# The Spring Framework

Simplifying J2EE

VIRTUAS

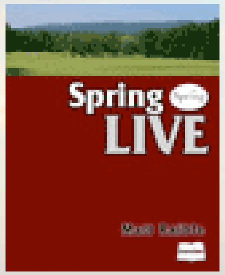
## Introductions

- Your experience with Spring?
- Your experience with J2EE?
- What do you hope to learn in this session?
- Open Source experience: Ant, XDoclet, Hibernate?
- Favorite IDE? Favorite OS?

VIRTUAS

# Who is Matt Raible?

- Developing websites since 1994 - Developing J2EE webapps since 1999
- Committer on several open source projects: AppFuse, Roller Weblogger, XDoclet, Struts Menu, Display Tag
- J2EE 5.0 and JSF 1.2 Expert Group Member
- Author: Spring Live (SourceBeat) and contributor to Pro JSP (Apress)



# Spring Mission Statement

- J2EE should be easier to use.
- It's best to program to interfaces, rather than classes. Spring reduces the complexity of using interfaces to zero.
- JavaBeans offer a great way of configuring applications.
- OO Design is more important than any implementation technology, such as J2EE.

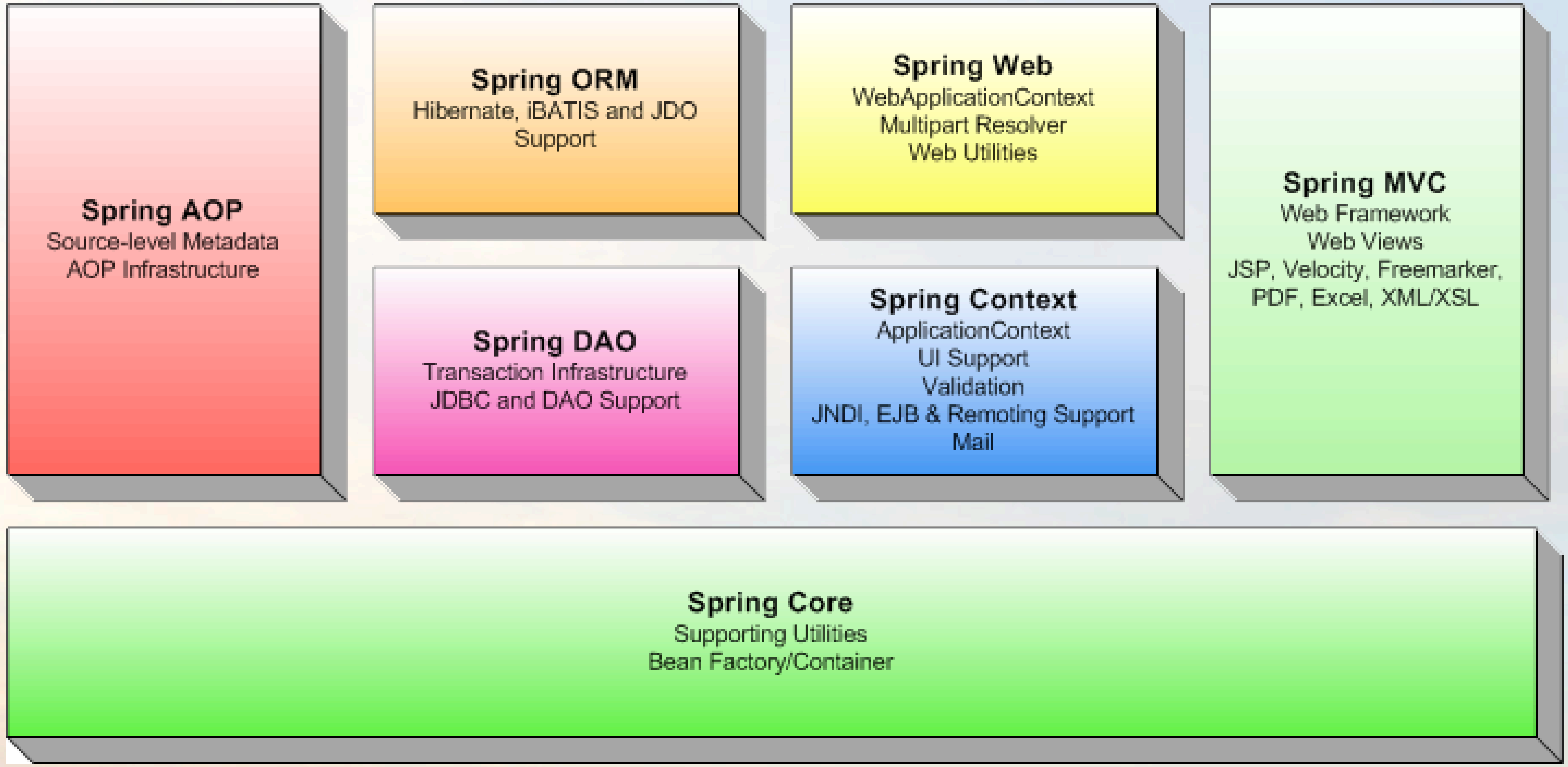
## Spring Mission Statement

- Checked exceptions are overused in Java.  A framework shouldn't force you to catch exception you're unlikely to recover from.
- Testability is essential, and a framework such as Spring should help make your code easier to test.

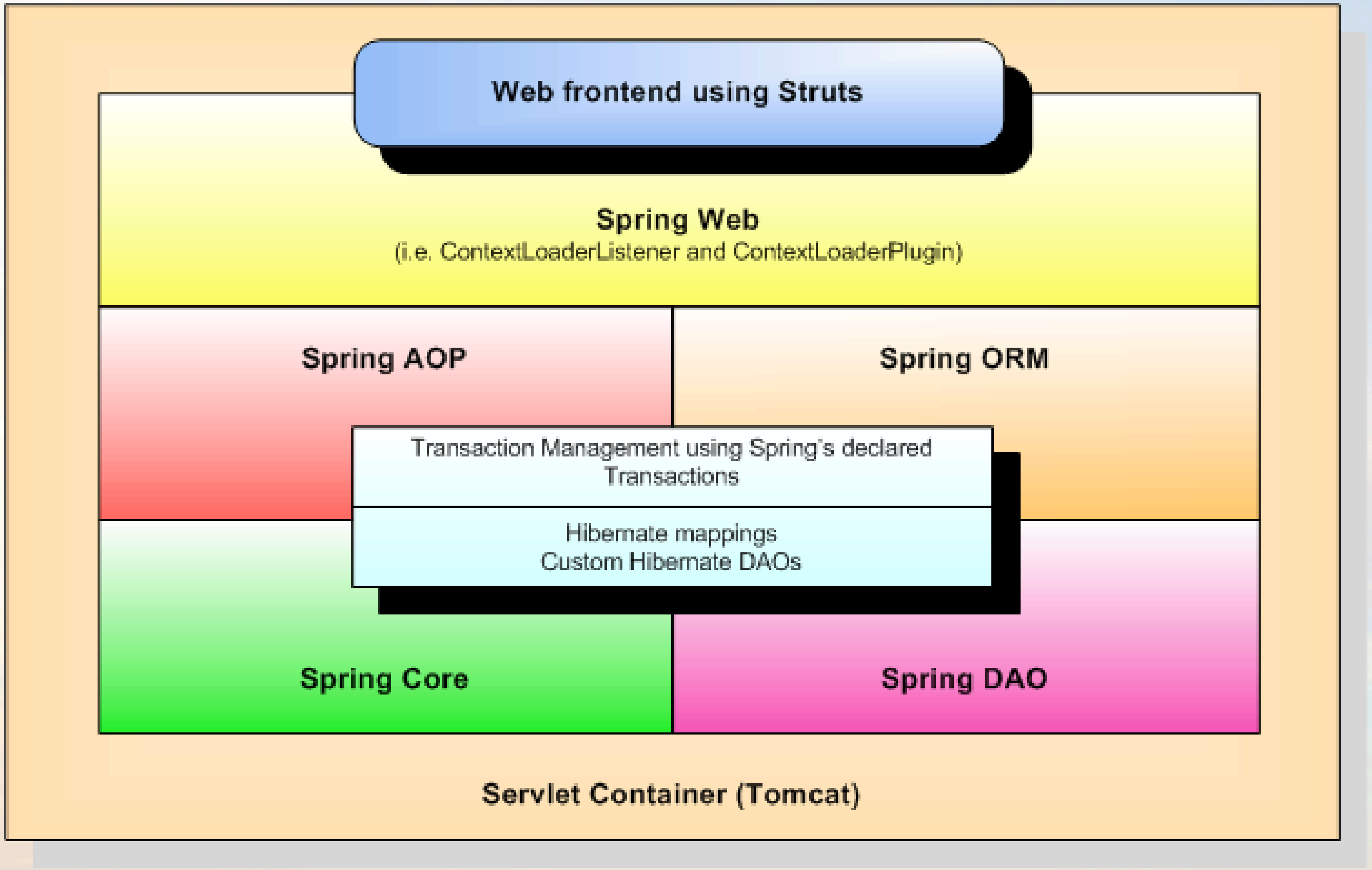
## What is Spring?

