Java and Java Batch applications on the IBM Mainframe

Christian Strauer
Field Technical Sales Support IBM System z9
Christian.Strauer@de.ibm.com

IBM Systems

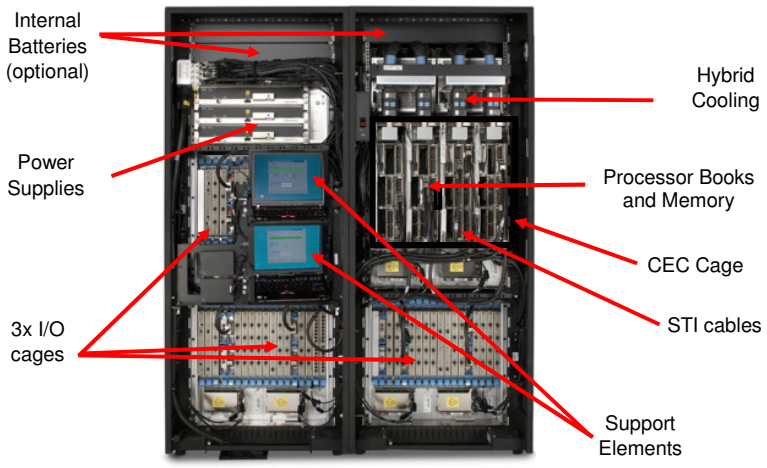
Agenda

- About the mainframe
- Java runtime environments under z/OS
  - IBM Java SDK for z/OS
  - z/OS Java APIs
  - WebSphere application server for z/OS
- For which applications should I use a mainframe?
  - Stand-alone Java applications
  - J2EE applications
- Java Development for z/OS
- Monitoring Java applications with JMX
- Summary and literature

IBM System z9 – The IBM Mainframe

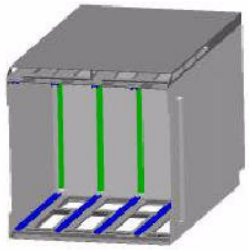
- 40 years of evolution
- Enterprise server for highest availability
- High efficient workload management
- Capacity on Demand Offerings
- Outstanding security concept:
  - Key management
  - Encryption
  - Data integrity
- Virtualisation engine
  - LPAR concept (EAL5 certified)
  - z/VM

Front View of a System z



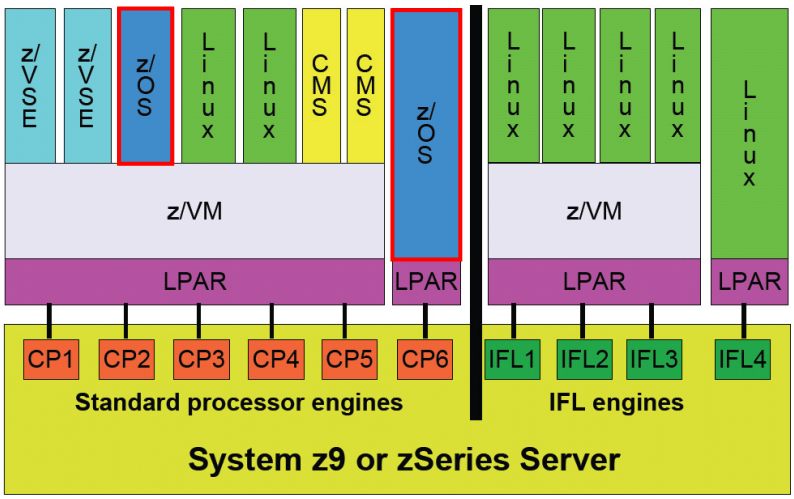
Front View

What is a "book"



