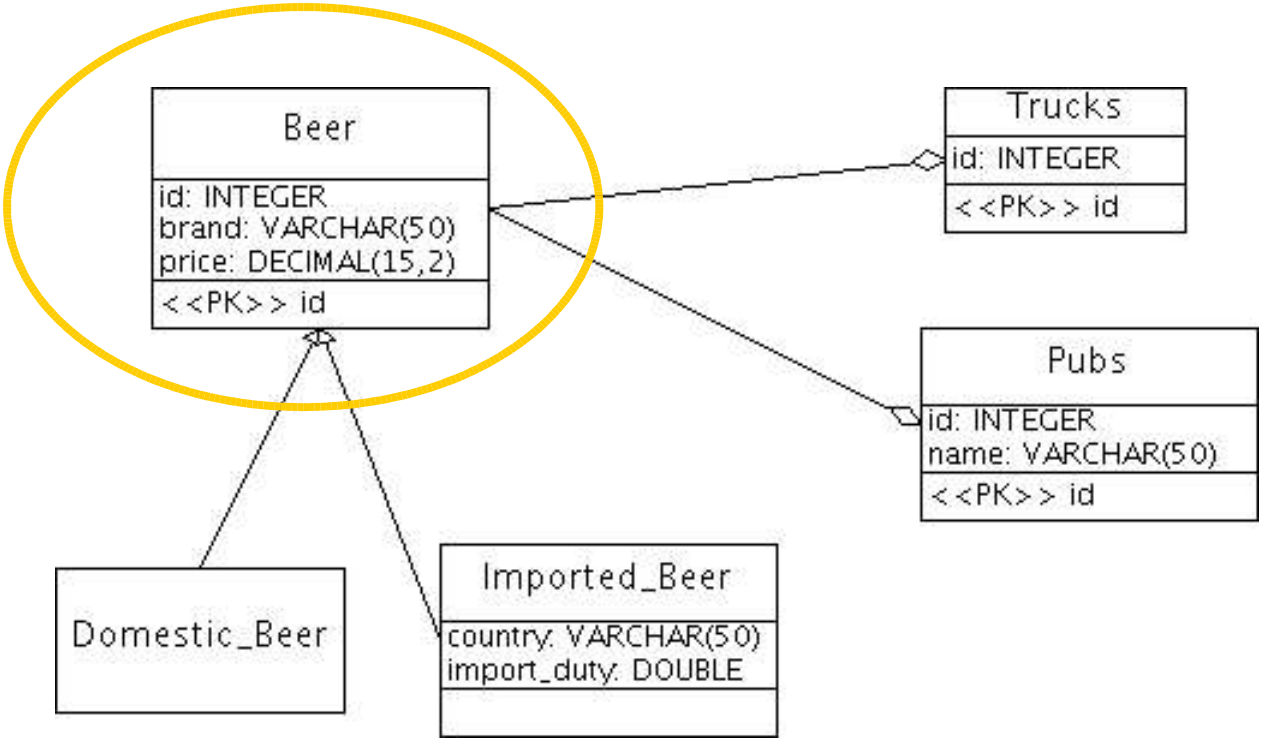
DAO and JDBC support

Thomas Risberg
Data Architecture
TargetRx, Inc.

Beer Example – Data Model



Beer.java

```
package org.buggybean.phillyjug;
import java.math.BigDecimal;

public class Beer {
  private long id;
  private String brand;
  private BigDecimal price;

  public long getId() {
    return id;
  }
  public void setId(long id) {
    this.id = id;
  }
  public String getBrand() {
    return brand;
  }
  public void setBrand(String brand) {
    this.brand = brand;
  }
  public BigDecimal getPrice() {
    return price;
  }
  public void setPrice(BigDecimal price) {
    this.price = price;
  }
}
```

