Eclipse Plug-in Architecture

EE 564
Lecture 8

Daqing Hou, Winter 2007
Eclipse plugin architecture

is built on following techniques
- Java reflection
- Java class loading
- Jar API
- XML processing
Explain this code

```java
Class.forName("oracle.jdbc.driver.OracleDriver");

Connection con = DriverManager.getConnection("jdbc:oracle:thin:@dbaprod1:1544:SHR1_PRD", username, passwd);
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>  
        <head>  
        <title>Hello World!</title>  
        </head>  
        <body>  
        <h1>Hello World!</h1>  
        </body>  
        </html> ");
    }
}

Daqing Hou, Winter 2007
Background

• Java byte code & class files

    dhous$ hexdump -C Factor.class
    00000000  ca fe ba be 00 00 00 31 00 50 07 00 02 01 00 14 |.......1.P......|
    00000010  63 6c 61 72 6b 73 6f 6e 2f 6d 61 74 68 2f 66 65 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |clarkson/math/Fa| 00000020  0a 00 03 00 09 0c 00 05 00 06 01 00 0f 4c 6a 61 76 61 2f 6c 2f 69 74 3e 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000030  0e 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000040  0b 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000050  00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000060  00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000070  0e 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000080  0b 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 00000090  00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 000000a0  0e 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 000000b0  0b 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 000000c0  00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...| 000000d0  0e 00 09 53 69 67 6e 75 6d 62 65 72 54 61 62 6c 65 01 00 12 4c 63 6c 61 72 65 74 75 72 65 66 72 25 2c 77 61 67 69 6f 6e 2f 68 61 73 68 4d 61 70 3b |...|

• Java class files contain also data about types (a.k.a meta-data)

Daqing Hou, Winter 2007
Reflection API

• Type info. made available at runtime

• java.lang.Class
  - forName("qualified class name")
  - newInstance()
  - getConstructor(Class[] paramType)
  - getMethod(String name, Class[] paramTypes)
  - getField(String name)
Reflection API

- java.lang.reflect.Method
  - Object invoke(Object, Object[]) throws IllegalAccessException - if this Method object enforces Java language access control and the underlying method is inaccessible.

Daqing Hou, Winter 2007
Java class loader

• Creates class from Java byte code
• Subclass java.lang.ClassLoader to create new loaders
• New loader knows where to load a class
• For each application,
  - JVM provides bootstrap loader
  - optionally, application may define loaders
  - All class loaders form a tree rooted at the bootstrap loader
java.lang.ClassLoader

- public loadClass()
  *call parent loader first if failed then findClass()*
- protected findClass()
- protected final defineClass()
- protected findLoadedClass()
Why class loader?

- Division and integration of work
- network computing
- security
Tomcat’s class loaders

• Bootstrap, system, common, catalina, shared, web app loaders

• Each loader knows a location to load classes

• See http://tomcat.apache.org/tomcat-4.1-doc/class-loader-howto.html
Whistles and bells

• Use XML to do configuration as eclipse manifest files do
• Package classes and resources of plug-ins in jar format
• Load classes using API from java.util.jar
Resources on internet

- **Class loading**

- **Jar**

- **XML processing**

Daqing Hou, Winter 2007