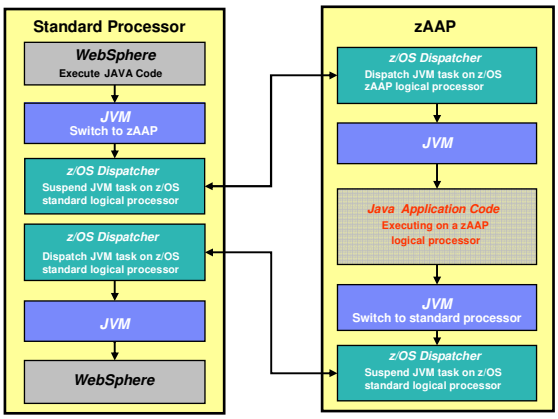
- **A System z9 book consists of**
  - Multi-Chip-Module: Processor Units (PUs)
    - 12 PUs (16 for Model S54) per book
    - 2 SAPs per book, standard
    - 2 spares per server
    - 8, 18, 28, 38 or 54 PUs available
      - CPs, IFLs, ICFs, zAAPs, zIIPs, optional SAPs
  - Memory
    - Minimum of 16 GB
    - Up to 128 GB per book
      - 16 GB increments
  - I/O
    - Up to 16 STIs per book
      - 2.7 GB/s for each I/O
    - Total system I/O bandwidth capability of 172.8 GB
    - Up to 4 Logical Channel SubSystems (LCSSs) and 60 LPAR (EC)

IBM System z9 – Operating systems



System z application assist processor (zAAP)

- **A special priced engine for Java workload on the mainframe**



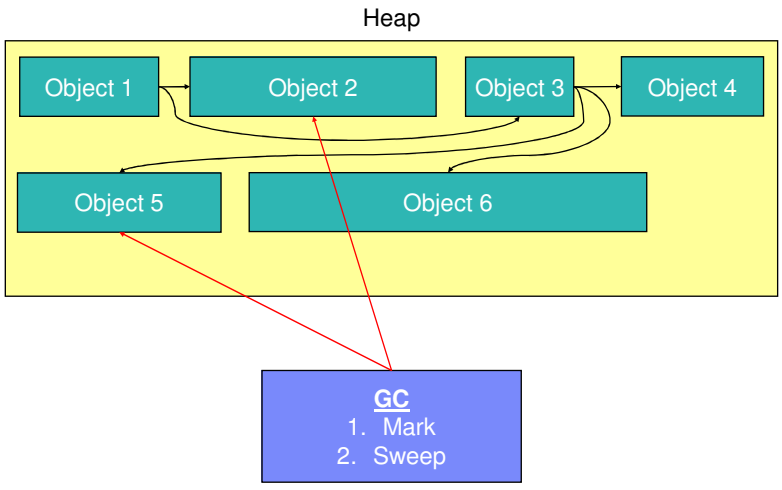
JVM 5.0 – A complete new JVM for z/OS

- **Sun IP-free, but Java 2 (1.3) compliant (J2ME) and J2SE (1.4.2, 5.0)**
- **Common code base across all platforms**
  - PowerPC, IA32, x86-64, and 390 (Linux or z/OS)
- **Flexible and sophisticated technology oriented to:**
  - **Performance (throughput and application startup)**
  - **Scalability**
  - **Reliability and Servicability (RAS)**

JVM 5.0 – Just-in-time compiler

- The just-in-time compiler (JIT) is not really part of the JVM, but is essential for a high performing Java application
- Java is Write Once Run Anywhere thus it is interpreted by nature and without the JIT could not compete with native code applications
- Due to different platforms having different JITs there is no standard method for when a method is compiled.
- As your code accesses methods the JIT determines how frequently specific methods are accessed and compiles those touched often quickly to optimize performance
- -Xquickstart helps to improve JVM startup time for short running Java applications
  - causes the JIT to run with a subset of optimizations

JVM 5.0 – Garbage collection



JVM 5.0 – Garbage collection

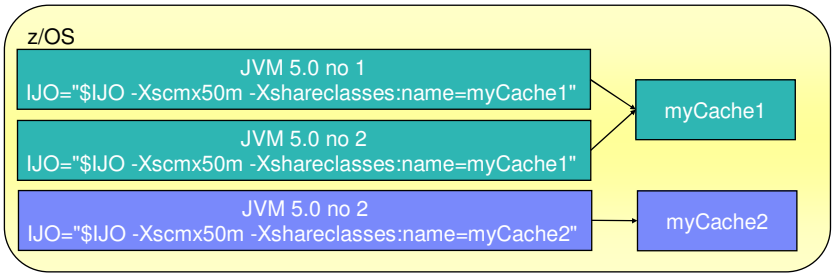
- **Memory management is configurable using four different policies with varying characteristics**