JDBC

```
public Beer getBeer(long id) {
  Beer beer = null;
  Connection conn = null;
  PreparedStatement ps = null;
  ResultSet rs = null;

  try {
    Class.forName("org.hsqldb.jdbcDriver");
    conn = DriverManager.getConnection(
           "jdbc:hsqldb:hsql://localhost", "sa", "");
    ps = conn.prepareStatement(
           " select id, brand, price from Beer where id = ?");
    ps.setLong(1, id);
    rs = ps.executeQuery();
    if (rs.next()) {
      beer = new Beer();
      beer.setId(rs.getLong("id"));
      beer.setBrand(rs.getString("brand"));
      beer.setPrice(rs.getBigDecimal("price"));
    }
  }
```

```
  catch (ClassNotFoundException e) {
    logger.error(e.toString());
  }
  catch (SQLException se) {
    logger.error(se.toString());
  }
  finally {
    if (rs != null) {
      try { rs.close(); }
      catch (SQLException ignore) {}
   }
    if (ps != null) {
      try { ps.close(); }
      catch (SQLException ignore) {}
    }
    if (conn != null) {
      try { conn.close(); }
      catch (SQLException ignore) {}
    }
  }
  return beer;
}
```

Spring

```
private class BeerMappingQuery extends MappingSqlQuery {

  public BeerMappingQuery(DataSource ds) {
    super(ds, "select id, brand, price from Beer where id = ?");
    super.declareParameter(new SqlParameter("id", Types.INTEGER));
    compile();
  }

  public Object mapRow(ResultSet rs, int rowNumber)
         throws SQLException {
    Beer beer = new Beer();
    beer.setId(rs.getLong("id"));
    beer.setBrand(rs.getString("brand"));
    beer.setPrice(rs.getBigDecimal("price"));
    return beer;
  }
}
```

Spring continued …

```
public Beer getBeer(long id) {
  BeerMappingQuery beerQry = new
                     BeerMappingQuery(dataSource);
  return (Beer) beerQry.findObject(id) ;
}
```

JDBC / Spring comparison

	JDBC	Spring
Connections	Need to explicitly open and close connections. Need a separate strategy for making code reusable in a variety of environments.	Uses a DataSource with the framework managing connections. Code following the framework strategy is automatically reusable.
Exceptions	Must catch SQLExceptions and interpret database specific SQL error code or SQL state code.	Framework translates exceptions to a common hierarchy based on configurable translation mappings.
Testing	Hard to test standalone if code uses JNDI lookup for connection pools.	Can be tested standalone since a DataSource is easily configurable for a variety of environments.
Transactions	Programmatic transaction management is possible but makes code less reusable in systems with varying transaction requirements. CMT is available for EJBs.	Programmatic or declarative transaction management is possible. Declarative transaction management works with single data source or JTA without any code changes.

Spring JDBC Division of Labor

Task	Spring	You
Connection Management	√	
SQL		√
Statement Management	√	
ResultSet Management	√	
Row Data Retrieval		√
Parameter Declaration		√
Parameter Setting	√	
Transaction Management	√	

Testing JDBC

```
public class BeerDistributorDAOTest extends TestCase {

  private BeerDistributorDAO dao;

  public void setUp() {
    dao = new JdbcBeerDistributorDAO();
  }

  public void testGetBeer() {
    Beer dist = dao.getBeer(1);
    assertEquals("got the right id", 1, dist.getId());
    assertEquals("brand is expected one", "Budweiser",
                 dist.getBrand());
  }
}
```

This would not work if DAO used JNDI lookup!!!

Testing Spring

```
public class BeerDistributorDAOTest extends TestCase {

  private SpringBeerDistributorDAO dao;

  public void setUp() {
    dao = new SpringBeerDistributorDAO();
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName("org.hsqldb.jdbcDriver");
    ds.setUrl("jdbc:hsqldb:hsql://localhost");
    ds.setUsername("sa");
    ds.setPassword("");
    dao.setDataSource(ds);
  }

  public void testGetBeer() {
    Beer dist = dao.getBeer(1);
    assertEquals("got the right id", 1, dist.getId());
    assertEquals("brand is expected one", "Budweiser",
                 dist.getBrand());
  }
}
```

Testing Spring with IoC

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
 "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
  <bean id="myDataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName">
      <value>org.hsqldb.jdbcDriver</value>
    </property>
    <property name="url">
      <value>jdbc:hsqldb:hsql://localhost</value>
     </property>
    <property name="username">
      <value>sa</value>
    </property>
    <property name="password">
      <value></value>
    </property>
  </bean>
  <bean id="myDao" class="org.buggybean.phillyjug.SpringBeerDistributorDAO">
    <property name="dataSource">
      <ref>myDataSource</ref>
    </property>
  </bean>
```

Testing Spring with IoC continued …

