The NetBeans E-commerce Tutorial - Introduction

Tutorial Contents

Welcome to the NetBeans E-commerce Tutorial. In this multi-part tutorial, you learn how to create a simple yet effective e-commerce application that demonstrates various important features of Java web and EE development. In doing so, you'll familiarize yourself with the NetBeans IDE and become capable of applying it to your own development purposes.

Taking the time to master the IDE will ultimately lead you to become more efficient and versatile as a developer. While you work through the tutorial units, you'll learn how to make best use of the IDE's facilities and tools. These include:

- **Editor support for different languages:** syntax highlighting, code completion, API documentation support, keyboard shortcuts, refactoring capabilities, and code templates
- **Window system:** Projects, Files and Services windows, the Tasks window, Javadoc window, HTTP Monitor, Navigator and Palette
- **Integration with other services:** automatic deployment to a registered server, database connectivity, browser interoperability
- **Development tools:** Debugger, Profiler, HTTP Server Monitor, Local History support, and a graphical Diff Viewer

The tutorial is modular in fashion, with each unit focusing on specific concepts, technologies, and features of the IDE. You can successfully follow a tutorial unit on its own using the provided setup instructions and application snapshots (from Unit 5 onward). However, you'll get the most benefit by working through all units consecutively, from beginning to end. This will also help to illustrate the development process.

Unit 3, Setting up the Development Environment introduces you to the NetBeans IDE. In it, you create a Java web project which is the basis for the work you undertake in later tutorial units. In Unit 4, Designing the Data Model, you primarily work with MySQL WorkBench, a visual database design tool, to create a data model for the application. Each successive tutorial unit provides you with a *project snapshot* that corresponds to the project's beginning state for that given unit. This enables you to work through a single tutorial unit outside of the E-commerce Tutorial's larger context. To use these snapshots, download them to your computer and open them in the IDE using the Open Project wizard (Ctrl-Shift-O; ⌘-Shift-O on Mac).

You can view a live demo of the application that you build in this tutorial: NetBeans E-commerce Tutorial Demo Application.

The remainder of this unit covers some information relevant to the tutorial, as well as basic concepts necessary for Java EE development. Make sure you understand the concepts outlined below before proceeding with development.

About this Tutorial

Who this Tutorial is for

The content of this tutorial caters to four demographics:

- Java developers interested in expanding their skill set to include Java EE technologies
- Newcomers to the NetBeans IDE wanting to try out its development environment
- Web developers wanting to see how Java compares to other web-based technologies
- Students wanting to understand the nuts and bolts a simple e-commerce application, and how its development could apply to a real-world use-case

If you fall into any of these categories, this tutorial will be helpful to you. Depending on your background, you may find that certain tutorial units are more difficult to grasp than others. Understanding how technologies work is key to leveraging the IDE for your purposes. Therefore, if you are really interested in learning the technologies involved, you may find that this tutorial works best as a companion to the Java EE Tutorial. For each tutorial unit, make best use of the provided links to relevant areas in the Java EE Tutorial, as well as to other useful resources.

What this Tutorial Covers

The application that you develop in this tutorial involves numerous concepts, technologies, and tooling components:

- **Concepts**
  - Front-end development
  - Web application project structure
  - Data modeling
  - Database connectivity
  - Object-relational mapping
  - Session management
  - Transactional business logic
  - Client and server-side validation
  - Localization
  - Web application security
  - Design patterns, including Model-View-Controller (MVC) and Session Facade
- **Technologies**
  - HTML, CSS, and JavaScript technologies
  - Servlet and JavaServer Pages (JSP) technologies
  - Enterprise JavaBeans (EJB) technology
  - Java Persistence API (JPA)
  - The JavaServer Pages Standard Tag Library (JSTL)
  - Java Database Connectivity (JDBC)
- **Development Tools**
  - NetBeans IDE
  - GlassFish, a Java EE application server
  - MySQL, a relational database management server (RDBMS)
  - MySQL WorkBench, a visual database design tool

What is an E-commerce Application?

