

Introduction to the
Cirquet™ Solution Suite 



Cirquet™ Training Series

Volume

1



Improv Technologies

Improv Technologies Cirquet Training

IMPROV TECHNOLOGIES AND/OR ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THE INFORMATION CONTAINED IN THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED IN THIS DOCUMENT FOR ANY PURPOSE. ALL SUCH DOCUMENTS AND RELATED GRAPHICS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND. IMPROV TECHNOLOGIES AND ITS RESPECTIVE SUPPLIERS HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS INFORMATION, INCLUDING ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT. IN NO EVENT SHALL IMPROV TECHNOLOGIES AND/OR ITS RESPECTIVE SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF INFORMATION AVAILABLE FROM THIS DOCUMENT.

THE DOCUMENTS AND RELATED GRAPHICS PUBLISHED IN THIS DOCUMENT MAY INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY MADE TO THE INFORMATION HEREIN. IMPROV TECHNOLOGIES AND/OR ITS RESPECTIVE SUPPLIERS MAY MAKE IMPROVEMENTS AND CHANGES IN THE PRODUCT(S) OR THE PROGRAM(S) DESCRIBED HEREIN AT ANY TIME. NOTICES REGARDING SOFTWARE, DOCUMENTS AND SERVICES AVAILABLE ON THIS WEBSITE. IN NO EVENT SHALL IMPROV TECHNOLOGIES OR ITS RESPECTIVE SUPPLIERS BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF SOFTWARE, DOCUMENTS, PROVISION OF OR FAILURE TO PROVIDE SERVICES, OR INFORMATION AVAILABLE FROM THIS SERVER.

COPYING OR REPRODUCTION OF THIS DOCUMENT TO ANY OTHER SERVER OR LOCATION FOR FURTHER REPRODUCTION OR REDISTRIBUTION IS EXPRESSLY PROHIBITED.

© 2002 Improv Technologies, Inc. Improv Technologies, Cirquet and Living Software are trademarks of Improv Technologies, Inc.

All other product, service and company names are trademarks or registered trademarks of their respective owners.

© Celeriti, LLC 2002



Table of Contents

Chapter 1	6
1 Introduction and Overview	6
1 About Improv Technologies	6
1.2 Overview and Objectives	6
1.3 This Workbook.....	2
2 Objectives.....	2
3 Lab Setup Instructions.....	2
4 Introduction to the Cirquet Solution Suite.....	4
4.1 What is a Cirquet?.....	4
4.2 What is Cirquet used for?.....	4
4.2.1 Cirquet Components thru Thick and Thin.....	6
4.2.2 The Cirquet Solution Suite Advantages.....	7
4.3 Basic Concepts.....	8
4.3.1 What is a JVM?.....	8
4.3.2 What Are JavaBeans?	9
4.3.2.1 Design Time Interfaces.....	9
4.3.3 What about software components?.....	9
4.3.3.1 Reusable Software Components	9
4.3.4 What Distinguishes JavaBeans from Java Classes?	10
4.4 About JXTA.....	10
4.5 About JNDI.....	11
4.6 What are Web Services?	12
4.7 The Cirquet Solution Suite Compared to Other Component Frameworks	13
Chapter 2	1
1 Introduction to the Cirquet Solution Suite	1
1.2 Cirquet General Concepts & Architecture	2
1.2.1 Cirquets – Challenge and Solution.....	2
2 Cirquet Solution Suite – Quick Overview	3
3 Why Use Distributed Components?	4
3.2 What is the Cirquet Solution Suite?	4
3.3 What are the elements of a Cirquet Solution?	5
3.3.1 The Cirquet Solution Suite	6
3.3.2 Elements in the Cirquet Solution Suite	6
3.3.3 Elements: More Important Information for Developers.....	7

3.3.4	Cirquet Components and Beans	9
3.3.5	What is SmartWrap?.....	9
3.4	Cirquet Architecture	10
3.5	Architecture Exercise [tbd]	14
3.6	Cirquet Application Lifecycle.....	14
4	Overview of Cirquet Design Studio	15
Chapter 3		1
1	Cirquet Basics Step-by-Step	1
1.2	Objectives.....	1
1.3	The Assembler Role	2
2	Assembling Cirquet Components in the Cirquet Design Studio (Exercise)	3
2.2	Introduction	3
2.3	Creating a New Cirquet Project.....	4
2.4	Saving a Cirquet Project.....	6
2.5	Opening an Existing Project.....	7
2.6	Your First Cirquet Project	9
2.6.1	Example Overview.....	9
2.8.2	Explanation of Example Code.....	10
2.6.1.1	Launchable.....	11
2.7	Connecting and Configuring Cirquet Components	12
2.7.1	Connecting via Properties	12
2.7.2	Connecting via Events	12
2.7.3	Source and Target Cirquet Components	12
2.8	Installing Cirquet Components on the Cirquet Palette	14
2.8.1	Adding a Category to the Cirquet Palette.....	14
2.8.2	Adding Cirquet Components to a Category in the Cirquet Palette.....	15
2.8.3	Add Beans to the Cirquet Palette (Exercise).....	16
2.9	Adding Cirquet Components to a Project in the Cirquet Inspector	16
2.9.1	Adding Beans to the Cirquet Inspector (Exercise).....	18
2.10	Configuring a Project's Components in the Cirquet Inspector.....	19
2.10.1	Changing Labels (Exercise)	20
2.11	Connecting a Project's Components in the Cirquet Connection Wizard.....	21
2.11.1	Target Operation (Exercise).....	24
2.12	Specifying More Complex Targets	25
2.12.1	One More Point	26
2.13	Launching the Assembled Cirquet Components from the Cirquet Inspector	27
2.13.1	Launching the Examples	29
2.13.2	Designating Remote Targets	30
2.14	Launching – Review.....	30
3	Exporting the Assembled Cirquet Components to a CAR file	31
4	Cleaning the Assembly Environment	33
5	Working with Multiple Assemblers	34
Chapter 4		2
1.0	Assembling Cirquets Programmatically.....	2
1.1	Overview	2
1.2	Initializing the Cirquet Directory	3
1.3	Binding and Configuring Cirquets in the Cirquet Directory	6
1.4	Looking Up Cirquets in the Cirquet Directory.....	8
1.5	Launching Cirquets Programmatically.....	8

1.6	Creating a Cirquet Project	9
Chapter 5	1
1	Installing the Cirquet Solution Suite.....	1
1.2	Setting Up the Cirquet Solution Environment.....	1
1.3	Configuring the Environment.....	3
1.4	System Requirements	3
1.4.1	System Requirements – Software	3
1.4.2	System Requirements – Hardware	4
1.5	Directory Structure	4
2	Configuring Network Options	5
3	Configuring the JXTA Transport	5
3.2	JXTA Virtual Network Overview	6
	Peer Addressing	6
	Message Relaying.....	6
	Firewall and NAT Considerations	7
3.3	Cirquet Network Architecture Considerations	8
3.4	Configuring Cirquet Peers on a JXTA Network	9
	Proxy Settings.....	10
	TCP Settings.....	10
	HTTP Settings	11
	Rendezvous and Relay Settings.....	11
3.5	Configuring a JXTA Rendezvous and Relay	13
3.6	Lab 2: Install Exercise.....	16
4	Troubleshooting.....	16
5	Configuring the Cirquet Peer Runtime	17
5.2	Configuration on Win NT, 2000, and XP	17
5.3	Configuration on Windows 98	17
5.4	Configuration on Solaris	17
5.5	Configuration on Linux.....	19
Chapter 6	21
1.0	Connecting to the Cirquet Network	21
	Getting the JNDI Context	21
	Logging in.....	21
	Getting the Cirquet Directory Context.....	22
	Creating A Cirquet Component	22
	Writing the Component	23
	Setting Cirquet Component Attributes.....	23
	Standard Attributes	24
	Registering a Cirquet	25
	Modifying a Cirquet.....	25
	Replacing a Cirquet	26
	Unregistering a Cirquet.....	26
	Using A Cirquet Component.....	27
	Looking up Cirquet Components.....	27
	Searching for Cirquets	27
	Invoking methods on Cirquet Components	28
	Building Applications with the Cirquet Solution Suite.....	30
	Making a Cirquet Component Launchable and Disposable	30
	Connecting Cirquet Components Together.....	31
	Deploying Cirquet Components on the Cirquet Network.....	32
Chapter 7 Administration and Security	35

Appendix 1: Cirquet APIs	38
CirquetConnection	38
CirquetStub	39
Launchable	41
Disposable	41
Response	41
ResponseHandler	42

*The Cirquet Solution Suite
enables dynamic businesses
to develop, deploy and
manage scalable distributed
applications rapidly and
cost effectively.*



Chapter 1

1 Introduction and Overview

This chapter provides an overview of this course.

1 About Improv Technologies

- Improv Technologies is a software infrastructure (middleware) company.
- The Cirquet Solution Suite enables dynamic businesses to develop, deploy and manage scalable distributed applications rapidly, in a cost effective manner, and with lower risk.

1.2 Overview and Objectives







- i. This course is intended for the Enterprise Java Developer and Java Systems Integrator. We will cover various topics related to Improv Technologies “Cirquet” Components and how to architect, develop and deploy distributed applications using the Cirquet Solution Suite. This course is NOT about Java programming although Java programming is an important prerequisite.
 - ii. The course is divided into 8 modules and is intended to be covered in 3 days. The second and third days’s sessions are heavily lab and exercise based and will involve hands-on Cirquet Component development.
-

1.3 This Workbook

The **course workbook** is divided into several sections. The first few modules review the history of distributed component design, provide a short history of Improv Technologies and the Cirquet Solution Suite, and present an overview of how the Cirquet Components work. Later sections deal with topics such as creating your first Cirquet Component, implementing Cirquet Components in your enterprise and finally architecting Cirquet Solutions.

DOCUMENT

ICON KEYS

	Valuable information
	Test your knowledge
	Lab
	Self Check
	Review
	Instructor Note

The course itself is intended for experienced Java solution developers with exposure to distributed component development. This includes JavaBeans, Enterprise JavaBeans, COM, DCOM, COM+, .NET, CORBA and JINI. If you're not familiar with these concepts, don't worry! The initial modules of this course will still be of interest and benefit to you. Modules 1 & 2 are useful for sales and marketing professionals, project managers and other staff in addition to developers and architects.

2 Objectives

Upon completion of this course, learners will:

- Understand how the Cirquet Solution Suite fits into a distributed Java architecture
- Learn how to use the Cirquet Component plug-in within Sun ONE Studio
- Learn how to create a Cirquet project
- Learn how to create Java components and add them to a Cirquet project
- Learn how to deploy and maintain Cirquet Component based projects
- Learn how to install, configure and troubleshoot the Cirquet Solution Suite
- Understand how to architect and plan a Cirquet Solution based application

3 Lab Setup Instructions

To get the most from the Cirquet Solution Suite labs and exercises, the following configuration is recommended:

- Cirquet Solution Suite Software and Hardware requirements – see pages ___ in this guide
- Two workstations – One with a complete Cirquet Solution Suite development setup, the second with, at a minimum, Cirquet Peer Runtime.

1.0 INTRODUCTION AND OVERVIEW

- A local network allowing workstations to communicate with each other. Students can use each others' workstations as opposed to having 2 per student.
- All work in this course is based in Sun's Forte for Java 3.0 Community Edition (Sun ONE Studio).
- Cirquet Solution Suite release 1.1 is used in this workbook.

4 Introduction to the Cirquet Solution Suite

The slide features a dark blue background with a yellow border. At the top left, there is a small 'IT' logo with a starburst. The title 'Introduction to the Cirquet Solution Suite' is centered in a white box. Below the title, the text 'WITHOUT Cirquet, I.T. organizations face three barriers:' is followed by a bulleted list: 'SPEED to develop and deploy', 'COST of projects', and 'RISK of taking on these initiatives'. Below the list, the text 'WITH Cirquet:' is followed by 'Rapid, cost effective solution with lower risk.'

4.1 What is a Cirquet?

The **Cirquet Solution Suite** is a platform for the development, deployment and management of distributed applications composed of JavaBeans and Web services.

Developers can rapidly assemble distributed applications from JavaBeans and Web services either programmatically or through visual assembly tools and wizards. Components and services in the Cirquet environment are totally location independent. **Cirquet Runtime** performs all discovery, binding and communication with remote objects transparently. Within the Cirquet Solution environment the programmer does not need to define component location or linking – these properties are dynamically configurable by an application assembler or administrator without coding.

4.2 What is Cirquet used for?

Cirquet is a general-purpose tool for building any distributed application. The goal of the Cirquet Solution Suite is to simplify and accelerate the development, management and deployment of distributed applications. Additionally, Cirquet Components enable dynamic re-configuration and modification with minimal administration and without application downtime and user disruption.

1.0 INTRODUCTION AND OVERVIEW

The Cirquet Solution Suite has similarities to other distributed component environments such as EJB, CORBA, COM+ and .NET and can be used in any place where those would be considered for use.

The Cirquet Component's unique capabilities of dynamic discovery, dynamic updates, peer-to-peer capabilities and peer group operations enable novel application solutions beyond the typical client/server and n-tier/web application architectures.



Describe the advantages that Cirquet provides to IT development.

4.2.1 *Cirquet Components thru Thick and Thin*

Cirquet Components work equally well in thin or thick client applications. The Cirquet Solution serves to abstract away the network architecture from the component. Additionally, dynamic deployment and update capabilities address one of the perceived key shortcomings of thick-client technology: the deployment/update issue.

Cirquet Components are maintained centrally, streamed on demand to the desired location and dynamically updated without administration or user intervention. This extends most of the benefits of thin client application deployment and management to rich, interactive applications that function even if a network connection is not available. Standard thin client technologies such as JSP and Servlets interoperate with Cirquet Components. The business logic components are shared across client types. As every Cirquet Component is also a Web service, non-Java applications easily access and re-use the same components.

Thin Clients



Centralized application servers typically employ a “thin client” model in which business logic and presentation components are hosted on a centralized server and accessed through a lightweight client (usually a web browser).

Thick Clients

Distributed object frameworks, like CORBA and DCOM, are often used in “thick client” applications. “Thick client” applications have software installed on each client machine and connect to a specific service or set of services hosted on one or more remote devices. In today’s application servers and distributed object frameworks, functionality must be designed, deployed and maintained to specifically address the hardware, transport and communications requirements of the underlying network architecture.



