Using Slice Checksums in Java

The Slice compilers can optionally generate checksums of Slice definitions. For `slice2java`, the `--checksum` option causes the compiler to generate a new Java class that adds checksums to a static map member. Assuming we supplied the option `--checksum Checksums` to `slice2java`, the generated class `Checksums.java` looks like this:

Java

```
public class Checksums {
    public static java.util.Map<String, String> checksums;
}
```

The read-only map `checksums` is initialized automatically prior to first use; no action is required by the application.

In order to verify a server's checksums, a client could simply compare the dictionaries using the `equals` method. However, this is not feasible if it is possible that the server might return a superset of the client's checksums. A more general solution is to iterate over the local checksums as demonstrated below:

Java

```
java.util.Map<String, String> serverChecksums = ...
java.util.Iterator<java.util.Map.Entry<String, String>> i =
    Checksums.checksums.entrySet().iterator();
while(i.hasNext()) {
    java.util.Map.Entry<String, String> e = i.next();
    String id = e.getKey();
    String checksum = e.getValue();
    String serverChecksum = serverChecksums.get(id);
    if (serverChecksum == null) {
        // No match found for type id!
    } else if (!checksum.equals(serverChecksum)) {
        // Checksum mismatch!
    }
}
```

In this example, the client first verifies that the server's dictionary contains an entry for each Slice type ID, and then it proceeds to compare the checksums.

See Also

- Slice Checksums