Universität Leipzig
Fakultät für Mathematik und Informatik
(Institut für Informatik)

Erstellung von PL/1 Tutorials mit dem Rational Developer for System z

Betreut von
Prof. Dr.-Ing. Wilhelm G. Spruth
Vorgelegt von
Karsten Kunze
Matrikelnummer: 8962610
an der
Universität Leipzig
Lehrstuhl Technische Informatik
Leipzig, den 23.07.2009

Zusammenfassung

Das Ziel der Diplomarbeit war es, PL/1 Tutorials unter dem Rational Developer for System *z* (RDz) für die akademische Lehre und Ausbildung zu erstellen und auf dem System z Großrechner der Universität Leipzig verfügbar zu machen.

Als Grundlage für die Arbeit dienten COBOL Tutorials, die bei einer IBM Veranstaltung an der Universität Hamburg von Frau Isabel Arnold im Sommer 2006 vorgestellt wurden.

Themen der Übungen sind neben einer Einführung in die Funktionsweise von RDz eine Installationsanleitung für das benötigte lokale Arbeitsumfeld, sowie lokale und remote Kompilierung von PL/1 Programmen mit RDz 7.5.

Die nötigen Dateien, Programme und Tutorials sind auf zwei DVDs vorhanden, die sich im Anhang dieser Arbeit befinden. Sie stellen eine VMware Player virtuelle Maschine dar, die auf einem Windows XP Rechner installiert werden kann.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe, insbesondere sind wörtliche oder sinngemäße Zitate als solche gekennzeichnet. Mir ist bekannt, dass Zuwiderhandlung auch nachträglich zur Aberkennung des Abschlusses führen kann.

_____ _____

Ort, Datum Unterschrift

Danksagung

Ich möchte allen Personen danken, die mich während der Diplomarbeit unterstützt haben, vor allem meinen Eltern und Großeltern, die mich während meines gesamten Studiums unterstützt haben.

Weiterer Dank gilt Herrn Dennis Behm IBM für seine Hilfe bei der Lösung mehrerer Probleme im Zusammenhang mit der Erstellung dieser Arbeit.

Besonderer Dank gilt meinem Betreuer Herrn Professor Dr. Wilhelm G. Spruth, der mich während der gesamten Zeit unterstützt hat.

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

1.1 Motivation

Im Rahmen der Vorlesungen Einführung in z/OS sowie Mainframe Internet Integration an der Universität Leipzig, findet ein begleitendes Praktikum statt.

Da zum Thema PL/1 noch kein Übungsmaterial vorhanden ist, ist die Aufgabe dieser Arbeit mehrere Tutorials zu erstellen.

Weiterhin wird ein Basiswissen über die Verwendung von Rational Developer for System z 7.5 vermittelt.

Grundlage für diese Tutorials ist ein in Hamburg abgehaltenes, zweiwöchiges Seminar über Anwendungsentwicklung und Host Modernisierung auf zSeries Rechnern von Frau Isabel Arnold von der IBM.

Die Tutorials werden voraussichtlich im Wintersemester 2009 im Rahmen des Client-Server-Praktikums an der Universität Leipzig zum Einsatz kommen.

1.2 Aufbau der Arbeit

Die erstellten Tutorials sind im Anhang der Arbeit in schriftlicher und elektronischer Form auf einer DVD beigelegt.

Im Kapitel zwei wird eine kurze Einführung in die verwendete Hardware des Servers gegeben, ein zSeries Rechner der Universität Leipzig, auf dem auch weitere schon bestehenden Übungen des Mainframe-Praktikums laufen.

Weiterhin gibt es in Kapitel drei eine Übersicht über die verwendete Software *Rational Developer for System z (RDz)* und die Entwicklungsumgebung *Eclipse*.

In Kapitel vier werden die im Rahmen der Diplomarbeit entwickelten Tutorials vorgestellt.

Das Kapital fünf gibt eine Zusammenfassung und einen Ausblick auf mögliche weitere Studien- oder Diplomarbeiten, die auf der verwendeten Arbeitsumgebung aufbauen könnten.

Die erstellten Tutorials sind im Anhang der Arbeit wiedergegeben.

Die beiden beigefügten DVDs enthalten alle schriftlichen Texte, alle elektronisch verfügbaren Literaturreferenzen, sowie alle für die Durchführung der Tutorials erforderliche Software.

2. Hardware: zSeries

Die zSeries (zwischenzeitlich auch als System z bezeichnet) ist die aktuelle Generation an Großrechnern der *International Business Machines Corporation* (IBM).

Die Firma IBM blickt auf eine äußerst erfolgreiche sowie richtungweisende Geschichte ihrer Großrechner zurück.

Mit dem System/360 brachte IBM 1964 den ersten Rechner auf den Markt, der die bis dahin getrennten Ausrichtungen der Computer in wissenschaftliche und kommerziell genutzte Systeme auf einen gemeinsamen Nenner brachte.

Möglich wurde diese allgemeine Ausrichtung durch die Einführung von Microcode, der die bis dahin fest verdrahteten Programme ablöste und den Rechner dadurch für die unterschiedlichsten Anwendungen öffnete.

Viele der ersten Programme, die in *Assembler, Fortran, Cobol* und *PL/1* geschrieben wurden sind heute noch in Gebrauch.

Die Abwärtskompatibilität der aktuellen gegenüber den älteren Modellen ist ein weiterer Meilenstein der IBM Großrechner Geschichte.

Um mit der rasanten Entwicklung der Computerindustrie Schritt zu halten, wurden regelmäßig neue Versionen von Großrechnern auf den Markt gebracht, die jeweils kompatibel zu ihren Vorgängern waren, gleichzeitig durch mehr und schnellere Prozessoren, mehr Arbeitsspeicher und neuen Automatismen zur Fehlererkennung und -behebung an das neue Arbeitsumfeld angepasst waren.

2.1 Eigenschaften der zSeries

Im Folgenden sollen einige der herausragenden Eigenschaften der zSeries hervorgehoben werden, ohne jedoch stark ins Detail zu gehen, da dies den Rahmen dieser Arbeit sprengen würde.

2.1.1 Abwärtskompatibilität

Die aktuellen Modelle der zSeries sind kompatibel zu allen älteren Großrechnergenerationen.

Deshalb ist es möglich, Jahrzehnte alten Code ohne Änderungen auf neuen Großrechnern auszuführen. Eine Neukompilierung der Programme ist nicht nötig.

Da es nicht produktiv ist, Abwärtskompatibilität auf Microcodeebene zu erhalten, werden bei der zSeries selten genutzte Maschinenbefehle, die nur dem Erhalt der Kompatibilität des Binärcodes dienen, durch den so genannten *Licenced Internal Code* (LIC) nachgebildet [6].

Diese Softwarekomponente vermittelt zwischen Hardware und Software, nur sie kann die Hardware direkt ansprechen.

LIC stellt Anwendungen alle nötigen Schnittstellen zur Verfügung. Dadurch können neue Programme von neuen, optimierten Befehlssätzen profitieren und ältere Programme bleiben lauffähig.

2.1.2 Virtualisierung

Unter Virtualisierung versteht man Methoden, durch die Computerressourcen aufgeteilt werden können.

Übertragen auf die zSeries bedeutet dies, dass mehrere Betriebssysteme gleichzeitig nebeneinander ausgeführt werden können, ohne sich gegenseitig zu beeinträchtigen.

Grundsätzlich bietet die zSeries zwei unterschiedliche Ansätze hierfür:

- das System kann in so genannte *Logical Partitions* (LPARs) aufgeteilt werden,
- mit z/VM steht ein Betriebssystem bereit, welches weitere Virtualisierungs-mechanismen anbietet [8].

2.1.3 Ausfallsicherheit

Die zSeries gilt aufgrund ihrer Architektur als äußerst ausfallsicher.

IBM nennt für einzelne Rechner eine Verfügbarkeit von 99.999%, dies ist besonders bei unternehmenskritischen Anwendungen von Bedeutung, da hier jede Minute Ausfallzeit bis zu 10 000$ kosten kann [2].

2.1.4 Skalierbarkeit, Parallel Sysplex und Coupling Facility

Im laufenden Betrieb können sich Anforderungen an die Hardware schnell ändern.

Damit es zu keinen langen Wartezeiten bei Anfragen oder zu Ausfällen durch Überlastung kommt, kann die zSeries flexibel auf geänderte Anforderungen reagieren.

Sind noch nicht alle Prozessoren einer zSeries aktiviert, können dynamisch weitere Prozessoren hinzugenommen werden um Lastspitzen auszugleichen.

Es können auch ganze Großrechner zugeschaltet werden, die einen als *Parallel Sysplex* bezeichneten Cluster –Betrieb ermöglichen.

Diese Cluster wirken nach außen wie ein einzelner Rechner.

Die Koordination zwischen den einzelnen Rechnern übernimmt die so genannte *Coupling Facility* (CF), eine Hardwarekomponente, die entweder Teil eines der im Verbund befindlichen Rechners ist, oder auf einem separaten Mainframe läuft.

Da der Verwaltungsaufwand für das Cluster auf die CF beschränkt ist, steigt die Leistung eines Clusters fast linear mit den eingebundenen Knoten.

2.1.5 Workloadmanager

Der *Workloadmanager* (WLM) ist Teil des z/OS Betriebssystems und ermöglicht eine dynamische Verteilung der Ressourcen zwischen einzelnen so genannten Dienstklassen *(Service Classes)*.

Hier können verschiedene Prioritäten und Zielvorgaben definiert werden.

Der WLM kümmert sich automatisch um eine Optimierung der Lastverteilung und berechnet die Verteilung kontinuierlich neu, indem er Daten über vergangene Ereignisse sammelt und auswertet.

Um dabei ein möglichst optimales Ergebnis zu erzielen, kommen verschiedene Lösungsansätze zum Einsatz [7].

Unter anderem wird im IBM Labor in Böblingen an einem WLM geforscht, dem ein neuronales Netzwerk zugrunde liegt.

3. Software: Eclipse und Rational Developer for System z

Rational Developer for System z (RDz) ist eine integrierte Entwicklungsumgebung *(integrated developement enviroment, IDE)* für Großrechneranwendungen.

RDz setzt auf die Open Source Platform *Eclipse* auf, deren gleichnamige Java-IDE laut [1] von über 50% der Java-Programmierer genutzt wird.

3.1 Eclipse

Eclipse ist eine Open Source Plattform zur Entwicklung von Software [4].

Besonders bekannt ist Eclipse für seine Entwicklungsumgebung für die Programmiersprache Java.

Es gibt eine Vielzahl an Plugins für weitere Programmiersprachen, unter anderem *C, C++, Perl, PHP, Ruby* und *Python*.

Eclipse basiert auf der IDE *Visual Age for Java 4.0*, die von IBM entwickelt wurde.

Im November 2001 wurde der Quellcode für das Programm freigegeben und seitdem kontinuierlich als Open Source Projekt erweitert.

Das von IBM geleitete *Eclipse Konsortium* umfasst über 80 Mitglieder neben IBM sind unter anderem Borland, MERANT, QNX Software Systems, Rational Software, Red Hat, SuSE, TogetherSoft und Webgain vertreten. Das Konsortium gründete im Januar 2004 die rechtlich unabhängige *Eclipse Foundation*, die seitdem für die Entwicklung von Eclipse verantwortlich ist [15].

Eclipse basiert auf einem Prinzip, bei dem sich alles als Plugin integrieren lässt.

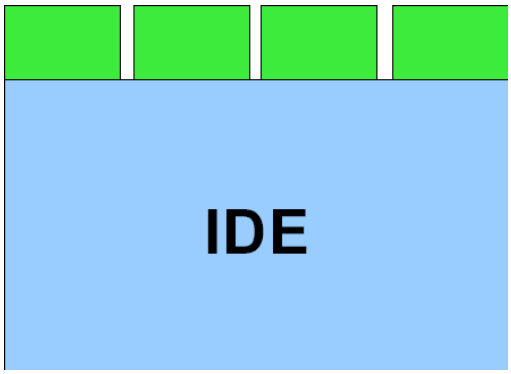


Abbildung 3.1.1 Eclipse als Entwicklungsumgebung

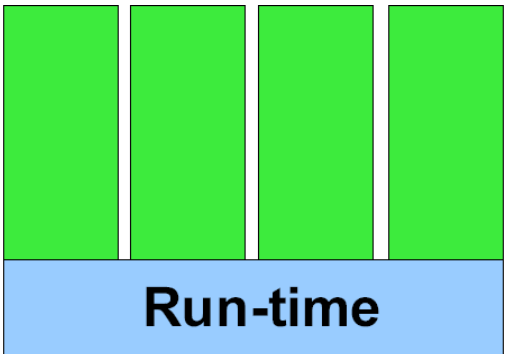


Abbildung 3.1.2 Eclipse als Run-time Enviroment

Die Eclipse Open Source Community versteht sich als Gemeinschaft, die eine kostenlose, erweiterbare Entwicklungsumgebung und ein Framework für die Laufzeit und Anwendungsentwicklung zur Verfügung stellt, welches die Entwicklung, Betreuung und Wartung eines Softwareprojekts über dessen gesamte Lebensdauer begleitet.

Die Community stellt über 60 verschiedene Open Source Projekte bereit, die in sieben Kategorien unterteilt werden [5]:

- Enterprise Development
- Embedded und Device Development
- Rich Client Platform
- Rich Internet Applications
- Application Framework
- Application Lifecycle Management (ALM)
- Service Oriented Architecture (SOA)

3.2 Rational Developer for System z

RDz beschleunigt die Entwicklung von Webanwendungen, konventionellen *COBOL*- und *PL/I*-Anwendungen, Web-Services und XML-basierten Schnittstellen.

RDz enthält wichtige Erweiterungen in Version 7.5, die die Erstellung konventioneller Mainframeanwendungen und Web-Services noch effizienter gestalten und steigert somit die Produktivität von Entwicklern bei der Erstellung und Wartung von Mainframeanwendungen und Web-Services, während und nach der Umstellung auf eine SOA-basierte Umgebung.

Neben Werkzeugen, die der schnelleren Entwicklung dynamischer Webanwendungen, *Java* und *J2EE* dienen, gibt es damit erstmals die Möglichkeit, dieselben Werkzeuge für traditionelle *COBOL*- und PL/I-Programmierung auf Großrechnerebene zu verwenden.

Rational Developer for System z ist der Nachfolger des WebSphere Developer for System z und baut als Plugin auf die Eclipse IDE auf.

In dem Plugin enthalten sind die beiden IBM Rational-Produkte:

- *Rational Developer for System z with Java*
  (Unterstützung für die Entwicklung von Mainframe- und J2EE-Anwendungen)
- und *Rational Developer for System z with EGL*
  (Unterstützung für die konventionelle Mainframeentwicklung und Anwendungsentwicklung mit EGL). [9]

Weiterhin umfasst die Rational-Software Werkzeuge zur Integration, Realisierung und zum Testen entwickelter Anwendungen.

Für weiterführende Informationen sei [12] empfohlen.

Das *Rational Developer for System* z-Plugin als umgebender Rahmen des Gesamtmoduls ist für die z/OS Anwendungsentwicklung, XML Services, *Basic Mapping Support* (BMS) Map Editor, COBOL und PL/I DB2 Stored Procedures und die EGL COBOL Funktionalität verantwortlich.

Im Folgenden werden die oben aufgeführten Kategorien genauer betrachtet.

3.2.1 z/OS Application Development

Mit RDz ist es möglich, sich auf ein z/OS System einzuloggen und mit den auf dem z/OS System liegenden Daten zu arbeiten, als ob es Dateien der Workstation wären.

Dies wird durch das so genannte *z/OS File System Mapping* ermöglicht, bei dem auf der Workstation zusätzlich zu einzelnen Datasets auch spezielle Member dieser Datasets gemappt werden können.

Das ermöglicht es, mit Datasets zu arbeiten, die Member unterschiedlichen Typs beinhalten. Die Daten werden von der Workstation gemäß ihrer Mappingkriterien behandelt.

Von der Workstation aus können auch neue Datasets und Member angelegt und bearbeitet werden.

Der bei der Bearbeitung der Dateien verwendete Editor hat volle ISPF *(Interactive System Productivity Facility)* Funktionalität und ist durch eine Vielzahl weiterer Module auf dem Stand aktueller Entwicklungsumgebungen.

Hervorzuheben sind hierbei ein *Content Assist* für *COBOL* und *PL/I*, der bei Syntaxvervollständigung automatisch alle Ressourcen integriert, auf die ein Entwickler Zugriff hat.

Weitere Funktionen sind lokale und remote Syntaxprüfung sowie die in Eclipse integrierten Werkzeuge *Compare With...* und *Replace with Local History*.

Ein weiterer Teil von RDz ist die Interaktion mit dem *Job Entry System* (JES), bei dem Jobs an den Großrechner übermittelt, überwacht und deren Ergebnisse betrachtet werden können.

Die dabei verwendeten *Job Control Language*-Files (JCL-Files) für compile, link-edit und run können automatisiert aus dem vorhandenen Quellcode generiert und an das entsprechende z/OS System zur Ausführung gesandt werden.

Generell sind alle typischen Editier-, Kompilier- und Debugfunktionen lokal und auf dem remote z/OS System von der Workstation aus verfügbar.

Die meisten Funktionen können auch im Offline-Modus verwendet werden. Hier ist als Voraussetzung nötig, entsprechende Projekte und Daten offline verfügbar zu machen, wofür eine Routine in RDz bereit steht.

Um Änderungen lokal und offline testen zu können, ist für die Testumgebung der Workstation ein CICS Transaktionsserver in RDz integriert, der CICS-Anweisungen lokal übersetzt und dadurch ein Testen ermöglicht. Hier gibt es jedoch seitens der IBM den Hinweis, ausschließlich lokal getestete z/OS Anwendungen nicht ohne weitere Prüfung auf einem Großrechner in die Produktion aufzunehmen.

3.2.2 XML Services for the Enterprise

In diesem Abschnitt werden der Zugang von *Service orientierter Architektur* (SOA) auf *CICS V3.1* und *IMS V9 COBOL*-Anwendungen, *COBOL* zu XML-mapping, COBOL XML Konverter und WSDL Erstellung behandelt.

Neben dem Zugriff auf CICS Anwendungen über WebSphere können CICS Anwendungen über das *Simple Object Access Protocol* (SOAP) auf den CICS Transaktionsserver V2 zugreifen.

Der Nachrichtenaustausch der dabei verwendeten, in XML kodierten Nachrichten kann dabei über HTTP oder WebSphere MQ erfolgen.

Durch die Verwendung von SOAP können so CICS-basierte Anwendungen als Web Services verfügbar gemacht werden, auf die Clients ohne einen dazwischen liegenden Anwendungsserver zugreifen können. Weitere Information bezüglich der *SOAP for CICS* Funktionalität ist auf [13] aufgeführt.

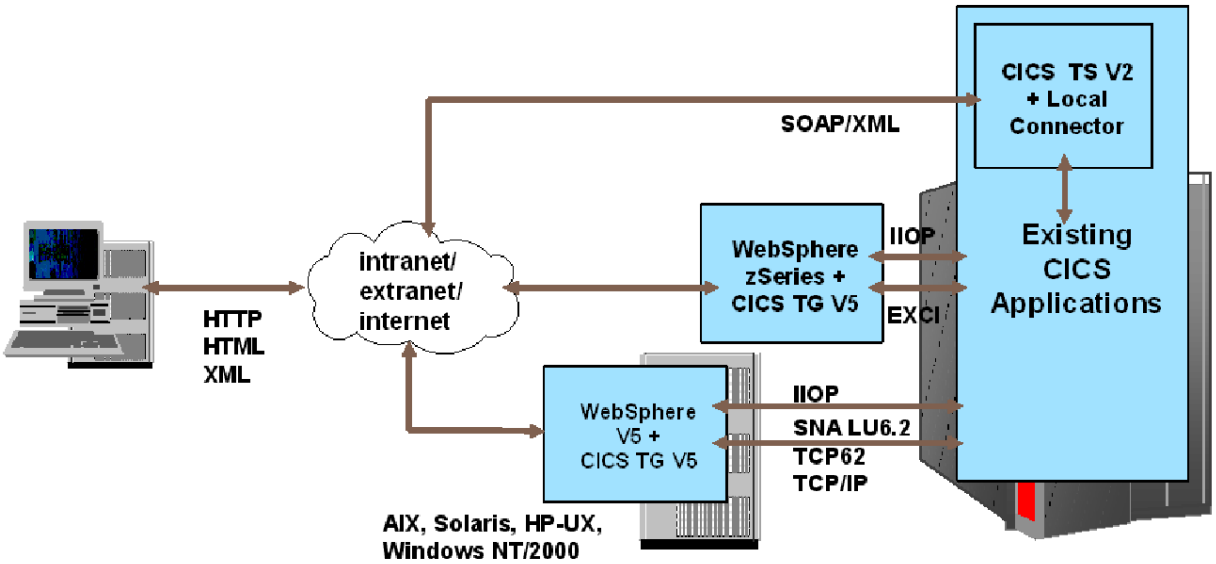


Abbildung 3.2.1 Übersicht externer Zugriffsmöglichkeiten auf CICS Anwendungen

Ähnlich kann das *Information Management System* (IMS) von IBM in eine SOA-Umgebung integriert werden. Für eine Ausführung über das *IMS SOAP Gateway* sei auf [10] verwiesen.

XML Enablement for the Enterprise ermöglicht es COBOL-basierten Anwendungen, aus XML Nachrichten COBOL-Daten zu generieren, und andersherum aus COBOL-Daten XML Nachrichten zu erstellen.

Die dabei verwendeten Konverter nutzen die hochperformanten XML Parserfähigkeiten des *IBM Enterprise COBOL Compilers* [11].

RDz bietet mehrere Möglichkeiten, aus bestehendem Code Web Services zu generieren und diese mit dem integrierten *Web Services Explorer* zu testen.

3.2.3 BMS Map Support

Ein weiterer Teil von WebSphere Developer for System z ist die Erstellung und Bearbeitung des IBM *Basic Mapping Support*s (BMS).

Der dabei verwendete Editor ermöglicht die Bearbeitung der BMS Maps über ein *Drag & Drop*-Verfahren.

Dargestellt und bearbeitet werden können die BMS Maps in einem *Design View*, bei dem direkt in eine angezeigte BMS Map Teile eingefügt werden können, oder klassisch als *Source View* mit Sourcecode Bearbeitung.

Es besteht die Möglichkeit, neue Map Sets zu erstellen oder Bestehende zu importieren.

Alle Arbeiten können lokal oder remote ausgeführt, sowie fertige Maps exportiert werden.

3.2.4 DB2 Stored Procedures – COBOL / PL/I

Mit RDz können COBOL und PL/I Stored Procedures unter z/OS erstellt, getestet und im Bedarfsfall auf der Workstation debugged werden.

Für die automatisierte Generierung der SQL Definitionen und der COBOL und PL/I Stored Procedure Programme steht ein Wizard als Teil von RDz zur Verfügung.

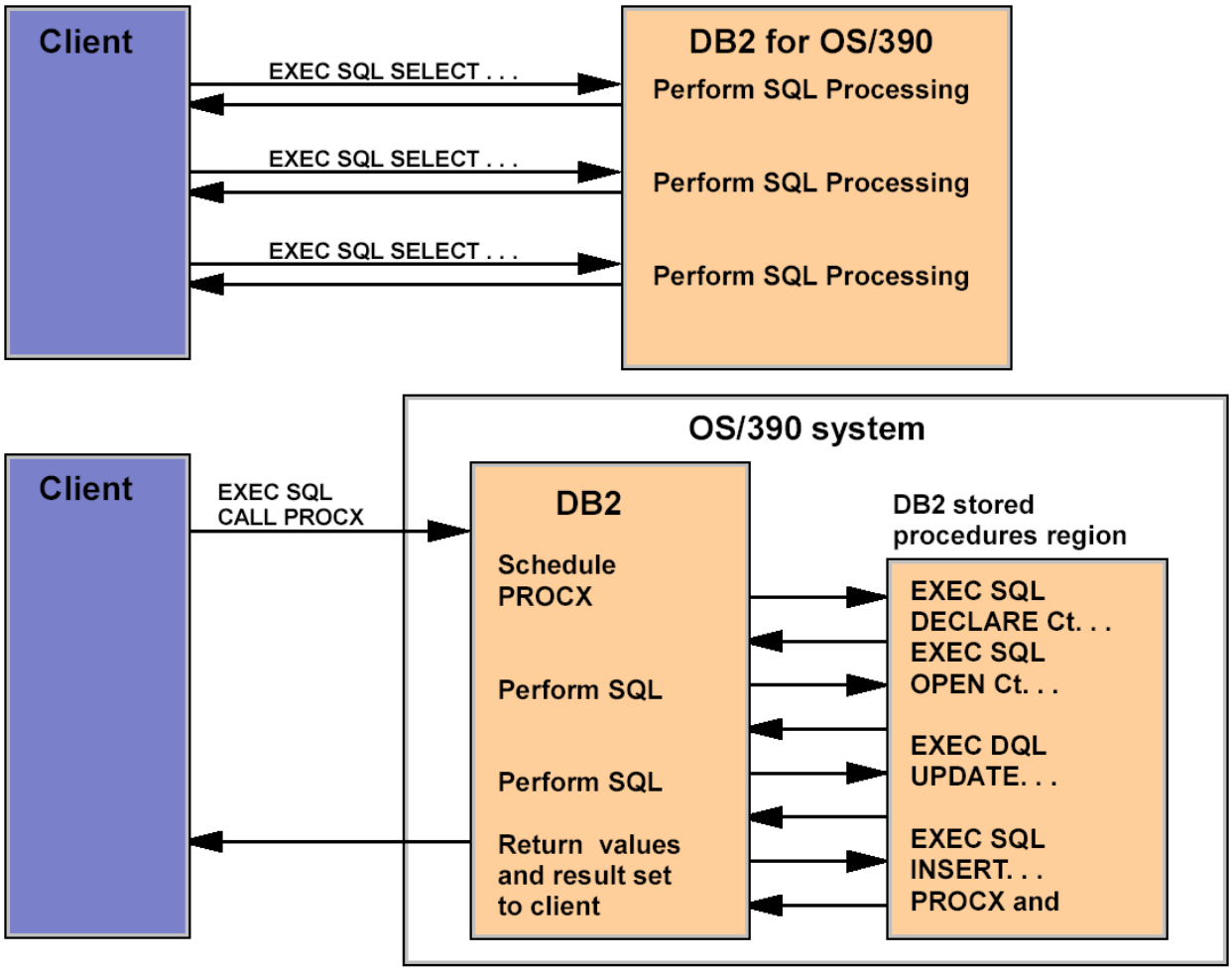


Abbildung 3.2.2 Geringer Netzwerkverkehr beim Einsatz von Stored Procedures

Einer der Nutzen der Verwendung von Stored Procedures ist in der obigen Abbildung dargestellt.

An Stelle vieler einzelner Anfragen und Antworten wird eine einzelne Anfrage ausgeführt, die die auf dem Server liegende Stored Procedure aufruft.

Bearbeitet wird die Anfrage bis zur Generierung eines Ergebnisses auf Serverseite, die Mitteilung dieses Ergebnisses an den Client geschieht wiederum in einer einzigen Nachricht. Für weiterführende Informationen bezüglich z/OS Stored Procedures sei [3] empfohlen.

3.2.5 EGL COBOL Generation

Die *Enterprise Generation Language* (EGL) ist eine von IBM entwickelte Programmiersprache der vierten Generation.

Verglichen mit Programmiersprachen der dritten Generation, zu denen unter Anderem *Java* und *C++* gehören, treten die softwaretechnischen Aspekte in den Hintergrund.

Der Entwickler konzentriert sich ausschließlich auf die Geschäftslogik und programmiert diese, EGL generiert daraus die benötigten *Java*- und/oder *COBOL*-Dateien.

Entsprechend der Plattform, auf der das Programm ausgeführt werden soll, werden die erforderlichen Laufzeitartefakte automatisch angepasst

4. Tutorials

In diesem Kapitel werden die im Rahmen der Diplomarbeit erstellten Tutorials vorgestellt. Sie geben eine Einführung in die Benutzung des Rational Developer for System z (RDz) und bieten Informationen zu den jeweils behandelten Themengebieten.

Da eine ausführliche Darstellung der Tutorials zu umfangreich wäre, werden die jeweiligen Themen kurz mit einigen Bildern aus den Übungen erläutert. Die vollständigen Tutorials befinden sich im Anhang auf DVD.

4.1 RDz Tutorial 01 – Inbetriebnahme von RDz Version 7.5

In diesem Tutorial wird die Installation von RDz 7.5 beschrieben und eine erste Verbindung zum zSeries Mainframe in Leipzig aufgebaut.

Die vorinstallierte Version ist auf den 2 DvDs verfügbar.

Eine Reihe von Screenshots soll die einzelnen Arbeitsschritte des Tutorials veranschaulichen und die Nachvollziehbarkeit der Arbeit gewährleisten.

Rational Developer for System z ist in eine virtuelle Maschine eingebettet, welche auf einer Windows XP basierte Workstation läuft. Dies kann zum Beispiel ein PC sein.

Als Virtualisierungssoftware dient *VMware* [14].

Durch diese Software wird eine einheitliche Arbeitsumgebung mit einem vorinstallierten Windows XP als Betriebssystem geschaffen.

Weiterhin ist Eclipse mit dem RDz-Plugin auf der virtuellen Maschine installiert. Dieses Paket wird den Teilnehmern des Client/Server Praktikums zur Verfügung gestellt.

Es wird die Software *Vmware Player* benötigt, welche als kostenloser Download für verschiedene Plattformen verfügbar ist.

Durch dieses Verfahren wird erreicht, dass das Gastsystem vollkommen vom Host isoliert ist und damit jederzeit eine Neuinstallation der virtuellen Maschine erfolgen kann, ohne dabei Eingriffe in das Betriebssystem des Hosts machen zu müssen.

Die Kommunikation zwischen Gastsystem und Host läuft über eine konfigurierte Netzwerkverbindung.

Das Tutorial ist in sieben Kapitel unterteilt.

Es werden die Installation des Vmware Player, die Anpassung des Gastsystems, die Konfiguration des internen Netzwerks und die Inbetriebnahme von RDz beschrieben.

Danach wird eine Verbindung zum zSeries Rechner Binks der Universität Leipzig hergestellt.

Das Tutorial wird durch einige vorbereitende Maßnahmen für die folgenden Übungen abgerundet.

Nach dem Start von RDz wird ein Workspace für die Speicherung der Projektdaten vorgeschlagen.

An dieser Stelle ist es dem Anwender überlassen, ob er den vordefinierten Pfad benutzt oder ein eigenes Verzeichnis erstellt. Änderungen an dieser Stelle haben keine Auswirkung auf den restlichen Verlauf der Übungen, der Workspace definiert nur den Ort in der virtuellen Maschine, an dem RDz die erstellten Arbeiten abgelegt.

Nachdem RDz vollständig geladen ist, bedarf es keiner weiteren Konfiguration. Das Programm ist sofort einsatzbereit.

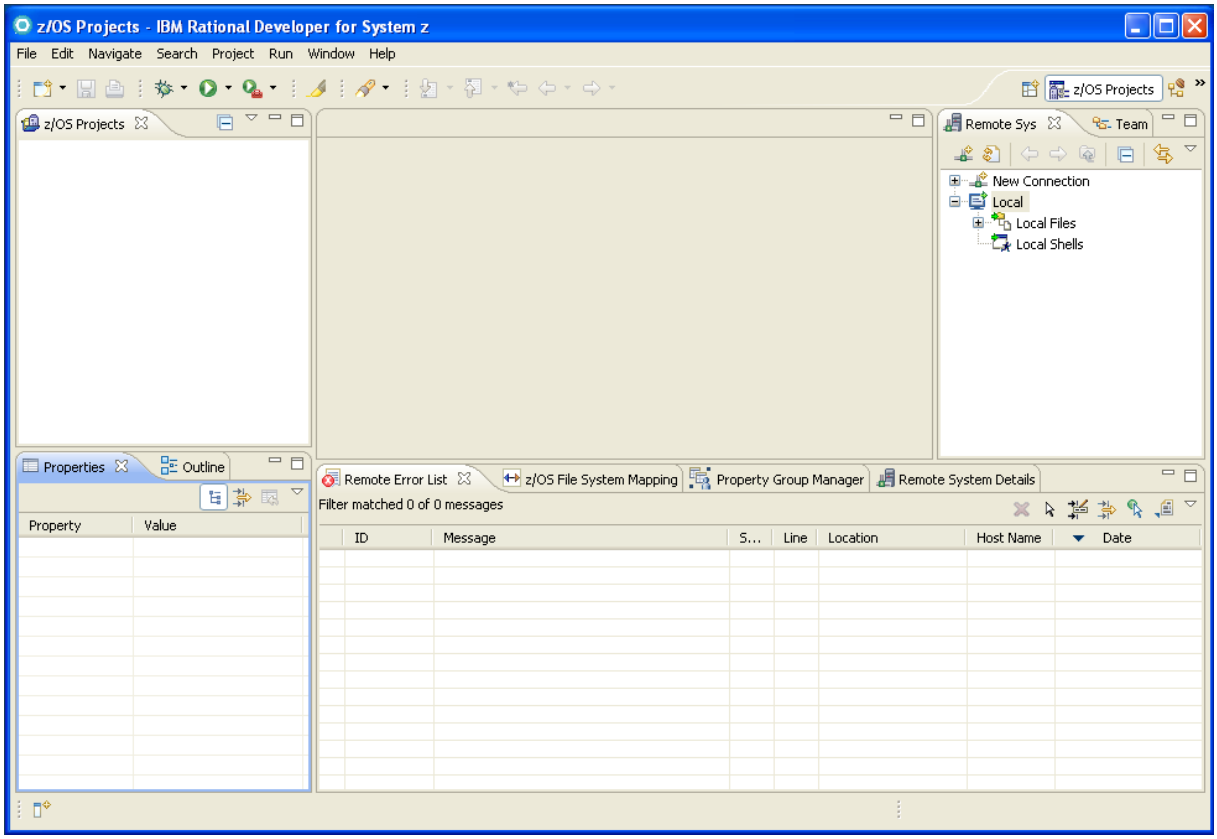


Abbildung 4.1.1 Programmstart von RDz

Nun wird mit Hilfe des New Connection Assistenten von RDz eine Verbindung zum z/OS-Rechner *Binks* in Leipzig mit der IP-Adresse 139.18.4.34 hergestellt. Die Standardangaben können übernommen werden.

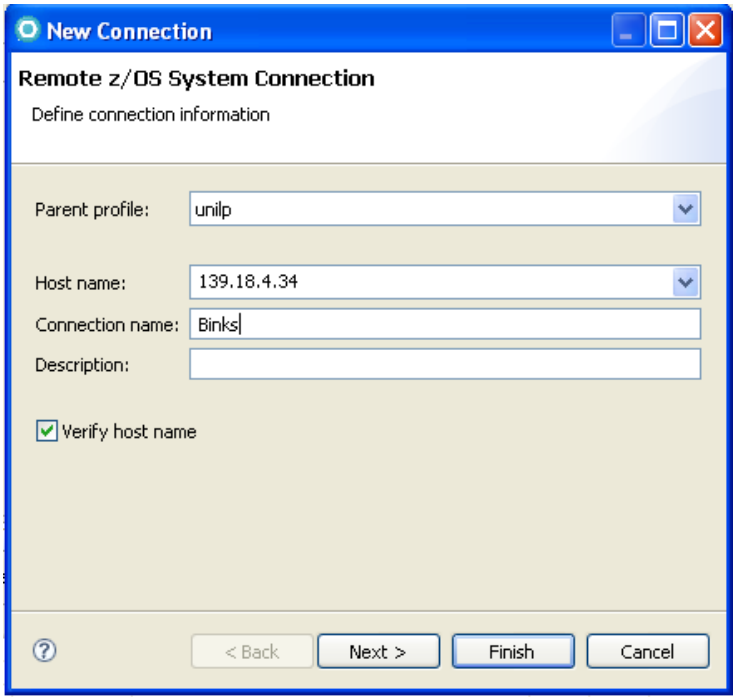


Abbildung 4.1.2 Erstellen einer neuen Verbindung

Es ist eine Benutzerkennung und ein Passwort erforderlich, um die Verbindung aufzubauen. Diese Daten erhalten die Studenten im Vorfeld vom betreuenden Praktikumsleiter.

Abbildung 4.1.3 Verbindungsaufbau

Nachdem die Verbindung hergestellt ist, erscheinen zwei Meldungen.

Die erste sagt dem Benutzer, dass keine gesicherte Verbindung (SSL) verwendet wird.

Mit der zweiten Meldung wird darauf hingewiesen, dass auf dem Host Server Binks eine ältere Version (WDz 7.1) läuft, als auf dem Client (RDz 7.5).

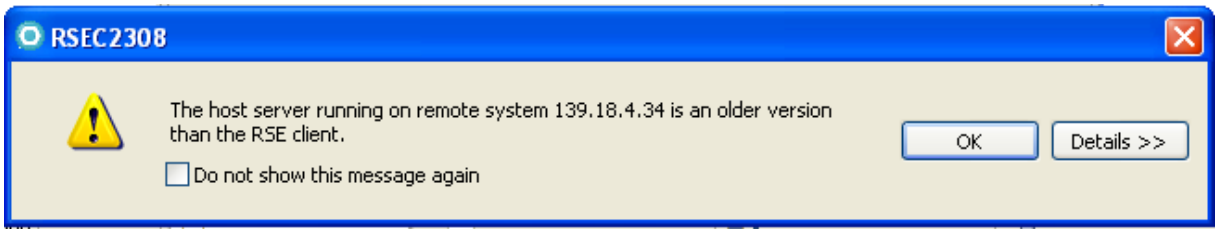


Abbildung 4.1.4 Abwärtskompatibilität bei RDz

Da RDz abwärtskompatibel ist, stellt dieser Versionsunterschied kein Problem dar und die Meldung kann ohne Probleme mit OK bestätigt werden.

Im Anschluss wird das ordnungsgemäße Trennen der Verbindung gezeigt.

Dies ist wichtig, da man sonst weiter als Benutzer auf dem zSeries Rechner eingeloggt ist und eine automatische Trennung erfolgen muss, was einige Zeit in Anspruch nimmt und währenddessen kein erneutes Einloggen mit derselben Benutzerkennung möglich ist.

Im vorletzten Schritt werden die für die weiteren Tutorials benötigten Daten auf das virtuelle Gastsystem kopiert.

4.2 RDz Tutorial 02 – Local PL/1

Dieses Tutorial ist in fünf Kapitel unterteilt.

Es wird gezeigt, wie man ein lokales PL/1 Programm schreibt und lauffähig macht.

Um das gelernte Wissen aus den ersten drei Kapiteln zu vertiefen, wird in Abschnitt 4 ein zweites PL/1 Programm erstellt, wobei hier der Schwerpunkt auf dem Verstehen des Quellcodes gerichtet ist.

Zum Schluss wird gezeigt, wie man mit dem Debugger unter RDz 7.5 arbeitet.

Zuerst wird ein neues Projekt erstellt, welches als lokales Projekt auf dem Gastsystem läuft.

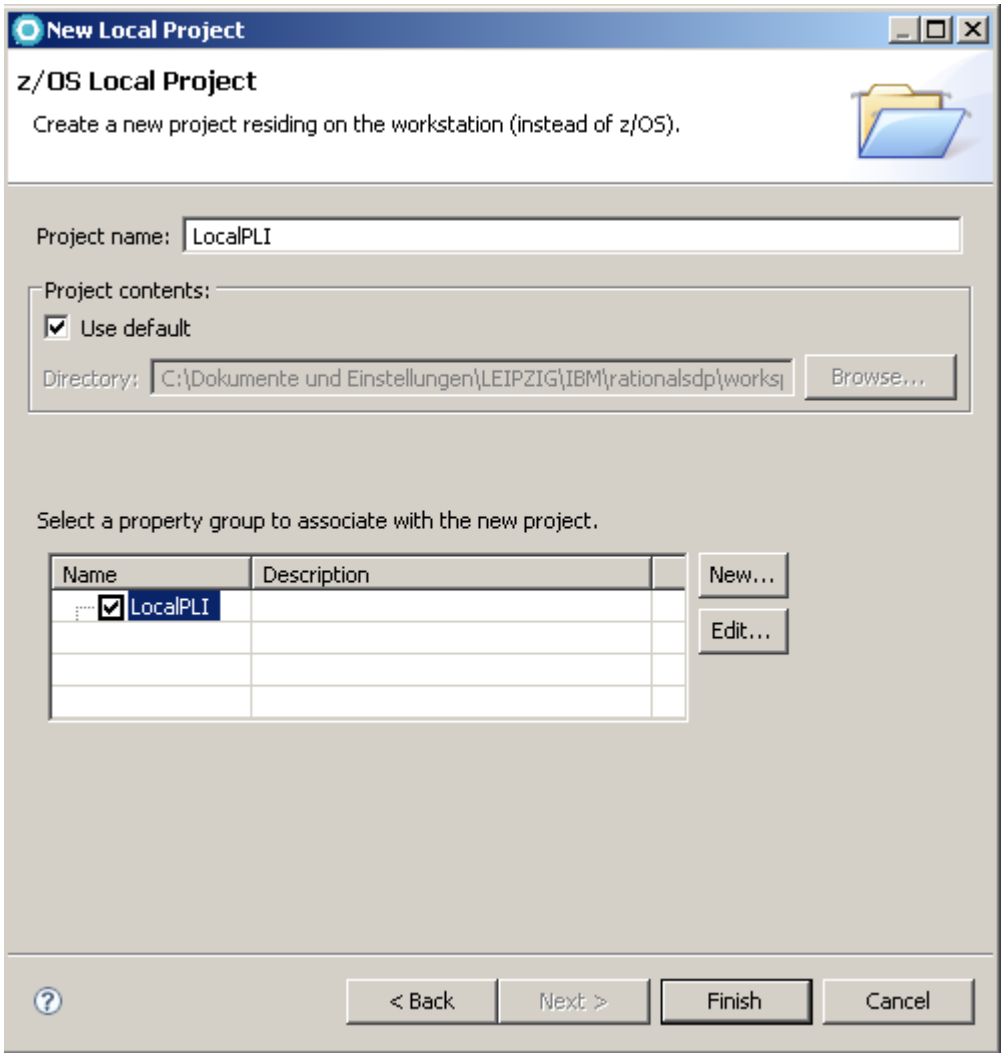


Abbildung 4.2.1 Neues Projekt erstellen

Während des Dialogs wird außerdem eine neue „Property Group“ erstellt.

Diese speichert alle Einstellungen wie zum Beispiel „Compiler Options“ für das Projekt und kann später für andere lokale Projekte wieder verwendet werden.

Diese Art der Definition und Speicherung der Einstellungen ist neu in RdZ 7.5.

Im nächsten Schritt wird ein „Hello World“ Programm erzeugt. Da es in den ersten drei Kapiteln um das Kennenlernen von RDz geht, wurde extra ein einfaches Codebeispiel gewählt.

Ein komplexeres Programm folgt in Kapitel 4.

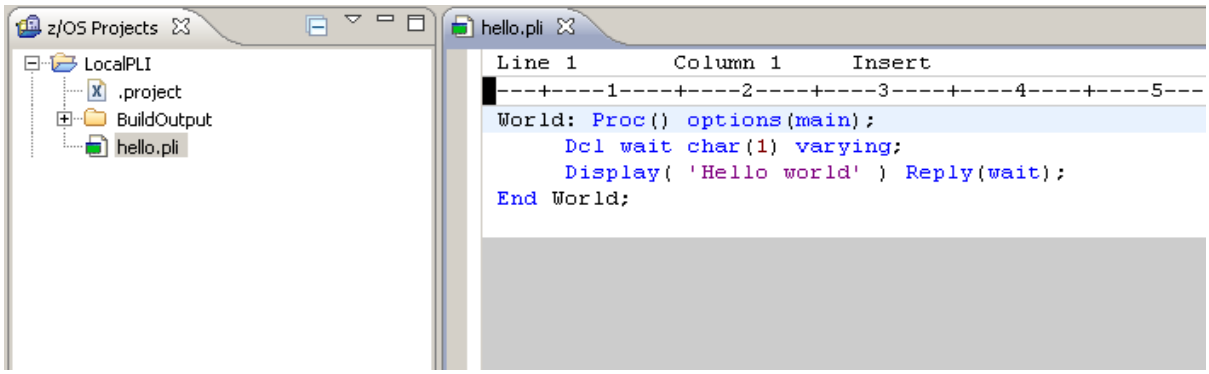


Abbildung 4.2.2 Lokales Programm unter RDz

Nun folgt ein lokaler Syntaxcheck und das kompilieren des Programms.

Danach befindet sich die ausführbare Datei „hello.exe“ im Ordner „Buildout“.

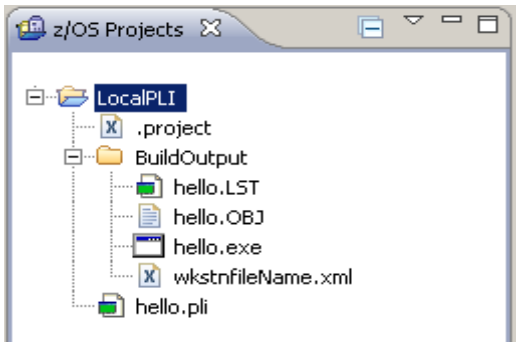


Abbildung 4.2.3 Projektübersicht bei RDz

Nachdem im letzten Schritt die „Run Configuration“ erstellt wurde, bekommen wir unsere Ausgabe in einer Dos-Kommandobox angezeigt.

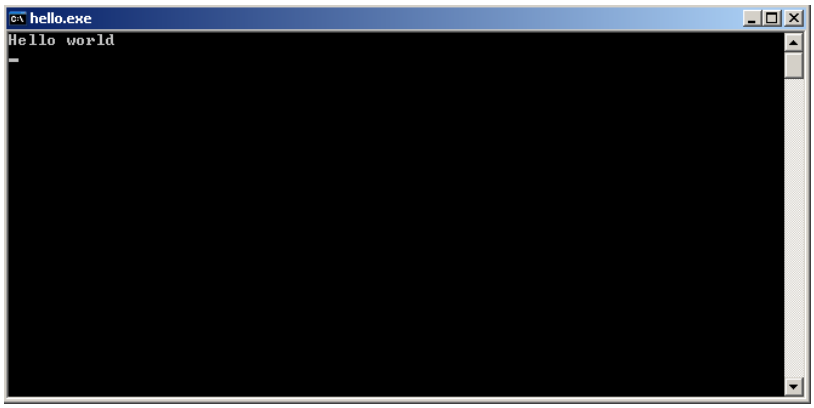


Abbildung 4.2.4 Ausgabe des lokalen Programms

Im 4. Kapitel soll nun der, in den vorherigen Kapiteln gelernte, Umgang mit RDz wiederholt und geübt werden.