The term *e-commerce*, as we think of it today, refers to the buying and selling of goods or services over the Internet. For example, you may think of Amazon, which provides online shopping for various product categories, such as books, music, and electronics. This form of e-commerce is known as electronic retailing, or *e-tailing*, and usually involves the transportation of physical items. It is also referred to as *business-to-customer*, or B2C. Other well-known forms include:

- **Consumer-to-consumer (C2C):** Transactions taking place between individuals, usually through a third-party site such as an online auction. A typical example of C2C commerce is eBay.
- **Business-to-business (B2B):** Trade occurring between businesses, e.g., between a retailer and wholesaler, or between a wholesaler and manufacturer.
- **Business-to-government (B2G):** Trade occurring between businesses and government agencies.

This tutorial focuses on business-to-customer (B2C) e-commerce, and applies the typical scenario of a small retail store seeking to create a website enabling customers to shop online. Software that accommodates a B2C scenario generally consists of two components:

1. **Store Front:** The website that is accessed by customers, enabling them to purchase goods over the Internet. Data from the store catalog is typically maintained in a database, and pages requiring this data are generated dynamically.
2. **Administration Console:** A password-protected area that is accessed over a secure connection by store staff for purposes of online management. This typically involves CRUD (create read update delete) access to the store catalog, management of discounts, shipping and payment options, and review of customer orders.

What is Java?

In the computer software industry, the term "Java" refers to the *Java Platform* as well as the *Java Programming Language*.

Java as a Programming Language

The Java language was conceptualized by James Gosling, who began work on the project in 1991. The language was created with the following 5 design principles[1] in mind:

1. **Simple, Object-Oriented, and Familiar:** Java contains a small, consistent core of fundamental concepts that can be grasped quickly. It

was originally modeled after the then popular C++ language, so that programmers could easily migrate to Java. Also, it adheres to an *object-oriented* paradigm; systems are comprised of encapsulated objects that communicate by passing messages to one another.

2. **Robust and Secure:** The language includes compile-time and run-time checking to ensure that errors are identified quickly. It also contains network and file-access security features so that distributed applications are not compromised by intrusion or corruption.
3. **Architecture Neutral and Portable:** One of Java's primary advantages is its *portability*. Applications can be easily transferred from one platform to another with minimum or no modifications. The slogan "Write once, run anywhere" accompanied the Java 1.0 release in 1995, and refers to the cross-platform benefits of the language.
4. **High Performance:** Applications run quickly and efficiently due to various low-level features, such as enabling the Java interpreter to run independently from the run-time environment, and applying an automatic garbage collector to free unused memory.

Duke, the Java mascot

5. **Interpreted, Threaded, and Dynamic:** With Java, a developer's source code is compiled into an intermediate, interpreted form known as *bytecode*. The bytecode instructional set refers to the machine language used by the Java Virtual Machine (JVM). With a suitable interpreter, this language can then be translated into *native code* for the platform it is run on. Multithreading capabilities are supported primarily by means of the `Thread` class, enabling numerous tasks to occur simultaneously. The language and run-time system are dynamic in that applications can adapt to environment changes during execution.

If you'd like to learn more about the Java language, see the Java Tutorials.

