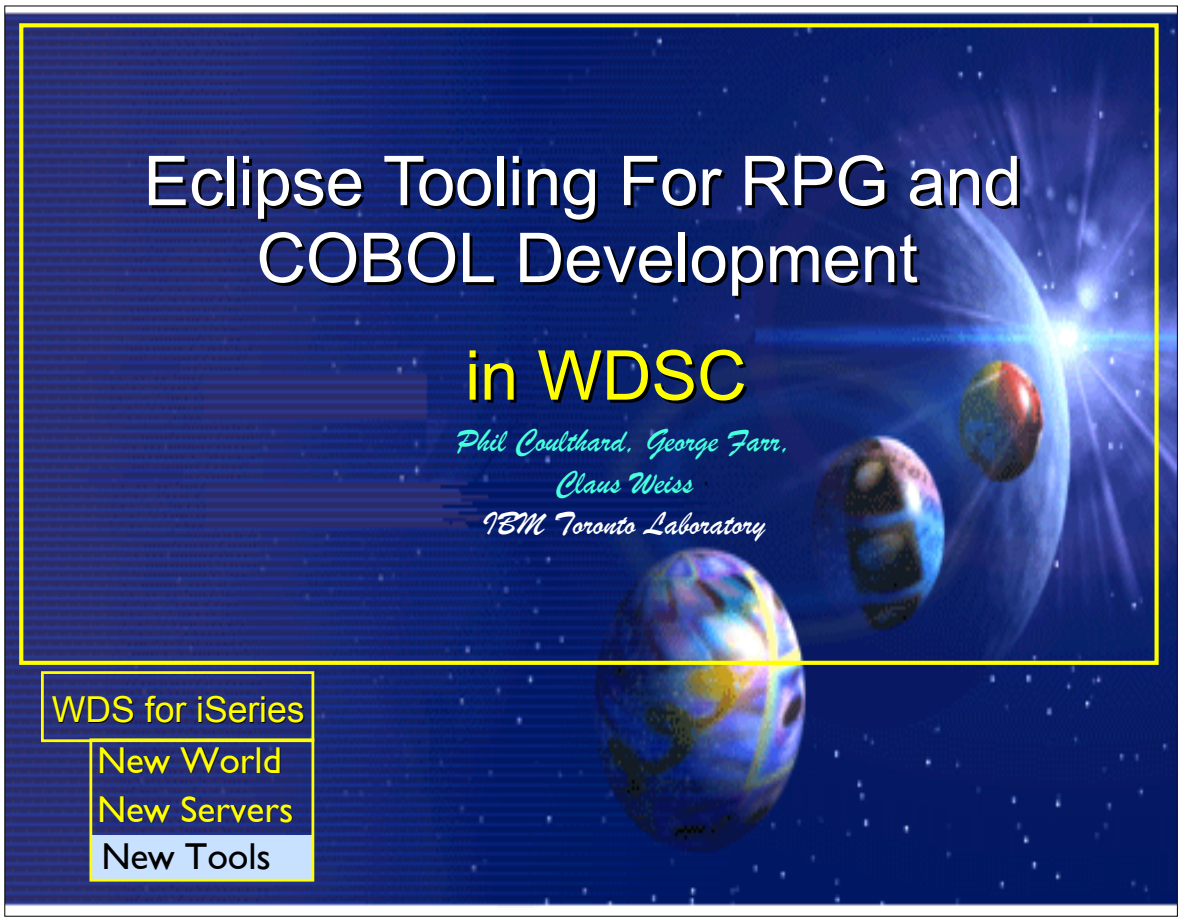


- This presentation covers the RPG and COBOL iSeries specific tools (Remote Systems Explorer and iSeries Project perspective) within the Development Studio Client workbench or IDE.

Disclaimer

Acknowledgement:

- This presentation is a collaborative effort of the IBM Toronto AS/400 Application Development presentation team, including work done by:
  - *Phil Coulthard, George Farr, Claus Weiss, Don Yantzi, David Slater, Alison Butteril, Linda Cole*