- A J2EE Framework designed to make building applications easier
- Provides a means to manage your business objects and their dependencies
- Inversion of Control allows classes to be loosely coupled and dependencies written in XML
- À la carte framework that allows you to pick and choose features to use

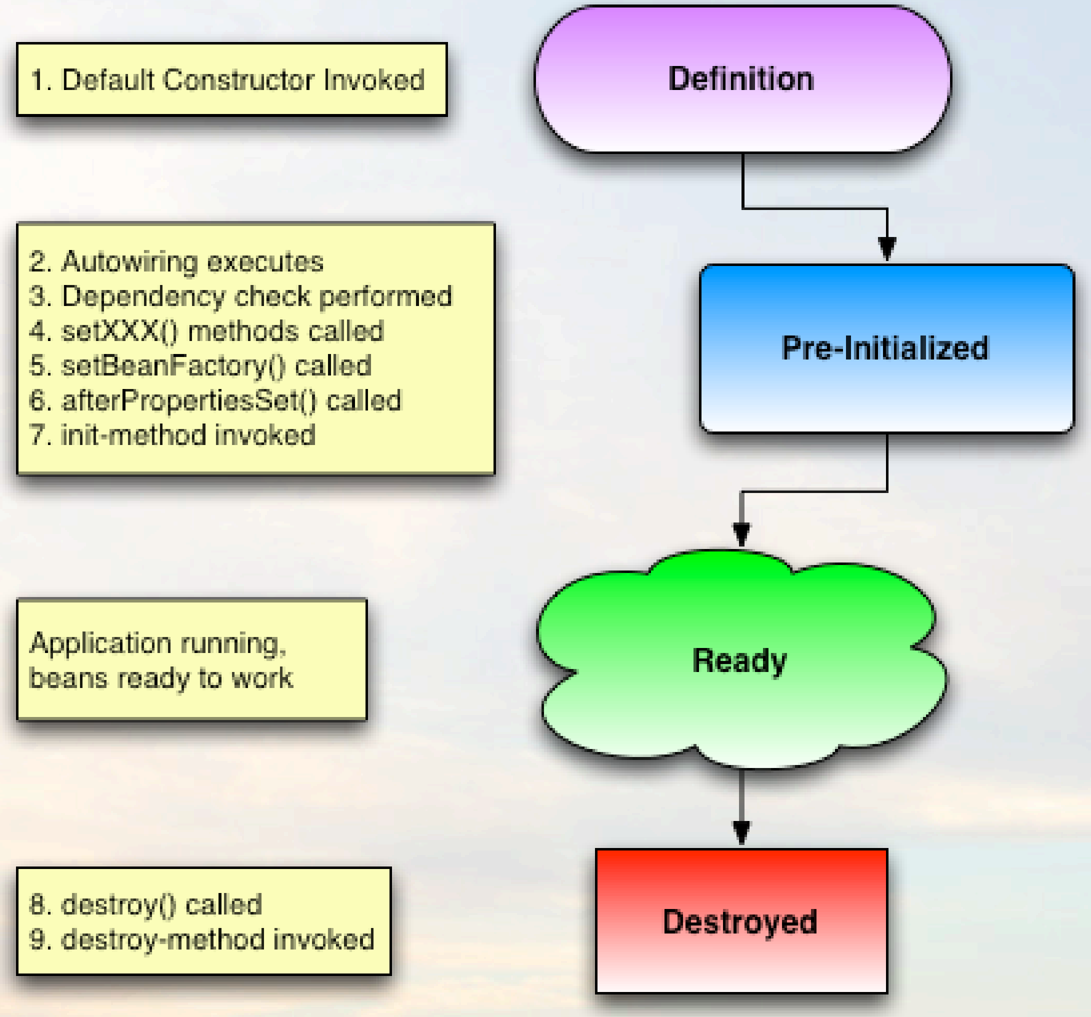
# Spring Modules



# Sample Architecture


Web frontend using Struts
Spring Web
(i.e. ContextLoaderListener and ContextLoaderPlugin)
Spring AOP
Spring ORM
Transaction Management using Spring's declared Transactions
Hibernate mappings
Custom Hibernate DAOs
Spring Core
Spring DAO
Servlet Container (Tomcat)

# The BeanFactory


1. Default Constructor Invoked
Definition
2. Autowiring executes
3. Dependency check performed
4. setXXX() methods called
5. setBeanFactory() called
6. afterPropertiesSet() called
7. init-method invoked
Pre-Initialized
Application running, beans ready to work
Ready
8. destroy() called
9. destroy-method invoked
Destroyed

# Dependency Injection

- Hollywood Principle:

  **“Don’t call me, I’ll call you.”**

- http://martinfowler.com/articles/injection.html
- Lookup dependencies vs. Exposing dependencies
- Easy to mock dependencies (i.e. in unit tests)
- Spring supports constructor injection, setter injection and method injection

## Example: Action and DAO

- Action has a setter or a constructor argument for the DAO
- DAO is configured as a dependency of the Action

```
<bean id="userAction" class="org.appfuse.web.UserAction">
    <property name="userDAO" ref="userDAO"/>
</bean>

<bean id="userDAO" class="org.appfuse.dao.UserDAOHibernate">
    <property name="sessionFactory" ref="sessionFactory"/>
</bean>
```

- Makes it easy to swap out implementations
- Tests can verify operations succeed on the interface