```
public class BeerDistributorDAOTest extends TestCase {

  private BeerDistributorDAO dao;

  public void setUp() {
    ApplicationContext ac = new
        FileSystemXmlApplicationContext("beer-context.xml");
    dao = (BeerDistributorDAO) ac.getBean("myDao");
  }

  public void testGetBeer() {
    Beer beer = dao.getBeer(1);
    assertEquals("got the right id", 1, beer.getId());
    assertEquals("brand is expected one", "Budweiser",
                 beer.getBrand());
  }
}
```

Spring and IoC or *Dependency Injection*

Spring supports both setter and constructor injection.

Beans are defined in XML or properties files.

Bean definitions and Application Contexts are usually loaded by startup code in a main or init method or from a Servlet listener.

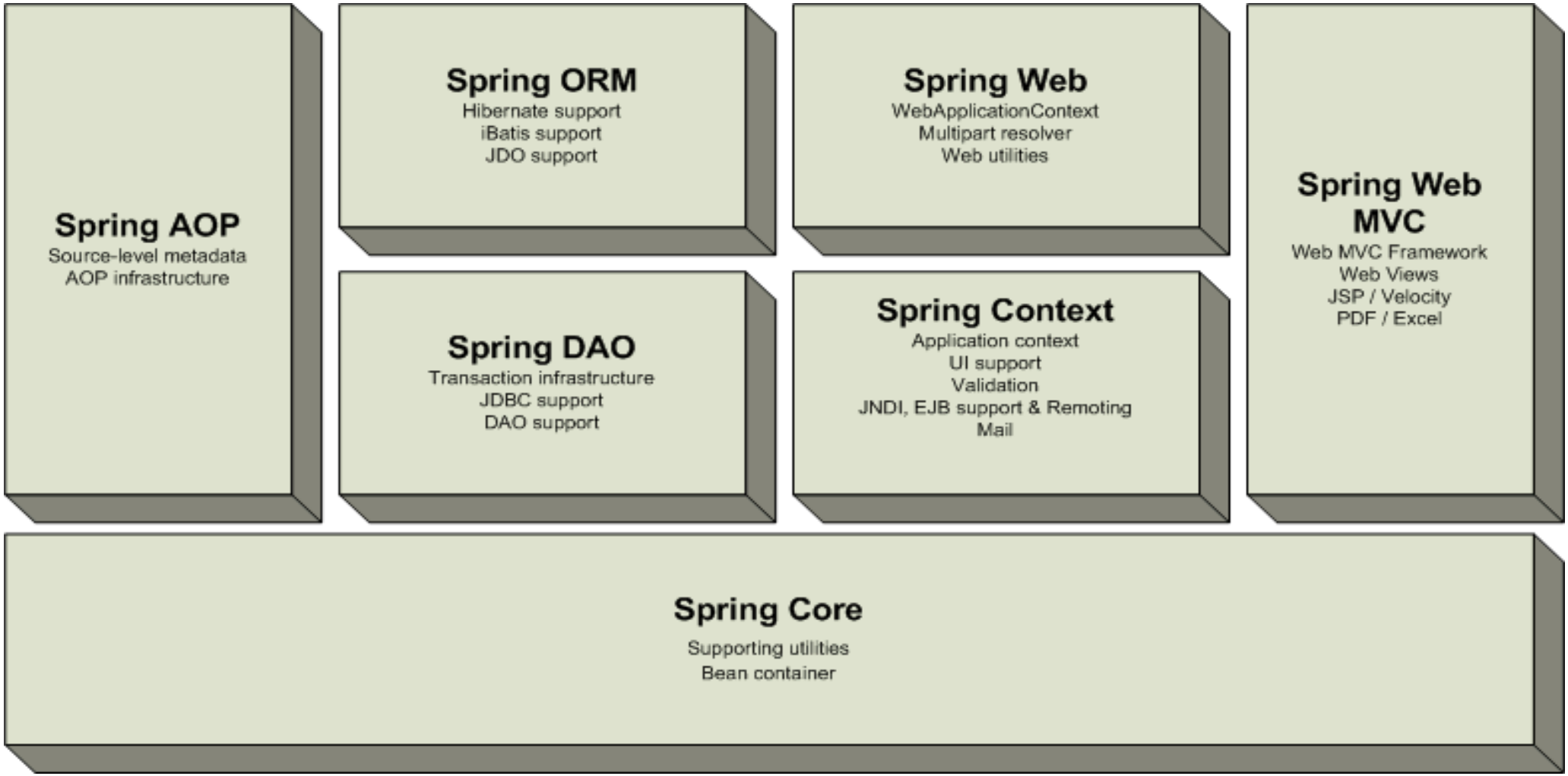
Martin Fowler has a good article about this:

http://martinfowler.com/articles/injection.html

Spring IoC example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
 "http://www.springframework.org/dtd/spring-beans.dtd">

<beans>

  <bean id="myBean" class="TestBean">
    <property name="name"><value>Foo</value></property>
  </bean>

  <bean id="myClass" class="TestClass">
    <constructor-arg><value>Bar</value></constructor-arg>
  </bean>

  <bean id="anotherBean" class="AnotherBean">
    <property name="bean"><ref bean="myBean"/></property>
  </bean>

</beans>
```