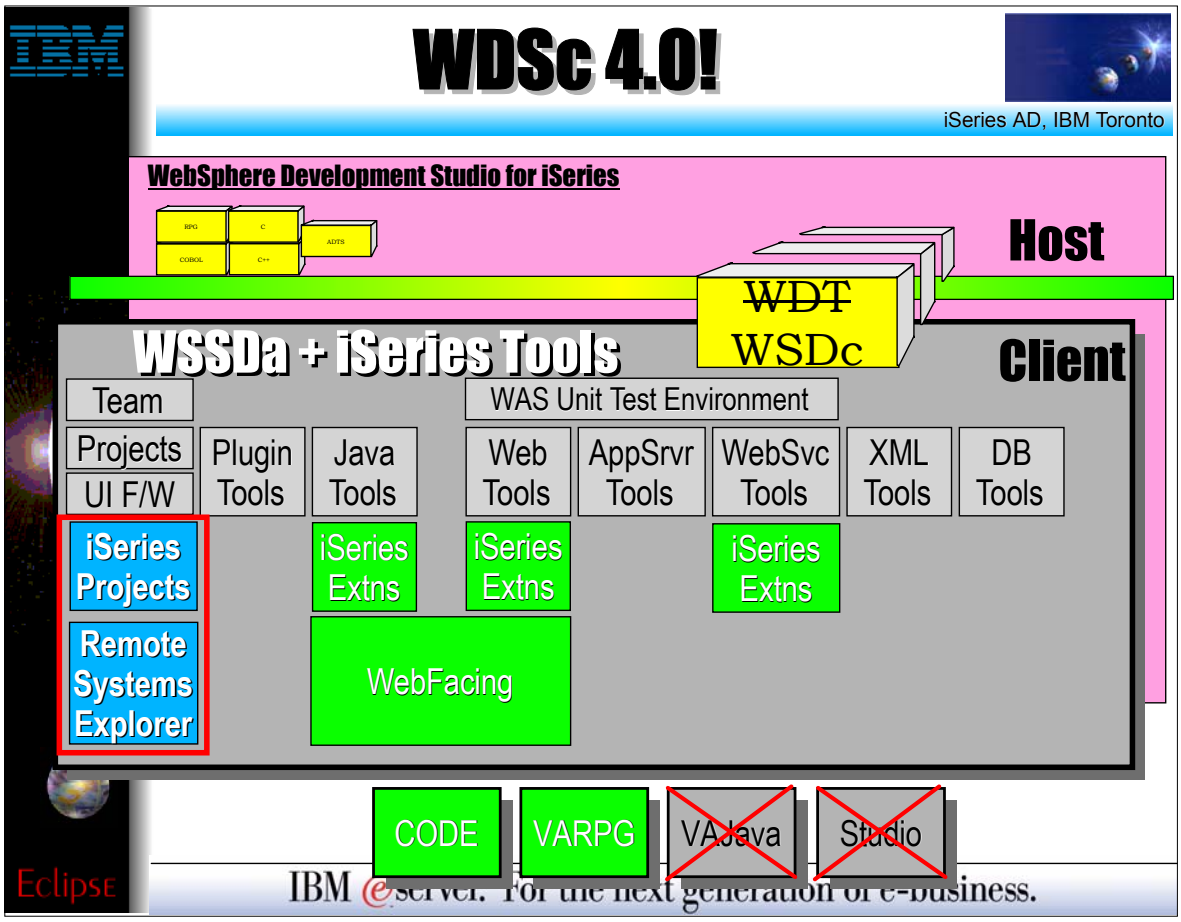
Disclaimer:

- The information contained in this document has not been submitted to any formal IBM test and is distributed on an as is basis without any warranty either express or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customers' ability to evaluate and integrate them into the customers' operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

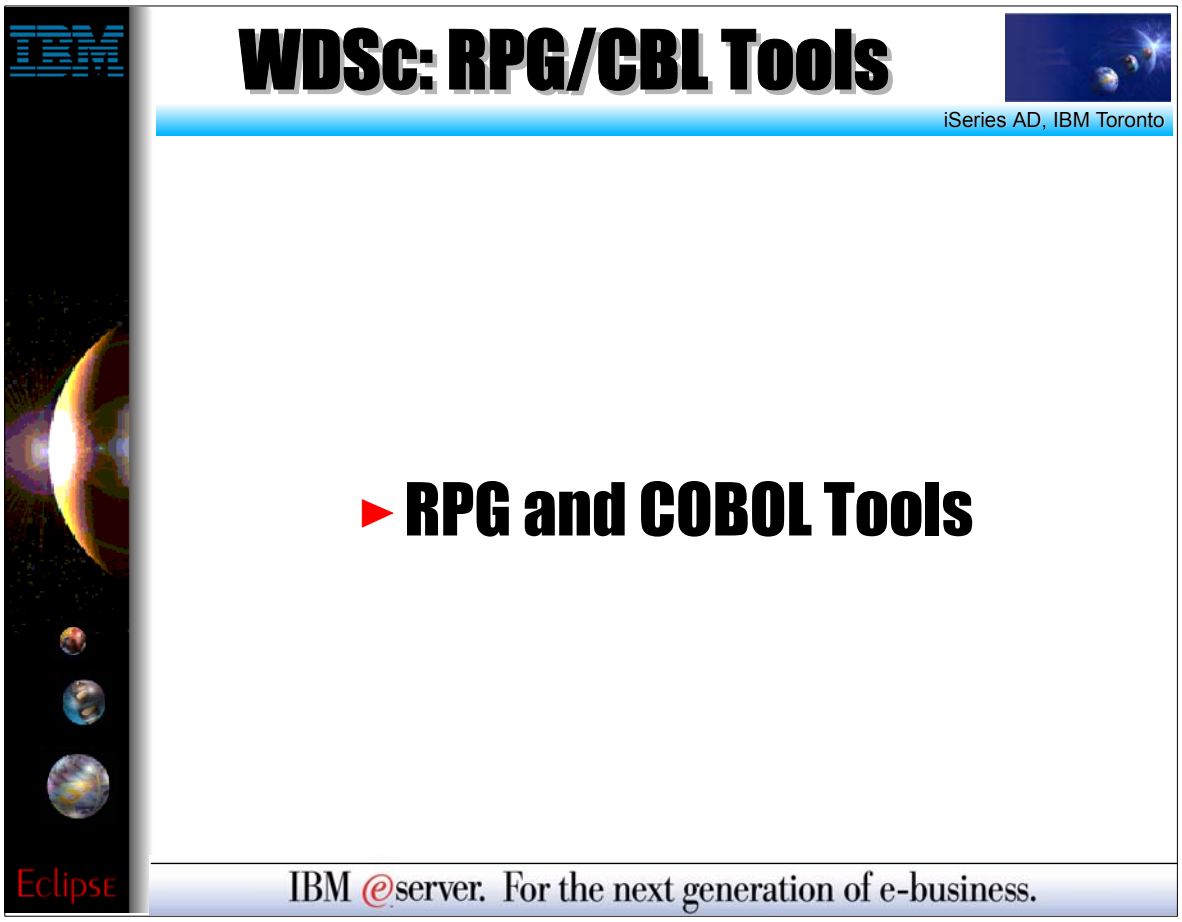
Reproduction:

- The base presentation is the property of IBM Corporation. Permission must be obtained PRIOR to making copies of this material for any reason.

