=http://www.bea.com/eWorld/><deploySOA> DEPLOY SOA. NOW. </deploySOA></soap:Bod

Dynamic Aspect-Oriented Programming (AOP): SOA for the Application

Jonas Bonér
Senior Software Engineer
Alexandre Vasseur
Senior Software Engineer

BEA eWorld 2004

Session Goal

What will you learn?

Learn how to apply AOP in J2EE™ projects.

Learn why AOP is SOA for the application.

Speaker's Qualifications

Jonas Bonér

- Senior Software Engineer at JRockit™ Group, BEA Systems
- Founder of the AspectWerkz AOP framework
- Speaker at JavaPolis 2003, eWorld 2004, AOSD 2004,
  JavaOne 2004, BEA User Group 2004

Speaker’s Qualifications

Alexandre Vasseur

- Software Engineer at JRockit™ Group, BEA Systems
- Co-founder of the AspectWerkz AOP framework
- Speaker at eWorld 2004, AOSD 2004,
  JavaOne 2004, BEA User Group 2004

Agenda

Dynamic AOP in action

AOP crash course

Add-on aspects

Architectural aspects

AOP container integration in J2EE™

Agenda

Dynamic AOP in action

AOP crash course

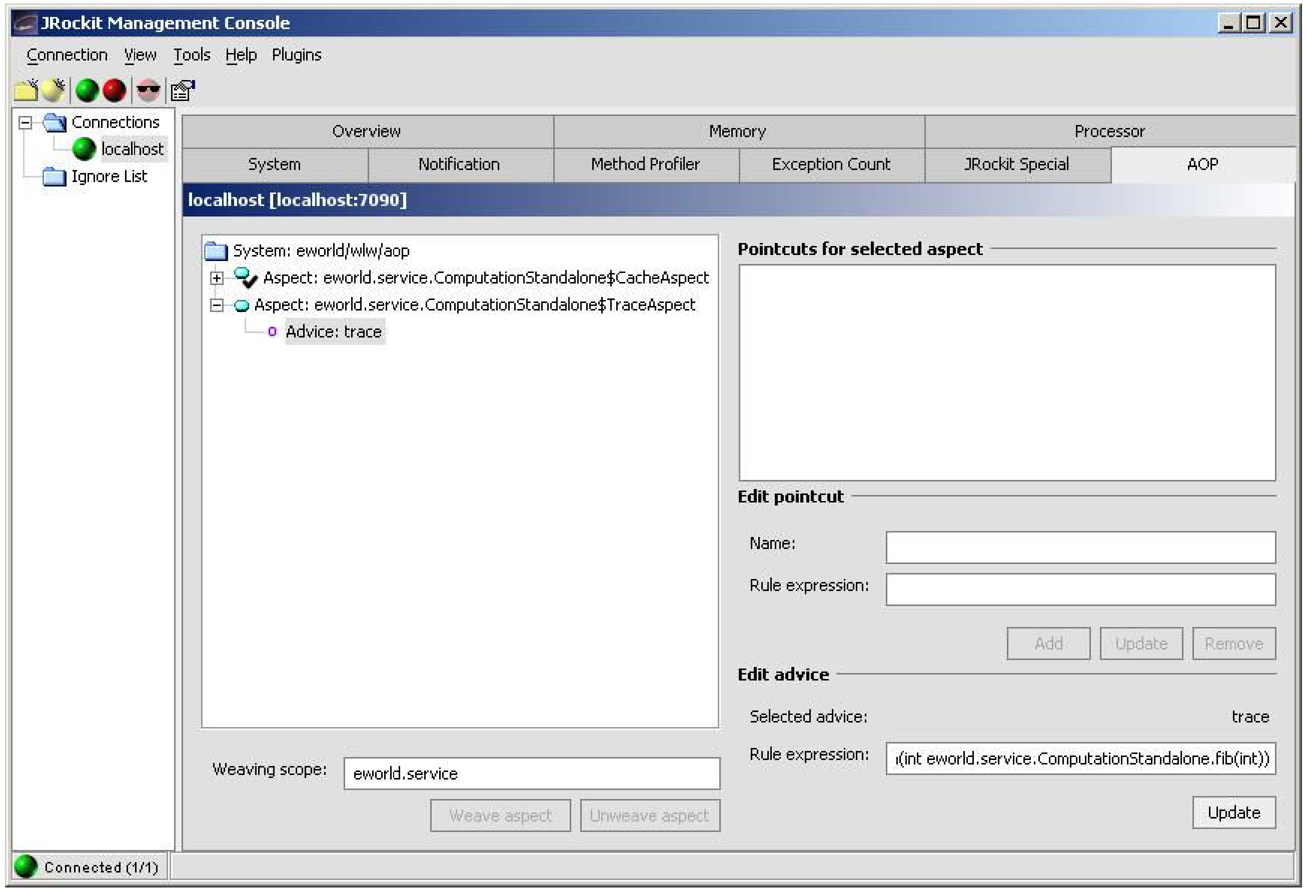
Add-on aspects

Architectural aspects

AOP container integration in J2EE™

Demo

- Manage your Aspects at runtime through the JRockit Management Console