## Code before IoC

```java
public class UserController implements Controller {

    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response)
    throws Exception {
        Connection conn = DatabaseUtils.getConnection();
        UserDAO dao = DAOFactory.createUserDAO("hibernate", conn);

        List users = dao.getUsers();

        DatabaseUtils.closeConnection(conn);

        return new ModelAndView("userList", "users", users);
    }
}
```

## Code after IoC

```
public class UserController implements Controller {
    private UserDAO dao = null;

    public void setUserDAO(UserDAO userDAO) {
        this.dao = userDAO;
    }

    public ModelAndView handleRequest(HttpServletRequest request,
                                      HttpServletResponse response)
    throws Exception {

        List users = dao.getUsers();

        return new ModelAndView("userList", "users", users);
    }
}
```

## Spring vs. J2EE

- It's not really a comparison, Spring builds off J2EE and merely makes the APIs easier
- It's still using many of J2EE's and Java's underlying infrastructure
- EJB vs. Spring Managed POJOs
- Opinions and Hesitations?

# Spring simplifies EJBs

- Abstract classes in org.springframework.ejb.support override methods that you normally don't implement
- Expose EJBs as Spring beans with classes in org.springframework.ejb.access package

```
<bean id="simpleService" lazy-init="true"
    class="org.springframework.ejb.access.SimpleRemoteStatelessSessionProxyFactoryBean">
    <property name="jndiName" value="jndiName"/>
    <property name="cacheHome" value="true"/>
    <property name="businessInterface" value="org.appfuse.ejb.SimpleService"/>
</bean>
```

- Configuring a JtaTransactionManager will allow Spring beans to participate in CMT

