



# Porting ASP.NET Applications to the J2EE™ Platform

*A case study that demonstrates the ease of porting the  
Commerce Starter Kit application to J2EE*

**January 2005**  
**[www.mainsoft.com](http://www.mainsoft.com)**

# Porting .NET Applications to the J2EE Platform

*A case study that demonstrates the ease of porting the Commerce Starter Kit application to J2EE*

<b>INTRODUCTION .....</b>	<b>3</b>
<b>VISUAL MAINWIN OVERVIEW .....</b>	<b>4</b>
BUILDING A VISUAL MAINWIN J2EE APPLICATION .....	4
DEBUGGING A VISUAL MAINWIN J2EE APPLICATION .....	5
RUNTIME LIBRARIES .....	5
<b>PORTING .NET APPLICATIONS TO J2EE .....</b>	<b>5</b>
<b>PORTING THE ASP.NET COMMERCE STARTER KIT TO J2EE.....</b>	<b>5</b>
WHAT IS THE COMMERCE STARTER KIT APPLICATION?.....	5
PORTING PROCESS.....	6
Step 1: Configure the porting machine .....	6
Step 2: Set up the Commerce Starter Kit.....	7
Step 3: Run the Commerce Starter Kit application.....	7
Step 4: Port the Commerce Starter Kit application to J2EE .....	9
PRODUCTIVITY BENCHMARKS.....	15
PERFORMANCE .....	15
<b>PORTING CHALLENGES.....</b>	<b>15</b>
THIRD-PARTY LIBRARIES .....	16
CODE MODIFICATIONS .....	16
<b>SUCCESS THROUGH PARTNERSHIP.....</b>	<b>17</b>
FIXED-COST, FIXED-TIME APPROACH FOR PORTING.....	17
THE MAINSOFT PORTING METHODOLOGY .....	18
Step 1: Requirement Analysis .....	18
Step 2: Detailed Project Plan .....	18
Step 3: Implementation.....	18
Step 4: On-Site Project Delivery .....	19
CONCLUSION.....	19
<b>ABOUT MAINSOFT .....</b>	<b>19</b>

## Introduction

Many developers favor the Visual Studio .NET® development environment to write their applications because it is easy to use and highly productive. However, due to changing business requirements, companies often choose to deploy business-critical applications on the J2EE platform.

The challenge facing Independent Software Vendors (ISVs) and IT organizations is to capitalize on the productivity of .NET without sacrificing the scalability and flexibility of J2EE servers. Until now, conflicting software standards have required these companies to invest in overlapping resources and skills, driving up their development and maintenance costs.

Many ISVs and IT organizations would welcome a cost-effective solution to deploy their .NET Web applications and Web services on J2EE, without having to rewrite their entire applications in Java. ISVs want to offer their products on both .NET and J2EE, without making significant investments in resources and skills. Enterprises face a similar challenge when they opt to deploy existing .NET enterprise applications and Web services on J2EE servers.

Mainsoft developed Visual MainWin® for the J2EE platform to resolve the development and deployment challenges arising from the .NET/J2EE divide. Visual MainWin enables Visual Studio developers to port .NET Web applications and Web services to the J2EE platform from Visual Studio, dramatically reducing time-to-market and development costs. With the Commerce Starter Kit case study, we demonstrate the ease of porting .NET applications written in C# or Visual Basic.NET® to the J2EE platform.

For the past decade, Mainsoft has helped many of the world's largest ISVs, such as Computer Associates, IBM Rationale, and Siebel Systems, to deploy enterprise-class applications on non-Microsoft® platforms on time and on budget, slashing development costs and time-to-market by as much as 75 percent.

## Visual MainWin Overview

Visual MainWin is a complete development solution that includes:

- A development environment that is fully integrated into Visual Studio .NET. Visual MainWin inherits all the productivity features that make Visual Studio such a popular development tool and can be used by .NET developers to develop, deploy, and debug J2EE applications, without additional training.
- A patent-pending compiler that compiles Microsoft Intermediate Language (MSIL) into standard Java bytecode.
- A set of .NET runtime extensions to J2EE servers that enable .NET applications to run on any standard J2EE application server, such as IBM WebSphere®, BEA WebLogic®, JBoss®, or Tomcat.

### Building a Visual MainWin J2EE Application

Developers can write programs in either C# or Visual Basic.NET and compile their source code directly into standard Java bytecode. Mainsoft's binary compiler is integrated seamlessly into the build process. When the user builds a project, Visual MainWin automatically compiles the MSIL assemblies and generates Java bytecode. The Java class files are then packed in standard JAR or WAR files and deployed transparently on the J2EE application server.

