JAVA 8 TO JAVA 17: THE NEW GOODIES

JESSE GALLAGHER

CTO - I KNOW SOME GUYS
IP MANAGER - OPENNTF
HTTPS://FROSTILLIC.US
@JESSE@PUB.FROSTILLIC.US

Prerequisites

- Comfort with (or willingness to learn) Java
- Familiarity with annotations and Java 8 constructs (Optional, etc.) a plus

YOU DO NOT NEED:

- Knowledge of OSGi
- Knowledge of XPages

JAVA RELEASE CADENCE AND DISTRIBUTIONS

The Old Ways

- Java 1 through 6 came out at a pretty steady pace
- Things slowed down after Java 6
  - Domino was stuck at that version even longer than everyone else

Java 1.0	1996
Java 1.1	1997
Java 1.2	1998
Java 1.3	2000
Java 1.4	2002
Java 5	2004
Java 6	2006
Java 7	2011
Java 8	2014

The New Ways

- Starting with Java 9, there's a new system
  - A new release every six months
  - Some are considered "LTS"
  - Originally the plan was every six releases, now every four

Java 9	Sept. 2017
Java 10	March 2018
Java 11	Sept. 2018
Java 12-16	March 2019-2021
Java 17	Sept. 2021
Java 18-20	March 2022-2023
Java 21	Sept. 2023
Java 22	March 2024

Java Distributions

- Java is open source, with multiple providers
  - Oracle maintains "normal" Java, but its licensing terms have gotten more restrictive
  - Eclipse maintains Adoptium, with their version called "Temurin": fully free and open source
  - IBM's J9 variant, used in Domino, is open source as "Semeru"
  - There are others, from Amazon, Microsoft, and more, often suited well for their cloud platforms

BETTER NULLPOINTEREXCEPTIONS

Better NullPointerExceptions

- This is a nice freebie: NPEs now tell you WHAT was null

```
Employee someGuy = new Employee("Some Guy", "IT", null);
System.out.println(someGuy.id().length());
```

```
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because the return value
of "j17test.Employee.id()" is null
    at j17test/j17test.Test.main(Test.java:15)
```

STRINGS

Strings

- .strip(), .stripLeading(), .stripTrailing() - like trim(), but with more knowledge of Unicode whitespace
- .stripIndent() - detects and removes leading indentation, such as in code blocks
- .indent(n) - creates indentation
- .repeat(n) - repeats a string in place, e.g. "foo ".repeat(2) -> "foo foo "
- .isBlank() - checks if the string is empty or only has whitespace
- .lines() - returns a Stream of the string split by newlines
- .transform(...) - apply a function to make another object, useful in functional-style code

Text Blocks

```
String foo = """
    I am a multi-line block of
    text, all in one string literal.
    The result is much, much better when
    writing, for example, HTML or other
    code inside Java
""";
```

HTTPCLIENT

HttpClient

- Java has long had HttpUrlConnection and friends
- It does the job, but only barely - the mechanics are awkward and it's pretty light on features
- Third-party libraries like Apache HttpClient fill the gaps, but require dependencies

HttpClient

- Added in Java 9, java.net.http.HttpClient is much more featureful
- Configuration options are provided in a builder
- Supports HTTP/2 and WebSocket (as a client)
- Can run async

```
HttpClient client = HttpClient.newBuilder()
    .authenticator(...)
    .sslContext(...)
    .followRedirects(Redirect.ALWAYS)
    .connectTimeout(Duration.ofMinutes(1))
    .version(Version.HTTP_2)
    .build();
```

HttpClient

- Create HttpRequest objects with a similar builder
- Provide a BodyHandler to handle the response - BodyHandlers contains useful default choices

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest req = HttpRequest.newBuilder(URI.create("https://mysite.com"))
    .GET()
    .header("Authorization", "Bearer 12345")
    .build();
String result = client.send(req, BodyHandlers.ofString()).body();
```

RECORDS

Records

- Records are a new type, alongside classes, interfaces, annotations, and enums
- Meant to represent immutable data in a compact way
- Somewhat conceptually similar to structs in C, though not identical
- Similar to Lombok's, but faster due to being built-in
- May be of limited use in Domino for a while - they're not bean-type, and our frameworks don't know what to do with them
  - (Jakarta EE will if it doesn't already, though)

Records

```
public record Employee(String name, String department, String id) {
  // Has implicit methods:
  //   String name()
  //   String department()
  //   String id()
  // Also implicit equals, hashCode, and toString
}
```

SYNTAX SUGAR

Better `instanceof`

`instanceof` can now do an implicit cast

```
if(obj instanceof String) {
    String someString = (String)obj;
    // ...
} else if(obj instanceof Number) {
    Number someNumber = (Number)obj;
    // ...
} // and so forth
```

```
if(obj instanceof String someString) {
    // ...
} else if(obj instanceof Number someNumber) {
    // ...
} // and so forth
```

Switch expressions

- `switch` can now act as an expression
- In preview mode and future versions, it can do casts

```
String result = switch (someString) {
    case "foo" -> "You want Foo";
    case "bar" -> "You want Bar";
    case "baz" -> {
      // extra code here
      yield "You wanted Baz";
    }
    default -> throw new IllegalArgumentException("Unhandled " + someString);
};
```

Type inference with `var`

- The new `var` keyword can be used to infer the type without writing it
- Unlike `Object`, the variable will be the actual type of the value