1. **Optimize for Throughput – flat heap collector focused on maximum throughput**
   - **-Xgcpolicy:optthruput**
2. **Optimize for Pause Time – flat heap collector with concurrent mark and sweep to minimize GC pause time**
   - **-Xgcpolicy:optavgpause**
3. **Generational Concurrent – divides heap into "nursery" and "tenured" segments providing fast collection for short lived objects. Can provide maximum throughput with minimal pause times**
   - **-Xgcpolicy:gencon**

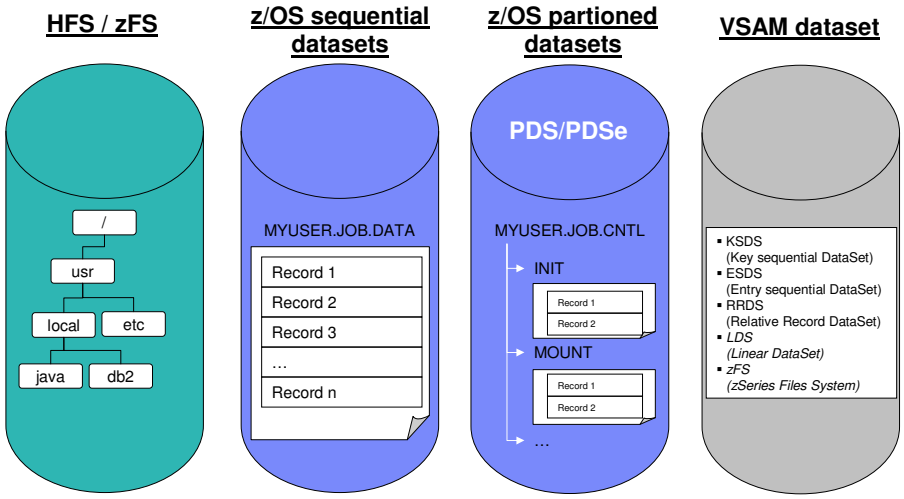
4. **Subpool – a flat heap technique to help increase performance on multiprocessor systems , commonly greater than 8. Available on IBM pSeries™ and zSeries™**
   - **-Xgcpolicy:subpool**

JVM 5.0 – Shared classes

- **A shared class area for one or more JVMs**
- **Improves startup time**
  - **Lots of classes are already preloaded**



A short z/OS data overview



Special z/OS APIs in the IBM Java SDK for z/OS

- **MVS dataset access**
  - **stream mode/record mode**
  - **DD statements possible**
  - **APIs: ZFile/JRIO**

```
ZFile zFile = new ZFile("//DD:INPUT", "rb,type=record,noseek");
try {
     byte[] recBuf = new byte[zFile.getLrecl()];
     String enc = Zutil.getDefaultPlatformEncoding();
     int nRead;
     while((nRead = zFile.read(recBuf)) > 0) {
               String line = new String(recBuf, 0, nRead, enc);
               System.out.println(line)
     };
} finally {
     zFile.close();
}
```

Based on JNI

- **USS file access**
  - **Normal file access like in Unix or Linux with java.io**

MVS console communication

- **MVS console commands**
  - **Start:** /*s jobName,commands*
  - **Modify:** /f *jobName,commands*
  - **Stop:** /p *jobName*

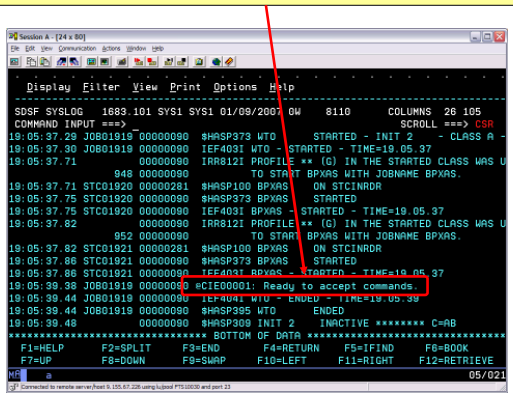
- **Code:**

```
MvsConsole.registerMvsCommandCallback(new MvsCommandCallback() {
        public void handleModify(String s) {

        }

        public void handleStart(String s) {

        }


        public boolean handleStop() {
            return true;
        }
    });
```

