

# Magic Box project: Effective K.I.S.S. with SpringFramework



**Magic Box**  
W9&IC BOX

<http://sourceforge.net/projects/magic-box>



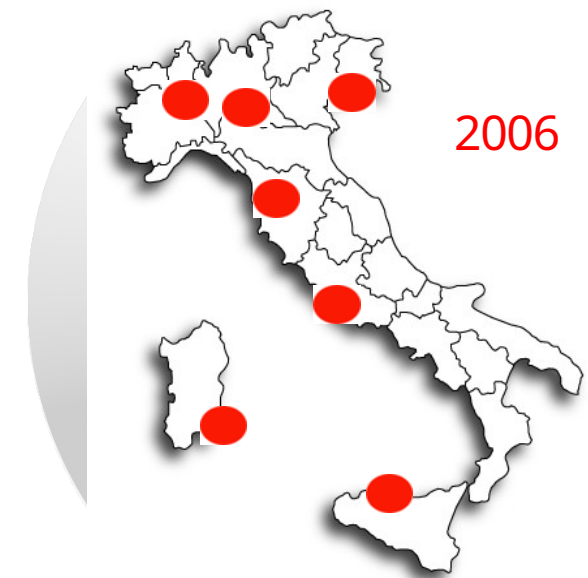
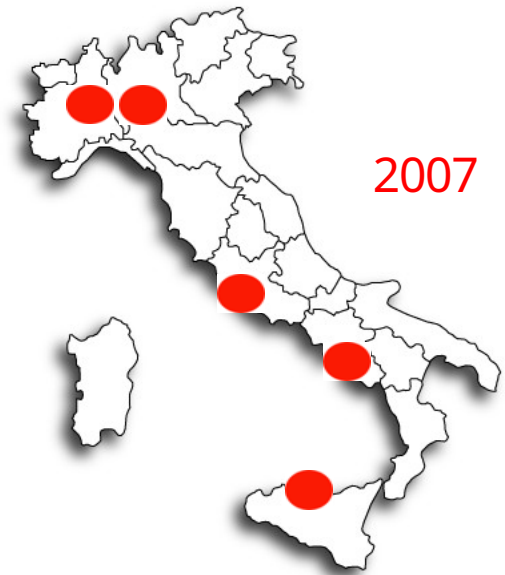
# The event that host this presentation: Javaday



Organizzato direttamente dalla collaborazione della community Java, gira l'Italia con passione

**Javaday è un evento  
unico nel mondo!**

L'edizione romana è a cura di  
**Jug Roma**  
**Java Italian Portal**  
**Java Italian Association**



# Requirements for this talk



“We believe not only that J2EE development should be much simpler than the mixture of drudgery and complexity it’s often made out to be, but that developing J2EE applications **should be fun**”

Rod Johnson

<http://www.springframework.org/>



Ralph, Erich, Richard, John at OOPSLA 1994

<http://c2.com/cgi/wiki?GangOfFour>

<http://c2.com/cgi/wiki?DesignPatternsBook>



Java Architect \*

Co-fondatore e consigliere

JugSardegna Onlus

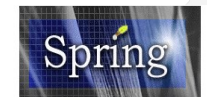
2003



Fondatore e coordinatore:

SpringFramework Italian User Group

2006



Jetspeed Italian User Group

2003



Groovy Italian User Group

2007





Spring User da Luglio 2004 (Spring 1.1)

Primo articolo italiano settembre 2004 sul JugSardegna.

Primo corso di formazione su Spring in Italia maggio 2005 per la Sistemi Informativi a Roma.



# MagicBox Project



The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://sourceforge.net/projects/magic-box/`. The page header includes the SourceForge logo and navigation links: SF.net, Projects, Services (with a BETA badge), My SF.net, and Help. A search bar is also present. The main content area shows the breadcrumb trail: SF.net » Projects » Magic Box » Summary. The title "Magic Box" is prominently displayed. Below the title is a horizontal menu with tabs: Project, Tracker, Mailing Lists, Forums, Code, Services, Download, Documentation, and Tasks. The "Project" tab is selected. The description states: "Magic Box is an J2EE solution with Java technology to coordinate blood donations." Below this, project details are listed: Project Admins: desmax74, Operating System: OS Independent (Written in an interpreted language), License: Apache License V2.0, GNU General Public License (GPL), and Category: Communications.

SourceForge.net: Magic Box - Mozilla Firefox

File Modifica Visualizza Cronologia Segnalibri Strumenti ?

http://sourceforge.net/projects/magic-box/

SOURCEFORGE.NET®

SF.net Projects Services **BETA** My SF.net Help

Search Advanced

SF.net » Projects » Magic Box » Summary

## Magic Box


Project Tracker Mailing Lists Forums Code Services Download Documentation Tasks

Magic Box is an J2EE solution with Java technology to coordinate blood donations.