Hierzu soll ein komplexeres Programm zur Berechnung des Flächeninhalts eines Dreiecks geschrieben werden.

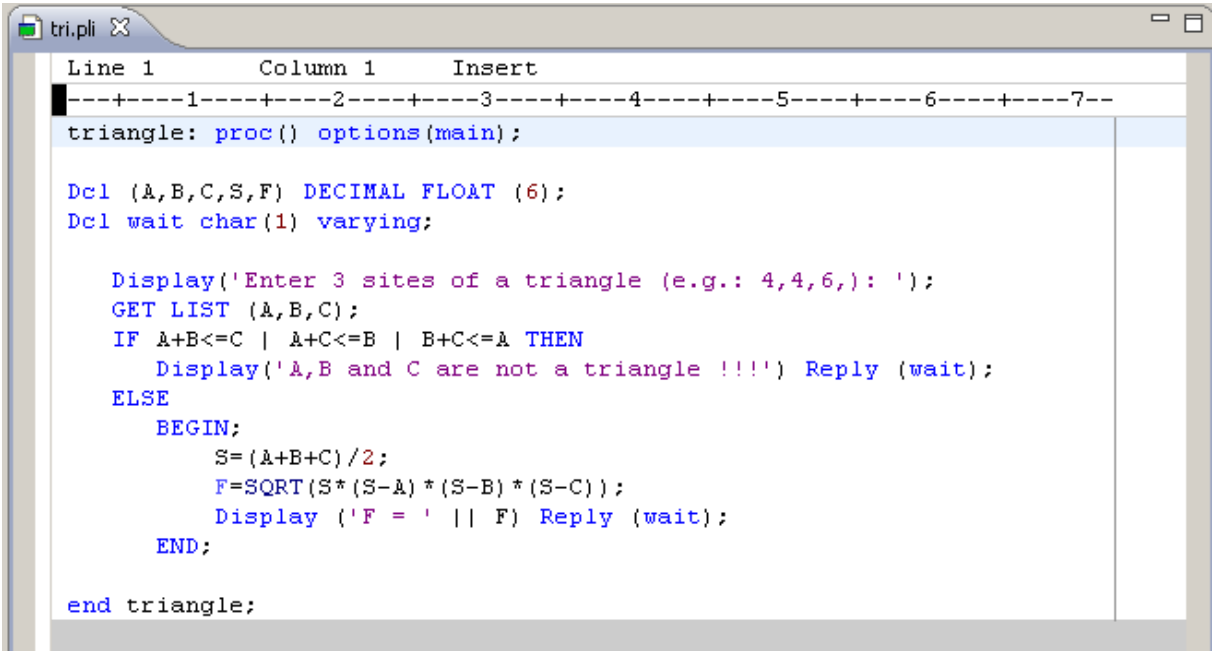


Abbildung 4.2.5 Komplexeres Programm unter RDz

Der Quellcode wird im Folgenden näher betrachtet, um dem Übungsteilnehmer komplexeres PL/1 näher zubringen.

Anschließend kann das Programm nach Belieben getestet werden.

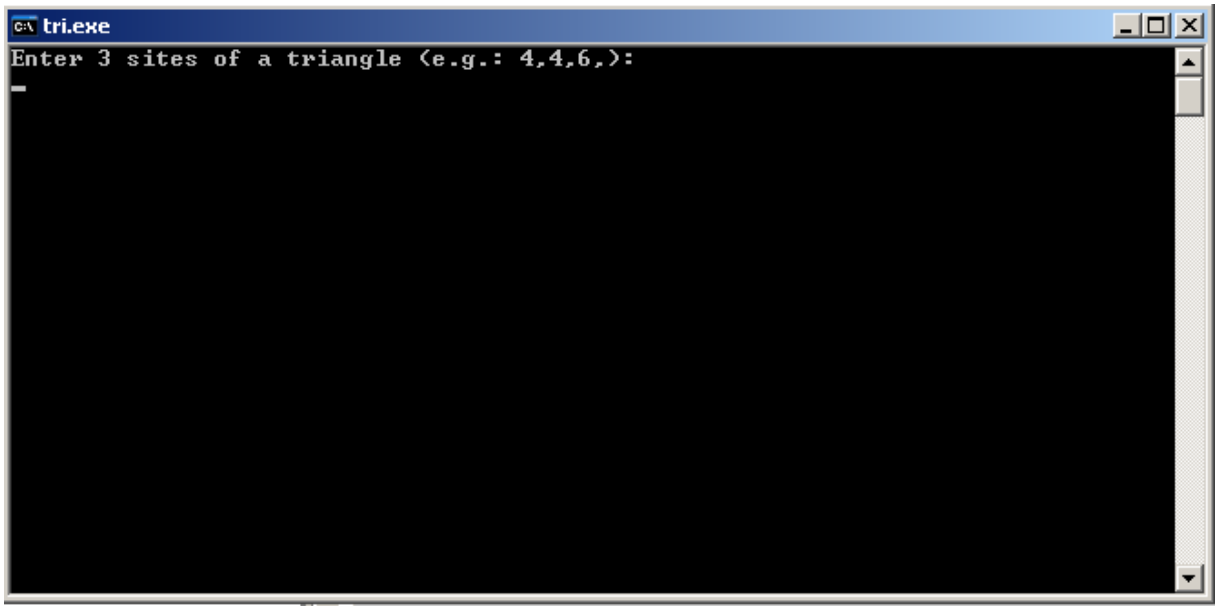


Abbildung 4.2.6 Ausgabe des komplexeren Programms

Im letzten Kapitel dieses Tutorials wird das PL/1 Programm zur Berechnung des Flächeninhalts eines Dreiecks debuggt.

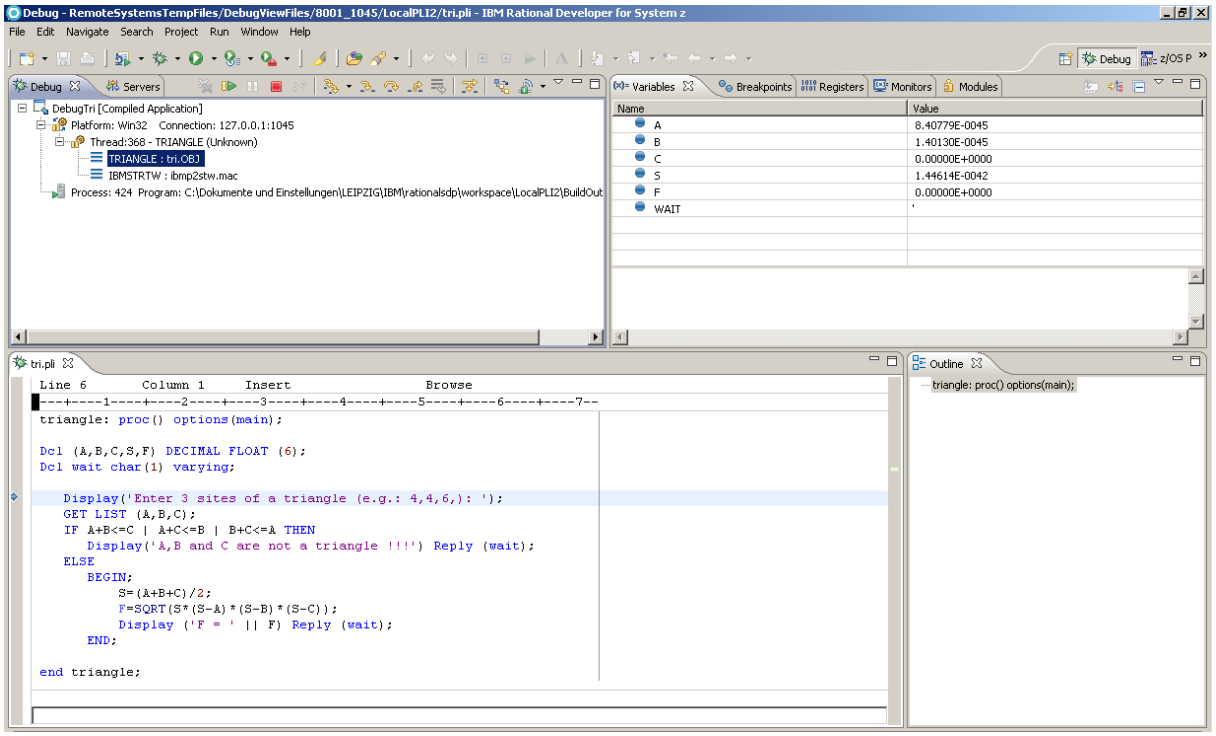


Abbildung 4.2.7 Debugumgebung unter RDz

Es wird der Umgang mit Breakpoints, sowie die Wertzuweisung einer Variablen gezeigt.

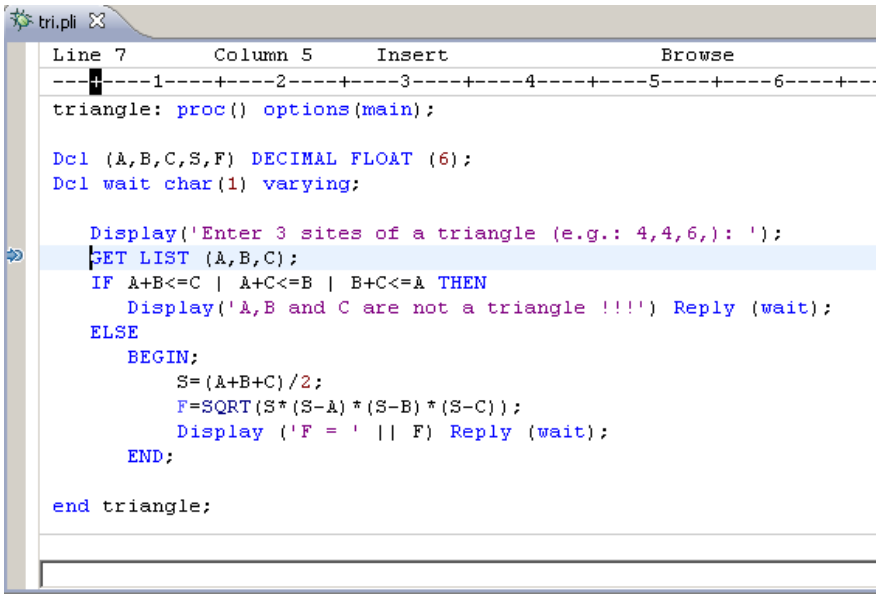


Abbildung 4.2.8 Breakpoints unter RDZ

4.3 RDz Tutorial 03 – Remote PL/1

In diesem Tutorial wird gezeigt, wie man das schon bekannte „Hello World“ Programm auf dem zSeries Großrechner in Leipzig (Binks) kompiliert und ausführt.

Der Schwerpunkt der Übung liegt auf dem Umgang mit Datasets und der Erstellung, Kompilierung und Ausführung von PL/1 Programmen auf dem Mainframe.

Nachdem die Verbindung zu Binks hergestellt ist, müssen verschiedene Datasets angelegt werden.

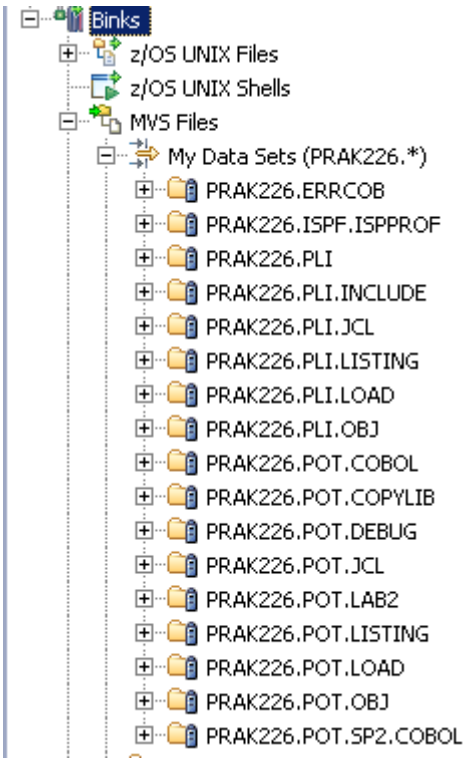


Abbildung 4.3.1 Arbeiten mit MVS Files

Um Fehlerquellen zu minimieren, werden eventuell bestehende Datasets zuerst gelöscht und danach mit vorgegebenen Parametern neu erzeugt.

Nun wird eine neue Property Group hinzugefügt. Hierfür muss die Benutzerkennung in der Datei „remotePLI.xml“, welche sich im Verzeichnis des Tutorials befindet, angepasst werden.

In dieser Datei befinden sich alle Informationen bezüglich der Eigenschaften des folgenden Projekts.

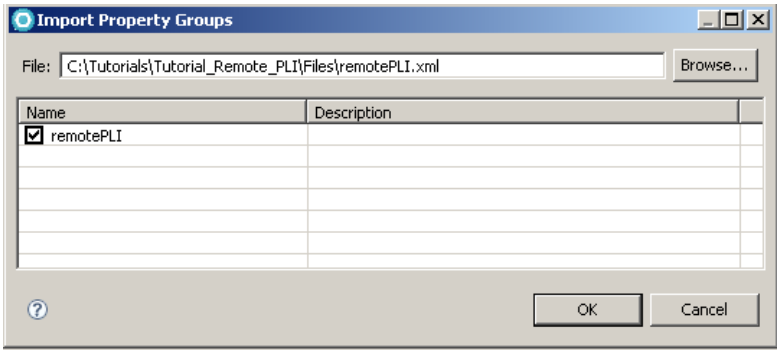


Abbildung 4.3.2 Importieren einer Property Group

Nach dem Einbinden der xml-Datei wird ein neues Projekt und ein neues Subprojekt erstellt, welche mit der importierten Property Group verknüpft werden.

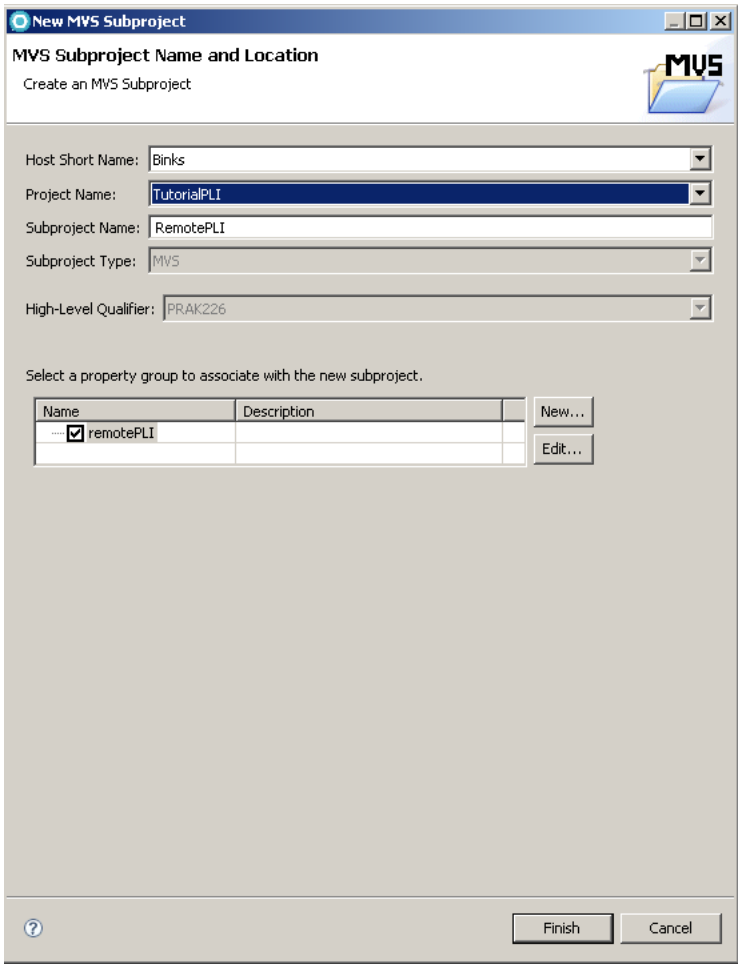


Abbildung 4.3.3 Erstellung eines Remote Projekts

Im Folgenden wird das Member „hello.pli“ erstellt, welches den PL/1 Code enthält. Anschließend führt der Nutzer einen Syntax Check durch und erzeugt automatisch eine „JCL – Datei“.

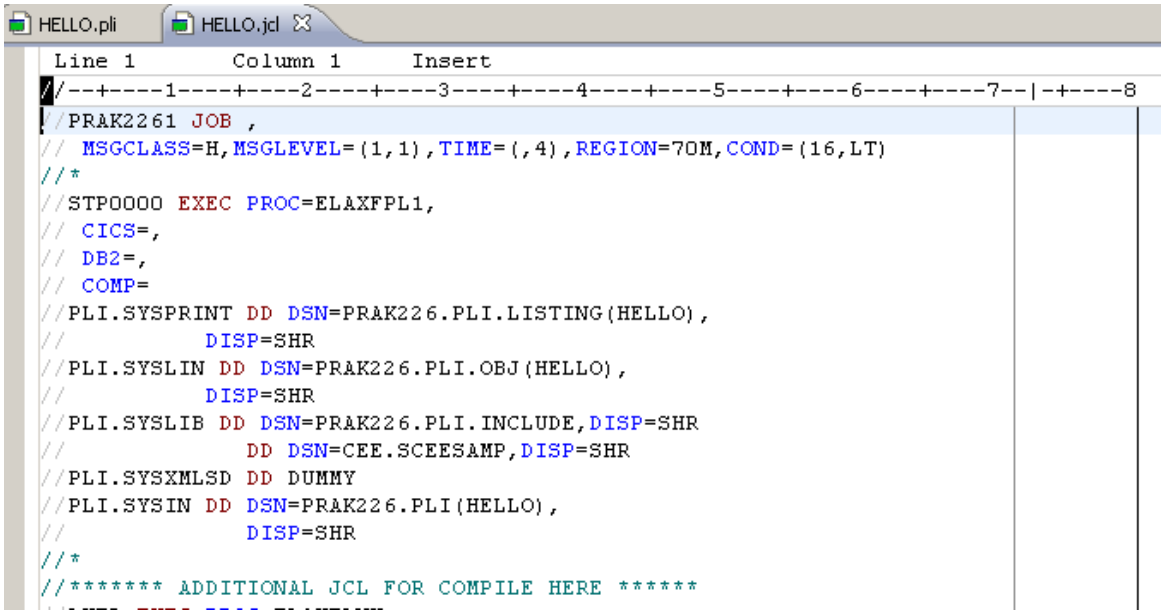


Abbildung 4.3.4 JCL File unter RDz

JCL steht für „Job Control Language“ und ist die Steuersprache für Stapelverarbeitungen in einem Großrechnerumfeld.

Aufgabe der JCL ist es, die auszuführenden Programme, deren Reihenfolge, sowie eine Laufzeitumgebung (Verbindung zu physischer Hardware, E/A und Dateien) vorzugeben.

Die heute auf Systemen unter z/OS eingesetzte JCL wurde 1964 für OS/360 IBM entwickelt. Bei der Weiterentwicklung wurde Abwärtskompatibilität gewährleistet.

Ursprünglich wurde JCL auf Lochkarten gespeichert. Jobs wurden dann per Kartenleser ins System eingespielt. Heute sind JCL-Bibliotheken mit Recordformat FB (Fixed Blocked) und Recordlänge 80 üblich.

JCL wird vom Job Entry Subsystem (JES2 oder JES3) eingelesen und interpretiert. Auch die Subsysteme, Systemfunktionen (Started Tasks) und die Anmeldungen eines Benutzers, am TSO, verwenden JCL Prozeduren zur Initialisierung [16].

Die Datei „hello.jcl“ wird dann an den Mainframe übermittelt, wo ein Job erstellt und ausgeführt wird.

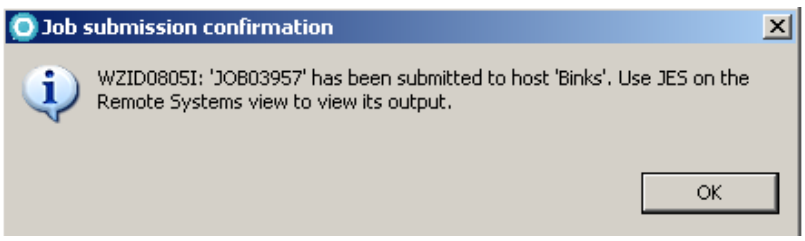


Abbildung 4.3.5 Bestätigung der Übertragung eines Jobs

Danach kann der Benutzer die Ausgabe des Programms überprüfen.

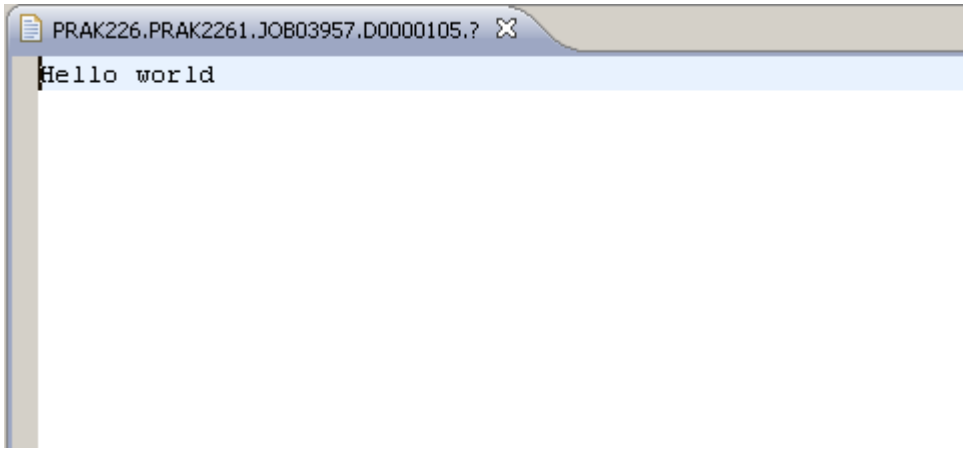


Abbildung 4.3.6 Ausgabe des Remote Programms

5. Zusammenfassung und Ausblick

In den ersten drei Kapiteln wurden die Grundlagen in Hard- und Software gegeben. Der Schwerpunkt der Arbeit liegt in Kapitel 4, welches die erstellten Tutorials erläutert, die in den folgenden Semestern Bestandteil des Client-Server Praktikums der Universität Leipzig sind.

Die Erstellung und Bearbeitung von PL/1 Programmen mit dem Rational Developer for System z wird auch in den kommenden Jahren eine große Rolle spielen.

Es sind heute noch PL/1 Programme im Einsatz, die mehr als 30 Jahre alt sind.

RDz ist, gerade für Neueinsteiger, ein wichtiges Hilfsmittel, um einen schnellen und problemlosen Einstieg in PL/1 zu bekommen.

Da Eclipse als weitverbreitete Entwicklungsumgebung schon von anderen bekannten Programmiersprachen, wie zum Beispiel Java, genutzt wird, erleichtert dies Anfängern den Einstieg.

Für die Zukunft sind, aufbauend auf dieser Arbeit, weitere Tutorials für das Client/Server Praktikum denkbar. Ein Beispiel wäre ein Tutorial, welches sich mit dem Debuggen eines Remote PL/1 Programm unter RDz beschäftigt.

6. Literaturverzeichnis

[1] F. Cassia: *Eclipse.org eclipsing Borland's Jbuilder*. VNU Business Publications, 2004. http://www.theinquirer.net/default.aspx?article=15862, Stand 12.05.2004

[2] *IBM Weighs in With Uptime Guarantee*s. Computergram International, 1999. http://findarticles.com/p/articles/mi_m0CGN/is_1999_March_25/ai_54207476/pg, Stand 18.08.2007

[3] P. Bruni u.a.: *DB2 for z/OS Stored Procedures: Through the CALL and Beyond*. IBM Form No. SG24-7083-00, 2006. http://www.redbooks.ibm.com/abstracts/sg247083.html, Stand 22.02.2006

[4] *Eclipse – an open development platfo*rm. http://www.eclipse.org, Stand 13.07.2009

[5] *Eclipse Newcomers FAQ*. http://www.eclipse.org/home/newcomers.php, Stand 13.07.2009

[6] P. Herrmann, U. Kebschull, W. G. Spruth: *Vorlesungsskript Einführung in z/OS und OS/390 – Teil 4.* Universität Leipzig, 2008.

[7] P. Herrmann, U. Kebschull, W. G. Spruth: *Einführung in z/OS und OS/390*. 2. Auflage, Oldenburg, 2004

[8] P. Herrmann, W. G. Spruth: *Vorlesungsskript Einführung in z/OS und OS/390 – Teil 7.* Universität Leipzig, 2008.

[9] *Rational Developer for System z*. IBM. http://www-142.ibm.com/software/dre/ecatalog/Detail.wss?locale=de_DE&synkey=H564288F91212R05, Stand: 15.04.2009

[10] *IMS SOAP Gateway*. IBM. http://www-306.ibm.com/software/data/ims/soap/, Stand 01.02.2009

[11] T. Ross, N. Tindall, S. Tampa: *XML, COBOL and Application Modernization*. IBM Reference No. 7004198, 2007. http://www-01.ibm.com/support/docview.wss?rs=492&context=SS6SG3&q=&uid=swg27004198&loc=en_US&cs=utf-8?=en+en, Stand: 13.08.2007

[12] Topics on Version 7 of Rational Developer for System z and IBM WebSphere Developer for System z http://www.redbooks.ibm.com/abstracts/sg247482.html?Open, Stand 02.02.2009

[13] *IBM SOAP for CICS feature delivers fully supported SOAP access to CICS*. IBM, Software Announcement 203-199, 2003. http://www-01.ibm.com/cgi-bin/common/ssi/ssialias?infotype=an&subtype=ca&htmlfid=897/ENUS203-199&appname=usn, Stand 13.07.2009

[14] *VMware*. http://www.vmware.com/de/, Stand 20.08.2007

[15] *Eclipse (IDE)*. Wikipedia. http://de.wikipedia.org/wiki/Eclipse_%28IDE%29, Stand 27.07.2007

[16] JCL. Wikipedia. http://de.wikipedia.org/wiki/Job_Control_Language, Stand 13.05.2009

7. Abkürzungsverzeichnis

ALM	Application Lifecycle Management
BMS	Basic Mapping Support
CF	Coupling Facility
CICS	Customer Information Control System
CICS TS V31	CICS Transaction Servers v3.1
EGL	Enterprise Generation Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines Corporation
IDE	integrated developement enviroment
IMS	Information Management System
ISPF	Interactive System Productivity Facility
JCL	Job Control Language
JES	Job Entry System
LIC	Licenced Internal Code
LPAR	Logical Partitions
RDz	Rational Developer for System z
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
WDz	WebSphere Developer for zSeries
WLM	Workloadmanager
WSDL	Web Service Description Language
XML	Extensible Markup Language

8. Verzeichnis der Anlagen

Anlage 01	RDz Tutorial 01 Inbetriebnahme von RDZ Version 7.5
Anlage 02	RDz Tutorial 02 Local PL/1
Anlage 03	RDz Tutorial 03 Remote PL/1
Anlage 04	DVD Nr. 1 - Imagedateien des vorinstallierten WindowsXP-Gastsystems - Diplomarbeit als PDF - Literaturreferenzen in elektronischer Form einschließlich Internetseiten - VMware, Installationsfile des Vmware-Players - RDz_Tutorials
Anlage 05	DVD Nr. 2 - Imagedateien des vorinstallierten WindowsXP-Gastsystems

Anlage 01

RDz Tutorial 01

Inbetriebnahme von RDZ Version 7.5

RDz Tutorial 01
Inbetriebnahme von RDZ Version 7.5

Inhalt

RDz ist ein *Integrated Development Environment* (IDE) für die Entwicklung von z/OS Anwendungen in Sprachen wie COBOL, PL/1, Java, Assembler, sowie einer 4th Generation Language *Enterprise Generation Language* (EGL). Die IDE besteht aus 2 Komponenten. Der Benutzer verfügt über eine umfangreiche Laufzeitumgebung auf seiner Windows XP basierten Workstation. Diese ist über das Netz mit einem z/OS Rechner verbunden, auf dem ein passendes Gegenstück zu der IDE Komponente auf der Workstation läuft.

Wir verwenden die RDz Version 7.5. RDz bedeutet **Rational Developer for System z**. Die Versionen bedingen unterschiedliche Installationen auf der Workstation und der z/OS Host Seite.

RDz Tutorial 01 Inbetriebnahme von RDZ Version 7.5

RDz Tutorial 02 Local PL/1

RDz Tutorial 03 Remote PL/1

Dieses und die folgenden Tutorials laufen auf den z/OS 1.8 System binks.informatik.uni-leipzig.de bzw. 139.18.4.34 der Abteilung Computersysteme, Institut für Informatik der Universität Leipzig. Die RDz Installation auf diesem System wurde von Uwe Denneler, Elisabeth Puritscher und Martin Benjamin Storz von der IBM vorgenommen, unterstützt von Martina Koederitz, Herb Kircher und Andrea Hermelink, ebenfalls Mitarbeiter der IBM. Lokaler Support wird von Herrn Andreas Nagel von der Universität Tübingen und von Herrn Dr. Paul Herrmann und Herrn Niels Michaelsen von der Universität Leipzig gegeben.

Der Text der meisten Tutorials basiert auf einer Reihe von Tutorials, die von Frau Isabel Arnold von der IBM im Rahmen eines z/OS Lehrgangs in Hamburg im August 2006 vorgestellt wurden. Die Übungen wurden von Karsten Kunze im Rahmen seiner Abschlussarbeit für PL/1 umgeschrieben und auf RDz 7.5 aktualisiert.

Übersicht

Das vorliegende RDz Tutorial 01 beschreibt die erstmalige Inbetriebnahme von RDz. Es besteht aus den folgenden Abschnitten:

Voraussetzung sind 2 DVDs , auf denen sich eine VMware virtuelle Maschine befindet. Diese VMware virtuelle Maschine enthält:

Windows XP Service Pack 2, mit einer Windows Lizenz der Universität Leipzig und
RDZ Version 7.5

Die beiden DVDs in ein geeignetes Verzeichnis, z.B. C:\RDZvm kopieren.

Zur Ausführung wird der VMWare-Player benötigt. Aus dem Internet die Datei **`VMware-player-2.0.2-59824.exe`** herunterladen und ausführen.

Der folgende Text verwendet diese Abkürzungen

1k	1 Klick auf die linke Maustaste
2k	2 Klick auf die linke Maustaste
1kr	1 Klick auf die rechte Maustaste

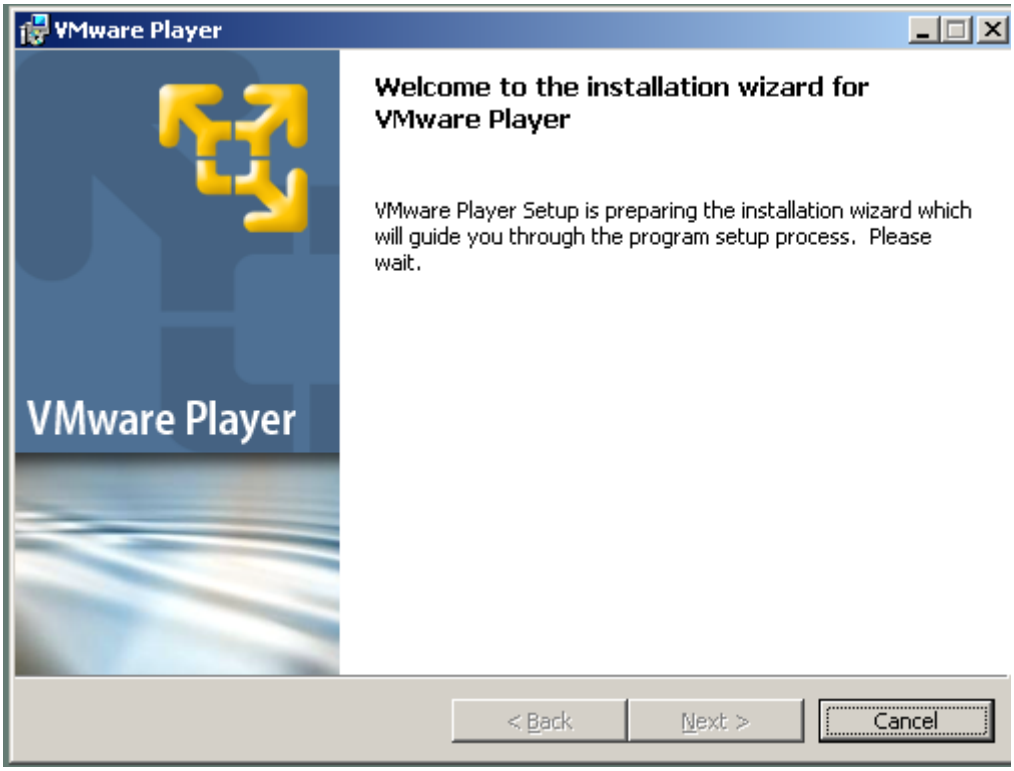
1 Vorbereitung: Kopieren der DVDs

Um in einem späteren Schritt mit Hilfe von **VMWare** ein Windows XP Betriebssystem als Gastsystem auf Ihrem Rechner ausführen zu können ist es nötig, den Inhalt der beiden zur Verfügung gestellten DVDs auf Ihren lokalen Rechner zu kopieren. Wählen Sie hierfür ein Verzeichnis mit ausreichend Speicherplatz, da die Daten knapp 8 GB belegen und bei intensiver Nutzung ansteigen können.

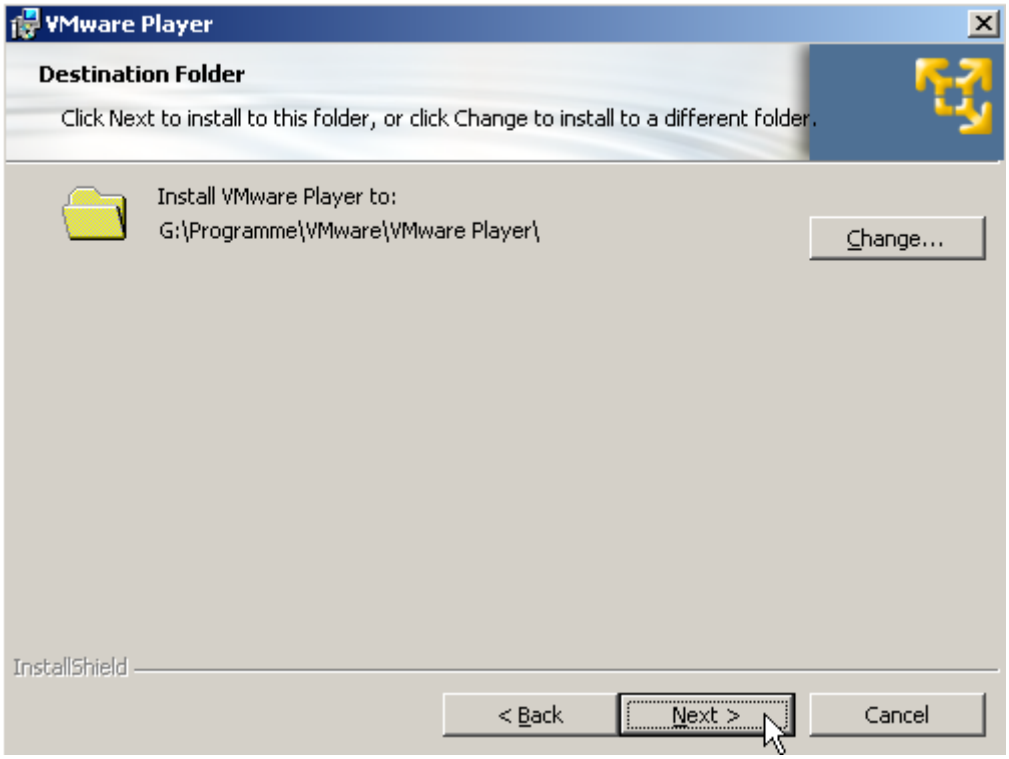
Achten Sie darauf, dass sie Schreibrechte in dem Verzeichnis haben, da die Systemdaten des Gastsystems bei individueller Anpassung überschrieben werden müssen.

2 Installation von VMWare Player

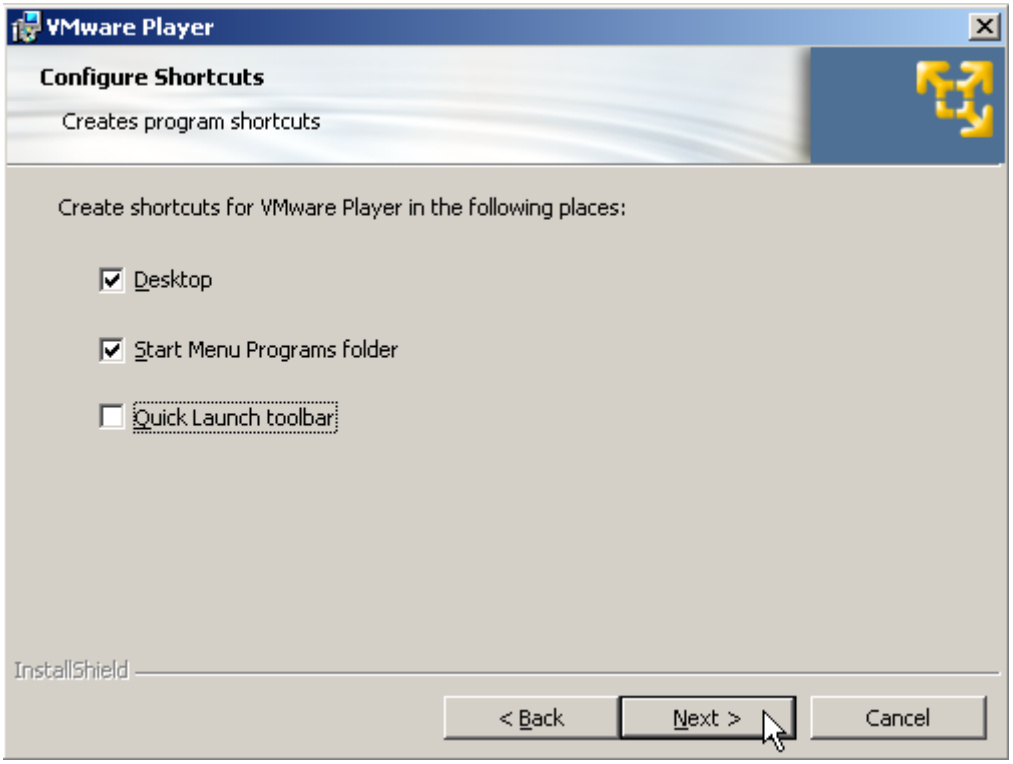
1. Die Datei VMWare-Player-.2.0.2-59824.exe ausführen. Next klicken.



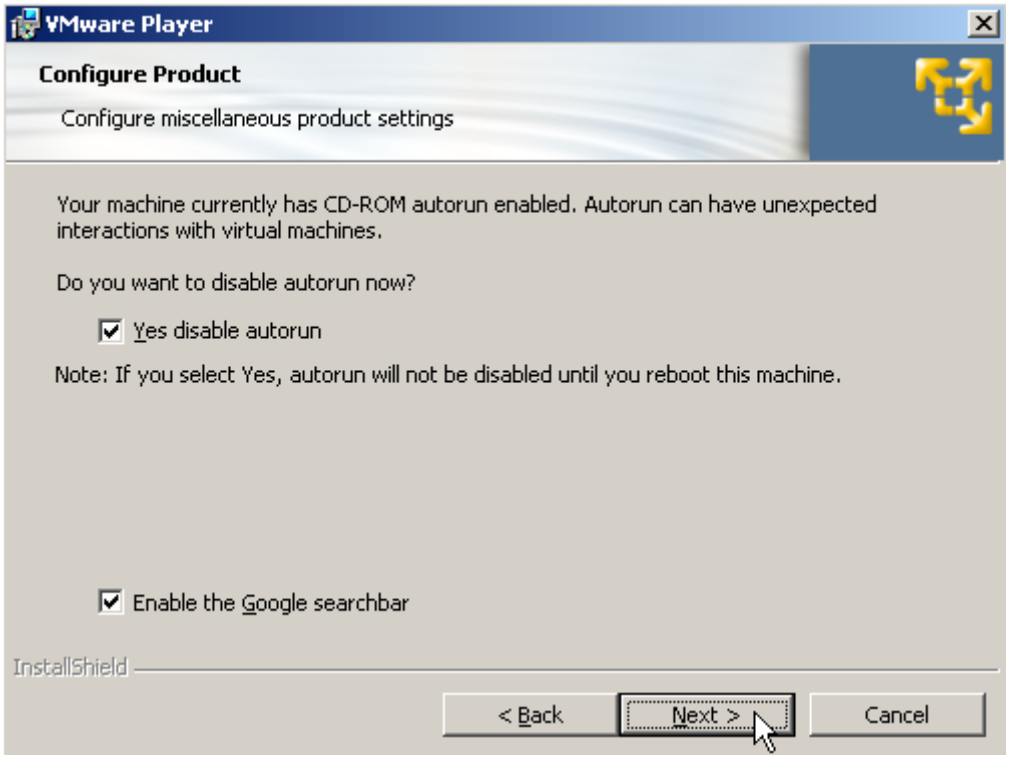
2. Next klicken.



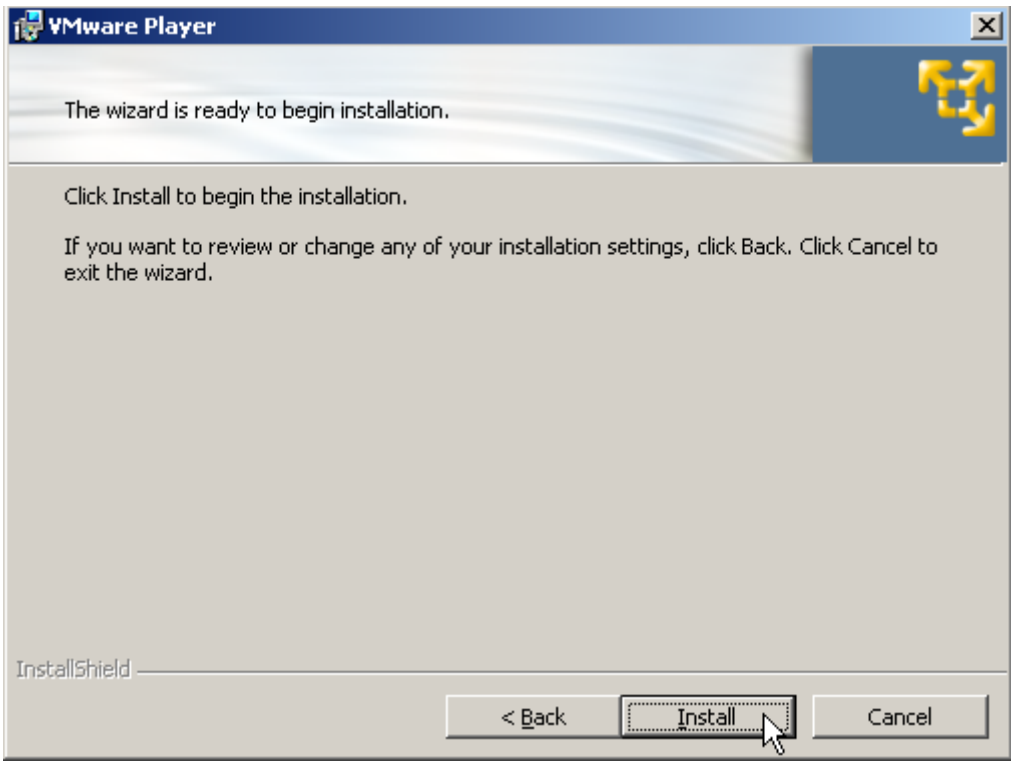
3. Next klicken.



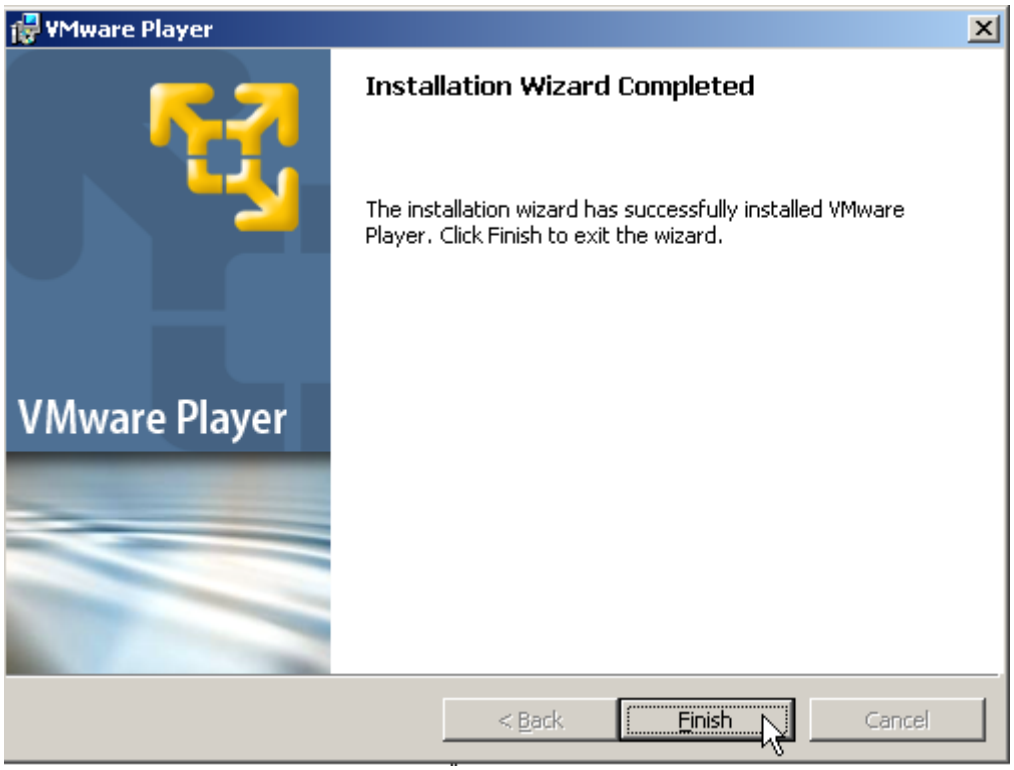
4. Option **Yes disable autorun** wählen und mit Next bestätigen.