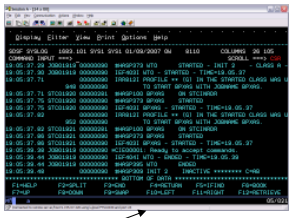

MVS console communication

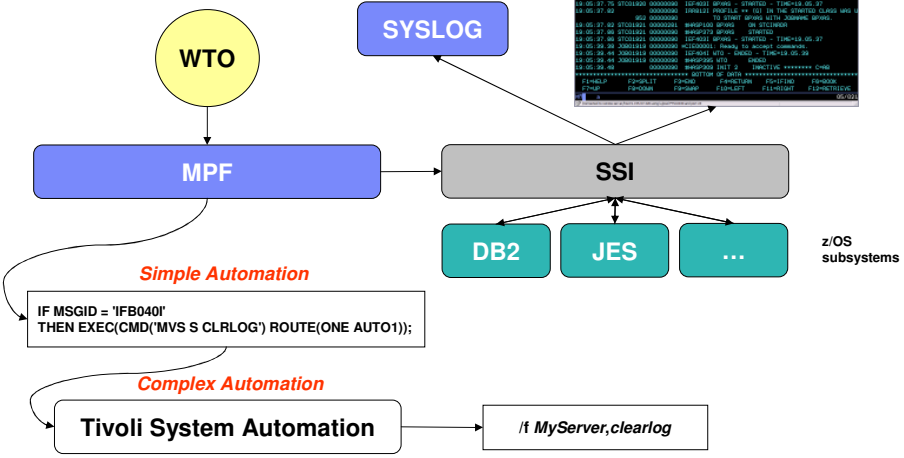
- **Write To Operator (WTO) API available**

```
{
    MvsConsole.wto("CIE00001: Ready to accept commands.",0x0020, 0x4000);
}
```

