Automatisierung von Java Anwendungen (10)

Bean Scripting Framework (BSF), 4

Automating/Scripting of OpenOffice.org (OOo)
Openplatform, Opensource

Prof. Dr. Rony G. Flatscher

OpenOffice.org/Staroffice
Sources of figures, examples and hints

- From the excellent OOo "Developer's Guide", cf.
  - http://www.OpenOffice.org
- Ahammer A. "OpenOffice.org Automation: Object Model, Scripting Languages, 'Nutshell'-Examples" at the "WU Wien", cf.
  - http://wi.wu-wien.ac.at/rgf/diplomarbeiten/
    - Also Augustin, W., Realfsen et.al. (!)
- Code snippets for OOo (different languages)
  - http://codesnippets.services.openoffice.org/
- From the excellent book, "OpenOffice.org Macros Explained" by Mr. Andrew Pitonyak, cf.
  - http://www.HetzenWerke.com

  → http://documentation.openoffice.org/HOW_TO/index.html

OpenOffice.org Brief History, 1

- StarOffice
  - Originates in Germany
  - Portable C++ class library ("Star")
    - Allow creation of a portable integrated office suite
    - Goal: compatibility to MS Office
  - 90'ies
    - OS/2
    - Windows
    - Explored Macintosh, Unix

OpenOffice.org Brief History, 2

- StarOffice, continued
  - Bought by Sun
    - Development transferred to the U.S.A.
  - Solaris
    - Allowed MS Office compatible office suite
  - Opensource
    - In parallel to commercial version "StarOffice"
    - "OpenOffice.org" (OOo)
      - Linux, Macintosh, OS/2, Solaris, Windows, …

OpenOffice.org
Developer's Bird Eye's View, 1

- Set of services to create and maintain documents
- All common functionality of all types of documents is extracted and organized as a set of interfaces
  - E.g. Loading, saving, printing documents
- For each type of document the specific functionality is extracted and organized as a specialized set of interfaces
    - E.g. TextCursors ("write"), Cell-Manipulation ("calc")

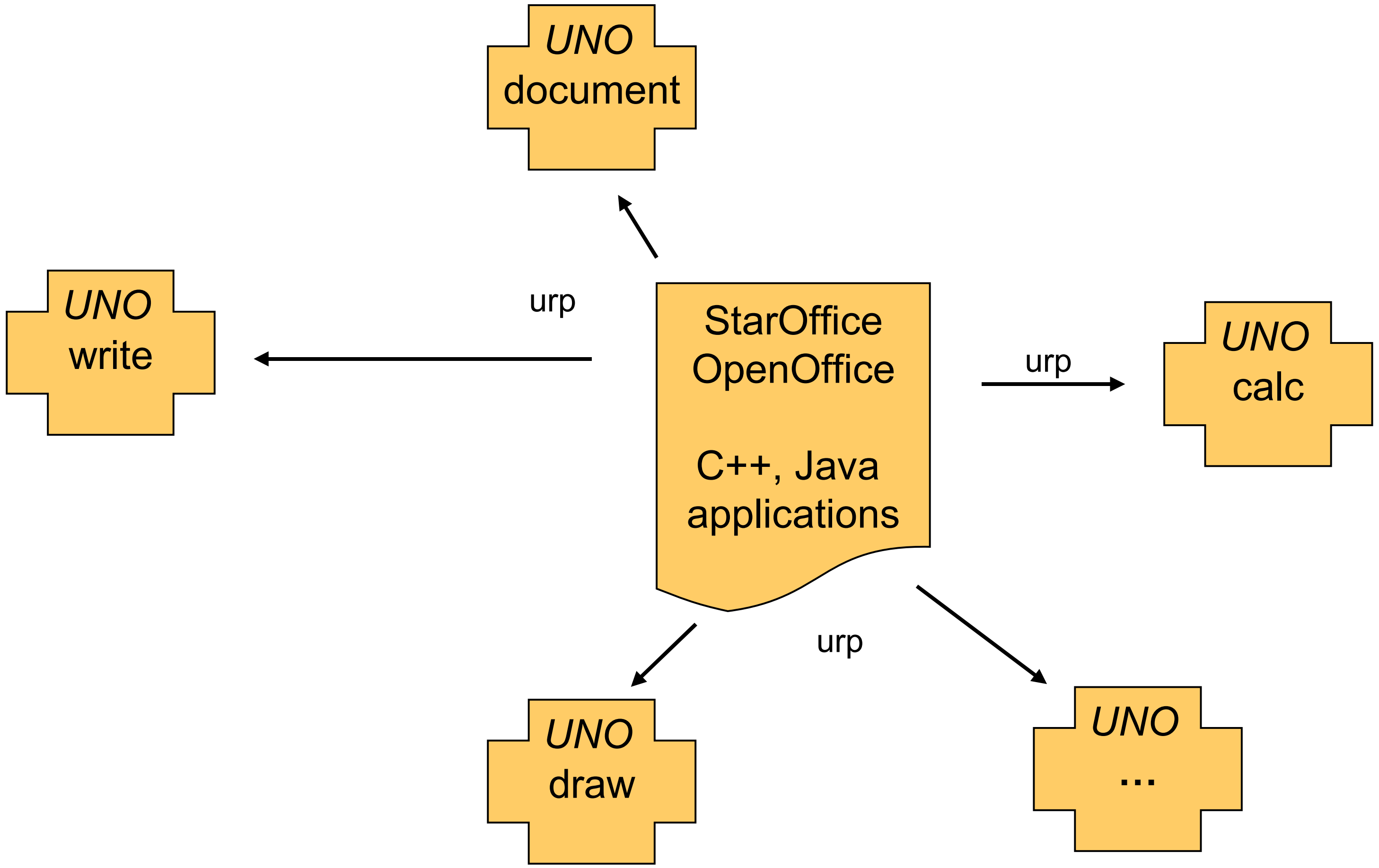
OpenOffice.org Developer's Bird Eye's View, 2