4.2.2 The Cirquet Solution Suite Advantages

Improv Technologies has taken a significantly different approach to the problem of distributed component development through the Cirquet Solution Suite. Cirquet Components create a layer of abstraction between standard component functionality and component communication across the network. This enables the Cirquet Components to employ:

- “fast-process” communication for components residing on the same peer
- cross-platform XML messaging where components are hosted on different peers on the network

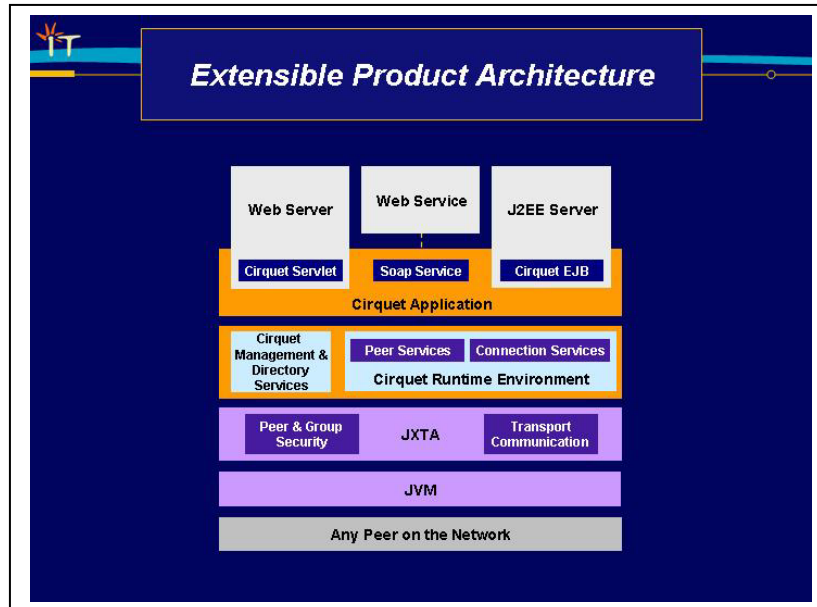
All without requiring additional work on the part of the developer.

The Cirquet Solution Suite’s dynamic deployment capabilities enable components to be distributed, managed and upgraded over a large number of servers, clients, and portable devices from a centralized location. This eliminates the maintenance issues associated with today’s distributed application framework.

Additionally, and equally important, the Cirquet Solution Suite:

- Supports existing component methodologies, enabling pre-existing components and components created using well known programming techniques to be immediately available as Cirquet Component based Web services.
- Eliminates most of the learning curve normally associated with the creation of network-based applications.
- Contains built-in support that enables Cirquet Components to interoperate with standard (non-Cirquet) Web services, such as those proposed by Microsoft’s .NET initiative and the Sun ONE Platform.

4.3 Basic Concepts



A thorough understanding of these basic concepts will result in the successful development of Cirquet Component based solutions. For many, these concepts are review.

4.3.1 What is a JVM?



A JVM is the "Java Virtual Machine". It may represent three different things when you say "Java Virtual Machine." You may be speaking of:

- the abstract specification,
- a concrete implementation, or
- a runtime instance.

Concrete implementations, which exist on many platforms and come from many vendors, are either all software or a combination of hardware and software. A runtime instance hosts a single running Java application.

Each Java application runs inside a runtime instance of some concrete implementation of the abstract specification of the Java Virtual Machine. More information on the abstract specification is described in detail in the book: *The Java Virtual Machine Specification*, by Tim Lindholm and Frank Yellin.



4.3.2 What Are JavaBeans?

A JavaBean is a reusable software component that is written in the Java programming language. It can be visually manipulated in builder tools. A JavaBean is often referred to simply as a Bean.

4.3.2.1 Design Time Interfaces

JavaBeans can define a design-time interface. This interface allows application designer tools, or builder tools, to query components. This query can determine the kinds of properties the components define and the kinds of events they generate, or to which they respond. In visual application builder environments, Beans are sometimes referred to as reusable software components or custom controls.

In addition, end users, or third party suppliers, can build custom Beans independently of the GUI platform or operating system. IS departments can distribute these third-party components to internal enterprise clients, or they can be sold as add-on components to anyone who uses builder tools to construct applications.



4.3.3 What about software components?

Software components are self-contained, reusable software units. Using visual application builder tools, software components can be composed into applets, applications, servlets, and composite components. When this composition is integrated within a graphical user interface, the results are immediately appreciated.

4.3.3.1 Reusable Software Components

Reusable software components apply the concept of interchangeable parts to the field of software construction. Other industries have long profited from reusable components.

Reusable software components can be simple, such as buttons, text fields, list boxes, scrollbars, and dialog boxes or highly complex such as transactional elements.



4.3.4 What Distinguishes JavaBeans from Java Classes?

Any Java class that adheres to certain conventions regarding property and event interface definitions can be a JavaBean. Beans are Java classes that can be manipulated in a visual builder tool and composed into applications.

Introspection, the process by which a builder tool analyzes how a Bean works, differentiates Beans from typical Java classes. Because Beans are coded with predefined patterns for their method signatures and class definitions, tools that recognize these patterns can "look inside" a Bean and determine its properties and behavior.

Introspection allows a Bean's state to be manipulated at design time — that is, at the time it is being assembled as a part within a larger application. For this to work, method signatures within Beans must follow a certain pattern so that introspection tools recognize how Beans are manipulated, both at design time and at run time.

In effect, Beans publish their attributes and behaviors through special method signature patterns that are recognized by Beans-aware application construction tools. However, these construction tools are not required to build or test Beans. The pattern signatures are easily recognized by human readers, as well as builder tools. One of the objectives of this course is to learn how to build Cirquet Components as Beans with methods that adhere to these patterns.



4.4 About JXTA

JXTA technology is a set of open, generalized peer-to-peer (P2P) protocols, defined as XML messages. These allow any connected device on the network, ranging from cell phones and wireless PDAs to PCs and servers, to communicate and collaborate in a P2P manner.

JXTA peers create a virtual network where any peer can interact with other peers and resources directly. This is true even when some of the peers and resources are behind firewalls and NATs or are on different network transports.

Each peer in a Cirquet Solution is a JXTA peer. Communication channels between these peers are dynamically bound and automatically routed through relays to cross NAT's and firewalls.

JXTA Rendezvous: is analogous to a DNS server, a rendezvous is a JXTA peer that maintains a list of peers to which it knows about, enabling a peer to discover other peers. Any JXTA peer can be configured as a rendezvous and any JXTA peer can be configured to query a set of rendezvous peers. Several public rendezvous peers are available for anyone to use. More information can be found at the Project JXTA site, www.jxta.org.

JXTA Relay: is analogous to a router, a relay bridges network domains and forwards messages as required. Any JXTA peer can be configured to be a relay and any peer can be configured to use a set relay of relays.



4.5 *About JNDI*

The **Java Naming and Directory Interface™ (JNDI)** is a standard extension to the Java™ platform. JNDI provides Java technology-enabled applications with a unified interface to multiple naming and directory services in the enterprise. This naming and directory functionality allows applications to use methods that associate attributes to objects and search for objects by using attributes.

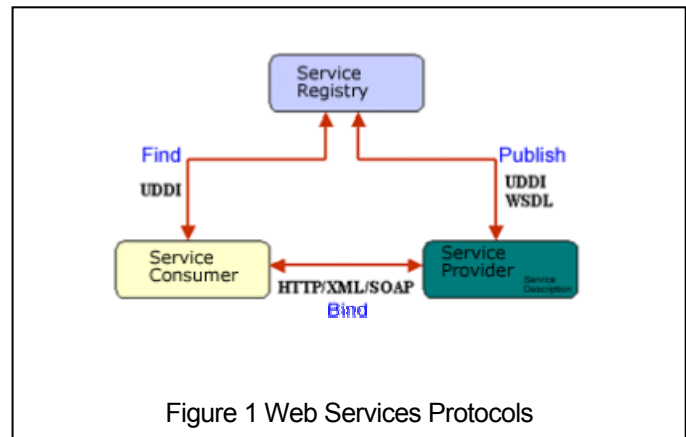
JNDI enables connectivity to heterogeneous enterprise naming and directory services such as LDAP, NDS, DNS and NIS. Developers can build powerful and platform independent, portable directory-enabled applications using this standard. The Cirquet Directory is exposed through a JNDI service provider, which enables users to bind, modify lookup and search for Cirquet Components through the JNDI APIs.



4.6 What are Web Services?

Web services are computing services offered through the web that are accessible from any web-service-enabled machine with Internet access. Web services enable interoperability through a set of XML-based open standards. To access a web service, clients need to know the URL and the data types used for the method calls. The client does not need to know if the service is running on Windows or Unix or Linux, nor does the client need to know if it was written in Java, or C#, or C++ or Visual Basic.

Businesses use the XML-based Web Services Description Language (WSDL) to describe their web services on the Internet and list them in an XML-based registry such as the Universal Description, Discovery, and Integration (UDDI) protocol. UDDI allows the search of publicly available web services as shown in Figure 1. A client sends a service request to the directory, which informs the client about the registered services that meet the criteria of the request. The SOAP protocol is then used to communicate, using HTTP and XML as an exchange mechanism, between the applications running on different platforms.



4.7 The Cirquet Solution Suite Compared to Other Component Frameworks

	Microsoft COM	EJB	CORBA	Cirquet
<i>Server Model</i>	<i>Requires Windows based servers</i>	<i>Utilizes a central server model</i>	<i>Requires remote server model</i>	<i>Components can be deployed on any peer and moved dynamically.</i>
<i>Deployment Model</i>	<i>Requires installation packages or manual distribution</i>	<i>Each EJB must be compiled and distributed for each version/change</i>	<i>Complex deployment with fragmented tools for deployment</i>	<i>Components can be manually deployed to any peer, or automatically deployed at run time by the application or by user invocation</i>
<i>Development Platform</i>	<i>Microsoft Windows with limited support for non-Windows OS and only Windows tools</i>	<i>Vendor implementations are often fragmented and not portable</i>	<i>Few tools with many vendor specific differences</i>	<i>J2SE JDK 1.3.1_02+, assembler plug-in to Sun ONE Studio (Optional)</i>

1.0 INTRODUCTION AND OVERVIEW

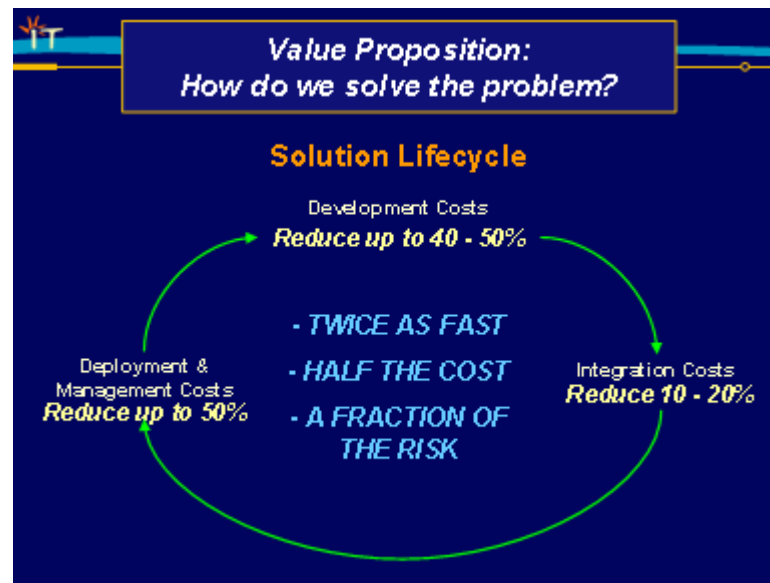


Chapter 2

1 Introduction to the Cirquet Solution Suite

This chapter explains the Improv Technologies Cirquet Solution Suite: What it does and how it can help shorten development cycles and lower costs.

Java programming has become a core skill set for application development. Cirquet Component development provides a faster, more efficient method for developing and deploying network based distributed applications. This course looks at how the Cirquet Solution Suite architecture helps you accomplish this, how to create a Cirquet Solution and how to implement and deploy Cirquet Components within an enterprise.



1.2 Cirquet General Concepts & Architecture

1.2.1 Cirquets – Challenge and Solution

IT

**Introduction:
What problem do we solve?**

Example: A Telco considers overhauling its OSS systems to “fix the past” and “prepare for the future”:

“Fix the Past”

- **Integrate multiple legacy systems**
 - Billing, Provisioning, Order processing
- **Provide functionality / support to users**
 - Management reporting
 - Organizational interfaces
- **Manage inter-connection with RBOCs and other carriers**

“Prepare for the Future”

- **Allow for new products and services**
 - DSL, Remote PBX (IP Telephony)
 - Consolidate other M&A activity

“Barriers to Success”

- **TIMING:** Takes too long
 - No capable platform exists
- **COST:** Prohibitive to build a new OSS
 - Knowledge base workers
 - Increased headcount
 - Computer infrastructure
- **RISK:** Too much to bite off at one time
 - No guarantee of project completion
 - “ALL or NOTHING venture”

The architecture and advantages of the Cirquet Solution Suite enable developers to build component driven solutions to address these very issues.

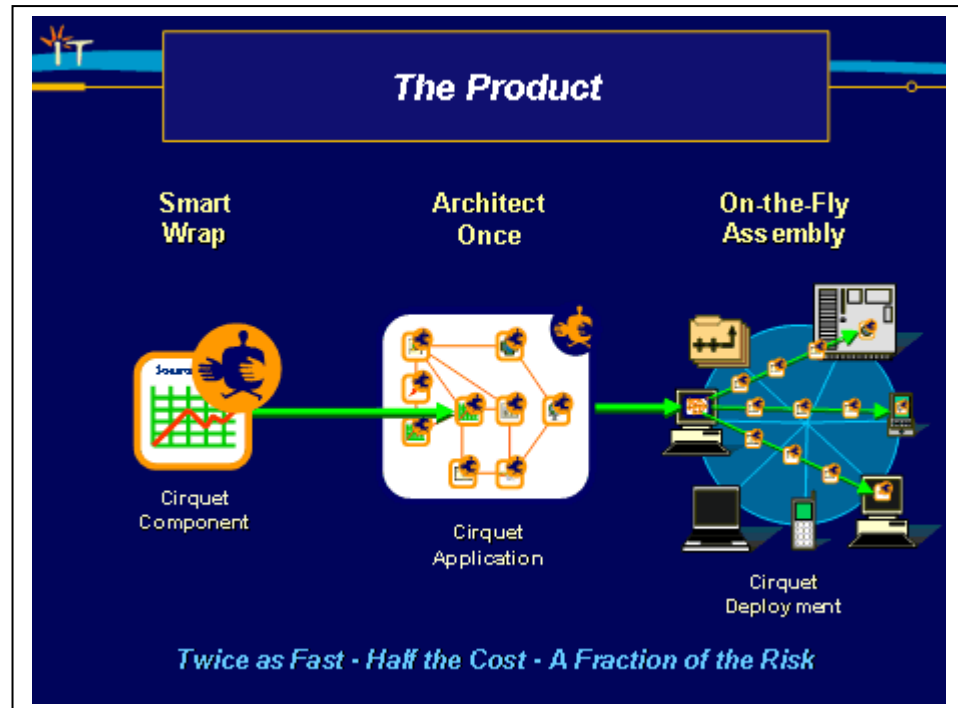
2 Cirquet Solution Suite – Quick Overview

The flexibility and advantages of the Cirquet Solution Suite come from:

- SmartWrap: SmartWrap allows developers to use existing JavaBeans and connect them at assembly time. Smartwrap “wraps” the existing beans into the Cirquet Peer Runtime.

