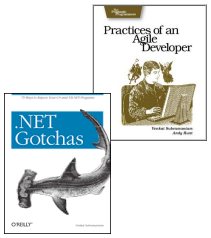
Annotation Hammer

spkr.name = 'Venkat Subramaniam'
spkr.company = 'Agile Developer, Inc.'
spkr.credentials = %w{Programmer Trainer Author}
spkr.blog = 'agiledeveloper.com/blog'
spkr.email = 'venkats@agiledeveloper.com'



Abstract

- Annotation is an interesting feature in Java. However, like any features, there are good uses and bad uses. When should you use Annotation? This presentation will answer that question for you.
- In this presentation we will take a closer look at annotation. We will see how to write them, how to use them. Then we will take a look at examples of annotation in various Java applications/frameworks. We will discuss examples of good use and not so good use. We will then lay out some good practices to follow.

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Inheritance as a Vehicle?

- How do you say that an instance of your class is Serializable?

```
public class MyClass implements java.io.Serializable
{
}
```

- How many methods does Serializable have?
- We've seen similar approach for Cloneable,...
- We've seen inheritance used when we want to convey an intent

Inheritance Hammer

- Inheritance in cases like the example is not to derive any behavior or express a specific contract
- It was to provide a consent that it's OK to serialize
- Code bloat
- Intent is not clear
- Lacks Granularity (how to say a field is not serializable?)

Invent Keywords?

```
public class MyClass implements java.io.Serializable
{
    private int val1;
    private transient int val2;
}
```

- Now we needed not only a dummy interface (Serializable), but also a keyword (transient)

Moving into Your Domain

- What if I am creating a framework that manages threading needs
- I need to specify if my object is thread safe or not

```
public class MyClass implements java.io.Serializable, VenkatsFramework.ThreadSafe
{
    private int val1;
    private transient int val2;
}
```


Is not Granular

- Now, what if not all method of a class are thread safe
- How do I say that only select methods are thread safe?

```
public class MyClass implements java.io.Serializable, VenkatsFramework.ThreadSafe
{
    private int val1;
    private transient int val2;

    public IsThreadSafe void foo() // Not valid Java Code
    {

    }
}
```

How about XML Config

- When we can't do it in code, why not outside, using XML?

```
<IsThreadSafe class="MyClass" method="foo" />
...
```


XML Hammer

- Not elegant
- How do you deal with method overloading
  - overload XML!
- Clutter
- Separated from code
- Hard to follow
- Hard to keep up with
- Leads to XML Hell

Metadata

- What's the concern?
- Classes, inheritance, ... help us great deal with abstraction
- Here, we're dealing with metadata
- We need something more powerful to express it

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Annotation

- Allows us to extend the language with new metadata
- Helps us "color" the code
- Great expressive power
- Elegant
- Granular—can be expressed on different constructs
- Can be accessed programatically

Using Annotations

- Used like modifiers
- Convention to place it before the other modifiers like public, static, etc.
- Has parenthesized list of name value pair
- For marker annotation, no parenthesis
- For single valued annotation, no need for the "elementName ="—simply provide the value
  - Annotation defines a `value` property in this case

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Java Annotation

- Java language has a few annotations built-in

Deprecated

- How do you deprecate methods?
- By using @deprecated in the comment part
- Not very elegant, lacks intent
- May not be supported by some tools as not a standard way to express
- Java 5 has annotation to deprecate methods
  - Sadly does not offer a deprecation reason facility
  - So you would most likely mix with old way

@Deprecated

```
/**
 * @deprecated Please use increment instead...
 */
@Deprecated public void inc()
{

}
```

Avoiding Accidental Override

- Methods not expressed as final are overridable
- What if you wrote a method but did not realize you're overriding a method in the base?
- What if method was added to base later?
- Would it be nice to express explicitly that you're overriding a method?
- Methods that actually don't override will result in warning
  - Does not still help fully—backward compatibility...