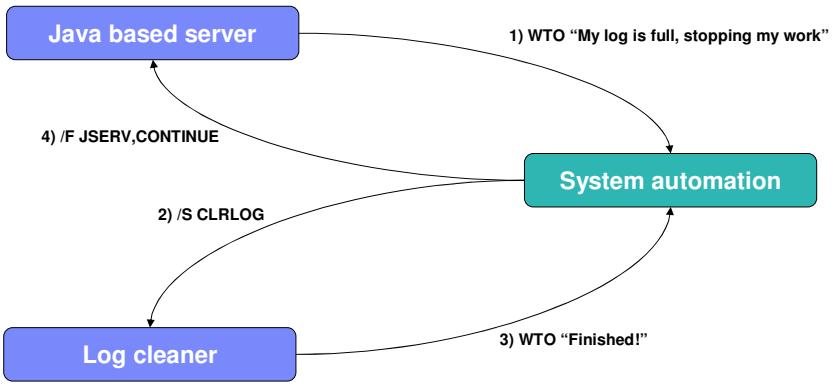


z/OS message processing

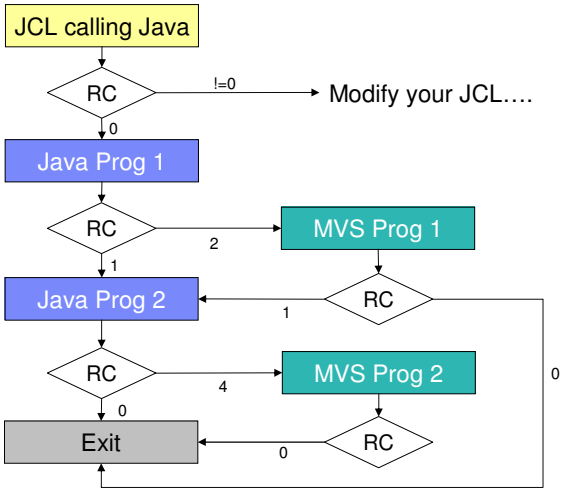




Java and z/OS console communication

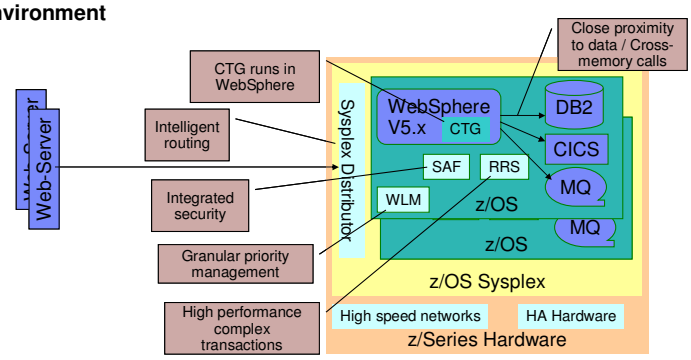


Condition Code passing for z/OS job nets



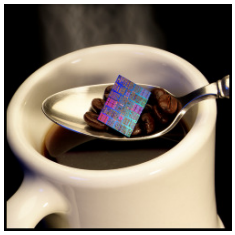
WebSphere Application Server for z/OS

- **... A normal J2EE compliant application server, but with additional features:**
  - **WAS Common code base across all platforms**
  - **Under z/OS, there is a tight integration into the mainframe specific environment**

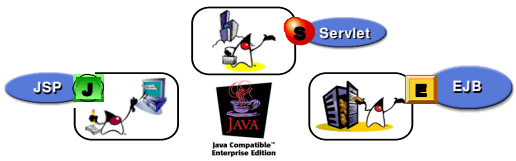


For which Java applications does a mainframe fit?

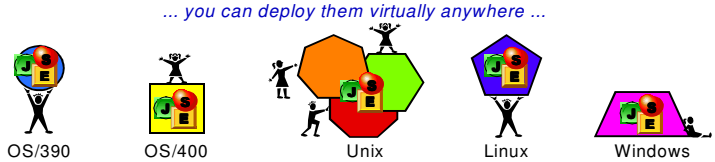
- **Batch is still one of the mainframes biggest strengths**
  - The mainframe was designed for batch (punch cards)
  - The mainframe has the longest experience in the batch environment
  - Special facilities in z/OS allow a huge complex job management for batch jobs (JES, SDSF,...)
  - Java inherits all these functionalities
- **Business critical Java based servers that need:**
  - High availibilty (99,999%)
  - Best security
- **Java applications which use lots of transactions**
  - Data proximity



J2EE: Which is the best platform to deploy?



... you can deploy them virtually anywhere ...

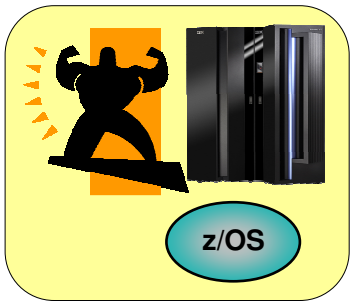


... but which option will best suit your needs??

During development, J2EE components are not optimized to any deployment platform
The server, operating system and WAS determine how well those components meet business objectives in production

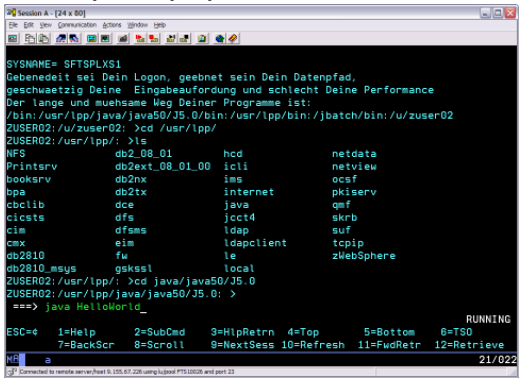
The value of the mainframe for J2EE applications

- **Strategic platform for mission-critical business processes, core business applications and data**
  - **Proximity: Fewer physical components and network connections result in less complex infrastructure**
  - **Integrity: System-managed resource recovery and transaction coordination**
- **Highest possible Qualities of Service**
  - **Security: Most stringent access control and asset protection**
  - **Availability: zero downtime z/OS brand promise**
    - **99.999% availability at the application and data level**
    - **Managed workload sharing toward achievement of business goals**
    - **Dynamic application of software changes**
  - **Efficiency: Full utilization of system capacity with same class of service**
    - **Less people to configure, monitor and adjust workload**
    - **Best Total Cost of Computing**
    - **zSeries Application Assist Processors (zAAPs) to execute Java workloads under z/OS**