Note that because the Cirquet Runtime is multi-threaded and can invoke methods at any time, you should ensure that your code is thread-safe.

- Architect Once concept: Cirquets allows the developer to write code that can be developed and tested locally with properties that can be set at deployment time.
- On-the-Fly Assembly: Components and services in the Cirquet Network are totally location independent. The Cirquet Runtime transparently performs all discovery, binding and communication with remote objects. Within the Cirquet Solution Suite development environment, the programmer does not need to define component location or linking – these properties are dynamically configurable by an application assembler or administrator without coding.



3 Why Use Distributed Components?



Distributed Components are de rigueur for today's network applications. Try to come up with 5 reasons why.

1. Distributed Components allow a business system to be more accessible.
2. Distributed Components allow parts of the system to reside on different computers. This allows specialized components to reside on specialized devices.
3. Distributed Components allow separation of business logic from data.
4. Distributed Components allow partitioning of applications to optimize computing and network resources.
5. Distributed Components allow scaling of applications to handle increasing loads, by adding additional CPUs.

3.2 What is the Cirquet Solution Suite?

The Cirquet Solution Suite is a distributed component infrastructure. Cirquet Components are written as independent entities that expose some properties, event interactions and methods. These exposed items can be used to assemble Cirquet Components into an application.

Cirquet Components within an application can be run on different machines, transparent to the code of the application. The assignment of a particular Cirquet Component to a particular machine can be deferred until deployment time, providing greater flexibility in how an application is deployed.

In addition, Cirquet Components run independently of one another. A Cirquet Component can be updated while the application continues to run.

The Cirquet Solution Suite infrastructure converts method calls between Cirquet Components into messages, and routes those messages to the appropriate Cirquet Components wherever they are running.

Like JavaBeans, Cirquet Components have properties that can be manipulated and they can fire and listen for events. For more information on JavaBeans, see the tutorials on Sun's Java Developer Connection (<http://developer.java.sun.com/developer/onlineTraining/Beans/>).

3.3 What are the elements of a Cirquet Solution?

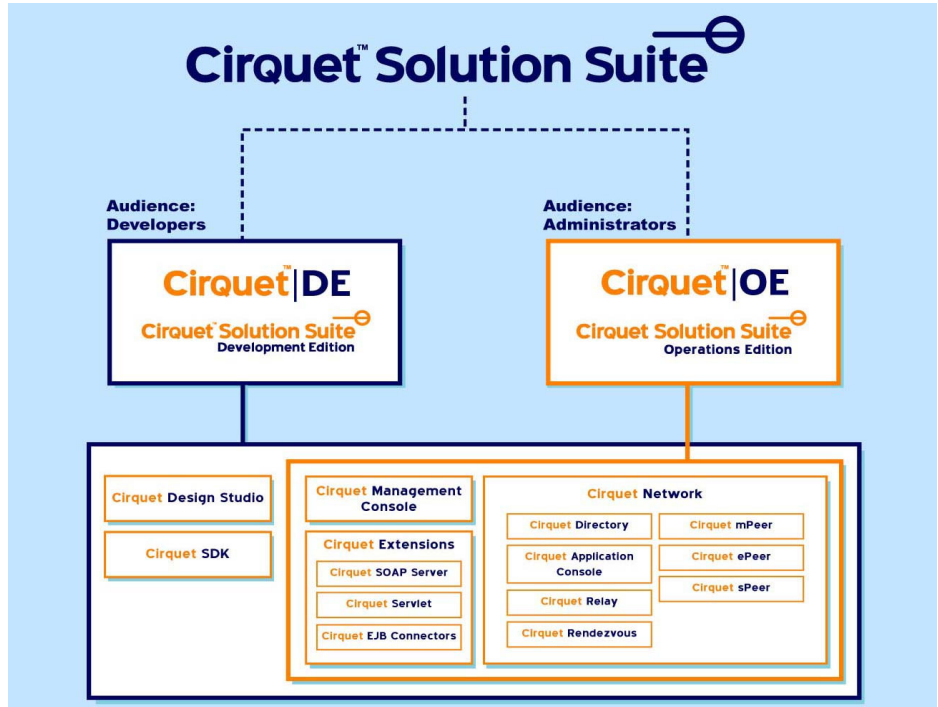



Figure 2 Cirquet Solution Product Architecture

There are 2 packages in the Cirquet Solution Suite: The Cirquet Solution Suite Development Edition (Cirquet | DE) and the Cirquet Solutions Suite Operations Edition (Cirquet | OE).

Cirquet Solution Suite Development Edition (Cirquet DE)	Cirquet Solutions Suite Operations Edition (Cirquet OE)
Develop distributed applications as Cirquets. Increasing the flexibility, interoperability, scalability and re-usability of the application.	Centrally manage Cirquet applications; upgrades, updates and bug fixes across the enterprise.

 Cirquet Servlet, Cirquet EJB Connectors, Cirquet Relay, Cirquet Rendezvous, Cirquet mPeer, and Cirquet ePeer are not supported in Release 1.1 and are enhancements planned for 2002/2003.

3.3.1 The Cirquet Solution Suite

Cirquet Components make the development and deployment of distributed components much easier than other, complex component technologies. The developer can develop locally, test locally and then deploy remotely all within a single tool. Applications are architected once and can be implemented on any device with a JAVA 2 JVM (Java Virtual Machine).

3.3.2 Elements in the Cirquet Solution Suite

The Cirquet Solution architecture contains these elements:

Cirquet Container	Holds the configurable properties of the Cirquet Component: <ul style="list-style-type: none"> • the code base • the class files • the access control information 	A lightweight, self-contained, Java component and (optionally) Web service
Cirquet Component	Represents: <ul style="list-style-type: none"> • the business logic • the presentation layer • other component functionality 	Generally in the form of a JAR file or Java class referencing any Java language component.
Cirquet Applications	Created by configuring and assembling pre-compiled Cirquet Components in a run-time environment.	No need to shut down, recompile, and re-install whole applications when new functionality is added.
Cirquet Peer Runtime	Manages: <ul style="list-style-type: none"> • the instantiation of Cirquet Components on a local peer • the communication between the Cirquet Components 	Executes as a persistent Windows service or UNIX/Linux daemon
Cirquet Directory	Provides a secure, reliable storage and retrieval facility for the Cirquet Solution Suite.	Stored in a relational database using the JDBC-compliant database product that was specified during installation of the Cirquet product.
Cirquet Users	<ul style="list-style-type: none"> • the set of peer devices on the network • the set of end users and assemblers who have 	Specified by the administrator for the Cirquet Peer Network

	access to Cirquet Solutions based distributed applications	
Cirquet Peers	The collection of individual computing devices where individual Cirquet Components may run or be accessed.	Peers can be configured in peer groups, with a given peer participating in multiple peer groups.



3.3.3 Elements: More Important Information for Developers

An assembled **Cirquet Application** is also known as a Project during its development.

Cirquet Peer Runtime sits on top of the Java Virtual Machine (JVM) on each peer in a Cirquet Solution network and manages the instantiation and communication of Cirquets on local peers. Cirquet Components residing on the same peer utilize fast intra-peer communication. Cirquet Peer Runtime utilizes an efficient message transport to communicate between peers in a Cirquet network, eliminating the SOAP when it's not required. However, when residing on different peers, Cirquet Components communicate using cross-platform SOAP-based messages over the JXTA peer-to-peer transport, leveraging JXTA's peer discovery and security features.

End users access Cirquet Solution Suite applications via their pre-configured Cirquet Application Console. The console is a Cirquet Peer Runtime client that displays the collection of Cirquet Component applications available to the user and enables users to start those applications.

The Cirquet Directory is the central repository for configuration information for Cirquet Components, users, and peers within a Cirquet Solution System. Specifically the Cirquet Directory contains:

- the Cirquet container and code base for individual Cirquet Components
- the Cirquet assemblies (projects)
- user and peer configuration data

Access to the Cirquet Directory is available for

- Developers and Assemblers via a Cirquet Solution enabled IDE
 - to create and/or assemble Cirquets Components.
- Administrators via the Cirquet Management Console (CMC)
 - to configure the Cirquet Solution network

- to specify Cirquet Solution deployment options

Please note that all of these tasks can be performed using the Cirquet Directory's API as well.

The Cirquet Directory resides on the peer where the Cirquet Solution Suite Development Edition and Cirquet Solution Suite Operations Edition products are installed. The database for the Cirquet Directory may be configured on a network server accessible by the system that runs the Cirquet Directory.

• **Cirquet Users**

Users in a Cirquet network may be assemblers who will be assembling Cirquet applications using a Cirquet Component enabled IDE, or they may be the individual end users who will be executing Cirquet applications in a production environment.

An assembler accesses the Cirquet Solution enabled IDE by logging onto the Cirquet Design Studio using a unique user name and password. Similarly, end users log onto the Cirquet Peer Runtime using a unique user name and password, and are presented with a Cirquet Application Console that contains a set of launchable Cirquet Components configured for their use.

Cirquet Peers Cirquet peer systems can be comprised of PCs, servers, or (ultimately) handheld wireless devices.

Peer groups enable the ability to deploy a Cirquet Component on a group; **Load Balancing** is built into the Cirquet Peer architecture. When a component is deployed to a defined group, the method call is sent to all peers in that group. However, the Cirquet will run on just one peer; the first one to respond. Typically, that first peer should be the one that is least active at that point in time.

There are programmatic means to implementing a complete broadcast method. This topic is covered in some of the advanced modules of this course.

3.3.4 Cirquet Components and Beans


In general, working with Cirquet Components is similar to, and as easy as, assembling a Java Beans application, but with the added advantages of distribution and independence of the components.

Like JavaBeans, Cirquet Components are components that can be configured and assembled to create useful applications. Although each Cirquet Component runs independently, Cirquets Components work together in an assembly to provide the application functionality. The Cirquet Solution Programmer is responsible for creating individual Cirquets Components, based on the needed functionality of the application in development.


3.3.5 What is SmartWrap?

The Cirquet Solution Suite provides SmartWrap functionality that automatically converts an existing JavaBean to a Cirquet Component. Existing JavaBeans are easily used in a Cirquet application.


3.4 Cirquet Architecture




Creating a Cirquet Application



Source Code



Cirquet Component



1. SmartWrap™

Automated encapsulation of Java components, with minimal code changes

SmartWrap turns JAVA components into Cirquet Components. Few, if any, changes to existing code are necessary.

A Cirquet Component can integrate with J2EE applications utilizing any Java standard. If the Cirquet Component is instantiated within the context of an application server, JSPs, Servlets and EJBs call those EJBs as they would call any other JavaBean. Cirquet Components can reference EJBs and Servlets as usual. In addition, standard RMI-IIOP access to the EJBs is available.

Creating a Cirquet Application

2. Architect Once
Concentrate on business logic, without concern for network distribution

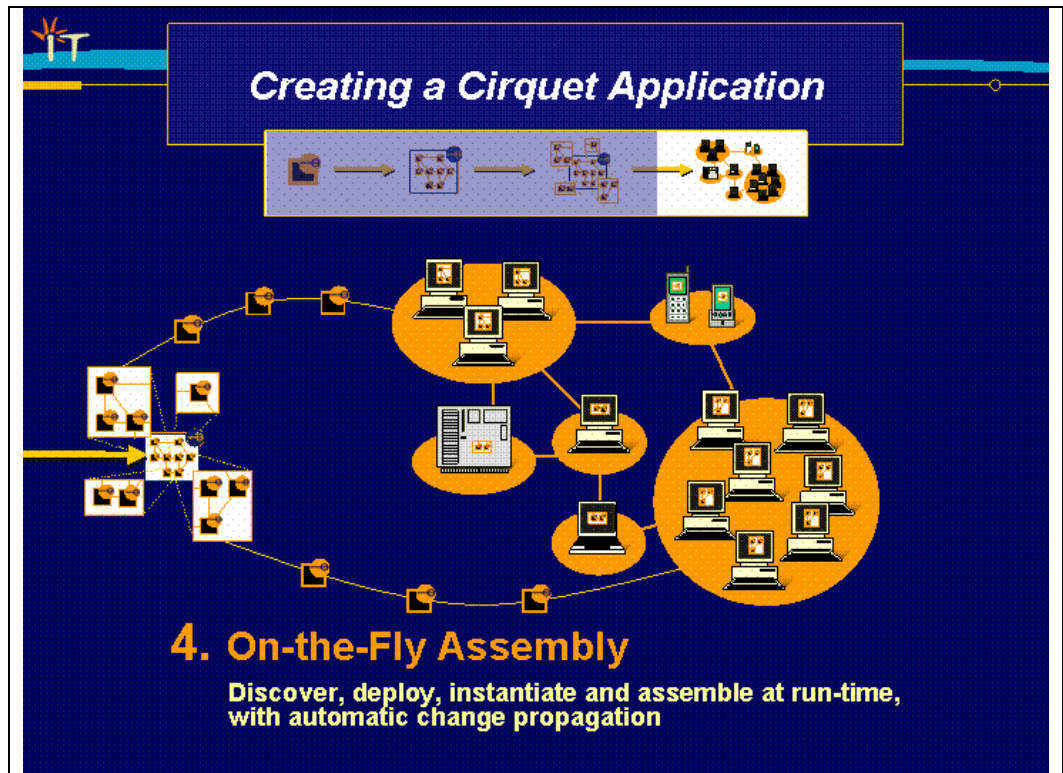
Architect Once – Focus on business

The Cirquet Solution environment intercepts communication between components and determines what effect it should have on the communication. Cirquet Runtime determines if the referenced component is instantiated. If yes, a direct method call is made to a local component, or a message is created and sent to a remote machine(s) hosting the component. If no, the component is not yet instantiated, Cirquet Runtime asks the Cirquet Directory to instantiate the component. The code and properties are streamed to the configured location and the component is instantiated.