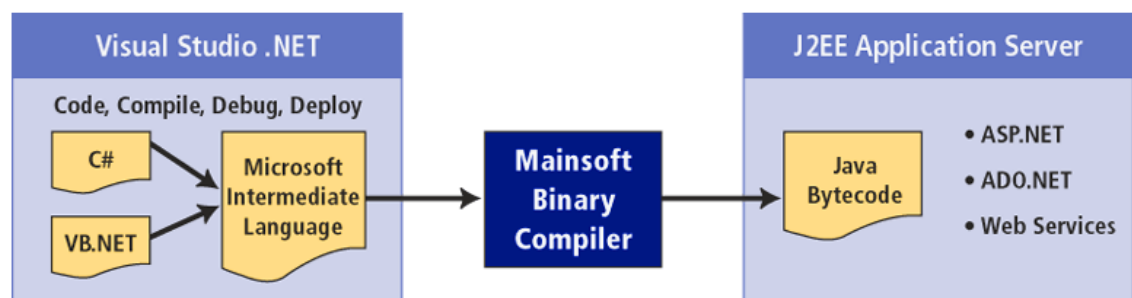


Figure 1 - Binary compilation from MS Intermediate Language to Java bytecode

Visual MainWin provides new types of C# and Visual Basic.NET projects, including templates for an ASP.NET Web Application, an ASP.NET Web Service, a Web Control Library, a Class Library, and a Console Application. When building any one of these projects, a J2EE application is created.

## Debugging a Visual MainWin J2EE Application

Visual MainWin's patent-pending debugger enables C# and Visual Basic.NET developers to debug applications running on a Java Virtual Machine (JVM) within Visual Studio .NET, without seeing the actual runtime Java symbols. The debugger connects to the JVM where the application is running and sends debugging commands such as breakpoints, steps, and continues. Data received from the debugger is translated back into the original source language and displayed in the Visual Studio debugger user interface.

## Runtime Libraries

Visual MainWin provides a Java implementation of the ASP.NET and ADO.NET class libraries. This implementation of .NET class libraries is based on the [Mono](#) project, a high-quality open source implementation of the .NET Framework. Visual MainWin provides a robust and scalable implementation of ASP.NET and ADO.NET for J2EE by relying on the rich functionality of the underlying JDK class libraries.

## Porting .NET Applications to J2EE

Visual MainWin dramatically simplifies the porting process. Whereas porting typically requires a proficiency with several tool sets, Visual MainWin enables a .NET development team to port Web applications or Web services from within Visual Studio .NET. The Visual MainWin project conversion tool guides the developer through the process of converting existing ASP.NET projects to Visual MainWin for J2EE projects. Existing .NET applications—Web applications, Web services, class libraries, and console applications—can be re-built and deployed on a J2EE application server from Visual Studio .NET. And because Visual MainWin provides a complete development environment, the ported J2EE application can be maintained without a Java IDE.

## Porting the ASP.NET Commerce Starter Kit to J2EE

### What is the Commerce Starter Kit Application?

The Commerce Starter Kit, previously known as IBuySpy, is an application publicly available at the [ASP.NET Web site](#). This application has many of the elements of a typical online storefront, including a product catalog, user authentication and personalization, shopping carts, and a Web service to submit orders.

Microsoft's Commerce Starter Kit was built as a two-tier application, including a presentation layer, and a business logic layer (BLL) combined with a data access layer (DAL). The presentation layer refers to the Web application pages, the BLL refers to the component that encapsulates all the business logic of the application, and the DAL refers to the database itself, the stored procedures, and the component that provides an interface to the database. Refer to the documentation on the ASP.NET site for further details on the architecture, design, implementation, and use of the .NET runtime.

The ASP.NET site provides several versions of the Commerce Starter Kit. The Visual MainWin platform fully supports the C# and Visual Basic.NET versions. In this case study, we focus on the C# version.

The sample incorporates many of the Web application features provided by the .NET Framework, including:

- Cross-browser support for Netscape and Internet Explorer.
- Clean code/HTML content separation using server controls and code-behind.
- High performance catalog pages that use output caching.
- ADO.NET data access using SQL.
- Forms authentication using a database for usernames/passwords.
- SOAP XML based ASP.NET Web services for B2B order entry and status.

The Commerce Starter Kit case study provides a step-by-step porting example that you can easily reproduce to preview Mainsoft's newest porting solution.

## **Porting Process**

### Step 1: Configure the porting machine

1. Verify that you have the following programs installed on your machine:
  - Microsoft Windows XP Professional
  - Microsoft Visual Studio .NET 2003 (7.1)
  - Visual MainWin for the J2EE platform 1.5 or later

- Microsoft SQL Server 2000 or MSDE 2000. The free version of MSDE is available at <http://www.microsoft.com/sql/msde/downloads/download.asp>
2. Visual MainWin for the J2EE platform fully supports the following application servers:
    - IBM WebSphere Application Server Version 5.1 (J2EE application server)
    - BEA WebLogic Platform 8.1 (J2EE application server)
    - JBoss 3.2.x (J2EE application server)
    - Apache Tomcat 5.0

By default, a bundled Tomcat application server is installed with the Visual MainWin installation.

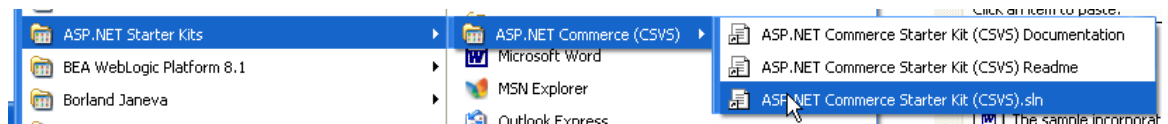
Since the Commerce Starter Kit can be implemented in a J2EE Web container and does not require the use of Enterprise Java Beans (EJBs), it can run on the Tomcat application server that comes bundled with Visual MainWin. In this case study, we use the Tomcat application server, and deploy the Commerce Starter Kit application on it. The procedure is similar for other supported J2EE application servers.