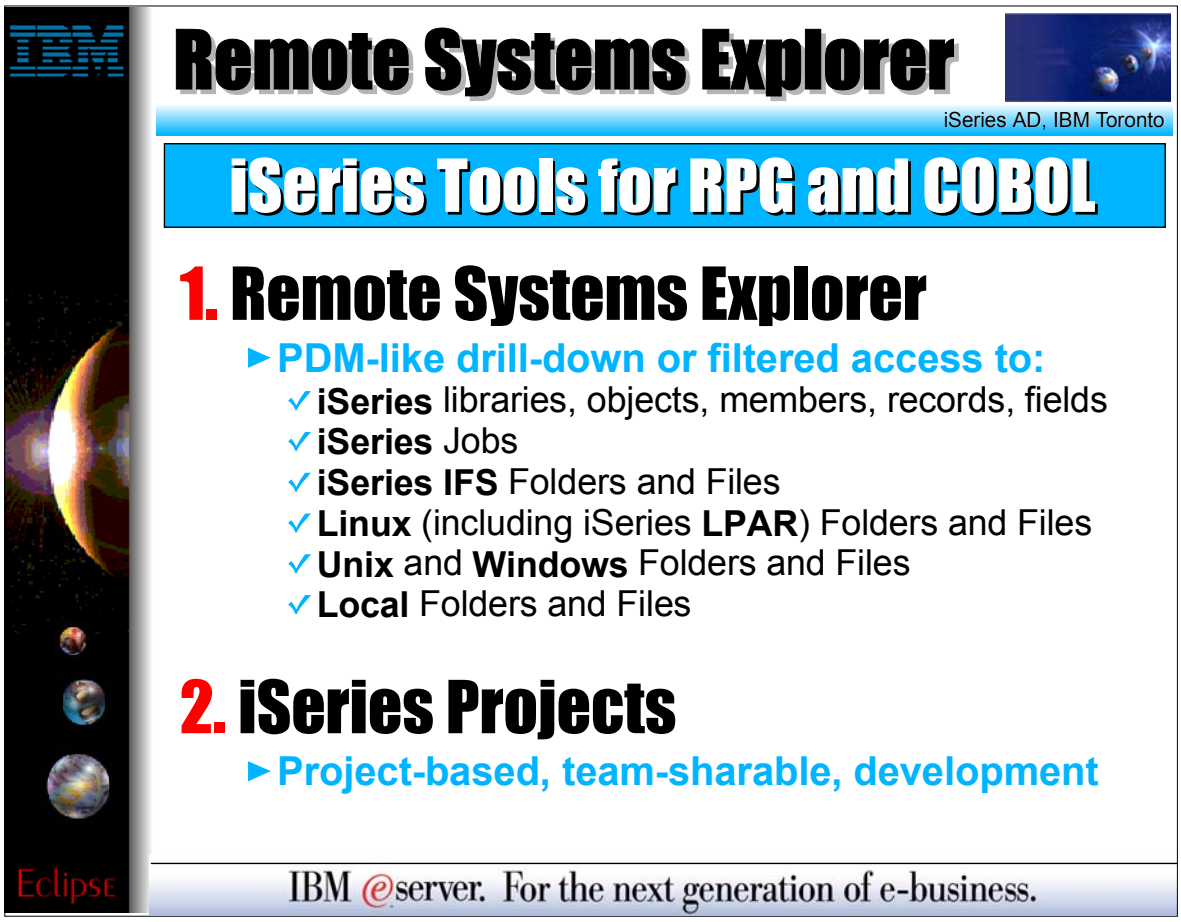
- This presentation is a collaborative effort from the very team that brings you WDSc!