**Project Admins:** desmax74  
**Operating System:** OS Independent (Written in an interpreted language)  
**License:** Apache License V2.0, GNU General Public License (GPL)  
**Category:** Communications







# Magic Box



HomeInfoGruppiRicercheComunicazioniImportazioneEsportazioneStatisticheLogout

**Ricerche utenti**

- Tutti
- Per Nominativo
- Per Cap
- Per Citta
- Per provincia

Centro: Centro Cagliari Uno - Donatori: 303 - Sms residui: 20 - Pagina 1 di 21

Donatore	Tessera		
baldussi aldo	694		
baldussu aldo	570		
baldussu antonella	90		
baldussu bruno	726		
baldussu efisio	126		
baldussu ermelinda	703		
baldussu giovanni	614		
baldussu guido	7		
baldussu massimiliano	585		
baldussu mattia	640		
baldussu michele	632		
baldussu monica	568		
baldussu renato	693		
baldussu sergio	429		
baldussu tullia			



AUTHORBLOGSPRINGPOSTGRESOL POWERED

# MagicBox Project



## Spring e Spring Web Flow nel progetto Jug Avis Web

(Aka Magic Box)

Java Summer Meeting Cagliari 16 Luglio 2005



## Spring, iBATIS e Transazioni AOP nel Jug Avis Web

Massimiliano Dessi Frame S.r.l  
<http://www.jugsardegna.org/vqwiki/jsp/Wiki?MassimilianoDessi>

Massimiliano Dessi, Spring, iBATIS e Transazioni AOP nel Jug Avis Web.

Linux Day 2006

GULCh - Gruppo Utenti Linux Cagliari  
[www.gulch.it](http://www.gulch.it)



## Spring Stack Testing: Continuous integration, Continuous Agitation

Massimiliano Dessi CRS4  
<http://wiki.java.net/bin/view/People/MassimilianoDessi>

Massimiliano Dessi, Spring Stack Testing: Continuous integration, Continuous Agitation  
Cagliari, 28 ottobre 2006





## Seven Principles Of Software Development

### Seven Principles of Software Development

by [DavidHooker](#) - 9/5/96.  
<mailto:DavidH@mindless.com>

#### The First Principle: The Reason It All Exists (Pattern: [The Reason](#))

A software system exists for one reason: *to provide value to its users*. All decisions should be made with this in mind. Before specifying a system requirement, before noting a piece of system functionality, before determining the hardware platforms or development processes, ask yourself questions such as: "Does this add real VALUE to the system?" If the answer is "no", don't do it. All other principles support this one.

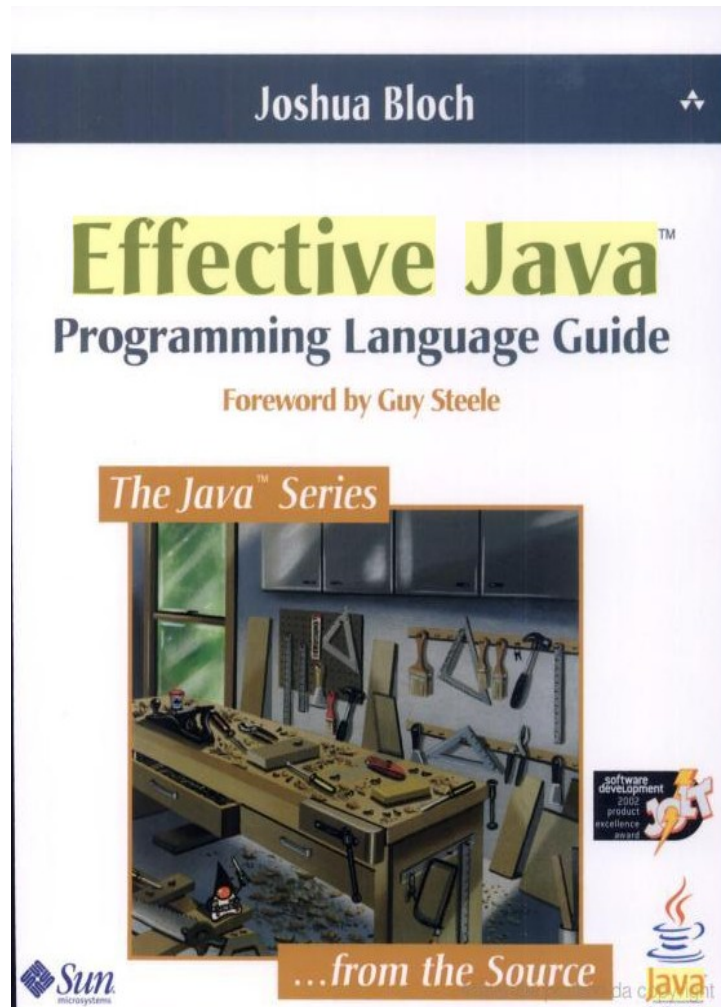
I agree about the value bit, but as [JerryWeinberg](#) noted in [QualitySoftwareManagement](#), volume 1 ([SystemsThinking](#)), value is a relative not an absolute

