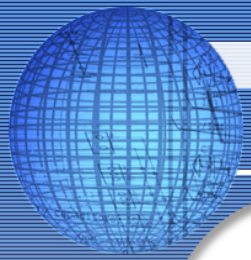
HitachiSoft

DIコンテナ Spring Frameworkによる次世代Java EEアプリケーション開発

河村 嘉之
日立ソフト
研究部
技師/ソリューションアーキテクト



自己紹介

HitachiSoft

河村 嘉之

- 日立ソフトウェアエンジニアリング㈱
  - 研究部
  - 技師/ソリューション・アーキテクト
  - http://hitachisoft.jp/research/techdoc

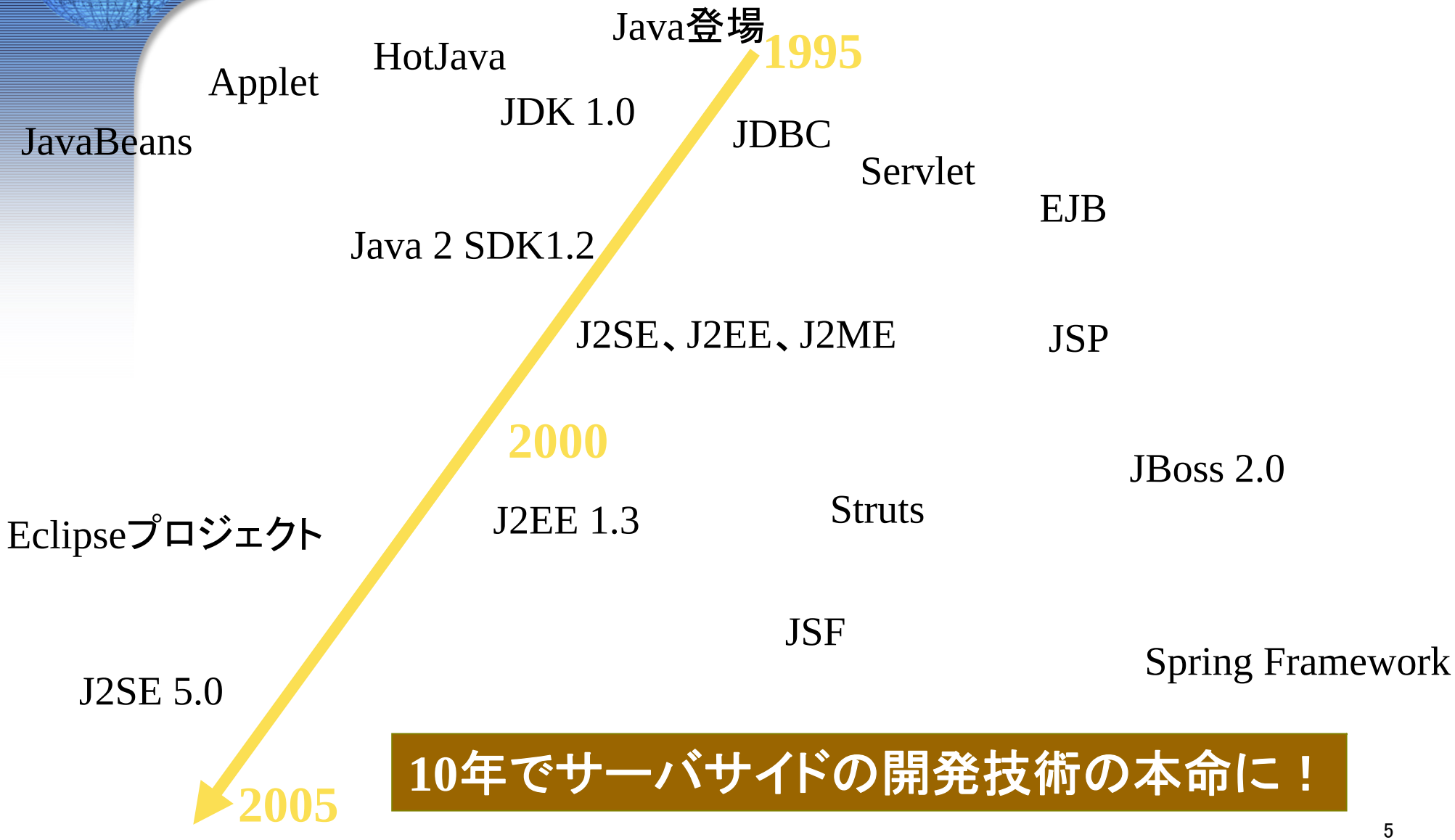
目次

- J2EE 現在までの道のりと問題点
- アプリケーション開発の新しい流れ
  - DIコンテナ
  - O/Rマッピングツール
  - Webフレームワークとの連携
  - Ease of Testing
  - チーム構成
- Java EE5、EJB3.0に向かって

目次

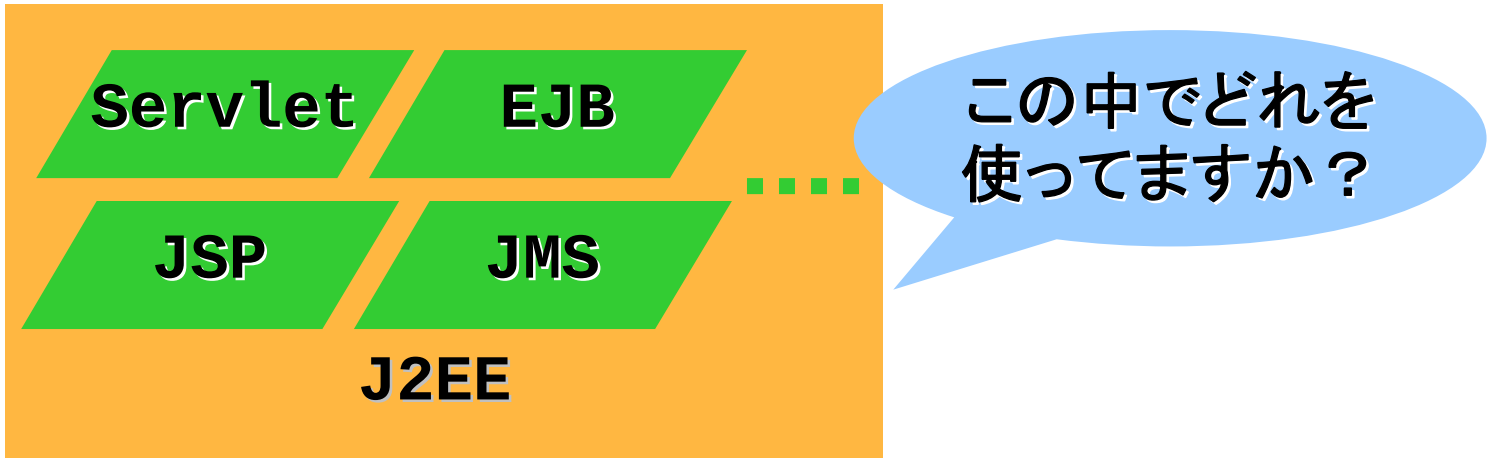
- J2EE 現在までの道のりと問題点
- アプリケーション開発の新しい流れ
  - DIコンテナ
  - O/Rマッピングツール
  - Webフレームワークとの連携
  - Ease of Testing
  - チームビルディング
- Java EE5、EJB3.0に向かって

Java 10年の歴史



10年でサーバサイドの開発技術の本命に！

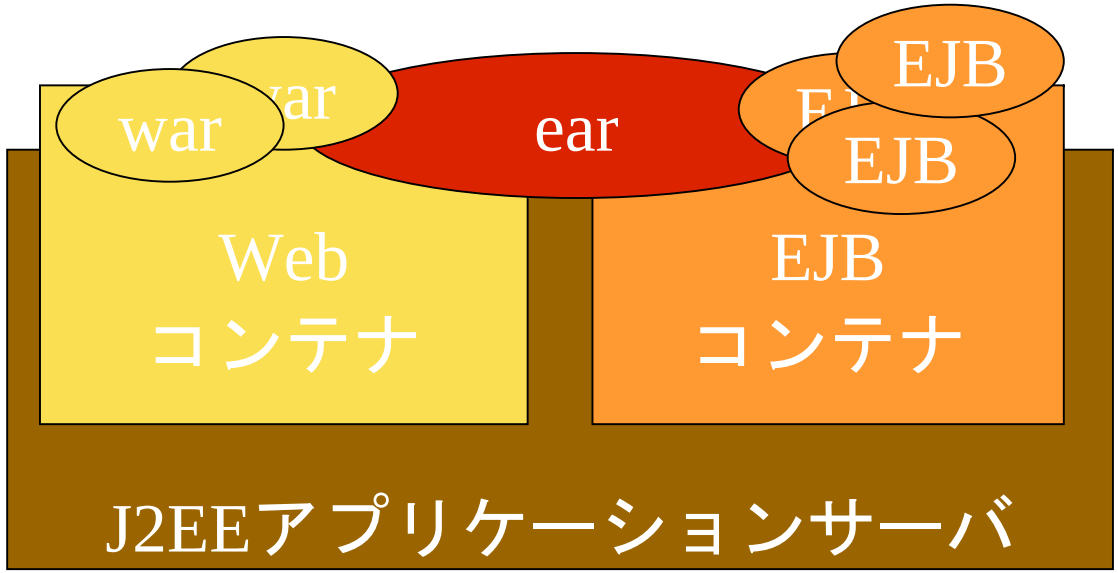
J2EE(EJB)の問題点



EJBを使っていますか？

- テストしづらい
- インターフェース、DDといった構成要素が多い
- プログラミング上の制約も多い

J2EEアプリケーションの特性



コンテナがないとプログラムが動かない
各コンポーネントは、**コンテナに依存**

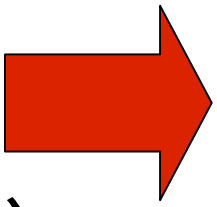
コンテナとコンポーネントは切っても切れない関係

HitachiSoft

コンテナベースの開発

たとえば、EJB

- 構成要素が多い
  - インターフェース x 4
  - デプロイメントディスクリプター
- プログラミング上の制約が多い
  - コールバックメソッド
  - 特定のインターフェースを実装
- デプロイしないと動かない
  - テストの効率が悪い