5. Install klicken.



6. Installation abgeschlossen, click **Finish**.



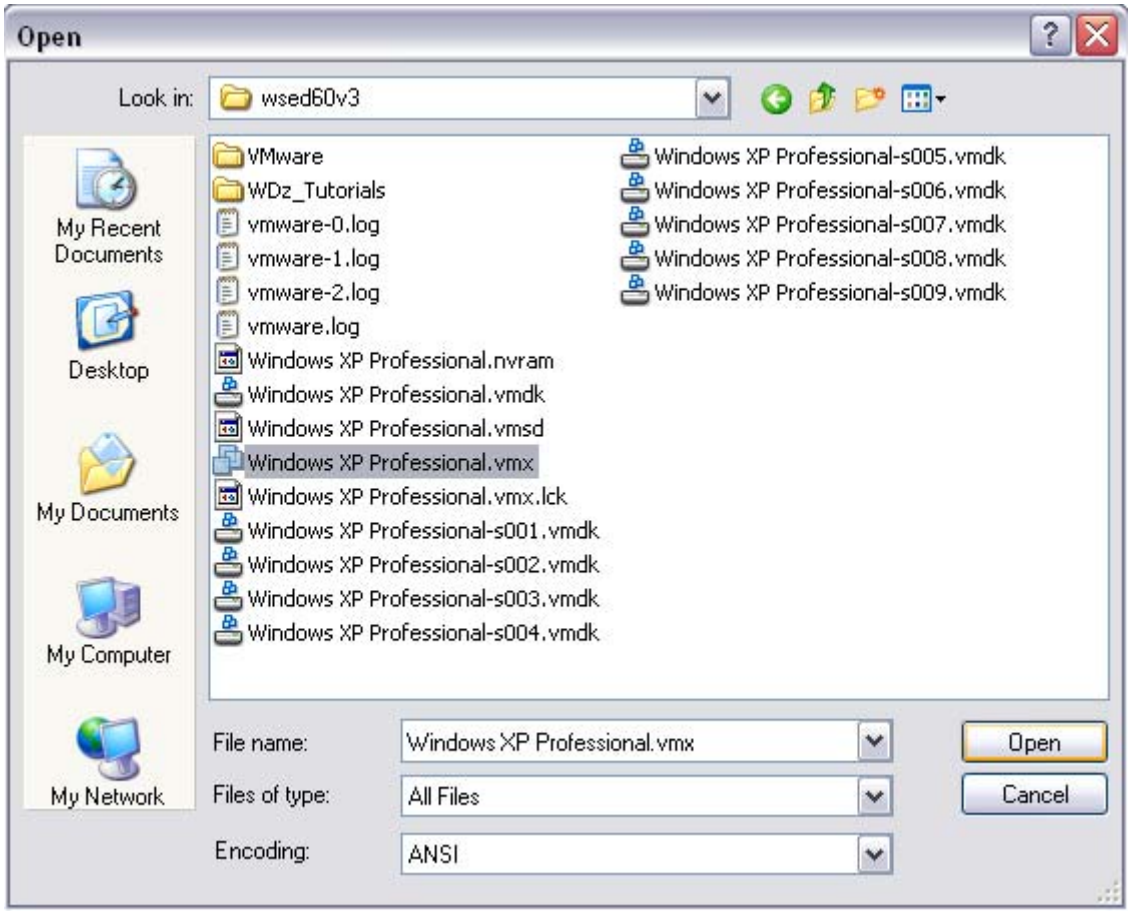
7. Der Rechner wird Sie auffordern, ein Restart durchzuführen. Starten Sie neu.

3 Windows XP Pro Professional der virtuellen Maschine (Gast Windows XP) anpassen

Auf Ihrem Rechner teilen sich der lokale Host und die virtuelle Maschine den Hauptspeicher. RDz verlangt möglichst viel Hauptspeicherplatz. Das Host Windows-Betriebssystem muss deshalb mit möglichst wenig Speicherplatz auskommen.

1. Den Zubehör Editor aufrufen: Start → Programme → Zubehör → Editor. In das Verzeichnis wechseln, in dem sie die RDz Dateien gespeichert haben. (Unter Umständen muss man unter Dateityp Alle Dateien auswählen).

   Die File **WindowsXP Professional.vmx** enthält das Konfigurationsfile. Hier findet sich der Eintrag **memsize = "700"**. Dies bedeutet, dass dem VM Gastsystem 700 MByte Hauptspeicher zugeordnet werden. Der Rest des realen Hauptspeichers steht dem Hostsystem zur Verfügung. Bei einem Rechner mit 1 GByte Haupspeicher bleiben somit für das Host-Betriebssystem 300 MByte übrig.



2. Diesen Parameter nach Ihren Gegebenheiten abändern. Bei 1 Gbyte Hauptspeicher kommen Sie evtl. mit dem Eintrag **memsize = "800"** klar, wenn unter dem Host Windows-Betriebssystem neben RDz möglichst wenig andere Anwendungen laufen.

   Wenn Ihr Rechner 2 Gbyte Hauptspeicher hat, ist das ideal. In diesem Fall wäre z.B **memsize = "1500"** eine gute Wahl.

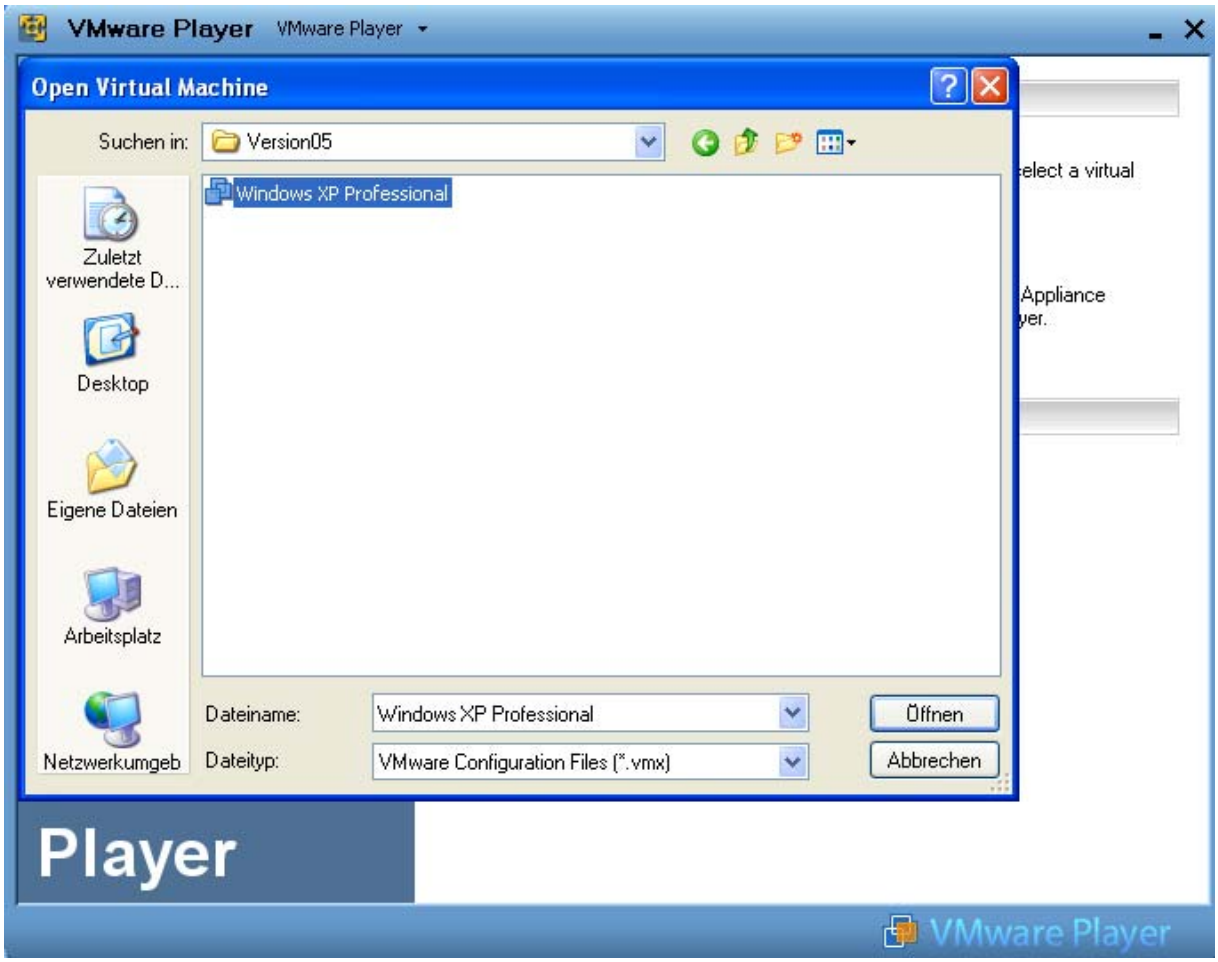
3. Vergessen Sie nicht die Änderung mit CTRL + S zu speichern und schließen Sie das Editorfenster.

4. Das VMWare Player Symbol ist nun auf dem Desktop. 2k um VMware Player aufzrufen.

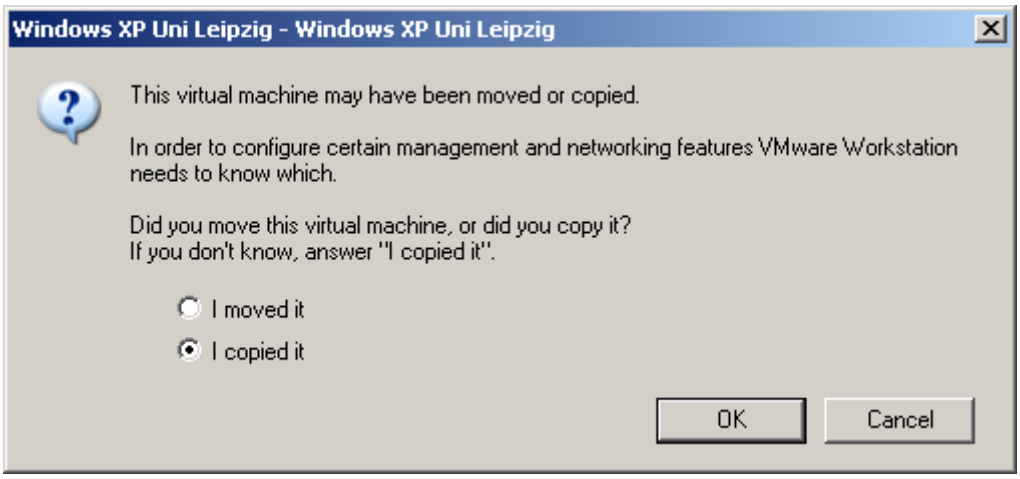
5. Auf dem Desktop erscheint das **VMware Player** Fenster.

   Im oberen Bereich unter Commands auf Open klicken, in das Verzeichnis wechseln, in dem Sie die Inhalte der DVDs gespeichert haben. 2k auf **Windows XP Professional.vmx**. Windows XP wird als Gastbetriebssystem geladen.

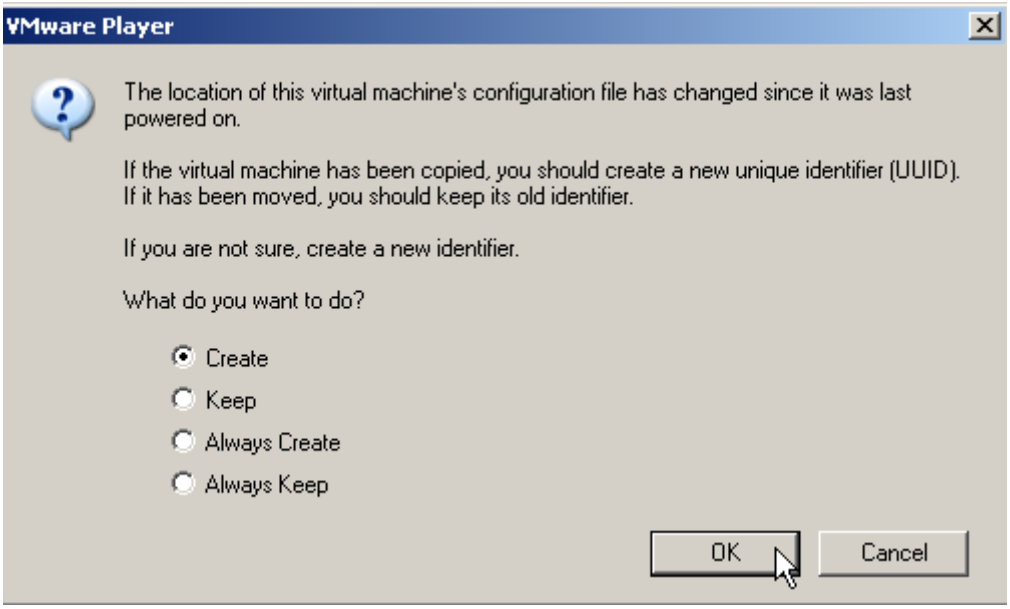
   (Note: Wenn Sie schon ein Mal eine virtuelle Maschine gestartet haben, dann können sie bequem unter Recent Virtual Machines eine der zuletzt verwendeten virtuellen Maschinen anklicken um sie zu laden.)



6. Dieses Fenster erscheint eventuell beim erstmaligen Aufruf der virtuellen Maschine. ***I copied it*** auswählen und ***OK*** **klicken**.

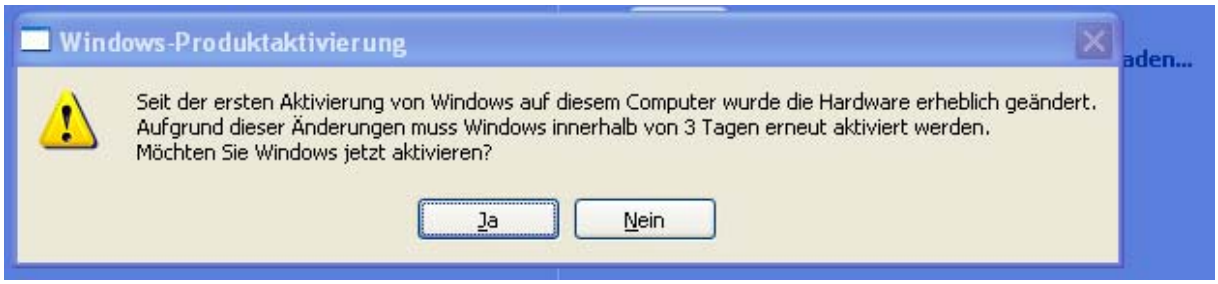


7. Auch dieses Fenster erscheint eventuell und nur beim erstmaligen Aufruf der virtuellen Maschine Create auswählen und ok klicken

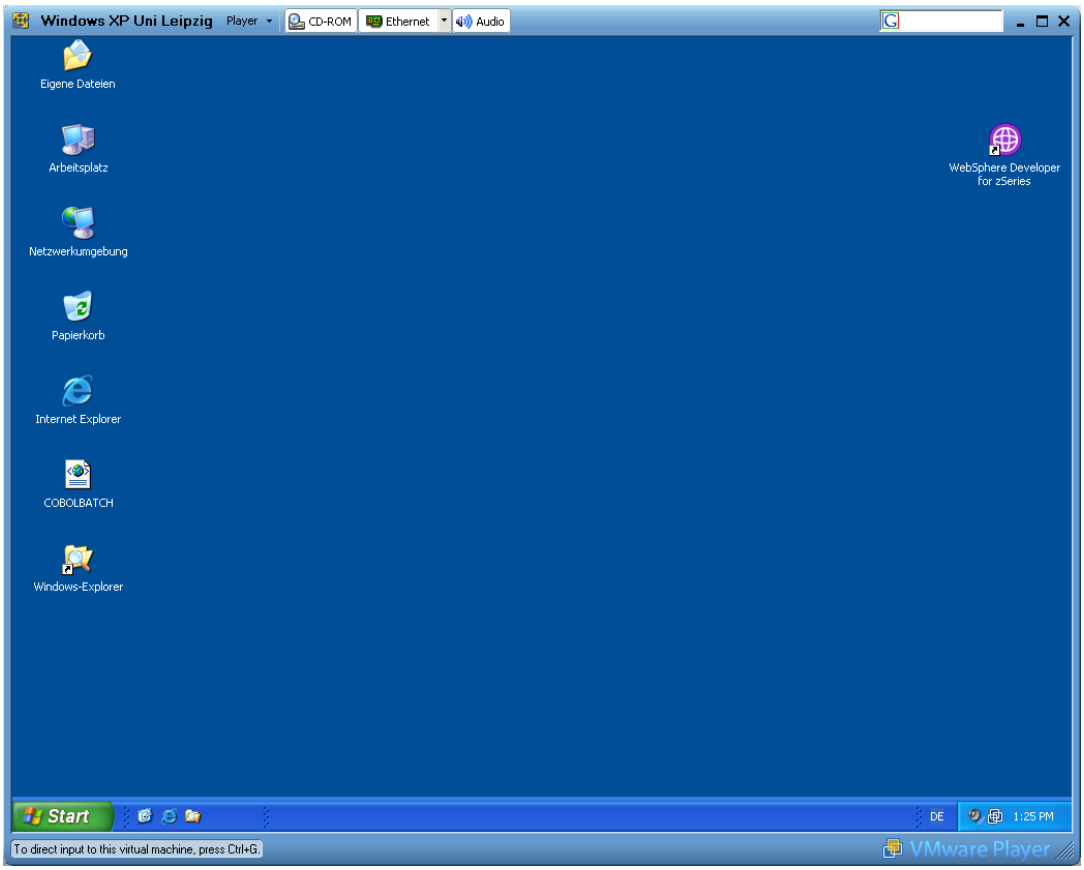


8. Zunächst 1k in das Feld „Kennwort“ (sonst ist keine Eingabe möglich). Als Passwort **unilp** eingeben. Einige Zeit warten, der VMWare Gast Desktop erscheint als Fenster innerhalb des Host Desktops.

9. Dieses Fenster muss nicht zwingend erscheinen. Sollte keine Benachrichtigung erscheinen, dann überspringen sie bitte den Rest dieses Abschnitts. Zunächst erst einmal nein. Die Registrierung muss zwar gemacht werden, aber die virtuelle Maschine hat möglicherweise noch keine Internet Verbindung. Geht also noch nicht – wir müssen erst die Internet Verbindung konfigurieren (möglicherweise reicht aber bereits die vorhandene Standard Konfiguration). Sie sollten aber registrieren, wenn sie das nächste Mal die virtuelle Maschine und Windows XP hochfahren. Wichtig **!!!**



10. Einige Zeit warten, der VMWare Gast Desktop erscheint als Fenster innerhalb des Host Desktops.



11. Wir haben jetzt hier ein Host Betriebssystem und seinen Host Desktop (beides war aktiv, ehe wir VMware starteten) und ein Gastbetriebssystem mit seinem Gast Desktop. Der Gast Desktop läuft in einem Fenster des Host Desktops.

    Möglicherweise ist das Fenster größer als der Bildschirm. Um die Fenstergröße an die Bildschirmgröße anzupassen, nach unten und nach links scrollen.

    Auf Start klicken, dann Einstellungen (1k) → Systemsteuerung (1k) → Anzeige (2k) →

    Einstellungen (1k). Auflösung auf 1024x768 einstellen (oder was immer für den verwendeten Bildschirm am besten ist).

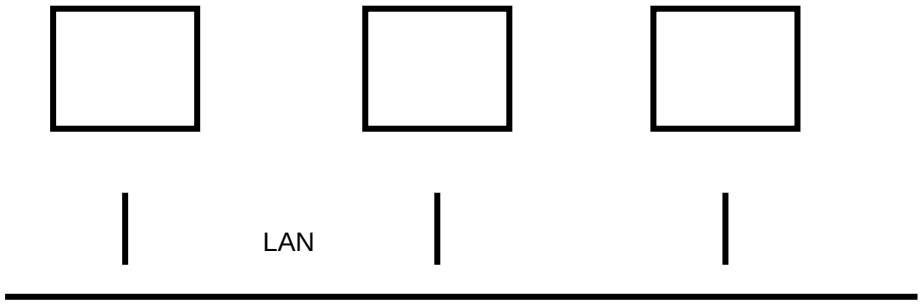
    Es ist sinnvoll (Geschmacksfrage), das Gast Desktop Fenster so groß zu machen, dass er fast (aber nicht ganz) den ganzen Bildschirm abdeckt. Dann können Sie mit einem Mausklick vom Gast Desktop zum Host Desktop wechseln. Das Gleiche erreichen Sie auch durch gleichzeitige Bedienung der Tasten Control + Alt.

    Sie wollen wahrscheinlich Daten zwischen dem Host und dem Gast austauschen. Dies geht über die Netzwerkverbindung. Hierzu geben Sie auf dem Host ein Laufwerk frei.

    Wenn Sie auf dem Gast System etwas in die Zwischenablage kopieren, können sie es im Host System aus der Zwischenablage übernehmen.

4 Netzwerk Konfiguration anpassen.

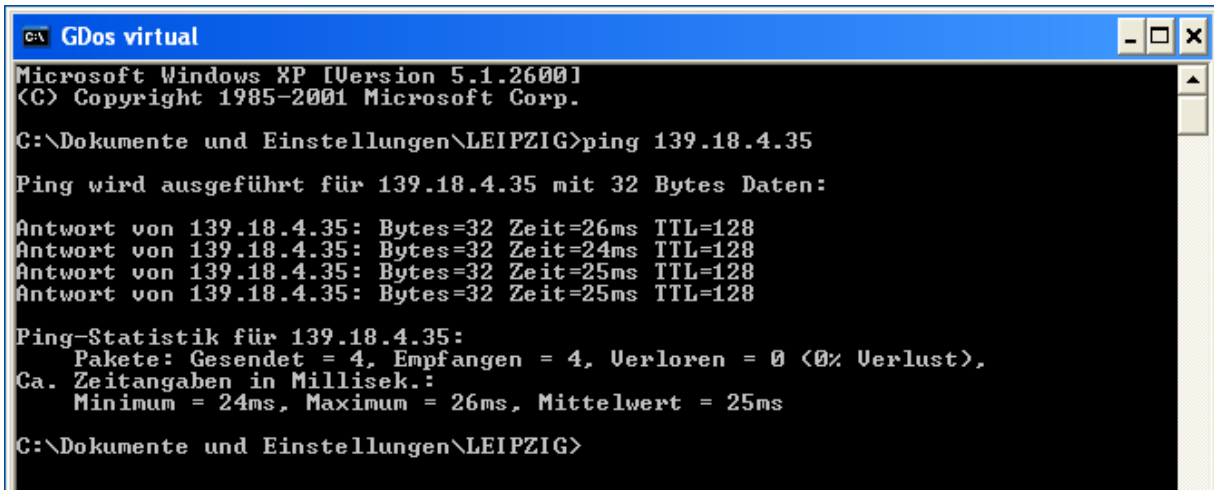
Ihre virtuelle Maschine ist ein eigener Windows XP Rechner, der nur zufällig als virtueller Gast auf Ihrem Host Windows XP Rechner läuft. Stellen Sie sich die folgende Konfiguration vor:



Gateway und DNS Server 192.168.0.1	Gast Rechner	Host Rechner

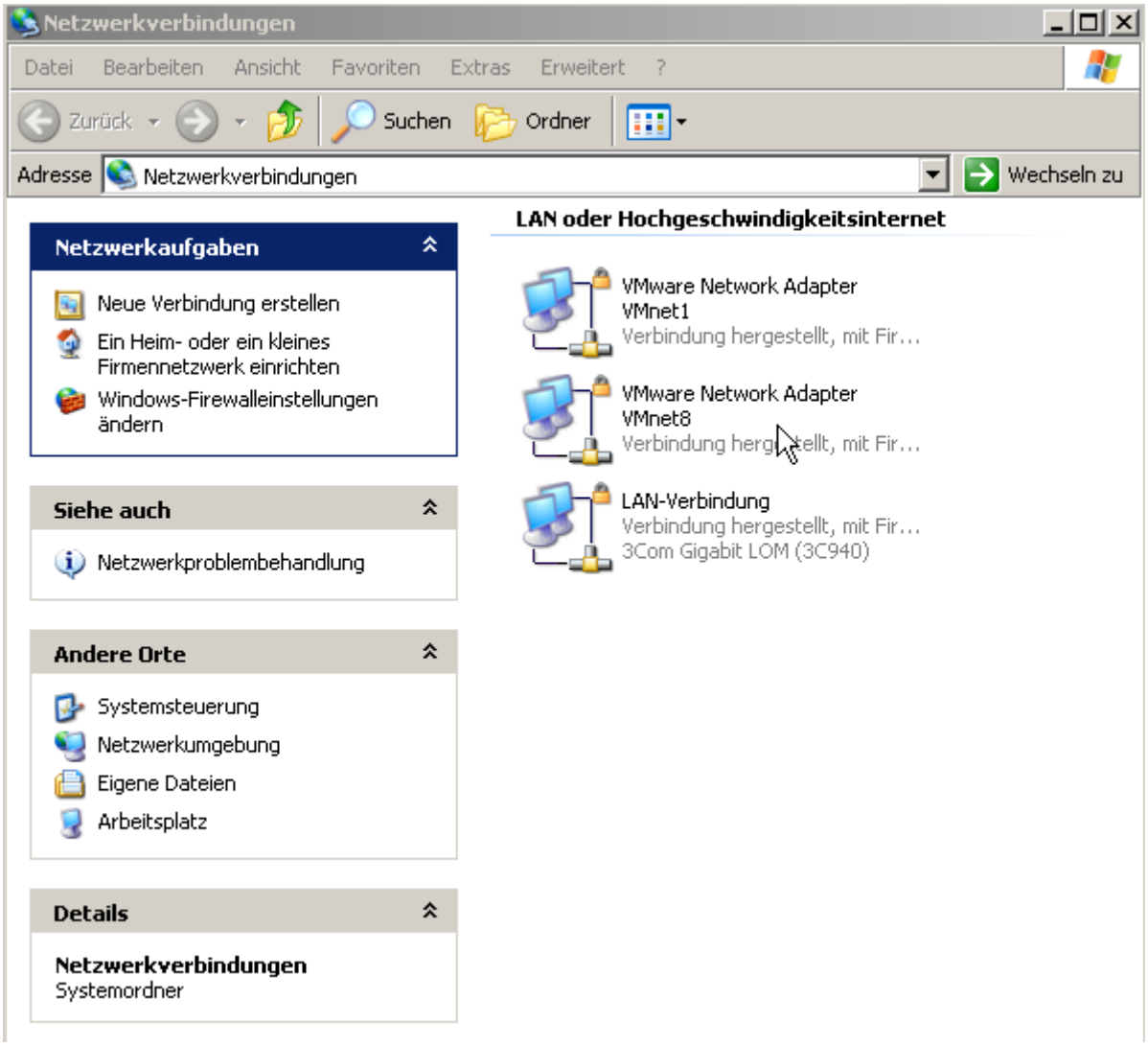
Diese Konfiguration ist vorkonfiguriert.

1. Jetzt öffnen Sie eine Shell in der virtuellen Maschine (Start → Ausführen… → cmd eingeben) und geben Sie **ping** auf **139.18.4.34** (unser Rechner binks.informatik.uni-leipzig.de) ein.



2. Ob das ping erfolgreich ist, hängt von der Konfiguration Ihres Rechners ab. Wenn das ping erfolgreich war, können Sie den Rest von Abschnitt 4 überspringen und gleich zu Abschnitt 5 **In Betriebnahme von RDZ** gehen. Andernfalls fahren sie fort mit Abschnitt 4 **Netzwerk Konfiguration anpassen**.

3. Zur Verifizierung: Im Host – Netzwerkumgebung 1kr, Eigenschaften 1k (Bitte im Host Fenster, nicht im Gast Fenster)

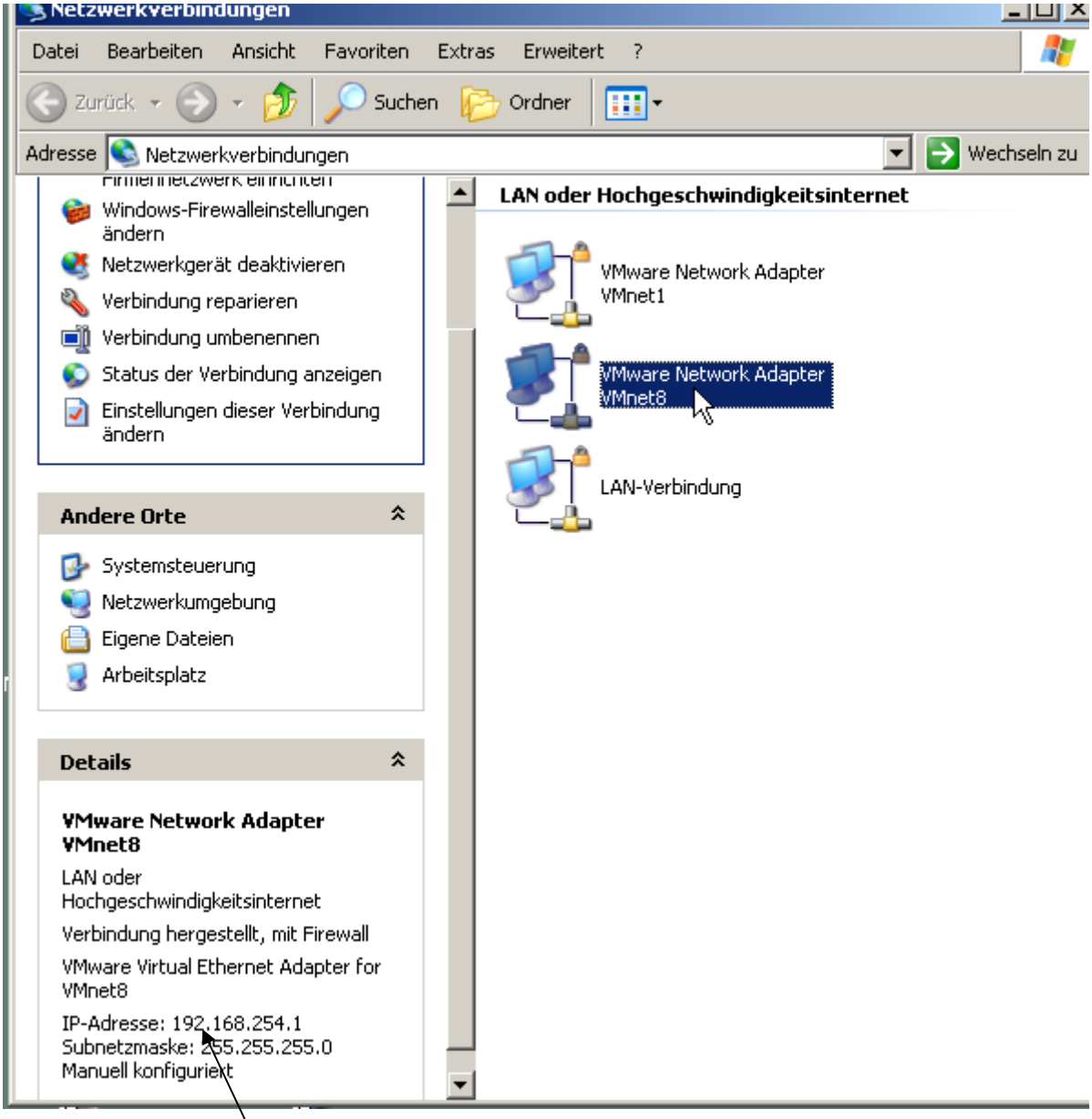


4. Angenommen, der Rechner mit dem Host Betriebssystem ist an ein Ethernet LAN angeschlossen. Dies wird durch das Symbol **LAN Verbindung** dargestellt. Beim ersten Aufruf der Virtuellen Maschine wurden im Host zusätzlich 2 VMware Network Adapter installiert. Diese werden benutzt um zusätzlich zu dem bereits vorhandenen realen Netzwerk ein weiteres virtuelles Netzwerk mit der virtuellen Maschine herzustellen.

   Vmnet1 bezieht sich auf Host-only Verbindung

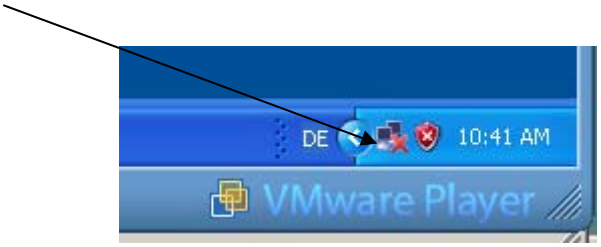
   Vmnet8 ist NAT Verbindung, erlaubt Zugriff von der virtuellen Maschine in das Internet. Diese benutzen wir.

   1k auf Vmnet08

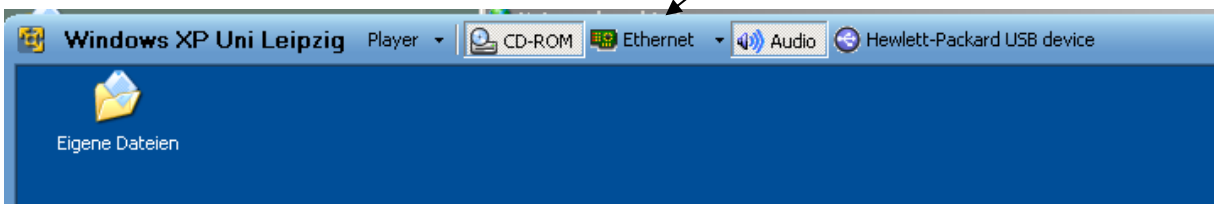


Die Adresse 192.168.254.1 ist vorkonfiguriert. Dies ist ok.

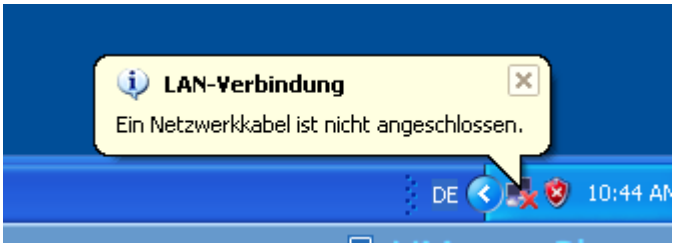
5. Jetzt die Virtuelle Maschine hochfahren. Rechts unten auf dem Gast Desktop sehen sie, wenn kein LAN Kabel angeschlossen ist.



6. Auf der Windows Oberfläche der Gast Maschine: Das Feld `Ethernet` ist nicht highlighted. 1k auf dieses Feld, etwas warten

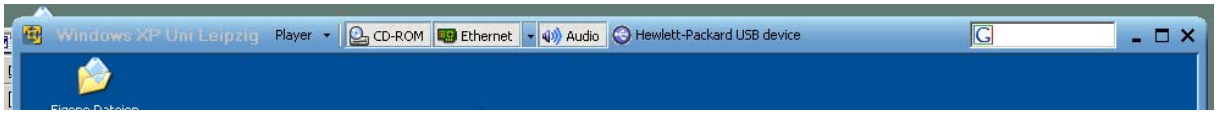


7. Zunächst existiert keine LAN Verbindung.

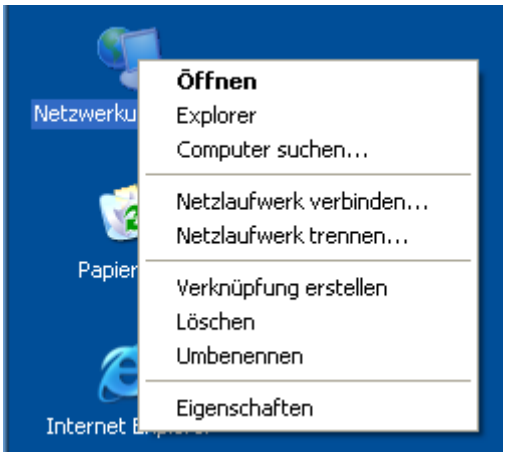


8. Nach einiger Zeit:

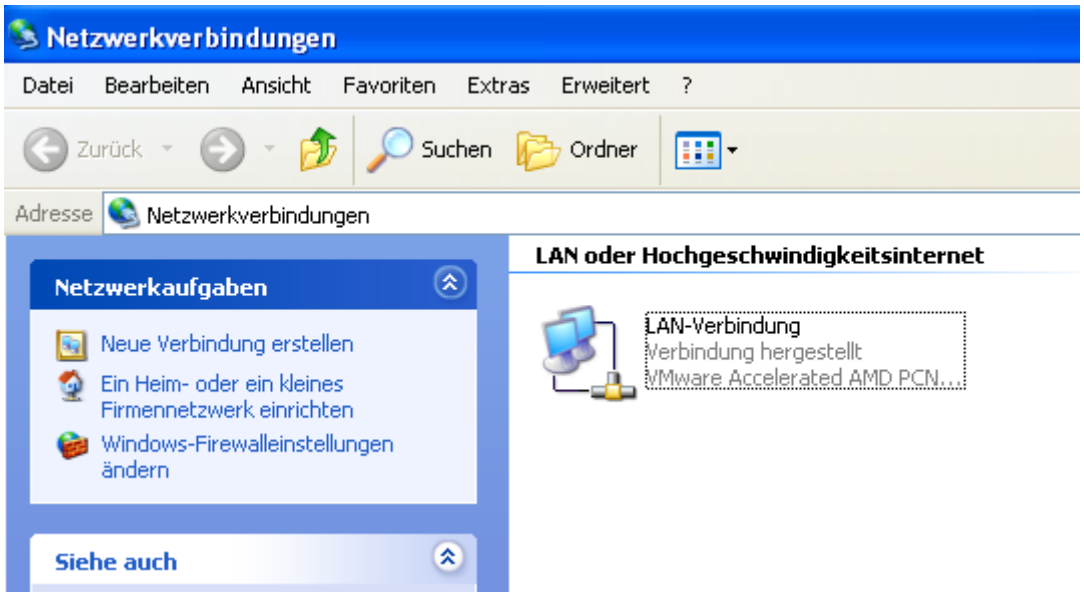
9. Den Mauszeiger vom Feld Ethernet wegziehen. Jetzt ist das Feld **`Ethernet`** highlighted.



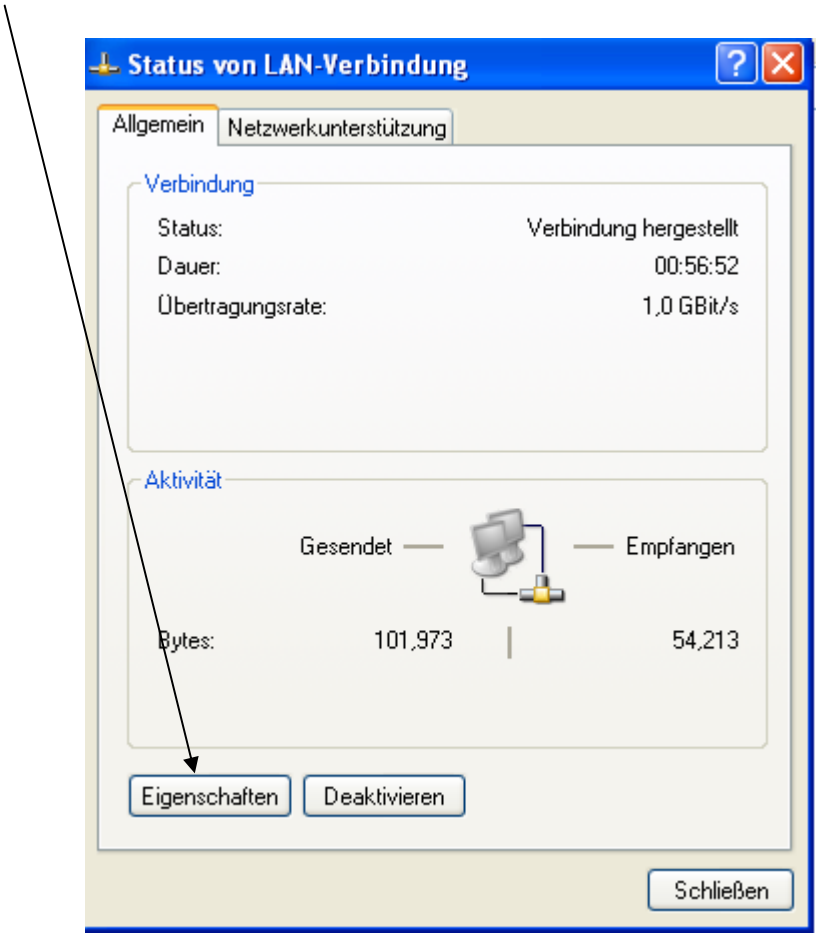
10.1kr auf Netzwerkumgebung, 1k auf Eigenschaften.



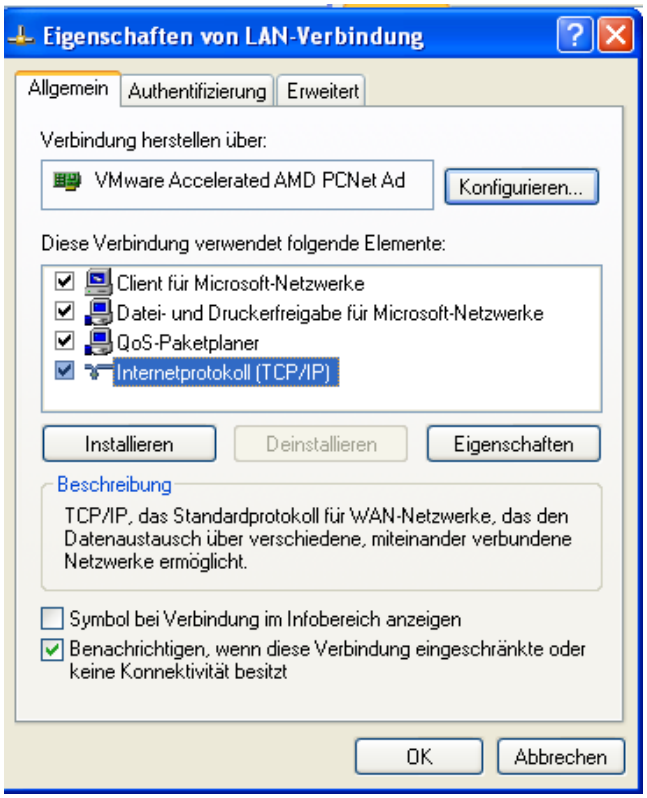
11.1k auf LAN-Verbindung.



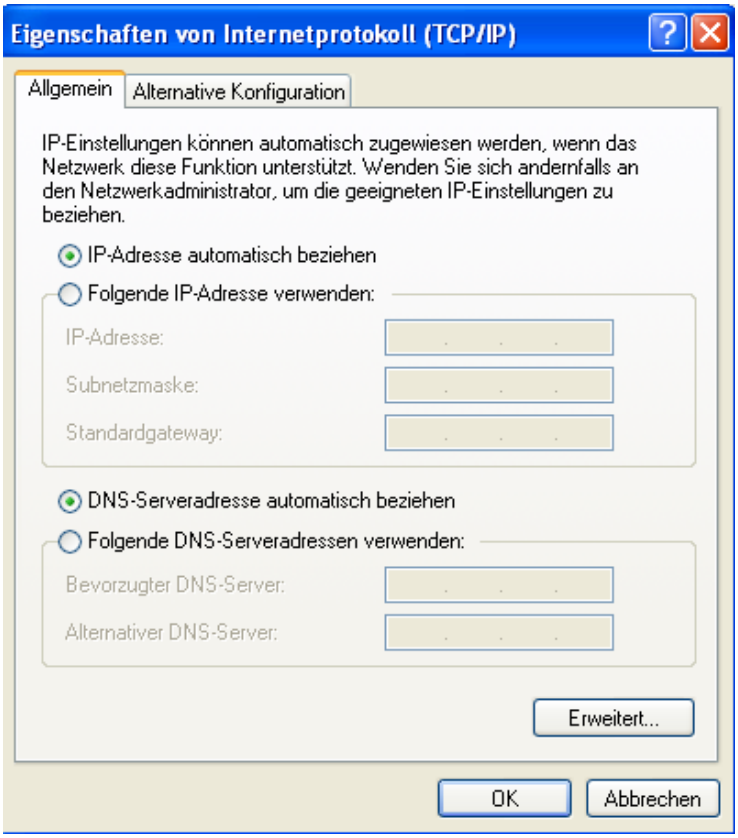
12.1k auf Eigenschaften.