Architectural Overview



General List of Features

- **Powerful JavaBeans-based configuration management**, applying Inversion-of-Control principles. This makes wiring up applications quick and easy. This core bean factory can be used in any environment, from applets to J2EE containers.
- **AOP functionality**, fully integrated into Spring configuration management. You can AOP-enable any object managed by Spring, adding aspects such as declarative transaction management.
- **Flexible MVC web application framework**, built on core Spring functionality. This framework is highly configurable via strategy interfaces, and accommodates multiple view technologies like JSP, Velocity, Tiles, iText, and POI.
- **Easy integration with a web tier based on any other web MVC framework**, like Struts, WebWork, or Tapestry.

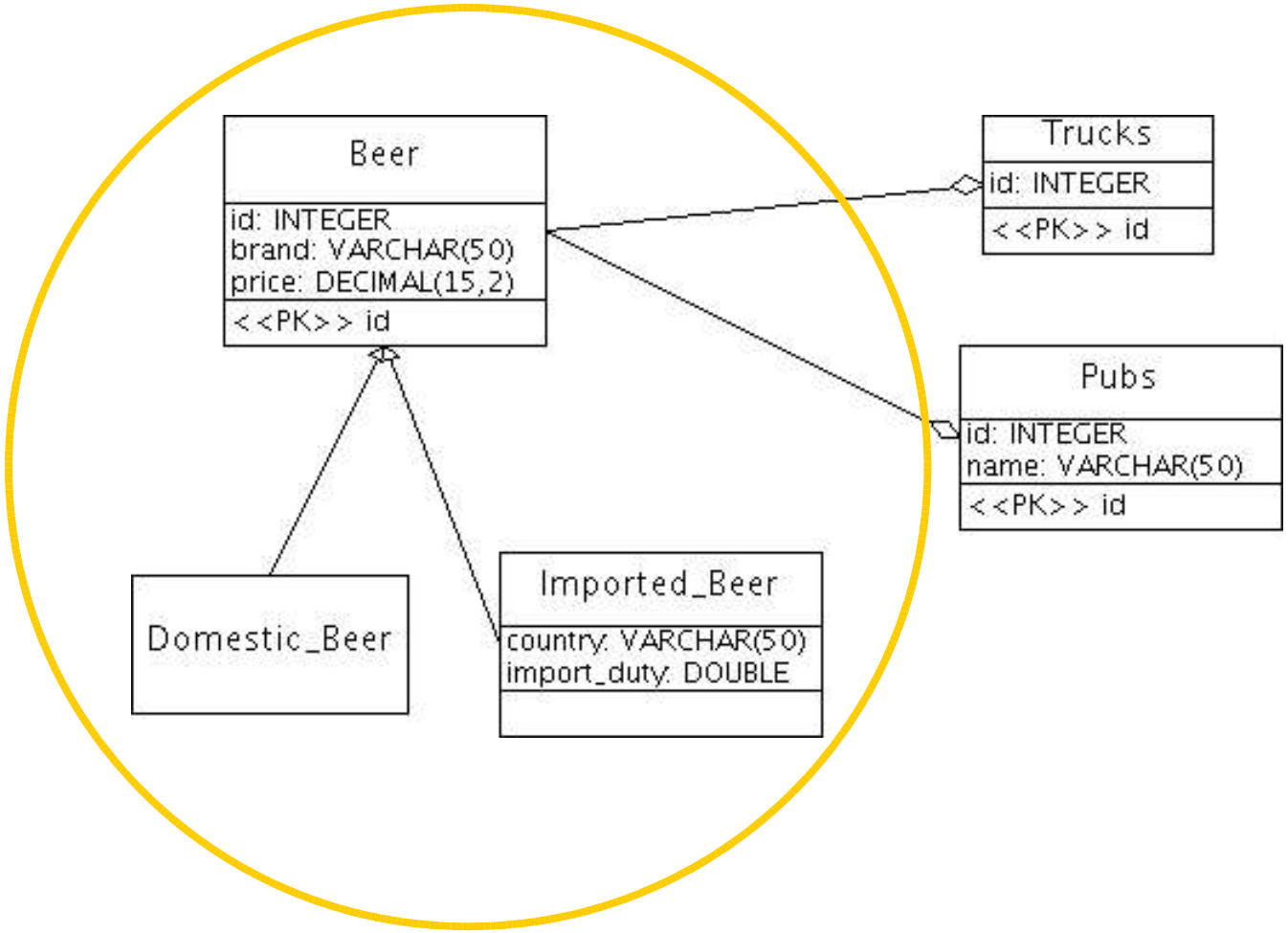
JDBC Features

- **JDBC abstraction layer** that
  - provides exception translation that offers a meaningful exception hierarchy and simplifies error handling.
  - Includes a JdbcTemplate with many convenience methods for easier data access.
  - Includes an object layer on top of the JdbcTempate. This layer gives you SqlQuery, SqlUpdate and StoredProcedure classes for more “object oriented” use.
  - manages the connections - you'll never need to write another finally block to use JDBC again.
  - greatly reduces the amount of code you'll need to write.

DAO / TX Features

- **Unified way of working** whether you use JDBC or an O/R Mapper like Hibernate or JDO
- **Integration with Hibernate, JDO, and iBATIS SQL Maps**: in terms of resource holders, DAO implementation support, and transaction strategies.
- **Generic abstraction layer for transaction management**, allowing for pluggable transaction managers, and making it easy to demarcate transactions without dealing with low-level issues. Generic strategies for JTA and a single JDBC DataSource are included.
- **Declarative transaction management without EJB**... even without JTA, if you're using a single database in Tomcat or another web container lacking JTA support.