難しい！！

- 開発が難しい
- テストが難しい
- 習得が難しい

目次

- J2EE 現在までの道のりと問題点
- アプリケーション開発の新しい流れ
  - DIコンテナ
  - O/Rマッピングツール
  - Webフレームワークとの連携
  - Ease of Testing
  - チーム構成
- Java EE5、EJB3.0に向かって

いままでのアプリケーション開発

Web
フレーム
ワーク

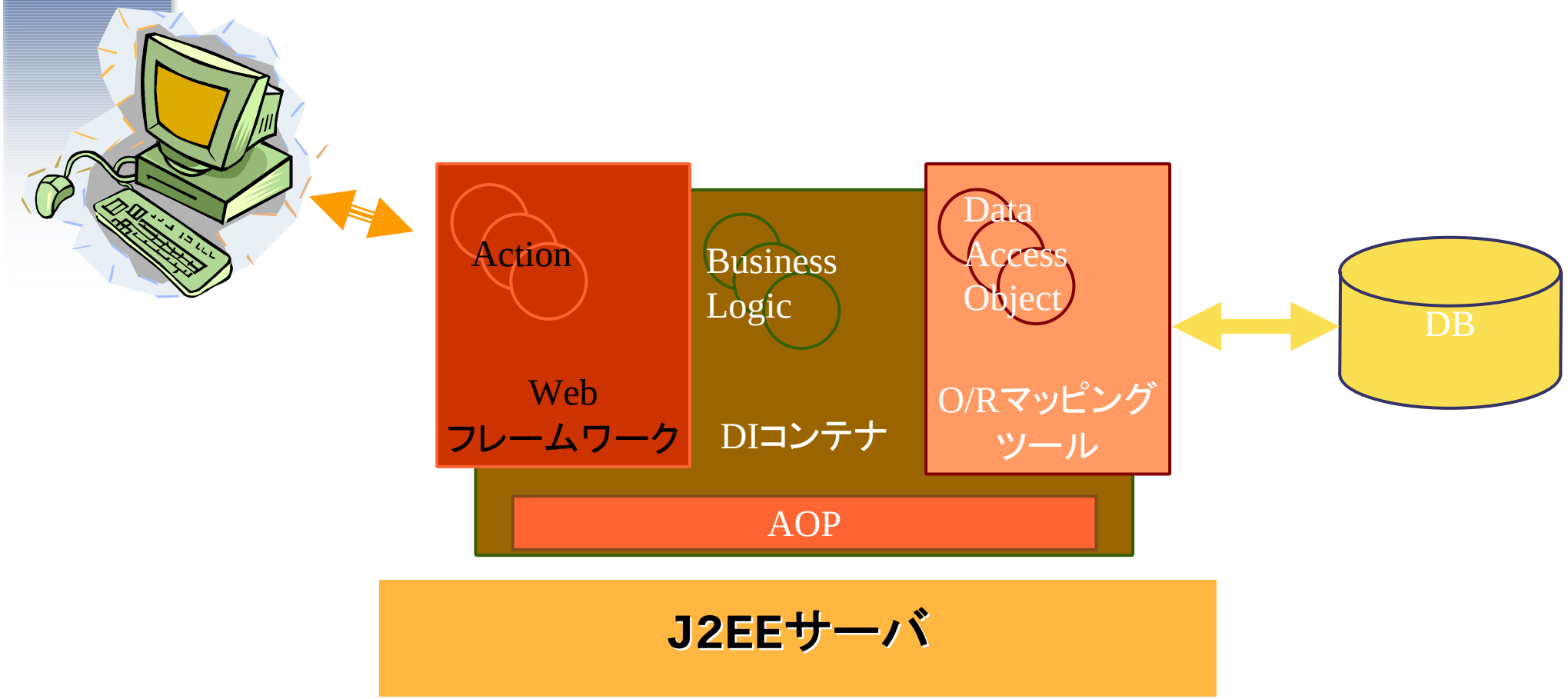
Business
Logic

EJB

DB

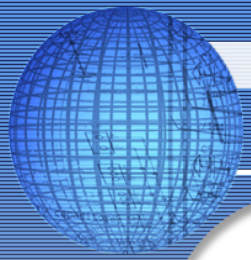
J2EEサーバ

アプリケーション開発の新しい流れ



DIコンテナ

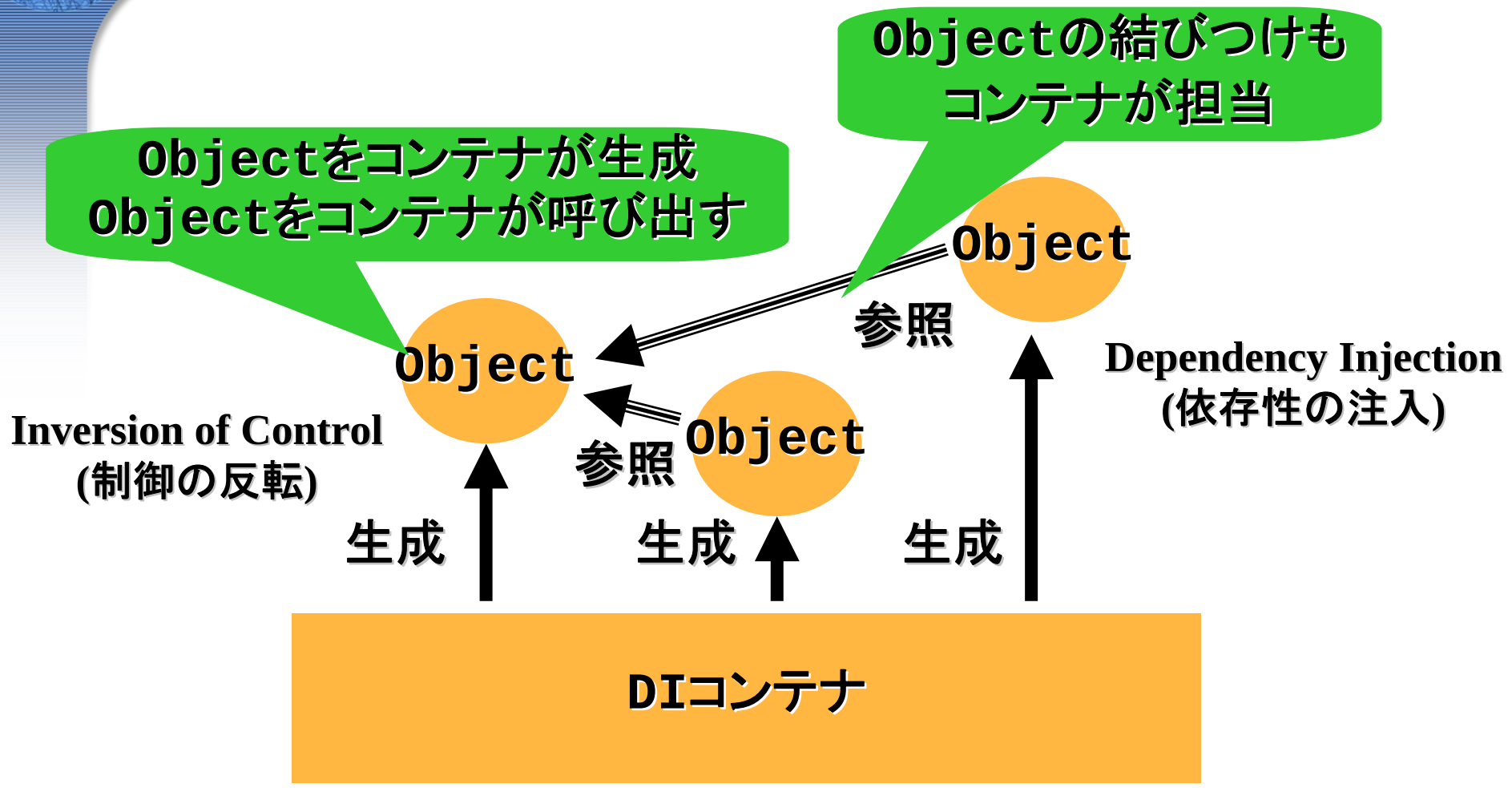
- DIコンテナとは？
  属するレイヤの異なるコンポーネントを組み合わせるもの
  - POJOの利用(コンテナ非依存・テストが簡単)
  - IoC/DIによるコンテナによるオブジェクト管理
  - AOPのサポート
- 代表的なDIコンテナ
  - Spring Framework
  - Seasar2
  - PicoContainer



