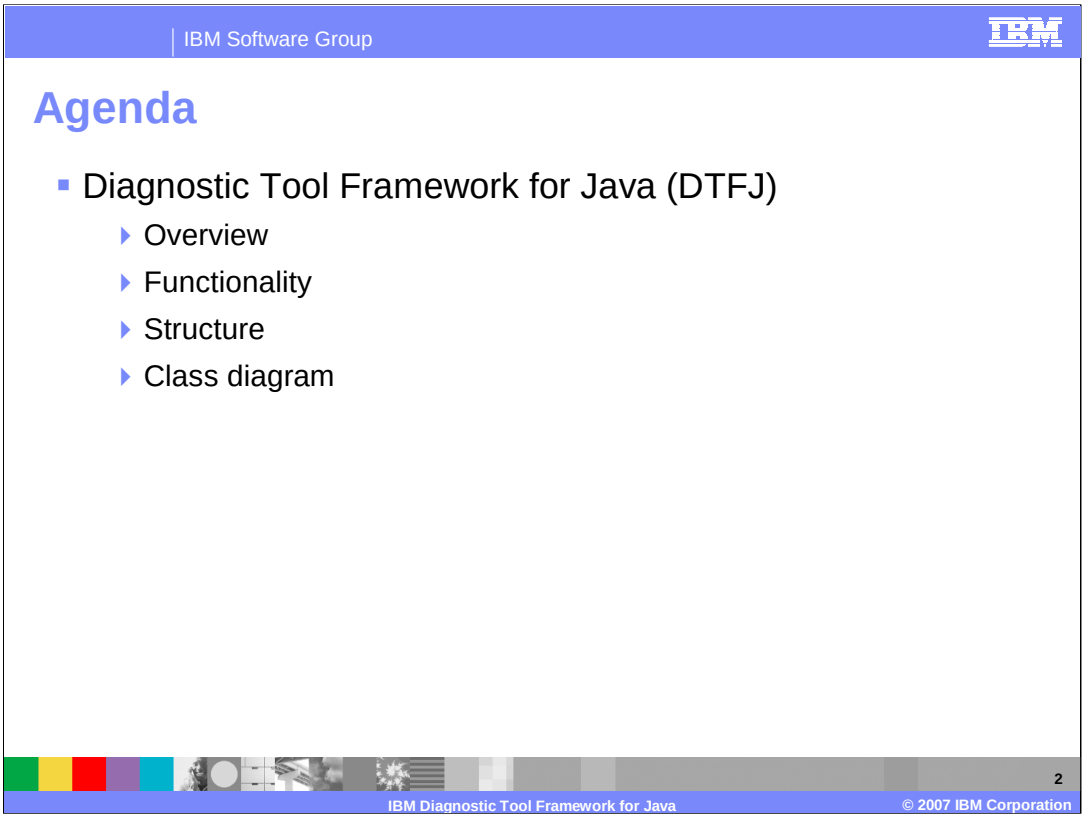

This presentation is about the IBM Diagnostic Tool Framework for Java.