#### The Second Principle: KISS (Keep It Simple, Stupid!) (Pattern: [KeepItSimpleStupid](#))

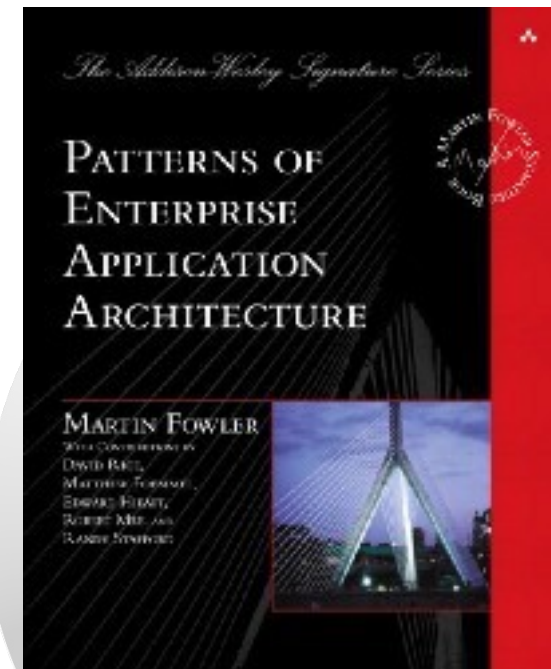
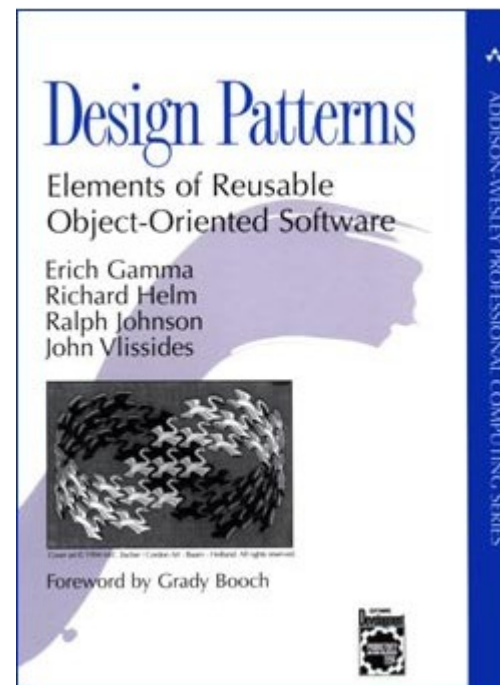
Software design is not a haphazard process. There are many factors to consider in any design effort. *All design should be as simple as possible, but no simpler*. This facilitates having a more easily understood, and easily maintained system. This is not to say that features, even internal features, should be discarded in the name of simplicity. Indeed, the more elegant designs are usually the more simple ones. Simple also does not mean "quick and dirty." In fact, it often takes a lot of thought and work over multiple iterations to simplify. The payoff is software that is more maintainable and less error-prone.