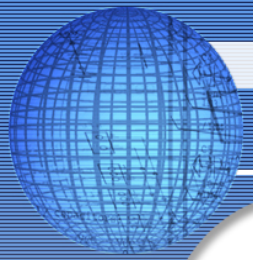
POJO

- POJO (Plain Old Java Object)
  - 特定のクラスを継承していたり、特定のインターフェースを実装していたりしない、シンプルな Java オブジェクト
  - 特定のフレームワークに依存しない。
  - 簡単に生成でき、安心して使える。
- POJOではないもの
  - Entity Beanを使うときは？

IoC/DI



インターフェースドリブンの開発
＝ レイヤ間の疎結合を実現

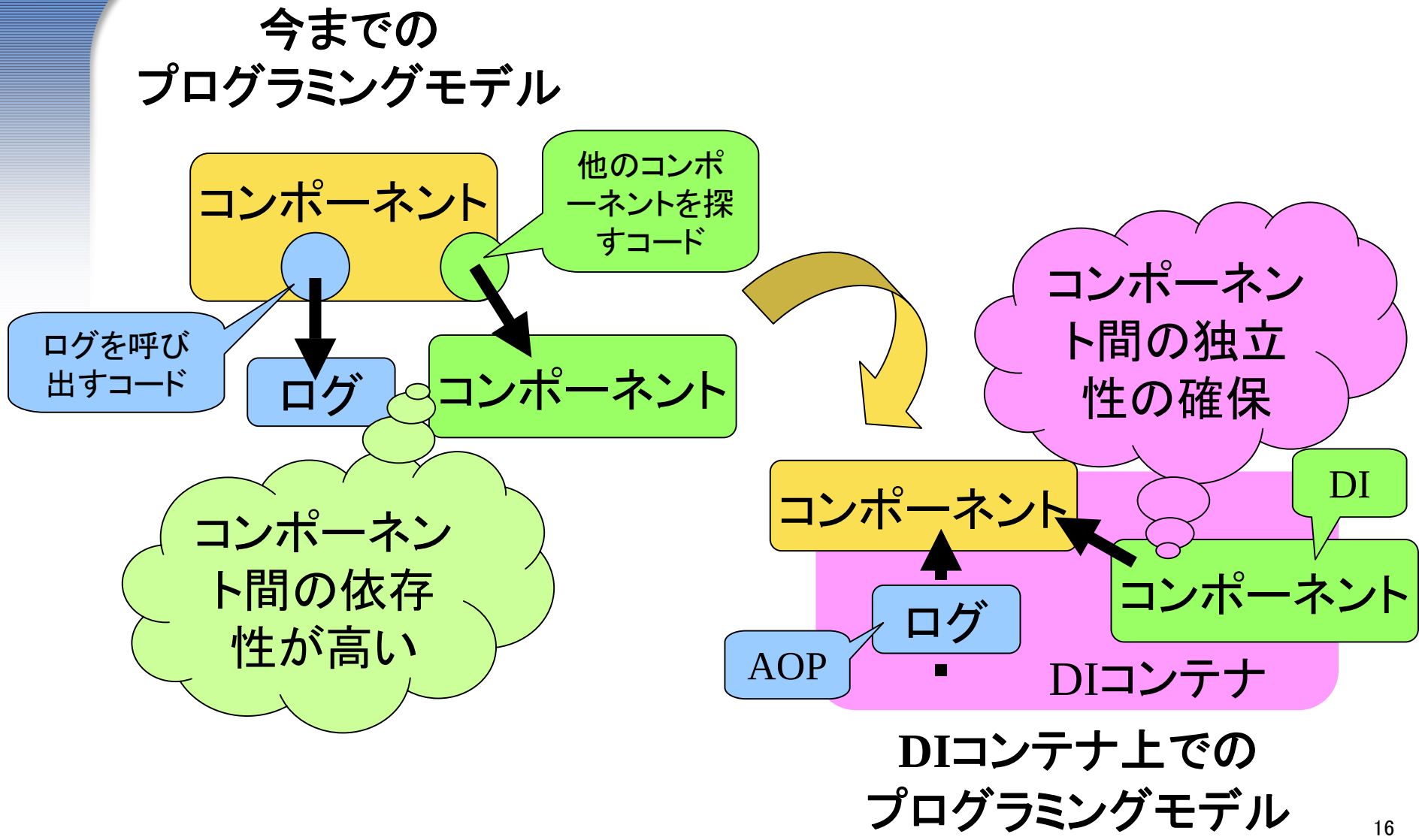


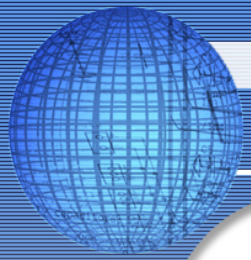
AOP

HitachiSoft

- Aspect Oriented Programming
  (アスペクト指向プログラミング)
  - プログラムの本来の目的とは違う処理(アスペクト)を切り出す
  - アスペクトを外から織り込む
  - プログラムの本来の処理とアスペクトの密結合を防ぐ
- AOPの使いどころ
  - ログの出力
  - トランザクション処理 など
- DIコンテナ＋AOP
  - アスペクトの織り込みを設定ファイルを用いて行う

依存性の排除





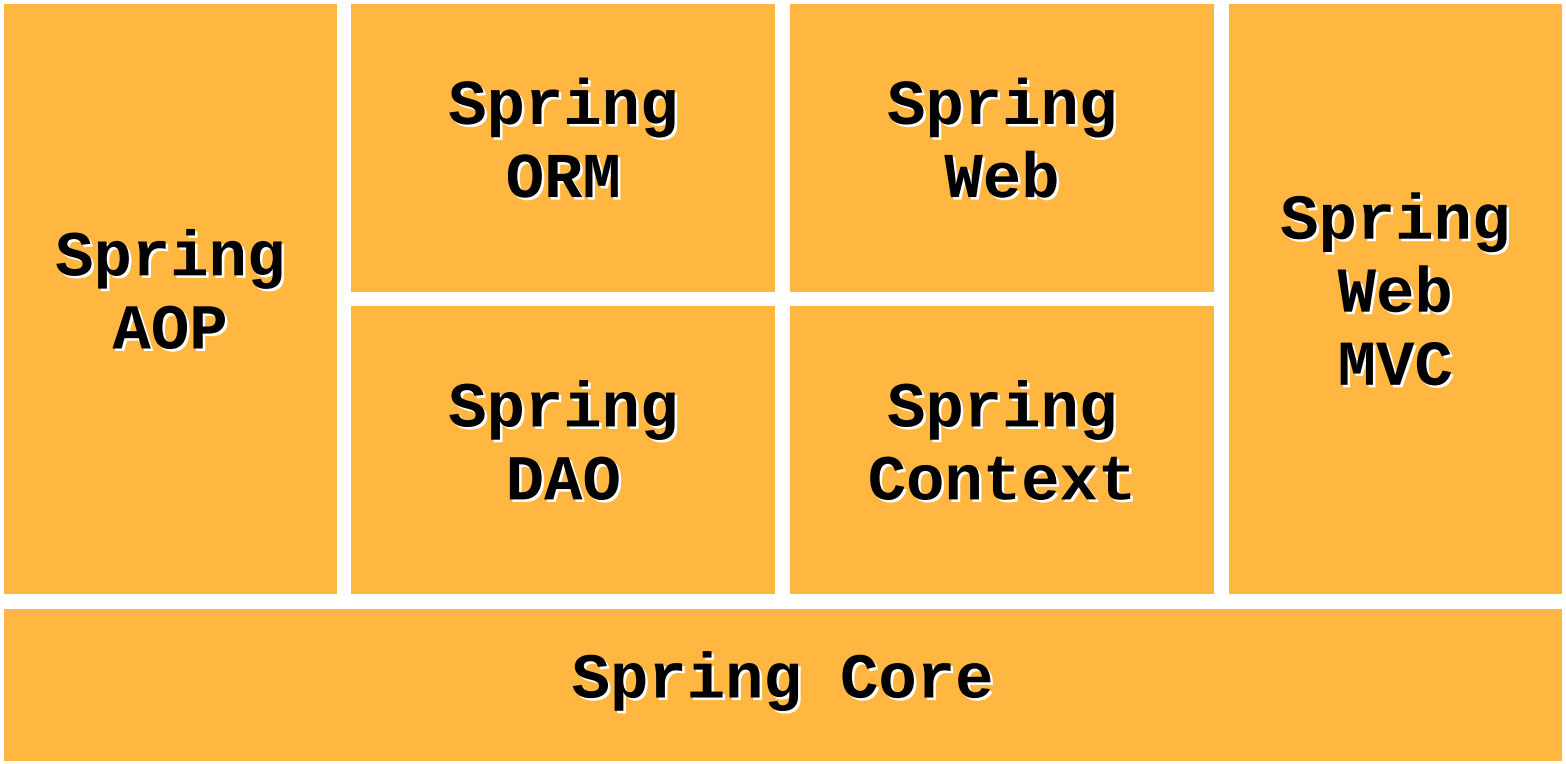
Spring Framework

HitachiSoft

Spring

- Rod Johnson氏が中心となって開発したJavaフレームワーク
- http://www.springframework.org にて公開
- 最新版は1.2.6 (2005年11月現在)
- IoC/DIをベースとしたフレームワーク
- DIコンテナ
  - 各要素がPOJOで構成
- Web、MVC、DAO、AOPなど豊富な機能を持つ