- Client/Server Architecture
  - Employing distributable components ("UNO")
    - Server can run on any computer in the world!
    - Operating system of server as well as that of the client is irrelevant!
  - Communication
    - TCP/IP sockets
    - Named pipes, if available
  - Client can run on the same machine as the server

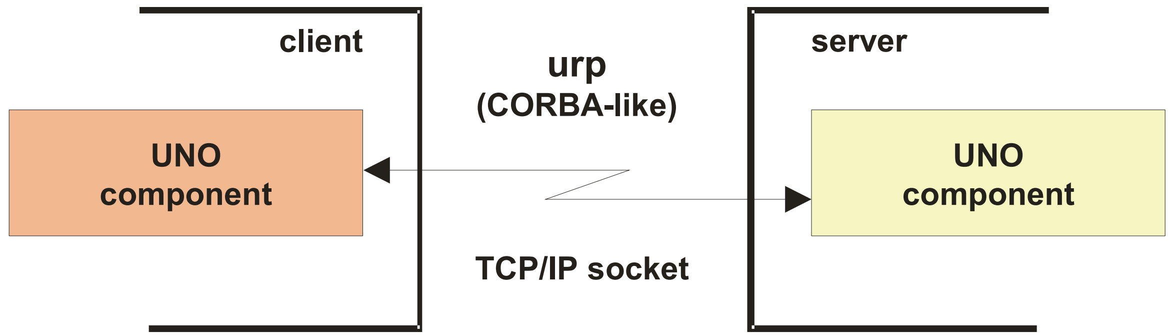
OpenOffice.org Building Blocks, 1

- "UNO"
  - **U**niversal **N**etwork **O**bjects
  - Distributable, interconnectible infrastructure
  - All functionality is organzied in the form of classes
    - "UNO classes"
- "urp"
  - "UNO remote protocol"
    - CORBA-like protocol