### Step 2: Set up the Commerce Starter Kit

1. Download the C# Visual Studio .NET Commerce Starter Kit installation from the ASP.NET Web site at the following URL: <http://www.asp.net/StarterKits/DownloadCommerce.aspx?tabindex=0&tabid=1>
2. Run the installation program and follow its instructions. The installation program will install the Commerce Starter Kit Visual Studio Solution and set up the application database on MS SQL Server. Remember the name of the local database on which you choose to install the Commerce database.

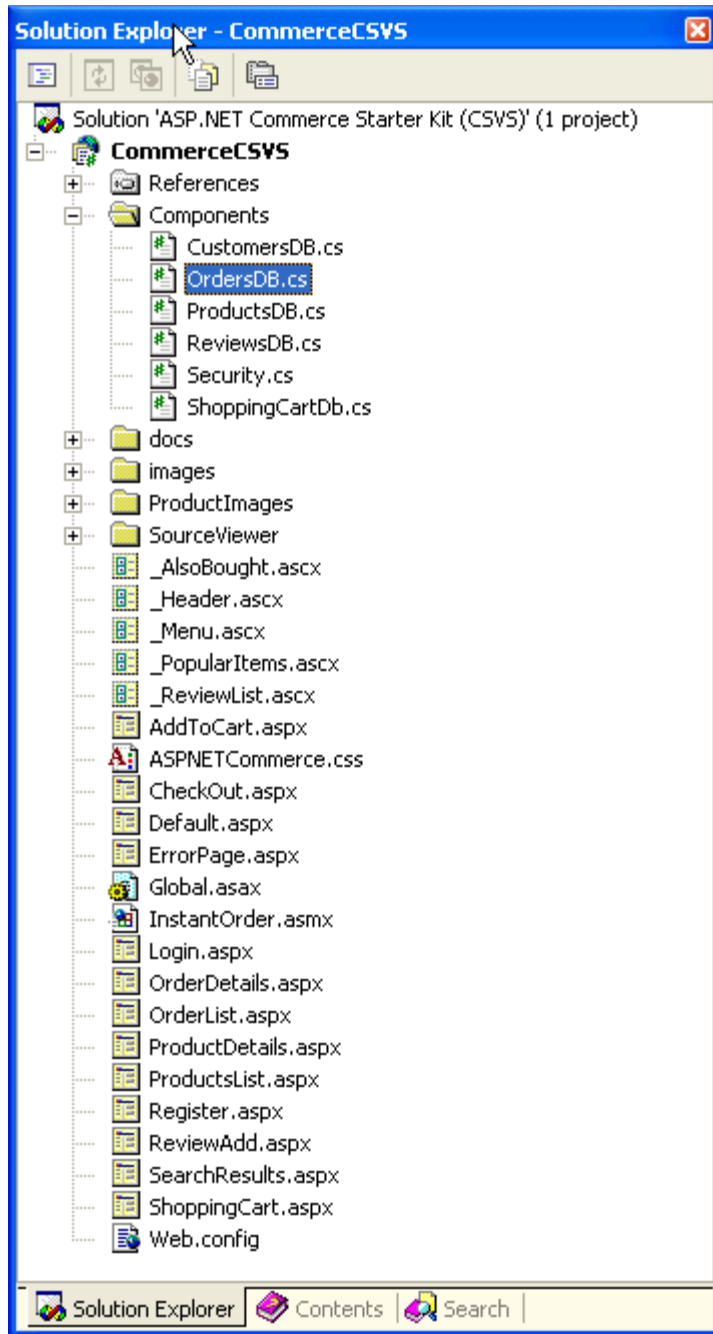
### Step 3: Run the Commerce Starter Kit application

1. Open the Commerce Starter Kit Visual Studio solution file (“ASP.NET Commerce Starter Kit (CSVs).sln”) from the Windows Start menu.