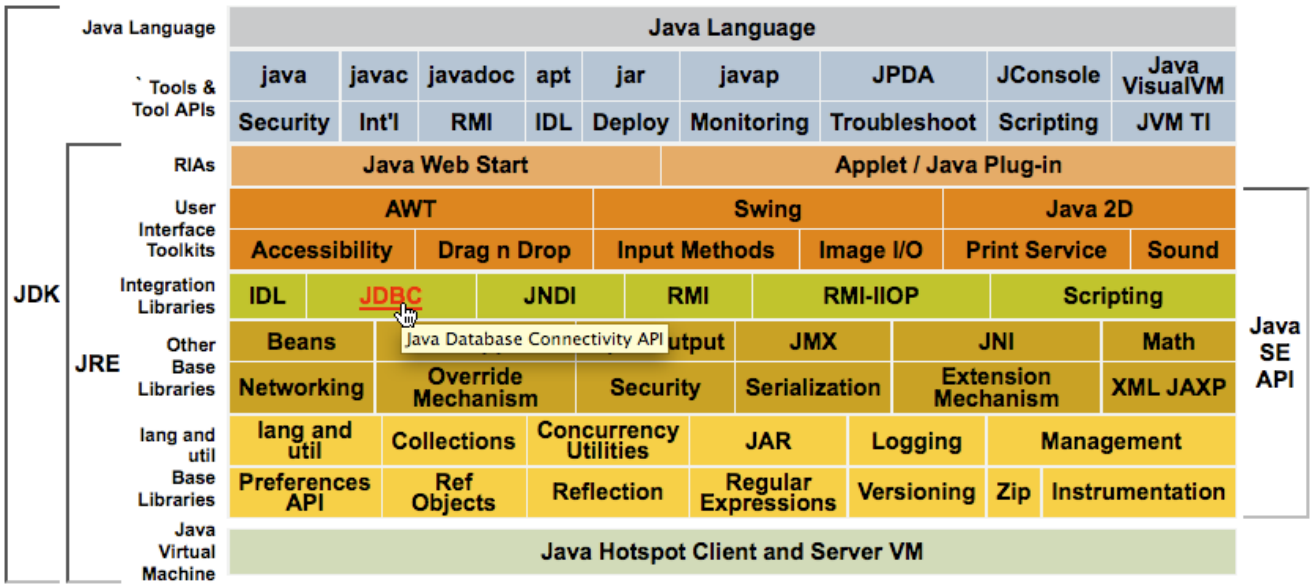
Java as a Platform

The Java Platform signifies a software-based platform that is comprised of two parts:

- **The Java Virtual Machine (JVM)**: The JVM is an engine that executes instructions generated by the Java compiler. The JVM can be thought of as an instance of the Java Runtime Environment, or JRE, and is embedded in various products, such as web browsers, servers, and operating systems.
- **The Java Application Programming Interface (API)**: Prewritten code, organized into packages of similar topics. For instance, the Applet and AWT packages include classes for creating fonts, menus, and buttons.

The Java Development Kit, or JDK, refers to the Java SE Edition, while other kits are referred to as "SDK", a generic term for "software development kit." For example, the Java EE SDK.[2]

You can see a visual representation of the Java platform by viewing the conceptual diagram of component technologies provided in the JDK Documentation. As shown below, the diagram is interactive, enabling you click on components to learn more about individual technologies.



As the diagram indicates, the JDK includes the Java Runtime Environment (JRE). You require the JRE to run software, and you require the JDK to develop software. Both can be acquired from Java SE Downloads.

The Java platform comes in several *editions*, such as Java SE (Standard Edition), Java ME (Micro Edition), and Java EE (Enterprise Edition).

Java EE

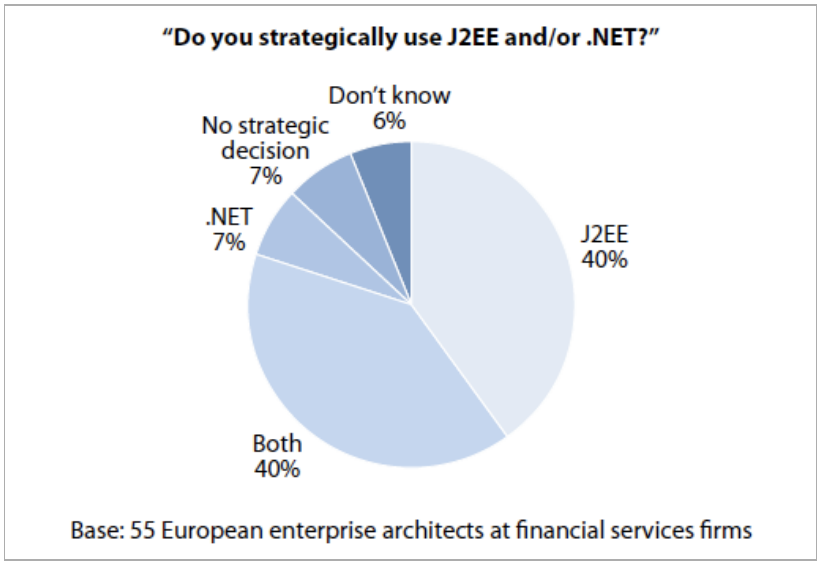
The Java Platform, Enterprise Edition (Java EE) builds upon the Java SE platform and provides a set of technologies for developing and running portable, robust, scalable, reliable and secure server-side applications.