Creating a Cirquet Application

3. Scale-at-Will
Re-partition application to optimize performance, without any code changes

Cirquet Solution applications can optimize CPU use and network latency by partitioning the application across multiple machines at the individual component level. In addition, a Cirquet Component can be configured to run on a group of machines to provide massive scalability, redundancy and load balancing. When a component is configured to be deployed on a group, a Cirquet Component dispatches the message to the most responsive machine in the group, which should be the least loaded machine. If a component or machine fails, all subsequent messages will be allocated across the remaining available components. The location of a component or the number of CPU's on which it runs may be dynamically re-configured without interrupting the application.



All JavaBeans within the Cirquet Solution are accessible as Web services. Cirquet Components generate the WSDL file on demand and the built in SOAP dispatcher routes the incoming SOAP request to the referenced component. Using the Cirquet SDK, developers can use Web services as if they were JavaBeans, without needing to know WSDL or SOAP.

The Cirquet Solution environment is Java based and can run anywhere that a JVM is present. Cirquet Runtime exists as a service or daemon on any machine that Cirquet Components are to be deployed. The underlying network technology uses JXTA protocols, which allow for dynamic discovery, peer-to-peer communication, peer group operations and communication across firewalls, NAT and proxy servers.



3.5 Architecture Exercise [tbd]

3.6 Cirquet Application Lifecycle

A Cirquet application is an assembly of Cirquet Components that are deployed in a flexible, distributed manner on a network. Cirquet applications move through the following lifecycle stages.

Cirquet Development – During development, the functionality of each component is implemented and the Cirquet Component is enabled. Each implementation may be:

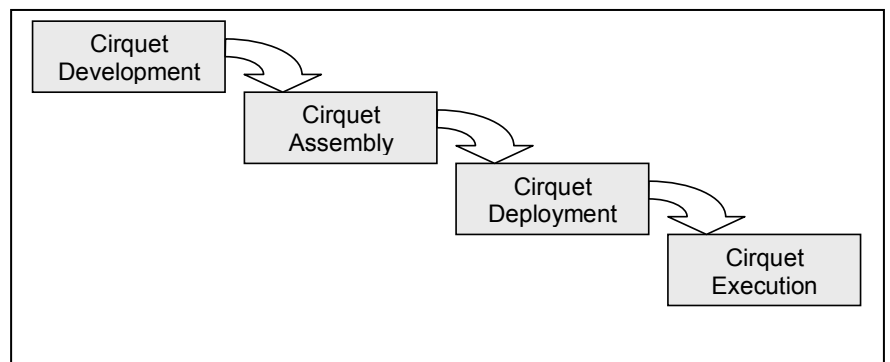
- A JavaBean™ that is SmartWrapped™ to become a Cirquet Component
- Developed from inception as a Cirquet Component.

All Cirquet Component implementations are stored in one or more JAR files and their interfaces are documented to enable use and reuse.

Cirquet Application Assembly – The Cirquet Component implementations are used to create Cirquet Components that instantiate the implementation. These components are assembled into a Cirquet application with configured properties that specify, among other things, the connections between the Cirquet components.

Cirquet Application Deployment – A Cirquet application is deployed by configuring the locations where the individual Cirquet components will execute on a Cirquet network. The Cirquet Administrator specifies the deployment configuration for a Cirquet application.

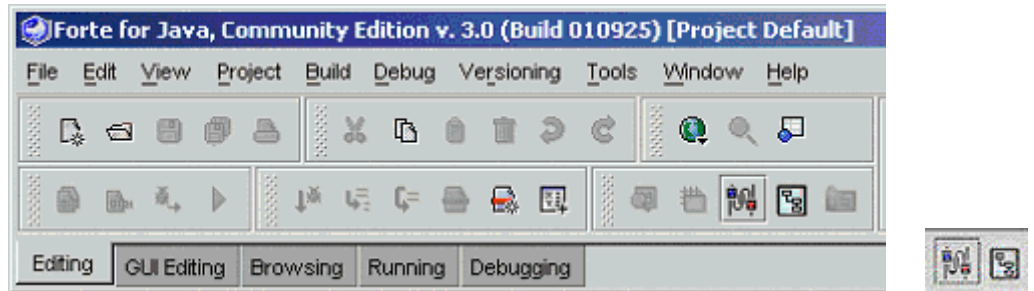
Cirquet Execution – Users launch the user-interface Cirquet Component(s) within a Cirquet application as the means of accessing the deployed Cirquet application. The Cirquet Administrator configures the set of Cirquet Components and applications that a user can access and launch.



4 Overview of Cirquet Design Studio

Cirquet Design Studio adds some new menu items, **toolbar** buttons and dialogs to Sun ONE Studio, as shown in the following figures:

There are two new **toolbar** buttons added to Sun ONE Studio's main window:

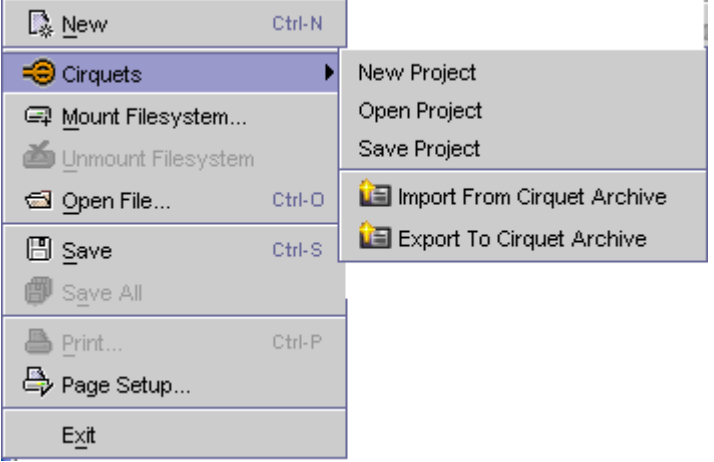
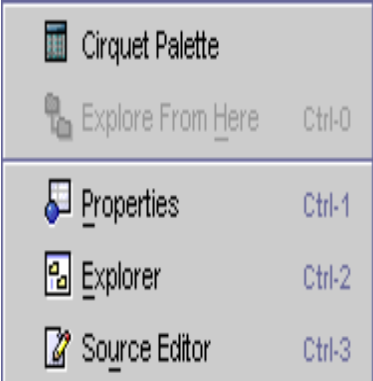


The new toolbar items: **Cirquet Connection Wizard** and **Cirquet Inspector**.

Figure 3 The Cirquet Design Studio Toolbar Buttons in Main Window

2.0 INTRODUCTION TO THE CIRQUET SOLUTION SUITE

In addition to the **toolbar** changes, the **File**, **View** and **Tool Menus** contain new Cirquet items:

<p><i>The Cirquet Design Studio File Menu (partial):</i> Cirquets with New Project, Open Project, Save Project, Import From Cirquet Archive <i>and</i> Export To Cirquet Archive <i>sub-items.</i></p>	
<p><i>The Cirquet Design Studio View Menu (partial):</i> new Cirquet Palette <i>item:</i></p>	

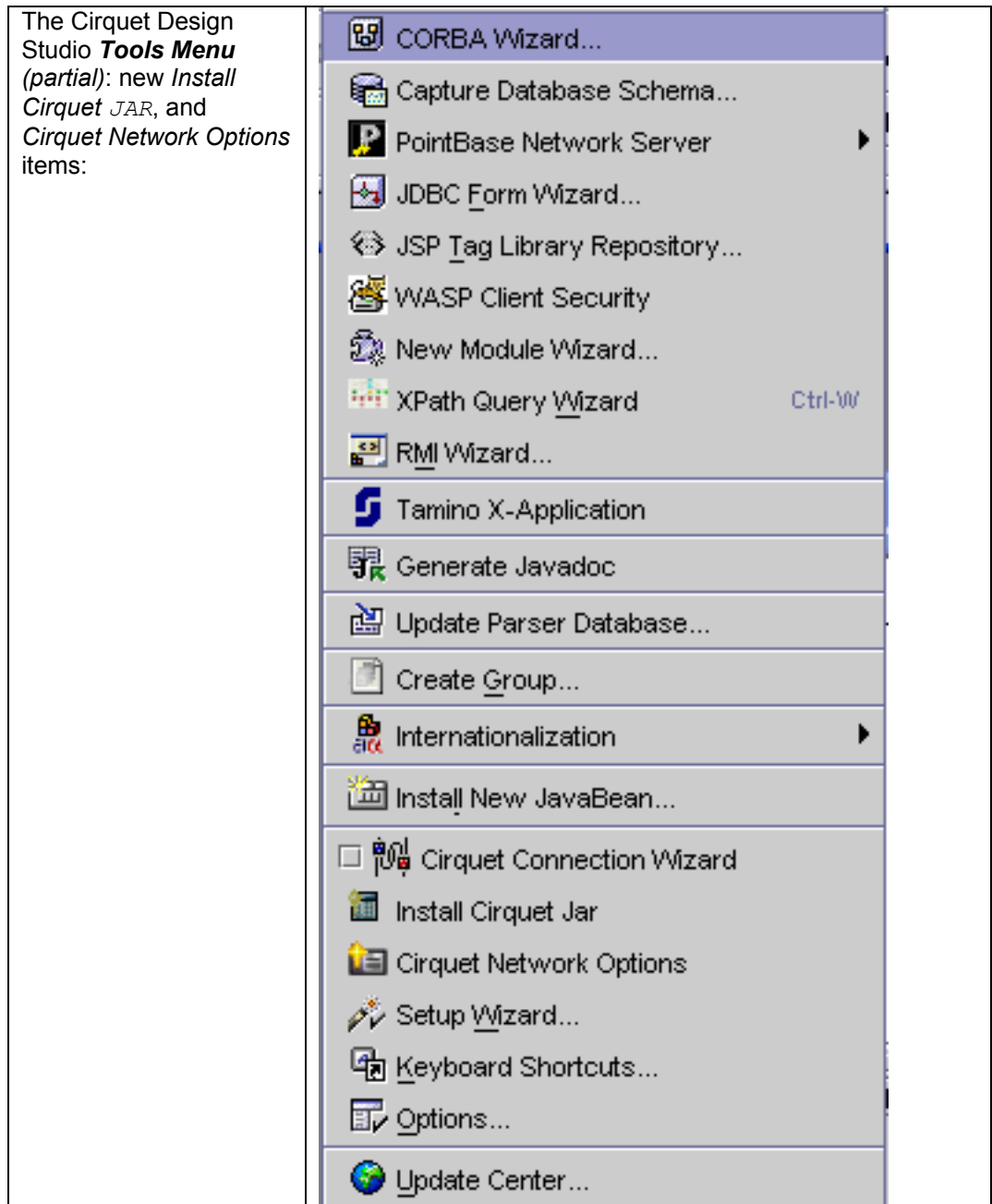


Figure 4: The Cirquet Design Studio *File*, *View* and *Tool Menus*



Chapter 3

1 Cirquet Basics Step-by-Step

This chapter provides a walk thru of a simple Cirquet Component implementation using an example that is included in the Cirquet Solution Suite. All of the examples can be found in the `\improv\examples\beanexample` directory.

Building applications with Cirquet Components consists of two steps.

1. Connecting Cirquet Components together so they can interact with each other
2. Deploying the Cirquet Components that make up the application on various hosts across the network

1.2 Objectives


During this section we will:

- Learn how to use the Cirquet plug-in with Sun ONE Studio
- Learn how to create and save a Cirquet Project
- Assemble a Cirquet Component
- Learn how Cirquet Components can be distributed, and deployed.

1.3 The Assembler Role

Cirquet Components are components that can be configured and assembled to create useful applications much like JavaBeans. They have properties that can be manipulated and they can fire and listen for events. Conveniently, the Cirquet Solution Suite provides SmartWrap functionality that automatically converts all existing JavaBeans to a Cirquet Component. So any existing JavaBean can be used easily in a Cirquet application.

In general, working with Cirquet Components is similar to, and as easy as, assembling a Java Beans application but with the added advantages of distribution and independence of the components. For more information on JavaBeans, see the tutorials on the Java Developer Connection at developer.java.sun.com/developer/onlineTraining/Beans/.


 Even when the same person is performing both assembly and administration, separate logins should be used for the Assembler and the Administrator to separate the roles and avoid confusion

Although each Cirquet Component runs independently, Cirquet Components work together in an assembly to provide the application functionality. The assembler is the person responsible for configuring and assembling already existing Cirquet Components into applications, by configuring and connecting them.

Cirquet users are administered using the Cirquet Management Console. See modules on *Cirquet Administration* for more information on how to register users and their passwords.



2 Assembling Cirquet Components in the Cirquet Design Studio *(Exercise)*

 Cirquet Components can be assembled in the Cirquet Design Studio, an IDE plug-in for Sun ONE Studio that permits the direct manipulation of Cirquet Component properties and connections. Use Sun ONE Studio for the development and testing of the Java components, then package them in a JAR file. Cirquet Design Studio can import the JAR file and Smartwrap the Java components. Alternatively, code can be written that programmatically manipulates Cirquet Component properties and connections to form assemblies.

2.2 Introduction

During this Lab learners will:

- Become familiar with Cirquet Design Studio
- Create a Cirquet Project in the **Explorer** window.
- Install Cirquets Components on the **Cirquet Palette**
- Add Cirquet Components to the Project in the **Cirquet Inspector**
- Configure a Project's Cirquet Components in the **Cirquet Inspector**
- Connect a Project's Cirquets Components in the **Cirquet Connection Wizard**
- Launch the Assembled Cirquet Components from the **Cirquet Inspector**
- Export the Assembled Cirquet Components to a CAR file
- Finally, clean up the assembly environment so that new projects can be initiated without clutter

2.3 Creating a New Cirquet Project

The first step in working with Cirquets in the Cirquet Design Studio is creating a Cirquet Project.