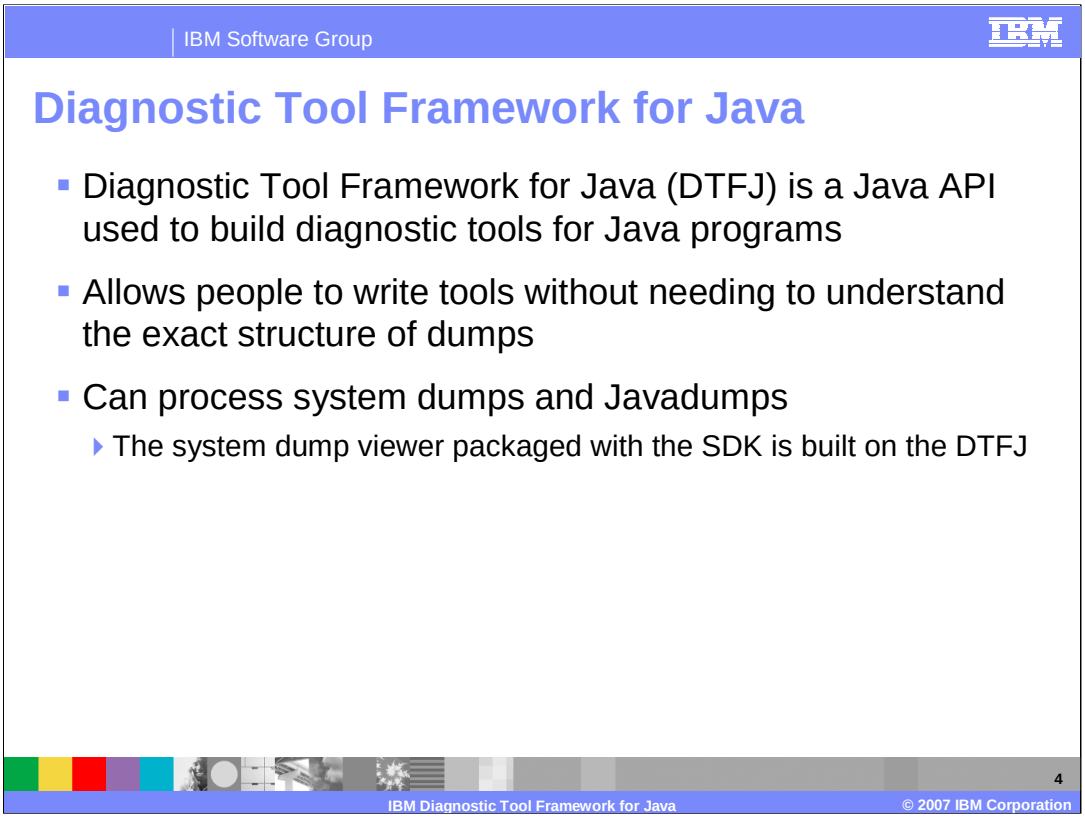

The Diagnostic Tool Framework for Java, often called the DTFJ, is a Java application programming interface for IBM used to support the building of Java diagnostic tools. This presentation will provide an overview of the DTFJ, the data that's available using the API, and how the APIs are structured.

## Section

# *Diagnostic Tool Framework for Java*




This section provides an overview of the Diagnostic Tool Framework for Java, including changes introduced in the Java 6 release.




The Diagnostic Tool Framework for Java acts as a layer of abstraction between a tool developer and the underlying structure of diagnostic data in the virtual machine. The DTFJ APIs allow Java tool developers to access data in a dump – like the Java version, threads, and heap data – without needing to understand the exact structure of the dump itself. DTFJ is implemented in pure Java and tools written using DTFJ can be cross-platform. Therefore, it is possible to analyze a dump taken from one machine on another (remote and more convenient) machine. To work with a system dump, the dump must first be processed by the dump extractor, jextract. The jextract tool produces metadata from the dump, which allows the internal structure of the JVM™ to be analyzed. It is recommended that you run jextract on the system that produced the dump; if that is not possible, you can use a system that is running the same operating system and virtual machine level as the system that produced the dump. In Version 6, DTFJ support has been added for Javadumps. To work with a Javadump, no additional processing is required.

## DTFJ functionality

- The DTFJ API helps diagnostic tools access this information:
  - Memory locations stored in the dump
  - Relationships between memory locations and Java internals
  - Java threads running within the JVM
  - Native threads held in the dump
  - Java classes and objects that were present in the heap
  - Details of the machine on which the dump was produced
  - Details of the java version that was being used
  - The command line that launched the JVM