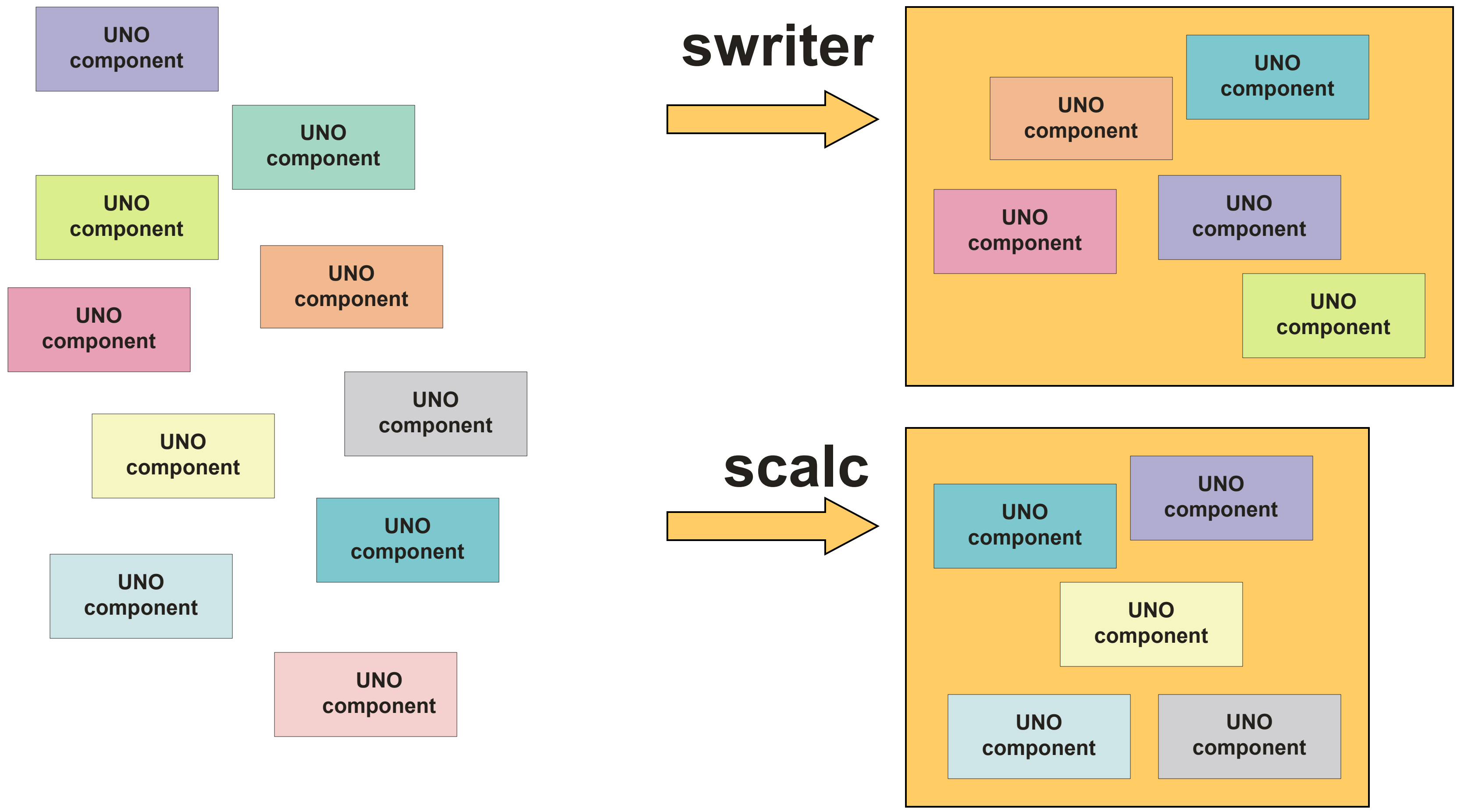
OpenOffice.org Building Blocks, 2



OpenOffice.org Building Blocks, 3



OpenOffice.org Building Blocks, 4



OpenOffice.org Building Blocks, 5

- "Service Managers"
  - Supplied by servers
  - Can be used to request services from the server
  - Returned service allows access to a part of the "office" functionality, E.g.
    - *com.sun.star.frame.Desktop*
    - *com.sun.star.configuration.ConfigurationProvider*
    - *com.sun.star.sdb.DatabaseContext*