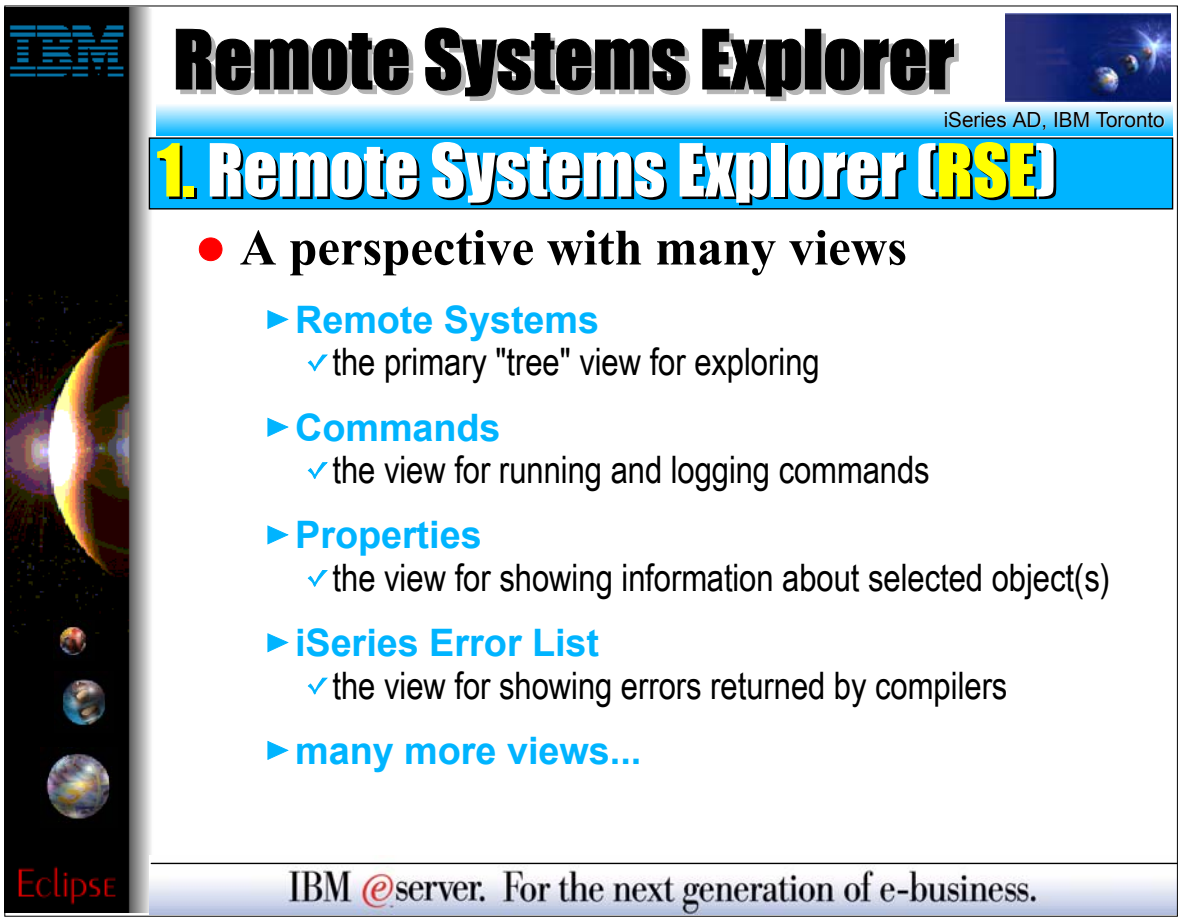
- This is the new release of WDSc that was announced and shipped in June 2002
- WebSphere Development Studio (WDS), as stated, is the overall umbrella product which includes the four host compilers and classic tools (ADTS), and unlimited licenses of the client tools previously known as WebSphere Development Tools for iSeries (WDT)
- As stated, WebSphere Development Studio Client 4.0 is the next release of WebSphere Development Tools V5R1
- There was no new release of the overall WDS product in June, only the client tools. These are only available for V5R1 or above, although they mostly do work with a V4R5 system
- Customers who have WDS are entitled to WDSc for free, but they must order it. This is done by re-ordering WDS and specifying feature code 2655.
- The overall WDS product itself is refreshed for V5R2, at which time the four host compilers are updated with new enhancements as is typical for a new release
- WDSc is a superset of the WebSphere Studio Site Developer Advanced product (WSSDa).
- As stated previously: WSSDa includes all of the Eclipse support for Team, Projects, User Interface Framework, plugin development and Java development. Further it also adds support for built-in WebSphere Application Server, for Web development, for remote application servers, for Web Services, for XML development and for Database development.
- To the WSSDa base, WDSc adds support for iSeries projects and exploration of remoteystems, as well as adding iSeries extensions to the Java Tools, Web Tools and Web Services tools. It also includes WebFacing, which builds on the Java and Web tools. All of these tools will be briefly described in the following slides.
- The classic CODE and VARPG products are also included with WDSc, but not so the classic VisualAge for Java and WebSphere Studio products. These last two have been subsumed by WSSDa and hence WDSc



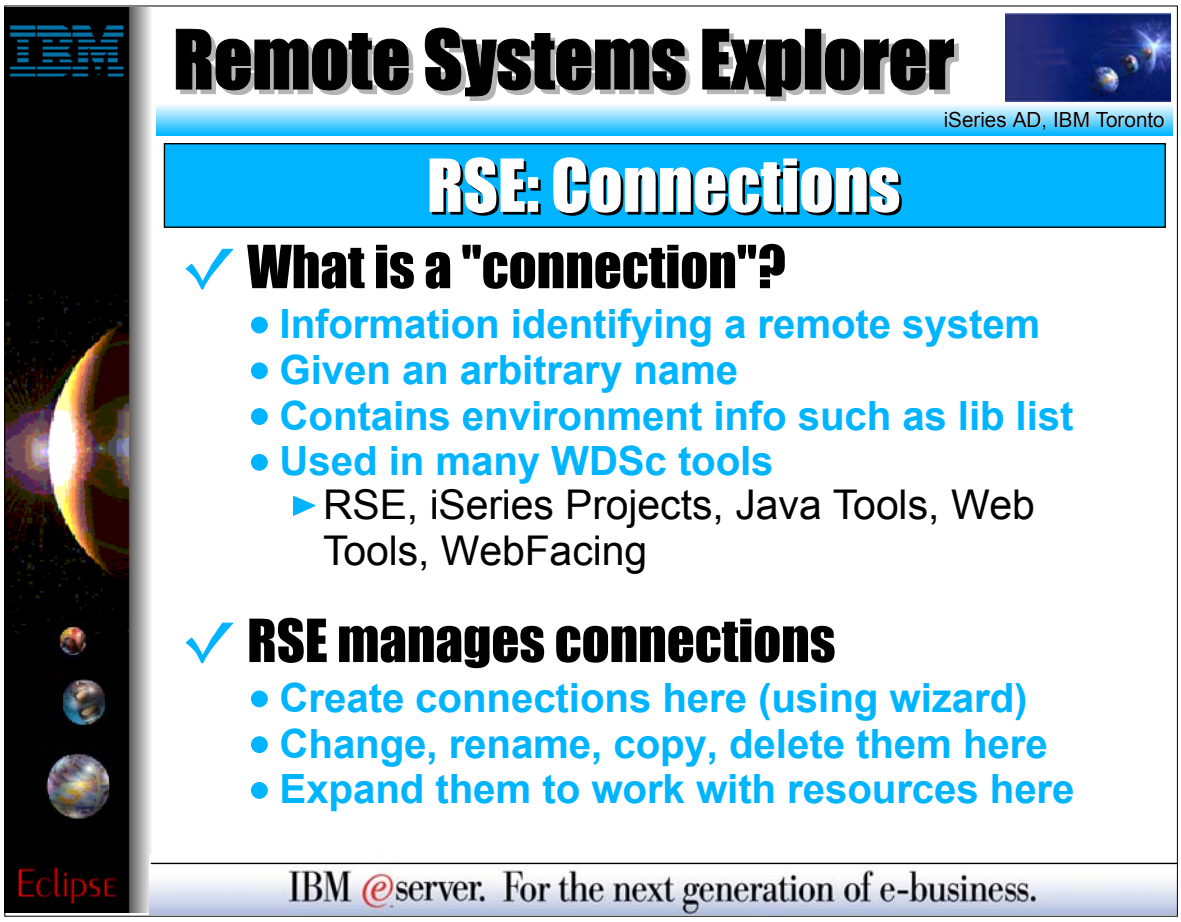
- Now we cover the RPG and COBOL tools within the WDSC IDE



