Reading annotations

Using the Annotations reader class

You can retrieve the backport175 annotations as well as regular Java 5 annotations (when running Java 5) at runtime using the API in the `Annotations` class.

You retreive the annotations by specifying the:

- name of the annotations - this must be the fully qualified name of the annotation interface.
- class or member that is annotated - this is the `java.lang.Class` (for class level annotations) or one of the member class in the `java.lang.reflect.*` package for member level annotations (methods, fields and constructors)

Here is a sample part of the API:

```
// class annotations
public static boolean
isAnnotationPresent(Class annotation, Class
klass)
public static Annotation getAnnotation(Class
annotation, Class klass)
public static Annotation[]
getAnnotations(Class klass)

// method annotations
public static boolean
isAnnotationPresent(Class annotation, Method
method)
public static Annotation getAnnotation(Class
annotation, Method method)
public static Annotation[]
getAnnotations(Method method)

...
```

The Annotation interface

All these methods in the `Annotations` reader class return an instance of the type `org.codehaus.backport175.reader.Annotation`.

This class has some useful methods like:

```
/**
 * Returns the annotation type, e.g. the annotation interface type.
 *
 * @return the type
 */
Class annotationType();

/**
 * Returns a string representation that is identical to the annotation
 * that was parsed by the compiler.
 *
 * @return the string representation
 */
String toString();
```

But most likely you want to cast the instance to the correct annotation type (interface).

Example

Here is an example on how to read in the EJB 3 transaction annotation:

```
if
(Annotations.isAnnotationPresent(javax.ejb.T
ransactionAttribute.class, method)) {

    Annotation annotation =
Annotations.getAnnotation(javax.ejb.Transact
ionAttribute.class, method);

    javax.ejb.TransactionAttribute
txAttribute =
(javax.ejb.TransactionAttribute)annotation;
}
```

Notes

If an annotation has an element with a default value, the reflective API will off course gives you access to it if the annotated class/member was not specifying it.