OpenOffice.org Building Blocks, 6

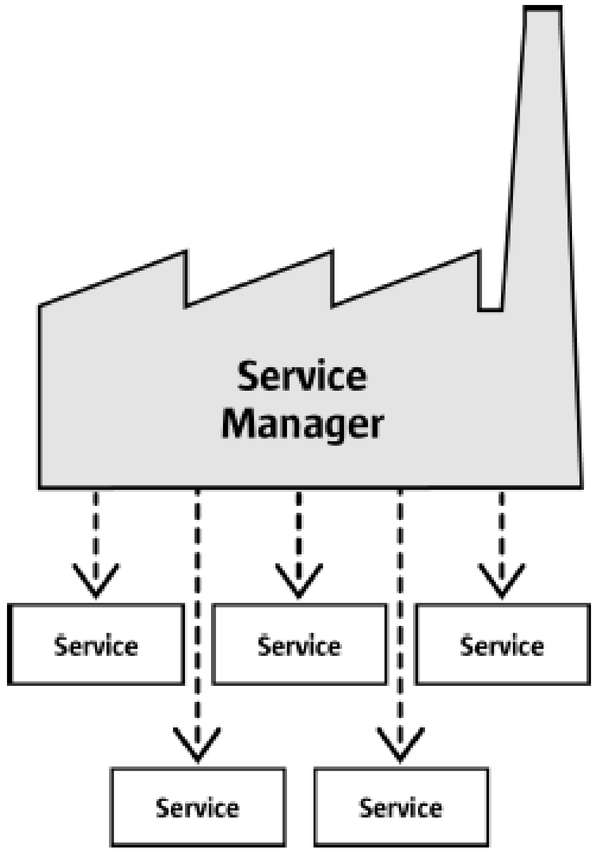


Illustration 2.1: Service manager

OpenOffice.org Building Blocks, 7

- "Services"
  - Can be comprehensive
  - Are organized in partitions named
    - "Interfaces" (group of functions/methods) and
    - "structs" (group of related properties only)
  - Depending on the desired task you need to request the appropriate interface, e.g.
    - com.sun.star.view.**X**Printable
    - com.sun.star.frame.**X**Storable
    - com.sun.star.text.**X**TextDocument

OpenOffice.org Building Blocks, 8

- An example
  - Two services with seven interfaces exposed
    - There are more available
  - "OfficeDocument"
    - Four interfaces
  - "TextDocument"
    - Three interfaces