Beer Example – Data Model



BeerApp :: Cheers - Mozilla

File Edit View Go Bookmarks Tools Window Help

http://localhost:8080/beerapp/cheers.htm Search

Philly JUG :: BeerApp

Cheers, it is now Sun Mar 28 22:04:36 EST 2004

Beers

Budweiser	*$2.58*
Heineken	*$3.90*
Spring Ale	*$9.43*
Bass	*$5.37*

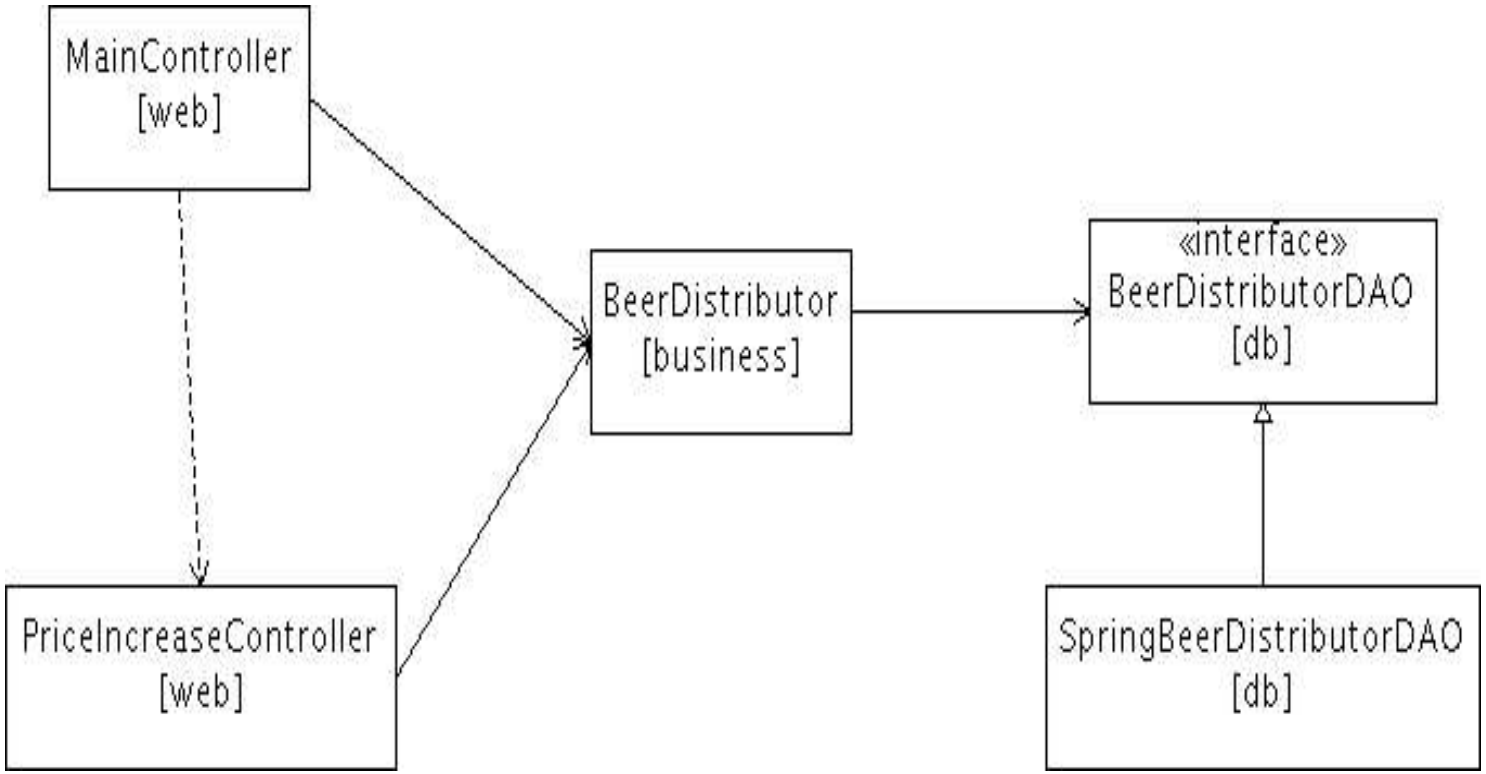
Beer of the Month - March 2004

Increase Prices

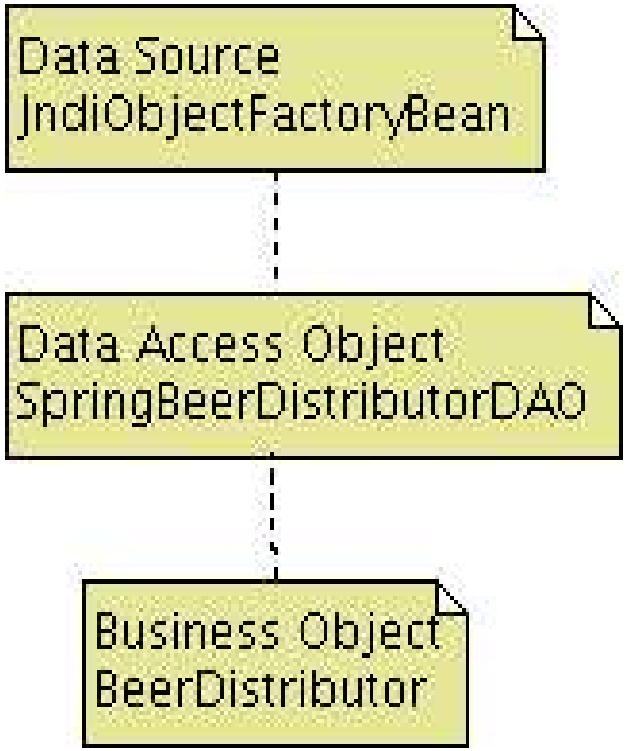