From the Sun ONE Studio menu

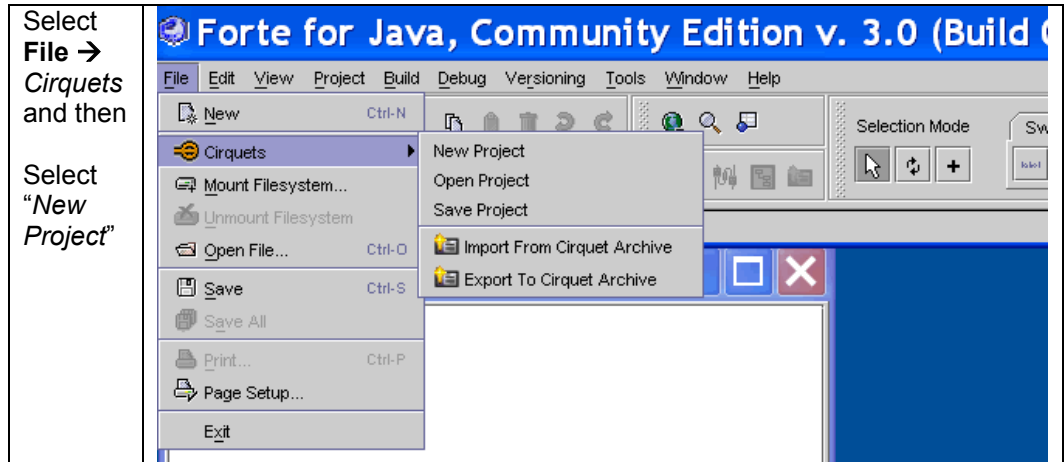


Figure 5 The Cirquet Design Studio "New Project" menu

When "New Project" is selected, a Cirquet Runtime userID and password prompt may appear. Respond to this prompt to login. This prompt will not appear if you have already logged in to the Cirquet environment.

3.0 CIRQUET BASICS STEP-BY-STEP

When the Cirquet Design Studio “New project” is requested, the Cirquet Inspector and Cirquet Palette appear on your desktop.

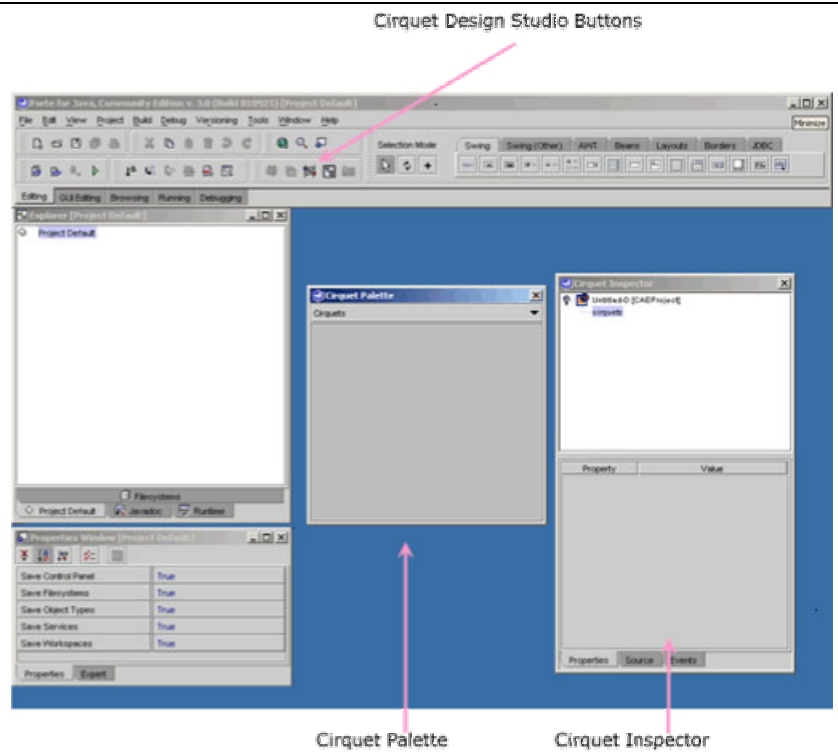


Figure 6: The Cirquet Design Studio Desktop in Sun ONE Studio

3.0 CIRQUET BASICS STEP-BY-STEP

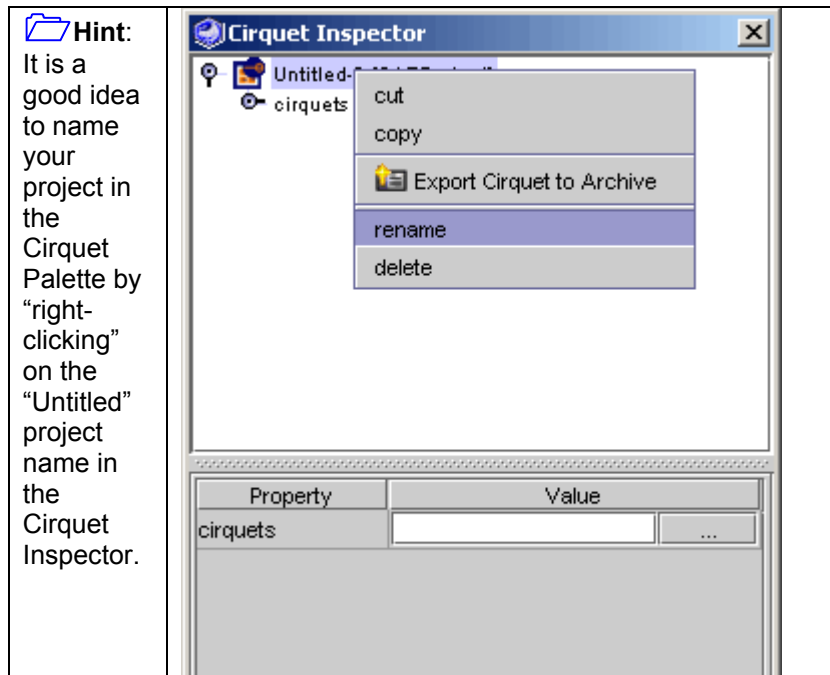


Figure 7: Naming the project

2.4 Saving a Cirquet Project

In order to come back to the sample exercise later in this course, we'll save it.

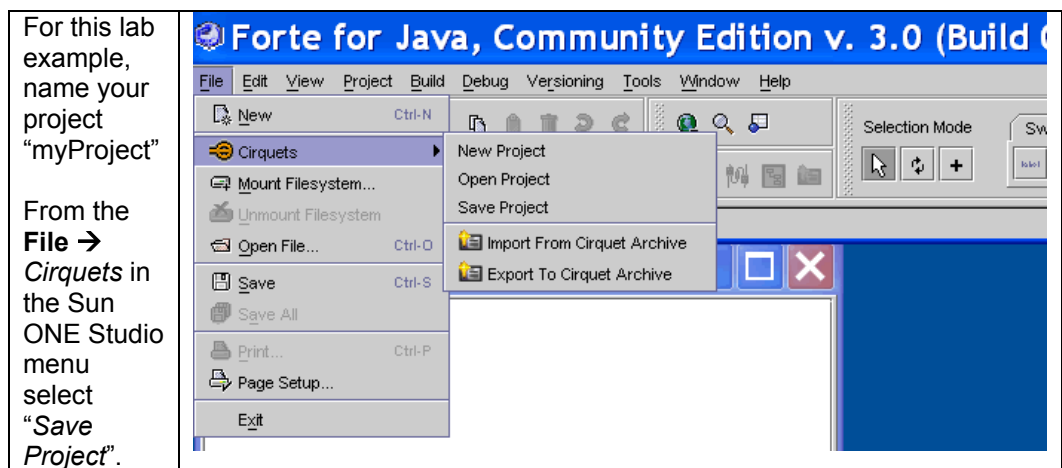


Figure 8 Saving a Cirquet Project

2.5 Opening an Existing Project

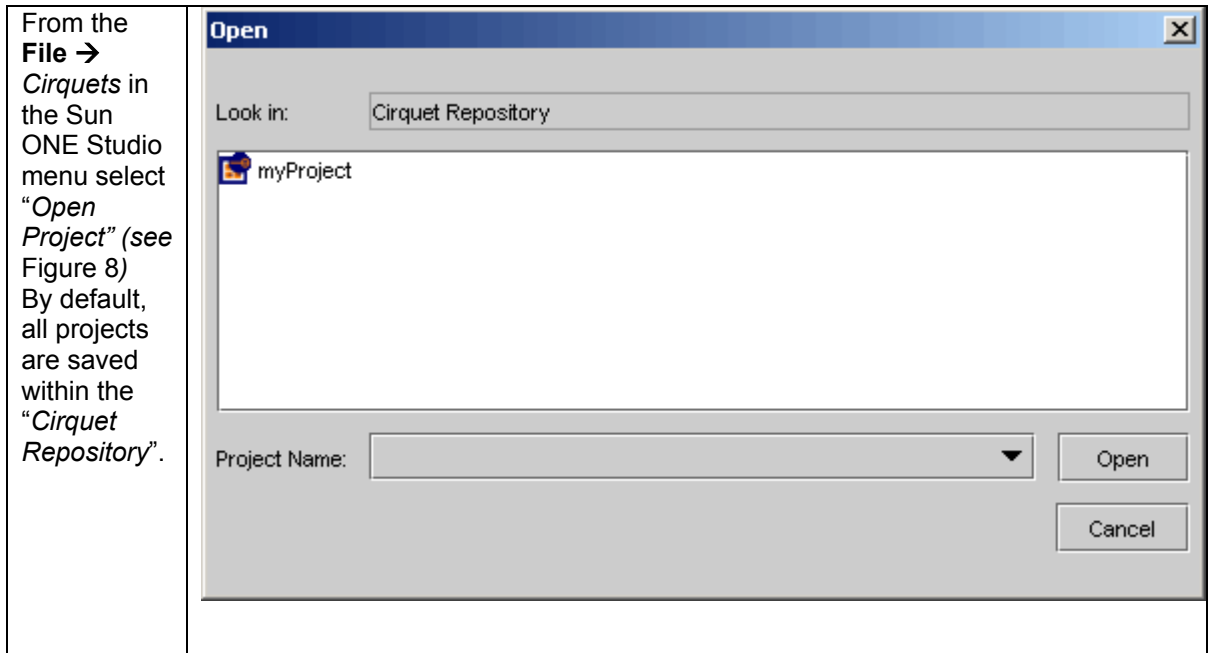


Figure 9 Opening a Project

3.0 CIRQUET BASICS STEP-BY-STEP

Once the project is opened, it will appear in the "Cirquet Inspector".

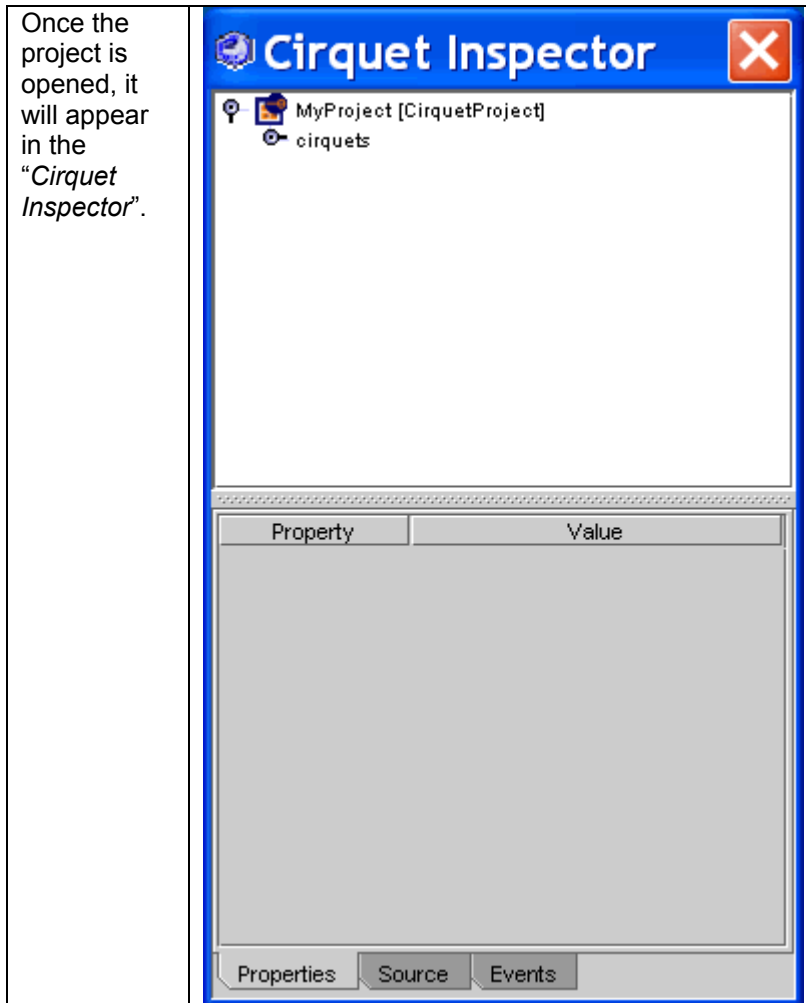


Figure 10 Project Displayed in Cirquet Inspector

2.6 Your First Cirquet Project

In this chapter, a simple set of Cirquet Components will be used as an example to see how these components are implemented and deployed.

2.6.1 Example Overview

This first example is named “ClickBean-CountBean”. The premise here is simple. Two Java Beans have already been created:

1. “ClickBean” is a simple push-button control
2. “CountBean” is a counter.

This example will explain how to configure and assemble two very simple JavaBeans as Cirquet Components that can then be run on different machines while maintaining their interaction¹.

These components have been compiled and created as a JAR file. The next section contains a review of the code used in this example.

These beans are written as ordinary JavaBeans. The Cirquet Solution System’s SmartWrap converts them into Cirquet Components.