<http://c2.com/cgi/wiki?SevenPrinciplesOfSoftwareDevelopment>

# Ideas behind the presentation



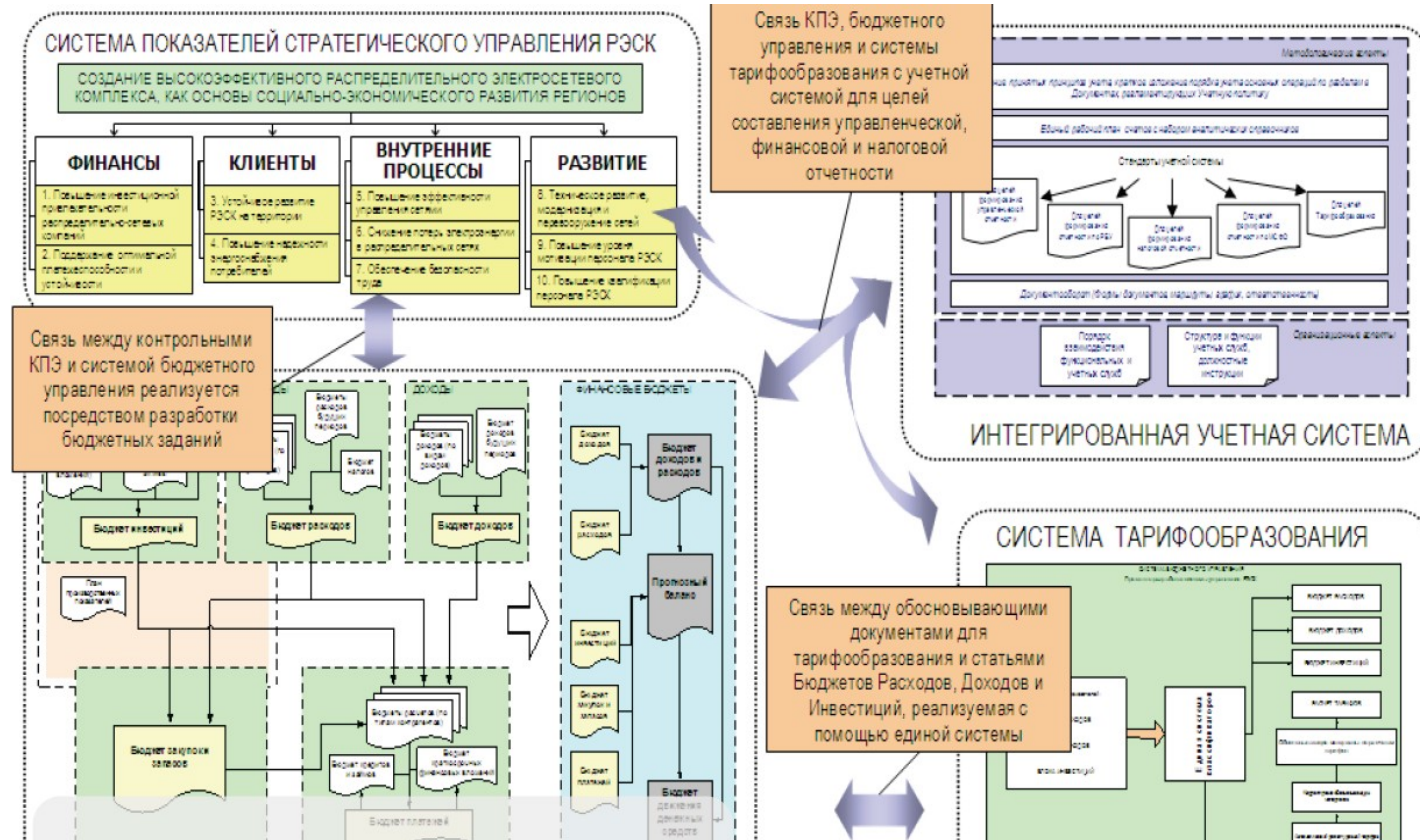
## iBatis



# This is not simple



Don't worry:  
knowing  
the language  
doesn't really help.





I actually enjoy complexity that's empowering. If it challenges me, the complexity is very pleasant. But sometimes I must deal with complexity that's disempowering. The effort I invest to understand that complexity is tedious work. It doesn't add anything to my abilities.

What's the Simplest Thing that Could Possibly Work ?

Simplicity: **the Shortest Path to a Solution**

Ward Cunningham

<http://www.artima.com/intv/simplest.html>  
<http://c2.com/cgi/wiki?WardCunningham>



Simplicity



!=

Quick and Dirty



# Do the simplest thing



The most important rule in our development is always to **do the simplest thing** that could possibly work.

**Not the most stupid thing**, not something that clearly can't work. But simplicity is the most important contributor to the ability to **make rapid progress**.

Ronald E. Jeffries

<http://www.xprogramming.com/Practices/PracSimplest.html>





## Simplicity is the Key

*Lessons  
Learned*

A simple design always takes less time to finish than a complex one. So always do the simplest thing that could possibly work. If you find something that is complex replace it with something simple. It's always faster and cheaper to replace complex code now, before you waste a lot more time on it. Keep things as simple as possible as long as possible by never adding functionality before it is scheduled. Beware though, keeping a design simple is hard work. ☹️

 [Wiki Wiki](#)  
The Portland  
Pattern Repository

 [XP](#)  
programming.com

 [Wiki Wiki](#)  
The Portland  
Pattern Repository

[ExtremeProgramming.org home](#) | [XP Rules](#) | [System Metaphor](#) | [Email the webmaster](#)

Copyright 1999 Don Wells all rights reserved.

# Do the simplest thing that could possibly work !





## 孫子兵法

Nella pianificazione, mai una mossa inutile;  
nella strategia, nessun passo compiuto invano.