Powered By Spring

Live Code Example

Application Architecture



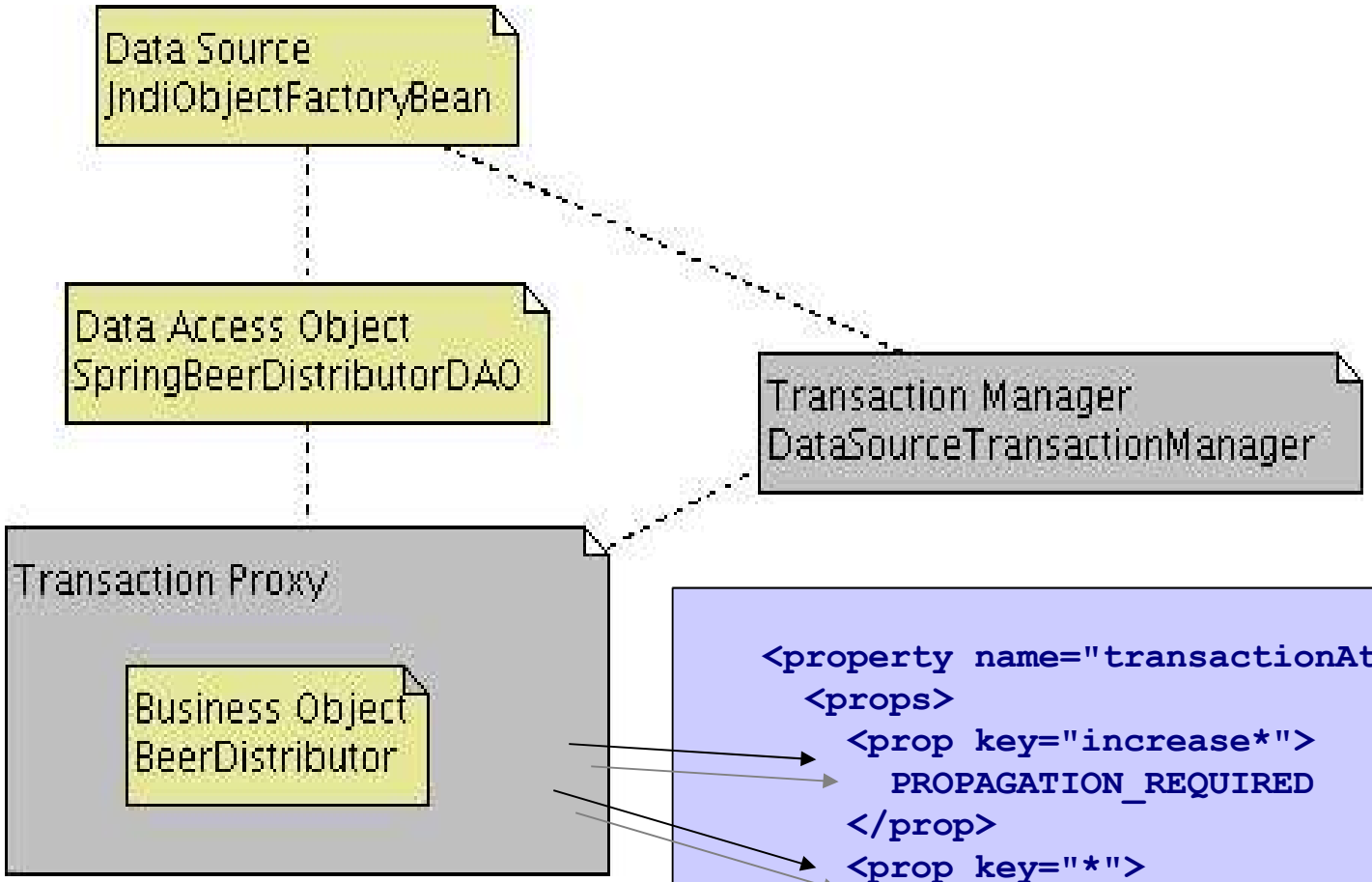
Application Context



BeerDistributor increasePrice()

```
public void increasePrice(int increase) {
    Beer[] beers = getAllBeers();
    for (int i = 0; i < beers.length; i++) {
        BigDecimal newPrice =
            (beers[i].getPrice().
                add(beers[i].getPrice().
                    multiply(new BigDecimal((double)increase/100))
                )
            ).setScale(2, BigDecimal.ROUND_HALF_EVEN);;
        beers[i].setPrice(newPrice);
        dao.updateBeer(beers[i]);
    }
}
```

Transactional Application Context




```
<property name="transactionAttributes">
  <props>
    <prop key="increase*">
      PROPAGATION_REQUIRED
    </prop>
    <prop key="*">
      PROPAGATION_SUPPORTS,readOnly
    </prop>
  </props>
</property>
```