The end result will function as shown in the following slide:

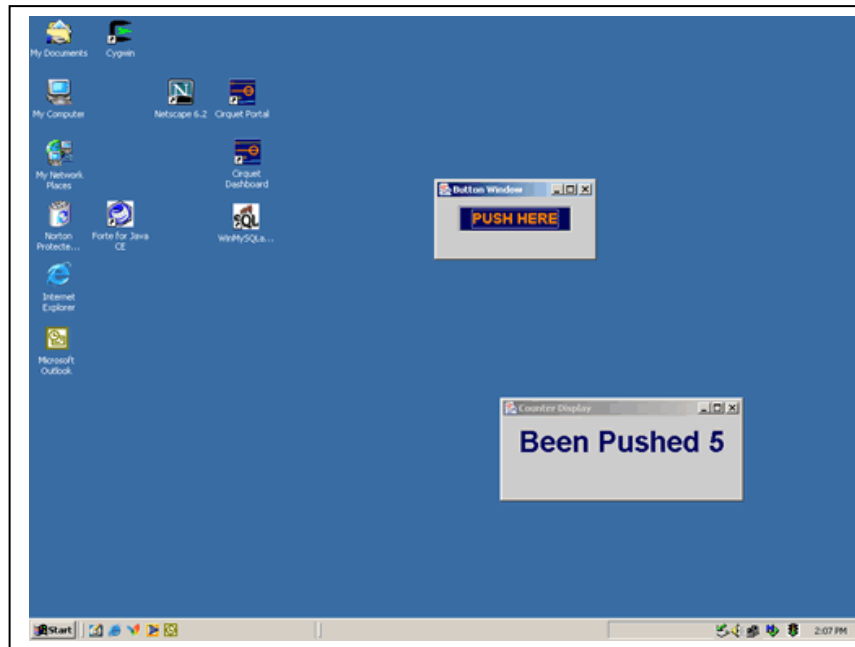


Figure 11 ClickBean and CountBean running after 5 clicks

¹ This example is based on the BeanExample directory in the examples directory of the Cirquet installation.

2.82 Explanation of Example Code

The first Bean is a window containing a button, which is labeled “Push Here”. This bean is called the ClickBean.

The second Bean is a window containing a text label and a text field that displays a counter. This bean is called the CountBean.

These beans will be assembled so that each mouse click on the ClickBean causes the counter in the CountBean to be incremented by 1. These beans are written as ordinary JavaBeans. The Cirquet Solution Suite’s SmartWrap converts them to Cirquet Components. Their interfaces are shown in Figure 12 and Figure 13.

```
/**
 * This software is the proprietary information of Improv Technologies,
 * Inc.
 * Use is subject to license terms.
 *
 * Copyright 1999-2002 Improv Technologies, Inc. All Rights Reserved.
 */

import java.awt.event.ActionListener;

public interface ClickBeanInterface {

    // to manage listeners for events:
    public void addActionListener( ActionListener ml );
    public void removeActionListener( ActionListener ml );

    // to start the Bean:
    public void launch();

    // to stop the Bean:
    public void dispose();
}
```

Figure 12: The ClickBean Interface

3.0 CIRQUET BASICS STEP-BY-STEP

```
/**
 * This software is the proprietary information of Improv Technologies, Inc.
 * Use is subject to license terms.
 * Copyright 1999-2002 Improv Technologies, Inc. All Rights Reserved.
 */

import java.awt.event.ActionListener;
public interface CountBeanInterface extends java.awt.event.ActionListener {

    // from ActionListener:
    public void actionPerformed(java.awt.event.ActionEvent evt);

    // to directly accept an event:
    public void acceptEvent();

    // the label property:
    public String getLabel();
    public void setLabel(String newlabel);

    // to start the Bean:
    public void launch();

    // to stop the Bean:
    public void dispose();
}
```

Figure 13: The CountBean Interface

2.6.1.1 Launchable

Cirquet Components can be run either locally or remotely. For a Cirquet to be launchable it must implement the `Launchable` interface class.

The `Launchable` interface specifies two methods:

```
public void launch()
public void dispose()
```

The `SmartWrap` conversion of a `JavaBean` into a `Cirquet Component` implements `Launchable`. The `JavaBean` should supply `launch` and `dispose` methods that take the appropriate actions. Both `ClickBean` (Figure 12) and `CountBean` (Figure 13) have launch methods. Launch methods make their windows visible; Dispose methods hide the windows.

2.7 Connecting and Configuring Cirquet Components

Cirquet Components are configured by setting their properties². Properties are defined by having public *get* and *set* methods. Property values can include things such as text, colors, and icons to customize the look of a Cirquet Component.

Cirquet Components can also be links to other Cirquet Components. This linking of Cirquet Components permits them to know about one another in order to communicate.

2.7.1 Connecting via Properties

Cirquet Components are linked together directly by their property values.

In this first example, the *CountBean* bean has a property called *Label*, which is used to set the text label in its window. This property is defined by the paired *getLabel* and *setLabel* methods in Figure 13.

2.7.2 Connecting via Events

A Cirquet Component can be designed to fire specific events under specific circumstances (much like a *JavaBean* button can fire a mouse clicked event when appropriate).

2.7.3 Source and Target Cirquet Components

The source Cirquet Component is referred to as the source of the event. In this example, the *ClickBean* will be the source Cirquet Component. It defines an event of interest to *ActionListeners* (which merely exposes the *ActionListener* events in its contained button). This event is defined by the paired *addActionListener* and *removeActionListener* methods in Figure 12.

The target Cirquet Component responds to the source Cirquet Component. This response may be to set a property or to call a public method in the target. In this example, the *CountBean* will be the target Cirquet Component. Its interface (Figure 13) has a public method called *acceptEvent*, which increments the counter and updates the display of the count. In addition, it implements the *java.awt.event* interface

² Some properties need not be set until deployment time.

3.0 CIRQUET BASICS STEP-BY-STEP

ActionListener. This interface defines the generic *actionPerformed* method. In the *ClickBean*, the *actionPerformed* method simply calls the *acceptEvent* method.

Events are defined by having public add and remove methods for listeners for that event. Another Cirquet Component can be designated as the target of the event.

In this example, the assembler's job is to:

- Designate the source and target Cirquet Components
- Choose the appropriate event from the source Cirquet Component
- Choose the appropriate response in the target Cirquet Component

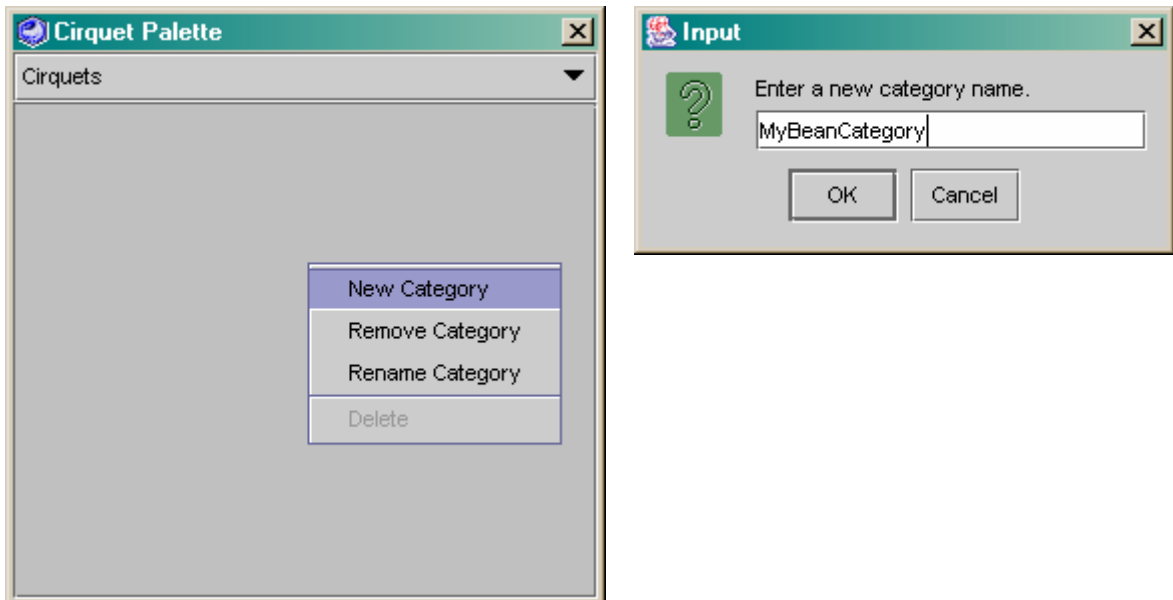


2.8 Installing Cirquet Components on the Cirquet Palette

All Cirquet Components must be put onto the **Cirquet Palette** in order to be manipulated in the Cirquet Design Studio. There is a default category called *Cirquets*, which can be used for general purposes. But it is usually preferable to group Cirquet Components into meaningful Categories. This makes them easier to find and manipulate.

2.8.1 Adding a Category to the Cirquet Palette

Right click anywhere within the **Cirquet Palette** to bring up a menu from which *New Category* can be selected. Select *New Category* to bring up a dialog box and enter a name for the new Category.



Step 1. Select the *New Category* menu item from the pop-up menu

Step 2. Name the new Category

Figure 14: Adding a Category to the Cirquet Palette

To make a Category the active category, select it from the drop-down menu at the top of the **Cirquet Palette**.

To rename an existing Category, make it the active category. Then select *Rename Category* from the pop-up menu.

2.8.2 Adding Cirquet Components to a Category in the Cirquet Palette

The **Cirquet Palette** is populated from Cirquet Components or JavaBeans contained in JAR files in the Cirquet storage directory. This is typically `<improv_home>/Cirquets/storage`. The **Tools Menu** in the main window has an *Install Cirquet Jar* item.

Select *Install Cirquet Jar* to bring up the **Install Cirquet Dialog** box. This box is used to browse the file system and choose the desired JAR file.

The data about Categories and their Cirquet Components are stored in the Cirquet Directory. Each user can view only his/her own Categories and Cirquet Components. Subsequent Cirquet Design Studio sessions show the **Cirquet Palette** containing all the Categories and Cirquet Components that were previously added (and not yet removed) by that user.

Step 1. Select *Install Cirquet Jar* from the **Tools Menu**

Step 2. Select the JAR file in the **Install Cirquet** dialog box

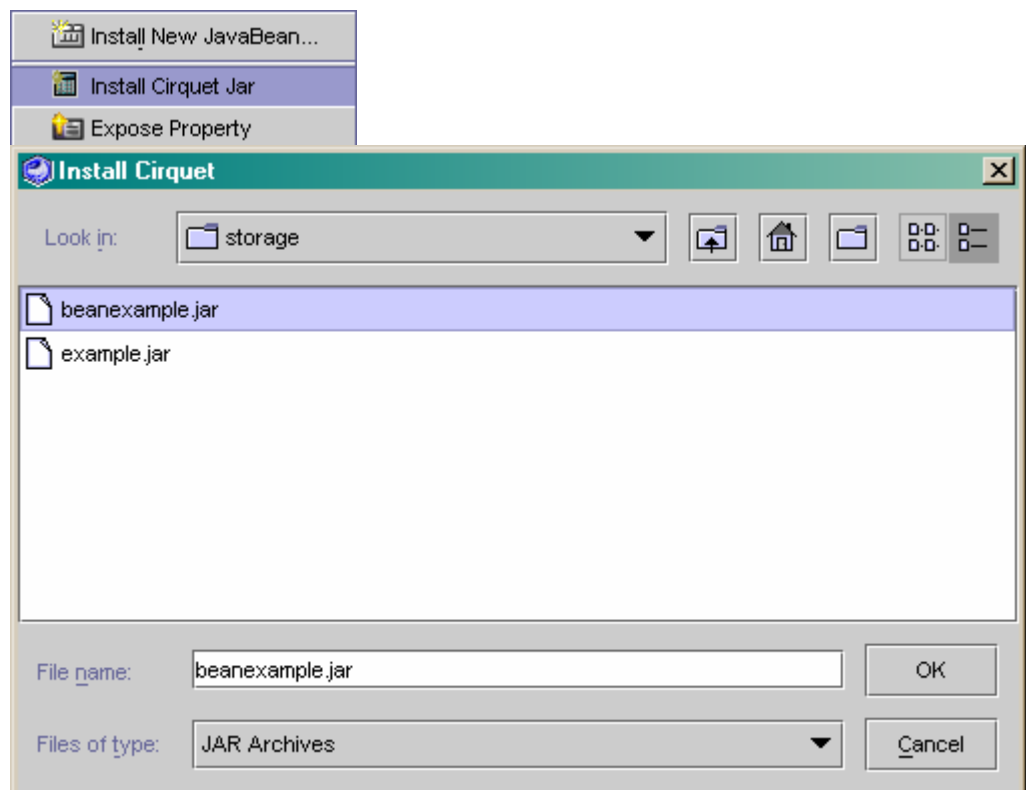


Figure 15: Installing a Cirquet JAR on the Cirquet Palette

2.8.3 Add Beans to the Cirquet Palette (Exercise)



To continue with the exercise:

- Copy the *beanexample.jar* file from the *cirquet/examples/BeanExample* directory to the *cirquet/storage* directory. This storage directory holds the code for all Cirquet Components that are used in the Cirquet application.
- Create a Category called “MyBeanCategory” on the **Cirquet Palette**.
- Populate “MyBeanCategory” with the items from the *beanexample.jar* in the storage directory. This JAR contains the ClickBean and the CountBean discussed above. When they are added to the **Cirquet Palette**, SmartWrap converts them to Cirquet Components.

Cirquet Palette with the two JavaBeans from the *beanexample.jar*

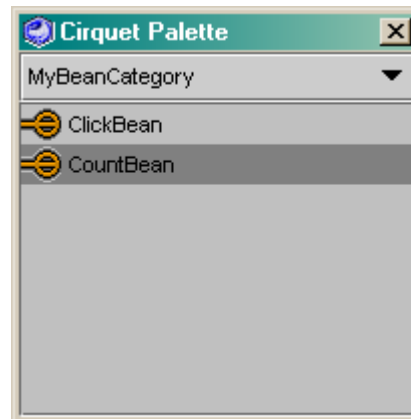



Figure 16: Populated Cirquet Palette



2.9 Adding Cirquet Components to a Project in the Cirquet Inspector

To assemble Cirquet Components, they must be added to a Project in the **Cirquet Inspector**, where they can be configured and connected. The  **Cirquet Inspector** shortcut in the main Sun ONE Studio window opens the **Cirquet Inspector** window and sets the focus on it. As described above, the **Cirquet Inspector** window for a particular Project can also be opened from the **Explorer** window.