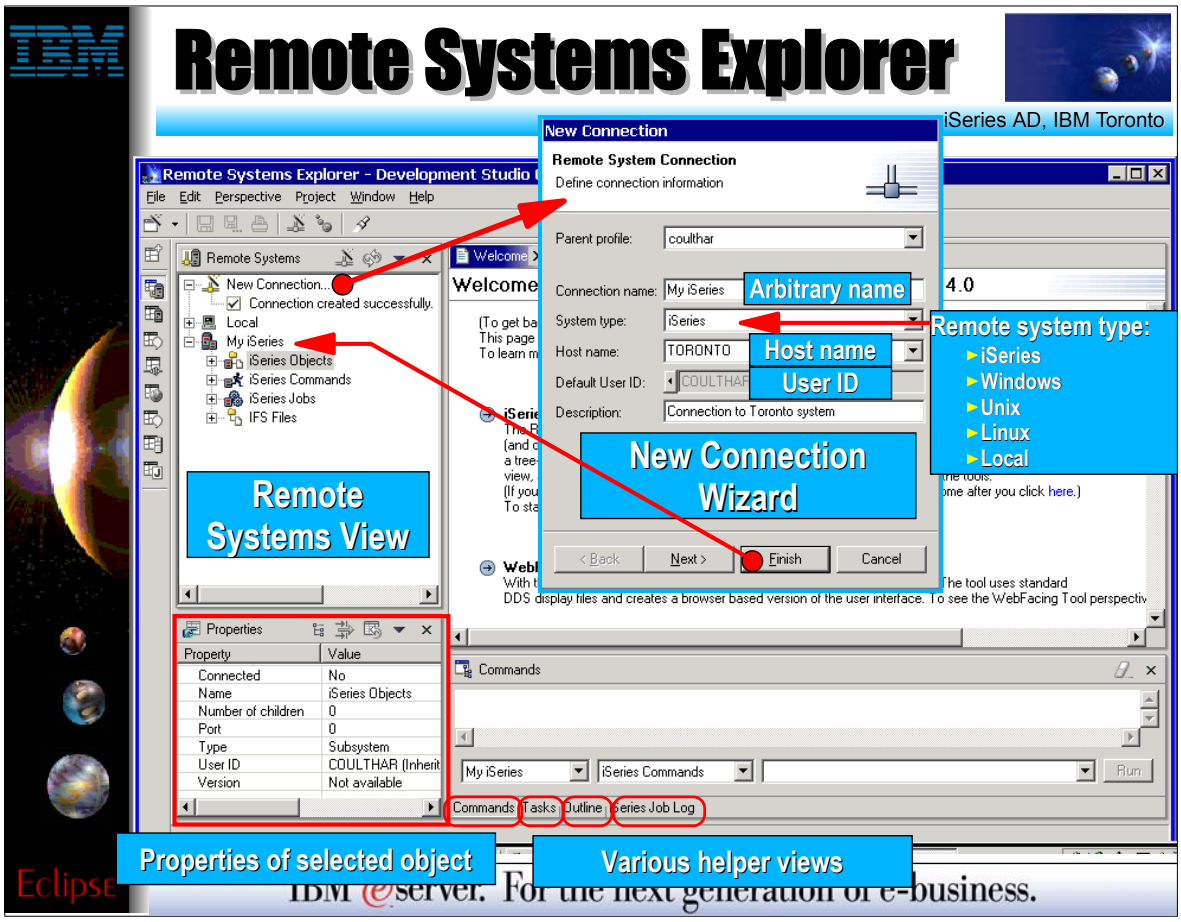
- There are two explicit areas of functionality, each with their own perspective, for iSeries programmers
- The first area of functionality is the **Remote Systems Explorer**, which has its own perspective and views
- This first area of functionality is similar to PDM in that it allows the developer to drill down into the QSYS file system, or use filters to list specific objects within the QSYS file system
- The Remote Systems Explorer goes well beyond PDM however! It also allows exploration of iSeries jobs and commands, and the IFS file system. Further, it can also be used to explore the file system of remote Linux, Unix and Windows systems. The Linux support works for any Linux, including Linux in an iSeries Logical Partition.
- The second area of functionality is iSeries Projects, which also has its own perspective and views.
- Unlike the Remote Systems Explorer, an iSeries project fully leverages the Eclipse support for resources. This means an iSeries project contains folders and files that exist in the local file system, yet can be shared and synchronized among a team, if an eclipse-compliant repository is used. The options for this today include Rational's ClearCase or the free and open-source product CVS or Concurrent Versioning Systems, or the MKS Integrity. CVS is available on the distribution CDs for the Linux LPAR on iSeries. In the near future, the remaining iSeries change management vendors will also be available as repository options.
- As you will see, an iSeries project allows RPG and COBOL developers to fully exploit the power Eclipse, while developing applications targeted to run on iSeries.
- We will explore each of these tools...