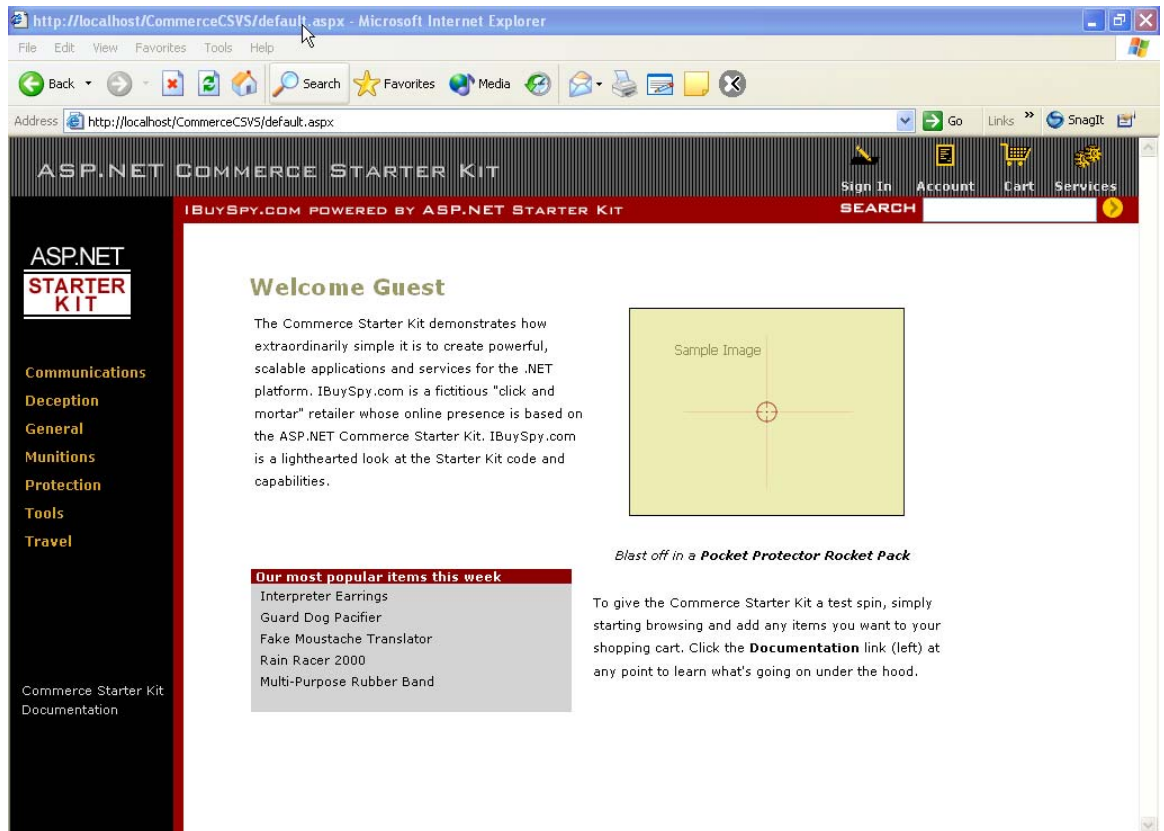


The project includes a large number of files, including HTML, style sheets, ASP.NET custom controls, and media files for the

user interface as well as a set of component classes that access the database using ADO.

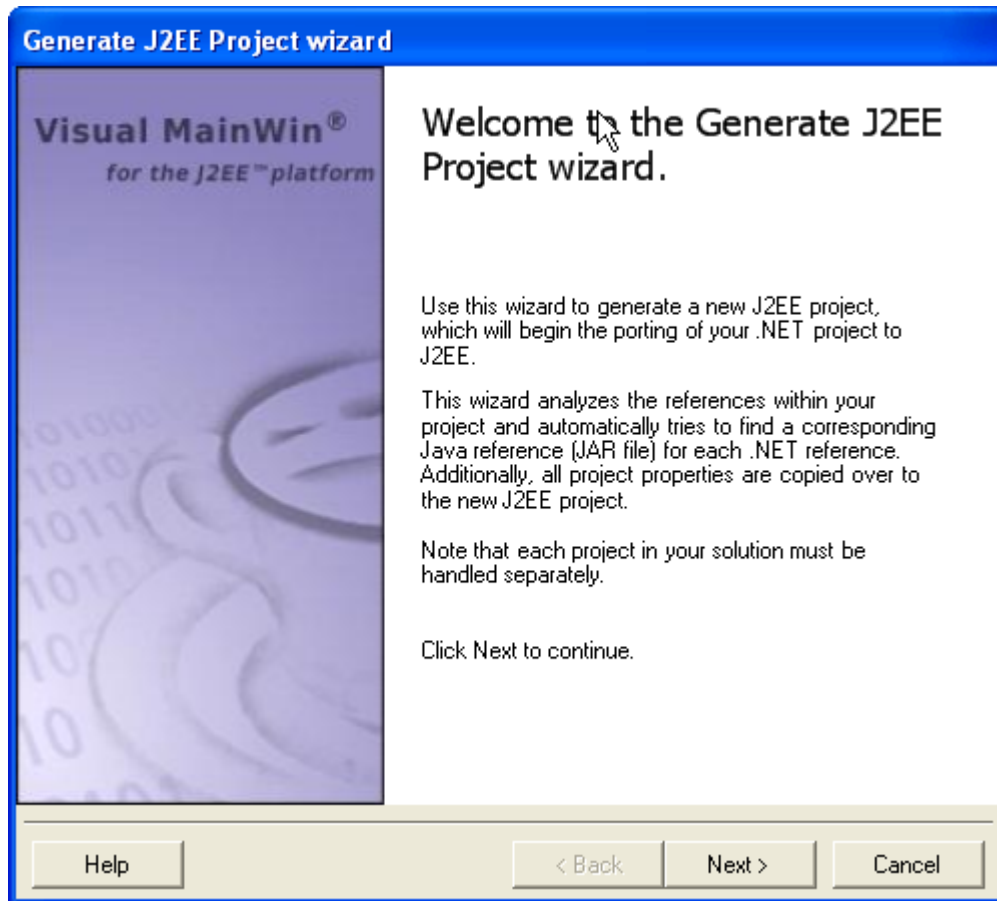


2. Run the Commerce Starter Kit .NET application using the URL <http://localhost/CommerceCSVS> in the Internet Explorer browser. The following screen shows the running application:



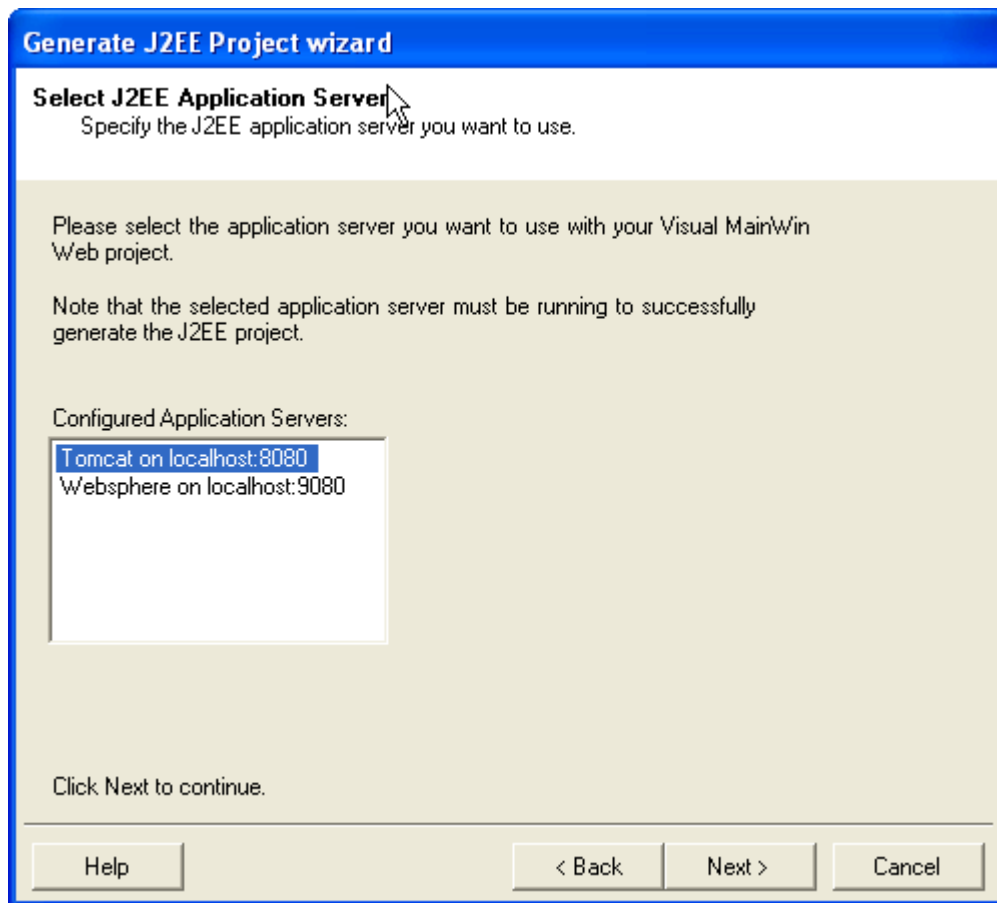
#### Step 4: Port the Commerce Starter Kit application to J2EE

1. Start the Tomcat application server, by choosing **All Programs** from the Windows Start menu, and then choosing **Visual MainWin for the J2EE™ platform > Start Tomcat**.
2. In the Solution Explorer, right-click the CommerceCSVS project and choose **Generate J2EE Project** from the shortcut menu.  
The Generate J2EE Project wizard opens.