System: eworld/wlw/aop
Aspect: eworld.service.ComputationStandalone$CacheAspect
Aspect: eworld.service.ComputationStandalone$TraceAspect
Advice: trace

Name:
Rule expression:

Selected advice: trace
Rule expression: ı(int eworld.service.ComputationStandalone.fib(int))

Weaving scope: eworld.service

AOP and J2EE in Action

Looks promising but…

…is AOP only for tracing and caching in J2EE?

- Demo shows only a tiny fraction of what AOP in J2EE™ can do for you!
- What are the requirements for AOP in J2EE™ environments?
- Is there a common theory behind AOP?
- Do I need to learn a new language and install new tools?
- What is this “Aspect”? Just a Java class?!
- How do I integrate AOP in my BEA WebLogic Platform environment and development process?

Agenda

Dynamic AOP in action

AOP crash course

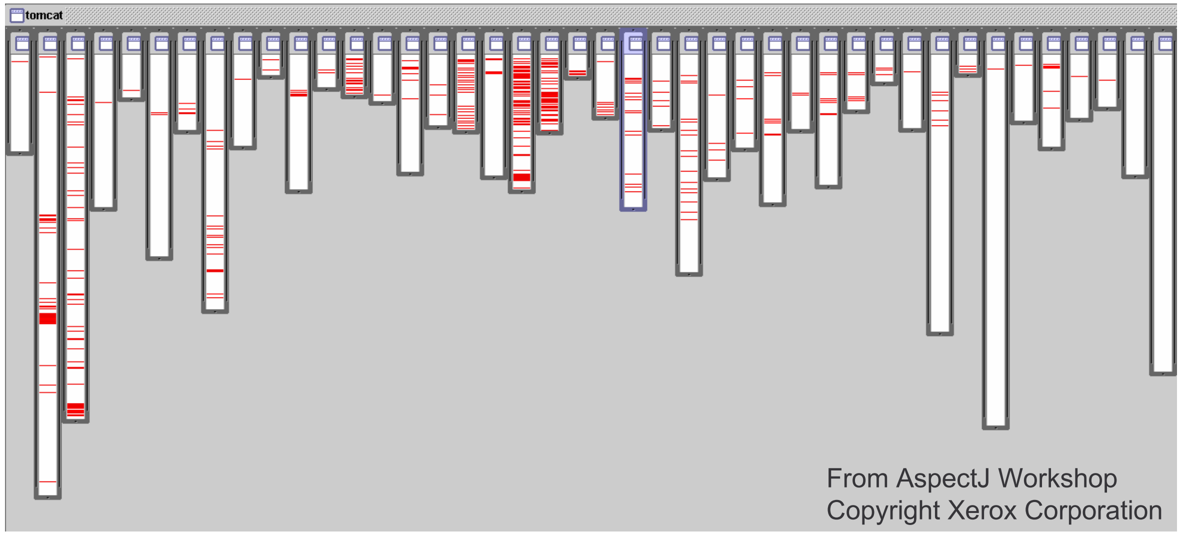
Add-on aspects

Architectural aspects

AOP container integration in J2EE™

AOP crash course

- OOP fails to address 'cross-cutting concerns'
  - Introduces 'code tangling' and 'code scattering'
  - Makes software harder to write, understand, reuse and maintain