@Override

```
public class MyBase
{
    public void method1()
    {

    }
}
```

```
public class MyClass extends MyBase
{
    @Override public void method1()
    {

    }

    @Override public void method2()
    Method does not override method from its superclass

    }
}
```

Suppressing Warnings

- Generally I recommend that you treat warnings as errors
- Sometimes you may want to quite a nagging warning that's not fully under your control
- @SuppressWarning tells compiler to suppress a particular warning
  - unchecked or deprecated

@SuppressWarning

```
public static void Main(String[] args)
{
    MyClass obj = new MyClass();

    obj.inc();
}
```

'inc()' is deprecated

```
@SuppressWarnings("deprecation")
public static void Main(String[] args)
{
    MyClass obj = new MyClass();

    obj.inc();
}
```

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Writing an Annotation

- Syntax to write an annotation is weird
- You use @interface to define an annotation
- Each method declaration declares an element
  - No parameters, no exceptions, no implementation for these methods
  - Return type must be primitive, String, Class, enum, annotation, or array of these
  - Can have default values
- Marker annotation have no methods
- If you single element, name it `value`

Writing Annotation

```
package VenkatsFramework;

public @interface ThreadSafe
{
}
```

```
package VenkatsFramework;

public @interface NotThreadSafe
{
}
```


Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Using Annotation

```
import VenkatsFramework.ThreadSafe;
import VenkatsFramework.NotThreadSafe;

@ThreadSafe
public class MyClass
{

    @NotThreadSafe
    public void foo()
    {

    }
}
```


Meta Annotations

- Use to talk about Annotations themselves
- Where can it go?
- Is it retained at runtime?
- ...

Retention

- Says whether VM retains the annotation for reflective access at runtime
- RetentionPolicy
  - Source—discarded at compile time
  - Class—Kept in bytecode, but not loaded into VM (This is the default)
  - Runtime—Available for reflective access at runtime

Target ElementType

- Specifies where it can be used
  - ANNOTATION_TYPE
  - CONSTRUCTOR
  - FIELD
  - LOCAL_VARIABLE
  - METHOD
  - PACKAGE
  - PARAMETER
  - TYPE

Inherited

- Tells whether the annotation specification in a class is inherited by its derived class
- Some annotations may make sense to be inherited (like Cloneability), others don't (like AuthorInformation)

Documented

- Details of annotation is documented in Javadoc

Annotation Elements

```
package com.agiledeveloper;

import java.lang.annotation.*;

@Retention(RetentionPolicy.RUNTIME)
@Target({ElementType.TYPE, ElementType.METHOD})
@Inherited
public @interface AuthorInfo
{
    public String name();
    public String email() default "";
    public String comment() default "";
}
```

Tells us that we can use this annotation only on classes and methods

Using Elements

```
package com.agiledeveloper;

@com.agiledeveloper.AuthorInfo(name = "Venkat Subramaniam")
public class SomeClass
{
    @com.agiledeveloper.AuthorInfo(name = "Venkat Subramaniam", comment = "bug free")
    public void foo()
    {

    }
}
```

Email has default value (empty String)

Invalid Usage

```
@com.agiledeveloper.AuthorInfo(
  @'com.agiledeveloper.AuthorInfo' not applicable to field .am", comment = "no comments")
private int val1;
```


Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Annotation Processing

- Java 5 has a tool called Annotation Processing Tool (APT)
- Java 6 has this capability built in
- `-processor` switch tells javac to use annotation processor before compiling the code

Annotation Processing

```
@SupportedAnnotationTypes({"VenkatsFramework.ThreadSafe", "VenkatsFramework.IsNotThreadSafe"})
@SupportedSourceVersion(RELEASE_6)
public class ThreadSafetyAnnotationProcessor extends AbstractProcessor
{
    public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment roundEnv)
    {
        System.out.println("-------------------------");
        System.out.println("Recevied Annotations: " + annotations);
        System.out.println("Received RoundEnvirtonment: " + roundEnv);
        for(TypeElement annotation : annotations)
        {
            System.out.println("Processing annotation: " + annotation);

            Set<? extends Element> elems = roundEnv.getElementsAnnotatedWith(annotation);

            System.out.println("Elements for that: " + elems);
        }

        return true;
    }
}
```