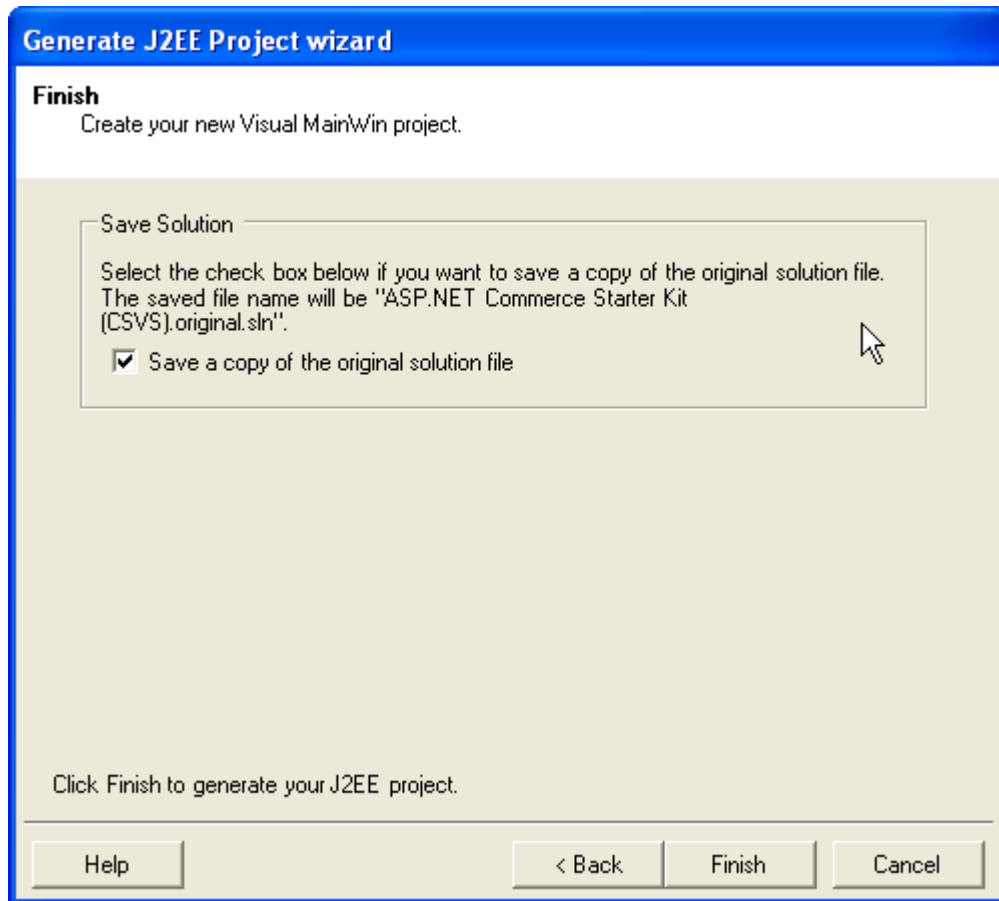


3. Click **Next** to continue.

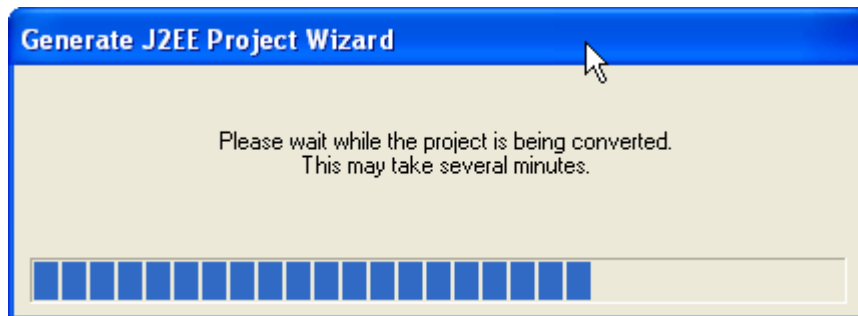
The Select J2EE Application Server screen of the wizard opens, listing the application servers currently available. To learn how to add an application server, refer to the product documentation. For this example, we will work with Tomcat 5.0.



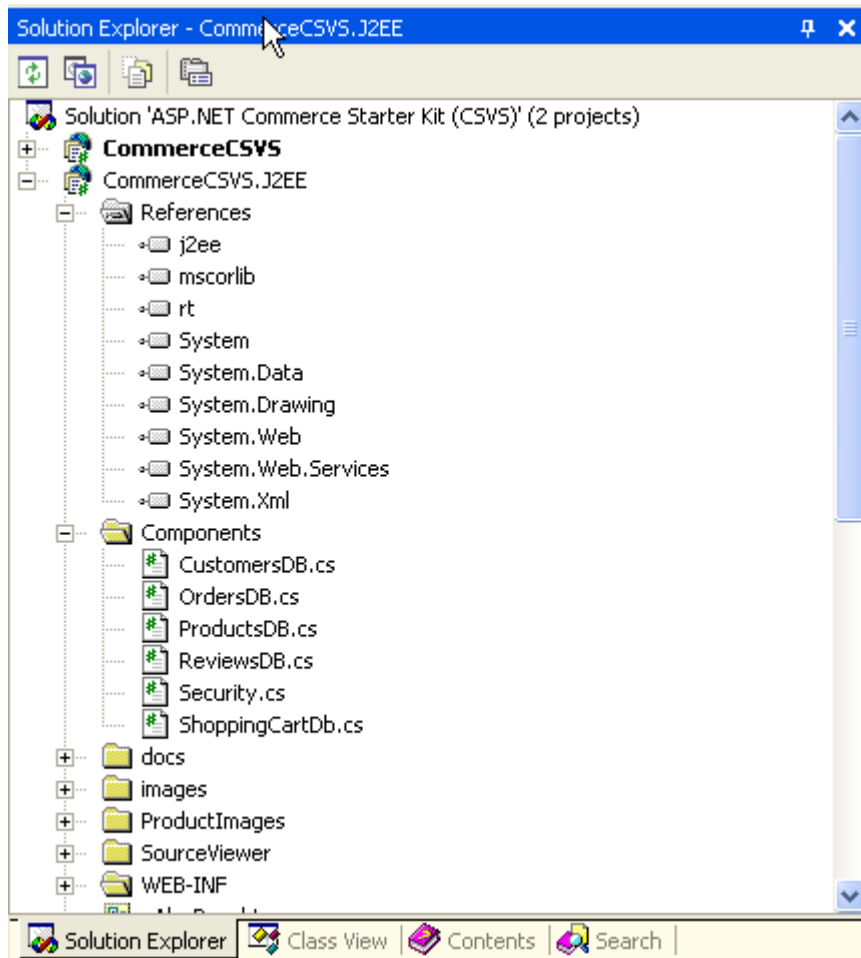
4. Select the Tomcat application server, and then click **Next**.  
The Finish screen of the wizard opens



5. Click **Finish** to generate the J2EE project.



When the wizard finishes generating the J2EE project, a new project called `CommerceCSVS.J2EE` is added to your solution. This is the project that is used for building the J2EE version of the Commerce Starter Kit. The `CommerceCSVS.J2EE` project is referencing the same set of source files referenced by the .NET `CommerceCSVS` project.