Spring Frameworkの構成要素



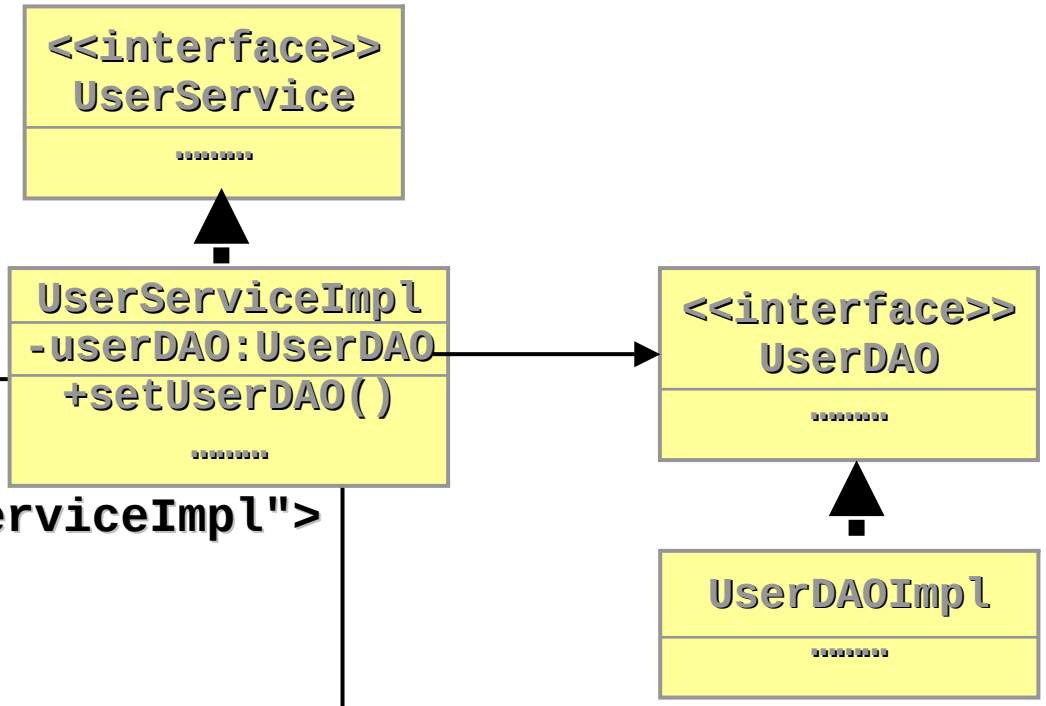
ORMとしてHibernate、JDO、
Webフレームワークとして Strutsなど
既存のフレームワークを有効に活用できる

Spring Framework コード例

HitachiSoft

```
BeanFactory bf =
        new ClassPathXmlApplicationContext("bean.xml");
UserService service =
        (UserService)bf.getBean("userService");
```

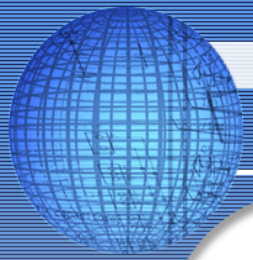
Beanを利用するコード



設定ファイル(bean.xml)

```
<beans>
 <bean id ="userService"
   class="sample.service.UserServiceImpl">
  <property name="userDAO">
   <ref local="userDAO"/>
  </property>
 </bean>
 <bean id="userDAO"
   class="sample.dao.UserDAOImpl">
</beans>
```

関連付け



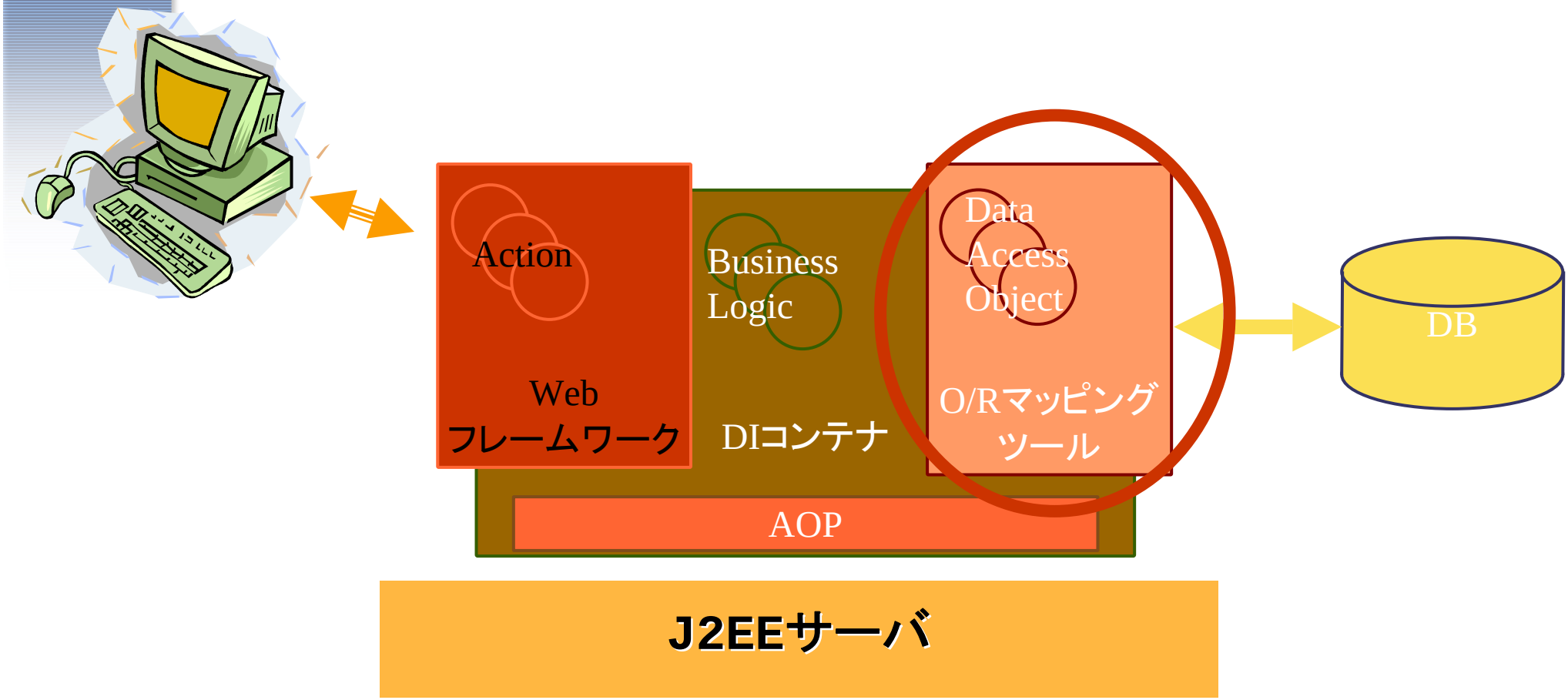
DIのメリット・デメリット

HitachiSoft

- メリット
  - コンポーネント間の依存性が軽減
  - 実装と設定の分離

- デメリット
  - コードを見ただけでは、コンポーネントをまたいだプログラムの動きが追いにくい
    →使う人のスキルを必要とする

O/Rマッピングツール



O/Rマッピングツール

Javaオブジェクト

```
Public class User {
  private long id;
  private String name;
  public void setId(long id) {
    this.id = id;
  }
  public long getId() {
    return id;
  }
  ……
}
```

Mapping

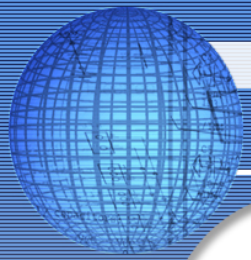
データベース

ID	Name
1	鈴木 一郎
……	……

O/Rマッピングツール

データベーステーブルとJavaオブジェクトを自動的にマッピング
開発者は、Javaオブジェクトに対して操作をする

データベースにアクセスするコードを
一気に削減



Hibernate

HitachiSoft

- Gavin King氏が中心となって開発したJavaのO/Rマッピングツール
- http://www.hibernate.org にて公開
- 最新版は3.0.5 (2005年11月現在)
- XMLを用いてJavaオブジェクトをRDBのテーブルとマップする
- 扱うJavaオブジェクトはPOJOなので、フレームワークへの依存か少ない。
- 高機能な検索言語 HQL
- キャッシュやLazy Loadingなどによるハイパフォーマンス