From AspectJ Workshop
Copyright Xerox Corporation

- AOP enables 'Separation Of Concerns'
- The Aspect modularizes a crosscutting concern

AOP crash course

Core concepts

1. Define well-defined points in the program flow
   - Join points
2. Pick out these points
   - Pointcuts
3. Influence the behaviour at these points
   - Advice, Introductions
4. Weave everything together in a functional system
   - Weaver

AOP crash course (with AspectWerkz)

AspectWerkz AOP code sample

```
public class CacheAspect {

    //...utility methods etc.

    /** @Expression execution(Integer Computation.exec(Integer)) */
    Pointcut toCache;

    /** @Around toCache */
    public Object cache(JoinPoint jp) {
        MethodRtti rtti = (MethodRtti)jp.getRtti();
        Integer parameter = (Integer)rtti.getParameterValues()[0];
        // if (inCache) return fromCache
        // else
        Object result = jp.proceed();
        // put result in Cache
        return result;
    }
}
```

Aspect is a Java class

Pointcuts are fields

Advice binding

Advice are methods

`proceed()` invokes the next advice or the target join point (method, field …), – only for Around advice

AOP crash course (with AspectWerkz)

Aspect XML deployment descriptor

Aspect class name

```
<aspectwerkz>
  <system id="computation">
    <aspect class="aspect.CacheAspect"/>
    <!-- other aspects -->
  </system>
</aspectwerkz>
```

Aspect reuse / refine:
new XML defined pointcut
and advice binding

```
<aspectwerkz>
  <system id="computation">
    <aspect class="aspect.CacheAspect">
      <pointcut name="toCache2" expression="..."/>
      <advice type="around" name="cache"
              bind-to="toCache2 OR ..."/>
    </aspect>
  </system>
</aspectwerkz>
```

AOP crash course (with AspectWerkz)

AOP Annotations

- Pointcuts as Annotations
  - **@Expression**

```
execution(* package.Class.method(p1, *))
call(* package..Interface+.callee(..))
get(fieldType package.Superclass+.field)
set(* package.Class.field)
handler(package.Exceptionclass+)
within(package.Class)
withincode(* package..Interface+.caller(..))
```

- Pointcut expression language
  - Regular expression based

AOP crash course (with AspectWerkz)

AOP Annotations

- Pointcuts are composable
  - `OR, AND, NOT, cflow(PCD)`
- Pointcuts are named or anonymous
  - `@Before namedPointcut OR execution(...)`
  - `@Around ...`
  - `@After  ...`
- Three different types of advice

AOP crash course

Main points

- AOP brings a new theory to address cross-cutting concerns
  - Join points and Pointcuts
  - Advice, Aspects and Introductions

- Different weaving schemes
  - Offline weaving (post-processing)
  - Load time weaving
  - Runtime weaving
  - Application server specific weaving

- AspectWerkz is a Java 1.5 ready solution
  - Aspects are Annotated Java™ classes
  - Activated or refined through a simple XML DD

