



**体感：
O/R マツパー**

**Hibernate
Torque
iBATIS**



設定ファイルを見れば一目瞭然



Hibernate

http://www.hibernate.org/hib_docs/v3/reference/en/html/mapping.html#mapping-declaration



```
<?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 package="eg">
  <class name="Cat" table="cats" discriminator-value="C">
    <id name="id">
      <generator class="native"/>
    </id>
    <discriminator column="subclass" type="character"/>
    <property name="weight"/>
    <property name="birthdate" type="date" not-null="true" update="false"/>
    <property name="color" type="eg.types.ColorUserType"
not-null="true" update="false"/>
    <property name="sex" not-null="true" update="false"/>
    <property name="litterId" column="litterId" update="false"/>

    省略

  </class>
</hibernate-mapping>
```



<class>
<property>
オブジェクト指向



Torque

<http://db.apache.org/torque/releases/torque-3.2/tutorial/step2.html>



```
<!DOCTYPE database SYSTEM "http://db.apache.org/torque/dtd/database_3_2.dtd">  
<database name="bookstore" defaultIdMethod="idbroker">  
  <table name="publisher" description="Publisher Table">  
    <column name="publisher_id" required="true"  
      primaryKey="true" type="INTEGER" description="Publisher Id"/>  
    <column name="name" required="true" type="VARCHAR" size="128"  
      description="Publisher Name"/>  
  </table>  
  
  省略  
  
</database>
```



<database>

<table>

<column>

データベース指向



iBATIC

<http://cvs.apache.org/dist/ibatis/ibatis.java/docs/iBATIC-SqlMaps-2.pdf>



```
<sqlMap id="Product">
  <select id="getProduct" parameterClass=" com.ibatis.example.Product"
    resultClass="com.ibatis.example.Product">
    select
    PRD_ID as id,
    PRD_DESCRIPTION as description
    from PRODUCT
    where PRD_ID = #id#
  </select>
</sqlMap>
```



<sqlmap>
<select>
SQL指向



かなり違います

hibernate : OOA

Torque : DOA

iBATIS : SqlIOA(?)



**体感：
DIコンテナ**

**Seasar2
SpringFramework**



AOPに対する 設定ファイル (ログを出力する)



Seasar2



```
<?xml version="1.0" encoding="Windows-31J"?>
<!DOCTYPE components PUBLIC "-//SEASAR2.1//DTD S2Container//EN"
    "http://www.seasar.org/dtd/components21.dtd">
<components>
  <component name="traceInterceptor"
    class="org.seasar.framework.aop.interceptors.TraceInterceptor"/>
  <component name="hello" class="hoge.HelloImp">
    <property name="helloWord">"Hello"</property>
    <aspect pointcut=".*Hello">traceInterceptor</aspect>
  </component>
</components>
```



**<aspect>あり
だって、
分かりやすいから**



SpringFramework



```
<?xml version="1.0" encoding="Windows-31J"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN/EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
  <bean id="debugInterceptor"
    class="org.springframework.aop.interceptor.DebugInterceptor"/>
  <bean id="debugAdvisor"
    class="org.springframework.aop.support.RegexpMethodPointcutAdvisor">
    <property name="advice"><ref bean="debugInterceptor"/></property>
    <property name="patterns"><value>.*Hello</value></property>
  </bean>
  <bean id="helloTarget" class="hoge.HelloImp">
    <property name="helloWord" value="Hello"/>
  </bean>
  <bean id="hello"
    class="org.springframework.aop.framework.ProxyFactoryBean">
    <property name="target"><ref local="helloTarget"/></property>
    <property name="interceptorNames">
      <list>
        <value>debugAdvisor</value>
      </list>
    </property>
    <property name="proxyInterfaces">
      <value>hoge.Hello</value>
    </property>
  </bean>
</beans>
```



**<aspect>なし
だって、
DIコンテナの機能
ではないから**



補足

Seasar2は、開発者を楽にするという明確な思想があります。ですから、積極的にSesarファミリーとして同じ思想のコンポーネントを提供しています。SesarはDIコンテナというよりも“DIコンテナを中心にしたWebアプリケーションフレームワーク”と捉えるべきでしょう。

SpringFrameworkは、無思想という思想と呼べるものです。統合される各コンポーネントの思想を守るために、Spring自身はなるべく汎用的な機能しか提供しません。ですから、AOPについても、AOP機構そのものを交換することが可能です。Springはまさに“DIコンテナ”であろうとしています。

よく見ていくと、Seasar2とSpringFrameworkは、そもそも違うカテゴリのプロダクトなのかもしてません。