Hibernate コード例

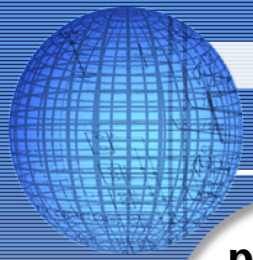
HitachiSoft

```
<?xml version="1.0"?>
<!DOCTYPE hibernate-mapping PUBLIC
  "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
  "http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
 <class
  name="sample.entity.User"
  table="User">
  <id name="id"
      type="long" column="ID">
     <generator class="native"/>
  </id>
  <property name="name"
    type="java.lang.String"
    column="USER_NAME"
    length="10"/>
 </class>
</hibernate-mapping>
```

マッピングファイル

POJO

```
public class User {
  private long id;
  private String name;
  public void setId(long id) {
    this.id = id;
  }
  public long getId() {
    return id;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getName() {
    return name;
  }
}
```



Hibernate コード例

```
public class UserDAOImpl implements UserDAO {
  public List findUser(String name) {
    Session session = null;
    try {
      SessionFactory sf =
        new
   Configuration().configure().buildSessionFactory();
      session = sf.openSession();
      String query = “from User user where user.name = ?”;
      return session.find(query, name, Hibernate.STRING);
    } catch (HibernateException he) {
      return null;
    } finally {
      if (session != null) {
        try {
          session.close();
        } catch (HibernateException he) {}
      }
    }
  }
}
```

Spring + Hibernate

HibernateTemplate

```
public class UserDAOImpl implements UserDAO {
  public List findUser(String name) {
    Session session = null;
    try {
      SessionFactory sf =
        new Configuration().configure().buildSessionFactory();
      session = sf.openSession();
      String query = “from User user where user.name = ?”;
      return session.find(query, name, Hibernate.STRING);
    } catch (HibernateException he) {
      return null;
    } finally {
      if (session
        try {
          session
        } catch (
      }
    }
  }
}
```

Hibernate Only

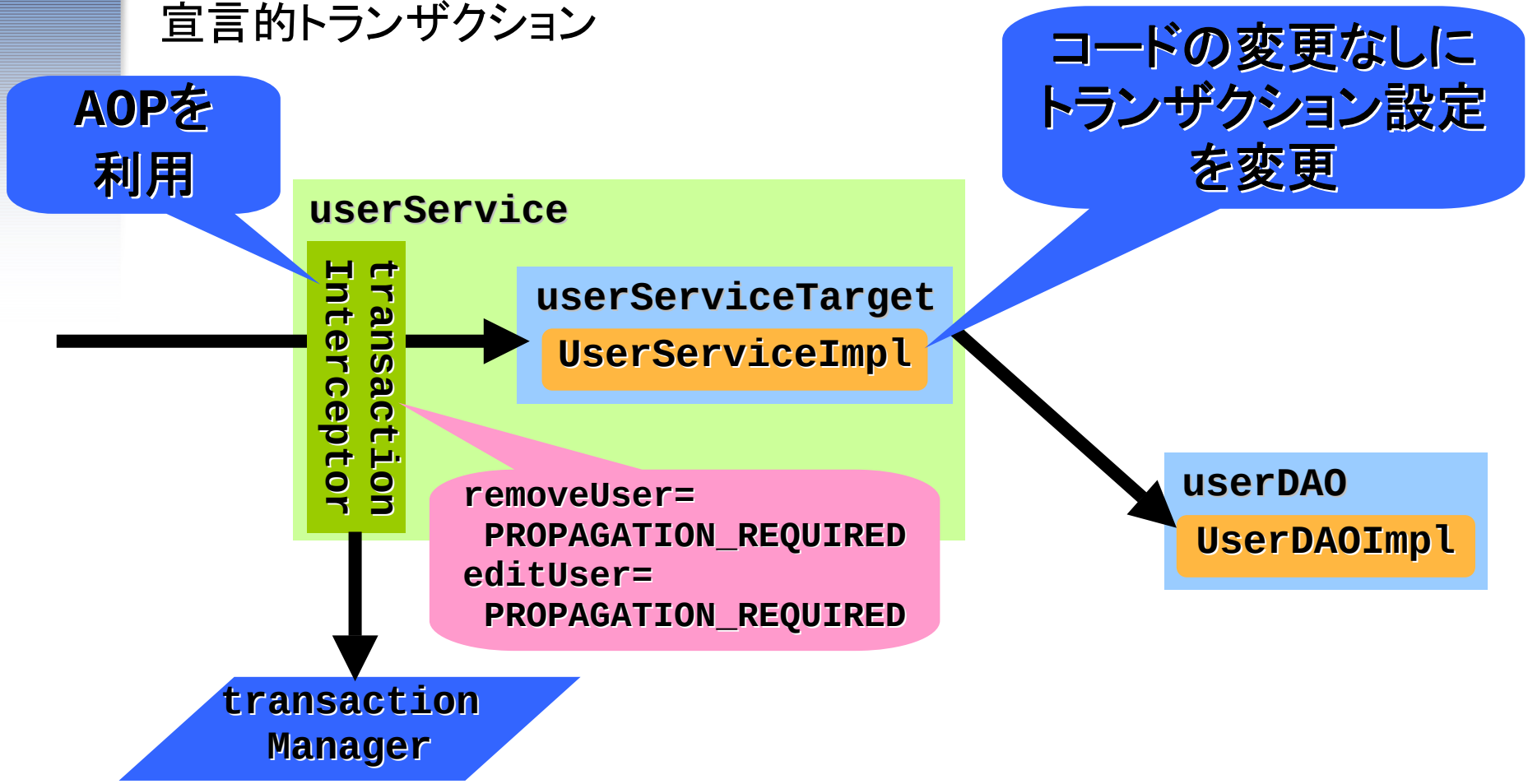
Hibernateを使うコードを一気に簡略化

```
public class UserDAOImpl extends HibernateDaoSupport
                          implements UserDAO {
  public List findUser(String name) {
    String query = “from User user where user.name = ?”;
    return getHibernateTemplate().find(query, name,
                                       Hibernate.String)
  }
}
```