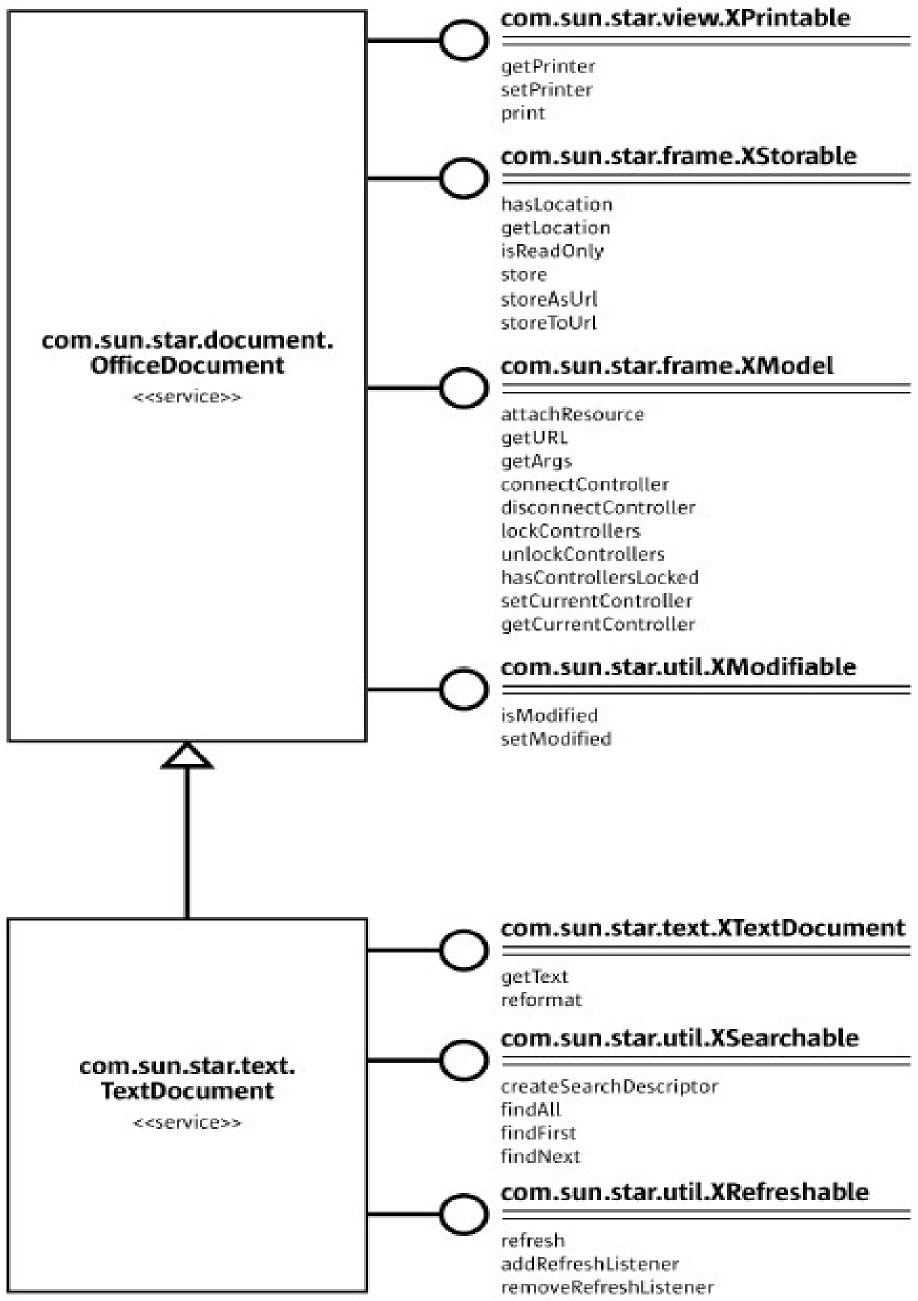


Illustration 2.3: Text Document

OpenOffice.org Building Blocks, 9

- Client needs to get in touch with the server
  - URL-style connection string
  - Server creates an object to interact with and returns a handle for it to the client

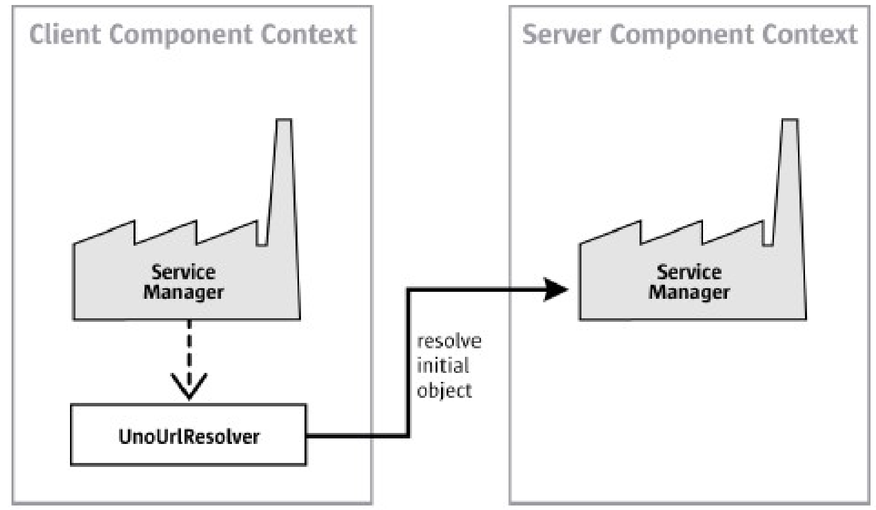


Illustration 2.2: UnoUrlResolver gets Remote ServiceManager

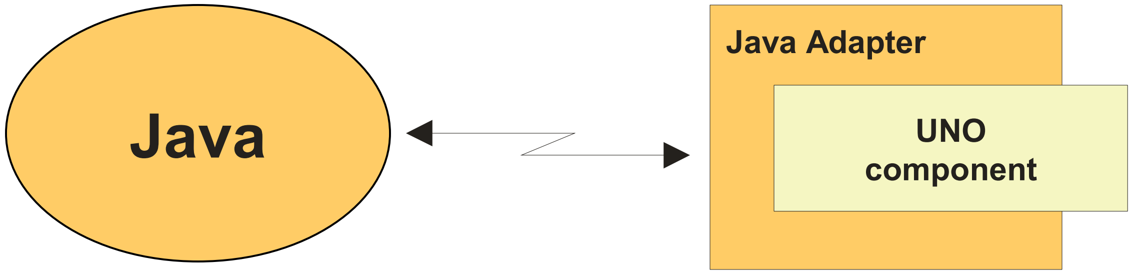
OpenOffice.org Programming languages

- OOo version 1.1
  - C++
  - StarBasic
    - Scripting language
  - **Java**
  - Python
- OOo version 2 (fall 2005) in addition
  - Java based Scripting Framework
    - BeanShell (interpretable Java)
    - JavaScript (Rhino)

OpenOffice.org
Java, 1

- Full implementation for UNO
  - "Java UNO"
- Every UNO component/class can be directly used by Java
- UNO components can also be developed in Java
- C++ UNO and Java UNO are fully interoperable!