View the newly ported project in the Solution Explorer. You should notice these features of the `CommerceCSV5.J2EE` project:

- The project looks like a regular Visual Studio .NET solution.
  - The project has `Debug_Java` and `Release_Java` configurations. These configurations are used to build the Debug and Release configurations of the J2EE version of the Commerce Starter Kit application.
  - The **References** folder contains normal .NET components. However, when you click on these references, you will note in the **Properties** pane that these references have a new property, a JAR path. This path points to a Java implementation of the appropriate .NET runtime classes.
  - There is a new folder, named "WEB-INF" that contains the `web.xml` file, which contains the J2EE servlet container deployment descriptor file, generated by Visual MainWin.
6. Configure your database connection string, as follows:

- a. In the Solution Explorer, double-click the file `Web.config` located under the `CommerceCSVS.J2EE` project.
- b. Locate the following XML code block:

```
<appSettings>
  <add key="ConnectionString"
  value="server=localhost\<instance
  name>;Trusted_Connection=true;database=Store" />
</appSettings>
```

- c. Modify the `<add>` element as follows:

```
<add key="ConnectionString" value="data
source=localhost\<instance name>;user
id=<username>;password=<password>;Trusted
Connection=no;initial catalog=Commerce" />
```

where the following items must be changed to match your environment:

- `User id=<username>`: Set the username that is used to connect to the Commerce database on the SQL Server.
- `Password=<password>`: Set the password used to connect to the Commerce database on the SQL Server.

Here is an example of a connection string:

```
data source=localhost\VSdotNET;initial catalog=Commerce;user
id=sa;password=;Trusted_Connection=no
```

- d. Save and close `Web.config`.
7. In the Solution Explorer, right-click the `CommerceCSVS.J2EE` project and select **Set as StartUp Project**.
8. In the Solution Explorer, under the `CommerceCSVS.J2EE` project, right-click the file `Default.aspx` and select **Set As Start Page**.
9. Choose **Build > Build Solution** to build the project.

Inspect the build output. Notice that both the `CommerceCSVS` project and the `CommerceCSVS.J2EE` project were built. Make sure that the build was completed successfully. When the build completes, a J2EE WAR file is created and automatically deployed on the Tomcat application server.

10. Choose **Debug > Start Without Debugging** to run the Web application.

The home page of the Commerce Starter Kit application opens in your default Web browser. Note that the URL for the application is now

<http://localhost:8080/CommerceCSVS/Default.aspx>,

where 8080 is the HTTP port of the Tomcat application server. The Commerce Starter Kit application looks the same but is now running on a J2EE application server.

11. If you want to debug the Commerce Starter Kit application, make sure your application server is running in debug mode. (If you ran the Tomcat application server by running **QuickStart Tomcat** from **Visual MainWin for the J2EE™ platform**, it is already in debug mode).

Set breakpoints in the source code and choose **Debug > Start** to start debugging. When stopped in a breakpoint you may inspect the value of local variables, evaluate expressions, etc.

Congratulations, you have ported the Commerce Starter Kit solution to the J2EE platform!

### Productivity Benchmarks

When comparing between porting the Commerce Starter Kit to J2EE and rewriting the Commerce Starter Kit in Java, we assume that the same database is used on .NET and J2EE and that the design of the pages remains unchanged. The following table summarizes the productivity gains:

	Porting the Commerce Starter Kit	Rewriting the Commerce Starter Kit
Project duration <i>(including setup, porting, and deployment on the application server)</i>	Half a day or less	Two to four weeks <i>(depending on the skills of the J2EE developer)</i>

### Performance

Visual MainWin produces 100 percent Java-compliant code in less than one-tenth the time it takes to recreate the application from scratch. Applications ported with Visual MainWin perform similarly to any standard J2EE application, taking advantage of the scalability and robustness of the J2EE platform.

### Porting Challenges

A typical enterprise-class application containing hundreds of thousands of lines of code raises additional issues that need to be

addressed during the porting process. Mainsoft minimizes the potential for rewriting portions of code and offers best practices to resolve the challenges faced during the porting process.

### **Third-Party Libraries**

.NET applications may use third-party libraries. If they do, those libraries need to be converted to J2EE, in order to port them.

Mainsoft has partnered with Infragistics™, a leading vendor of third-party UI components, to provide J2EE versions the NetAdvantage™ presentation layer development toolset. If a .NET application uses NetAdvantage ASP.NET components, a simple installation of Mainsoft's J2EE version will allow the application to be ported successfully.