Agenda

Dynamic AOP in action

AOP crash course

Add-on aspects

Architectural aspects

AOP container integration in J2EE™

Add-on Aspects

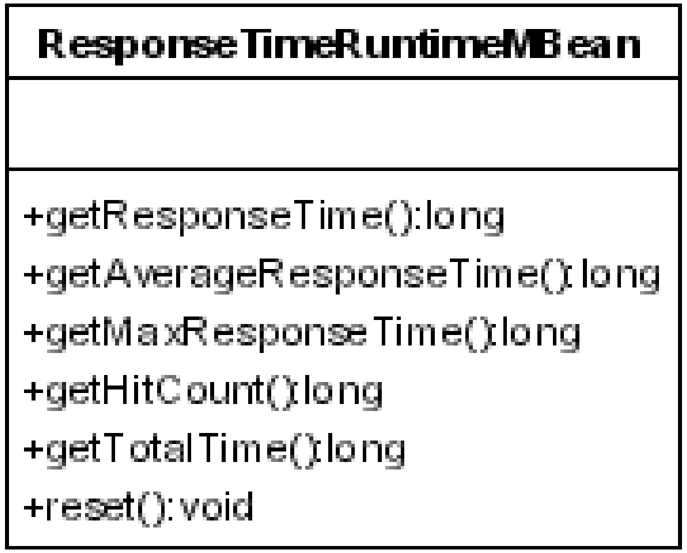
System level aspects

- Aspects deployed at Application Server level
  - Can be provided by Application Server vendor
  - Can be bundled as extensibility libraries
- Highly reusable
- Loosely coupled to functional requirements
  - General purpose tracing
  - Performance diagnostic
  - Arbitrary LRU cache
  - Asynchronous call, thread pool based or JMS queue based

Add-on Aspects

Performance reporting Aspect with JMX

- Aspect bundles with the Application Server
- The around advice can be bound to arbitrary methods or constructors, plugged and unplugged at runtime
- Exposes runtime performance to JMX



```
ResponseTimeRuntimeMBean

+getResponseTime():long
+getAverageResponseTime():long
+getMaxResponseTime():long
+getHitCount():long
+getTotalTime():long
+reset():void
```

Add-on Aspects

The Aspect itself addresses only the crosscutting

- Aspect is a glue, you capitalize on your J2EE assets

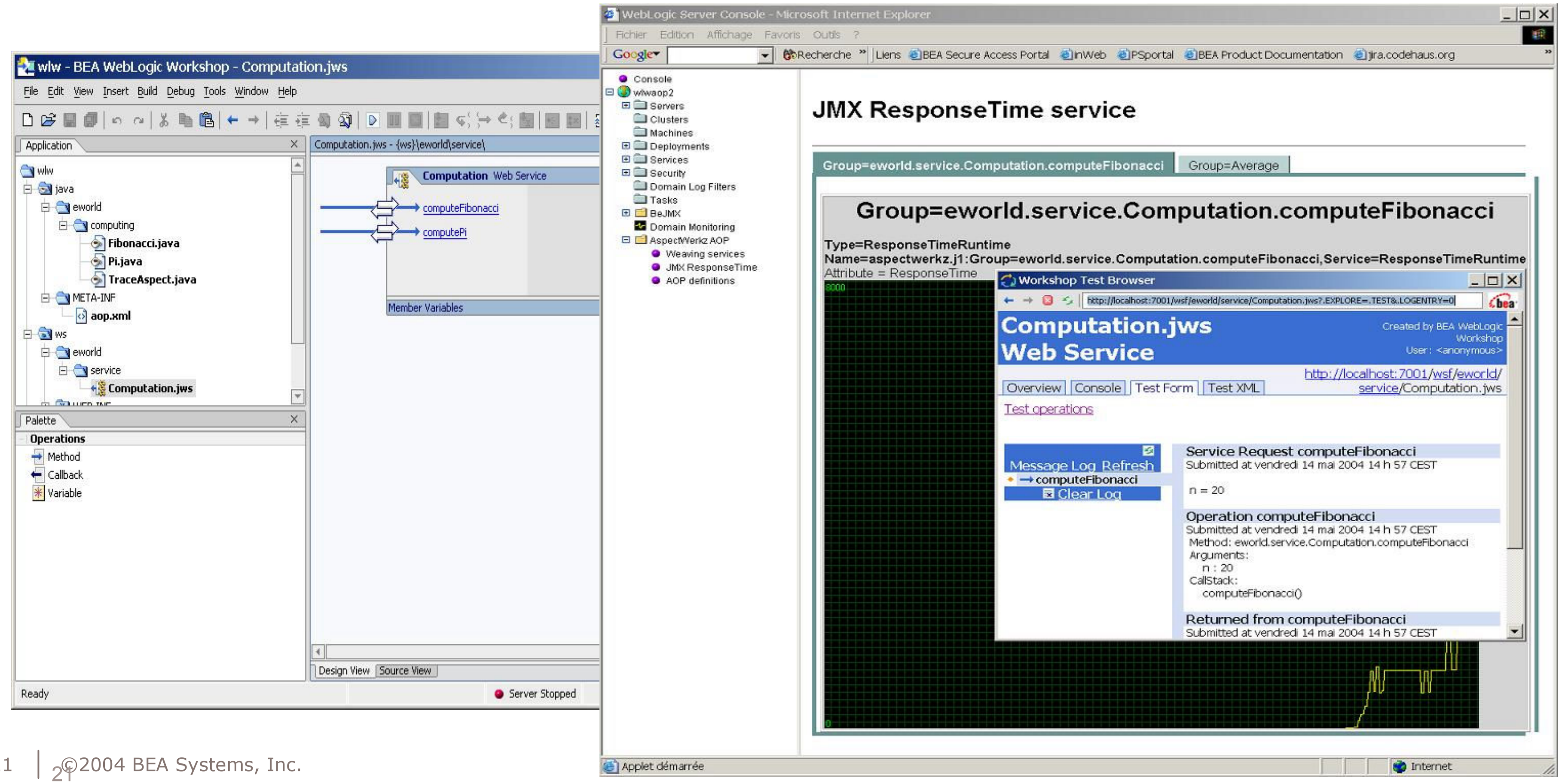
Actual implementation is more complex

```
public class ResponseTimeAspect {

    ResponseTimeRuntimeMBean m_mbean;

    // pointcuts will be added at runtime
    // some generic pointcuts could be defined for J2EE components

    /** @Around toMonitor */
    public Object monitor(JoinPoint jp) {
        long ts = System.currentTimeMillis();
        Object result = jp.proceed();
        m_mbean.update(System.currentTimeMillis() - ts);
        return result;
    }
}
```

