Runtime management

Introduction

backport175 has been designed to work in highly dynamic environment (such as load-time and runtime weaving for AOP). In environments like this the bytecode on disk is not the "current" one being executed. Since the bytecode has been transformed during the execution of the application. What this means is that annotations can have been added or removed.

We therefore need two things:

1. a way for the vendor/user to define a bytecode provider since the current one is not available on disk
2. an API for refreshing/updating the annotation reader, so we know that the annotations we have reflects the annotation in the latest bytecode

All these operation should be fully thread-safe, if not, then we have a bug.

Defining the bytecode provider

This is done by implementing the `org.codehaus.backport175.reader.bytecode.spi.BytecodeProvider` interface. This interface is a strategy with one single method you need to implement:

```
public interface BytecodeProvider {
    /**
     * Returns the bytecode for a specific
class.
     *
     * @param className the fully qualified
name of the class
     * @param loader the class loader that
has loaded the class
     * @return the bytecode
     */
    byte[] getBytecode(String className,
ClassLoader loader);
}
```

Registering the bytecode provider

The bytecode provider implementation class is registered in the `org.codehaus.backport175.reader.byteco`

`de.AnnotationReader` class by invoking:

```
AnnotationReader.setDefaultBytecodeProvider(
new BytecodeProviderImpl());
```

If no bytecode provider is registered then a default one that uses `getResourceAsStream()`, e.g. reads in the bytecode from disk, is used.

It is also possible to set a bytecode provider on a per class basis, which can be usefull in some circumstances. Refer to the API documentation for more details.

Refreshing the annotation reader

When a change to the bytecode has been made it is needed to refresh the annotation reader, so that it can pull the new bytecode from the `BytecodeProvider`, parse it and update the reader with the new information about the annotations.

This is done by invoking one of the `refresh(..)` methods in the `org.codehaus.backport175.reader.bytecode.AnnotationReader` class:

```
// refreshes one single class
AnnotationReader.refresh(POJO.class);

// refreshes all classes (that has been
accessed through the Annotations interface
so far)
AnnotationReader.refreshAll();
```