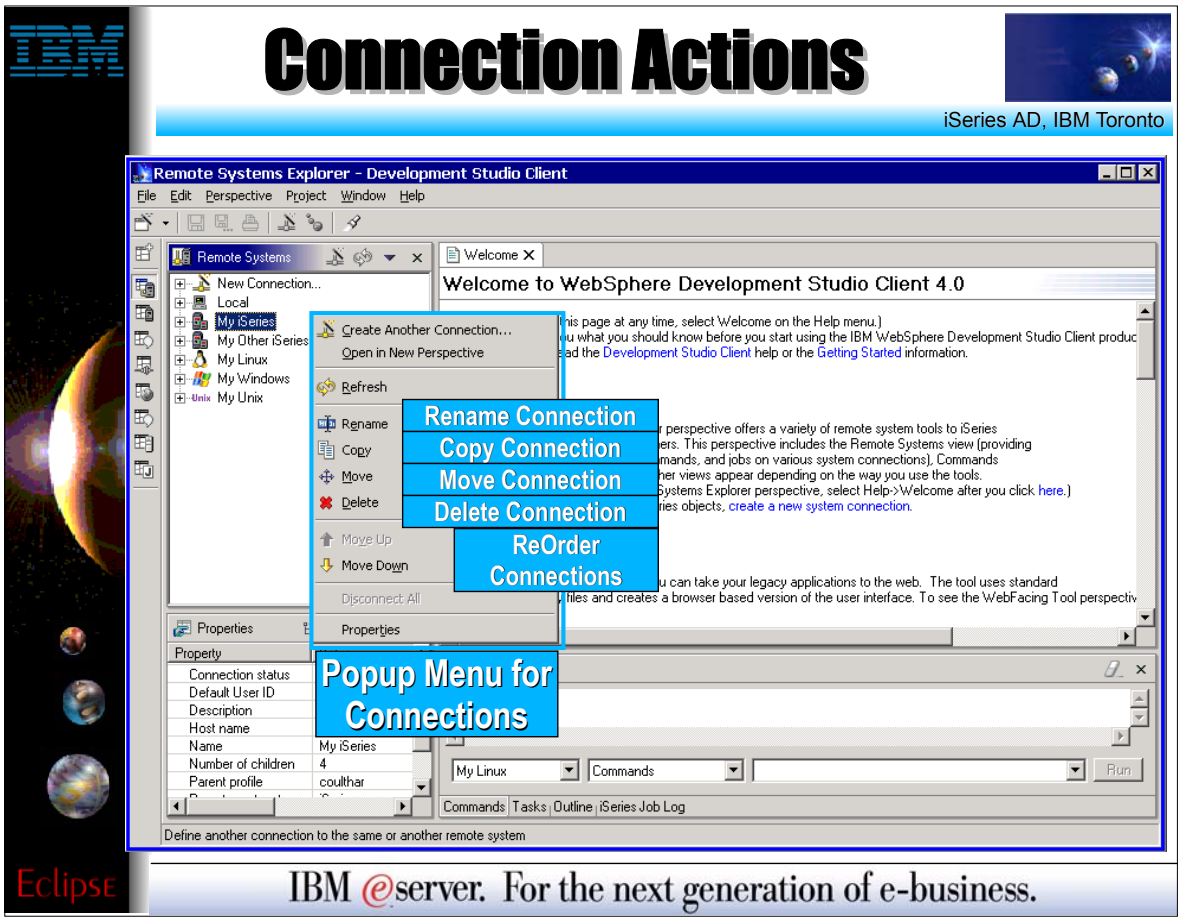
- The Remote System Explorer is an eclipse perspective with many views to help developers work with resources in a remote system
- The primary view is the Remote Systems tree view, where remote system resources are explored, similar to the Windows Explorer
- There is also a commands view, or shell, for entering commands to be run remotely, and for logging the results of all commands
- There is also a Properties view, which is common throughout Eclipse. It shows interesting information about the object currently selected in the primary view
- There is an iSeries Error List view where errors are shown after performing a remote compile
- There are many other views which will appear when requested via popup menu actions