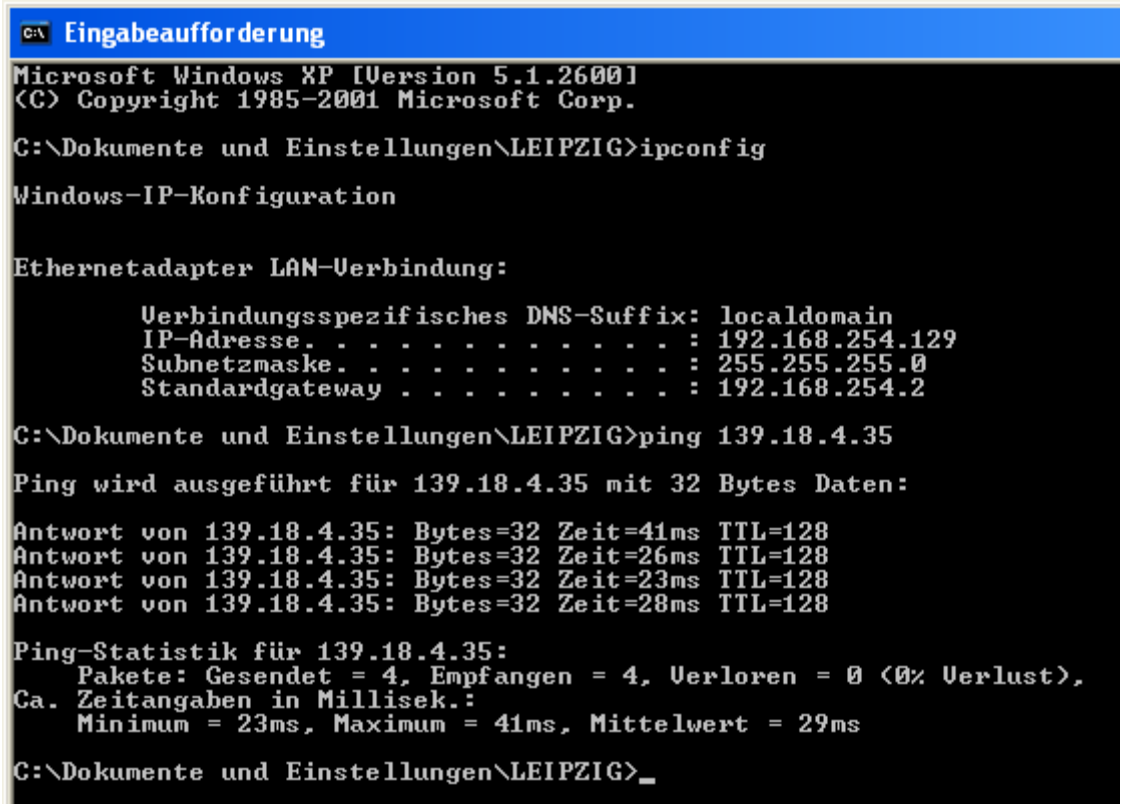
13.1k auf Internet Protokoll



14. IP Adresse automatisch beziehen und DNS Server Adresse automatisch beziehen muss eingestellt sein. Ok, Schließen



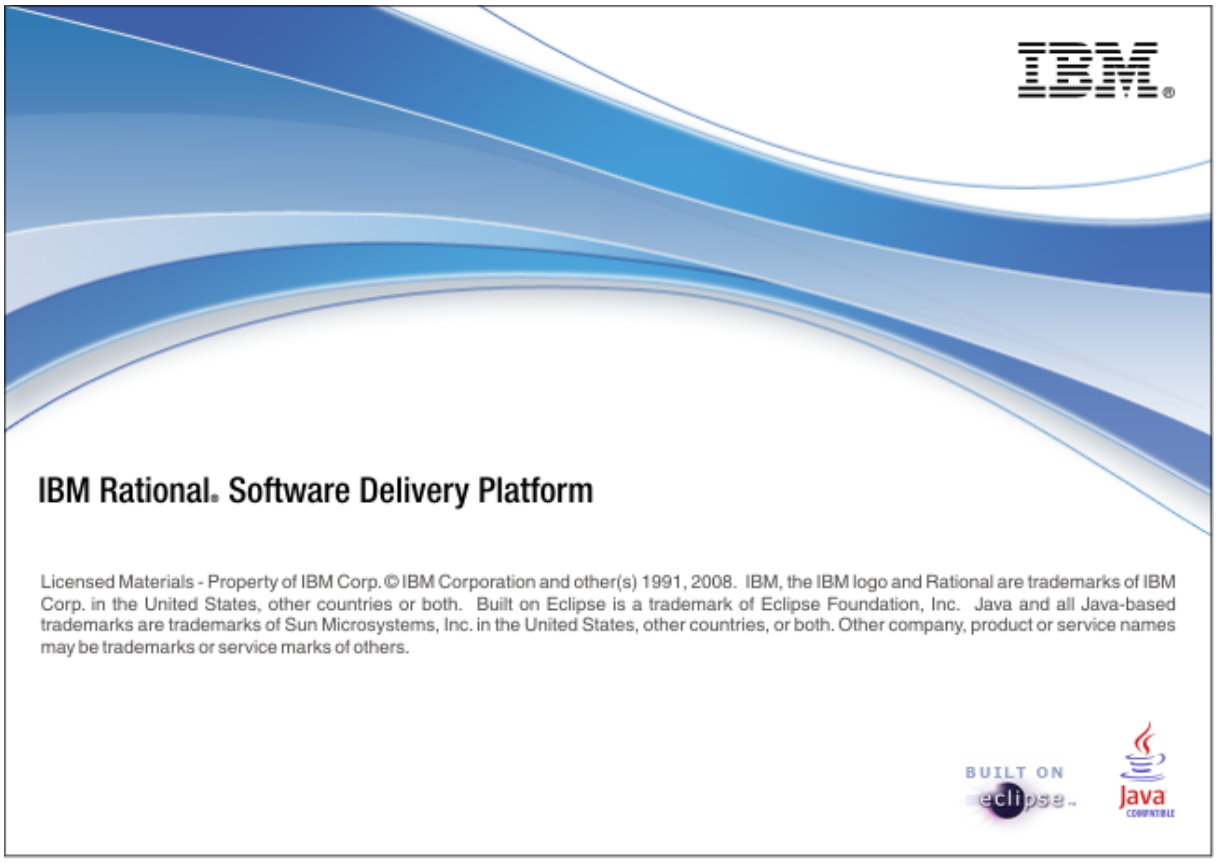
15. ipconfig zeigt die eingestellten Adressen an. Der Internet Zugang läuft jetzt. Test mit ping auf Binks. (**139.18.4.34).**



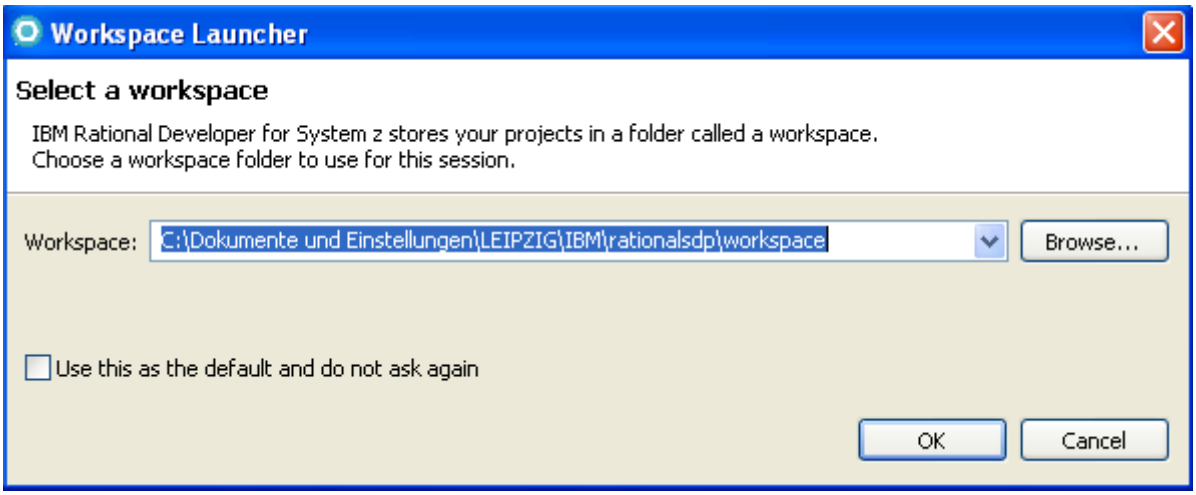
16. Ebenfalls funktioniert der Internet Zugriff mit dem Windows Internet Explorer.

5 Inbetriebnahme von RDZ

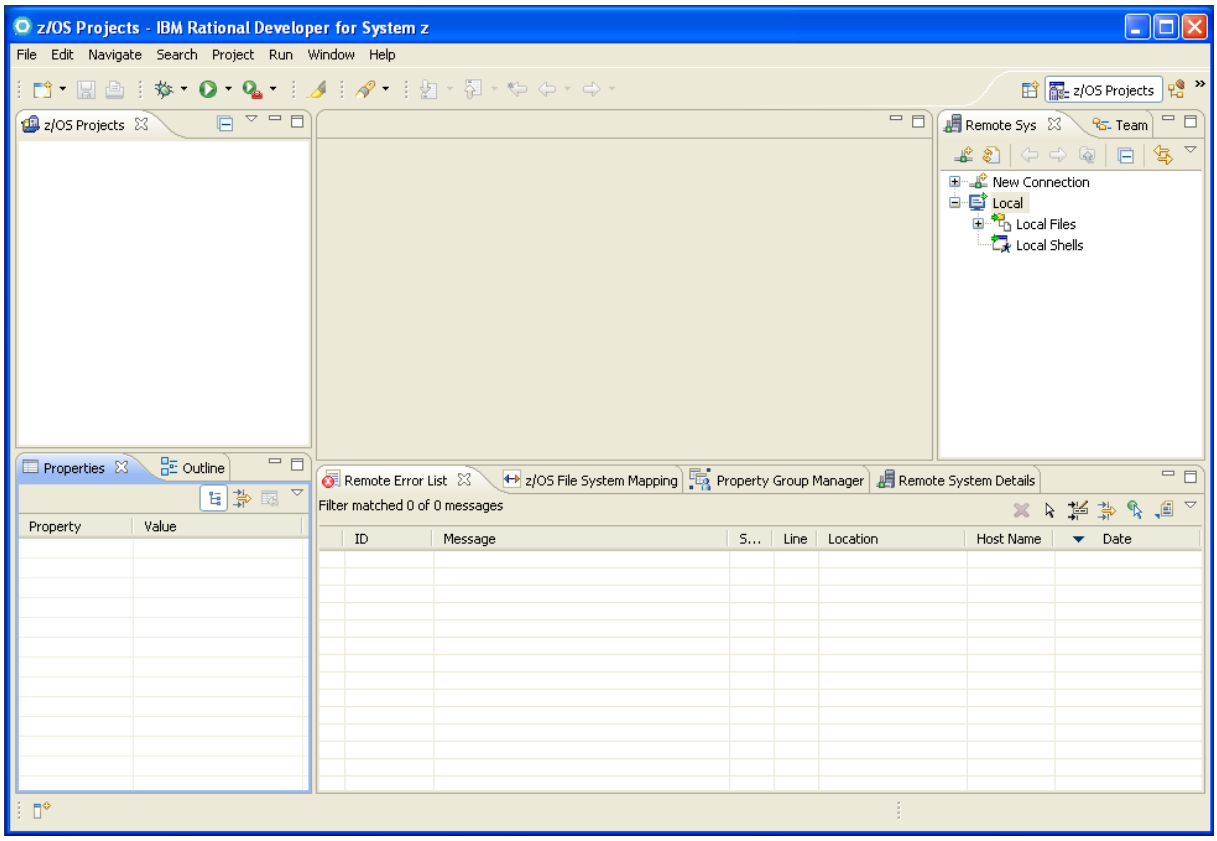
1. Auf dem Gast System öffnen sie RdZ über Start → Programme IBM Software Delivery Platform → IBM Rational Developer for System z → IBM Rational Developer for System z.



2. RDZ bietet Ihnen an, einen voreingestellten Workspace zu verwenden. Dies ist der Ort an dem Ihre Projekte etc. abgelegt werden. Sie könnten auch Ihren eigenen Workspace verwenden, z.B. **C:\WDz\Workspace.** Klicken sie OK.



3. RdZ ist nun geöffnet. Schließen sie ggf. den Welcome screen auf der rechten Seite.



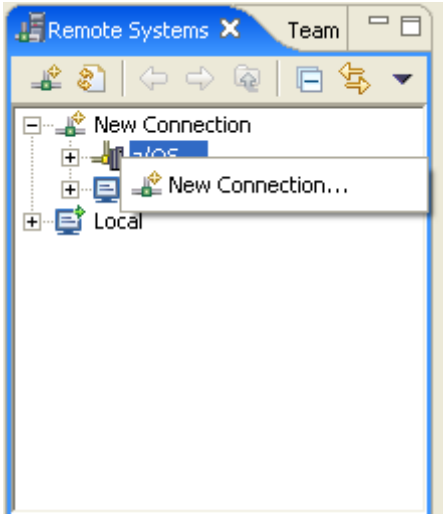
6 Zugriff auf den remote z/OS Host (BINKS)

1. Vergewissern Sie sich, dass Sie mit der z/OS perspective arbeiten in dem Sie in der Menüleiste auf: Window -> Open Perspective -> Other… gehen und dann mit Doppelklick z/OS Projects auswählen.

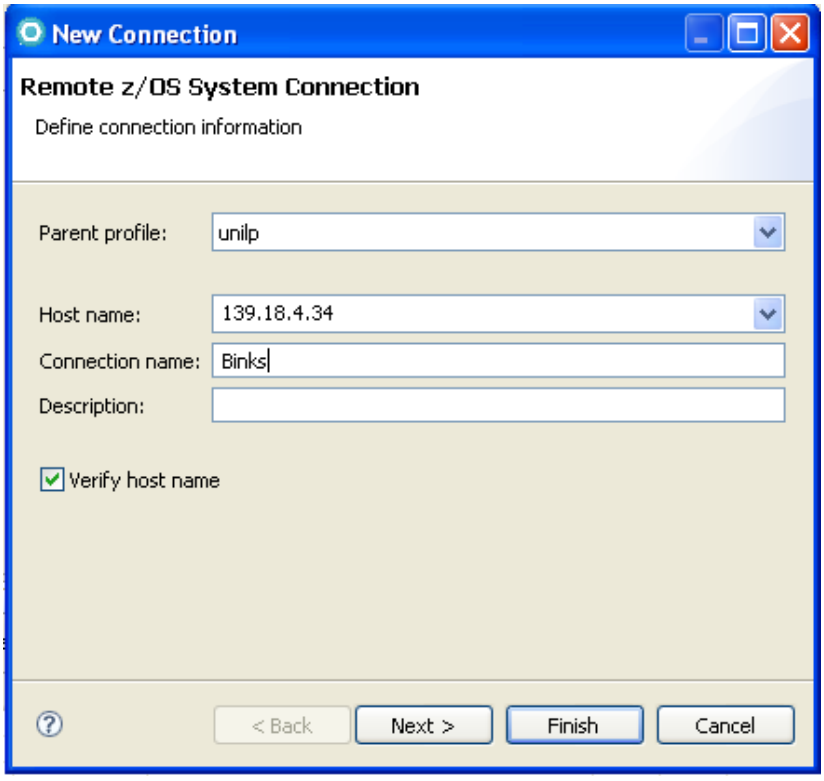
   (Falls ein Info-Fenster auftaucht, schließen Sie es mit dem üblichen Klick auf das Kreuz)

   Nun können wir unsere Verbindung zum Host erstellen.

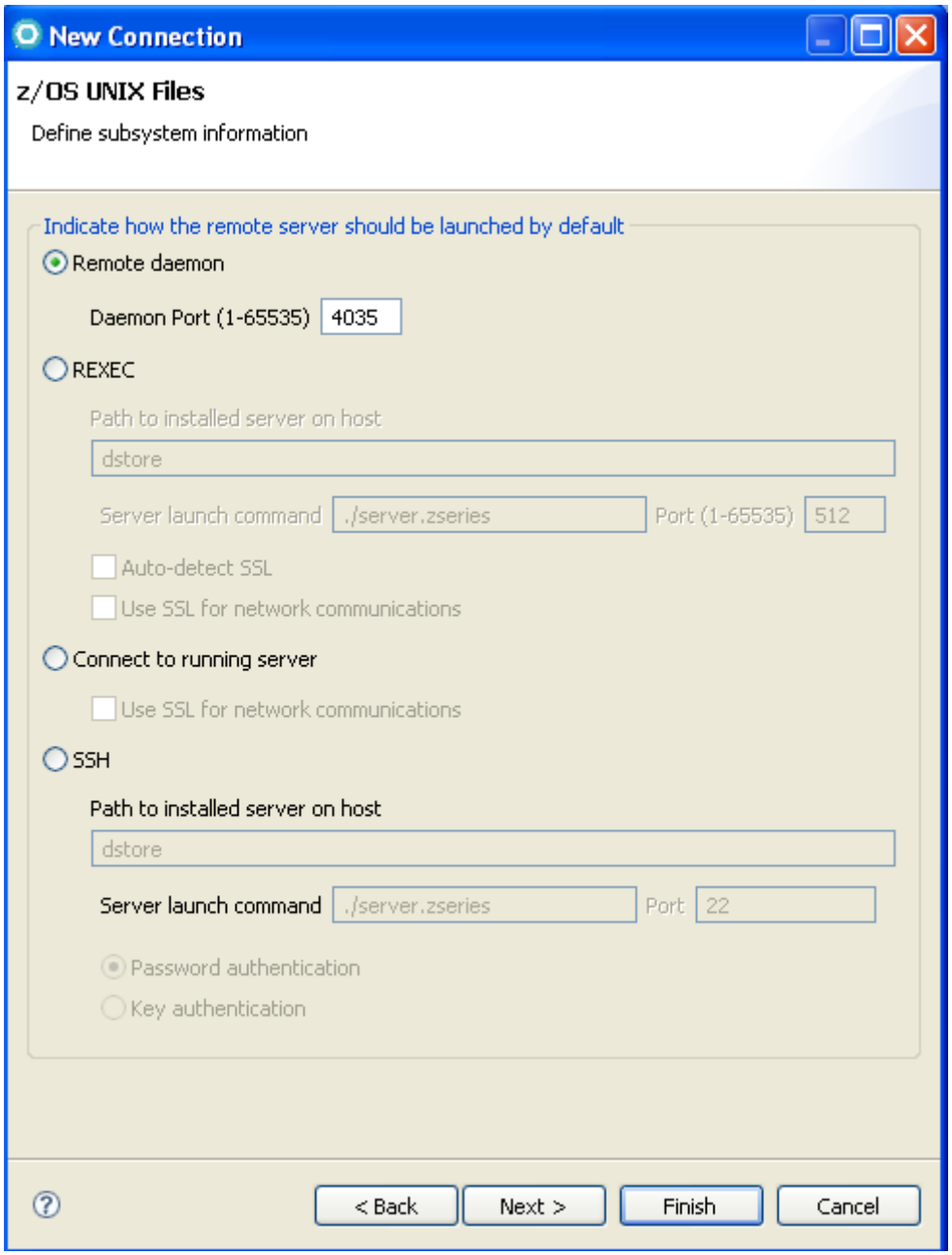
   Rechtsklick auf auf New Connection ? z/OS 1k, und New Connection… auswählen.



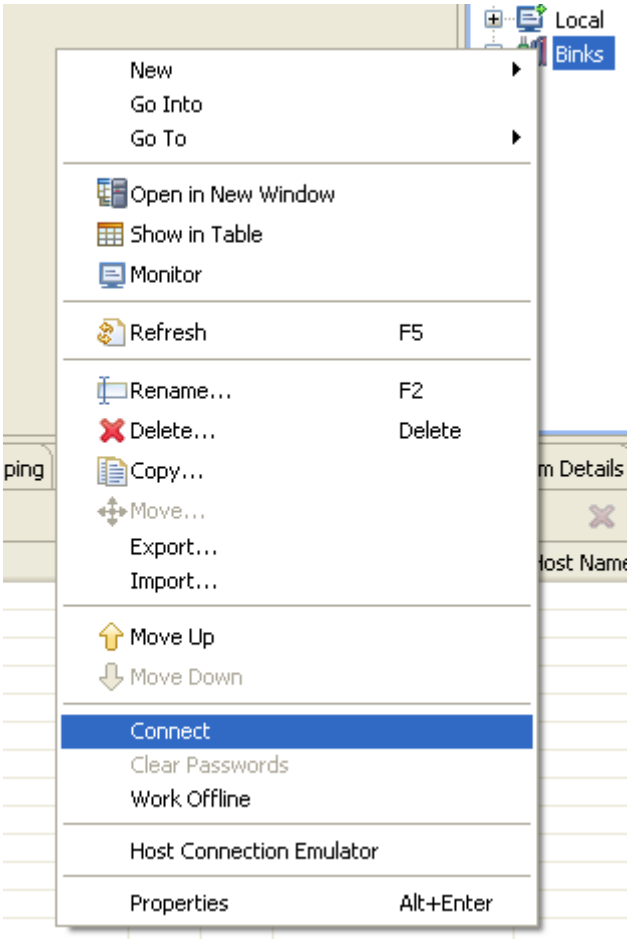
2. 1kr auf **z/OS** , 1k auf **New Connection…**.
   139.18.4.34 als *Host name*, für *Connection name* **Binks** eingeben, 1k auf **Next**



3. Übernehmen wie vorgeschlagen, **Finish**

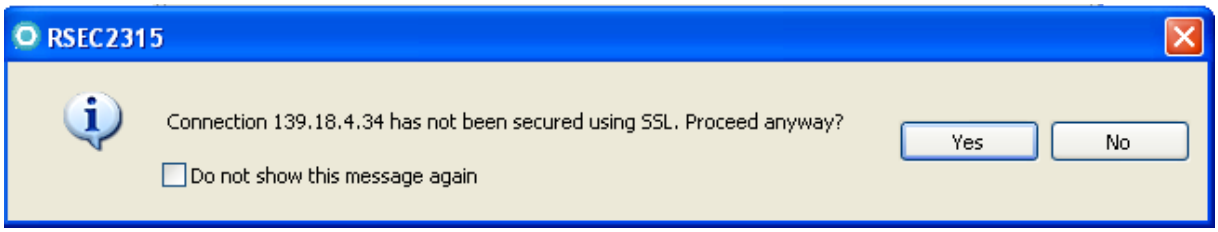


4. Binks ist nun als **Remote System** verfügbar. Wir erstellen nun eine Verbindung mit dem remote host. 1kr auf Binks und 1k auf **Connect**

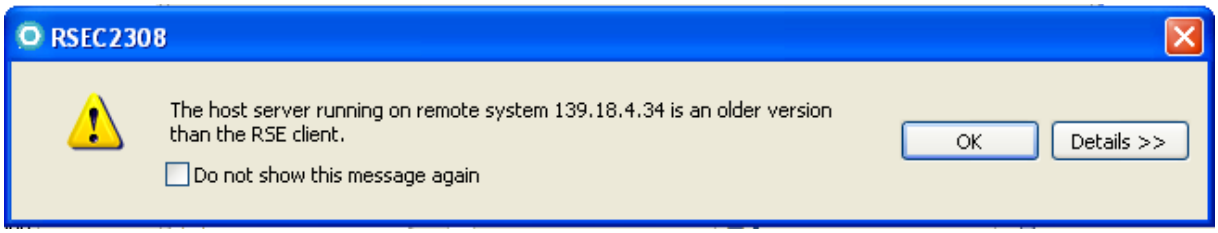


5. Wir benutzen als Beispiel eine User ID, die schon (von einem anderen Rechner aus) intensiv benutzt wurde. Sie würden hier Ihre eigene User ID und Ihr Passwort benutzen.

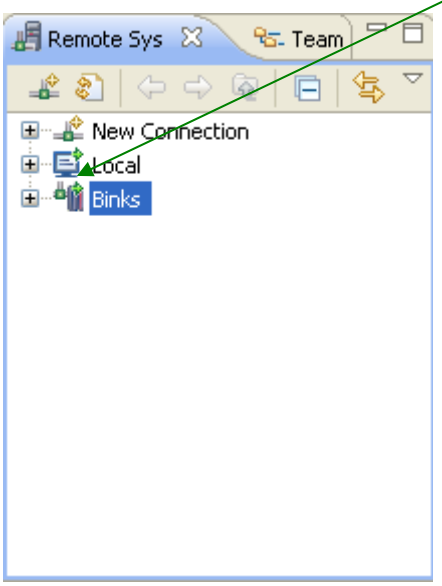
6. 1k auf yes



7. Diese Meldung kann auch mit OK bestätigt werden. Wir werden einfach nur darauf hingewiesen, dass auf dem Host System eine ältere Version läuft. Dies stellt aber kein Problem dar, da RdZ abwärtskompatibel ist.

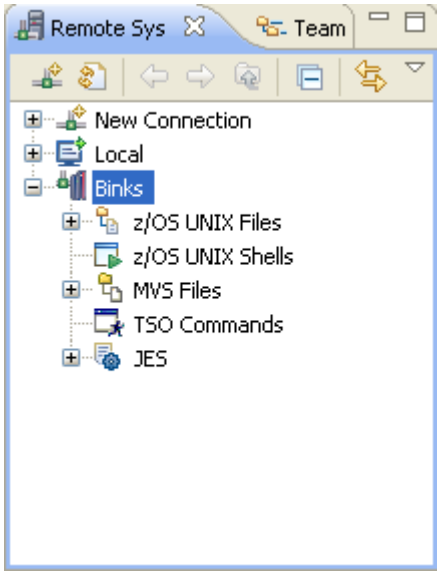


8. Sie haben nun erfolgreich eine Verbindung zum Host aufgebaut. Der winzige grüne Pfeil besagt, die Verbindung ist aufgebaut.

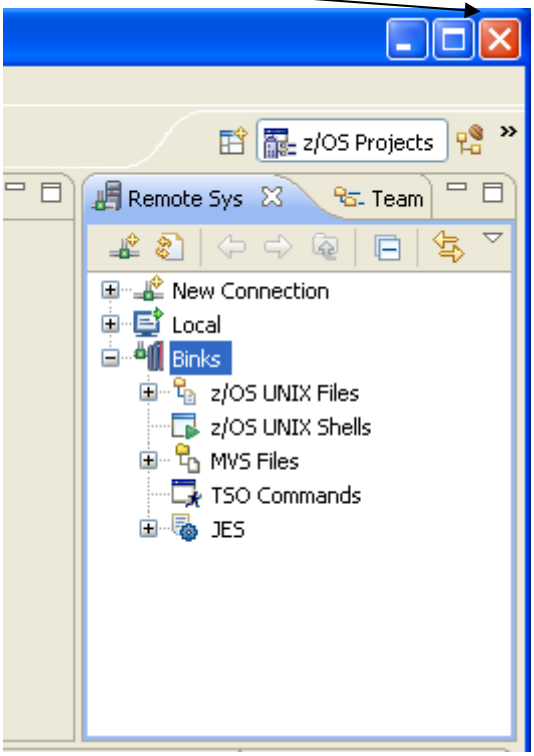


9. Damit ist RDZ funktionsfähig.

   Wir schließen nunmehr die Sitzung ab.1kr auf Binks → Disconnect. Der kleine grüne Pfeil ist nun verschwunden.



10. 1k auf das Kreuz rechts oben. Dies beendet RDz. Folgendes Fenster noch mit OK bestätigen.



7 Vorbereitung für die folgenden Tutorials

Einige der folgenden Tutorials benötigen Daten auf dem virtuellen-System, die sich auf der auf den Seiten des Lehrstuhls im Internet befinden. Außerdem befinden die Tutorials sich im Verzeichnis *Tutorials* auf der ersten DVD. Um die Daten jederzeit verfügbar zu haben sollten diese auf das Gastsystem kopiert werden. Dies geschieht über die interne Netzwerkverbindung zwischen Host- und Gastsystem.

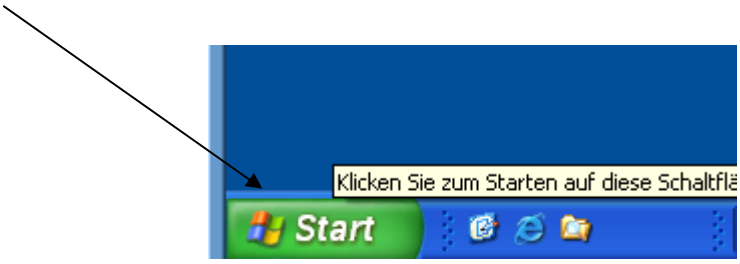
Laden sie sich nun alle benötigten Dateien herunter und speichern sie sie in einem geeigneten Ordner in dem Gastsystem.

Die Dateien befinden sich nun auf dem Gastsystem und Sie können mit den folgenden Tutorials beginnen.

8 Herunterfahren

1. Jetzt die virtuelle Maschine schließen. Das Windows Betriebssystem der virtuellen Maschine muss wie ein reguläres Betriebssystem heruntergefahren werden.

   1k auf Start im Fenster der virtuellen Maschine.



2. Ausschalten. Das Herunterfahren kann einige zeit beanspruchen. Damit wird das Fenster, in dem die virtuelle Maschine läuft, geschlossen. Das Betriebssystem des Hosts ist noch aktiv.

Damit ist das RDz Tutorial 01 abgeschlossen. Sie sind jetzt in der Lage mit RDz 7.5 zu arbeiten.

Anlage 02

RDz Tutorial 02

Local PL/1

RDz Tutorial 02
Local PL/1

This Tutorial will take you through the steps of using the z/OS Application Development component of Rational Developer for zSeries version 7.5 to work with local systems. In this Tutorial, we will be working with a PL/1 program, including editing, compiling, and debugging it. After finishing the tutorial you should be familiar with the basic workstation facilities of RDz, and should have acquired the knowledge to develop simple PL/1 applications using RDz.

Overview:

1. Write a Local PL/1 program
2. Working with a Local PL/1 program
   - 2.1.Edit a PL/1 source file
   - 2.2.Checking the PL/1 source file syntax
3. Compiling, linking and executing the local PL/1 program
   - 3.1.Creating an executable PL/1 program
   - 3.2.Check the local PL/1 compilation and link edit
   - 3.3.Creating a Run launch configuration and executing the PL/1 program
4. Write another local PL/1 program
   - 4.1. Creating the program
   - 4.2. Explore the code
   - 4.3. Execute the program
5. Testing/Debugging the PL/1 Program

The original COBOL version of this tutorial has been prepared by Mrs Isabel Arnold and presented in a z/OS Summer class at Hamburg University. It has been modified for PL/1 by Mr. Karsten Kunze at the Department of Computer Science at Leipzig University.

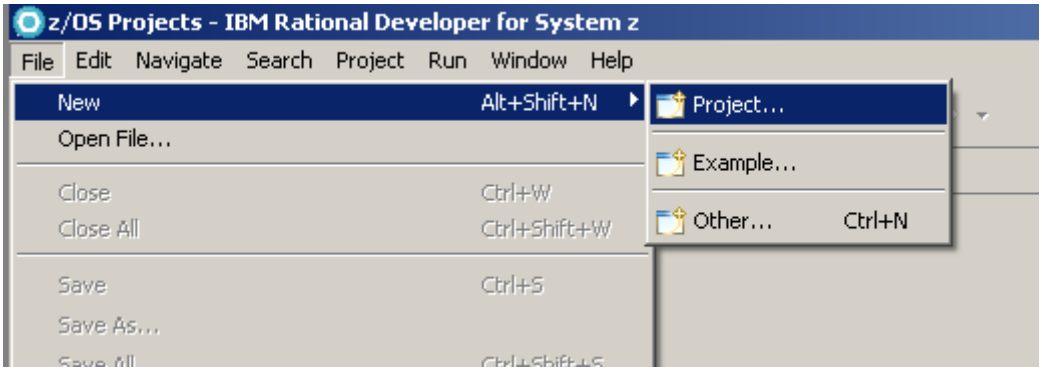
This tutorial uses the following conventions

1k	means 1 click with the left mouse button
2k	means 2 click with the left mouse button
1kr	means 1 click with the right mouse button

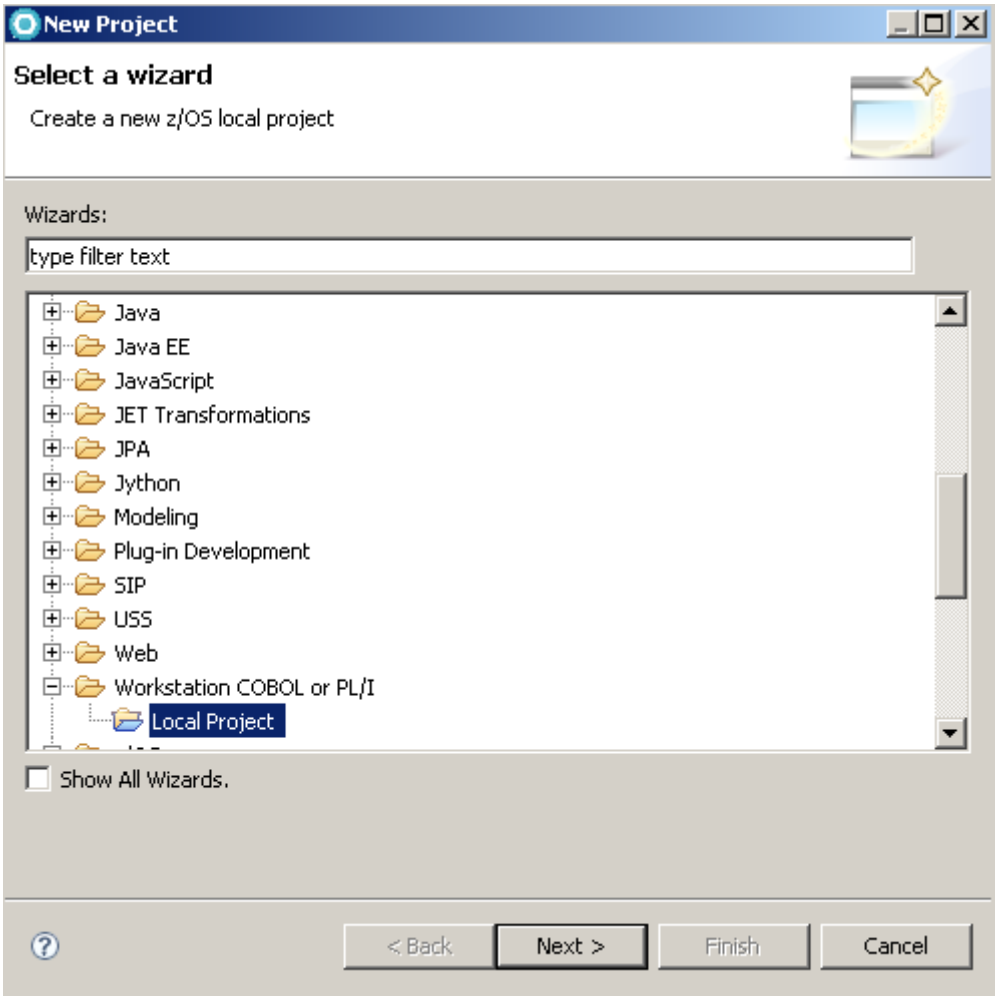
1 Write a Local PL/1 program

You will now write a sample PL/1 program into your local workspace.

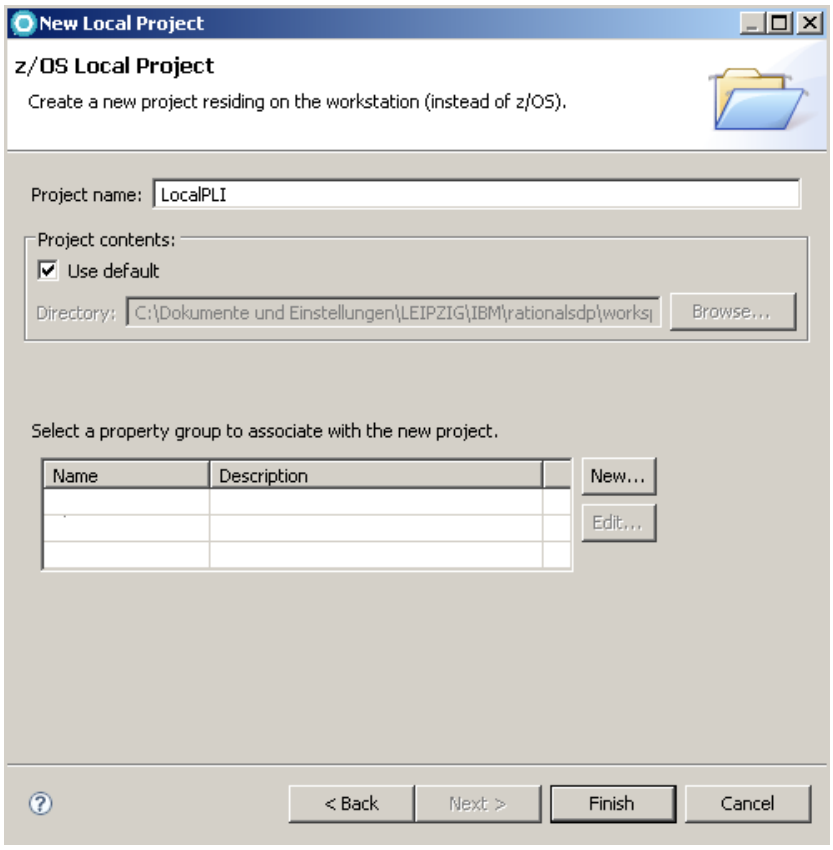
1. From the z/OS Projects perspective, select **File → New → Project** at the menu bar.



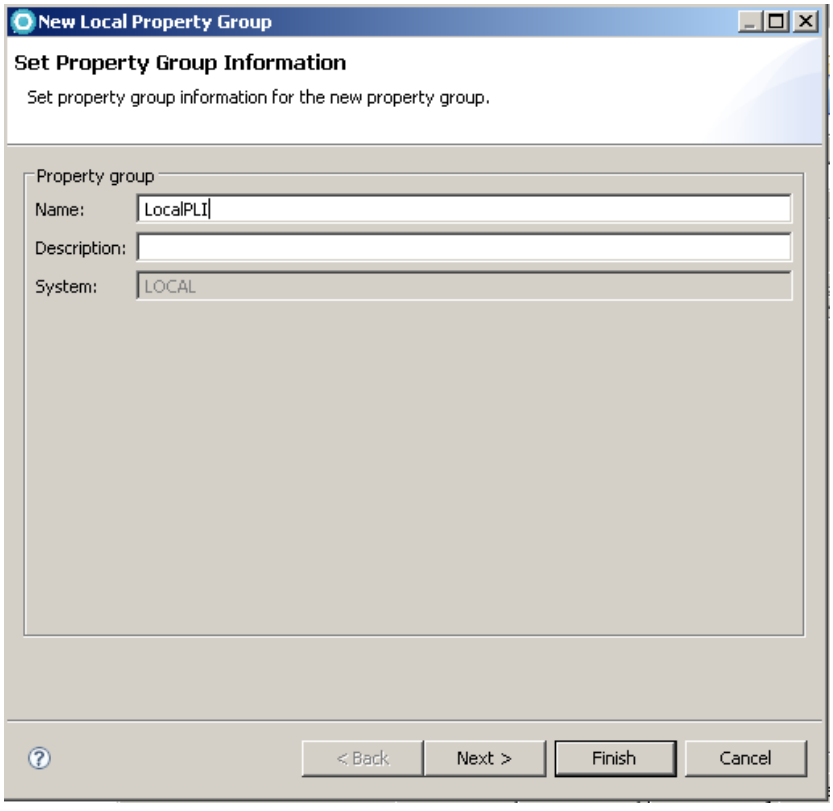
2. In the Select a wizard panel, select Workstation COBOL or PL/I → Local Project. Click next.



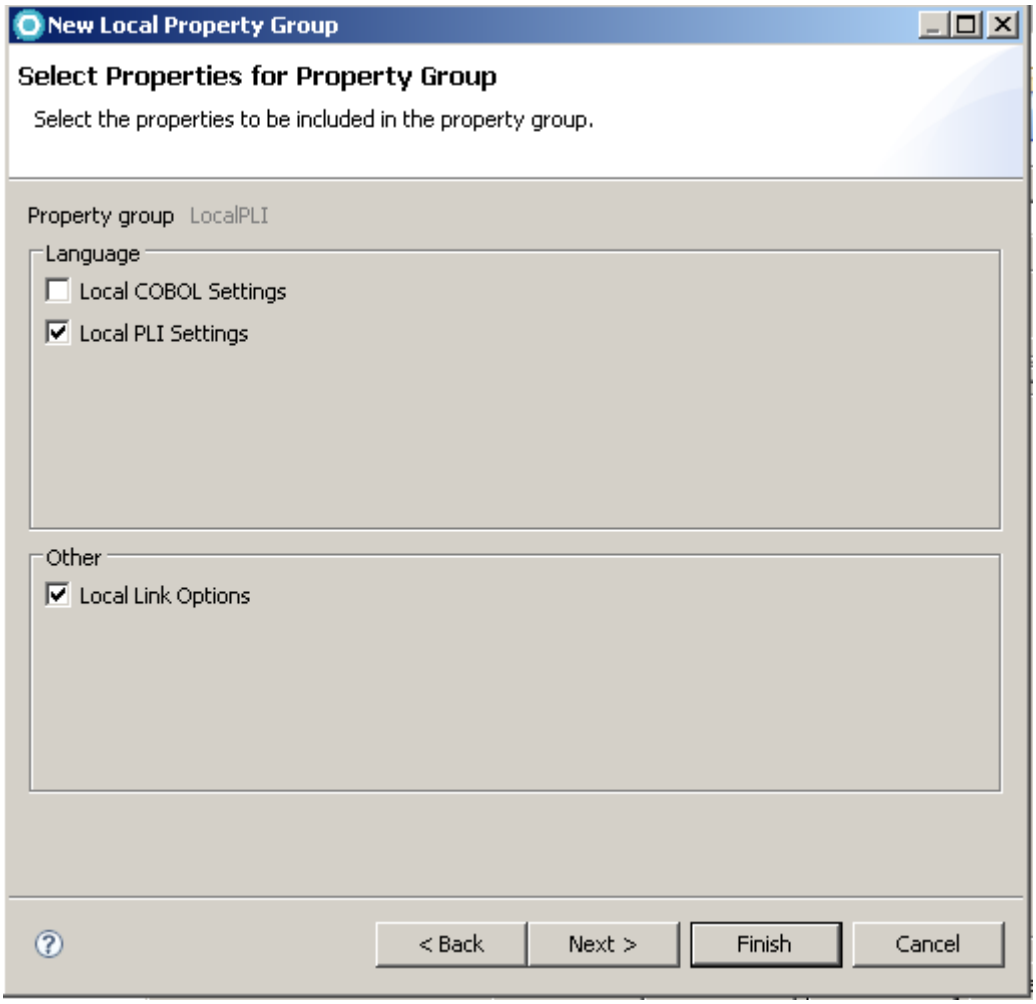
3. .On this panel, enter **LocalPLI** as the Project name, check **Use default** and click **New**.



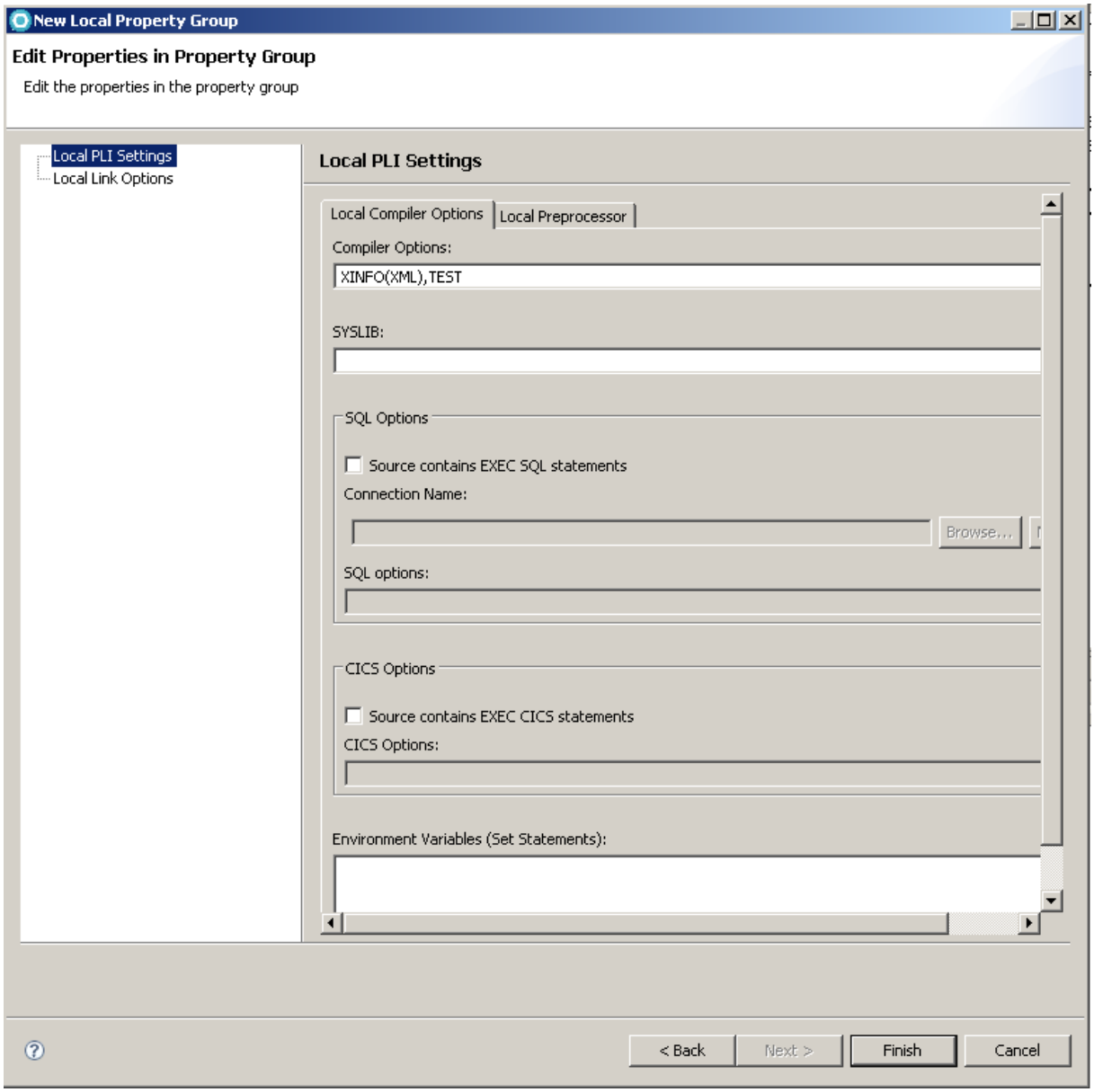
4 Now we have to create a new Property Group for our Project. Enter “LocalPLI” as Name and click Next



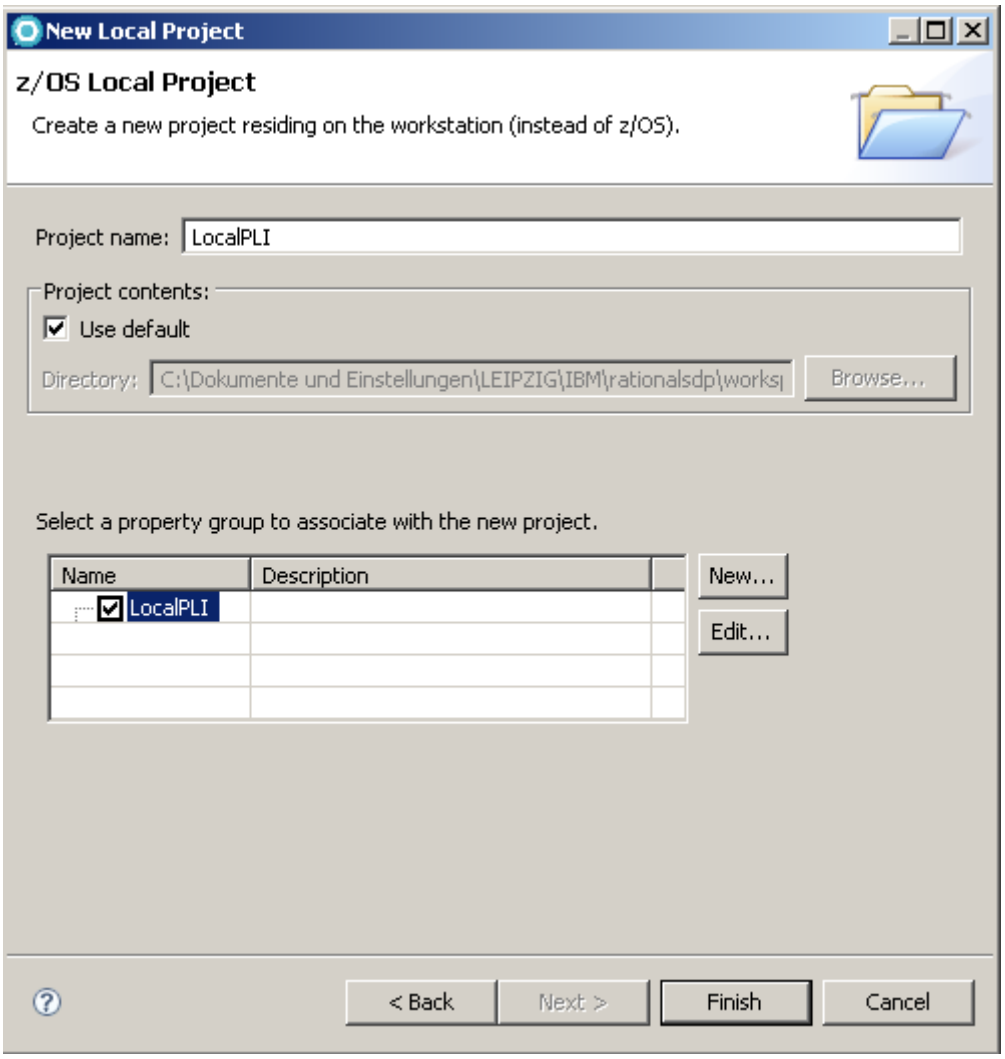
5. Uncheck “Local COBOL Settings” and Click Next.