Binding is optional since we we will declare pointcuts at runtime

Demo

- JMX Monitoring Aspect hot-deployed in BEA WebLogic™ Platform



JMX ResponseTime service

Group=eworld.service.Computation.computeFibonacci

Type=ResponseTimeRuntime
Name=aspectwerkz.j1:Group=eworld.service.Computation.computeFibonacci,Service=ResponseTimeRuntime
Attribute = ResponseTime

http://localhost:7001/wsf/eworld/service/Computation.jws?.EXPLORE=.TEST&.LOGENTRY=0

Computation.jws
Web Service

Created by BEA WebLogic Workshop
User: <anonymous>

http://localhost:7001/wsf/eworld/service/Computation.jws

```
Service Request computeFibonacci
Submitted at vendredi 14 mai 2004 14 h 57 CEST

n = 20

Operation computeFibonacci
Submitted at vendredi 14 mai 2004 14 h 57 CEST
Method: eworld.service.Computation.computeFibonacci
Arguments:
   n : 20
CallStack:
   computeFibonacci()

Returned from computeFibonacci
Submitted at vendredi 14 mai 2004 14 h 57 CEST
```

Agenda

Dynamic AOP in action

AOP crash course

Add-on aspects

Architectural aspects

AOP container integration in J2EE™

Architectural Aspects

Aspect-Oriented Analysis and Design

- AOP is not only for add-on and patch-like behaviour
- Should be a part of analysis and design
- Design core cross-cutting behaviour as Aspects
- Can be refactored out (when working with an existing application)
- Deployed within the application and defined in

  `META-INF/aop.xml`

Architectural Aspects

Examples from the J2EE world

- Business Rules
- Design Patterns
- Role-Based Security
- Transaction Demarcation
- Persistence
- Synchronization
- Messaging
- Monitoring
- Much more…

Architectural Aspects

Example: Address book web application

- Requirements
  - Login and logout
  - List user’s contacts
  - Add a contact
  - Remove one or more contacts

- Services
  - Authentication
  - Authorization
  - Persistence of the address books
  - Transaction integrity