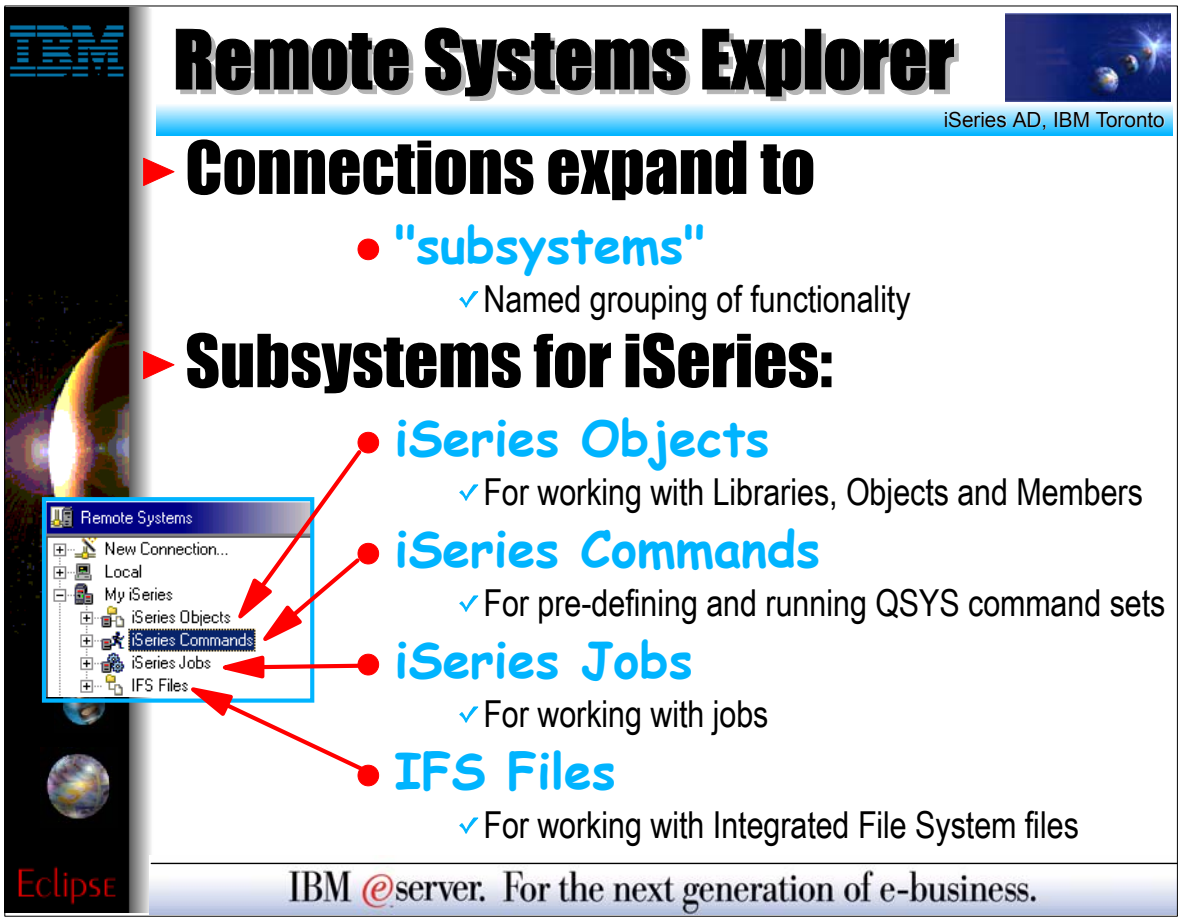
- A very central concept to all of WDSc is that of connections.
- A connection defines information needed to access a remote system. Each connection is given an arbitrary name by the user, and so multiple connections to the same system are permitted
- Each connection also captures information that is applied when connecting to that remote system, such as the initial library list for iSeries connections
- All iSeries tools within WDSc use connections to access a remote iSeries system
- Connections are created and managed in the Remote Systems Explorer
- Further, the Remote Systems Explorer is also used to explore objects in a remote system, by expanding a connection



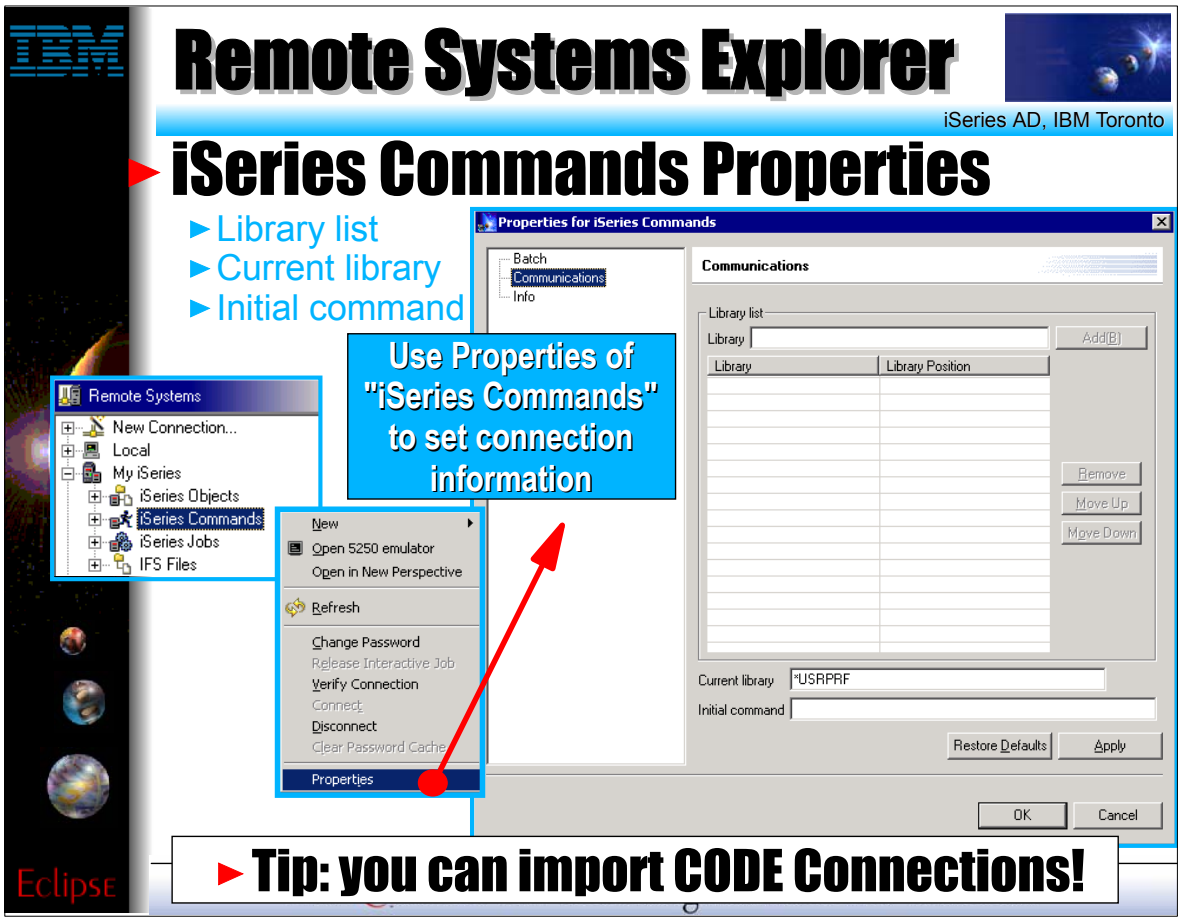
- This is the Remote Systems Explorer perspective. This is the first perspective you see when starting WebSphere Development Studio Client for the first time.
- The Remote Systems view lists your existing connections, and contains an item that when expanded launches the New Connection wizard for creating a new connection
- The New Connection wizard prompts for information about the connection, including an arbitrary but unique name, the type of the remote system, the TCP/IP hostname for the system, the User ID to connect to the system with, and optionally a description of the connection
- Once created, a connection is listed in the tree and can be expanded to explore the contents of that remote system, which we will describe soon
- The Remote Systems Explorer contains other views of interest, including the Properties view in the lower left, and various other views in the lower right that are in a tabbed notebook
- Like all views in Eclipse, these views can be moved to a different location by dragging and dropping them