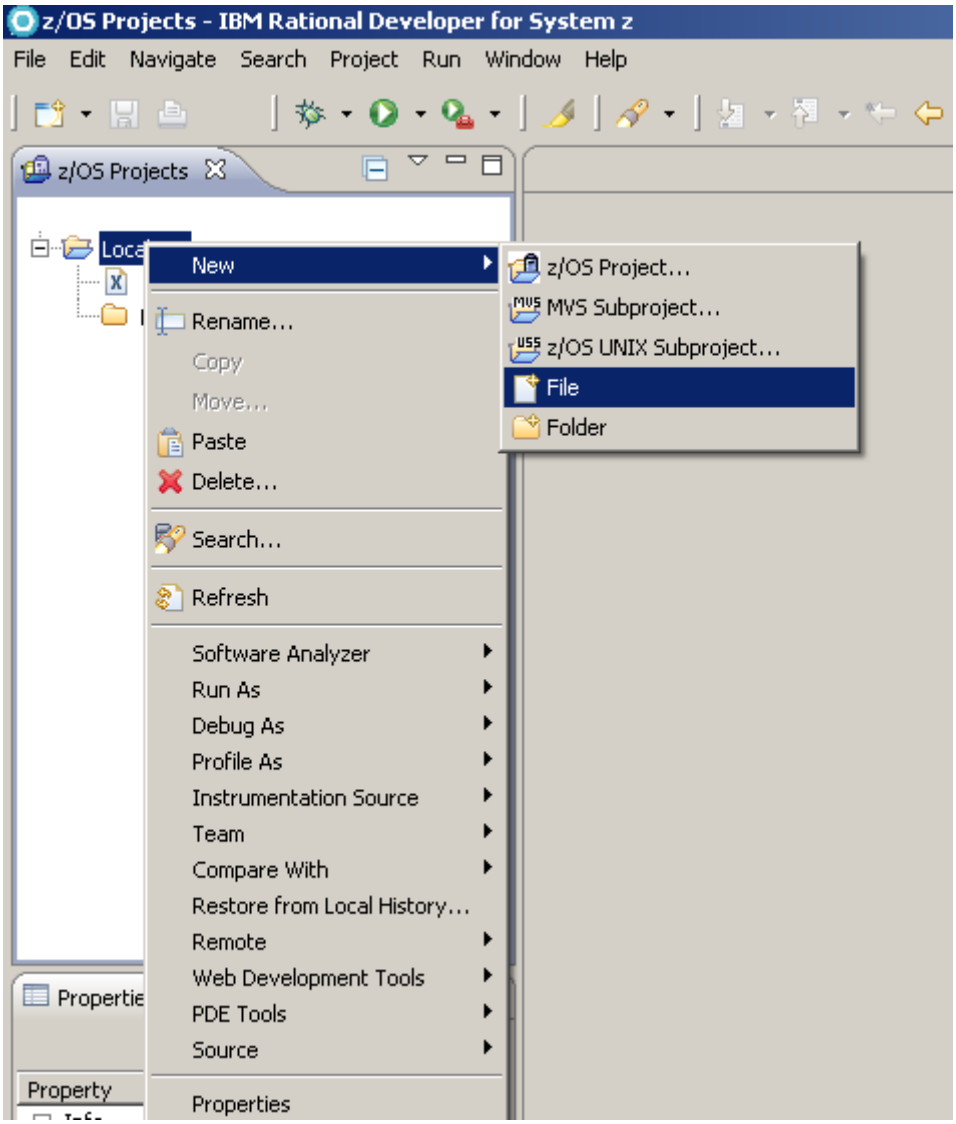
6. At least click Finish



7. Choose “LocalPLI” as Property group and Click Finish.



8. Now go to the z/OS Projects view and Right Click on LocalPLI. Choose **New – File.**



9. Enter **hello.pli** and Click Finisch.

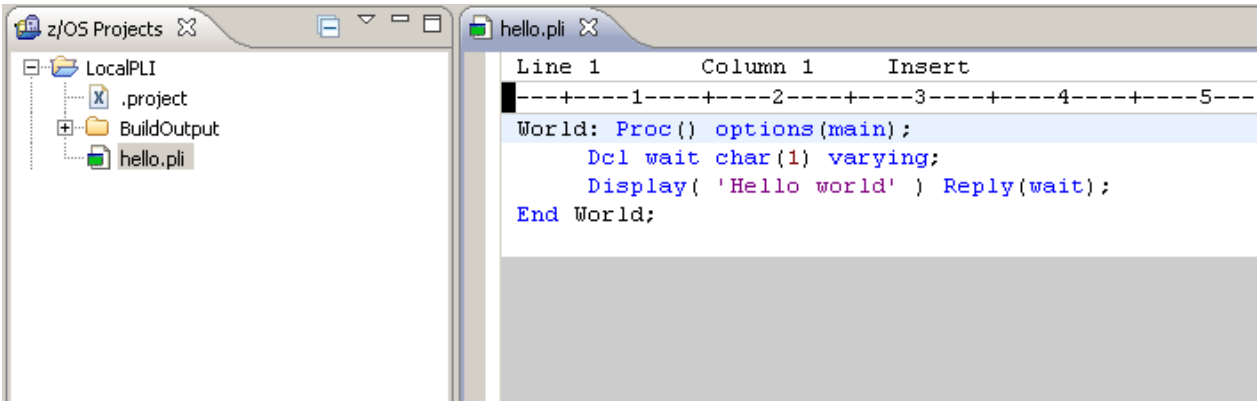
2 Working with a local PL/1 program

In this chapter you learn how to edit source code and how to perform a syntax check with the RDz PL/1 editor.

2.1 Editing a PL/1 source file

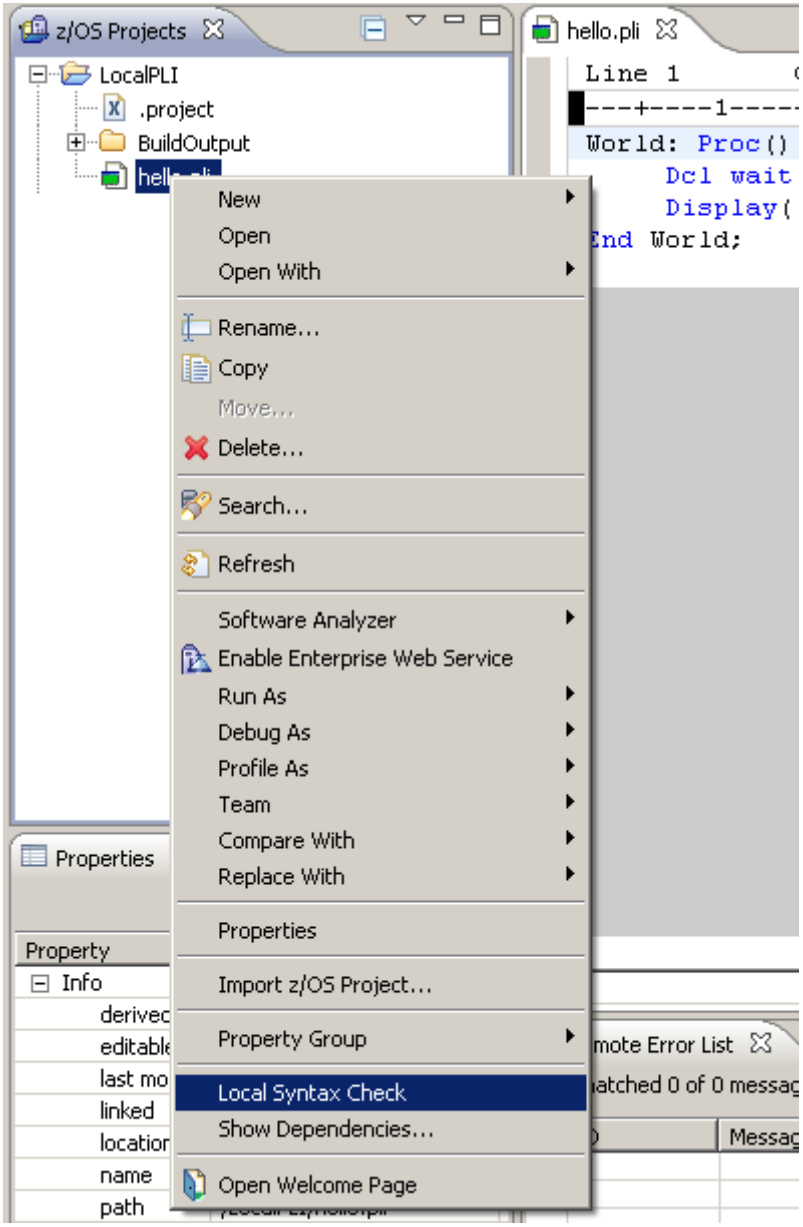
Make sure the z/OS Projects perspective is opened (Window → open Perspective → z/OS Projects).

1. Expand the project **LocalPLI**. Double click on hello.pli. Now enter the PL/1 Code like it is given in this picture. After that press Ctrl + S to save the file.

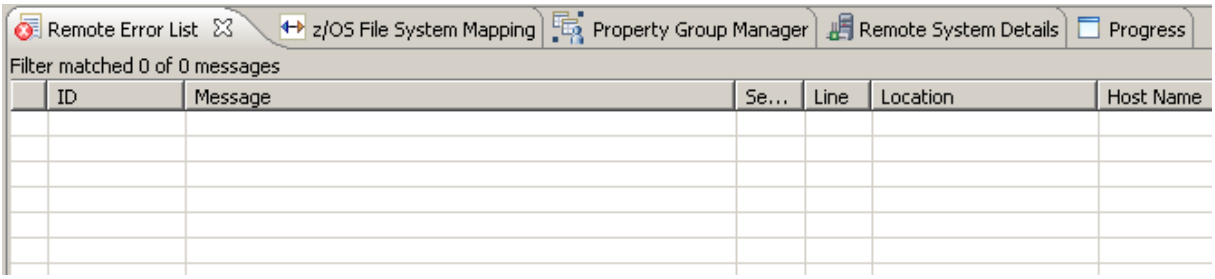


2.2 Checking the PL/1 source file syntax

1. Using the **z/OS Projects** view, highlight **hello.pli**, 1kr right (click mouse button 2), and select **Local Syntax Check** from the pop-up menu:



2. Look at the bottom. The tab **Remote Error List** will show the compile errors. There shouldn´t be any errors.



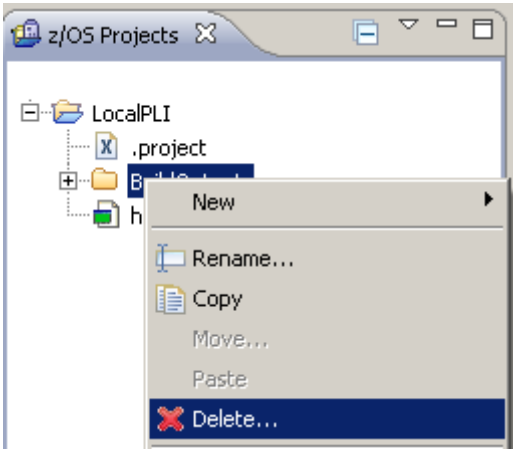
3 Compiling, linking, executing, and debugging the local PL/1 program

After you finished preparing your source code you will now execute your program. It will run in the local PL/1 runtime environment that comes with RDz.

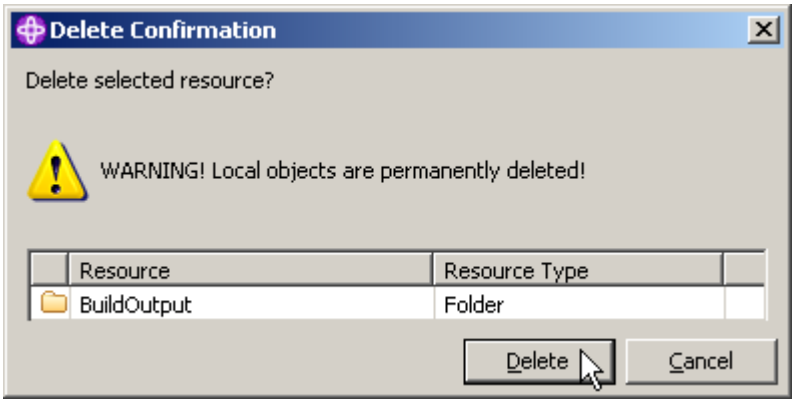
3.1 Creating an executable PL/1 program

Now that you have successfully checked the syntax of your PL/1 program, you can execute it locally, but before creating the executable, let's delete the existing executable code.

1. Select the folder **BuildOutput, 1kr** and use the context menu to delete it.

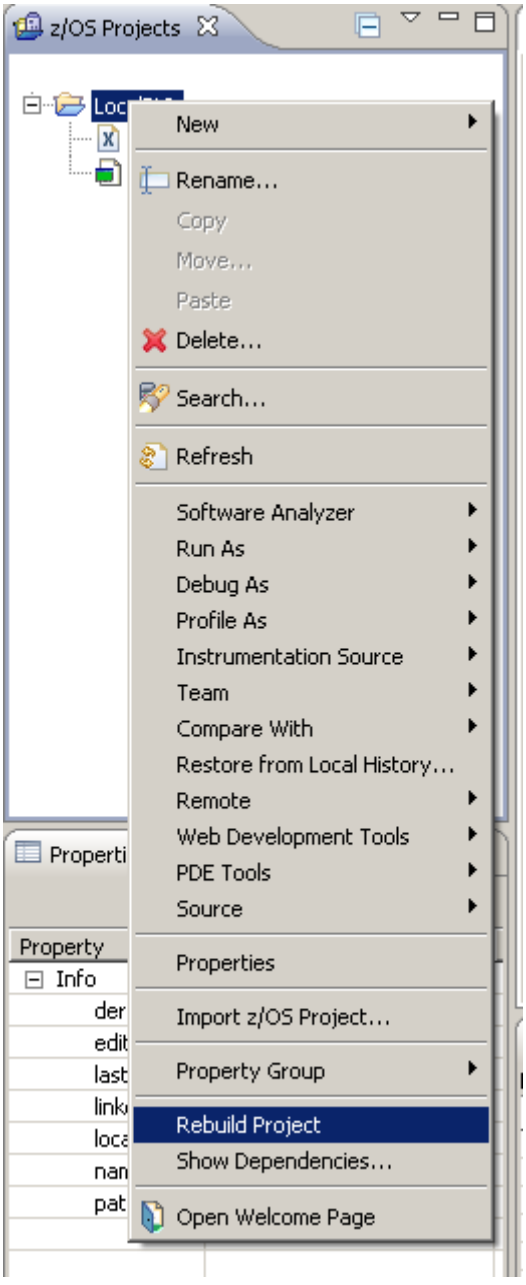


2. 1k Click on **Delete** when the **Delete Confirmation** window pops-up.

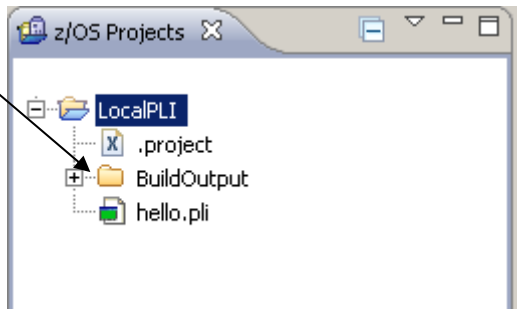


Note: The reason that we delete the **BuildOutput** folder is just to show that it will be recreated. This step is not really necessary.

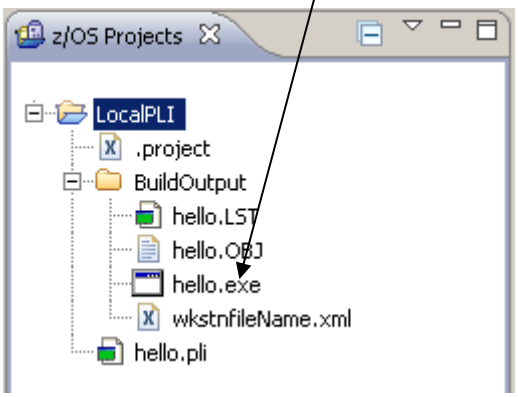
3. To build the program:
   Using the z/OS Projects view, Right-click on LocalPLI and select Rebuild Project.



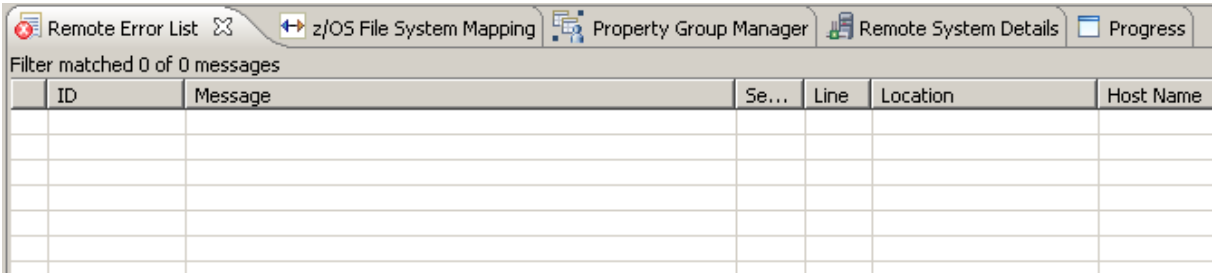
Note that **BuildOutput** is created again.



4. Expand **BuildOutput** to show the folders content. An exe-file is created. Optionally, a DLL could be created instead (needed when using CICS TS).



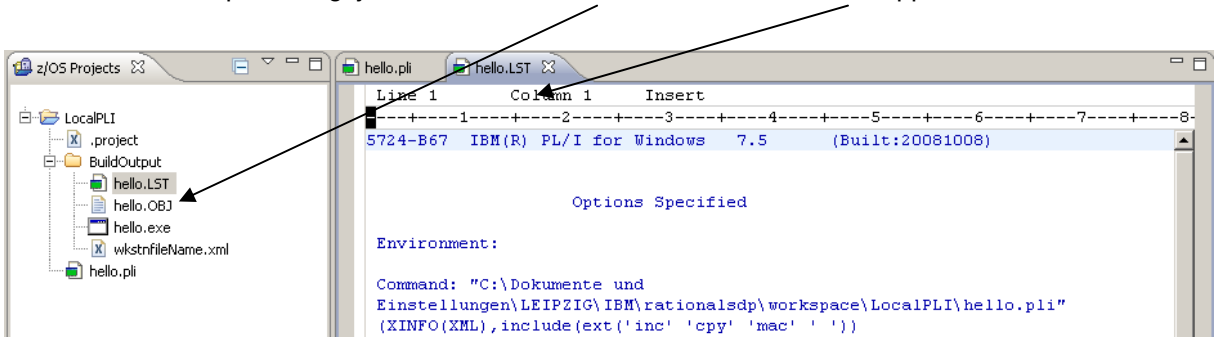
Also, no errors should be listed in the **Remote Error List** view.



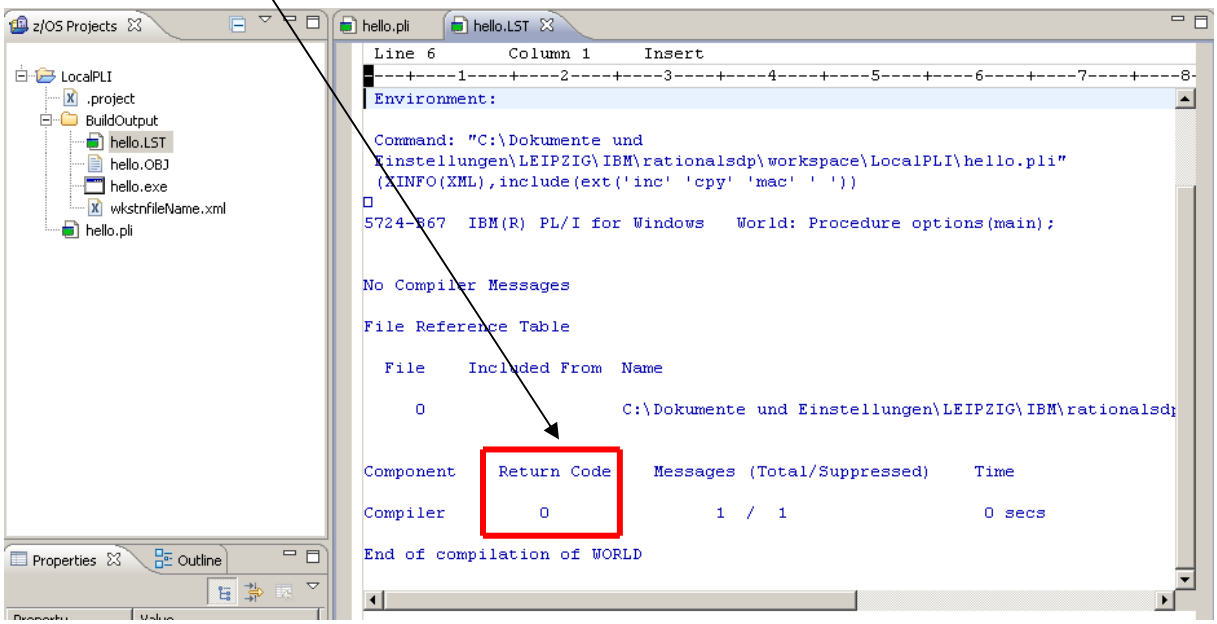
3.2 Check the local PL/1 compilation and link edit

The COBOL compiler stores the output at the **BuildOutput** folder.

1. To see the compile listing, just double-click on **hello.LST**. Another Tab appears.



2. Scroll down to the end of the listing and you see that there are no compilation errors (Return Code 0).

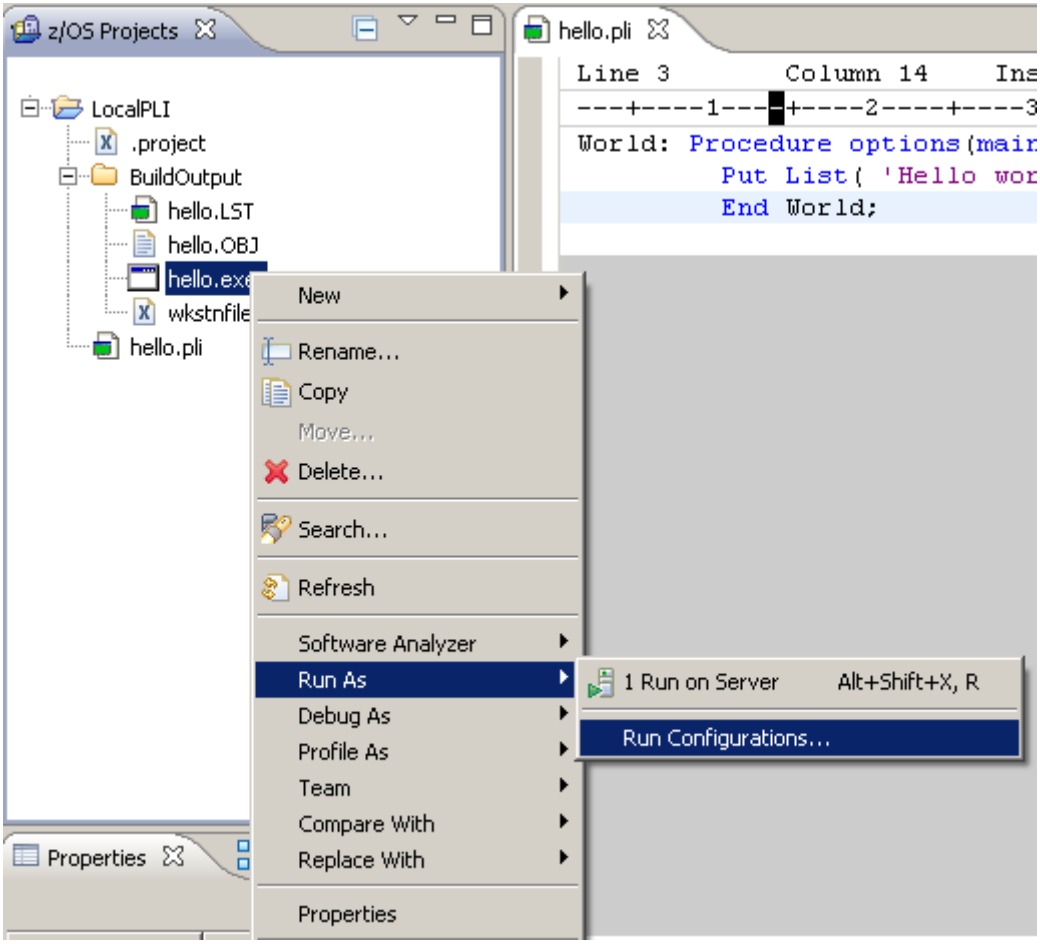


3. **Close** the editor

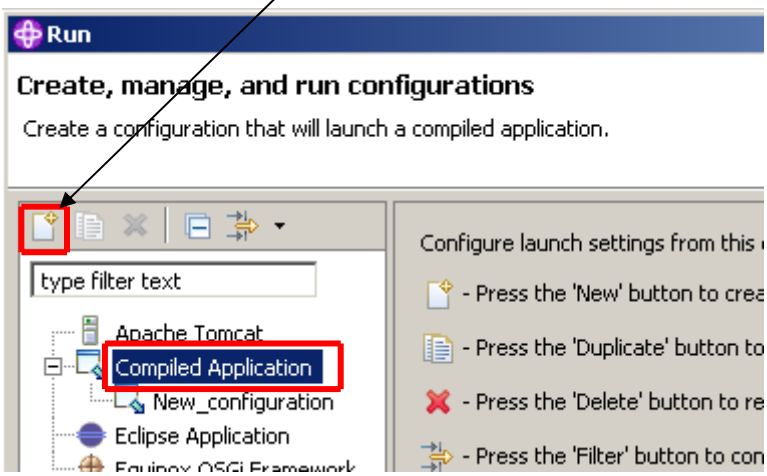
3.3 Creating a Run launch configuration and executing the PL/1 program

Since we might want to run this program many times, it's a good idea to create a debug launch configuration. This makes running and debugging a specific program easier. To create a launch configuration that will load the compiled PL/1 program do the following:

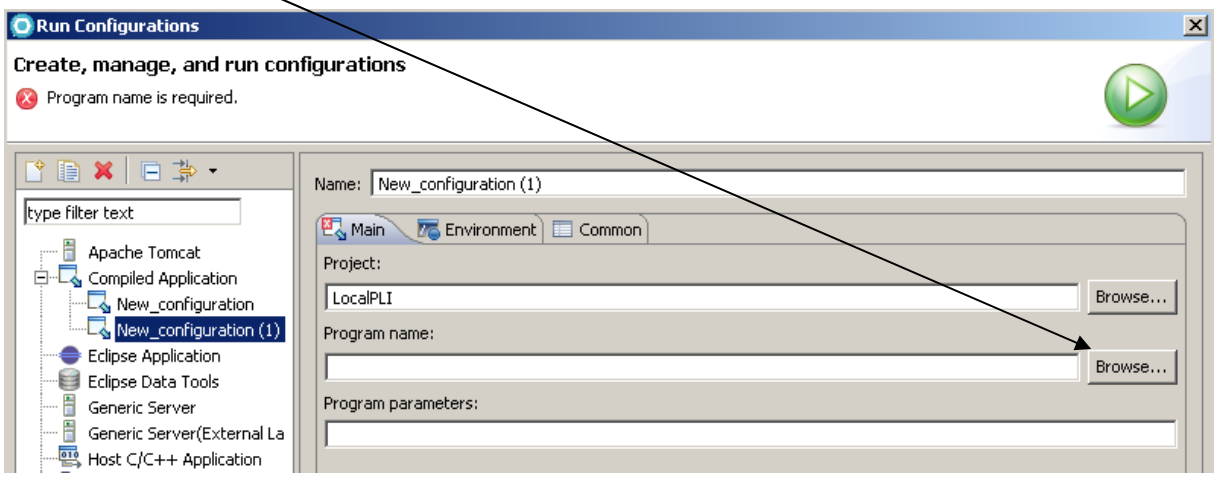
1. Just right click on **hello.exe** and select **Run as → Run Configuration…** (**not** Run on Server)



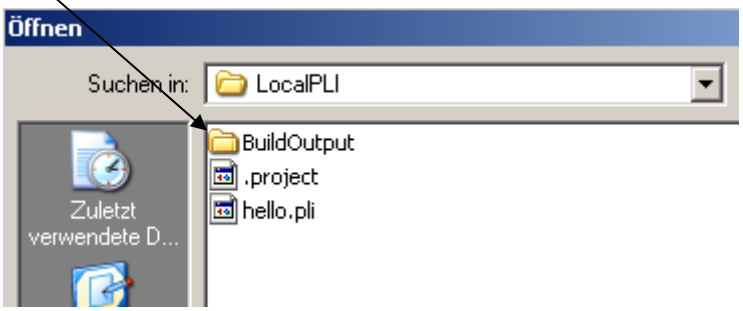
2. Select Compiled Application and click on the **New Button**. This will cause the launch configuration tabs and entry fields to display on the right-hand side of the dialog box.



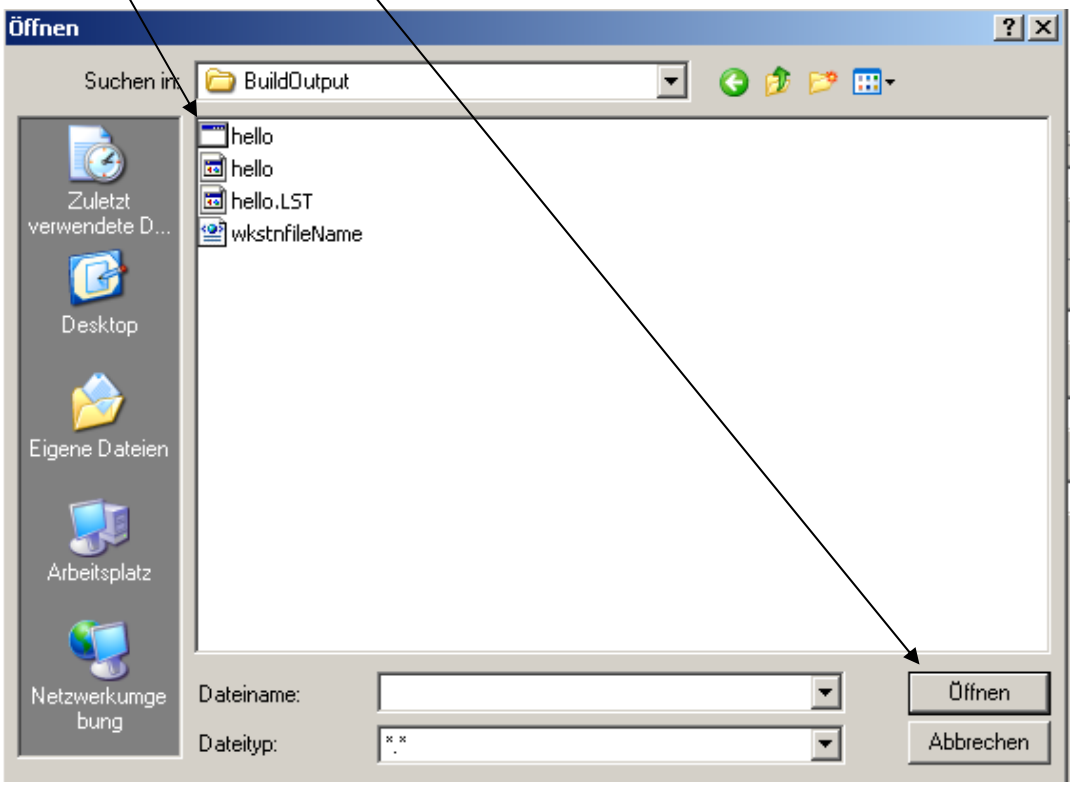
3. Use the Browse button next to **Program name** to locate hello.exe under the BuildOutput folder.



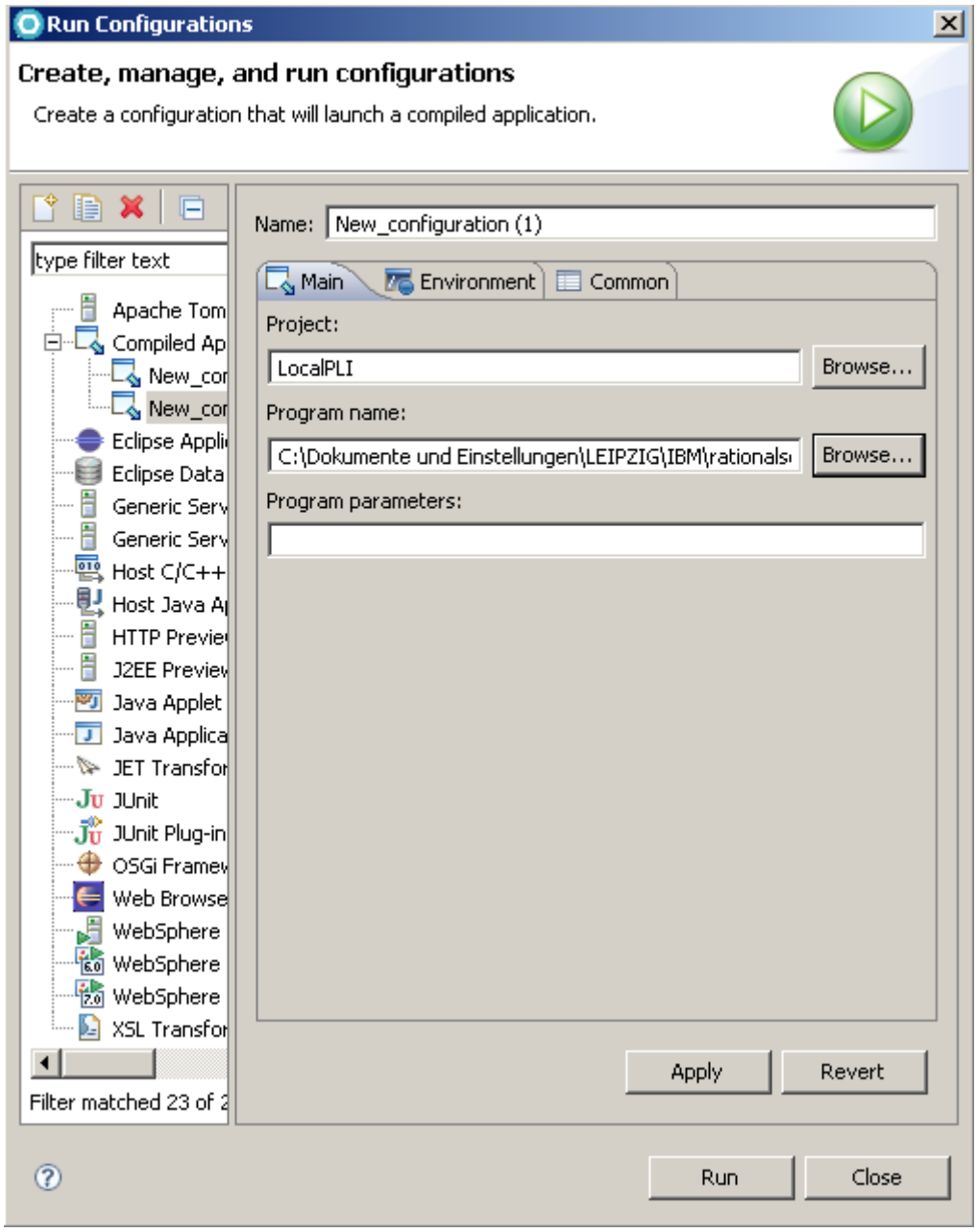
4. Click on the BuildOutput folder to open it.



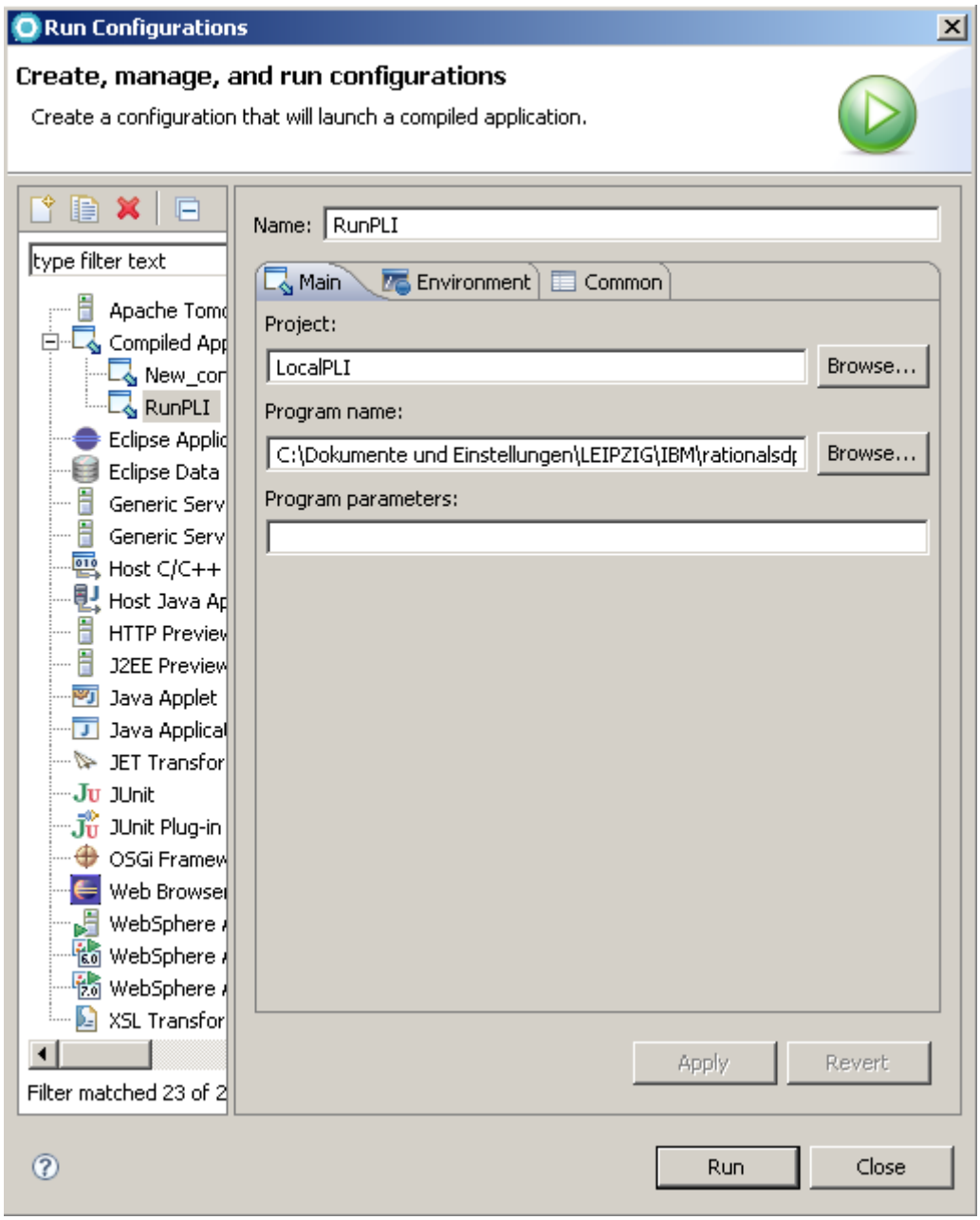
5. Select hello.exe and click on open to select it.



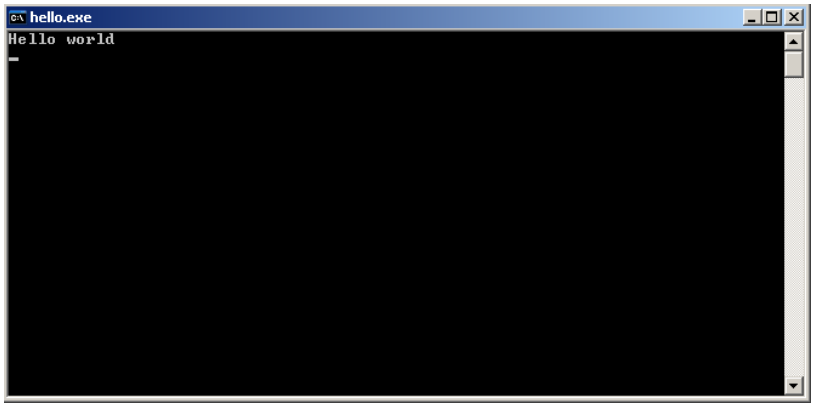
6. In the **Name** field, enter a name like **RunPLI.** Click Apply to save your changes.



7. Hit **Run**.



8. You should see the following screen:

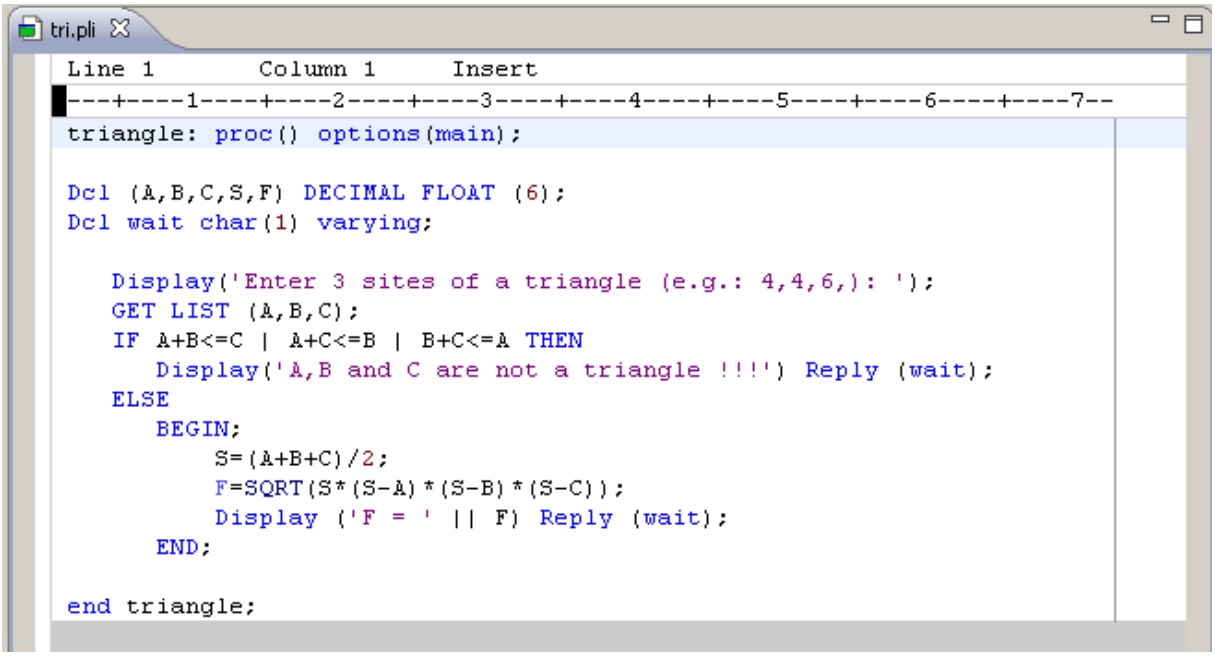


4 Write another local PL/1 program

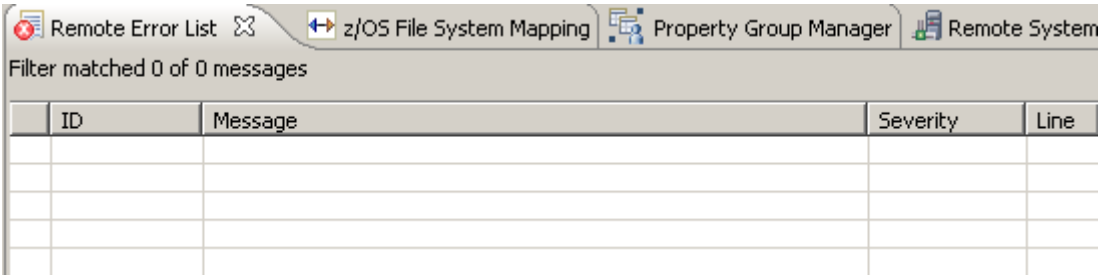
Now we want to create a PL/1 Program with input and output. It's a simple program to calculate the area of a triangle.

4.1 Creating the program

1. The steps are the same like in chapter 1 to chapter 3. Practice these steps and create a new local project named "LocalPLI2" and a file called "tri.pli". Use as property group "LocalPLI".
   After that enter the code from the following picture.



2. Now make "Save and Syntax check". You should not see any errors in the Remote error list.



Note: If the is any error, check your code again

4.2 Explore the code

Now we want to have a look at the code of our new program.

These two lines are the body.

```
triangle: proc() options(main);
end triangle;
```

First we create the variables. A, B, C, S and F are decimals. The variable “wait” we need to have a break at the end of the program.

```
Dcl (A,B,C,S,F) DECIMAL FLOAT (6);
Dcl wait char(1) varying;
```

Now we make an output with the “Display” command to say the user that we want to have the input for our triangle. The “Get List” command reads the input and save it in our variables A, B and C.