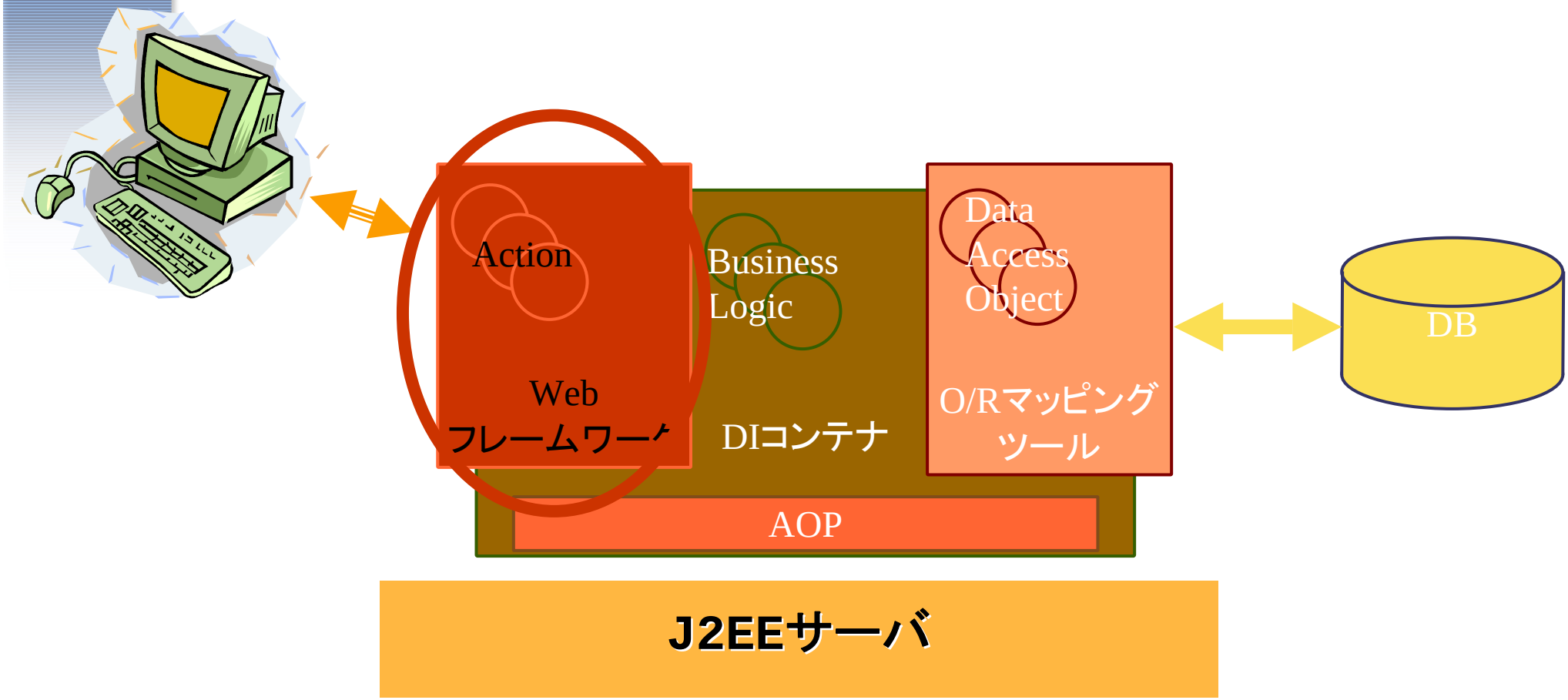
Hibernate + Spring

Spring + Hibernate

宣言的トランザクション



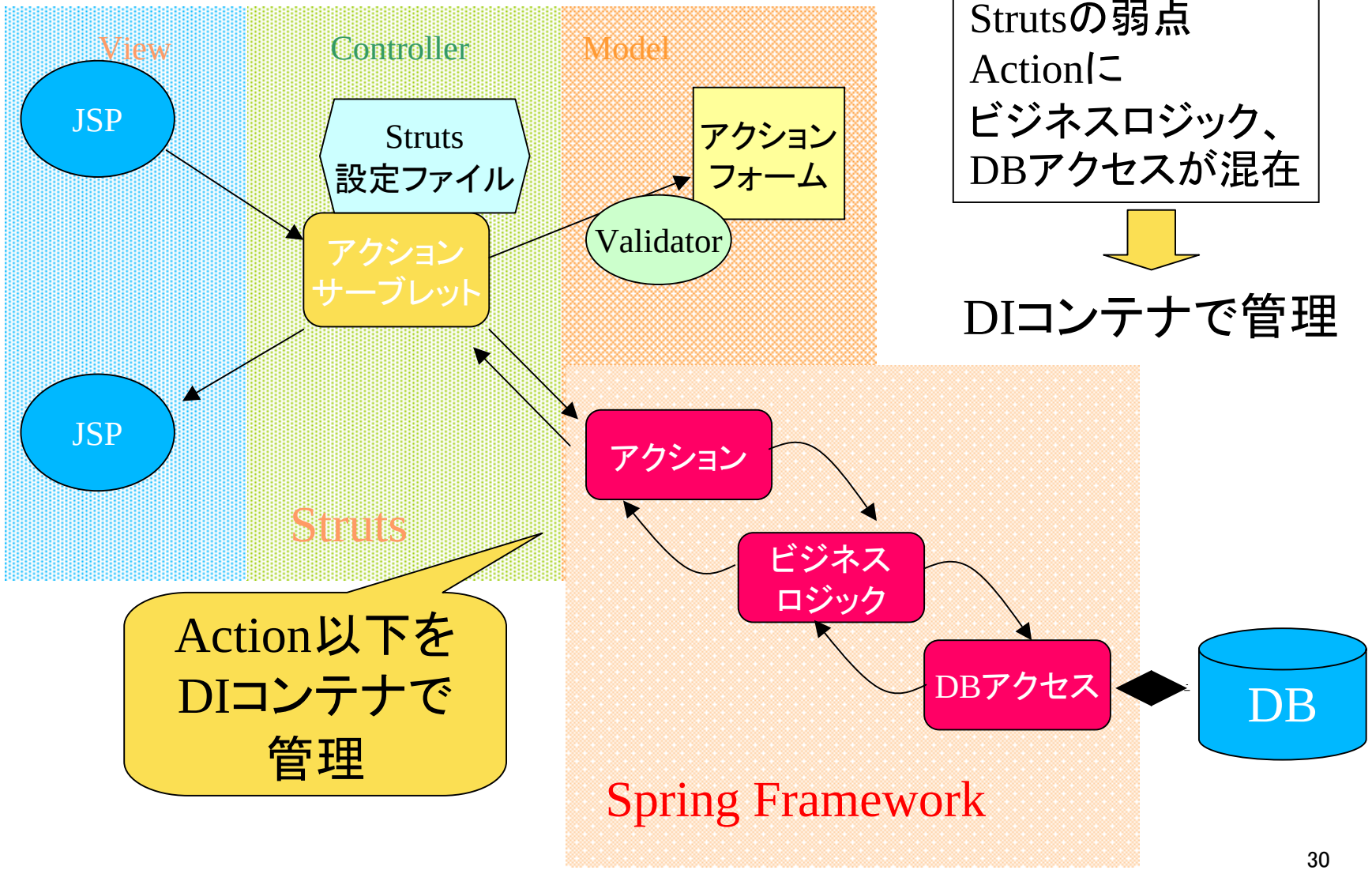
Web フレームワーク



Webフレームワークとの連携

- Web MVC Framework
  - Model-View-Controlデザインパターンをもとに、ページ遷移などのコントロールを行うWebアプリケーション用フレームワーク
  - Struts (struts.apache.org) が有名
  - Java EEの世界では、JSF (JavaServer Faces)が標準
- Spring Frameworkとの連携
  - Struts、JSFとの連携モジュールを提供
  - その他のフレームワークとの連携も進んでいる

StrutsとSpring Frameworkの連携



Strutsの弱点
Actionに
ビジネスロジック、
DBアクセスが混在

DIコンテナで管理

J2EEアプリケーションのテスト

いままでJ2EEアプリケーションの
Unitテストをしていましたか？

コンテナに
デプロイしないと
テストできない

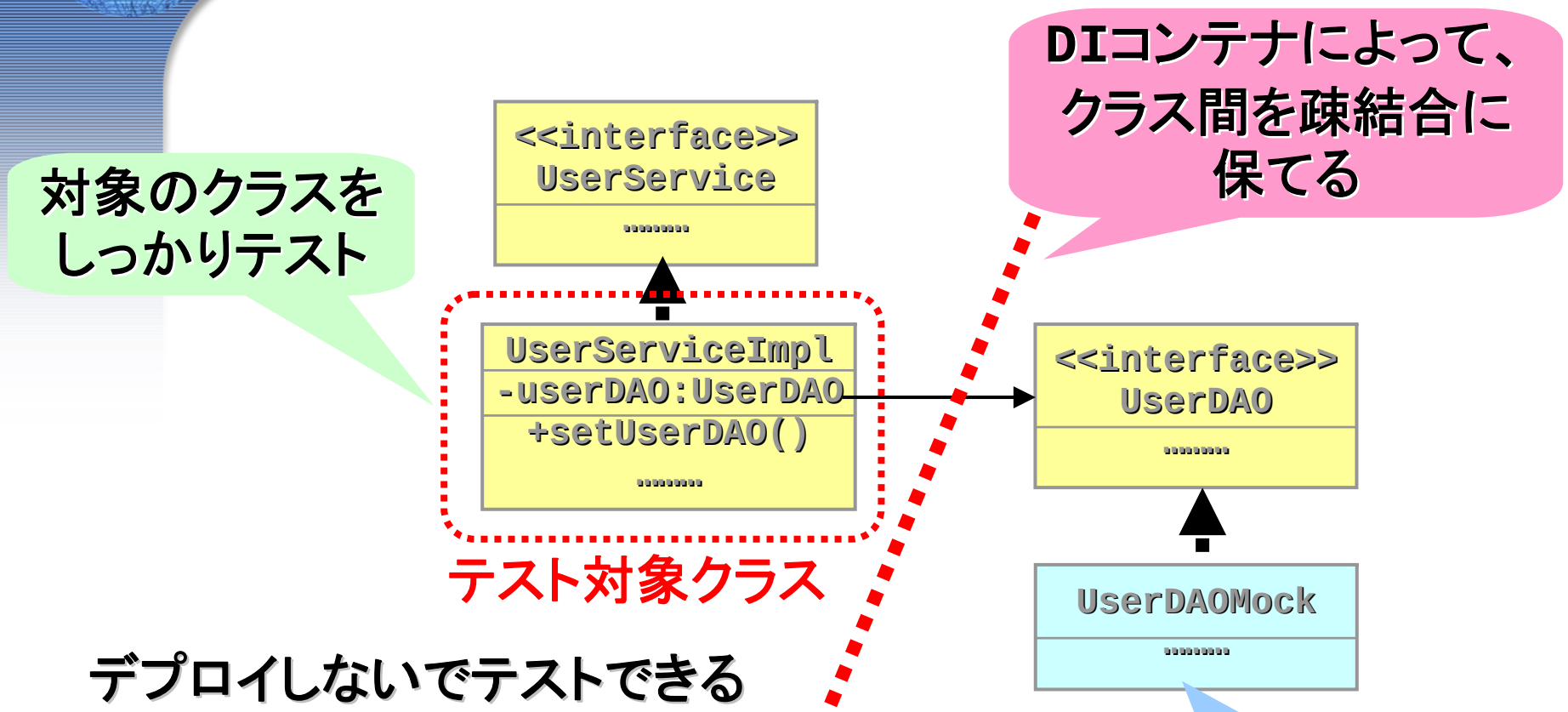
Cactus、
JUnitEE、
難しい...