Annotation Processing

```
>javac -classpath %CLASSPATH% -processor VenkatsFramework.ThreadSafetyAnnotation
Processor src\com\agiledeveloper\*.java
--------------------------
Recevied Annotations: [VenkatsFramework.ThreadSafe, VenkatsFramework.IsNotThread
Safe]
Received RoundEnvirtonment: [errorRaised=false, rootElements=[com.agiledeveloper
.MyClass], processingOver=false]
Processing annotation: VenkatsFramework.ThreadSafe
Elements for that: [com.agiledeveloper.MyClass]
Processing annotation: VenkatsFramework.IsNotThreadSafe
Elements for that: [com.agiledeveloper.MyClass]
--------------------------
Recevied Annotations: []
Received RoundEnvirtonment: [errorRaised=false, rootElements=[], processingOver=
true]
```


Runtime Reflection API

- isAnnotationPresent()
- getAnnotation()

Exploring Annotation

```
package com.agiledeveloper;

public class Explorer
{
    public static <T> void report(Class<T> theClass)
    {
        if (theClass.isAnnotationPresent(AuthorInfo.class))
        {
            AuthorInfo authInfo =
                    theClass.getAnnotation(AuthorInfo.class);

            System.out.println(theClass.getName() +
                    " has @AuthorInfo annotation:");
            System.out.printf("name = %s, email = %s, comment = %s\n",
                              authInfo.name(),
                              authInfo.email(), authInfo.comment());
        }
        else
        {
            System.out.println(theClass.getName() +
                    " doesn't have @AuthorInfo annotation");
        }

        System.out.println("-----------------");
    }

    public static void main(String[] args)
    {
        report(SomeClass.class);
        report(DerivedFromSomeClass.class);
    }
}
```


Exploring Annotation

AuthorInfo has been marked with @Inherited
(not that it makes sense to)

```
com.agiledeveloper.SomeClass has @AuthorInfo annotation:
name = Venkat Subramaniam, email = , comment =
-----------------
com.agiledeveloper.DerivedFromSomeClass has @AuthorInfo annotation:
name = Venkat Subramaniam, email = , comment =
-----------------
```

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

In Web Service

```
package com.agiledeveloper;
import javax.jws.WebService;

@WebService
public class WeatherInfo
{
	public double getTemperature(String city)
	{
		return Math.random() * 100; //... do real work...
	}
}
```

Transaction In Spring

```
package com.bank;

import org.springframework.transaction.annotation.Transactional;

@Transactional
public class AccountService
{
```

```
    <tx:annotation-driven transaction-manager="txManager" />
</beans>
```

One line in config to enable this

Configuration In Spring

```
import org.springframework.beans.factory.annotation.Configurable;
import org.springframework.beans.factory.annotation.Autowire;

@Configurable(autowire = Autowire.BY_NAME)
public class AccountService
{
```

In JUnit 4.0

```
public class MyTest
{
    @Test
    public void testSomeUsefulUnitOfWork()
    {
      //...
    }

    @Test(timeout = 10000)
    public void testSomeUsefulCriticalOp()
    {
        //...
    }
```


And More...

- AspectJ
- Hibernate 3.2

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Annotation Hammer

- A guy with only a hammer looks at everything as a nail
- Not everything should become a Annotation
- @Persist in Tapestry to indicate a Bean needs persistence—sounds reasonable
- If I decide I don't want it, must in the code may be affected anyways

Annotation Hammer

- What about security settings for a method?
- Does that belong in the code?
- If you need to change this setting, would you want to modify the code, recompile, and redeploy?

What's Annotation for?

- Annotation must express intrinsic behavior
- If something is extrinsic and better expressed outside code, it should be
- If you can choose Convention over Configuration, you should instead of any form of configuration, XML, Annotation, ...

Use Annotation if...

- Metadata is intrinsic
- Is simpler to express and easier to work with as annotation
- What you're specifying is class based, not instance specific

Avoid Annotation if...

- Don't blindly convert configuration to code
- Some are better to be in external configuration
  - If you hate XML, find alternatives, don't assume it's annotation
- If you have a elegant convention driven approach, don't be pressure to use annotation
- If change is often and you don't want to recompile the code
- If you can use convention instead

Agenda

- Metadata
- Annotation
- Java Annotations
- Writing Annotation
- Using Annotation
- Processing
- Real Uses
- Proper Usage
- Conclusion

Quiz Time

References

- http://java.sun.com/docs/books/tutorial/java/javaOO/annotations.html
- http://www.infoq.com/articles/Annotation-Hammer

Thank You!

http://www.agiledeveloper.com — download