Note: The user have to make a comma at the end of the line to say the “Get List” command that this is the last input!

```
Display('Enter 3 sites of a triangle (e.g.: 4,4,6,): ');
GET LIST (A,B,C);
```

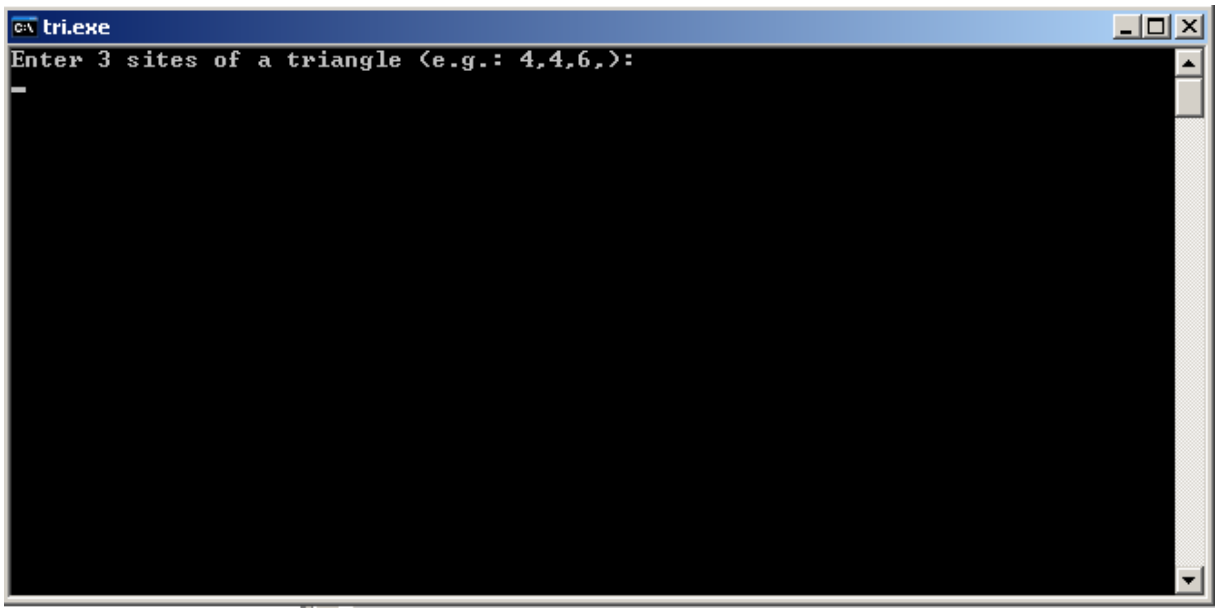
In these lines we first check whether our input is a triangle or not. If not we display this on the screen. If it’s a triangle we calculate the area of it and display the result on the screen. The “BEGIN – END” construct summarize the following 3 lines.

Note: The “Reply” command waits for an input from the user. So we have a break after the output to see the result before the console is closing.

```
IF A+B<=C | A+C<=B | B+C<=A THEN
   Display('A,B and C are not a triangle !!!') Reply (wait);
ELSE
   BEGIN;
      S=(A+B+C)/2;
      F=SQRT(S*(S-A)*(S-B)*(S-C));
      Display ('F = ' || F) Reply (wait);
   END;
```

4.3. Execute the program

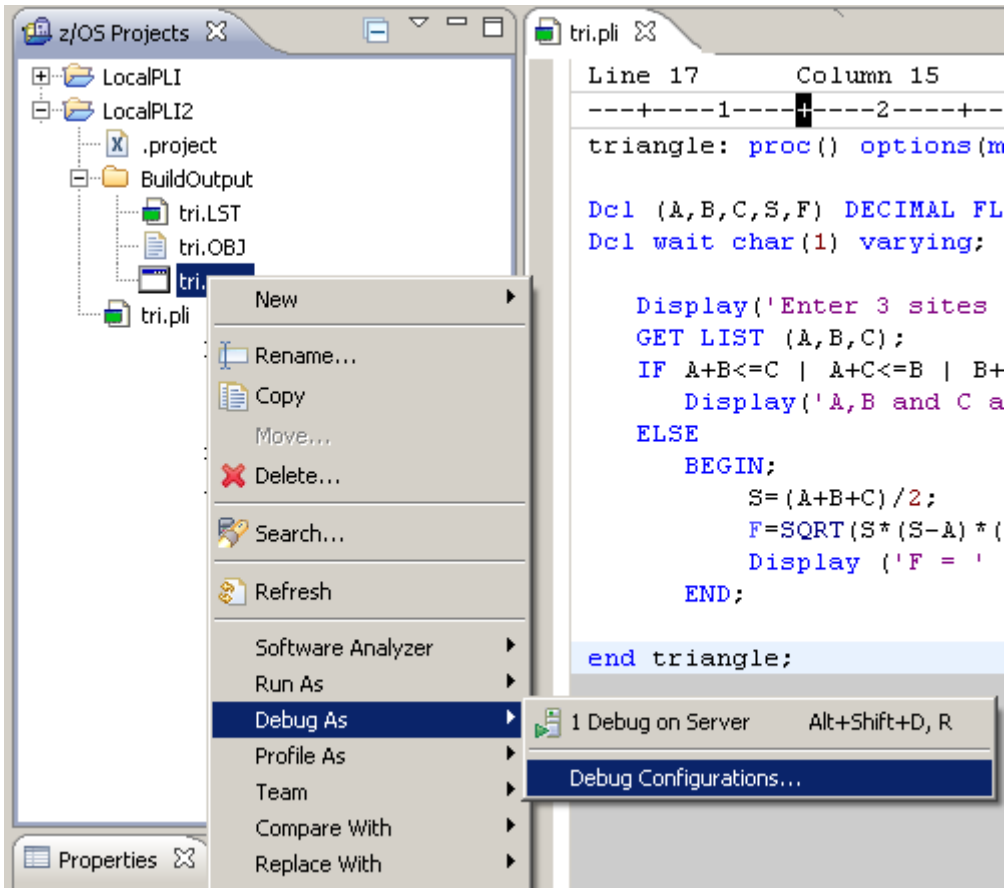
1. Now rebuild and run the project like it's explain in chapter 3. Use as name for the Run Configuration "Runtri".
   Then you should see this:



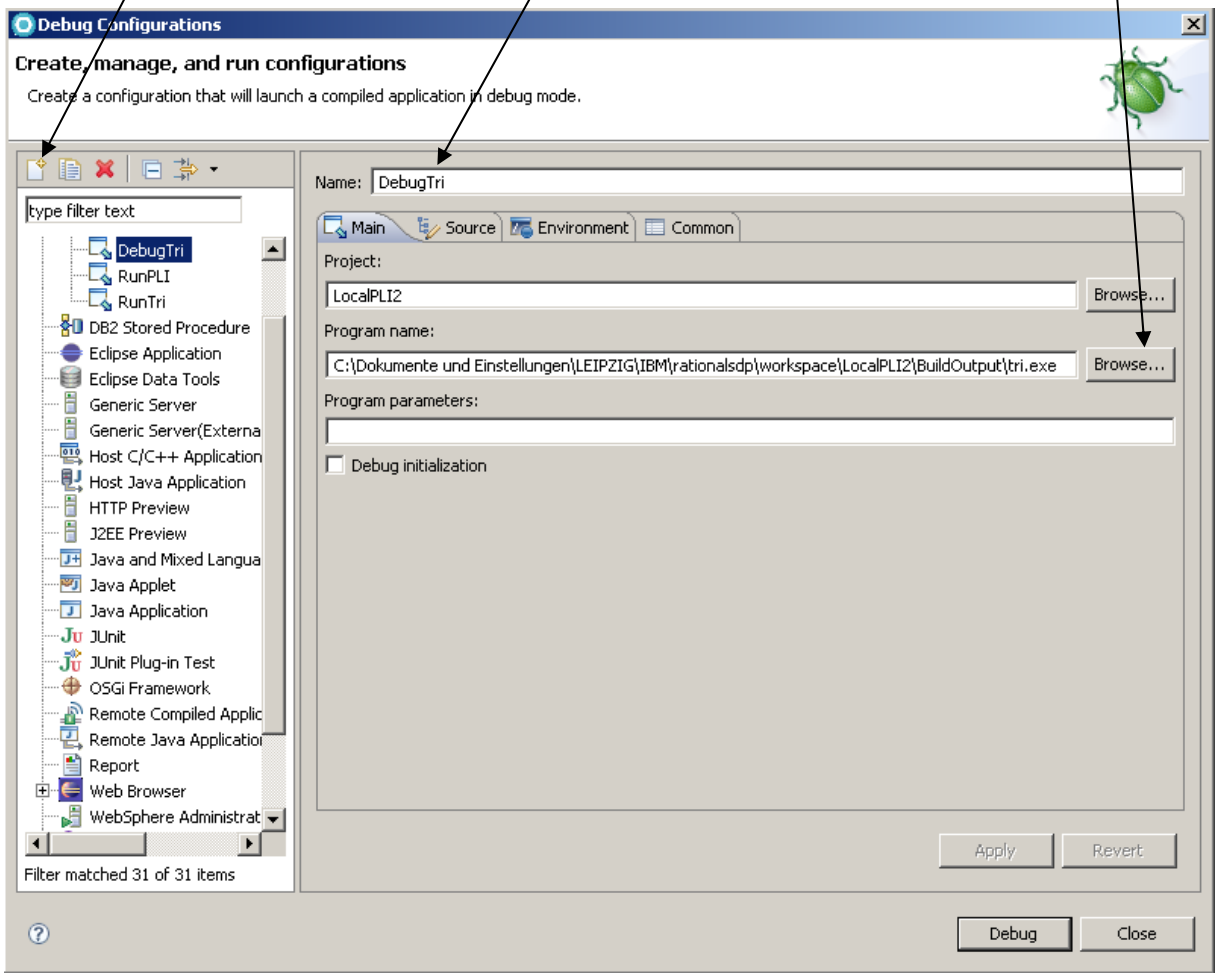
2. Follow the instructions of the program to calculate the area of a triangle.

5 Testing/Debugging the PL/1 Program

1. Since we have already created the launch configuration, just right click on **tri.exe** and select Debug as and then Debug Configurations.



2. When the Debug window opens, create a new Debug Configuration by clicking **“New launch configuration”.** Name it **“DebugTri”** and choose **“tri.exe”** by clicking **“Browse”.**



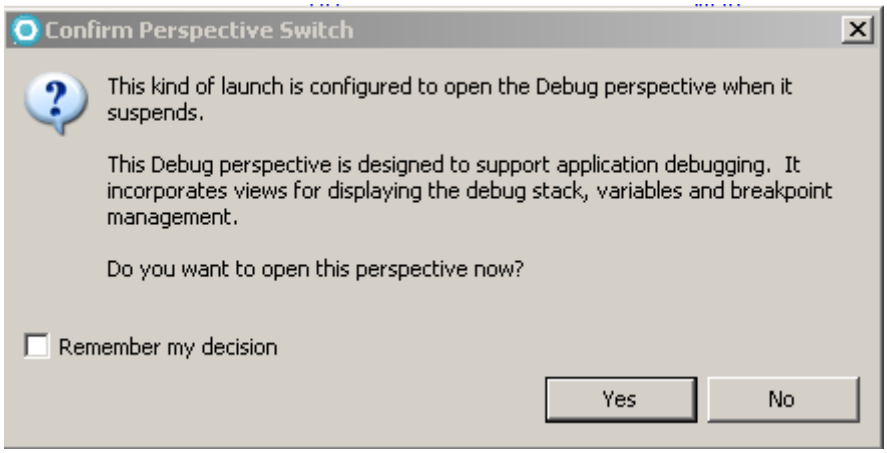
Note: After clicking “Browse” you have to go into the “BuildOutput” folder.

3. After that apply the changes and click “**Debug**”

4. The dialog below will ask if you want to switch to the **Debug perspective**. Remember, right now we are working with the z/OS Projects perspective. Switching perspectives is a normal procedure when using RDz. You can always switch back to the z/OS Projects perspective by clicking on **Window**.

   Click **Yes.** This switches from the z/OS Projects perspective to the Debug perspective.

   Note that the above dialog asking to switch to the debugger perspective could be hidden by the console/dos window. You might need to minimize the console window to be able to see the dialog.

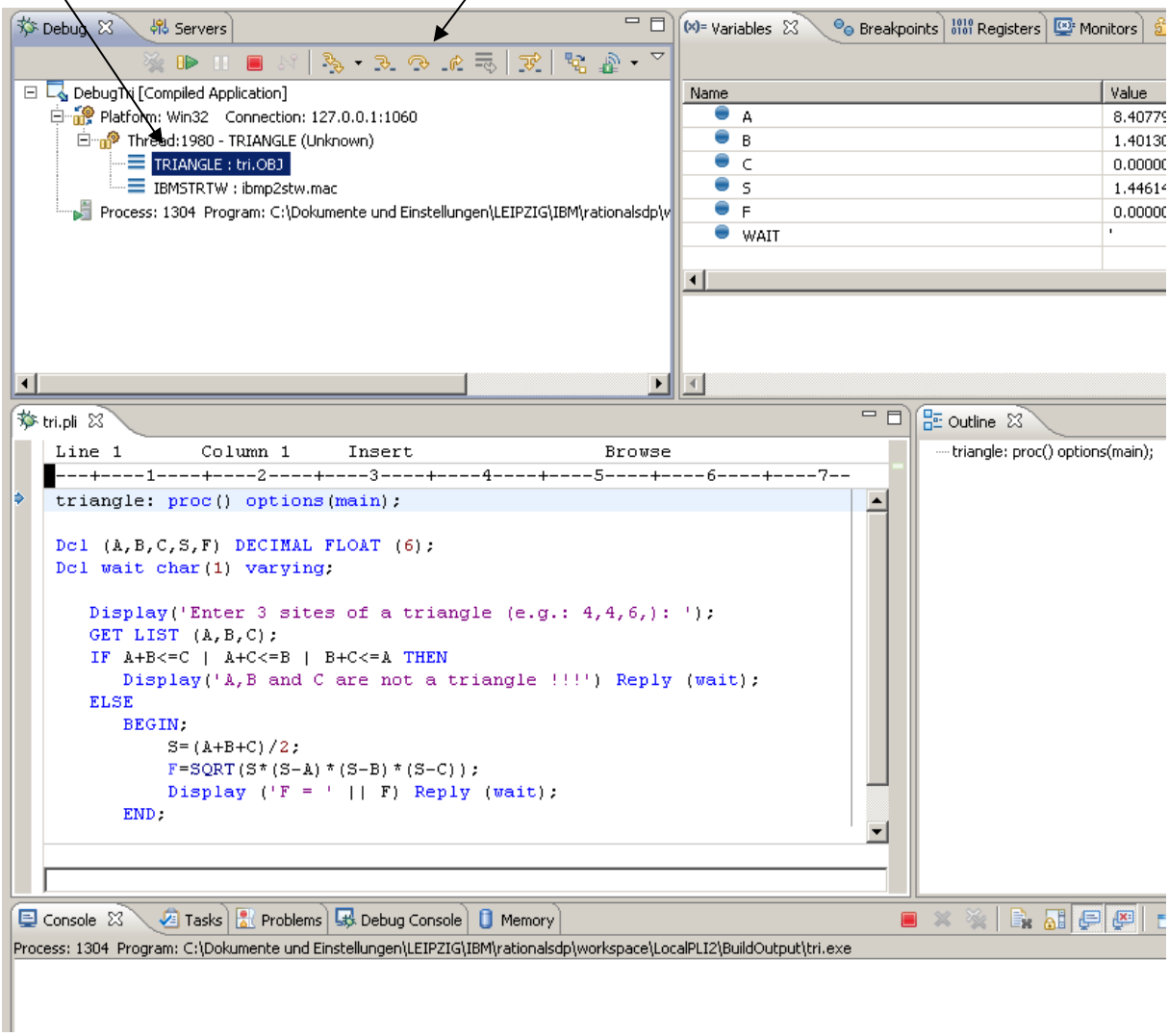


   The **Debugger** perspective is opened and the debug starts.

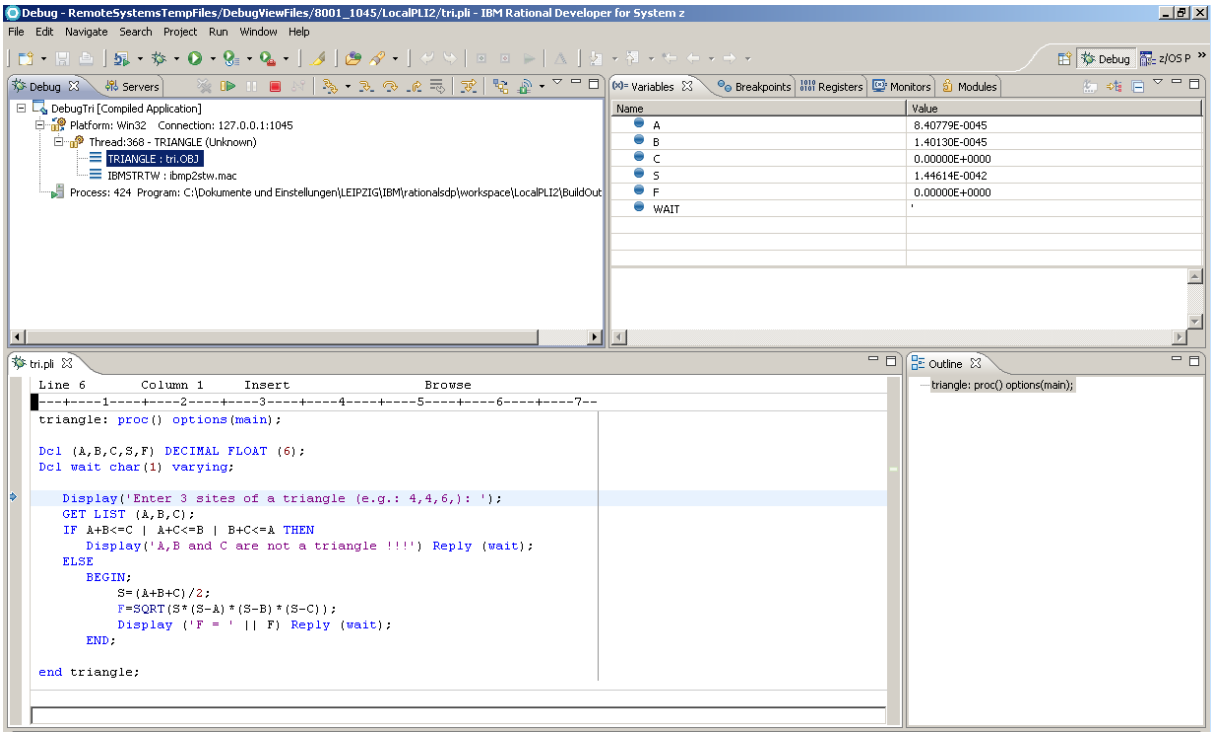
5. This entry should be highlighted.

In the Debug view, click in the **Step Over** icon ( ) (or alternatively hit **F6 twice).** .
(Do not use *Step Into)*

If you missed something, close the Debug perspective and the file being edited start from step 5. **Testing/Debugging the PL/1 Program** again.



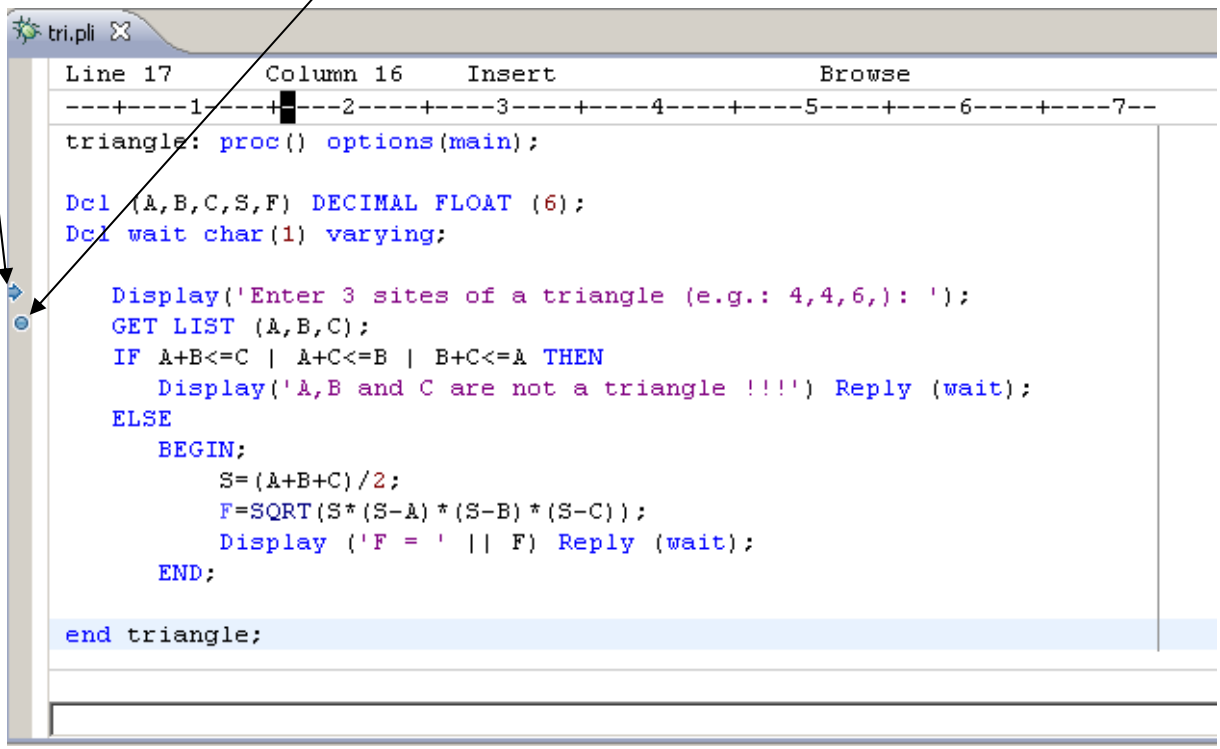
6. The PL/1 execution starts and you will see something like the screen.



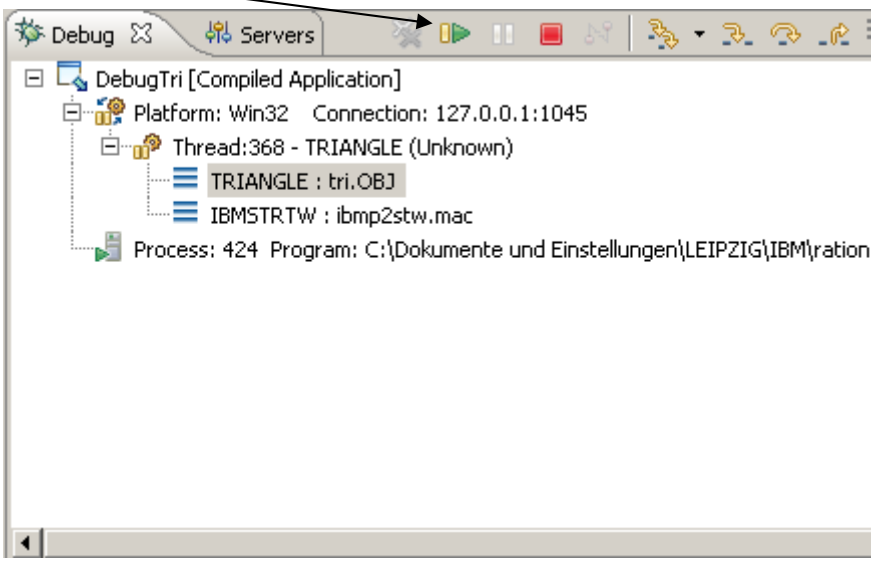
7. This small arrow indicates the program execution stopped after executing this line.

   Add a breakpoint in the **GET LIST** statement. Just move the mouse in the grey area before the line and double click. A small circle indicates a break at this line.

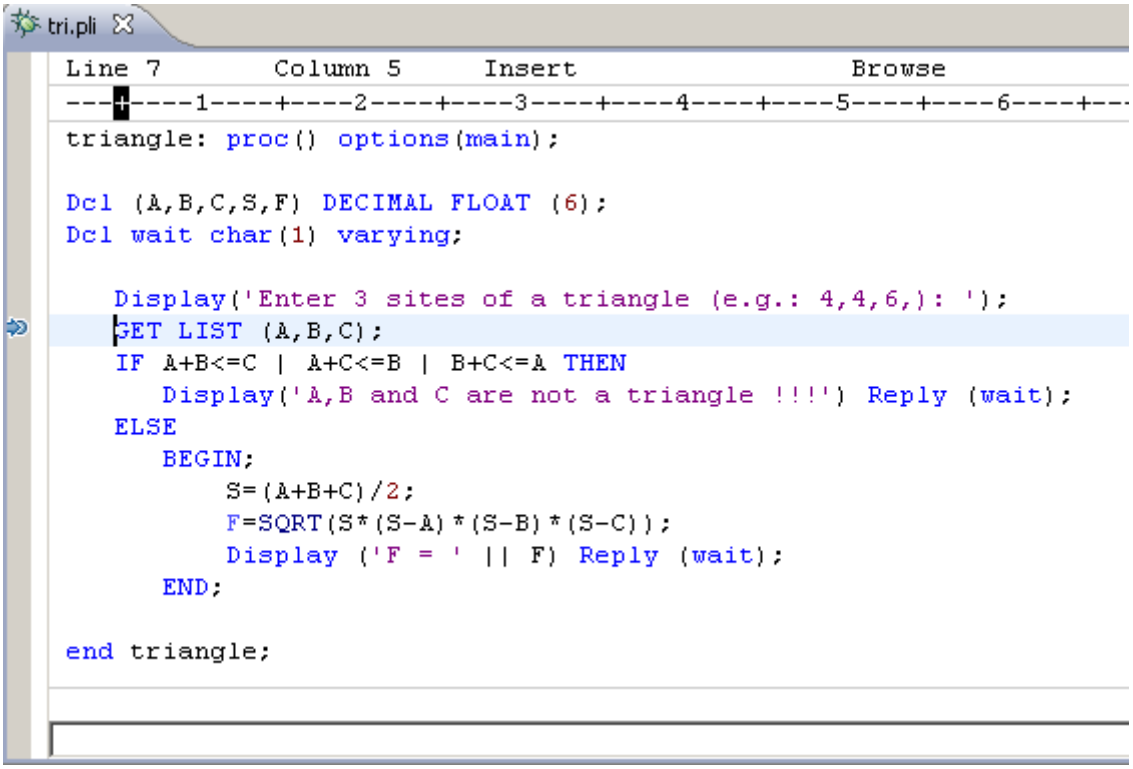
   A breakpoint causes the execution of a thread to suspend at the location where the breakpoint is set.



8. Now, to resume the execution of the program until the breakpoint is found, click on the **Resume** icon ( ) or press **F8**.

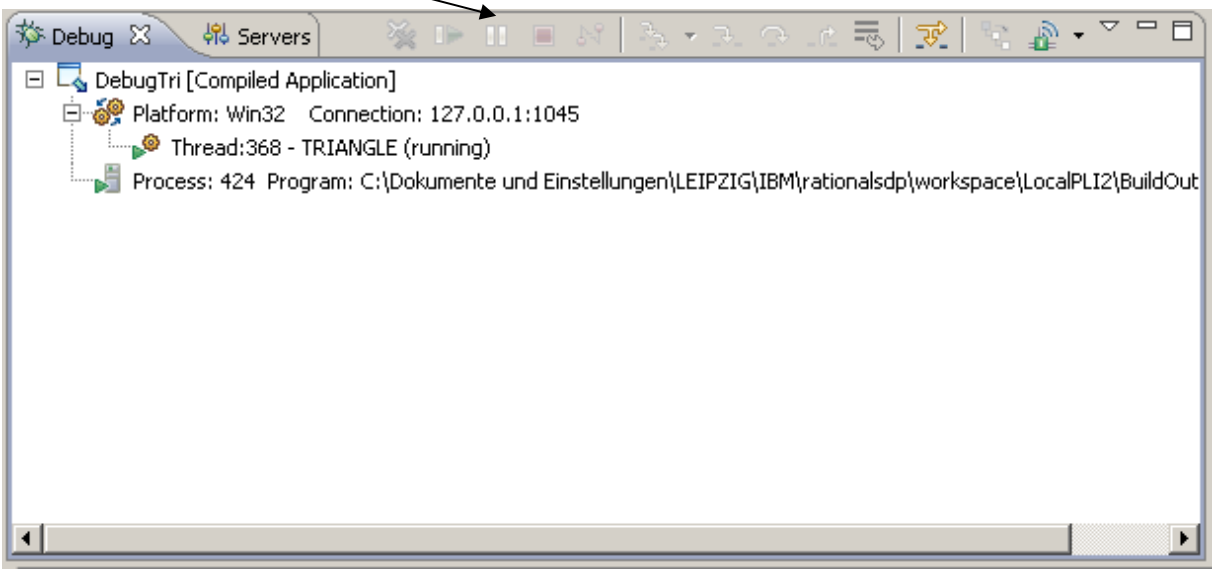


9. The execution will stop at the GET LIST statement at Line 7.



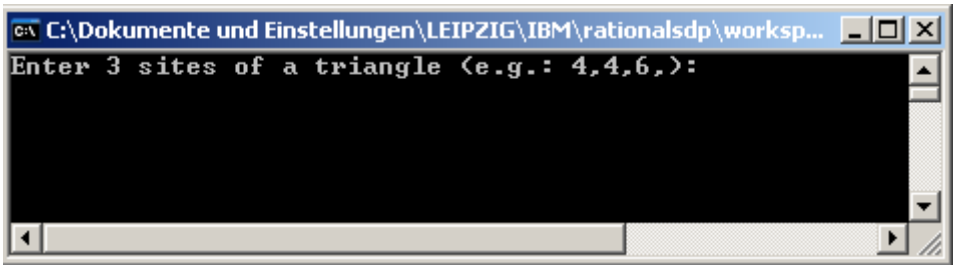
10. Execute one statement using the **Step Over** icon or F6.

    The commands are now disabled since you need to go to the console window to type an answer. The icons are now grey. We are waiting for input in the console window.



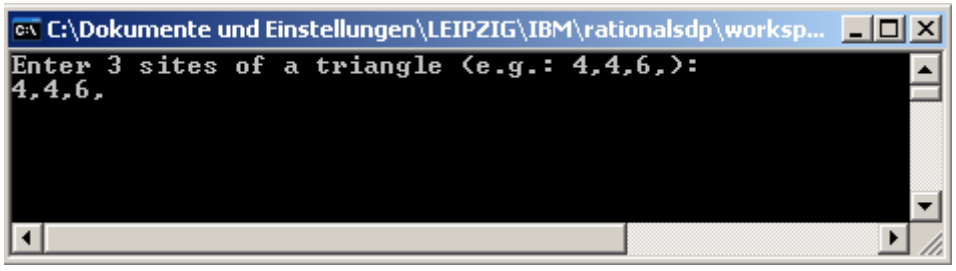
11. A console window opens. It may be minimized, or hidden under the RDz window. Click on the tab to open it.

12. Restore the console window.



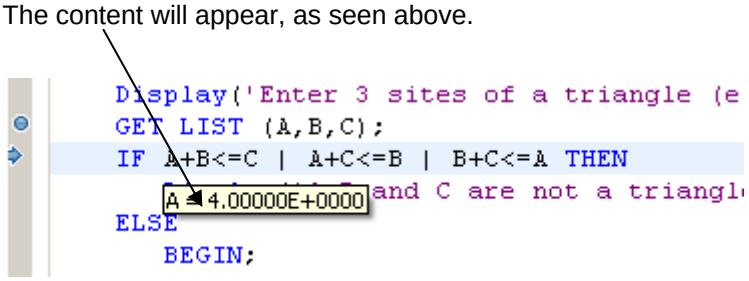
13. Enter the tree sites of a triangle and hit enter.

    Note: The Input has to end with a comma, because we are using the GET LIST command.

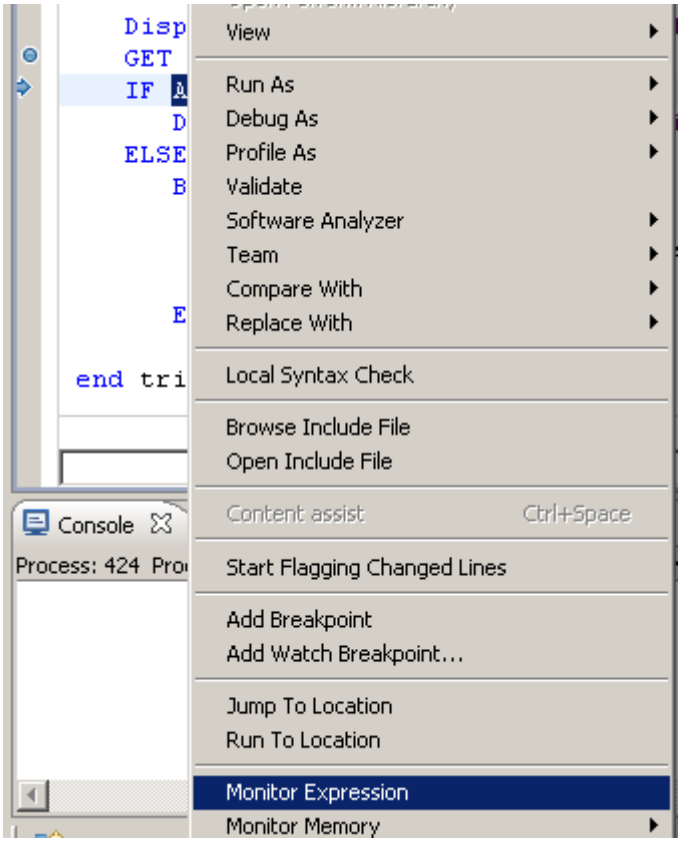


14. Minimize the console window to continue with the debug (do not close it).

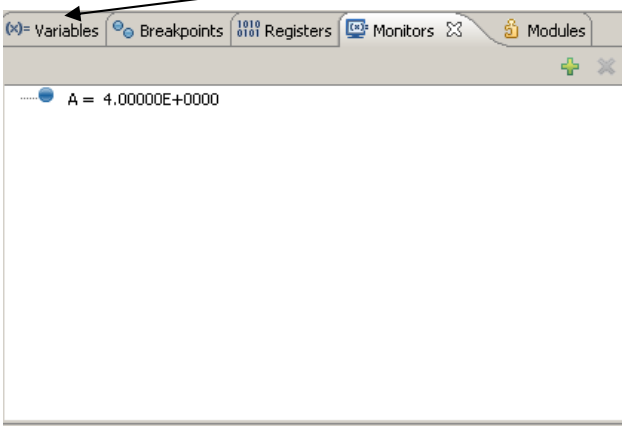
15. To evaluate variable contents, like the variable **A**, just move the mouse to the data-name and wait 5 – 10 seconds. The content will appear, as seen above.



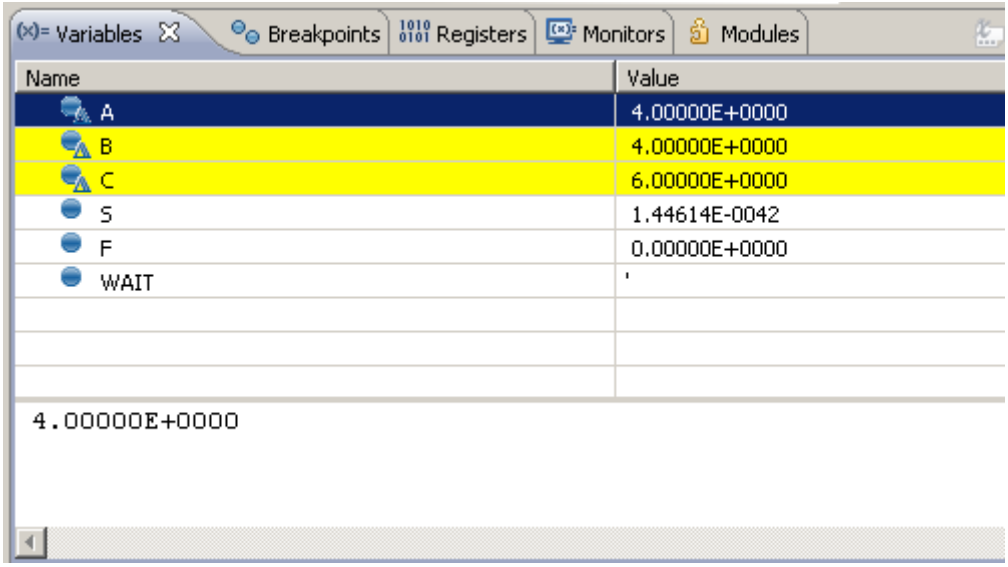
16. Also you could highlight (select) the field “A” and right click on it. On the context menu click **Monitor Expression.**



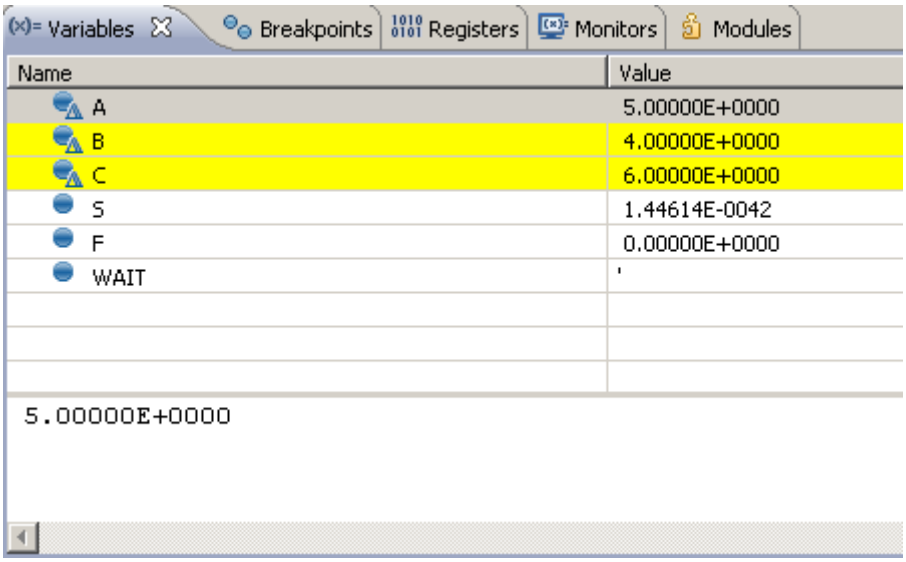
17. In the right upper corner the value of the variable is displayed. Now click on the variables tab to change the window back.



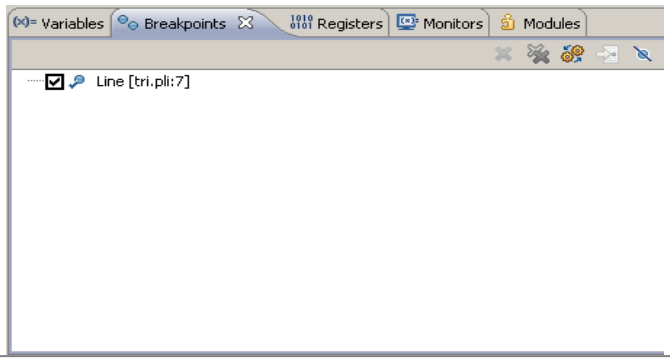
18. It shows the values of the variables.



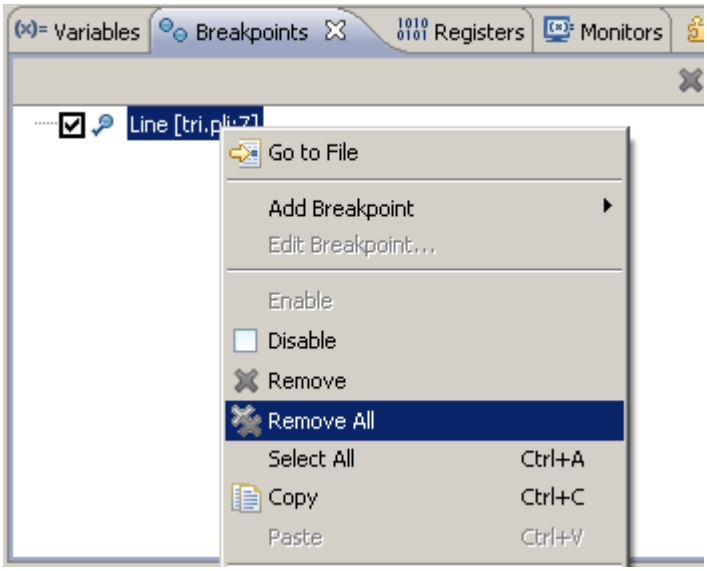
19. Now click on the variable A and overtype the value with 5.00000E+0000. Now if you move the cursor again on the variable A in line 8, it will show the modified value.



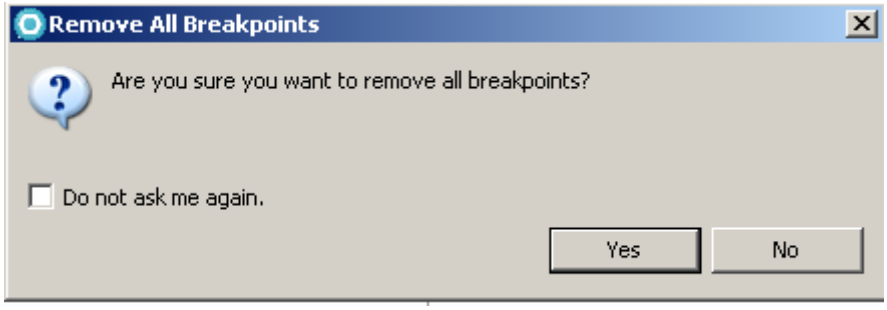
20. In the right upper corner, click on the **Breakpoints** tab to see all breakpoints assigned to this program.



22. Remove the breakpoint (you could disable the breakpoint instead of removing it, allowing you to re-enable it later). Go to the **Breakpoints** view, right-click within the window, and select **Remove All** from the context menu (Right Mouse Click).

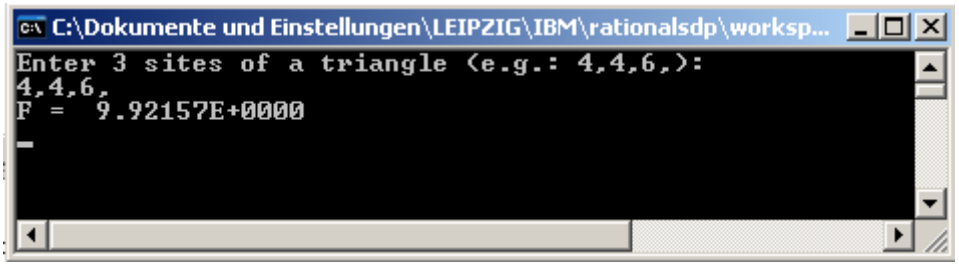


23. This clears the Breakpoints window.

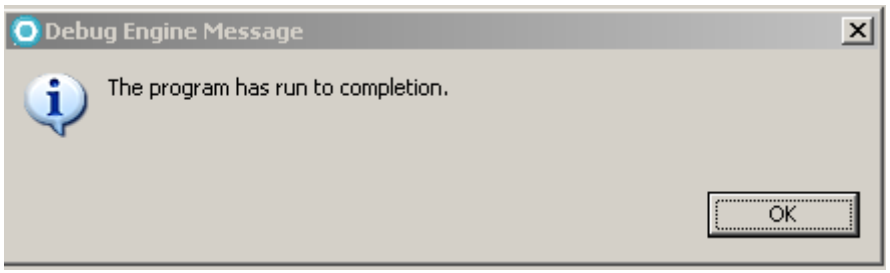


24. If you want to Resume use the Resume icon ().

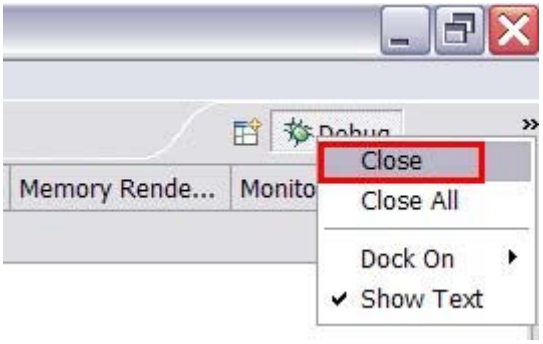
You will see the result of the Program in the console. Have a look and hit enter.



25. When the program ends a message is shown:



26. Click OK and Close the Debug perspective (Upper Right).



Congratulations, you successfully completed this Tutorial. You have learned the basics of your RDz workstation PL/1 facilities. So far, this may be interesting, but does not help to write PL/1 mainframe programs. We will address this in the tutorial “Remote PL/1”. Patience!

Anlage 03

RDz Tutorial 03

Remote PL/1

RDz Tutorial 03

Remote PL/1

(Development of a remote application using RDz 7.5)

Overview

This tutorial will show you how to develop PL/1 applications running on a System z Mainframe system. You will define a remote z/OS connection, set up a MVS project, remote edit and compile.

The tutorial consists of following main tasks:

1. Initial preparation for RDz remote development
   1.1 Define and connect to a remote system
   1.2 Creating needed z/OS System datasets
2. Working with remote files
   2.1 Import a property group
   2.2 Create a MVS Subproject
   2.3 Add resources to your project and create a member
   2.4 Compile/link/execute the remote PL/1 program

1 Initial preparation for RDz remote development

Before we can start programming some preparations have to be done.

First you will connect to a remote system which has been set up for you. You will learn how to allocate data sets using RDz.

1.1 Define and connect to a remote system

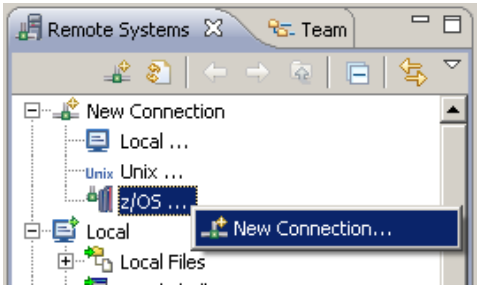
The Remote Systems view shows all existing connections to remote systems. Connections are System Connection objects that are persisted, containing the information needed to access a particular remote host. The view contains a prompt to create new connections, and pop-up menu actions to rename, copy, delete, and reorder existing connections.

Connections contain attributes, or data, that is saved between sessions of the workbench. These attributes are the connection name, the remote system's host name and system type, an optional description, and a user ID that is used by default by each subordinate subsystem, at connection time. Underneath, all connections are stored as files in an Eclipse project named RemoteSystemsConnections, which the user can enable for team support, allowing connections to be shared by a team.