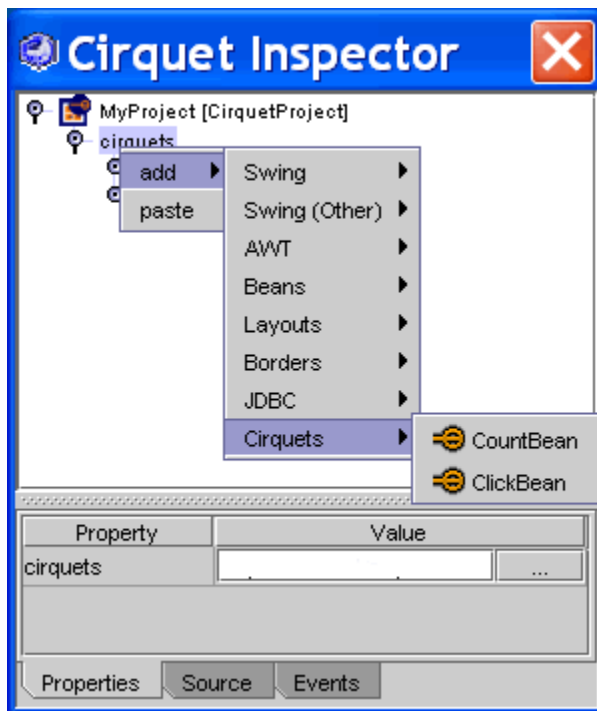
Right-click on the *cirquets* item in the **Cirquet Inspector** window to bring up a menu. Select *add* from the menu to open a cascading menu of Bean and Cirquet Component Categories. The cascading menu will include:

- All the Bean categories from Sun ONE Studio

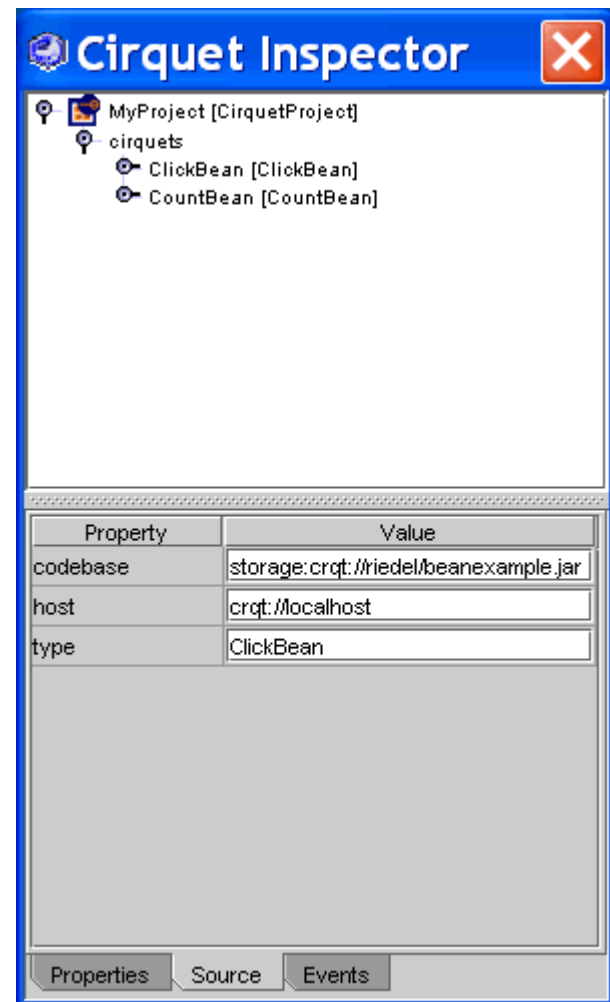
3.0 CIRQUET BASICS STEP-BY-STEP

- Any user-defined beans that have been added to Sun ONE Studio via the **Tools** → *Install New Java Bean* menu
- All populated Categories from the **Cirquet Palette**.

Select a Category to open a cascading menu of the Cirquet Components in that Category. Select a Cirquet Component to add it to the active Project in the **Cirquet Inspector** window.



Adding a Cirquet to the Project in the **Cirquet Inspector** window



The **Cirquet Inspector** window with Cirquets added

Figure 17: Adding Cirquet Components to a Project in the Cirquet Inspector



2.9.1 Adding Beans to the Cirquet Inspector (Exercise)

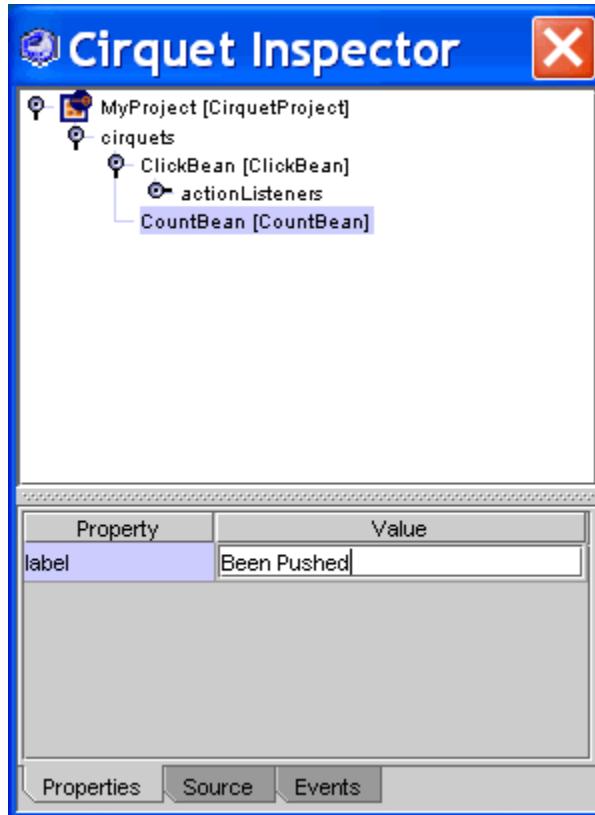
To continue with the example:

- Add the *CountBean* and *ClickBean* Cirquets from *MyBeanCategory* to *myProject* in the **Cirquet Inspector** window.

The **Cirquet Inspector** window should look like Figure 17.

2.10 Configuring a Project's Components in the Cirquet Inspector

Cirquets have properties that can be configured using the **Cirquet Inspector**. Select a Cirquet Component and then select the **Properties** tab to view the properties and enter values for them.



The *CountBean* has the SmartWrap property *label*. The *label* property's value will appear in the window of the running *CountBean*.

Figure 18: Configuring Properties in the Cirquet Inspector




2.10.1 Changing Labels (Exercise)

Continuing with the example,

- Set the `CountBean`'s *label* property to "Number of Pushes" (or similar text). Remember to hit the *Return* key after entering the value, so that Sun ONE Studio registers the entry.

The result should look like the **Cirquet Inspector** window in Figure 18

2.11 Connecting a Project's Components in the Cirquet Connection Wizard

The **Cirquet Connection Wizard** is used to specify Cirquet Component events to trigger actions in other Cirquet Components. The main Sun ONE Studio window has a toolbar shortcut for the **Cirquet Connection Wizard**: . This can also be accessed from the Sun ONE Studio "Tools" menu.

To open the **Cirquet Connection Wizard**:

1. Click on the **Cirquet Connection Wizard** shortcut.
2. Click on the Source Cirquet Component. A "source" Cirquet is the component calling a method or firing an event.
3. Click on the Target Cirquet Component. A "target" Cirquet provides the method or handles the event.

The **Cirquet Connection Wizard** starts with the **Activation Event on Source** window. This window is used to specify the event of interest from the source Cirquet Component.

3.0 CIRQUET BASICS STEP-BY-STEP

Step 1. Specify
the Source
Event

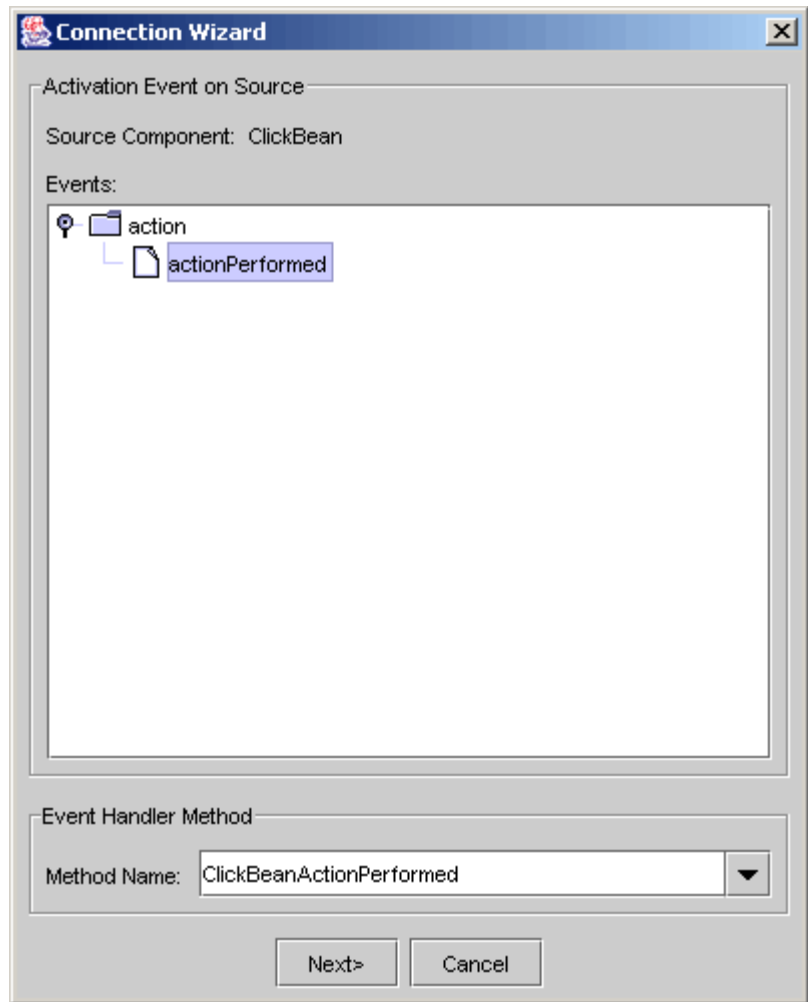


Figure 19 Specifying the Source Cirquet

3.0 CIRQUET BASICS STEP-BY-STEP

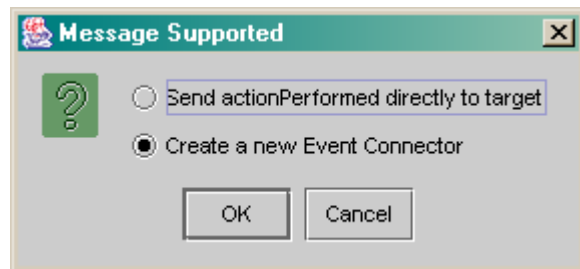
Click *Next* in that window for the **Specify Target Operation** window to appear. This window is used to specify the target Cirquet Component's reaction to the event if the target Cirquet has a method that matches the selected event. The reaction can be *Set Property* or *Method Call*.

Set Property – Sets a property on the target Cirquet Component when the source event is fired. The property value is specified in the next window of the **Cirquet Connection Wizard**.

Method Call - Calls a method on the target Cirquet Component when the source event is fired. The parameters for the method call are specified in the next window of the **Cirquet Connection Wizard**. If the method call has no parameters, the *Next* button of the **Cirquet Connection Wizard** changes to *Finish*.

If there is no matching method, the Specify Target Operation window appears.

Step 2. Specify how to send the Event Message. (This dialog box only occurs when the target has a method matching the action of the Source Event.)
Choosing the *Create a new Event Connector* radio button allows specification of any method in the target.



Step 3. Specify the Target Operation

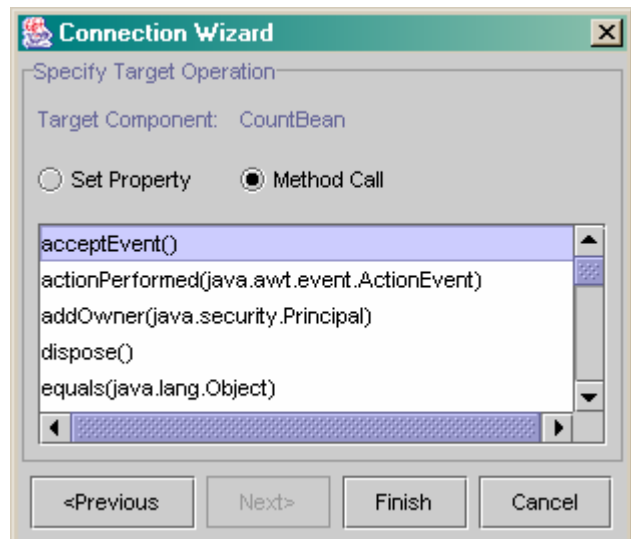


Figure 20: Specifying Source Event and Target Operation in the Cirquet Connection Wizard



2.11.1 Target Operation (Exercise)

The goal of this exercise is to have a mouse click on the `ClickBean` trigger the `acceptEvent` method to call the `CountBean`

The steps:

1. Launch the **Cirquet Connection Wizard** for these two Cirquet Components:
 - Click on the **Cirquet Connection Wizard** shortcut in the Sun ONE Studio main window
 - Select the *ClickBean* in the **Cirquet Inspector**
 - Select the *CountBean* in the **Cirquet Inspector**
2. In the **Activation Event on Source** window of the wizard:
 - Expand the action folder
 - Select the *actionPerformed* event
 - Click *Next*
 - In the resulting dialog box, select Create a new Event Connector
 - Click *OK*
3. In the **Specify Target Operation** window of the wizard:
 - Select the Method Call radio button
 - Select the *acceptEvent* method
 - Click *Finish*

Figure 20 illustrates these steps.

Alternatively, select the Send `actionPerformed` directly to target radio button at the wizard's Activation Event on Source window.. This choice is available because the target `CountBean` has an `actionPerformed` method (see Figure 20) matching the `actionPerformed` event in the source `ClickBean` (see Figure 12). With this approach, any `actionPerformed` event generated by the `ClickBean` will cause the `CountBean`'s `actionPerformed` method to be invoked, with the event as its parameter.