Sun Tzu  
The art of war

# Interface in MagicBox



## EJ Item 16: Prefer **interfaces** to abstract classes

For **all** Objects

Domain  
Services  
Dao

- org.magicbox.domain 96
  - Amministratore.java 460
  - AmministratoreImpl.java 401
  - AmministratoreLight.java 401
  - AmministratoreLightImpl.java 401
  - Annuncio.java 401
  - AnnuncioImpl.java 463
  - Centro.java 401
  - CentroImpl.java 401
  - CentroLight.java 401
  - CentroLightImpl.java 401
  - Gruppo.java 401
  - GruppoImpl.java 401
  - Indirizzo.java 401
  - IndirizzoImpl.java 401
  - Page.java 401
  - PageImpl.java 401
  - PageLight.java 401
  - PageLightImpl.java 401
  - RecapitoTelefonico.java 401
  - RecapitoTelefonicoImpl.java 401
  - Storico.java 401
  - StoricoImpl.java 462
  - Utente.java 401
  - UtenteImpl.java 401
  - UtenteLight.java 401
  - UtenteLightImpl.java 401
  - UtenteUltraLight.java 401
  - UtenteUltraLightImpl.java 401

- org.magicbox.ibatis 593
  - AmministratoreCentroDaoImpl.java 550
  - AnnunciDaoImpl.java 475
  - GruppiDaoImpl.java 562
  - PagesDaoImpl.java 562
  - StoricoDaoImpl.java 562
  - UtentiDaoImpl.java 592

- org.magicbox.admin.service 542
  - AmministratoreManager.java 542
  - AmministratoreManagerImpl.java 542
  - AnnunciAdminService.java 467
  - AnnunciAdminServiceImpl.java 467
  - CentroManager.java 542
  - CentroManagerImpl.java 542
  - CredentialService.java 467
  - CredentialServiceImpl.java 537
  - InserimentoFacade.java 467
  - InserimentoFacadeImpl.java 542
  - RolesService.java 467
  - RolesServiceImpl.java 467

- org.magicbox.dao 582
  - AmministratoreCentroDao.java 542
  - AnnunciDao.java 469
  - GruppiDao.java 563
  - PagesDao.java 563
  - StoricoDao.java 563
  - UtentiDao.java 582



Felino micio = `new` Gatto();

Switch from one **implementation**,

```
List entries = new java.util.ArrayList();
```

```
Map map = new java.util.HashMap();
```

to another more **aggressive**

```
List entries = new javolution.util.FastList();
```

```
Map map = new javolution.util.FastHashMap();
```

without affecting the client



Felino micio = `new` TigerInsideImpl();





## EJ Item 14: Favor **composition** over inheritance

This design is called **composition** because the existing class becomes a component of the new one.

Each instance method in the new class invokes the corresponding method on the contained instance of the existing class and returns the results.

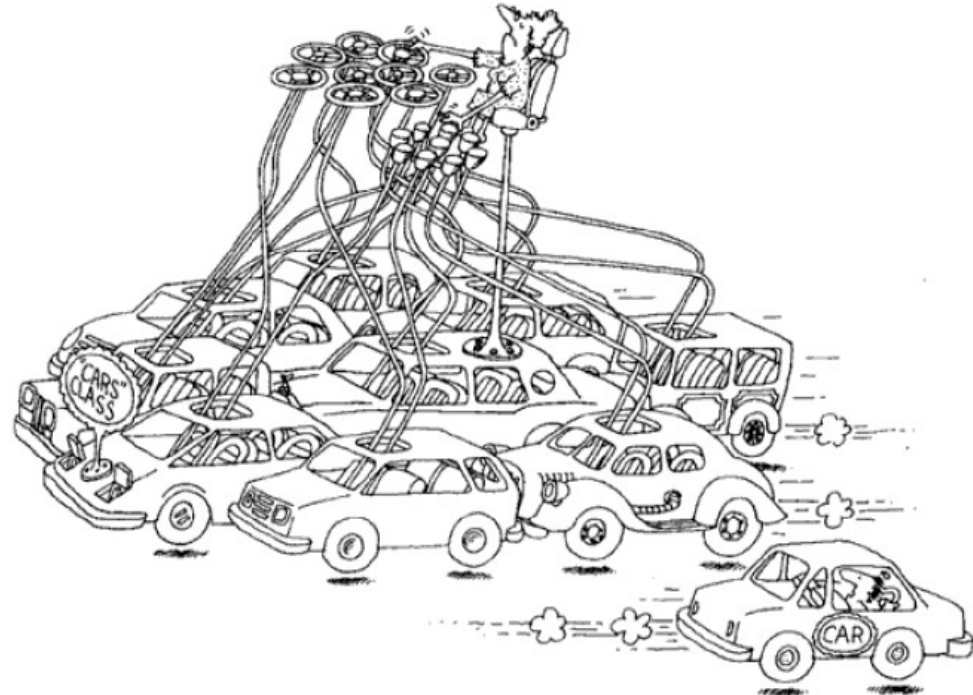
This is known as **forwarding**, and the methods in the new class are known as forwarding methods.



