

Class 22: Introspection in Java

Held: Monday, March 12, 2007

Summary: Today we consider the basics of introspection in the Java Programming language and some reasons we might want to use introspection.

Related Pages:

- EBoard.
- Reading: Sun Microsystems: Trail: The Reflection API.

Notes:

- Are there questions on the exam?
- So, what did you learn from the video?

Overview:

- Introspection basics.
- A problem: Factory Methods.
- A problem: Dynamic Loading.
- A problem: Serializing.
- A problem: Profiling.

Introspection Basics

- Introspection lets you write code that looks into existing objects and classes, permitting you to use such things differently.
- In some ways, introspection is like Scheme's ability to treat the same structure as either a list or a bit of code.
- Basic introspection operations in Java:
 - Given an object, determine its class.
 - Given its class, determine its methods, fields, and constructors.
 - Given a method or constructor, call it.
- Other introspection operations in Java:
 - Build a proxy
 - Modify the class loader
 - ...
- Why have introspection?
 - Easy way to have factory methods.
 - Many many more.

Example One: Factory Methods

- Sometimes you want to be able to build a lot of objects of the same type, and don't necessarily know that type in advance.
- By grabbing an object's class, and then grabbing the constructor from that class, you can easily make new objects.

Example Two: Dynamic Loading

- You can even load classes whose names you did not know when you wrote the program.
- How? Use `Class.forName(string)`.
- This strategy is often used for plug-in based programs.
 - That is, the program can read the name of the plug-in from the directory structure or from an initialization file, and then load the appropriate plug-ins using reflection.