

Java Platform Enhancements

A Quick Look at the Platform Enhancements

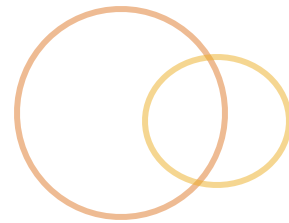
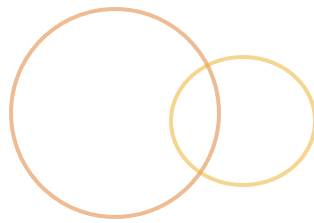


Java Package Enhancements

New and Modified Packages

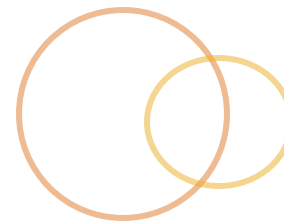


Reflection



- Provides run-time information discovery for classes, fields, methods, and constructors
- Introduced in 1.1
- Updated to support new language features
- Performance enhancements made in 1.4

JavaBeans API





- ◎ Formalized component model for Java
- ◎ Introduced in 1.1
- ◎ Spec driven; current spec is 1.01
- ◎ Few new features added

JDBC (Java Database Connectivity)



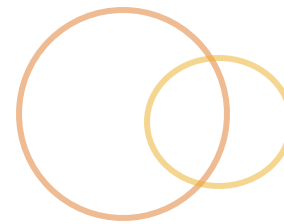
- ◎ JDBC extends write-once run-anywhere (WORA) to database world
- ◎ Introduced in Java 1.2
- ◎ Current spec release is 4.0; Java 5 contains spec 3.0
- ◎ Formalized facilities added to `javax.sql`

Utilities API



- ⦿ Contains APIs for common utility classes
- ⦿ Introduced in 1.0
- ⦿ Updated to support new language features
- ⦿ Few new classes introduced
- ⦿ New “sub-packages” introduced

Collections API



- ⦿ Contains well-defined data structure implementation resources
- ⦿ Introduced in 1.1
- ⦿ Updated to support new language features
- ⦿ Additional functionality to support concurrency packages

Concurrent Collections



- ◎ Extension of Collections Framework
- ◎ Added in 1.5
- ◎ Provide thread safety
- ◎ More “lightweight” than synchronized
 - ◎ Typically synchronize on manipulation
 - ◎ Typically retrieval is not synchronized
- ◎ Provides many queue implementations

Java API for XML Processing



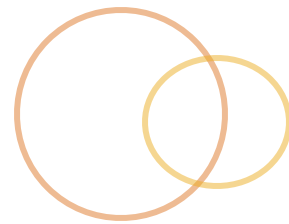
- ◎ Provides XML processing capabilities
- ◎ Spec driven; current spec 1.3
- ◎ Improvements to core XML support, including XML Schema and DOM 3

Java Management Extensions



- ◎ Provides management and monitoring capabilities to Java platform
- ◎ *New packages*
 - ◎ `java.lang.management`
 - ◎ `javax.management`
- ◎ Bundled and integrated into 1.5
- ◎ Spec driven; current spec 1.4

Concurrency Utilities



- ◎ Provides robust concurrency libraries to Java
- ◎ *New package* - `java.util.concurrent`
- ◎ Formalized and bundled with 1.5
- ◎ JSR driven – JSR 166

Other Library Enhancements



Instrumentation Utilities

- New package - `java.lang.instrument`
- Provides byte-code level modification at run-time

Networking

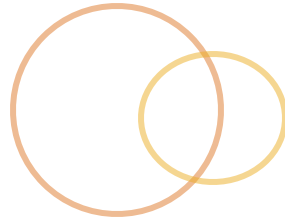
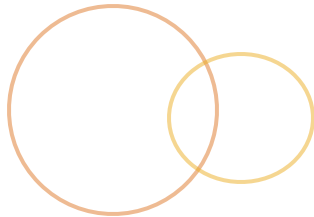
- Complete IPv6 support
- Support for `inetd`

Internationalization – now supports Unicode 4.0 standard

Wrappers – now support bitwise operations

Math – has new geometry functions and more precision

Java Platform Enhancements



Overview of Platform Enhancements

Operating Environment Modifications

- High-precision timing
- API support for nanosecond based time captures
- `System.nanoTime() : long`
- Capabilities are platform dependent

Environment Variable Access

- Mechanism to access system environment variables
- Differs from `System` properties
- Formal introduction of `getenv`
- Provides `Map` collection of system environment variables

System.getenv Example



```
package examples.platform;

import java.util.Map;
import static java.lang.System.*;

/** ... */
public class EnvironmentAccess {

    /** ... */
    public static void main(String[] args) {
        if(args.length == 0) {
            System.out.println("Unable to access specific env var, retrieving all env vars");
            Map<String, String> envValues = getenv();
            for(String s : envValues.keySet()) {
                System.out.println(s + "=" + envValues.get(s));
            }
        } else {
            String envValue = System.getenv(args[0]);
            System.out.println(args[0] + "=" + envValue);
        }
    }
}
```

Overview of Platform Enhancements [cont.]

🕒 New `ProcessBuilder` class

- 🕒 Mechanism for invoking sub-processes
- 🕒 More configurable and customizable than `Runtime.exec`
- 🕒 Can start a process in its own “environment”

```
ProcessBuilder pb = new ProcessBuilder("myCommand", "myArg1", "myArg2");  
Map<String, String> env = pb.environment();  
env.put("VAR1", "myValue");  
env.remove("OTHERVAR");  
env.put("VAR2", env.get("VAR1") + "suffix");  
pb.directory("myDir");  
Process p = pb.start();
```


Modifications made to the JVM



◎ Class Data Sharing

- ◎ Reduces startup time and footprint of JVM
- ◎ Core platform libraries shared across JVM instances
- ◎ Can be turned off

◎ Garbage Collector Ergonomics

- ◎ Uses parallel collector instead of serial collector
- ◎ Provides automatic adaptive tuning to VM
- ◎ Instead of tuning VM, specify goals and VM will tune itself

Modifications made to the JVM [cont.]

VM Mode detection

- VM will attempt to detect server or client mode at startup
- Only occurs if `-server` or `-client` aren't present
- Makes decision based on platform architecture

Platform		Default VM		
Architecture	OS	client VM	if server-class, server VM; otherwise, client VM	server VM
SPARC 32-bit	Solaris		X	
i586	Solaris		X	
	Linux		X	
	Microsoft Windows	X		
SPARC 64-bit	Solaris	—		X
AMD64	Linux	—		X
	Microsoft Windows	—		X

Legend: **X** = default VM — = client VM not provided for this platform