EE technologies are loosely divided into two categories:

- Web application technologies
- Enterprise application technologies

Depending on your needs, you may want to use certain technologies from either category. For example, this tutorial makes use of Servlet, JSP/EL, and JSTL "web" technologies, as well as EJB and JPA "enterprise" technologies.

Java EE currently dominates the market, especially in the financial sector. The following diagram is taken from an independent survey for European markets performed in 2007.



For a recent, informal comparison of Java EE to .NET, see the blog post Java EE or .NET - An Almost Unbiased Opinion by a well-known member of the Java EE community.

What's the Difference Between...?

There are many abbreviations and acronyms to parse. If you're new to all of this and find the above explanation somewhat confusing, the following resources can help explain what the differences are between some of the commonly used terminology.

- What's the Difference between the JRE and the JDK?
- What's the Difference between the JRE and the Java SE platform?
- What's the Difference between Java EE and J2EE?
- Unraveling Java Terminology

What is the Java Community Process?

The Java Community Process (JCP) is a program that manages the development of standard technical specifications for Java technology. The JCP catalogs Java Specification Requests (JSRs), which are formal proposals that document the technologies which are to be added to the Java platform. JSRs are run by an *Expert Group*, which typically comprises representatives of companies that are stakeholders in the industry. The JCP enables Java technology to grow and adapt according to the needs and trends of the community.

The JSRs of technologies used and referred to in this tutorial include the following:

- JSR 52: A Standard Tag Library for JavaServer Pages
- JSR 245: JavaServer Pages 2.1
- JSR 315: Java Servlet 3.0
- JSR 316: Java Platform, Enterprise Edition 6
- JSR 317: Java Persistence 2.0
- JSR 318: Enterprise JavaBeans 3.1

You can use the JCP website to search for individual JSRs. You can also view all current EE technologies (Java EE 6) at:

- http://java.sun.com/javaee/technologies/index.jsp

Java EE 5 technologies are listed at:

- http://java.sun.com/javaee/technologies/javaee5.jsp

A JSR's final release provides a *reference implementation*, which is a free implementation of the technology. In this tutorial, you utilize these implementations to develop the sample e-commerce application. For example, the GlassFish v3 application server, which is included in the standard Java download bundle for NetBeans 6.8, is the reference implementation of the Java EE 6 platform specification (JSR 316). As a reference implementation for the Java EE platform, it includes reference implementations for the technologies included in the platform, such as Servlet, EJB and JPA technologies.

Why use an IDE?

Firstly, the term *IDE* stands for *integrated development environment*. The purpose of an IDE has traditionally been to maximize a developer's productivity by providing tools and support such as:

- a source code editor
- a compiler and build automation tools
- a window system for viewing projects and project artifacts
- integration with other commonly-used services
- debugging support
- profiling support

Consider what would be necessary if you wanted to create a Java-based web application manually. After installing the Java Development Kit (JDK), you would need to set up your development environment by performing the following steps.[3]

1. Set your `PATH` environment variable to point to the JDK installation.
2. Download and configure a server that implements the technologies you plan to use.
3. Create a development directory where you plan to create and work on the web application(s). Furthermore, you are responsible for setting up the application directory structure so that it can be understood by the server. (For example, see Java BluePrints: Strategy for Web Applications for a recommended structure.)
4. Set your `CLASSPATH` environment variable to include the development directory, as well as any required JAR files.
5. Establish a deployment method, i.e., a way to copy resources from your development directory to the server's deployment area.
6. Bookmark or install relevant API documentation.

For educative purposes, it is worthwhile to create and run a Java web project manually so that you are aware the necessary steps involved. But eventually, you'll want to consider using tools that reduce or eliminate the need to perform tedious or repetitious tasks, thereby enabling you to focus on developing code that solves specific business needs. An IDE streamlines the process outlined above. As demonstrated in Unit 3, Setting up the Development Environment, you'll install NetBeans IDE with the GlassFish application server, and be able to set up a web application project with a conventional directory structure using a simple 3-step wizard. Furthermore, the IDE provides provides built-in API documentation which you can either call up as you code in the editor, or maintain open in an external window.