Architectural Aspects

The OO implementation is not modularized

- Security is a cross-cutting concern
- In an regular OO based implementation; the concern is scattered all over the code base:
  - Authentication code at all service methods
  - Authorization code at all critical business methods
- A **`ServletFilter`** could only implement authentication and URL based authorization, and would be *web specific*

Architectural Aspects

Enter Aspect-Oriented Programming

- AOP makes it possible to capture the concern in one single modular unit (the Aspect)
- Security Aspect that implements
  - Authentication
  - Authorization (on method or even field level)
  - Role-Based using JAAS (pluggable)
- Use of abstraction makes the Aspect reusable

Architectural Aspects

Authentication advice

```
/** @Around authenticationPoints */
 public Object authenticateUser(JoinPoint joinPoint) throws Throwable {
     if (m_subject == null) {
        // no subject => authentication required
        Context ctx = ... // principals and credentials
        m_subject = m_securityManager.authenticate(ctx);
     }
     Object result = Subject.doAsPrivileged(
        m_subject, new PrivilegedExceptionAction() {
           public Object run() throws Exception {
              return joinPoint.proceed();
           };
        }, null
     );
     return result;
   }
```

Architectural Aspects

Authorization advice

```
/** @Around authorizationPoints */
public Object authorizeUser(JoinPoint joinPoint) throws Throwable {
   MethodRtti rtti = (MethodRtti)joinPoint.getRtti();
   if (m_securityManager.checkPermission(
      m_subject,
      rtti.getTargetClass(),
      rtti.getMethod())) {
      // user is authorized => proceed
      return joinPoint.proceed();
   }
   else {
      throw new SecurityException(...);
   }
}
```

Architectural Aspects

Integration in the web application

- Extend the **`AbstractRoleBasedAccessProtocol`** aspect and define the pointcuts:
  - **`authenticationPoints`**
  - **`authorizationPoints`**

- Authenticate the user at service methods:
  - **`* ServiceManager.*(..)`**

- Authorize on methods that modifies and accesses the **`AddressBook`**:
  - **`* AddressBookManager+.*(..)`**

Architectural Aspects

Define the pointcuts in the META-INF/aop.xml

```
<aspect class="security.RoleBasedAccessProtocol">

   <pointcut name="authenticationPoints"
      expression="execution(* ServiceManager.get*(..))"/>

   <pointcut name="authorizationPoints"
      expression="execution(* AddressBookManager+.*(..))"/>

</aspect>
```

Demo

- Role-Based Security for web applications

Architectural Aspects

Reusable Aspect libraries

- AWare (http://docs.codehaus.org/display/AWARE)
  - Aspect repository with reusable aspects for J2EE
  - AspectWerkz based
- aTrack (https://atrack.dev.java.net/)
  - Best practices on how to use AOP in J2EE application environments
  - Library with both add-on and architectural aspects
  - AspectJ based