2.12 Specifying More Complex Targets

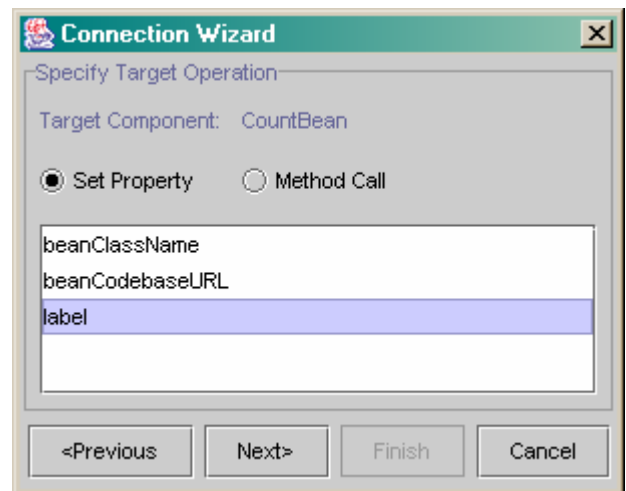
For methods with no parameters, the selection of the target Cirquet Component method is all that is necessary to finish the specification in the **Cirquet Connection Wizard**.

Properties and methods with parameters require additional steps to specify the value of the property or parameter.

In both cases, the *Next* button advances the **Cirquet Connection Wizard** to a window where the value can be specified. There are 3 ways to specify the value:

1. *Value* – Enter a fixed value into the text field. This option is available only if the value is a primitive type (int, float, and so on) or a String.
2. *Property* - Acquire the value from a property of another component. The ellipsis (...) button displays a dialog box to select the component and its property. Only properties of the correct type are listed.
3. *Message* - Acquire the value from a method call of another component. The ellipsis (...) button displays a dialog box to select the component and its method. Only methods that return the correct type and that do not take any parameters are listed.

Step 1. Select the label Property of the CountBean



Step 2. Set the label Property of the CountBean to a String value



Figure 21: Setting a Property in the Cirquet Connection Wizard



2.12.1 One More Point

The **Cirquet Connection Wizard** could also be used to change the value of the CountBean's label when the ClickBean performs an action. While this is not a particularly useful operation, it does illustrate the setting of a property. Figure 21 illustrates the steps. However, the examples that follow assume that this connection has **not** been made.

2.13 Launching the Assembled Cirquet Components from the Cirquet Inspector

Right click on the Cirquet Component to be launched to open a menu. Select *launch* from this menu to bring up the **Launch Cirquet** dialog box to specify where to run the Cirquet. The default is set to run locally. If a different machine is desired, set the *host* property in the *Source* tab of the **Cirquet Inspector**.

Step 1. Specify where to run the Cirquet. The default will run it locally.

To run it remotely, change the *host* property to the name of the desired machine.

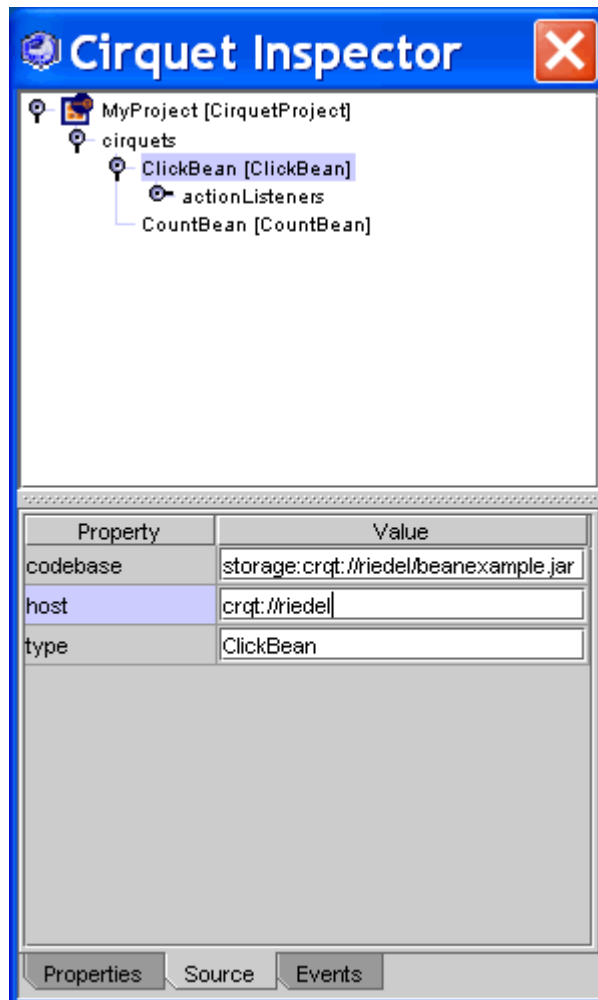
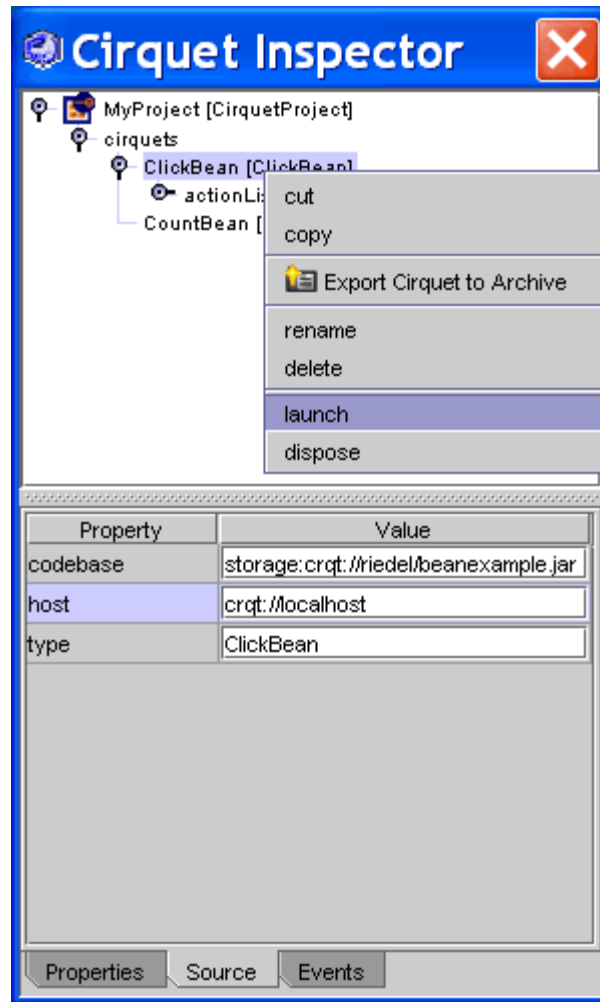


Figure 22 Complete Cirquet with properties

3.0 CIRQUET BASICS STEP-BY-STEP

Step 2. Select *launch* from the Cirquet's right-click menu.



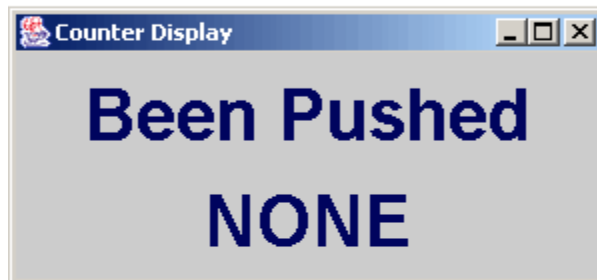


2.13.1 Launching the Examples

The `ClickBean` and the `CountBean` have been configured and connected in the steps above. They can now be launched and observed to work.



immediate ClickBean window



immediate CountBean window



CountBean after 5 "clicks" of ClickBean

Figure 23: The CountBean and the ClickBean



2.13.2 Designating Remote Targets

Once a developer has created the beans, and created a working Cirquet project, the components can be deployed to other peers within the Cirquet network.

Take a look at Figure 22. Change the “host” setting for the CountBean to another peer in the network. The instructor will provide the class with machine names that can be used.

Be sure to hit *Return* after changing any properties to ensure they are recognized by the Sun ONE Studio.



2.14 Launching – Review

Cirquets can be run either locally or remotely. In order for a Cirquet to be launchable it must implement the Launchable interface class, which specifies two methods:

```
public void launch()  
public void dispose()
```

The SmartWrap conversion of a JavaBean into a Cirquet Component implements `Launchable`; the JavaBean should supply `launch` and `dispose` methods that take the appropriate actions. Both `ClickBean` and `CountBean` have `launch` methods, which make their windows visible; their `dispose` methods hide the windows.

The default is to run a Cirquet locally; a different machine can be specified by setting the `host` property in the `Source` tab of the Cirquet Inspector. Right clicking on the Cirquet to be launched opens a menu; selecting `launch` from this menu launches the Cirquet.

3 Exporting the Assembled Cirquet Components to a CAR file

Once an assembly project is complete it can be exported to a Cirquet Archive file (CAR file). A CAR file is similar to a JAR file.

The steps:

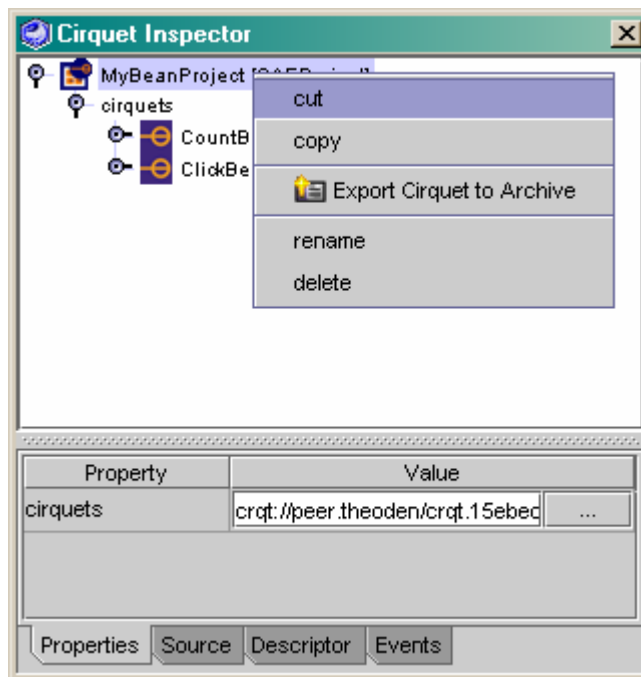
- Right-click on the Project in the **Cirquet Inspector**
- Select *Export Cirquet to Archive*
- Click Yes to “Export Dependent Cirquets?”

Note: This will save each Cirquet’s configuration to the CAR file. The associated JAR files of those components will be included in the CAR file as well.

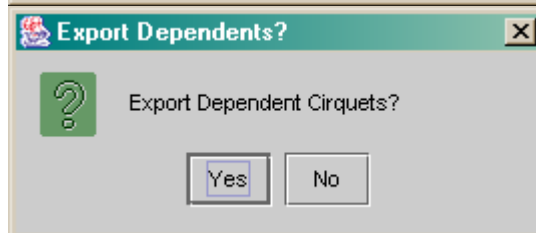
- Select the directory and filename for the CAR file

These steps are demonstrated in Figure 24.

Step 1. Right-click on the Project in the Cirquet Inspector and select *Export Cirquet to Archive*



Step 2. Answer Yes to “Export Dependent Cirquets?”



3.0 CIRQUET BASICS STEP-BY-STEP

Step 3.
Select the
directory and
file name for
the CAR file

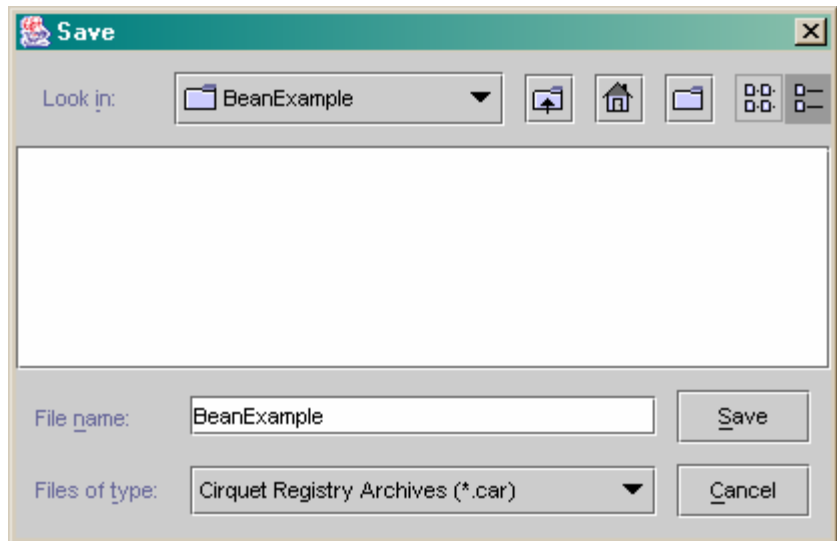


Figure 24: Exporting a Cirquet Project to a CAR file

An entire working Cirquet Directory can be exported by choosing *Export To Archive* from the *Cirquets* item in the **File** menu. The steps are shown in Figure 24. This action creates a CAR file containing the assembler's entire Cirquet Directory. The resulting CAR file can be used to transfer work from the assembly stage to the deployment stage.



4 *Cleaning the Assembly Environment*

If the Cirquet Design Studio is being run with its own directory, cleaning it after a project is completed can simplify future work. Visual clutter will be eliminated by deleting items that are no longer needed. The following items can be removed from the environment when they are no longer relevant:

1.2 **Cirquet Palette Categories**

To delete a Cirquet Palette category from the **Cirquet Palette**

- Make it the active category
- Right click anywhere on the **Palette**
- Select *Remove Category* from the pop-up menu.

1.3 **Cirquet Palette Items**

To delete a Cirquet Palette item:

- Select it
- Right click anywhere on the **Cirquet Palette**
- Select *delete* from the pop-up menu.

1.4 **Assembly Projects**

To delete a project

- Right click on the project in the **Explorer** window
- Select *Unregister*.

1.5 **Registered Cirquet Components**

A Cirquet Component can be unregistered only after all projects using it have been unregistered. To unregister a Cirquet Component:

- Expand the *Cirquets* item in the *Cirquet Directory*
- Right click on the Cirquet Component in the **Explorer** window
- Select *Unregister*



5 Working with Multiple Assemblers

Some projects may have multiple assemblers who need to cooperate and share their work. Work can be shared via `CAR` files, or by sharing a Cirquet Directory.

To transfer a project from one assembler to another in a `CAR` file.

The first assembler:

- Exports the project, as shown in Figure 24

The second assembler:

- Imports the resulting `CAR` file
 - Select *Import Directory From Archive* from the *Cirquets* item in the **File** menu.

The projects in the `CAR` file and their Cirquet Components are added to the second developer's Cirquet Directory. They will appear in the **Explorer (Runtime)** window under *Cirquet Projects*.

3.0 CIRQUET BASICS STEP-BY-STEP