# Composition in Domain Objects



```
public final class UtenteImpl implements UtenteView{  
  
    public UtenteImpl (){  
        utenteLight = new UtenteLightImpl();  
        indirizzo = new IndirizzoImpl();  
        recapiti = new RecapitoTelefonicoImpl();  
    }  
    ...  
}
```



# Abstractions from a hierarchy

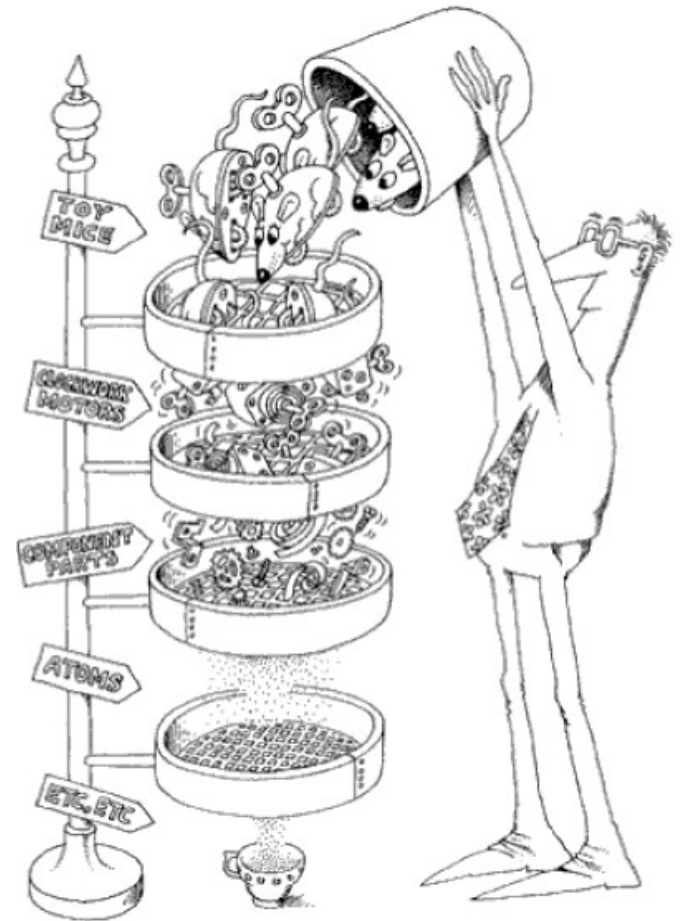


## Inheritance on interfaces

```
public interface UtenteView extends UtenteLightView{  
    ...  
}
```

```
public interface UtenteLightView extends UtenteUltraLightView{  
    ...  
}
```

```
public interface UtenteUltraLightView {  
    ...  
}
```





## Item 13: Favor **immutability**

An immutable class is simply a class whose instances cannot be modified.

All of the information contained in each instance is provided when it is created and is **fixed** for the **lifetime** of the object.

Immutable classes are easier to design, implement, and use than mutable classes. They are less prone to error and are more secure



Only **accessors** on public interfaces.

**Final** implementation class.

All fields **private**.

No **mutable** component.

**Mutators** accessible only on implementation class, for **web binding** and in the **Data Mapper** Layer

```
public interface UtenteUltraLightView {  
    public boolean isNew();  
    public long getId();  
    public String getNominativo();  
    public long getIdGruppo();  
}
```

```
public class CentroFormController extends SimpleFormController{  
    public CentroFormController(){  
        setCommandClass(CentroImpl.class);  
        setCommandName(Constant.CENTRO);  
        setFormView("admin/centroForm");  
        setSuccessView(Constant.REDIRECT_ELENCO_CENTRI);  
    }  
    ...  
}
```



The Data Mapper is a layer of software that separates the in-memory objects from the database. Its responsibility is to transfer data between the two and also to isolate them from each other. With Data Mapper the in-memory objects needn't know even that there's a database present; they need no SQL interface code, and certainly no knowledge of the database schema. (The database schema is always ignorant of the objects that use it.) Since it's a form of Mapper (473), Data Mapper itself is even unknown to the domain layer.