お互いが依存していて
どこをテストしてるか
わからない

テストに時間がかかる

テストの効率が悪い

テストをしよう！



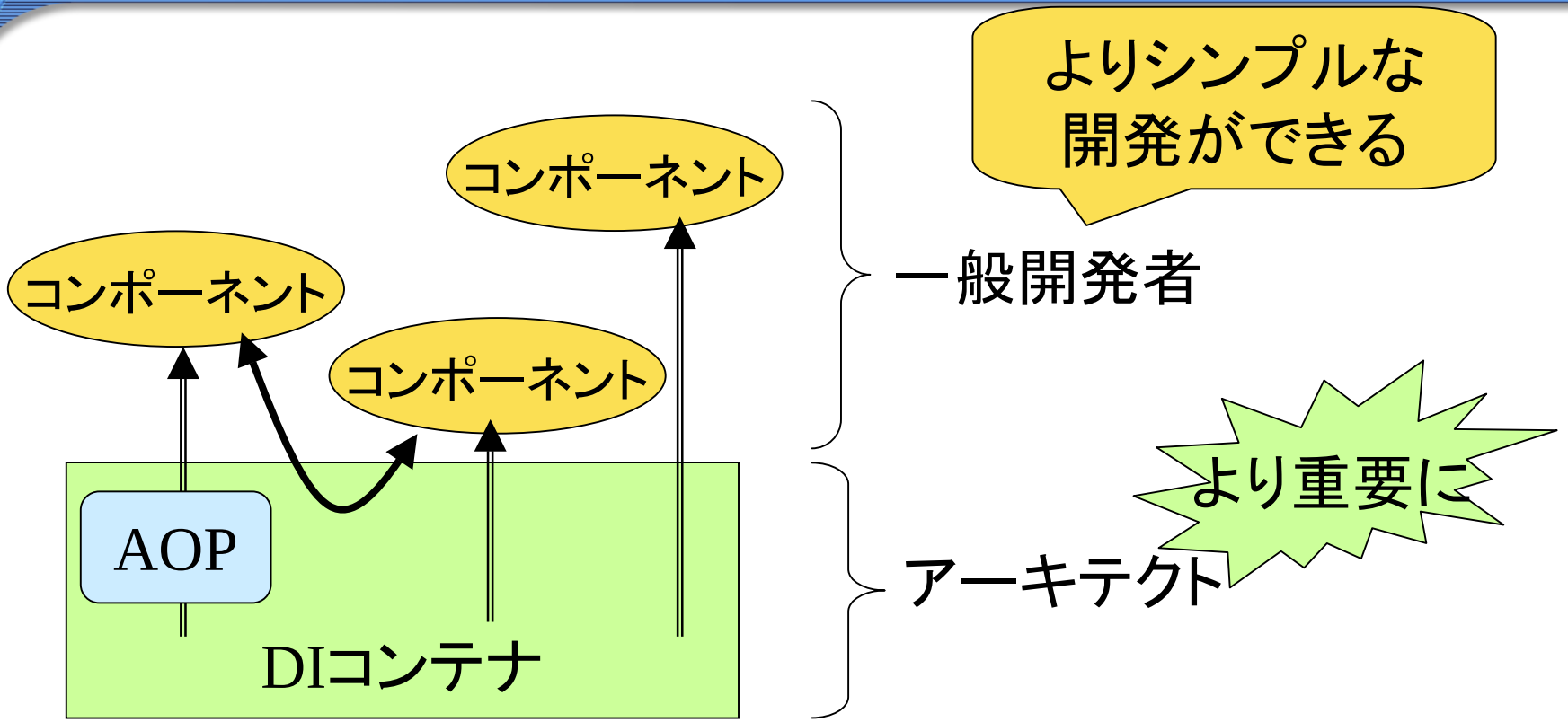
デプロイしないでテストできる
→簡単にテストができる
→J2SEだけでもテストができる
→Eclipseの上でもテストができる
→ビルドプロセスに統合し自動化

テストをしよう！

テストができるようになると

- 繰り返しテストを行う
  - 品質の向上
  - 他の修正によるデグレードを回避
- テストしやすいコードを書く
  - プログラムの可読性の向上

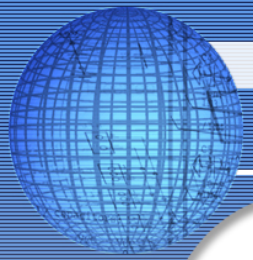
DIコンテナとチームビルディング



DIコンテナ＋AOP

＝

アーキテクトの強力な武器に！



目次

- J2EE 現在までの道のりと問題点
- アプリケーション開発の新しい流れ
  - DIコンテナ
  - O/Rマッピングツール
  - Webフレームワークとの連携
  - Ease of Testing
  - チーム構成
- Java EE5、EJB3.0に向かって

HitachiSoft

EJB 3.0 (JSR220) Public Draft

- Expert Group
  - Spec Lead: Sun Microsystems Linda DeMichel
  - BEA、IBM、Oracleなどに加えて、JBossからGavin Kingなどが参加
- ドキュメントは3部構成
  - EJB 3.0 Simplified API
    EJB開発の簡略化
  - Enterprise JavaBeans Core Contracts and Requirements
    EJBの利用および実装に必要な項目の定義
  - Java Persistence API
    POJO Persistenceを中心とした新しい永続化API
    →このAPI単独で使用可能

HitachiSoft

EJB 3.0の方向

- POJO/POJIによるコンポーネント開発
  - 特定のクラス・インターフェースに依存しない
  - POJOパーシステンス
- EoD
  - Annotationによるメタデータの記述
  - Deployment Descriptorが必須ではない
  - インターフェースの簡略化
- Resource Injectionのサポート
  - リソース参照、EJB参照などをコンテナが注入する
- AOPのサポート

EJB 3.0

- POJO Persistence

```
@Entity
public class User{
 private Long id;
 private String name;
 @Id(generate=AUTO)
 public Long getId() {
  return id;
 }
 public void setId(Long id) {
  this.id = id;
 }
 public String getName() {
  return name;
 }
 public void setName(String name) {
  this.name = name;
 }
}
```

```
@Stateless
public class UserService {
 @PersistenceContext
 EntityManager em;
 public User findUser(Long id) {
  return (User)em.find("User", id);
 }
}
```

EJB 3.0

- DI/AOP (from EJB 3.0 Simplified API)

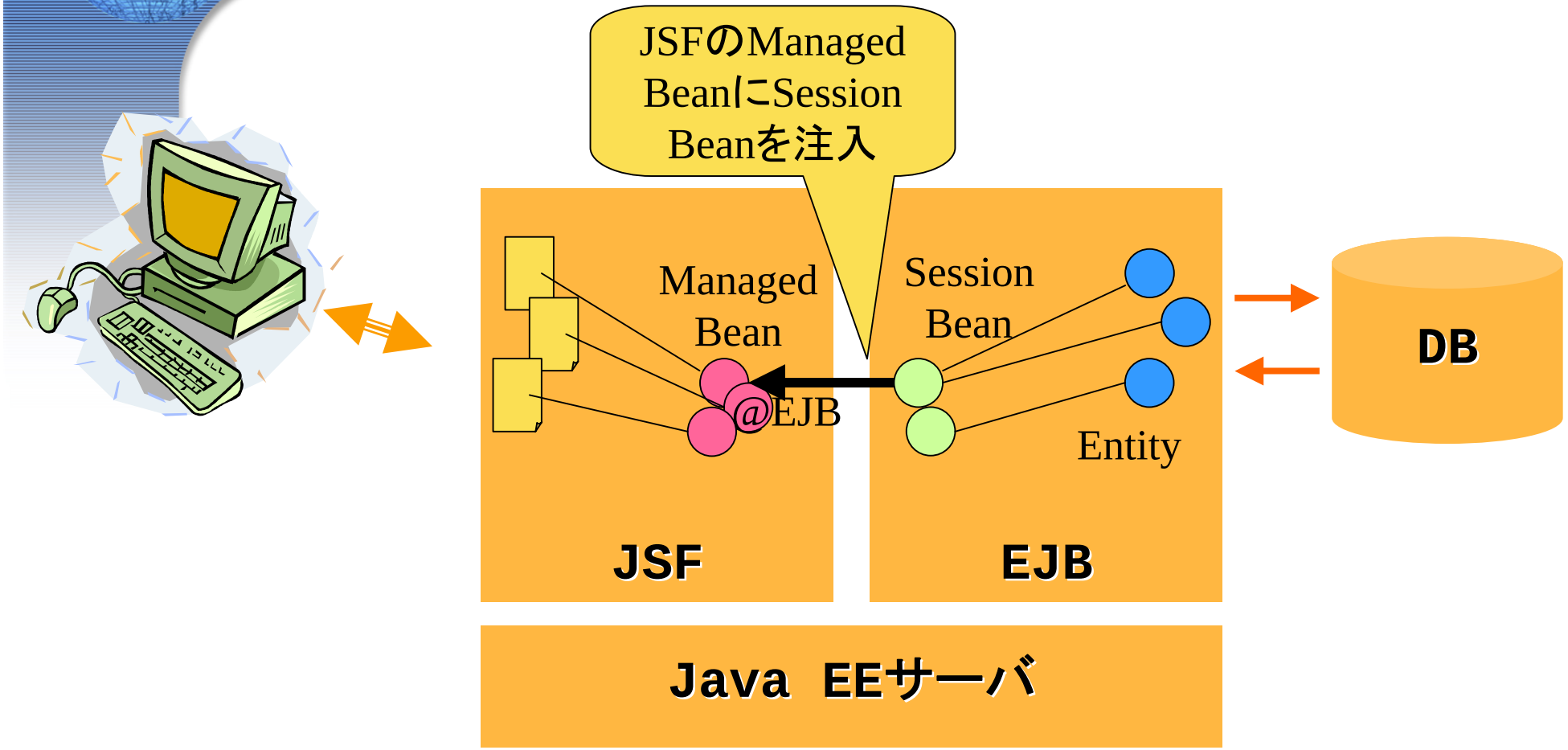
```
@Stateless
public class UserDAO {
 @Resource(name="UserDB")
 public DataSource userDB;
 public void removeUser(int id) {
  try {
   Connection con =
     userDB.getConnection();
   ……
  } catch (Exception ex) {
   ……
  }
}
```