OpenOffice.org
Java, 2



OpenOffice.org – Create a Connection Java, 3

```
XComponentContext xLocalContext =
com.sun.star.comp.helper.Bootstrap.createInitialComponentContext(null);
// initial serviceManager
XMultiComponentFactory xLocalServiceManager = xLocalContext.getServiceManager();
// create a URL resolver
Object urlResolver = xLocalServiceManager.createInstanceWithContext(
    "com.sun.star.bridge.UnoUrlResolver", xLocalContext);
// query for the XUnoUrlResolver interface
XUnoUrlResolver xUrlResolver = (XUnoUrlResolver)
    UnoRuntime.queryInterface(XUnoUrlResolver.class, urlResolver);
// Import the object
Object rInitialObject = xUrlResolver.resolve(
    "uno:socket,host=localhost,port=2002;urp;StarOffice.ServiceManager");
// XComponentContext
if (null != rInitialObject) {
     System.out.println("initial object successfully retrieved");
} else {
     System.out.println("given initial-object name unknown at server side");
}
```

OOo and ooRexx ?

- No direct support for ooRexx in OOo
- No external Rexx functions available for OOo
- BUT
  - **If** there was a way to bridge ooRexx with Java and then use Java to bridge to UNO, **then** it would be **possible** to team OOo with ooRexx!
  - … and there **is** a means available for that:

    BSF4Rexx !

Making Ends Meet
Setting Up, 1

- Install BSF4Rexx
  - Follow the instructions coming with BSF4Rexx
  - Run the supplied test/nutshell programs
- Configure the OOo Java archives
  - Make sure OOo is enabled for Java
    - Check "Tools → Options... → Security → OpenOffice.org → Java → Enable"
  - Add the following OOo "jar"-files (in ...\program\classes) to the environment variable "CLASSPATH"
    - OOo 1.1.x: jurt.jar, unoil.jar, ridl.jar, juh.jar, **sandbox.jar**
    - OOo 2.x: jurt.jar, unoil.jar, ridl.jar, juh.jar

Making Ends Meet
Setting Up for OOo 1.x, 2

- Either
  - Start OOo ("soffice.exe") with the following command line

    ```
    soffice -accept=socket,host=localhost,port=8100;urp;
    ```
- Or
  - Configure OOo to always listen on the given socket and communicating with 'urp' as explained in the OOo Developers Guide, p. 31ff
  - Start one instance of OOo
    - Possible to start an explicit server instance of OOo!

Making Ends Meet
Get the Ball Rolling, 1

- Get in contact with the server and request access to OOo using Java UNO
  - Create a local (client-side) OOo context and get its ServiceManager from it
    - Get a URLResolver service from the local ServiceManager
    - Use the URLResolver service to establish a connection to the server returning the RemoteContext
    - Request the remote ServiceManager from the received RemoteContext

Making Ends Meet
Get the Ball Rolling, 2

- With the help of the remote ServiceManager request the "Desktop" service on the server
  - Of all of the interfaces defined for the "Desktop" service, request the interface "XComponentLoader" allowing the loading (creation) of components (documents)
  - Use the functionality of the XComponentLoader to load (create) an empty text document

Making Ends Meet, An Example, 1

```
/* initialize connection to server, get its Desktop-service and XComponentLoader interface */
CALL BSF.CLS                                               /* get full access to Java using BSF4Rexx */
xComponentContext = .bsf~new("com.sun.star.comp.helper.Bootstrap") -
     ~createInitialComponentContext(.nil)
xUrlResolver = xComponentContext~getServiceManager() -
     ~createInstanceWithContext("com.sun.star.bridge.UnoUrlResolver", xComponentContext)

unoResolverName = .bsf4rexx~Class.class~forName("com.sun.star.bridge.XUnoUrlResolver")
unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")
urlResolver = unoRuntime~queryInterface(unoResolverName, xUrlResolver)

unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.NamingService"
rInitialObject = urlResolver~resolve(unoUrl)
namingServiceName = .bsf4rexx~Class.class~forName("com.sun.star.uno.XNamingService")
rName = unoRuntime~queryInterface(namingServiceName, rInitialObject)

rXsmgr = rName~getRegisteredObject("StarOffice.ServiceManager")
msfName = .bsf4rexx~Class.class~forName("com.sun.star.lang.XMultiServiceFactory")
xMsf = unoRuntime~queryInterface(msfName, rXsmgr)

-- Retrieve the Desktop object, we need its XComponentLoader interface
-- to load a new document
aDesktop = xMsf~createInstance("com.sun.star.frame.Desktop")
xDesktop = .bsf4rexx~Class.class~forName("com.sun.star.frame.XDesktop")
oDesktop = unoRuntime~queryInterface(xDesktop, aDesktop)
xComponentLoaderName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XComponentLoader")
xComponentLoader = unoRuntime~queryInterface(xComponentLoaderName, oDesktop)
```