<http://martinfowler.com/eaCatalog/dataMapper.html>

# iBATIS mapping class - table



```
<typeAlias type="org.magicbox.domain.AmministratoreImpl" alias="admin"/>
<typeAlias type="org.magicbox.domain.AmministratoreLightImpl" alias="adminLight"/>

<resultMap class="adminLight" id="resultAdminLight">
  <result column="id" property="id" jdbcType="INTEGER" />
  <result column="centro" property="centro" jdbcType="BIGINT" />
  <result column="nominativo" property="nominativo" jdbcType="VARCHAR" />
</resultMap>

<resultMap class="admin" id="resultAdmin" extends="resultAdminLight">
  <result column="email" property="email" jdbcType="VARCHAR" />
  <result column="username" property="username" jdbcType="VARCHAR" />
  <result column="cellulare" property="cellulare" jdbcType="VARCHAR" />
</resultMap>

<select id="getAdminByUsername" resultMap="resultAdmin">
  SELECT
    id, centro, email, cellulare, nominativo, username
  FROM amministratori
  WHERE username = #username#
</select>
...

<insert id="insertAdmin" parameterClass="admin">
  <selectKey keyProperty="id" resultClass="long">
    select nextval('amministratori_id_seq')
  </selectKey>
  INSERT INTO amministratori (id, centro, nominativo, email, username, cellulare)
  VALUES (#id:INTEGER#, #centro:BIGINT#, #nominativo:VARCHAR#,
    #email:VARCHAR#, #username:VARCHAR#, #cellulare:VARCHAR#)
</insert>
...
```





```
public class AmministratoriDaoImpl extends SqlMapClientDaoSupport implements AmministratoriDao{

    public Integer deleteAmministratore(long idAdmin, long idCentro) throws DataAccessException{
        Map map = new FastMap();
        map.put("id", idAdmin);
        map.put("centro", idCentro);
        return getSqlMapClientTemplate().delete("deleteAdmin", map);
    }

    public Long insertAmministratore(AmministratoreView admin) throws DataAccessException {
        return (Long)getSqlMapClientTemplate().insert("insertAdmin", admin);
    }

    public Integer updateAmministratore(AmministratoreView admin) throws DataAccessException {
        return getSqlMapClientTemplate().update("updateAdmin", admin);
    }

    public AmministratoreView getAmministratore(String username) throws DataAccessException{
        return (AmministratoreView)getSqlMapClientTemplate().queryForObject("getAdminByUsername", username);
    }

    ..
}
```

# Transaction on service layer



```
public class AmministratoreManagerImpl implements AmministratoreManager{

    public long saveAmministratore(AmministratoreView admin) {
        return admin.isNew() ? amministratoreDao.insertAmministratore(admin) : amministratoreDao.updateAmministratore(admin);
    }

    public AmministratoreView getAmministratore(String username) {
        return amministratoreDao.getAmministratore(username);
    }

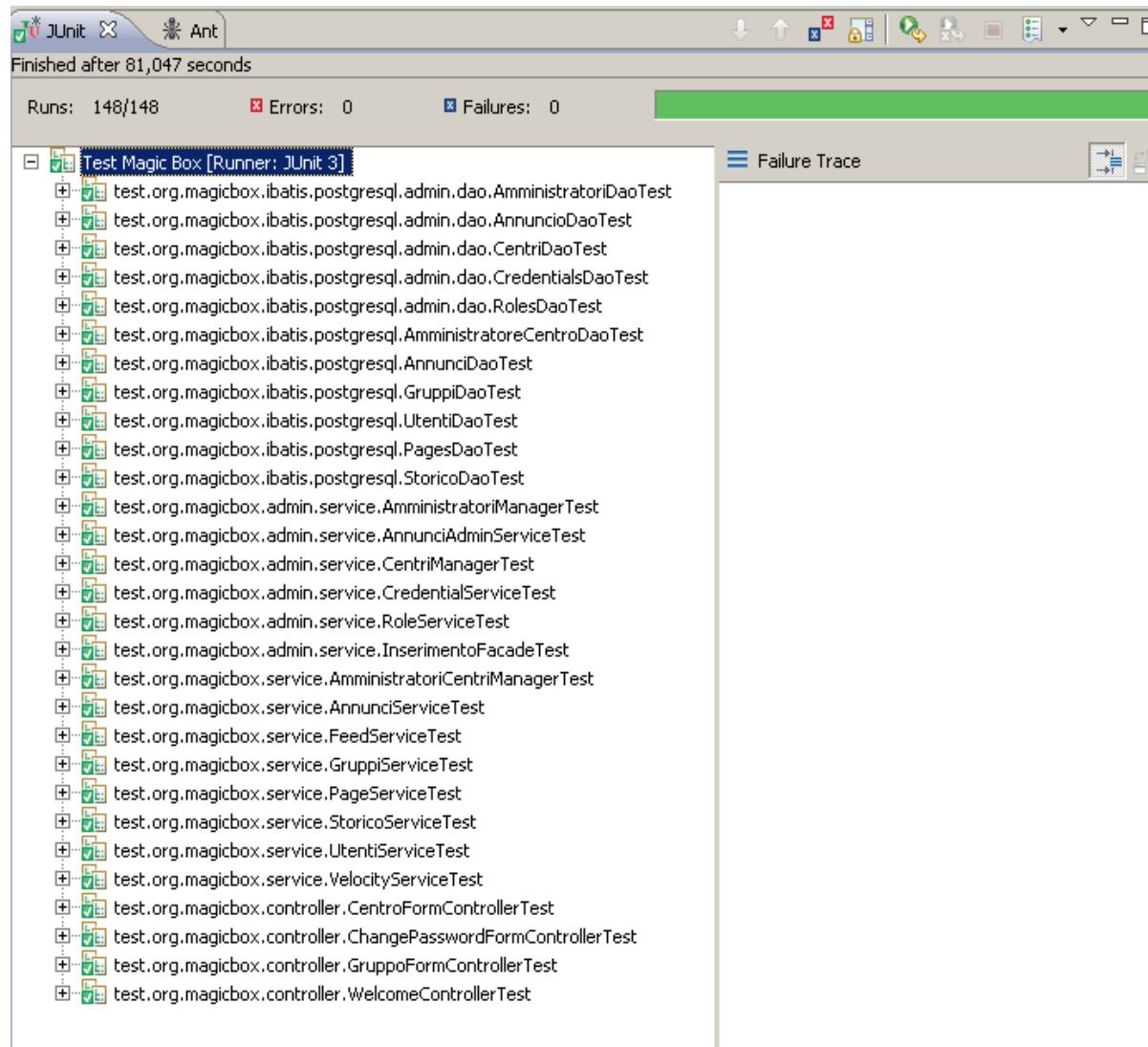
    public boolean deleteAmministratore(long idAdmin, long idCentro) {
        return amministratoreDao.deleteAmministratore(idAdmin, idCentro) == 1 ? true : false;
    }

    public List<AmministratoreView> getAmministratori() {
        return amministratoreDao.getAmministratori();
    }
}
```