1. If RDz is not already up and running, start it. When prompted for a workspace location you can either continue using a workspace from a previous tutorial, create a new workspace or take the default one below.
2. If you created a new workspace, the Welcome screen will be displayed that you will have to close.
3. Ensure you are working with the z/OS perspective by selecting

   Window → Open Perspective → Other… and select z/OS Projects.
4. Click on the Remote Systems view and expand the **New Connection node**, right-click on z/OS… and select New Connection to open the wizard.

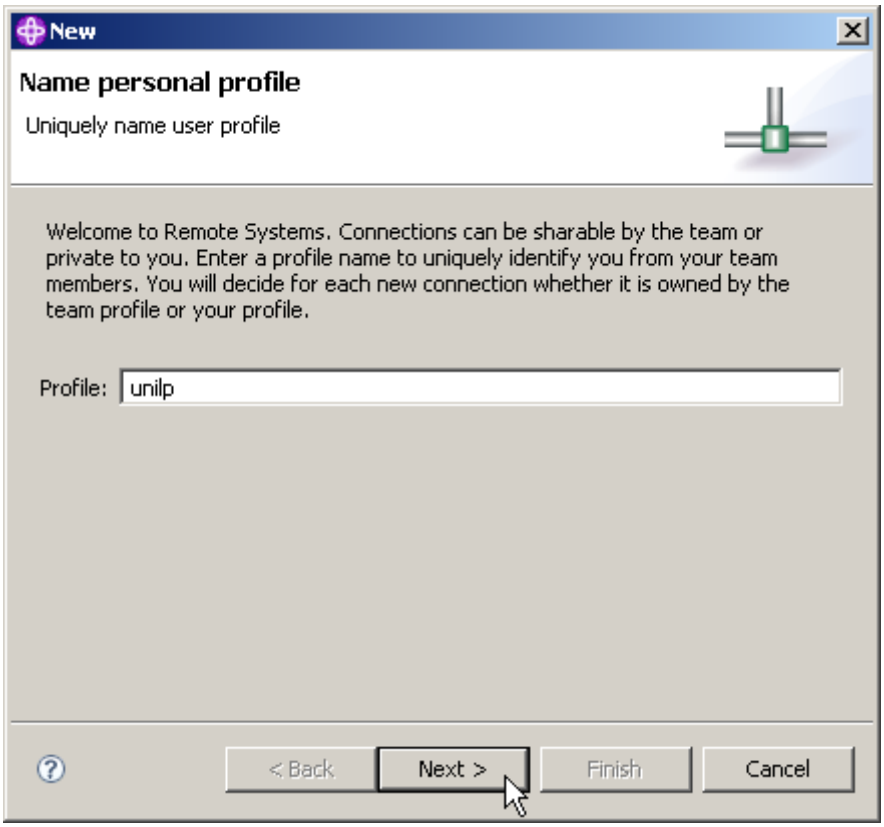
   Note: If you already created a connection to Binks, continue on step 11.



5. If this is the first time that you have attempted to create a connection, you are prompted to create a profile before you can create the new connection. Name your profile and Click Next.

   (Note: If the screen does not appear there are already existing profiles and you can ignore this step)

   In the Profile field, the profile named after your workstation appears by default (Note that after you create the connection, you can share this profile to allow other users to have this connection in their Remote System perspective)



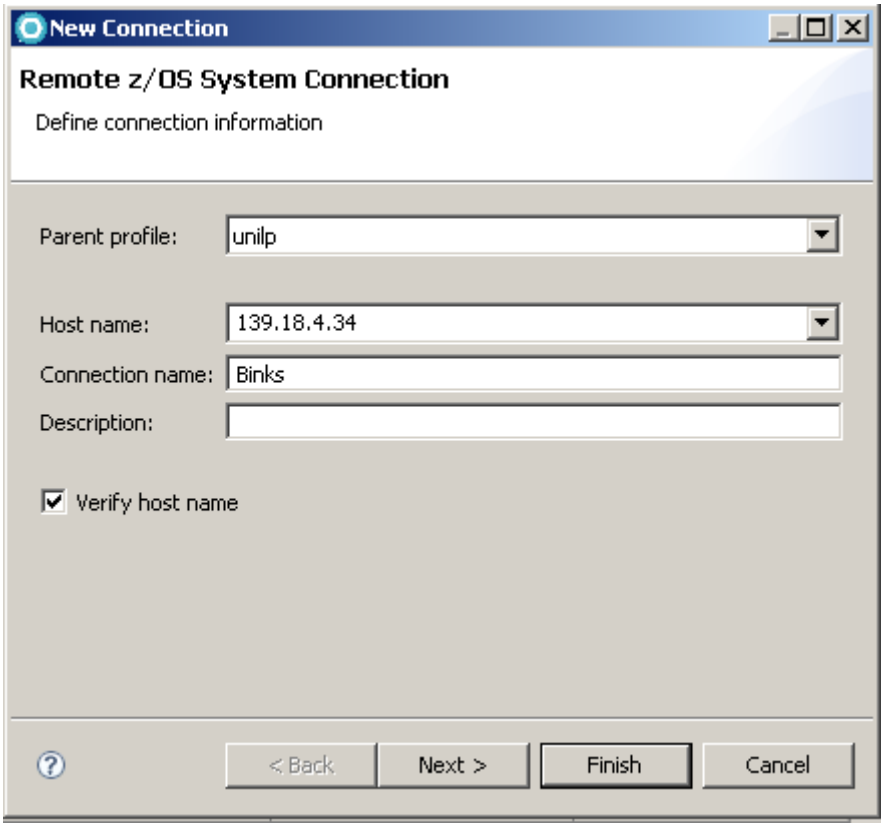
6. In the Connection name field, type Binks as name to identify your connection in the Remote Systems view.

   In the Host name field, type 139.18.4.34.

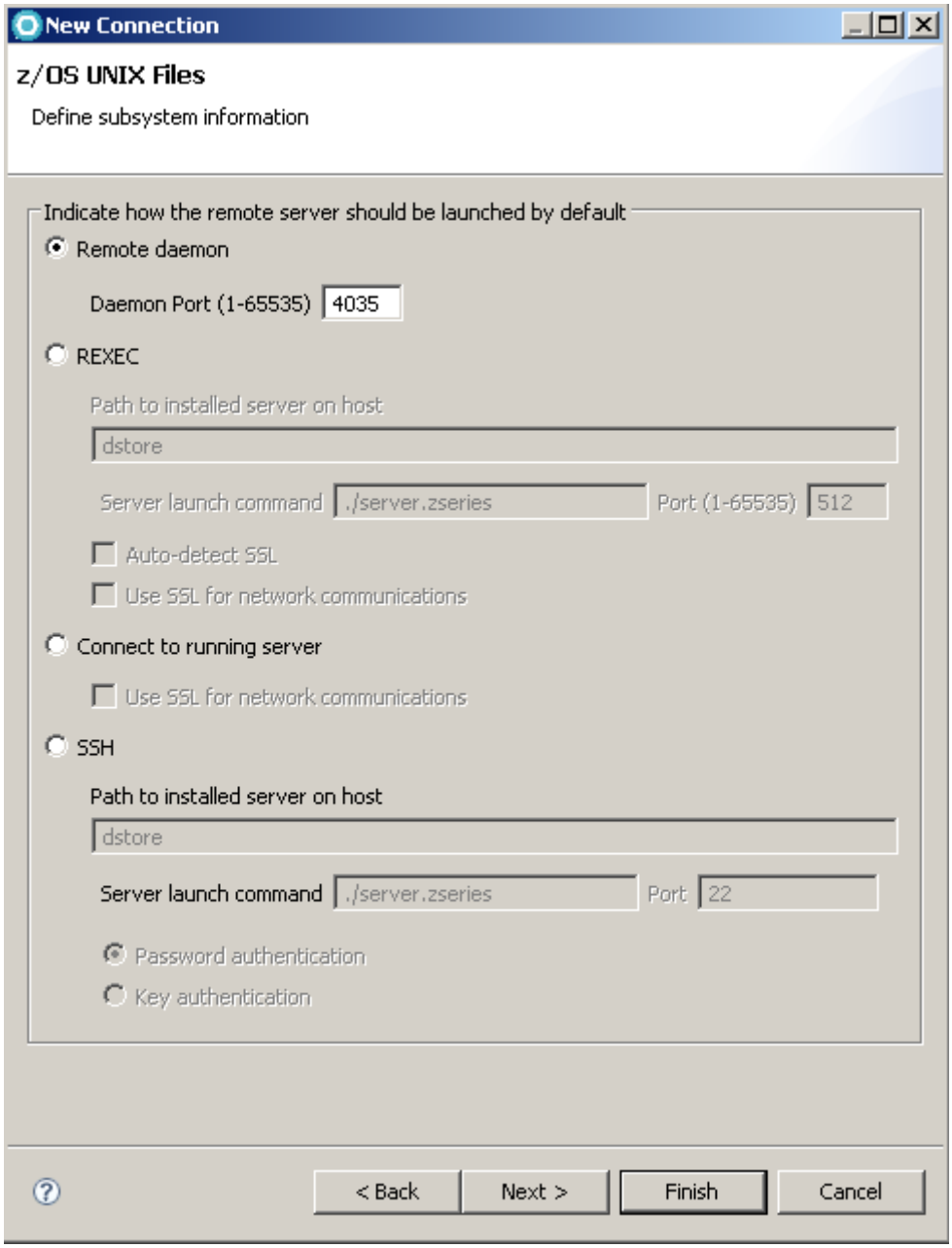
   Optionally in the Description field, provide a short description of the z/OS system that you want to connect to.

   To verify that the hostname or IP address in the Host name field is valid, select the Verify host name check box.

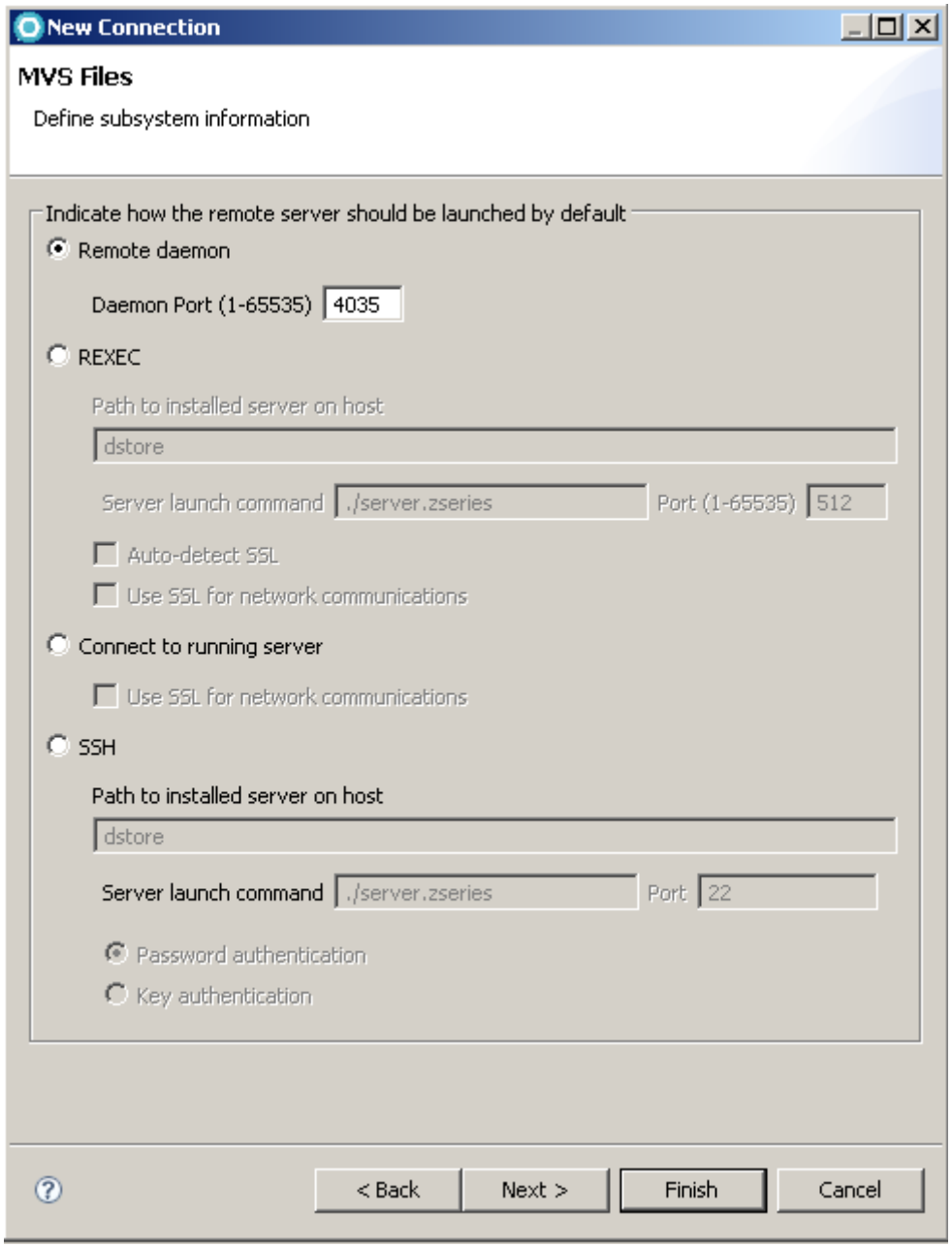
   Click Next.



7. On this panel, leave the defaults and click Next.

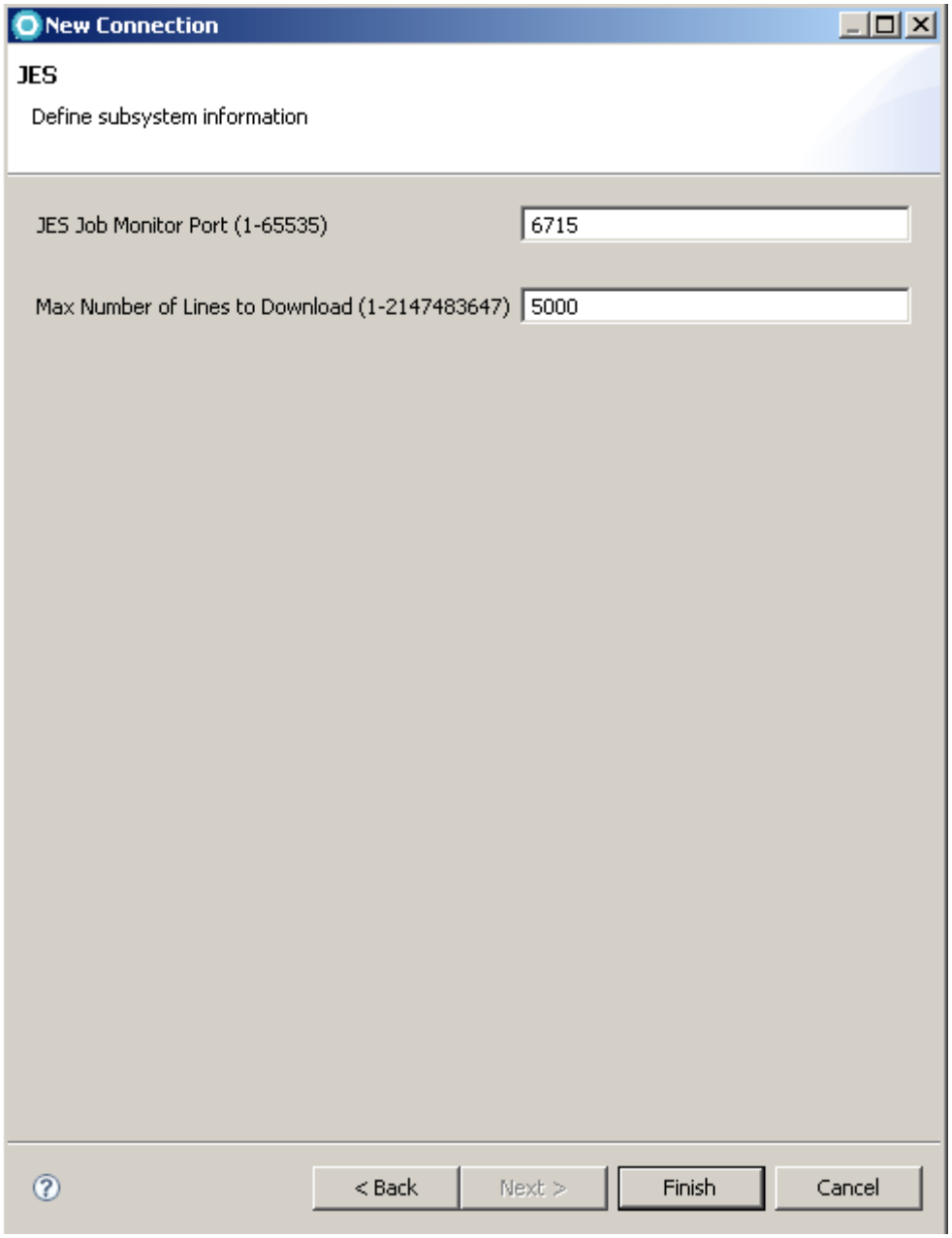


8. On the MVS Files panel, just leave the defaults and click Next.

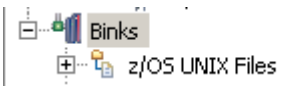


9. And finally again leave the defaults and click Finish to create the new z/OS connection and add it to the Remote Systems perspective.

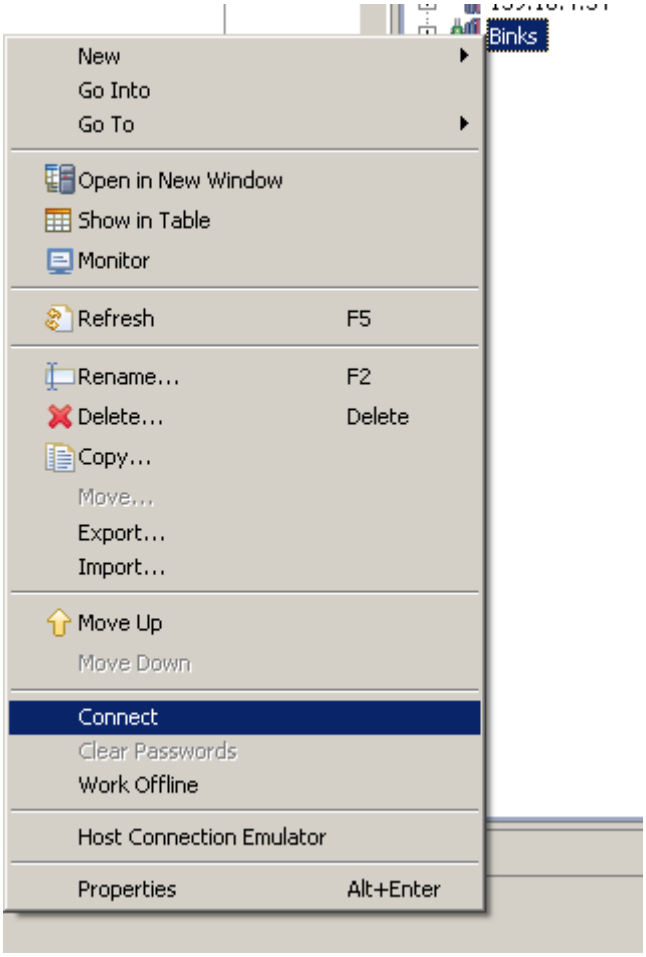
   Note: If you have errors during the connection creation it is because the z/OS system name is not correct or not available (since you specified verify the host name on the step 6).



10. The connection should now be available in your Remote Systems perspective.



11. Now connect to your system by right-clicking Binks and selecting connect from the context menu.



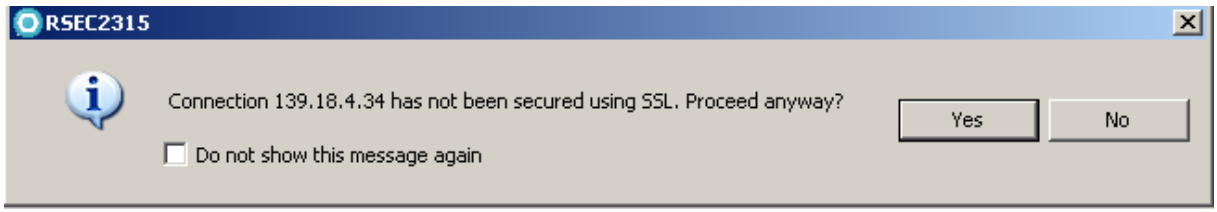
12. You will be prompted for your z/OS userid and password.

    Type the assigned userid and password (Replace PRAK226 by your UserID).

    Optionally check Save user ID and Save password. You will be automatically connected at later times.

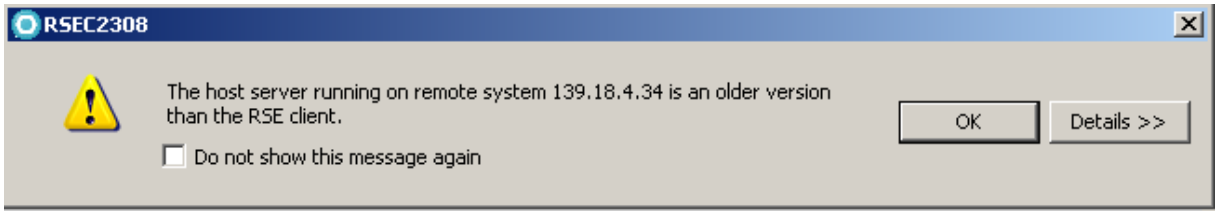
    Click OK to connect to Binks.

13. In RDz Version 7.5 you can secure your connection using SSL. If this has not been done you will be informed by the following message dialog. Click Yes.

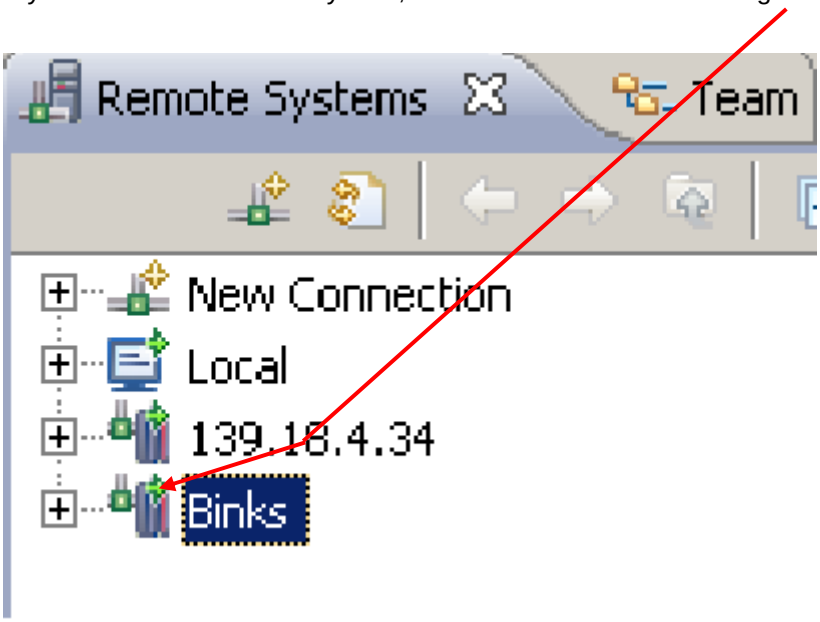


14. If you get a warning like that click OK.

Note: This warning cause from the different versions on the remote system (7.0) and the client (7.5). It is not a problem, because RdZ is backward compatible.



15. If you successfully connect to the remote system, the icon for Binks will have a green arrow.



1.2 Creating needed z/OS datasets

We connected to a remote system and we successfully set up the z/OS system settings.

Now we will allocate some datasets that are required for this Tutorial.

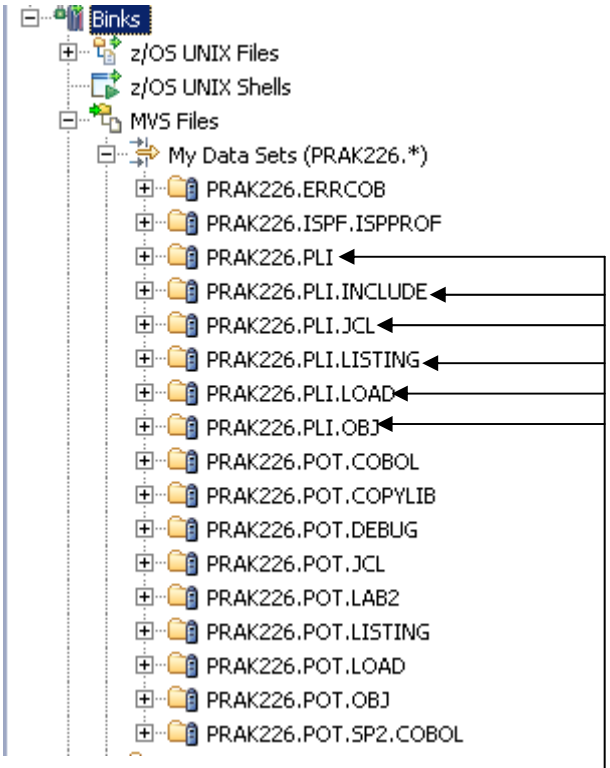
NOTE: In the next steps you have to replace all “PRAK226” with your own id !!!

1. We have to create 6 Datasets:

   PRAK226.PLI
   PRAK226.PLI.INCLUDE
   PRAK226.PLI.JCL
   PRAK226.PLI.LISTING
   PRAK226.PLI.LOAD
   PRAK226.PLI.OBJ

2. Expand Binks -> MVS Files -> My Data Sets

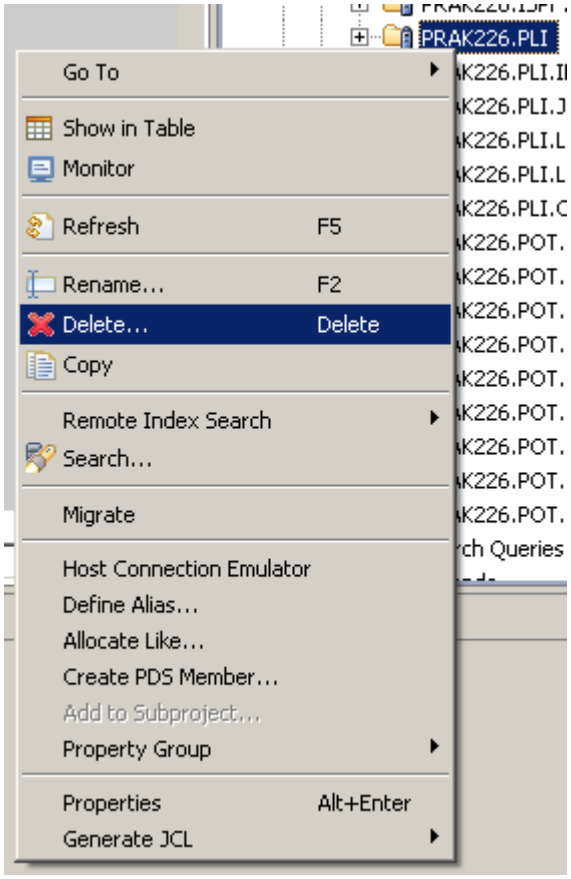
You will see a List of Datasets. (Note: The list can be different from the one below.)



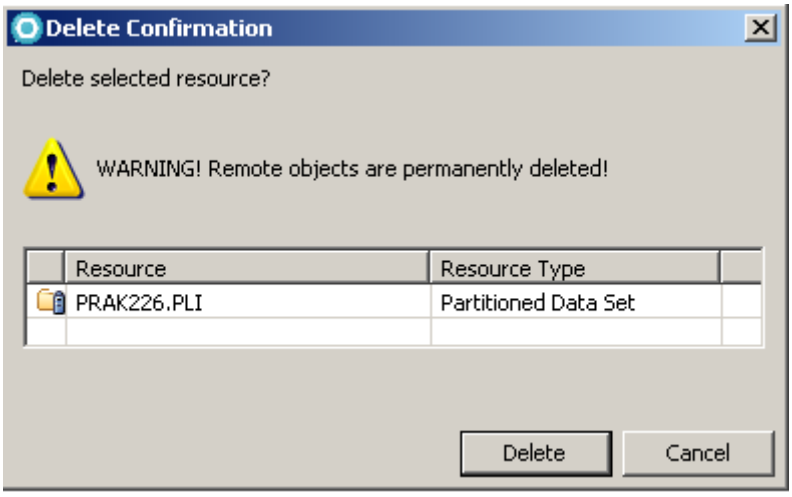
Now we have to delete any dataset that are in the list from step 1.

(Note: If there are no datasets from the list in your MVS, continue on step 6.)

3. Right click on the dataset you want to delete. Select delete.

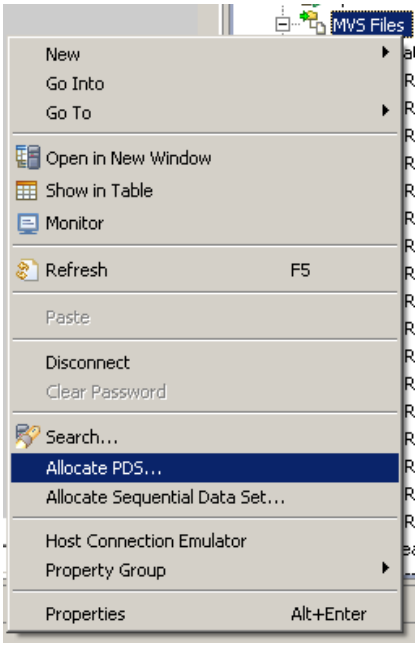


4. Confirm by clicking delete.

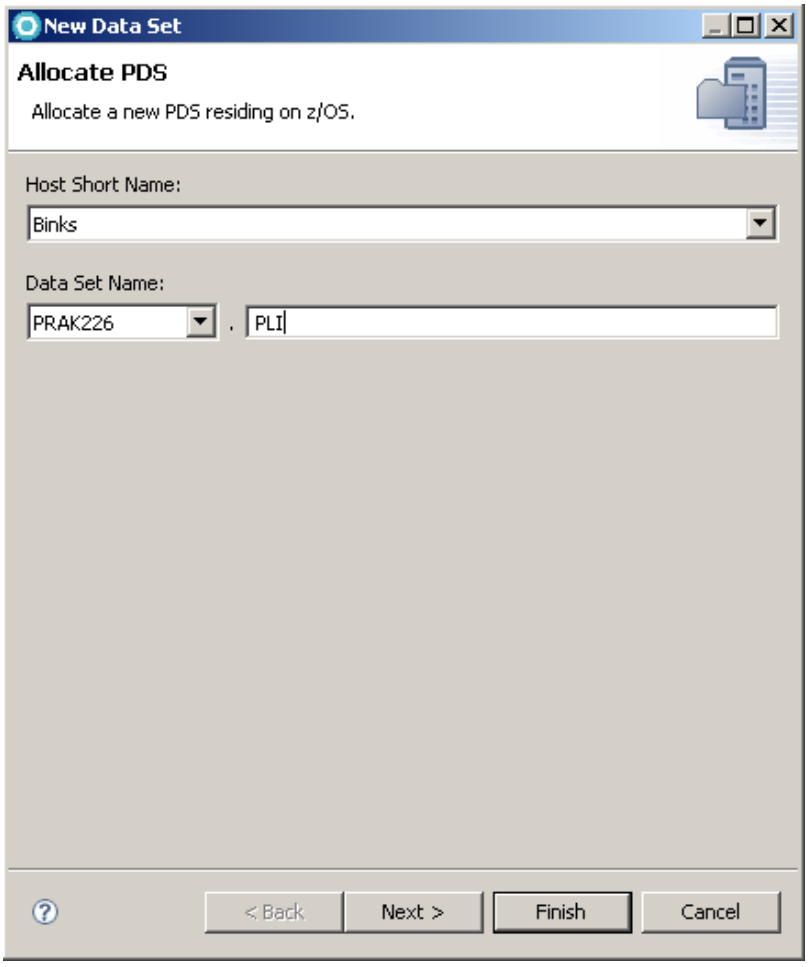


5. Repeat step 3 and step 4 until all datasets from step 1 are deleted.

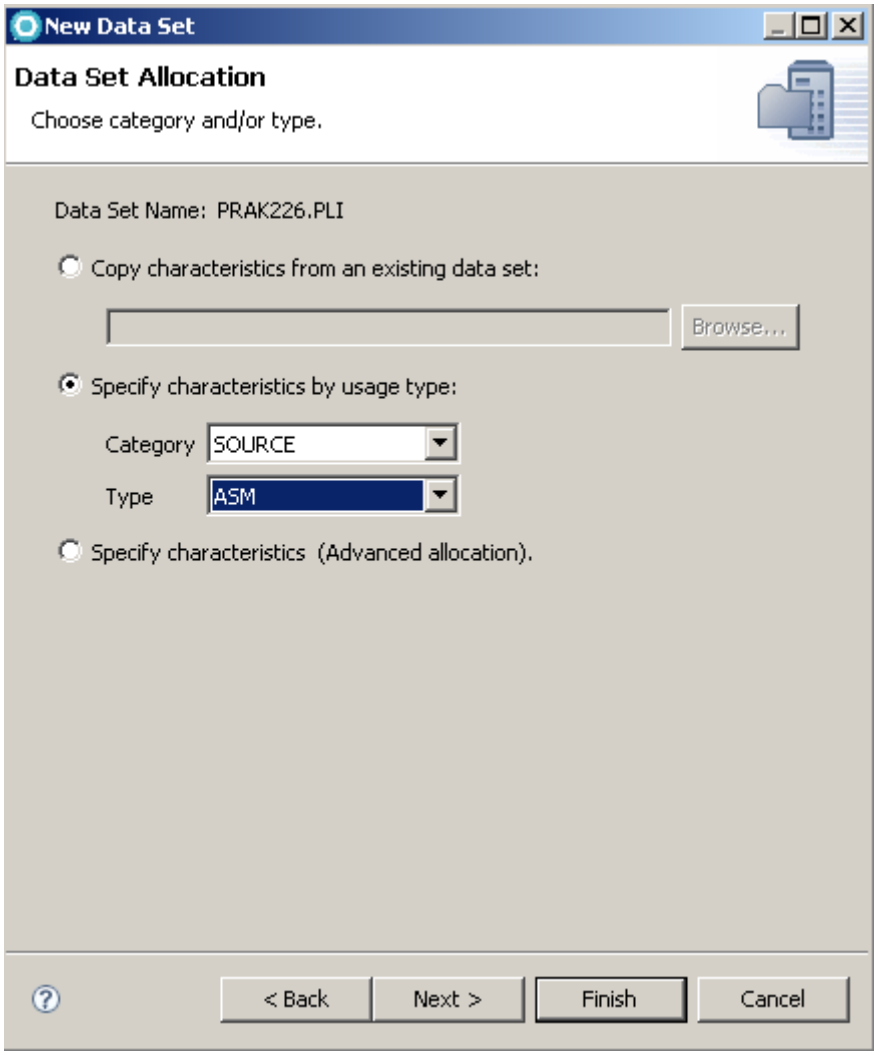
6. Now we will allocate the datasets from step 1. Right click on MVS Files and select Allocate PDS.



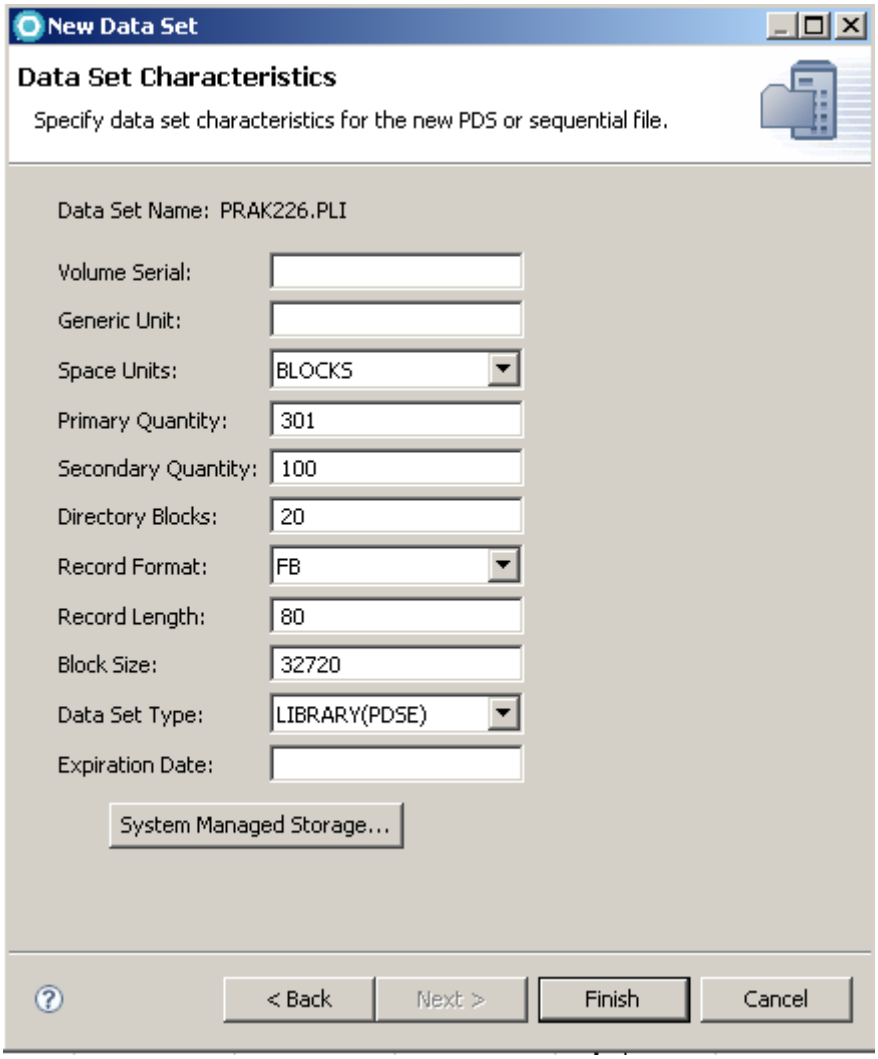
7. Allocate the first dataset by enter PLI. Click on Next.



8. On this panel only click on Next.



9. Enter the data from the picture below and Click Finish.



10. Repeat step 6 to step 9 using the data from the table below to create the other 5 datasets.

Dataset	Space Units	Primary Quantity	Secondary Quantity	Directory Blocks	Record Format	Record Length	Block Size	Data Set Type
PRAK226.PLI.INCLUDE	CYLINDER	2	1	20	FBA	133	1330	PDS
PRAK226.PLI.JCL	CYLINDER	2	1	20	FB	80	8000	PDS
PRAK226.PLI.LISTING	BLOCK	4	150	20	VBA	137	0	LIBRARY
PRAK226.PLI.LOAD	CYLINDER	2	1	20	U	80	6233	PDS
PRAK226.PLI.OBJ	CYLINDER	2	1	20	FB	80	8000	PDS

2 Working with remote files

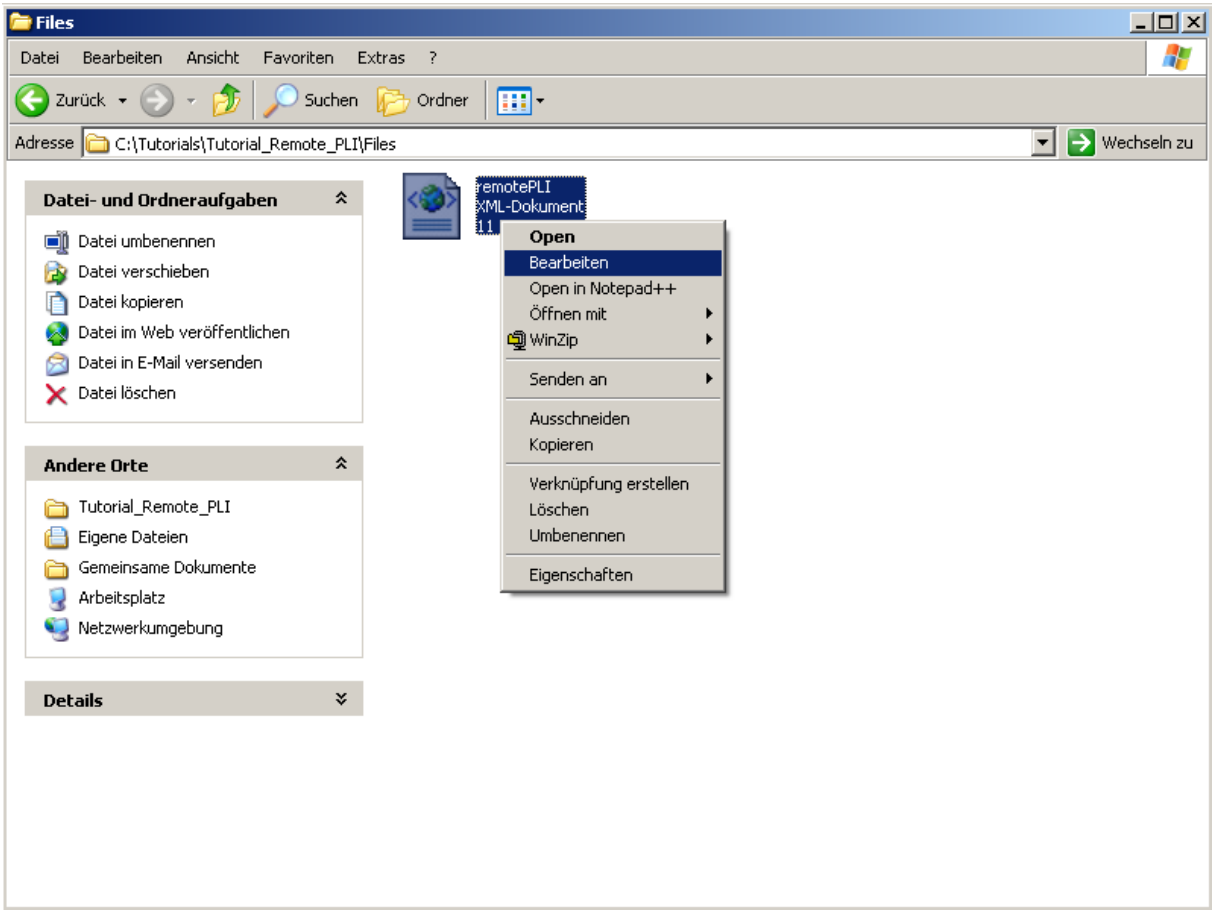
Working in RDz is traditionally realized with projects. For z/OS this means that you will link your assets into a project structure: into z/OS projects. These projects can have either Unix System Services (USS) or MVS subprojects, depending on whether you want to work with files residing in a USS hierarchical file system or in PDS members in MVS. In both cases your assets stay on the host and are only logically linked into the sub projects.

This chapter will guide you through the process of creating a remote project and setting its properties for different languages and runtimes.

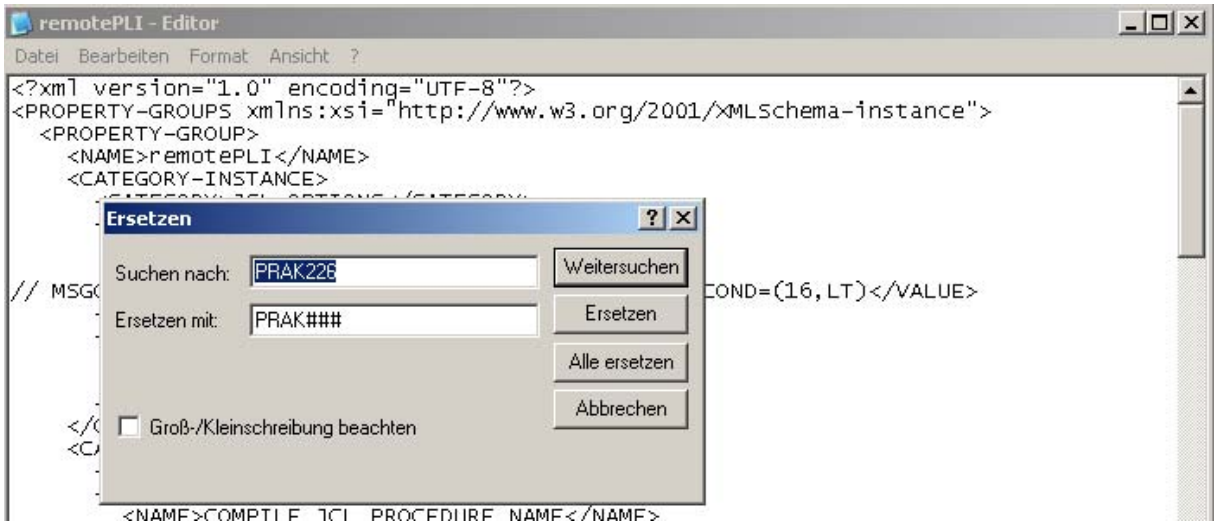
2.1 Import a property group

1. First we have to import a property group for remote PL/1 projects. A property group includes all setting you need for our remote PL/1 project. You can use it for further projects, too.

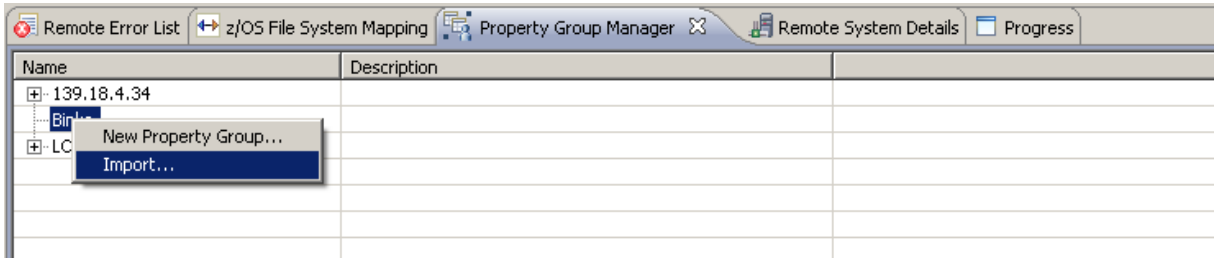
   Go to place were you have saved the files from this tutorial and open remotePLI.xml to modify it.