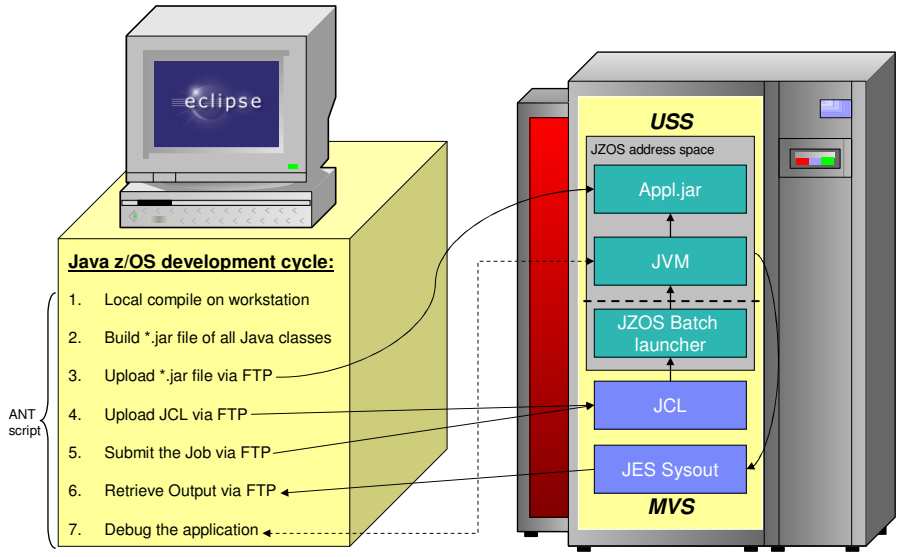
How do I develop z/OS Java applications?

- **This is a picture people often associate with the mainframe:**



- **... But it is much easier!**
  - **Eclipse as an IDE can be easily used for Mainframe Java development**

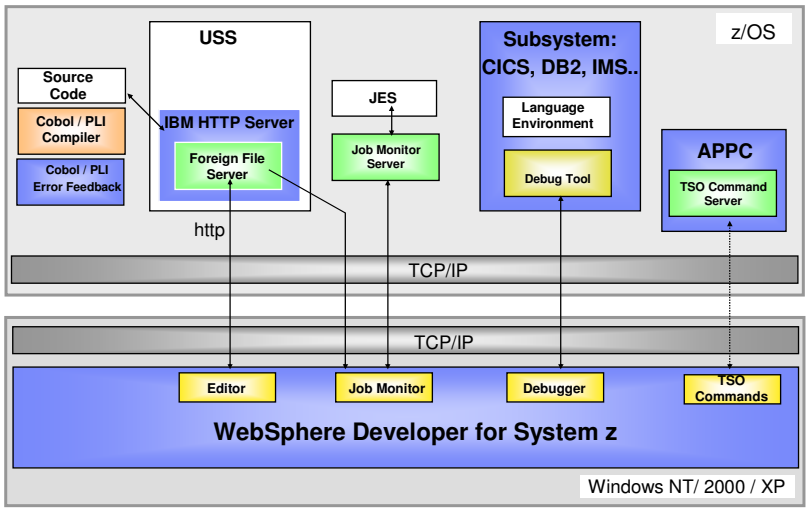
Development tools for Java Batch: 1) Eclipse



Development tools for Java Batch: 2) WDz

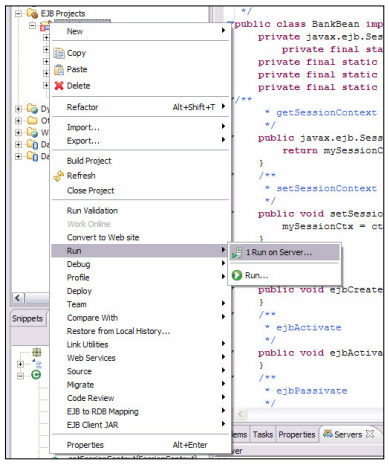
- **Based on Eclipse**
- **Inherits Eclipse functions plus:**
  - **JES integration**
  - **J2EE and Web programming**
  - **UML modeling**
  - **Cobol, PL/I and C/C++ development**
  - **CICS Web Service Support**
  - **Web Service Test Client Generation**
  - **IBM Debug Tool integration**
  - **…**

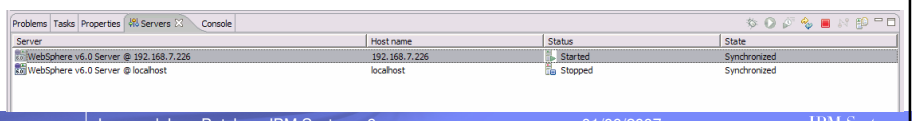
Development tools for Java Batch: 2) WDz