Java - BeerDistributor.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

BeerDistributor.java - PhillyJUG/src/org/buggybean/phillyjug/

```
public class BeerDistributor {

    private BeerDistributorDAO dao;

    public Beer selectGoodBeer(int id) {
        return dao.getBeer(id);
    }

    public void makeChangesTo[illegible]r(Beer [illegible]er, [illegible]tr[illegible]g [illegible] double price, double importDuty) {
        if (brand != null) {
            beer.setBrand(bran[illegible]);
            dao.updateBeer(beer);
        }
        if (price != 0) {
            beer.setPrice(price);
            dao.updateBeer(beer);
        }
        if (importDuty != 0 && beer instanceof ImportedBeer) {
            ((ImportedBeer) beer).setImp[illegible]tDuty(importDuty);
            da[illegible]up[illegible]te[illegible]ee[illegible] be[illegible]r)
        }
    }

    public BeerDistributorDAO getDao() {
        return dao;
    }
    public void setDao(BeerDistributorDAO dao) {
        this.dao = dao;
```

Live Code Example

Writable Smart Insert 6 : 31

trisberg@ton HSQL Databa HSQLDB Doc /home/trisber Java - BeerDi 96% 10:51 PM

Who's using Spring

- **Synapsis Technology, Inc.**, Springhouse, PA

  EMARS™ - Managing Environmental Compliance using Spring IoC, TX, Hibernate and iBATIS
- **Rutgers University**, NJ

  Graduate Admissions System, myRutgers Portal, HR Time Reporting using Spring MVC, AOP, IoC, JDBC, TX
- **Ilse Media / Sanoma**, Netherlands

  sales process management using Spring MVC, AOP, IoC
- **Global investment bank**, London, UK

  2 projects live with Spring MVC, IoC, JDBC, 10,000 users
- **Global investment bank**, New York

  and a lot more **...**

Quotes

- Lightweight containers like Spring are spreading rapidly, because they can solve problems that other containers can't. ... The most interesting aspects, to me, are the transparency of the objects, clean and pluggable services, and the light footprint. [**Bruce Tate** – *Bitter Java, Bitter EJB*]
- I took some time last weekend and refactored AppFuse to use Spring to replace my Factories and Hibernate configuration. It only took me a couple of hours, which says a lot for Spring. I was amazed at how many things just worked. [**Matt Raible** – *Pro JSP 3rd Edition*]
- The talk of the show, both officially in my talks and in Bruce Tate's talks and unofficially in the expert panels and hallway discussions, was the Spring Framework. I'm excited to see this fantastic framework gain so much popularity. [**Craig Walls** – *XDoclet in Action*]
- It is a truly world class framework, with stability and quality rarely seen in the Open Source enterprise Java space. In ~6 months of using Spring, I don't recall ever finding a serious flaw or bug, and certainly the minor ones were fixed extremely quickly. [**Mike Cannon-Brokes**, *Java Open Source Programming*]

Polls

What is the hottest new Java technology today?

Current result of this poll

AspectJ or Aspect Oriented Programming	8% (9)
Eclipse Microkernel and Rich Client Platform	9% (10)
Hibernate	18% (21)
JDK 1.5 Generics	6% (7)
Java Server Faces or Rave	1% (2)
JXTA or Overlay Networks	0% (1)
Spring Framework	40% (45)
Other (Please Specify)	14% (16)
Java Bluetooth	0% (0)

Total number of votes: 111

The poll is not scientific and reflects only the opinions of those users who chose to participate. The results should not be considered representative of either users opinions in general, or the public as a whole.

From http://www.manageability.org/polls/the-hottest-java-based-technology/view by Carlos E. Perez (3/17/2004)

Spring History and Roadmap

- Project based on code published in
  - ***Expert one-on-one J2EE design and Development*** (November 2002) Wrox
    Rod Johnson
- SourceForge Project since February 2003
- 1.0 available since March 2004
- Alpha version of Eclipse plugin available
- Two books coming
  - ***J2EE without EJB*** (May 2004) Wrox
    Rod Johnson / Jürgen Hoeller
  - ***Professional Spring Development*** (Q4 2004) Wrox
    Johnson / Hoeller / Risberg / Arendsen
- JMS, JMX and Rich Client support scheduled for second half of 2004
- Extensive reference manual in the works

Links

http://www.springframework.org

http://www.theserverside.com/articles/article.tss?l=SpringFramework

http://www.theserverside.com/articles/article.tss?l=SimplerJava

Q&A