# Spring MVC

- Front-Controller Servlet: DispatcherServlet
- Controllers loaded as beans from servlet's XML context files
- ContextLoaderListener loads other XML context files
- Many different Controller options: Controller, SimpleFormController, Wizard

# PropertyEditors

- PropertyEditors allow you to convert Dates, Integers and custom types
- Built-in PropertyEditor examples: ByteArrayPropertyEditor (file upload), CustomDateEditor, CustomNumberEditor
- Register for context files by defining CustomEditorConfigurer bean
- Register in Spring MVC in initBinder() method of SimpleFormController

## Examples: Controllers

- The Controller interface is generally used to display data, not for forms
- SimpleFormController easiest for forms
- JSP Tags, Velocity, FreeMarker, PDF, Excel
- Validation: Commons Validator or Validation classes
- Supports Tiles for page composition

## Page Decoration and Composition

- Tiles vs. SiteMesh
- Tiles works great for page composition - soon to be come top-level project at Apache
- SiteMesh is easiest to use for page decoration
- Can be used together or separately

# Spring Supports many Java MVC Frameworks

- ContextLoaderListener loads beans into ServletContext - can retrieve with:

```
WebApplicationContext ctx =
            WebApplicationContextUtils.getWebApplicationContext(servletContext);
```

- Struts has ContextLoaderPlugin and ActionSupport classes
- WebWork has SpringObjectFactory
- Tapestry - override BaseEngine and stuff the context into the Global
- JSF - DelegatingVariableResolver

# Integrating Struts with Spring

- ContextLoaderPlugin - match action “path” with bean “name”
  - Override default RequestProcessor with Spring's DelegatingRequestProcessor
  - Use the DelegatingActionProxy class in the "type" attribute of your <action-mapping>
  - TIP: Load all context files in plugin to make testing with StrutsTestCase easier
- Use a Spring subclass of the standard Struts Actions and call getWebApplicationContext()