Agenda

Dynamic AOP in action

AOP crash course

Add-on aspects

Architectural aspects

AOP container integration in J2EE™

Aspect Container integration in J2EE™

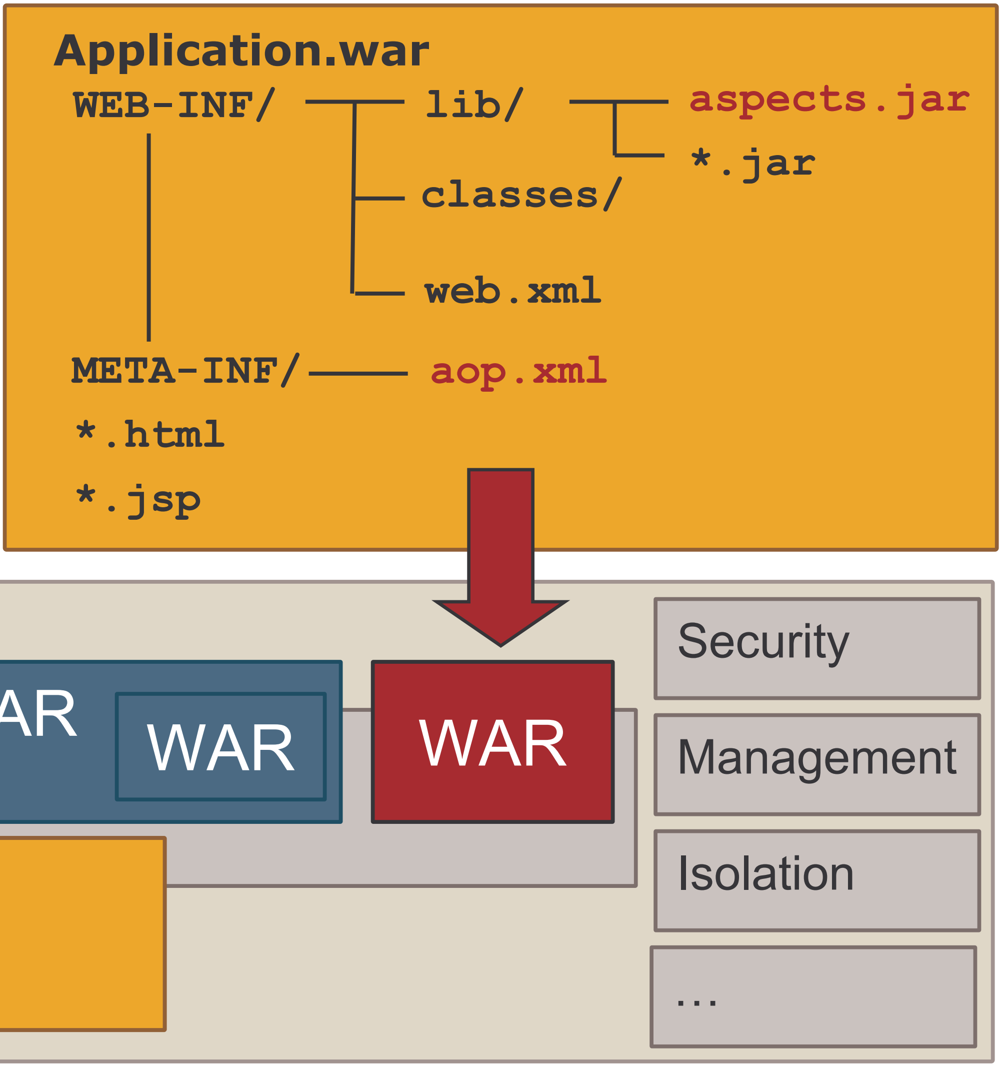
Second generation is integrated

- AspectWerkz
  - Cross server integration, Java 1.3+
  - Aspects are plain Java classes with Annotations (Tiger)
  - Provides cross-platform AOP container
  - Runtime weaving support through BEA JRockit
  - Seamless integration in BEA WebLogic Server

Aspect Container integration in J2EE™

AspectWerkz's Aspect Container

- Define the `META-INF/aop.xml`
- Package aspects
- Deploy as usual



AOP: SOA for the Application

- AspectWerkz AOP framework can be used today in your BEA WebLogic environment
  - Flattens out the learning curve
  - Seamless integration and value added tool along BEA JRockit
  - Lots of added value, almost plug'n play
- Add-on Aspects and Architectural Aspects capture different cross-cutting concerns at different J2EE™ levels
  - System Aspects at Application Server level
  - Aspect deployed alongside applications with XML DD
  - Application services layer thru dynamic AOP
- J2EE™ integration of AOP is **`META-INF/aop.xml`**
  - Made seamless and cross-platform with AspectWerkz

For More Information

- http://dev2dev.bea.com/products/wljrockit81/index.jsp
- http://aspectwerkz.codehaus.org
- http://docs.codehaus.org/display/AWARE/
- http://aosd.net
- http://blogs.codehaus.org/people/jboner
- http://blogs.codehaus.org/people/avasseur

Q&A

Questions?

=http://www.bea.com/eWorld/><deploySOA> DEPLOY SOA. NOW. </deploySOA></soap:Bod

Dynamic Aspect-Oriented Programming (AOP): SOA for the Application

Jonas Bonér
Senior Software Engineer
Alexandre Vasseur
Senior Software Engineer

BEA eWorld 2004