Making Ends Meet, An Example, 2

```
-- … continued …

/* Open a blank text document    */

/* No properties needed          */
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~bsf.createArray(propertyValueName, 0) /* 0=no elements, i.e. empty Java
array */

/* load an empty text document */
xWriterComponent = xComponentLoader~loadComponentFromURL(    -
          "private:factory/swriter", "_blank", 0, loadProps)
```

file:///c:/docs/aFile.sxw
http://www.RexxLA.org/aFile.sxw
ftp://www.RexxLA.org/aFile.sxw

scalc
swriter
simpress
sdraw

Roundup and Outlook, 1

- OOo
  - Opensource, openplatform
  - UNO, urp
    - C++, Java
  - Client/server architecture
- ooRexx
  - BSF4Rexx as bridge
- Full openplatform control by ooRexx
  - Not restricted to C++, Java, StarBasic or Python!

Roundup and Outlook, 2

- Creating an ooRexx package
  - Simplifying recurring tasks, like establishing a connection with a server
  - Simplifying access to components, e.g. making it easier to manipulate cells of the spreadsheet
- With the advent of OOo 2.0
  - Devised a plug-in for BSF4Rexx, allowing ooRexx to be dispatched from within OOo
  - Makes it possible to use ooRexx wherever StarBasic is used!

OpenOffice.org – An example using Java 1:1

```
/* initialize connection to server, get its Desktop-service and XComponentLoader interface */
sComponentContext = .bsf~new("com.sun.star.comp.helper.Bootstrap") ~createInitialComponentContext(.nil)
unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")

sUrlResolver = sComponentContext~getServiceManager() ~createInstanceWithContext("com.sun.star.bridge.UnoUrlResolver", sComponentContext)
XUnoUrlResolver = .bsf4rexx~Class.class~forName("com.sun.star.bridge.XUnoUrlResolver")
oUrlResolver = unoRuntime~queryInterface(XUnoUrlResolver, sUrlResolver)

unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.NamingService"
oInitialObject = oUrlResolver~resolve(unoUrl)
XNamingService = .bsf4rexx~Class.class~forName("com.sun.star.uno.XNamingService")
sNamingService = unoRuntime~queryInterface(XNamingService, oInitialObject)

oServiceManager = sNamingService~getRegisteredObject("StarOffice.ServiceManager")
XMSFactory = .bsf4rexx~Class.class~forName("com.sun.star.lang.XMultiServiceFactory")
sMSFactory = unoRuntime~queryInterface(XMSFactory, oServiceManager)

-- Retrieve the Desktop object, we need its XComponentLoader interface
-- to load a new document
sDesktop = sMSFactory~createInstance("com.sun.star.frame.Desktop")
XDesktop = .bsf4rexx~Class.class~forName("com.sun.star.frame.XDesktop")
oDesktop = unoRuntime~queryInterface(XDesktop, sDesktop)
XComponentLoaderName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XComponentLoader")
sComponentLoader = unoRuntime~queryInterface(XComponentLoaderName, oDesktop)

/* Open a blank text document   */
/* No properties needed         */
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0) /* 0=no elements, i.e. empty Java array */
/* load an empty text document */
oWriterComponent = sComponentLoader~loadComponentFromURL("private:factory/swriter", "_blank", 0, loadProps)
::requires BSF.CLS
```

OpenOffice.org, Summer Semester 2005 "OOo.cls"

- "OOo.cls"
  - Initializing OOo a recurrent issue
    - Load off the needed statements
  - Support an OOo-proxy
    - Makes it easy to get XInterfaces from the objects
    - Works closely with BSF
      - Wraps up BSF proxies
  - Eases coding of OOo considerably

OpenOffice.org – An example using „OOo.CLS“

```
/* initialize connection to server, get its Desktop-service and XComponentLoader interface */
xMsf=ooo.connect()   -- connect to server and retrieve remote multi server factory

-- Retrieve the Desktop object, we need its XComponentLoader interface
-- to load a new document
oDesktop          = xMsf~createInstance("com.sun.star.frame.Desktop")
xDesktop          = oDesktop~XDesktop          -- get desktop interface
xComponentLoader = xDesktop~XComponentLoader  -- get componentLoader interface

/* load an empty text document */
xWriterComponent = xComponentLoader~loadComponentFromURL("private:factory/swriter", "_blank", 0, .OOo~noProps)

::requires OOo.cls    -- get OOo support
```