An IDE also typically handles project compilation and deployment in a way that is transparent to you as a developer. For example, the web project that you create in NetBeans includes an Ant build script that is used to compile, clean, package and deploy the project. This means that you can run your project from the IDE, and it will automatically be compiled and deployed, then open in your default browser. Taking this a step further, many IDEs support a Deploy on on Save feature. In other words, whenever you save changes to your project, the deployed version on your server is automatically updated. You can simply switch to the browser and refresh the page to view changes.

IDEs also provide templates for various file types, and often enable you to add them to your project by suggesting common locations and including default configuration information where necessary.

Aside from the "basic support" described above, IDEs typically provide interfaces to external tools and services (e.g., application and database servers, web services, debugging and profiling facilities, and collaboration tools) which are indispensable to your work if Java development is your profession.

Finally, IDEs usually provide enhanced editor support. The editor is where you likely spend most of your time working, and IDE editors typically include syntax highlighting, refactoring capabilites, keyboard shortcuts, code completion, hints and error messages, all aiming to help you work more efficiently and intelligently.

Why use NetBeans?

The NetBeans IDE is a free, open-source integrated development environment written entirely in Java. It offers a range of tools for create professional desktop, enterprise, web, and mobile applications with the Java language, C/C++, and even scripting languages such as PHP, JavaScript, Groovy, and Ruby.

People are saying great things about NetBeans. For a list of testimonials, see NetBeans IDE Testimonials. Many developers are migrating their applications to NetBeans from other IDEs. For reasons why, read Real Stories From People Switching to NetBeans IDE.

The IDE provides many features for web development, and several advantages over other IDEs. Here are several noteworthy points:

- **Works Out of the Box:** Simply download, install, and run the IDE. With its small download size, installation is a breeze. The IDE runs on many platforms including Windows, Linux, Mac OS X and Solaris. All IDE tools and features are fully integrated - no need to hunt for plug-ins - and they work together when you launch the IDE.
- **Free and Open Source:** When you use the NetBeans IDE, you join a vibrant, open source community with thousands of users ready to help and contribute. There are discussions on the NetBeans project mailing lists, blogs on Planet NetBeans, and helpful FAQs and tutorials on the community wiki.
- **Profiling and Debugging Tools:** With NetBeans IDE profiler, you get real time insight into memory usage and potential performance bottlenecks. Furthermore, you can instrument specific parts of code to avoid performance degradation during profiling. The HeapWalker tool helps you evaluate Java heap contents and find memory leaks.
- **Customizable Projects:** Through the NetBeans IDE build process, which relies on industry standards such as Apache Ant, make, Maven, and rake - rather than a proprietary build process - you can easily customize projects and add functionality. You can build, run, and deploy projects to servers outside of the IDE.
- **Collaboration Tools:** The IDE provides built-in support for version control systems such as CVS, Subversion, and Mercurial.
- **Extensive Documentation:** There's a wealth of tips and instructions contained in the IDE's built-in help set. Simply press F1 (fn-F1 on Mac) on a component in the IDE to invoke the help set. Also, the IDE's official knowledge base provides hundreds of online tutorials, articles and screencasts that are continuously being updated.

For a more extensive list of reasons why you should consider choosing NetBeans, see NetBeans IDE Connects Developers.

Send Us Your Feedback

See Also

Online Resources

- The Java Tutorials
- Java EE FAQ
- Java EE APIs & Docs
- Unraveling Java Terminology
- The History of Java Technology
- New to Java Programming Center

Books

- Pro NetBeans IDE 6 Rich Client Platform Edition
- Core Servlets and JavaServer Pages, Volume 1: Core Technologies, 2nd Edition
- Core Servlets and JavaServer Pages, Volume 2: Advanced Technologies, 2nd Edition
- The Java FAQ

References

1. ^ The white paper, The Java Language Environment, outlines the 5 design principles.
2. ^ Current version names and numbers are defined in Java SE 6, Platform Name and Version Numbers.
3. ^ These steps are loosely based on those outlined in Chapter 2: Server Setup and Configuration, from Core Servlets and JavaServer Pages, by Marty Hall and Larry Brown. This book is freely available in PDF format from: http://pdf.coreservlets.com/