## Spring Way: declarative

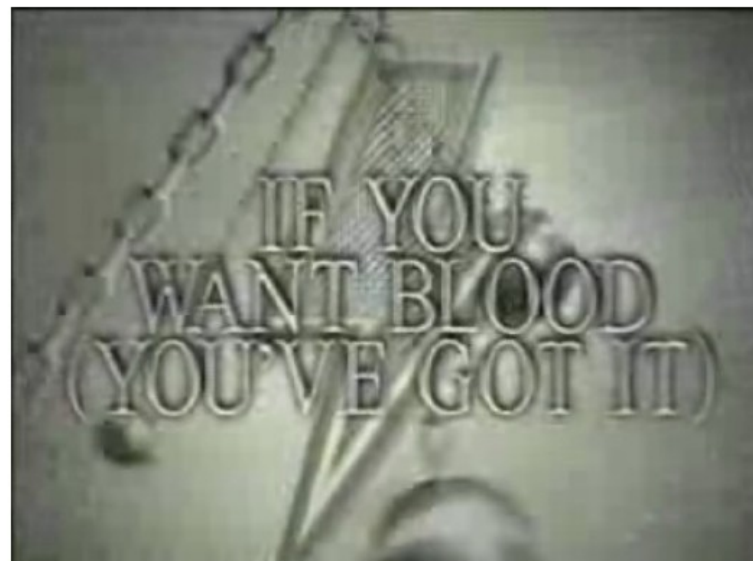
```
<bean name="txAttributes" class="org.springframework.transaction.interceptor.NameMatchTransactionAttributeSource">
    <property name="properties">
        <props>
            <prop key="*">PROPAGATION_SUPPORTS, readOnly</prop>
            <prop key="save*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="insert*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="update*">PROPAGATION_REQUIRED, -Exception</prop>
            <prop key="delete*">PROPAGATION_REQUIRED, -Exception</prop>
        </props>
    </property>
</bean>
```

# Test !





MagicBox it' s a JEE solution to coordinate **blood donations**





All this leads to  
**great  
software  
projects**



# Thanks for your attention



Massimiliano Dessì  
desmax74 at yahoo.it

<http://wiki.java.net/bin/view/People/MassimilianoDessi>  
<http://www.jugsardegna.org/vqwiki/jsp/Wiki?MassimilianoDessi>  
<http://jroller.com/desmax>

Spring Framework Italian User Group  
<http://it.groups.yahoo.com/group/SpringFramework-it>