For other third-party components, it is possible to port the libraries to J2EE when the source code for the third-party libraries is available. When this is not the case, users can replace .NET libraries with equivalent Java libraries without compromising functionality. Most major third-party components, including Crystal Reports, IBM MQseries, Corda Mapping, Software FX, and SoftArtisan ExcellWriter, are available in both .NET and J2EE. Visual MainWin provides an easy way to import Java libraries into the Visual MainWin projects and call Java classes and methods directly from C# or Visual Basic.NET. Refer to the product documentation for details regarding how to add Java references to Visual MainWin projects.

### **Code Modifications**

During the porting process of a .NET application to J2EE, a small percentage of source code has to be modified. Customer experience shows that usually not more than 1% of an application needs to be changed, to successfully build the J2EE project.

Changes are usually required due to semantic differences between the .NET and J2EE platforms, such as case-sensitivity in file names. The .NET file system APIs are case-insensitive, following the Windows file system, while the J2EE file systems' APIs are case-sensitive. This results in different behaviors when using APIs that rely on the underlying file system. Special care must therefore be taken to match the actual case of the physical resources such as strings representing file names and paths on the file system.

Additional code modifications may be needed to address APIs that are not supported by Visual MainWin. In many cases, it is possible to substitute the API in question with other .NET APIs that are

supported by Visual MainWin. These include a full implementation on J2EE of the following .NET assemblies:

- System.Web
- System.Web.Services
- System.Data
- System.XML
- System
- Mscorlib
- Microsoft.VisualBasic

Visual MainWin supplies tools to quickly identify these code change tasks, including compilation errors and warnings, documentation of supported namespaces, and an online class library reference book.

## **Success through Partnership**

For more than a decade, Mainsoft has helped many of the world's largest ISVs develop mission-critical applications with Visual Studio and deploy them natively on multiple operating platforms such as J2EE, UNIX®, and Linux®.

That's why today Mainsoft can guarantee a smooth and efficient port of your .NET applications to the J2EE platform.

### **Fixed-Cost, Fixed-Time Approach for Porting**

The biggest problems software organizations face in porting mission-critical applications are the uncertainties that surround the cost and duration of the project. Mainsoft reduces the time it takes to bring applications to market by dramatically shortening the porting process. In fact, our experience shows that Mainsoft solutions can reduce development costs and time-to-market by 75 percent.

Mainsoft also guarantees the work for a fixed price. Most companies are forced to accept either in-house time-and-materials - based projects that creep through delivery milestones and engineering budgets. Mainsoft bases its costs estimations for porting projects on our capacity to port 5,000 lines of code per day. This estimation may change depending on the size and complexity of the project.

## **The Mainsoft Porting Methodology**

With its unrivaled cross-platform porting experience, Mainsoft Professional Services clearly understands most commonly encountered porting challenges and issues. This knowledge has been translated into a proven four-step methodology to expedite porting projects.

### Step 1: Requirement Analysis

First we do a technical review with the developer to define the requirements and success criteria of the project:

- Scope of the functionality that needs to be ported
- Dependency analysis of application modules including third-party components
- Identification of unsupported classes
- Application testing requirements

### Step 2: Detailed Project Plan

Our cross-platform experts perform a detailed analysis of the source code in order to identify porting issues. The Professional Services team details findings for the customer along with an estimate of the time required to successfully complete the porting project. Based on this information and customer requirements for time-to-market and costs, the Professional Services team produces a detailed project plan that includes scheduling, milestones, resource allocation, and a fixed cost to complete the project.

### Step 3: Implementation

The end result of this phase is an application that is up and running on the J2EE platform. It includes the following steps:

- Building of the application on J2EE on the application server of choice
- Implementation of non-supported .NET APIs required by the application
- Resolution of runtime issues and integration with third-party components
- Test suite validation

#### Step 4: On-Site Project Delivery

Mainsoft Professional Services provides training and education with the handover of the final application to ensure it can be successfully deployed and maintained.

#### **Conclusion**

With Visual MainWin, Mainsoft offers a cost-effective, guaranteed solution for porting .NET Web applications and Web services to the J2EE platform. Mainsoft is a dedicated partner that can help software organizations to reduce time-to-market and dramatically lower porting costs to deliver applications on both the .NET and J2EE platforms.

Development organizations can concentrate their development resources on the .NET platform and deliver applications on J2EE without a Java development team. As a complete development environment, Visual MainWin can be used as a strategic, long-term development and porting solution for the J2EE platform, allowing users to develop and maintain their applications across multiple versions and releases.

#### **About Mainsoft**

Founded in 1993, Mainsoft Corporation, the cross-platform development company, enables businesses to develop mission-critical applications with the Visual Studio development system and deploy them natively on J2EE, UNIX, and Linux platforms, dramatically reducing development costs and time-to-market. The company is a first-mover in cross-platform development. Its world-class research and development team has created patented cross-platform products that solve critical problems facing ISVs and IT organizations. Many of the world's largest ISVs including Siebel, Computer Associates, and IBM Rational, use Mainsoft's products to extend the productivity of Microsoft Visual Studio software, deploying more than \$1 billion worth of software annually on multiple operating systems. Headquartered in San Jose, Calif., the company has offices in Chicago, New York and Israel.

©Copyright 2005, Mainsoft Corporation

All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.