

AnnaBot

Annotations
Assertions
Testing
Java

Ian Darwin

<http://www.darwinsys.com/annabot/>
<http://www.darwinsys.com/javacook/>

Annotations are sticky notes

@AnnotateThis

- Java Annotations are like Sticky Notes
 - Attach metadata to Java source code
 - Retained in compiled class file (usually)
 - Standard mechanism for runtime discovery
 - `someClass.getMethod("fred").getAnnotations()`
- In Java language since 1.5 (2004)
- Used in Spring, Hibernate, Seam, JavaEE(EJB/JPA), JAX-WS, etc.

Annotations In Action

Run-time (JAX-WS)

```
@WebService
```

```
public class Fred extends Caveman {
```

```
    @Override
```

Compile-time
(not saved in .class file)

```
    public void callHome() {
```

```
        // call Wilma here
```

```
    }
```

```
}
```

@Not Without Problems

- Extra annotations are ignored!
 - 100% open-ended namespace
 - A framework only looks for annotations it expects, where it expects them
 - If developer puts annotation on wrong class or method or field, *framework will not see it*
 - No standard tools for checking
- To a tools developer, problem == opportunity

Enter AnnaBot

- The bot that checks your annotations?
 - Not smart enough to be a Bot (“Marketing”)
- Simple mechanism
 - Depends on meta-meta-data
 - “Claim” file == Assertion about certain Annotations and their usage
 - Input = claim file(s) + target files
 - Test each class in targets against each claim.

It really stands for
Annotation **A**ssertion **B**ased **O**bject **T**esting!

Example Error

What's wrong with this picture?

```
@Entity public class Person {  
    @Id int id;  
    @Column(name="given_name")  
    public String getFirstName() {  
        return firstName;  
    }  
}
```



AnnaBot code to catch it

```
claim JPAEntityClaim {  
  if (class.annotated(Entity)) {  
    atMostOne  
      method.annotated(javax.persistence.*),  
      field.annotated( javax.persistence.*)  
    {  
      error "Annotate JPA methods OR fields";  
    }  
  }  
}
```

Inside is simple

- “Claim” language will be compiled into Java bytecode using Antlr and Javassist
 - Antlr grammar is done; need to write actions
- For now write Claims as Java method calls
- Main program is simple
 - Find claim and target classes
 - Run each claim against each target

“Run claim against target?”

- Uses Java Reflection API to check if method is annotated, etc.
- “Claim” class is Composite of Operators
- “Operator” interface: process(Class)
- “tree” package classes implement Operator
 - *Annotated*, *MethodAnnotated*, *FieldAnnotated*
 - *IsAnnotated* method – does “real” tests
 - Operators for *And*, *Or*, *AtLeastOne*, etc.

Shine that mirror under the hood

```
Field[] fields = c.getDeclaredFields();    // From CVS -r 1.1.1.1
boolean fieldHasJpaAnno = false;
for (Field field : fields) {
    Annotation[] ann = field.getDeclaredAnnotations();
    for (Annotation a : ann) {
        Package pkg = a.annotationType().getPackage();
        if (pkg != null && pkg.getName().startsWith("javax.persistence")) {
            fieldHasJpaAnno = true;
            break;
        }
    }
}
```

Can Write Claims in Java

```
public class JPAEntityMethFieldClaim extends Claim {  
    public String getDescription() {  
        return "Annotate JPA methods OR fields";  
    }  
    public Operator[] getClassFilter() { // The "if" part  
        return new Operator[] {new ClassAnnotated("javax.persistence.Entity") };  
    }  
}
```

// Continued...

Java Claim (continued)

```
public Operator[] getOperators() {  
    return new Operator[] {  
        new AtMostOne(  
            new FieldAnnotated("javax.persistence.*"),  
            new MethodAnnotated("javax.persistence.*"))  
        };  
    }  
}
```

AnnaBot Performs

- Runs on 150 classes with a dozen claims in a few seconds
 - Too slow to run on every save in IDE
 - Too fast **not** to run it (hourly CI server)

Recent Changes

- Lots of slacking (with good excuse)
- Regex matching for names, e.g.,
`MethodAnnotated(javax.persistence.*, "set.*")`

It will be even better

- “Claim Compiler” (AnnaBotC) will make it easier to add claims
 - Compile down to .class file
 - Can read claims from Jars
 - Will ship with jpa-claims.jar, spring-claims.jar, ...
- Claim language refinement
 - Annotation Attribute Assertions

...

More Better

- Annotations can have *attributes*

```
@Column(name="my $*!# SQL column")
```

```
public String getDepartment() { ... }
```

- Results may be “undefined” - invalid table name
 - implementations, portability, ...
- Annotated subclasses need to do matching

Cross-class Checking is Hard

- JPA requires that Entity modifier annotations apply to methods or fields
 - This is per ClassLoader not per class!
 - Need to extend syntax to handle this.

The Source, Of Course

- AnnaBot is open-source, BSD-licensed

```
export CVSROOT=:pserver:\
anoncvs@cvs.darwinsys.com:/cvspublic
cvs checkout annabot
```

- Please contribute back:
 - New claims!
 - “cvs diff -u” patches to existing

There's a bit more detail, but...

- Technical paper
 - Submitted to ISSTA (ACM SIGSoft / SIGPLAN)
 - In “quarantine” due to standard non-publication-elsewhere clause

A photograph of the Chicago skyline with a seagull in flight. The skyline includes several prominent skyscrapers like the Willis Tower. The seagull is white with dark wings, flying across the blue sky.

ISSTA 2009

July 19th-23rd, 2009

**International Symposium on
Software Testing and Analysis**

Chicago, Illinois, USA

Changes Since Paper

- No longer M.Sc. Candidate
 - Defended thesis May 14th (& survived!)



AnnaBot

Q & A

<http://www.darwinsys.com/annabot/>