WDz and J2EE development

- **WDz is put on top of Rational Application Developer**
- **Integrated WebSphere Application Server test environment**
- **Remote deployment of applications**
- **Wizards for EJB creation**
- **EJB Test client**





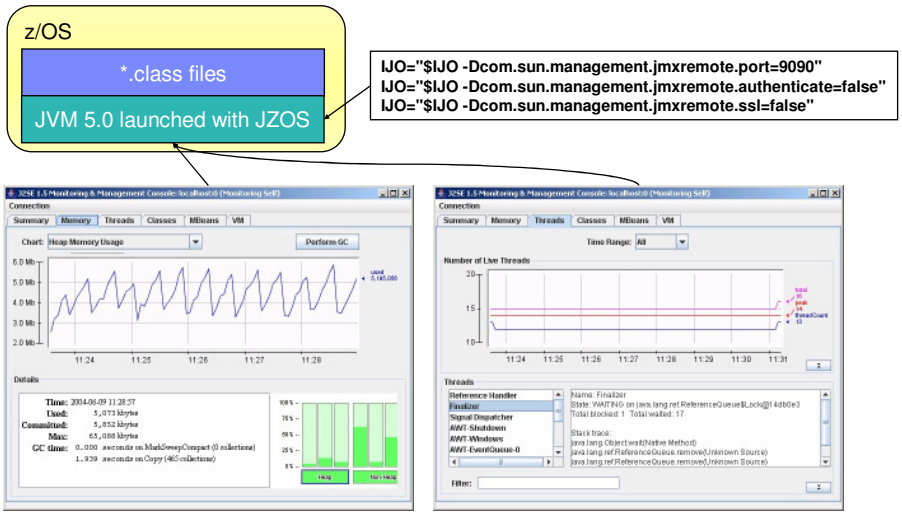
WDz and J2EE / Web development

- Web Development:
  - JSF support
  - Web site designer
  - JSP and servlet wizard
  - Java Visual Editor for JSF Component Layout
  - Portlet developemt
- Web Service development
  - Web Services Wizard
  - Web Services Explorer
  - Generation of Cobol Web Services

Resource access and backend integration

- System resources like the security facility RACF can be accessed via easy Java APIs
- The mainframe provides special, well approved transaction monitors (CICS and IMS) that can easily be integrated into Java applications
- Local connectors, that use cross-memory functions allow high transactional performance computing
  - Example: Local JDBC Driver for DB2 z/OS

Monitoring Java applications with JMX



Summary

- **The mainframe is the ultimate server for enterprise and mission critical applications**
- **There are more or less no differences in the development of Java applications between distributed servers and mainframes**

Literature

- **IBM z/OS Java Homepage**
  http://www-03.ibm.com/servers/eserver/zseries/software/java/
- **Redbooks:**
  - **Java Stand-alone Applications on z/OS, Volume I**
    http://www.redbooks.ibm.com/abstracts/sg247177.html
  - **Java Stand-alone Applications on z/OS, Volume II**
    http://www.redbooks.ibm.com/abstracts/sg247291.html?Open

- **IBM JVM 5.0 Diagnostics Guide**
  http://download.boulder.ibm.com/ibmdl/pub/software/dw/jdk/diagnosis/diag50.pdf
- **IBM z/OS Java SDK 5.0 User Guide**
  http://www-03.ibm.com/servers/eserver/zseries/software/java/pdf/sdkguide.zos.pdf

Free z/OS Java Workshops in Germany

- **From zero to zHero: System z optimized Java applications**
  - Target audience: developers

Stuttgart	München	Düsseldorf	Frankfurt	Hamburg
~~23.04-25.04~~	~~07.05-09.05~~	~~25.06-27.06~~	16.07-18.07	20.08-22.08

- **From zero to zHero: System z modernized applications**
  - Target audience: developers

Stuttgart	München	Düsseldorf	Frankfurt	Hamburg
~~26.04-27.04~~	~~10.05-11.05~~	~~28.06-29.06~~	19.07-20.07	23.08-24.08

- **http://www.ibm.com/de/events/fromzerotozhero**

Questions