OpenOffice.org, Wintersemester 2005/06 "UNO.CLS"

- "UNO.CLS"
  - Builds on the experiences made with "OOo.cls"
  - Supercedes (replaces) "OOo.CLS"
  - Generalizes interaction at the granular level of "UNO" service objects

  → ***No need to individually set up an OOo installation to listen at a specific port!***
- Public routines for reflection (also methods of UNO proxy objects), e.g.
  - uno.findInterfaceWithMember(object, memberName, bString, iMatches)
  - uno.getDefinition(object)
  - uno.getProperties(object)
  - uno.getTypeName(object)
  - uno.getXTypeProviderTypeNames(object)
  - uno.queryInterfaceName(object, name)
  - uno.queryServiceName(object, name)
  - ...
- Research "UNO.CLS" for addtional public routines yourself!

OpenOffice.org – An example using "UNO.CLS" "swriter" (Word Processor Module)

```
oDesktop          = UNO.createDesktop()    -- get the OOo Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface

     /* open the blank *.sxw - file */
xWriterComponent = xComponentLoader~loadComponentFromURL("private:factory/swriter", "_blank", 0, .UNO~noProps)
::requires UNO.CLS    -- get UNO support
```

OpenOffice.org – An example using "UNO.CLS" "scalc" (Spreadsheet Module)

```
-- Ahammer: Example 14
-- a new document
oDesktop          = UNO.createDesktop()    -- get the UNO Desktop service object
xComponentLoader = oDesktop~XDesktop~XComponentLoader  -- get componentLoader interface
url = "private:factory/scalc"
xCalcComponent = xComponentLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
/* get first sheet in spreadsheet */
xSheet = xCalcComponent~XSpreadSheetDocument~getSheets~XIndexAccess~getByIndex(0) ~XSpreadSheet
/* insert some text */
CALL UNO.setCell xSheet, 0, 0, "1"                   -- cell "A1"
CALL UNO.setCell xSheet, 1, 0, "=(A1*3)"             -- cell "B1"
CALL UNO.setCell xSheet, 2, 0, "=($A$1*10*RAND())" -- cell "C1"
CALL UNO.setCell xSheet, 3, 0, "1"                   -- cell "D1"
/* and AutoFill it */
to_bottom = bsf.getConstant("com.sun.star.sheet.FillDirection", "TO_BOTTOM")
getCellSeries(xSheet, "A1:C10")~fillAuto(to_bottom, 1)
getCellSeries(xSheet, "D1:D10")~fillAuto(to_bottom, 2)
/* save the result - we need it for the next example */
storeURL = makeURL("testnumbers.sxc")  -- save the document in the current folder
xCalcComponent~XStorable~storeAsURL(storeURL, .UNO~noProps)

::requires UNO.CLS     -- get UNO support
::routine getCellSeries
  use arg xSheet, aRange
  return xSheet~XCellRange~getCellRangeByName(aRange)~XCellSeries
::routine makeUrl -- A function for getting the file in the current folder
  return  UNO.ConvertToURL(directory() || "\" || arg(1))
```

Roundup and Outlook

- UNO.CLS
  - Needs BSF4Rexx
  - Full control over Open Office
  - Eases programming considerably
    - Making it easy to request interface objects
- You can directly apply all OOo information
  - StarBasic documentation, books
  - UNO documentation, books for C++, Java
- With OOo v2.0
  - Possible to get ooRexx to be usable from within OOo

Überblick, Aufgabenstellungen

- Erstellen von einfachen Beispielen
  - BSF4Rexx-Beispiel
    - Erstellen Sie ein einfaches Beispiel, das Java's awt/swing benutzt
  - OpenOffice.org 2.4.0 (OOo, Stand: März 2008)
    - Automatisierungsmöglichkeiten, z.B. über Java-Bindings
      - http://udk.openoffice.org/
      - WU-Wien
        - Schiseminar WS 2004/05, Seminar SS 2005, Schiseminar WS 2005/06, Seminar SS 2006, Schiseminar 2006/07, Seminar SS 2007, Seminar WS 2007/08
        - Bakk-Arbeiten, vgl. **http://wi.wu-wien.ac.at/rgf/diplomarbeiten/**
    - Einbinden als Makrosprache ins OOo
    - Je ein Beispiel für swriter, scalc, simpress von außerhalb und innerhalb von OOo