration testing
Rich Domain

VM argument for JUnit testing

- -javaagent:../location/testlib/spring-instrument.jar

Maven dependency configuration

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-agent</artifactId>
    <version>3.2.6/version>
    <scope>test</scope>
</dependency>
```

Exclude version number in spring-instrument

```
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-agent</artifactId>
    <version>3.2.6/version>
    <scope>test</scope>
</dependency>
...
<plugin>
  <artifactId>maven-dependency-plugin</artifactId>
  ...
  <executions>
    ...
    <execution>
      <id>copy-instrument</id>
      <phase>validate</phase>
      <goals>
        <goal>copy-dependencies</goal>
      </goals>
      <configuration>
        <outputDirectory>${testlib.location}</outputDirectory>
        <stripVersion>true</stripVersion>
        <includeArtifactIds>spring-instrument</includeArtifactIds>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Quirks JUnit Setup

- Only in the JUnit environment the code snippet below is a no go, due to issues with the WebSphere Liberty Profile. Due to this local runtime conflict @Configurable will not work on these classes.
    - Base baseEntity = new Extended();
    - Extended extendedEntity = (Extended) base;
    - extendedEntity.save();
- Do not use Transactions on JUnit methods
    - Its highly likely inconsistent with your production setup, as your view is not transactional
    - Transaction propagation does not work
- When creating an object outside the JUnit method scope weaving will fail. In de example below none of the @Autowired attributes will be set and hence will be null
    - @Test
    - public void reserveFlight() {
    - Reservation reservation = createReservation();
    - reservation.save();
    - }
    - private Reservation createReservation() {
    - return new Reservation();
    - }

All issues described are non existent inside the WebSphere Container!

Example Reservation Entity

The Reservation Class

```
@Configurable
@Entity
@AttributeOverrides({@AttributeOverride(name = Constants.DOMAINID, column = @Column(name = "RESERV_" + Constants.ID) })

public class Reservation extends Entity {
    private static final long serialVersionUID = 502745151854211986 9L;
    /** functional interface, will be injected with daoImpl runtime. */
    @Autowired
    private ReservationInterface reservationInterface;
    @Column(name = "NAME")
    private String name;
    @Column(name = "FLIGHTNUMBER")
    private String flightnumber;

    public Reservation() { /** empty. */ }

    @Transactional
    public Reservation save() {
       return this.reservationInterface.save(this);
    }
}
```

Example JUnit Part I

Base Class for all Integration Tests

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations = { "classpath:/context-test.xml" })
public class TestBase extends AbstractJUnit4SpringContextTests {
    public TestBase() { /** empty. */ }

    @Before
    public void setUp() {
        ClassPathXmlApplicationContext ctx =
            new ClassPathXmlApplicationContext("context-test.xml");
        }
    }
}
```

Example JUnit Part II

Reservation JUnit Integration Tests

```
@TestExecutionListeners(DependencyInjectionTestExecutionListener.class)
public class TestReservation extends TestBase {
    public TestReservation() { /** empty. */}

    @Test
    public testReservation() {
        Reservation reservation = new Reservation();
        reservation.setName("name");
        reservation.setFlightnumber(1);
        reservation.save();
    }
}
```

For questions or comments
ruimtefotografie.org@gmail.com