DI

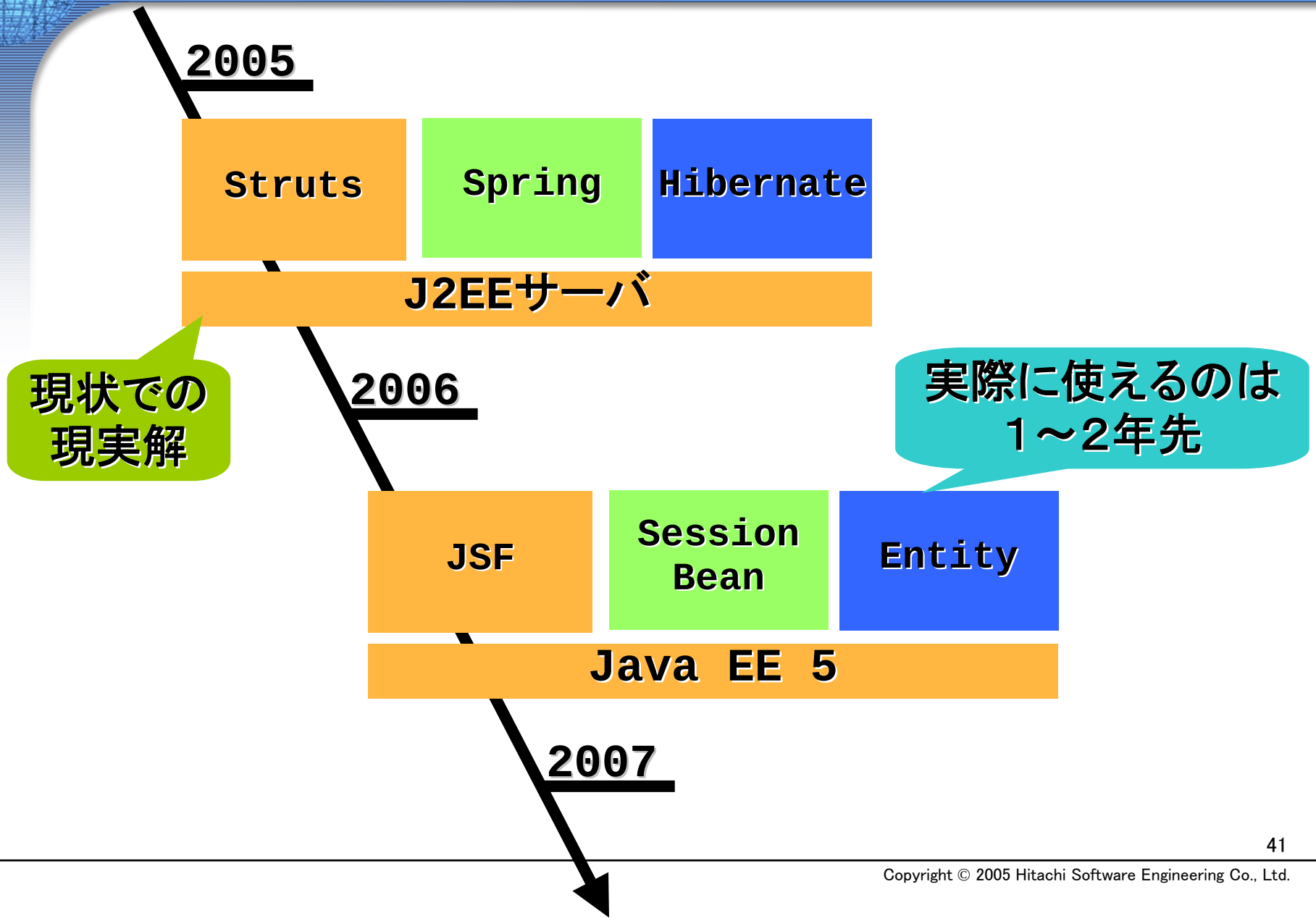
```
@Stateless
@Intercepters({sample.Debug})
public class UserService {
 .....
}

public class Debug
 @AroundInvoke
 public Object debugOut(
  InvocationContext i) {
 System.out.println(i.getMethod()+
                    " is called.");
  return i.proceed();
 }
}
```

AOP

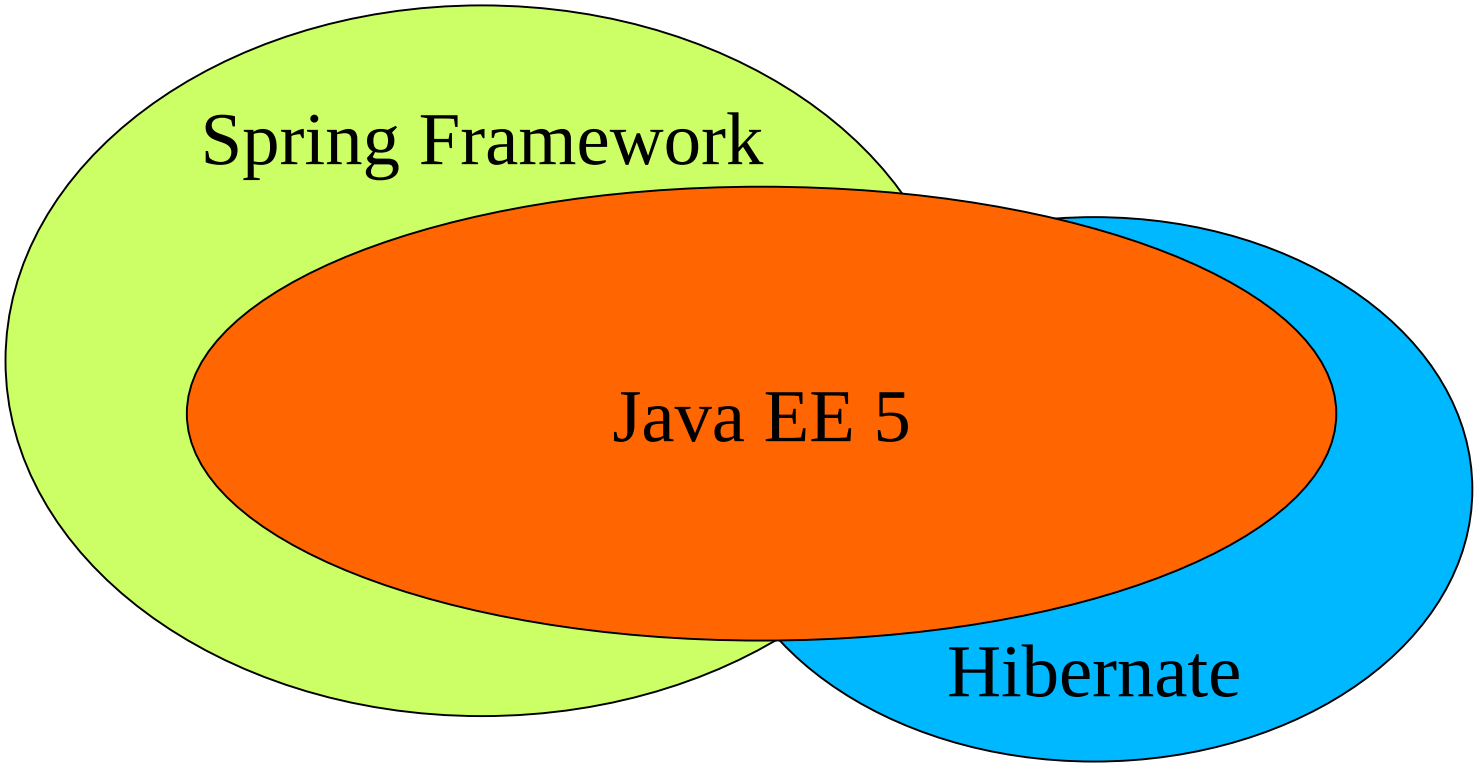
Java EE 5 ～JSFとEJB3.0の連携～



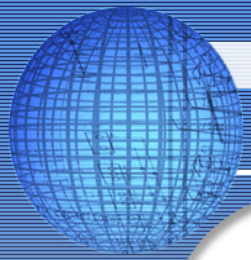
J2EE5.0(EJB3.0)へ向かって



Java EE 5とDIコンテナ



それぞれカバーする範囲が異なる



まとめ

HitachiSoft

- サーバサイドのJavaアプリケーション開発における変化
  - その中心はDIコンテナやO/Rマッピングツールなど POJOを中心に据えたフレームワーク/ツール
- この流れは、次期のJava EE 5にも取り込まれている

せっかくなので

- 実践 Spring Framework
  - 日経BP社
  - 3,990円
- 展示会場の書店にて発売中！

ご質問をどうぞ

Q & A

補足

- 商標
  - JavaおよびEJB、J2EE、J2SE、J2ME、JavaBeans、JDK、HotJava、JDBC、JSPは、Sun Microsystems社の米国およびその他の国における商標です。
  - JBossは、JBoss社の米国およびその他の国における商標です。