```
var someString = "hello";
someString.length();

var someNumber = 3;
var methodResult = someMethod();
```

GRAB BAG

Grab Bag

- Private interface methods
- "Diamond" operator on anonymous classes
- .orElseThrow() on Optional - like .get(), but with clearer meaning
- Can't use sun.misc.Base64Encoder anymore
  - Use java.util.Base64 in 8 and above
- java.policy moved from jvm/lib/security -> jvm/conf/security
  - Use ~/.java.policy on Linux/Docker or whatever it is on Windows

"GOOD TO KNOW" CHANGES

jvm/lib/ext

- Gone!
- This was common for Domino developers (and, painfully, HCL) to stash JARs to be present to the whole Java classpath
  - Useful for agents to avoid memory problems, and useful for XPages to avoid permissions problems
- Domino has long had "ndext" that was on the runtime classpath
  - Now it's also on the build-time classpath in Designer
  - ...except for XPages, where you have to add libraries manually

Security Manager

- Deprecated!
- This is a mechanism that dates from the early days of Applets but has, for some reason, survived to this day
- It still remains for now, and still hinders XPages development in the same way, but it's not long for this world in the core JVM
- This is fine, actually:

```
> WARNING: A terminally deprecated method in java.lang.System has been called
WARNING: System::setSecurityManager has been called by lotus.notes.AgentSecurityManager (file:/C:/Domino/ndext/Notes.jar)
WARNING: Please consider reporting this to the maintainers of lotus.notes.AgentSecurityManager
WARNING: System::setSecurityManager will be removed in a future release
```

Java EE Component Move

- Several components properly part of (or adjacent to) Java EE are in the core JVM in Java 8:
  - JAX-B (Object <-> XML mapping)
  - JAX-WS (SOAP web services)
  - Activation (don't worry about it)
  - Common Annotations (don't worry about it)
  - CORBA (only shows up if you try to compile something with Notes.jar, really)
- They're gone in Java 11+, but Domino includes them in "ndext", so you only need to care if you're doing external development
- https://openjdk.org/jeps/320

LESS USEFUL IN DOMINO

Modules

- The module system (JPMS) is actually a huge deal
  - It involved restructuring the internals of the JVM into modules
  - This happened in Java 9, as part of "Project Jigsaw"
- It involves cleaner separation between components
- Allows you to define just which parts of the JVM you need, which can create a slim build for microservices and the like

Modules

- We probably won't use them, though
- App servers generally paper over the requirements
- While some concepts are like OSGi, they are not the same
- Domino will always have the full JVM, so the "slim build" aspect doesn't mean anything

Sealed Classes

- Allows you to specify which other classes may extend a base class
  - public class SomeBase permits ImplClass1, ImplClass2 { }
- Of use primarily for things like internal implementations in frameworks
- Might be potentially useful in a large team to signal intent
- For apps, though, it'd usually be more annoying than it's worth

JAVA 21

Java 21

- Domino 14 doesn't include this, so it's not immediately useful
- Still, it's good to know for the future and to see where the language is going

Sequenced Collections

- New interfaces more specific than Collection for when the meaning makes sense
- SequencedCollection: there's *some* sequence - may be a Set, a List, or a Queue, but there's at least getFirst()/getLast() and friends - plus reversed()
- SequencedSet: A nice combination of Set and SequencedCollection. TreeSet is one of these (and also a SortedSet). You can think of a NotesDocumentCollection as this sort of thing
- SequencedMap: Like SequencedCollection, but for Maps. LinkedHashMap is one of these (and a good candidate for housing a JSON object)

Virtual Threads

- New type of thread that tries to re-use OS threads as possible to improve scalability
- You can *mostly* use them like normal threads and, for the right type of heavy workload, they can dramatically reduce resource usage
- If you end up trying to use them with a Domino runtime somehow, uh... good luck - Domino cares a *lot* about native threads, and virtual threads are a weird match

Decomposing Record

- Combined the improved `instanceof` from 17 with Record types

```
public class RecordTester {
  record Point(int x, int y) {
  }

  public void doSomethingWith(Object o) {
    if(o instanceof Point(int x, int y)) {
      System.out.println("I have a point with coords " + x + ":" + y);
    }
  }
```

Pattern Matching

- Combined the improved `instanceof` from 17 with the `switch` statement
- Can also switch to null, which was a constant annoyance

```
public void doSomethingWith(Object o) {
  switch(o) {
    case String str -> System.out.println("hey, it's a string: " + str.length());
    case Number num -> System.out.println("It's a number: " + num.doubleValue());
    case null -> System.out.println("I was sent null!");
    default -> System.out.println("I don't know about " + o);
  }
}
```

Enum Pattern Matching

- You can combine the previous with specific enum values for complex cases

```
enum ItemType {
  TEXT, NUMBER, COMPOSITE
}

public void checkItemType(Object o) {
  switch(o) {
    case String str -> System.out.println("hey, it's a string: " + str.length());
    case ItemType.TEXT -> System.out.println("I have a text item");
    case ItemType.NUMBER -> System.out.println("I have a number item");
    default -> System.out.println("I don't know about o");
  }
}
```

Weirder Pattern Matching

- You can use `when` to add a "guard" statement
- You probably don't want to go too nuts with these

```
public void guardSomethingWith(Object o) {
  switch(o) {
    case String str when (!str.isEmpty()) -> System.out.println("non-empty string");
    case String str when (str.startsWith("Hello")) -> System.out.println("Friendly greeting");
    default -> System.out.println("I don't know about o");
  }
}
```

And More!

- Snippets in Javadoc
- Simple Web Server
- Lots of preview features, including further incubation of native function calls

RESOURCES

Resources

- https://www.baeldung.com/java-migrate-8-to-17
- https://reflectoring.io/java-release-notes/
- https://nipafx.dev/road-to-21-upgrade/
- https://www.baeldung.com/java-lts-21-new-features
- https://mydeveloperplanet.com/2023/11/01/whats-new-between-java-17-and-java-21/
- https://frostillic.us/blog/posts/2023/12/15/notes-domino-14-fallout

THANK YOU
+ QUESTIONS