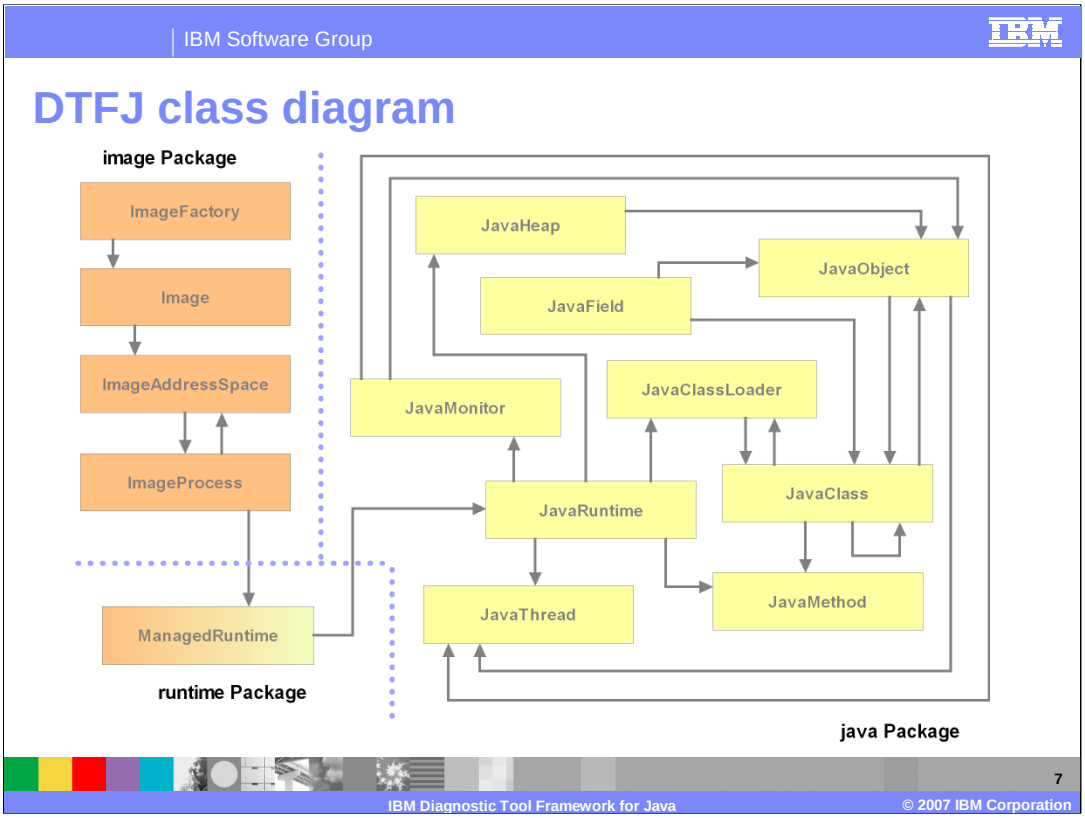

Some of the data that's accessible with the Diagnostic Tool Framework for Java is available in both Javadumps and system dumps; other data is only available in system dumps. This slide shows which data is available in which dumps. If your DTFJ application requests information that is not available in the Javadump, the API will return null or throw a DataUnavailable exception. You might need to adapt DTFJ applications written to process system dumps to make them work with Javadumps

## DTFJ structure

- The DTFJ interface is separated into two parts
  - Classes with names that start with **image** (com.ibm.dtfj.image)
  - Classes with names that start with **Java** (com.ibm.dtfj.java)
  - **Image** and Java classes are linked using a **JavaRuntime** object (com.ibm.dtfj.runtime)
- Generally, a DTFJ program will start by using an ImageFactory to create an image based on a dump file
  - Then traverse down the class hierarchy to eventually reach the JavaRuntime
- Examples can be found in the diagnostics guide




To create applications that use DTFJ, you must use the DTFJ interface. Implementations of this interface have been written that work with system dumps from IBM SDK for Java versions 1.4.2, 5.0, and 6, and Javadumps from IBM SDK for Java 6. The full details of the DTFJ Interface are provided with the SDK as Javadoc in the docs/content/apidoc directory. DTFJ classes are accessible without modification to the class path. The starting point for working with a dump is to obtain an Image instance by using the ImageFactory class supplied with the concrete implementation of the API. Initial setup – opening the dump file and preparing to access useful information stored in the dump – is done using classes in the com.ibm.dtfj.image package. After you have obtained an Image instance, you can begin analyzing the dump.  Examples of how to write DTFJ applications are available in the Diagnostics Guide.




This slide contains a partial class diagram of the Diagnostic Tool Framework for Java. A DTFJ application would start in the upper left of the diagram with the ImageFactory class. You will need to create your ImageFactory based on different classes depending on whether your application is going to process system dumps or Java dumps. For processing system dumps, the DTFJ expects that you will provide the zip archive from the dump extractor as input to the application; Javadump files do not require any pre-processing to be used as input to the tool framework. Once you have created the appropriate ImageFactory, you need to traverse down the class hierarchy and pull out the ImageProcess. From there, you can begin programmatically accessing the components of your Java runtime environment. The majority of DTFJ applications will follow this structure.

## Section

***Summary and references***




This section contains a summary and references.

## Summary

- Diagnostic Tool Framework for Java is a Java API used to build diagnostic tools for Java programs
- Supports system dumps and Javadumps
- Sample applications are available in the diagnostics guide




The Diagnostic Tool Framework for Java is a set of APIs that allow you to write Java applications to process system dump and Javadump files without needing to understand the exact layout of those files. The data that's available in your DTFJ application will vary depending on the type of dump that you are processing. Many DTFJ applications will have the same structure, and fully functioning examples are available in the Diagnostics Guide.

# References

- Diagnostics Guide
  - http://publib.boulder.ibm.com/infocenter/javasdk/v6r0/index.jsp

# Feedback

## Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_Java6_DTFJ.ppt

This module is also available in PDF format at: ../Java6_DTFJ.pdf




You can help improve the quality of IBM Education Assistant content by providing feedback.

# Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM

Java, JVM, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.



Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.