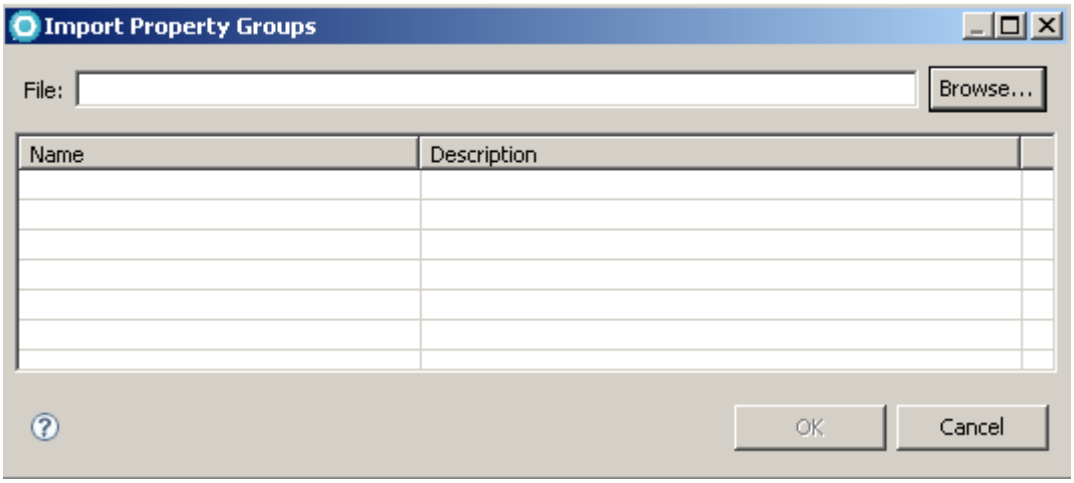
2. Replace all “PRAK226” with your id (e.g. PRAK007). After that save and close the file and return to RDz.



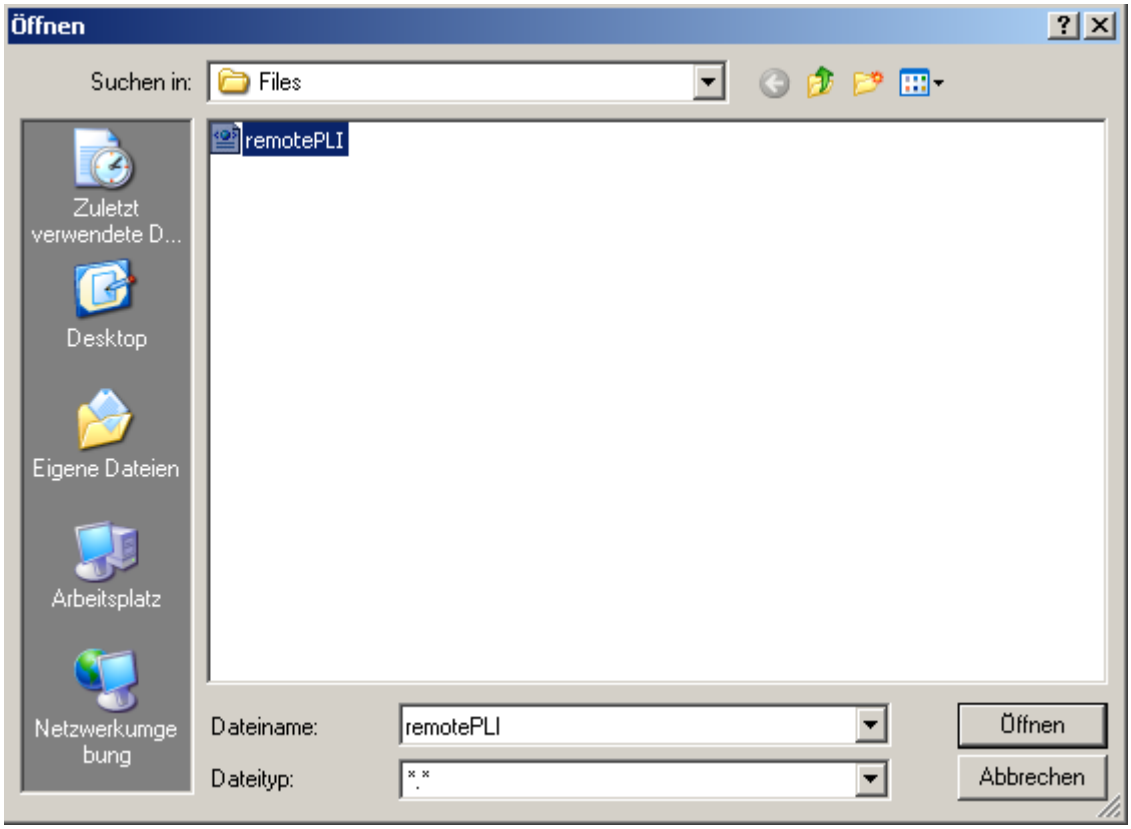
3. Go to the Property Group Manager. Right click on Binks and select Import.



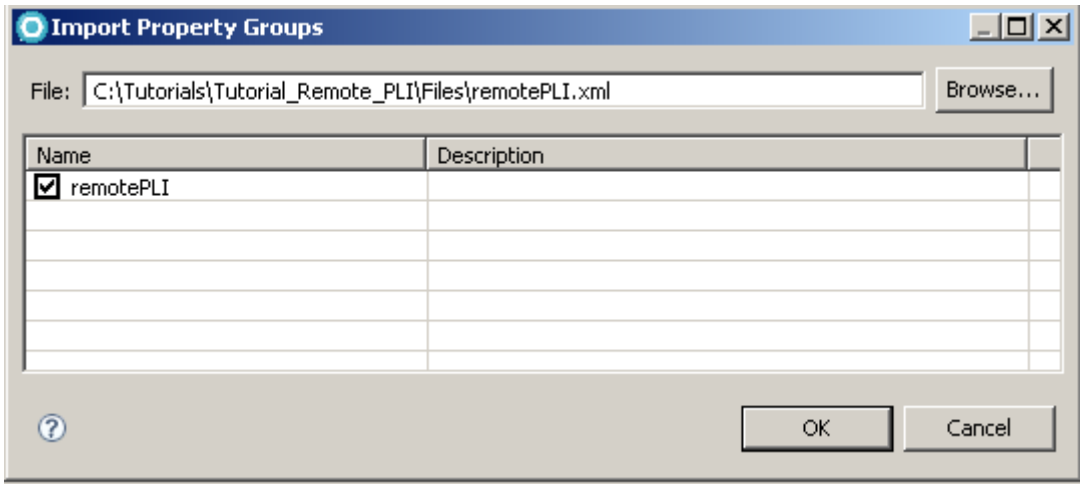
4. Now click on Browse….



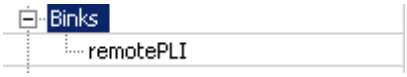
5. …go to the location where you saved the remotePLI.xml. Select it and press enter.



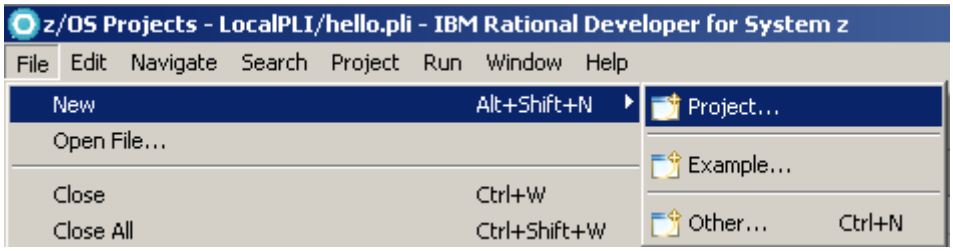
6. Check remotePLI and click OK.



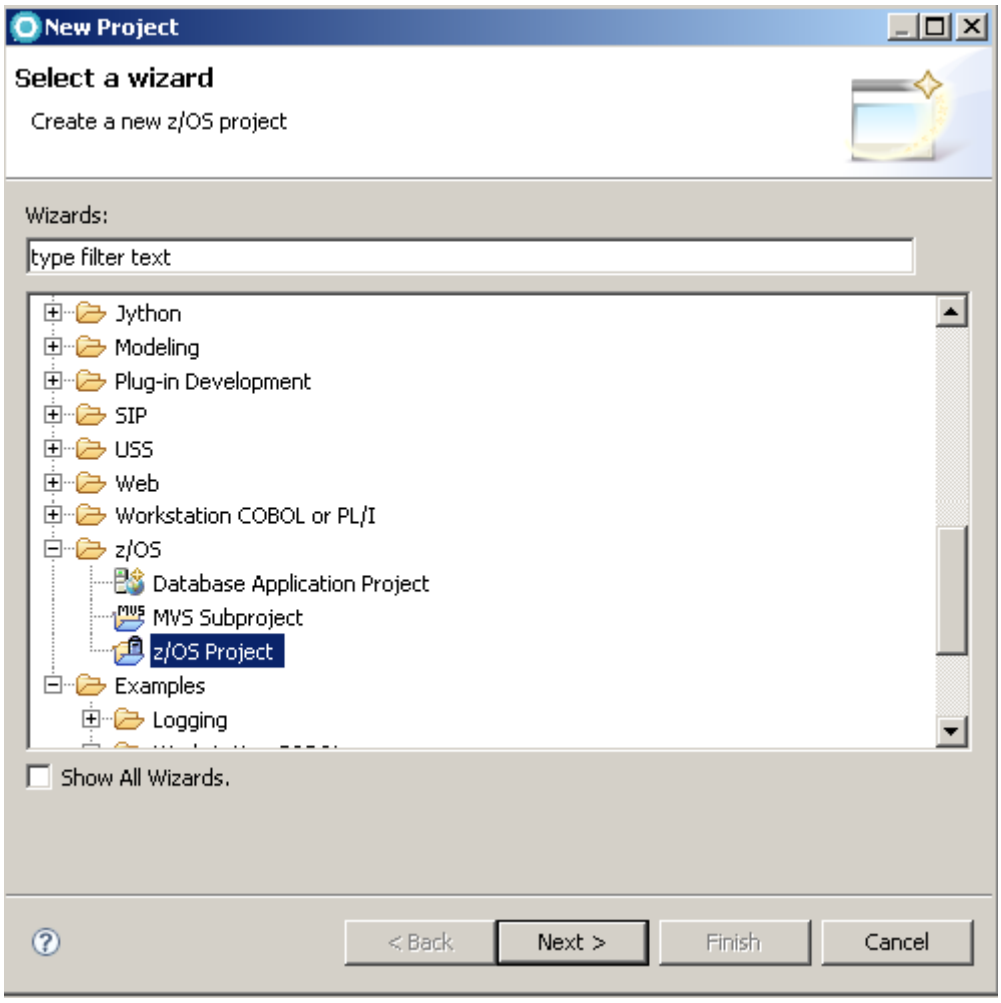
7. Now Binks is connected with the property group remotePLI.



8. Select File → New → Project… from the menu bar.

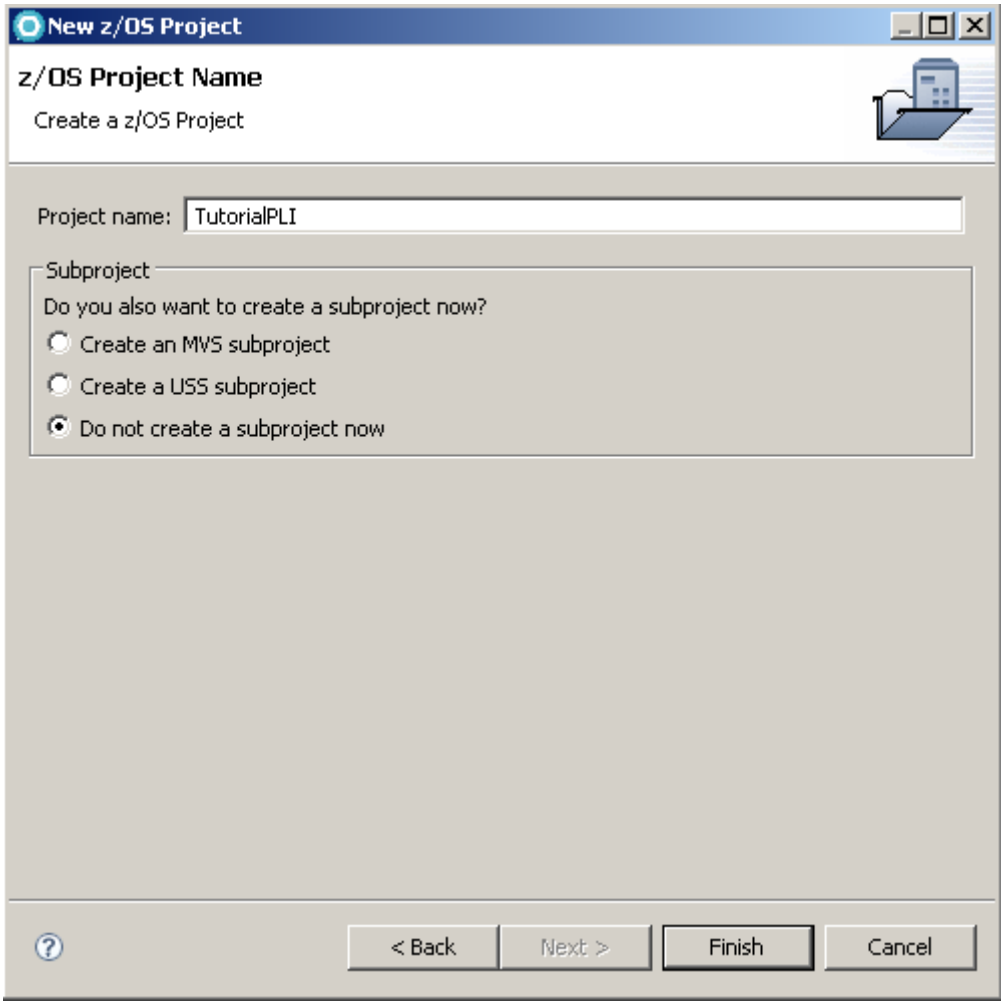


9. Scroll down, Expand z/OS, select z/OS Project and click Next.



10. On the z/OS Project Name panel, name the project TutorialPLI and select the “Do not create a subproject now” radio button. Click Finish.

    (Note: We will create a Subproject later. Of course, you could have done it straight away with creating a z/OS Project)



2.2 Create a MVS Subproject

First of all, you can create a MVS Subproject only when you are connected to the system.

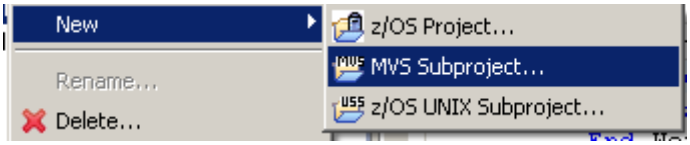
An MVS project is in either of two states:

In online state, the project is connected to the system to which the project refers. You can directly change the data sets that are stored in that system.

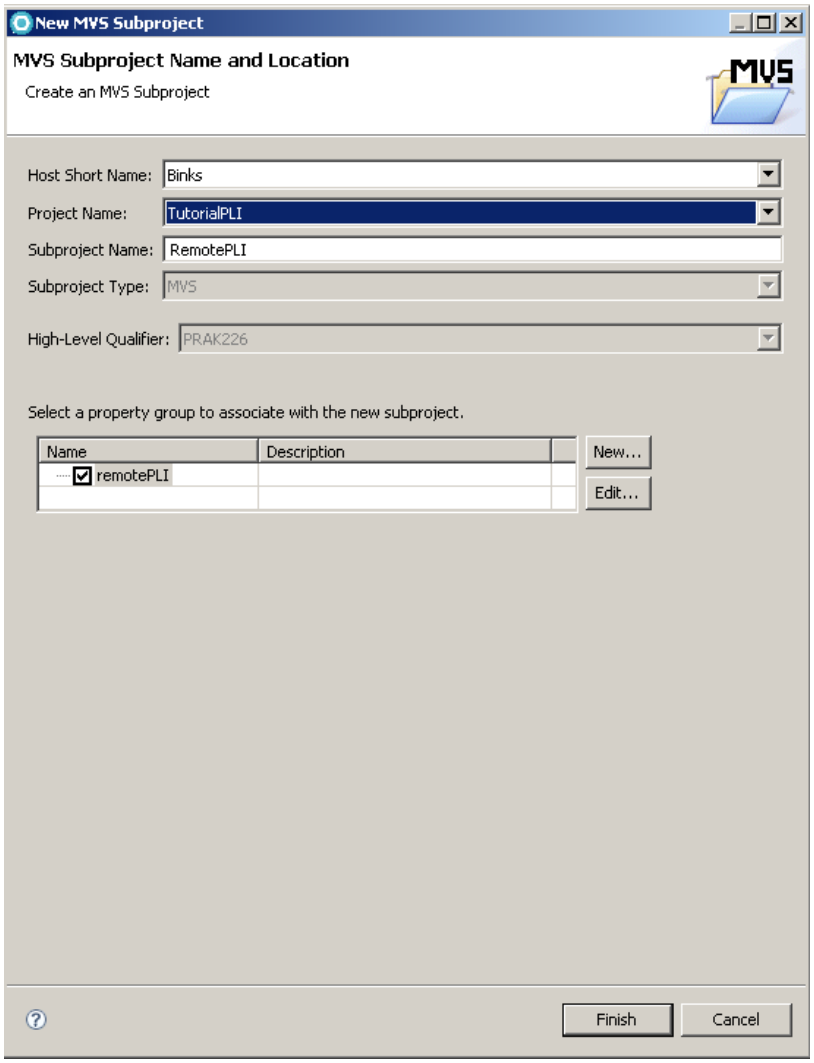
In offline state, the project can access only workstation-based files, which may be new or may be copies of mainframe resources.

As you disconnect from z/OS, you can specify the data sets and members to be transferred to the workstation. When you switch back to online state, the specified files are automatically uploaded to the mainframe, with a confirmation message that keeps you from unintentionally overwriting resources.

1. In the z/OS Projects view, right-click TutorialPLI and select New → MVS Subproject from the context menu.

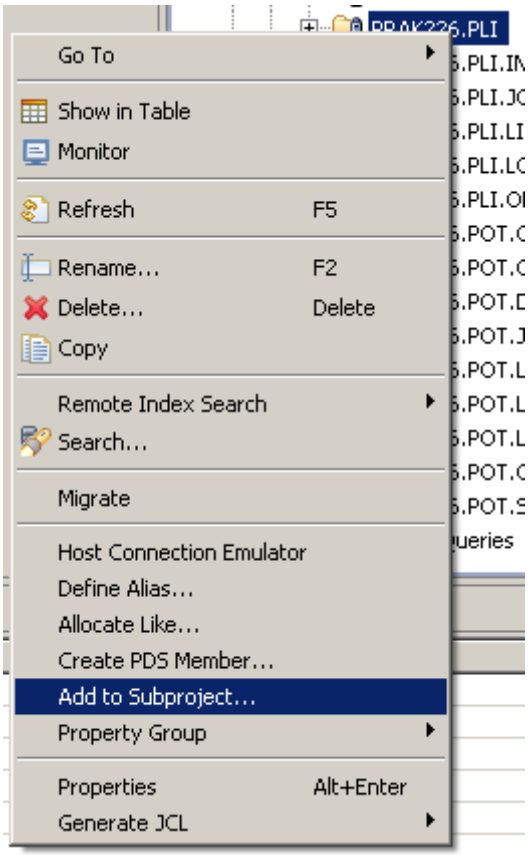


2. Name your MVS subproject RemotePLI and choose Binks as Host Short Name. Also check remotePLI as property group and click finish.

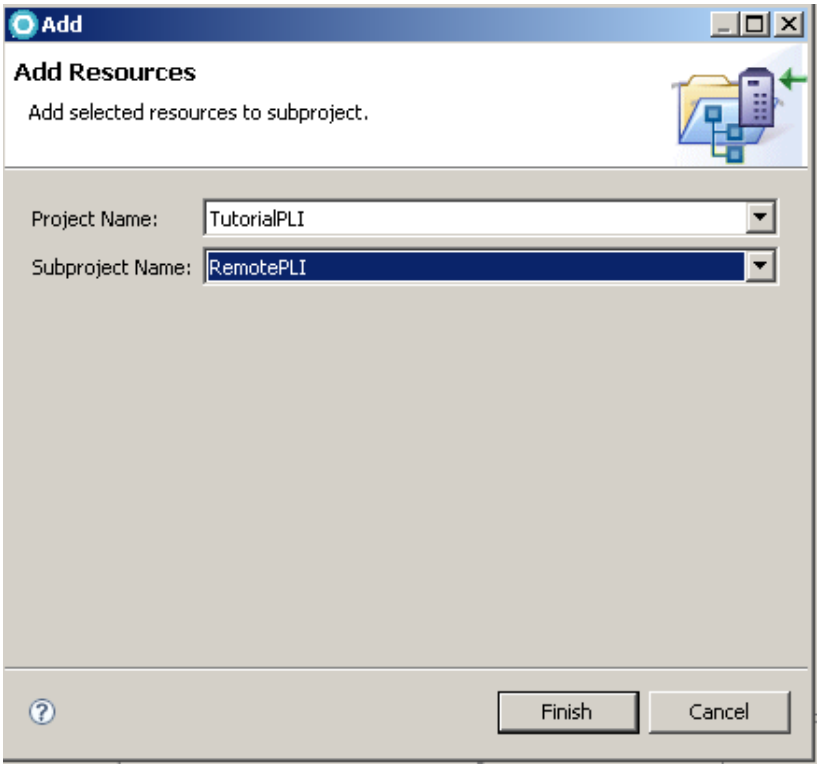


2.3 Add resources to your project and create a member

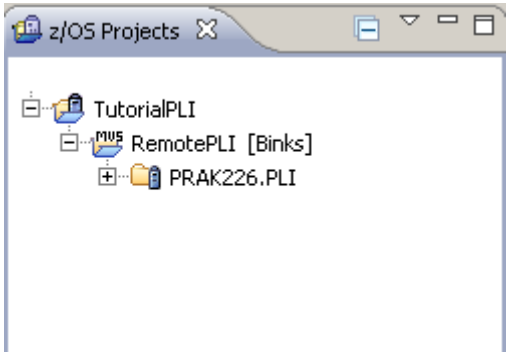
1. Switch to the Remote Systems view. Expand the MVS Files under Binks. Right click on PRAK226.PLI and select Add to subproject.



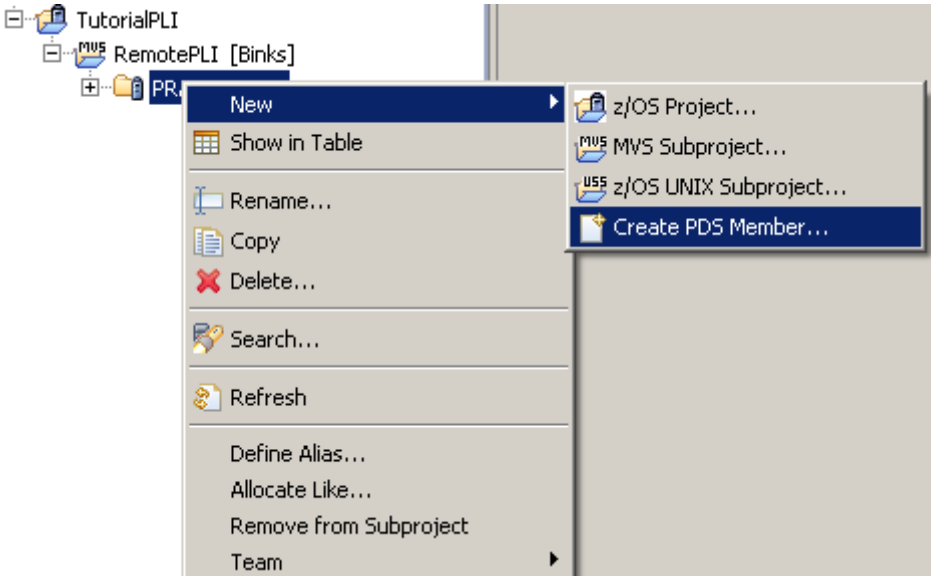
2. Accept project name TutorialPLI and subproject name RemotePLI. Click Finish to add the dataset.



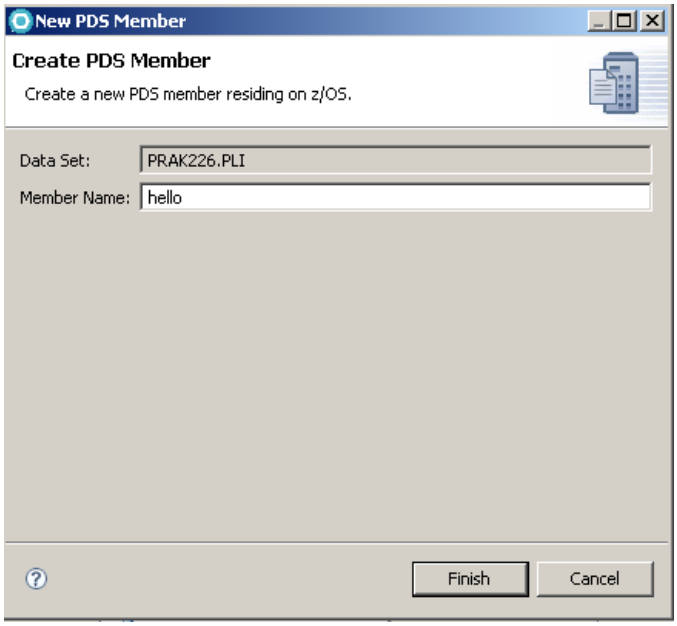
3. Switch to the z/OS Projects view and you will see that the dataset have been added to the RemotePLI project. The z/OS Projects view should look like the following:



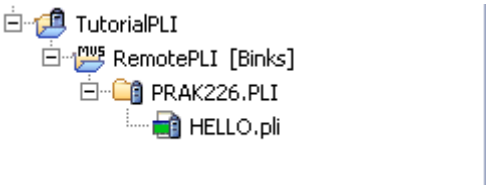
4. Right click on PRAK226. Select new -> Create PDS Member.



5. Enter hello as Member Name and click Finish.



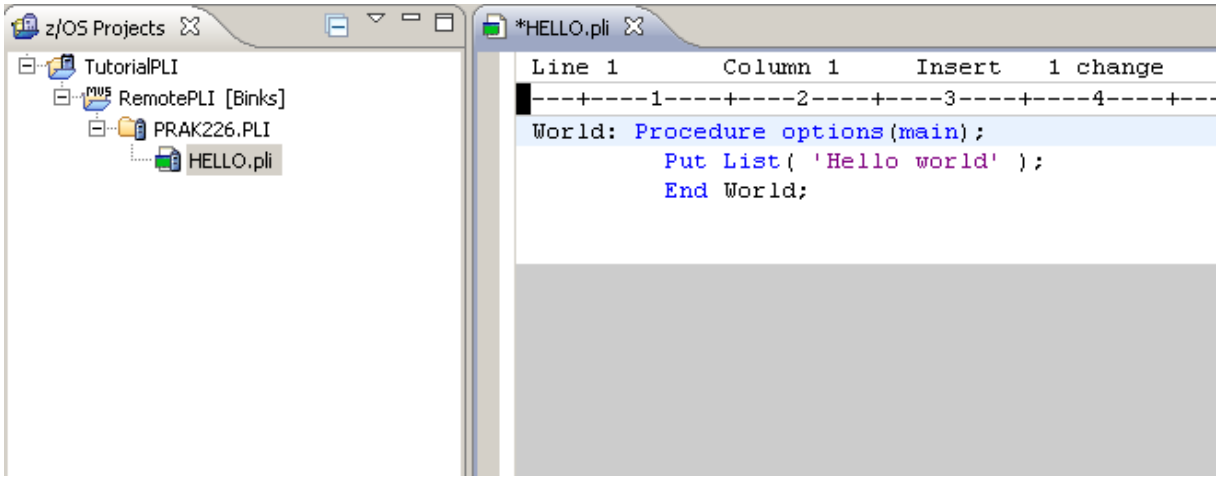
6. The z/OS Projects view should look like the following:



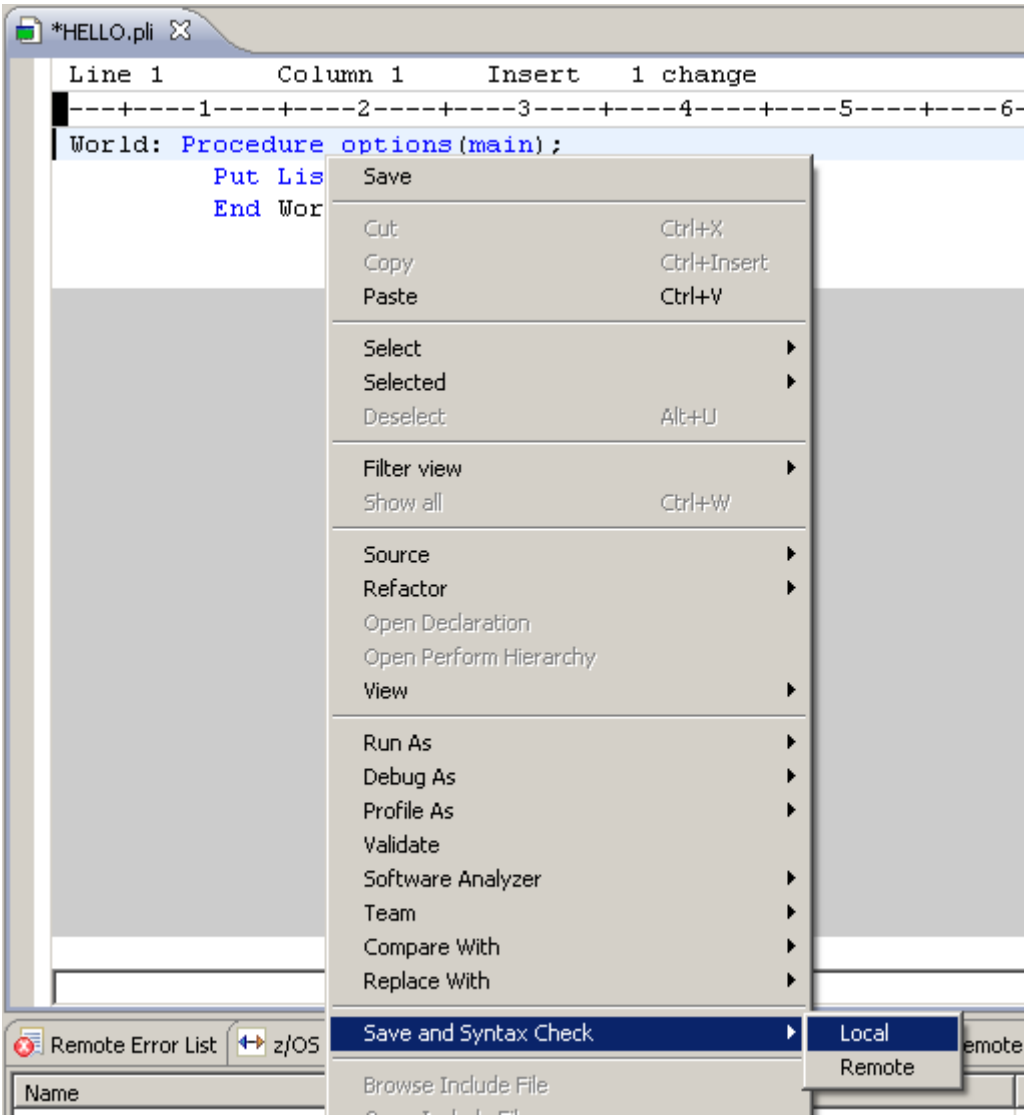
7. Double click on HELLO.pli and enter following code:

World: Procedure options(main);
Put List('Hello world');
End World;

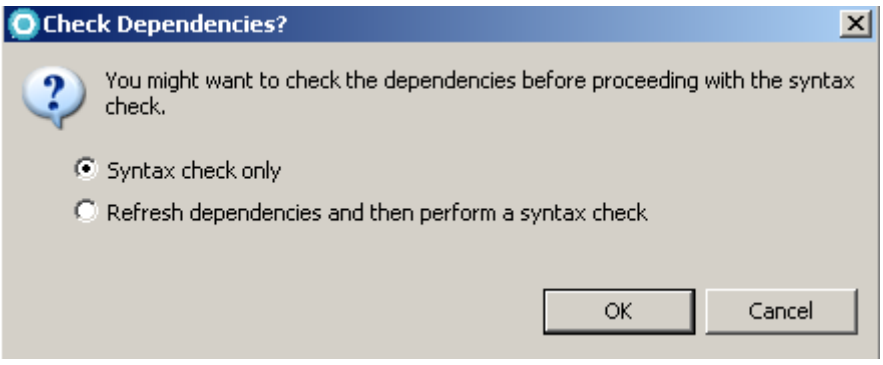
Now it should look like the following:



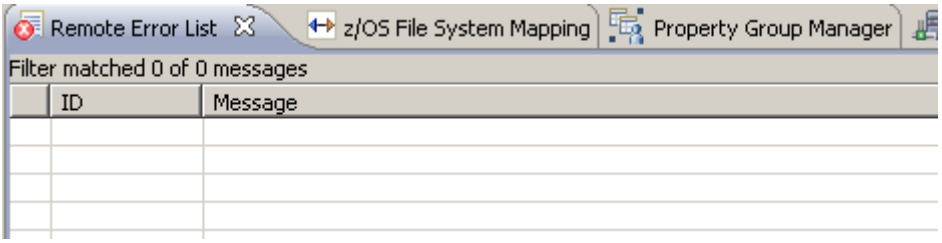
8. Now Right click. Select Save and Syntax Check -> Local.



9. Choose Syntax check only and click OK.



10. Be sure that there are no errors in the Remote Error List.

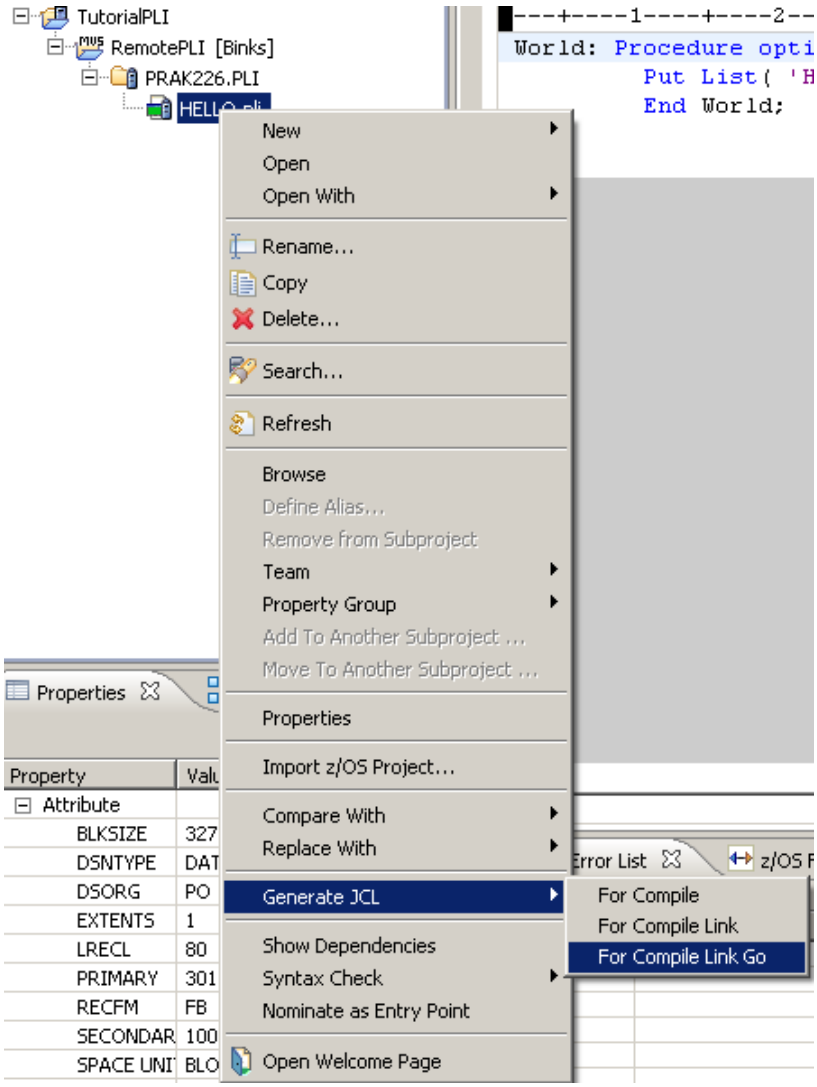


2.4 Compile/link/execute the remote PL/1 program

In this part you will learn how to compile your programs, link them and execute them on z/OS. You will therefore use a feature called JCL generation, where the properties of your Remote z/OS project are used to generate JCL for Compile only, Compile and Link or Compile, Link and Go.

We are using this feature for demonstration purposes. In real world of course you will either use you company's provided compilation procedures or take advantage of a source code management system like SCLM or Endeavor.

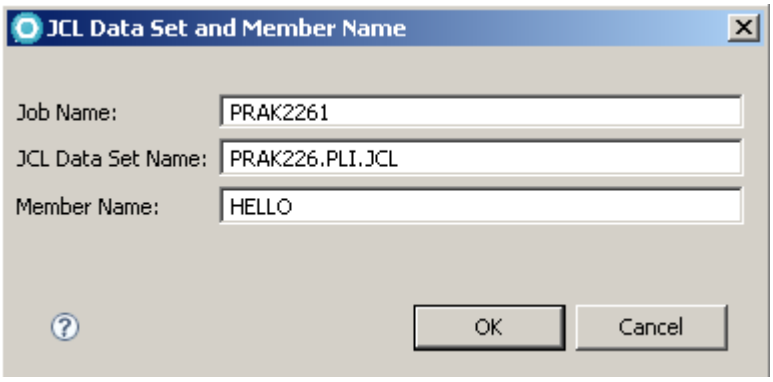
1. Using the z/OS projects View, right-click on hello.pli and select Generate JCL → For Compile Link Go.



2. On the JCL Data Set and Member Name window, notice that the JCL Data Set Name is set to the value you specified for your project settings.

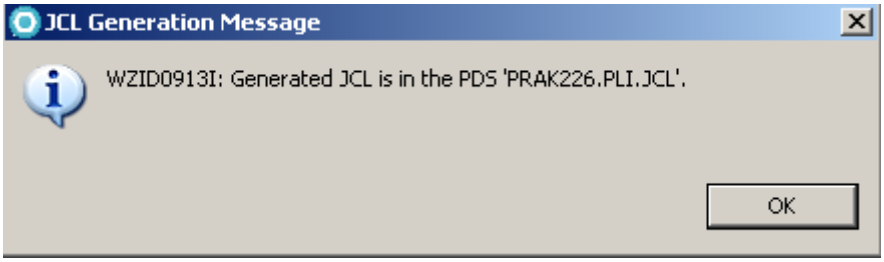
   (PRAK226 should be replaced with your User ID)

   Click OK.

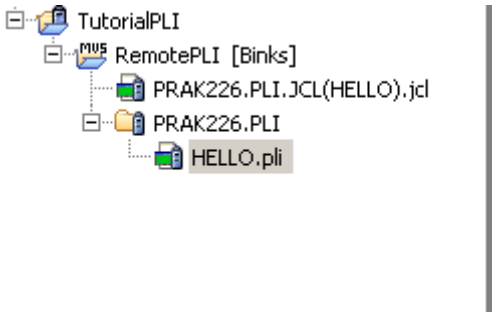


3. You should see this message (Here for UserID PRAK226).

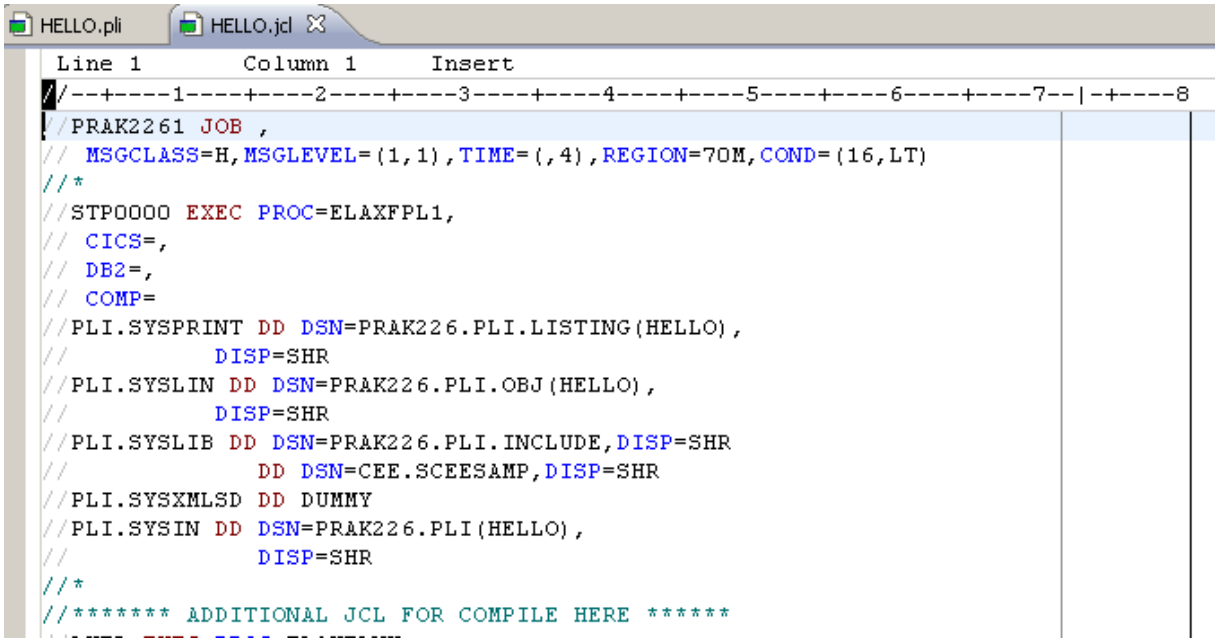
   Again click OK.



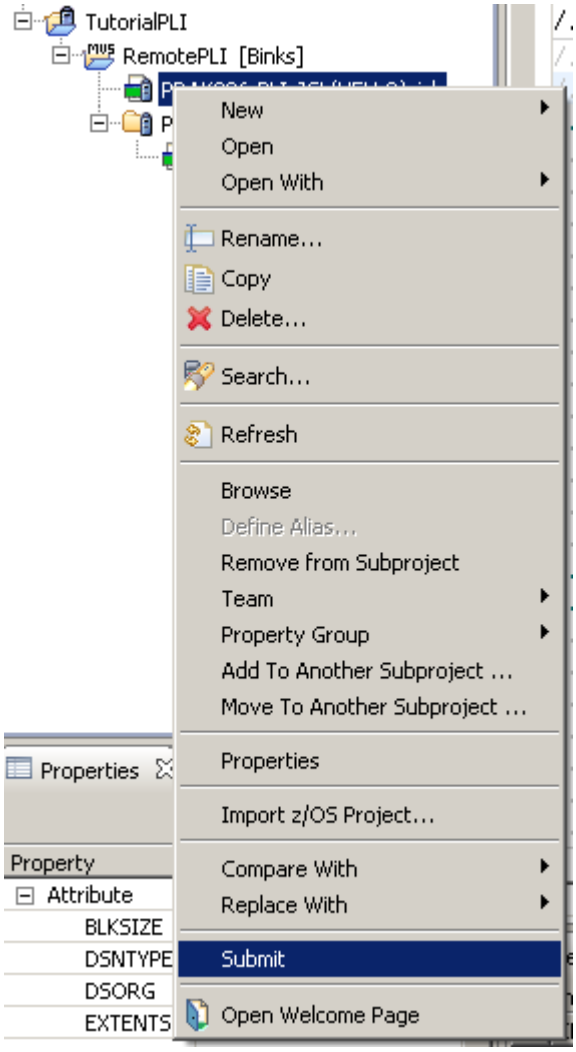
4. Go to your z/OS Projects view and you will see that hello.jcl was generated.



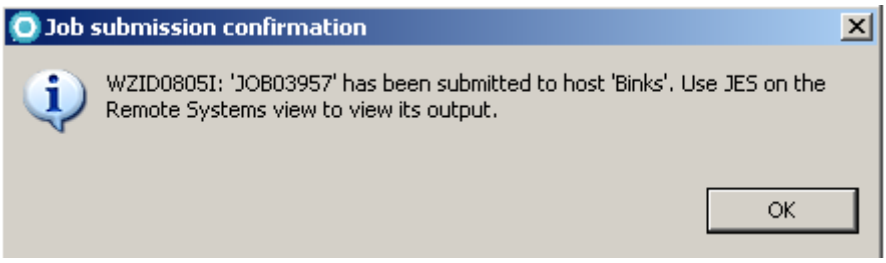
5. In the z/OS Projects view , double click on hello.jcl to open the JCL file in the editor window.
   Take a look to the JCL code. After that close Hello.jcl.



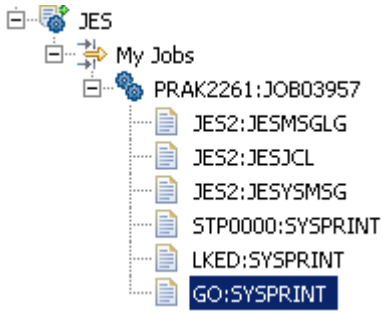
6. Right click on PRAK226.PLI.JCL(HELLO).jcl and select Submit.



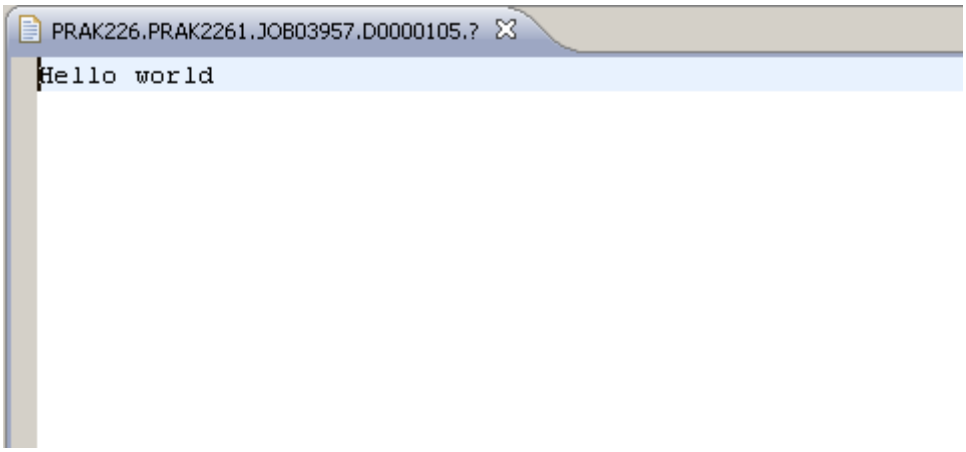
7. You will get a Job submission confirmation message.
   Click OK.



8. To see the output of your program, switch to the Remote Systems window. Expand JES and My Jobs. Look for the JOB ID of your submitted job, expand it. and double-click GO:SYSOUT.
   Note: If you don't see your Job, right click on "My Jobs" and choose "Refresh"



9. If you did everything right, you should see the above Output in the editor window.



Congratulation, you completed Tutorial Remote PL/1!