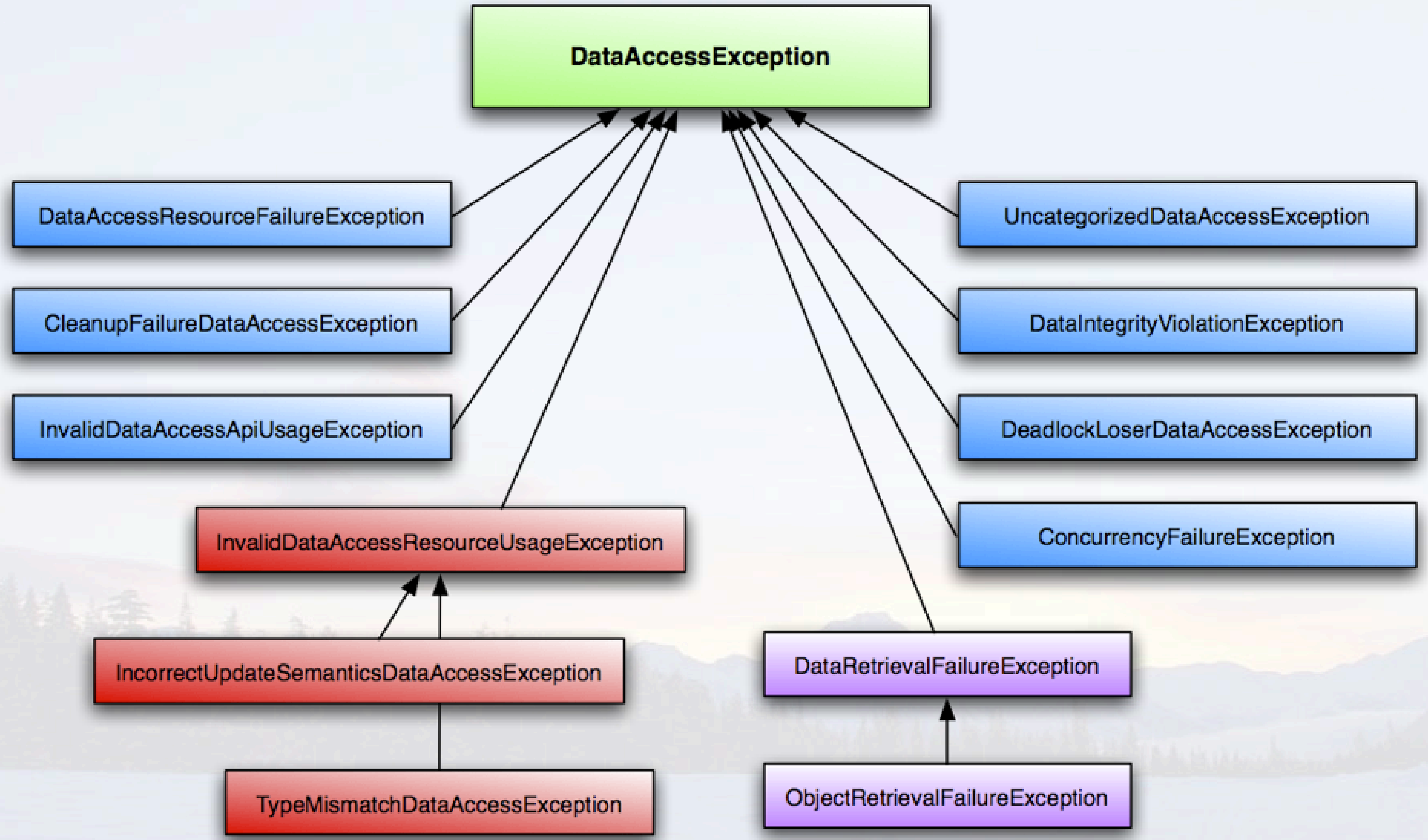
## MVC Tips

- Using IoC for setting Controller/Action dependencies makes it easy to mock them
- JSP 2.0 Tag files in JIRA - for syntax simplification
- Use SiteMesh - simple yet powerful
- AppFuse and Equinox have good integration examples

## Data Access

- Spring JDBC Framework
  - No exceptions to catch or resources to close
- Base DAO and Template classes for many frameworks:
  - Hibernate, iBATIS, OJB, JDO - even TopLink

# DataAccessException


DataAccessException
DataAccessResourceFailureException
CleanupFailureDataAccessException
InvalidDataAccessApiUsageException
InvalidDataAccessResourceUsageException
IncorrectUpdateSemanticsDataAccessException
TypeMismatchDataAccessException
UncategorizedDataAccessException
DataIntegrityViolationException
DeadlockLoserDataAccessException
ConcurrencyFailureException
DataRetrievalFailureException
ObjectRetrievalFailureException

# Data Access Examples

- UserDAOTest
- Hibernate
- iBATIS
- Spring JDBC
- JDO

## Transactions

- Much easier to use than UserTransaction in J2EE
- CMT-like capabilities with XML and transparent AOP
- Supports Commons Attributes and JDK 5 Annotations
- Pluggable Transaction Managers, including JTA

# Testing Spring Applications

- Easy to mock dependencies with EasyMock and jMock
- Load context in tests and bind dependencies as you would in production
- TIP: Use a static block or Spring classes in Spring Mock (in org.springframework.test package)

# AOP

- AOP - Reduces duplication among classes
- Interceptors make it easy to add before, around and after method advice
- Useful for caching, logging and security
    - EHCache, Performance/Tracing interceptors, Acegi for Security

## App Server Support

- Spring should run on any app server
- Hooks into many app server's Transaction Manager
- Built-in Support in Geronimo
- Survey: which app servers are you running it on?

## Tools

- Spring IDE Plugin For Eclipse
  - http://springide.org
- BeanDoc
  - http://opensource.atlassian.com/confluence/spring/display/BDOC/Home
- Gaijin Studio
  - http://gaijin-studio.sf.net
- Screenshots ➪

# Resources

- Web: www.springframework.org and forum.springframework.org
- Print: Spring Live, Pro Spring, Spring in Action and Professional Spring Development
- Examples: AppFuse (appfuse.dev.java.net), Equinox (equinox.dev.java.net), JPetstore
- Training: Virtuas, Interface21, Arc-Mind