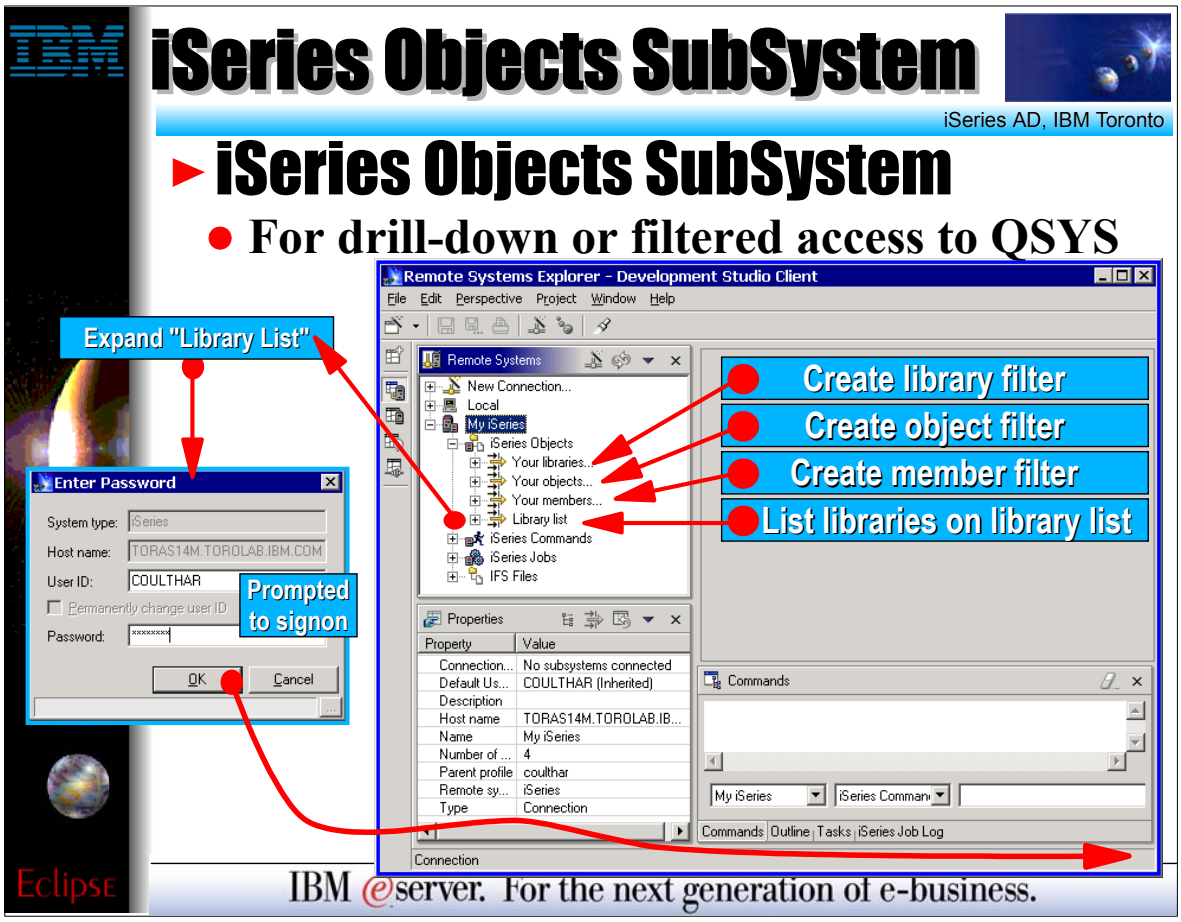
- Connections support actions in their popup menu for manipulating them.
- These include:
- **Rename** Connection. For giving the connection a new name
- **Copy** Connection. For creating a new connection based on this connection. You will be prompted for a new name. You can also copy a connection to another profile. We will cover profiles later.
- **Move** Connection. For moving a connection to another profile. We will cover profiles later.
- **Delete** Connection. For deleting a connection.
- **Move Up** Connection. For moving the selected connections up in the connection list.
- **Move Down** Connection. For moving the selected connections down in the connection list.
- You can also change the attributes of a connection, such as the host name or user Id, either by selecting the Properties popup menu item, or by editing the attributes in the Properties view (be sure to press Enter!)



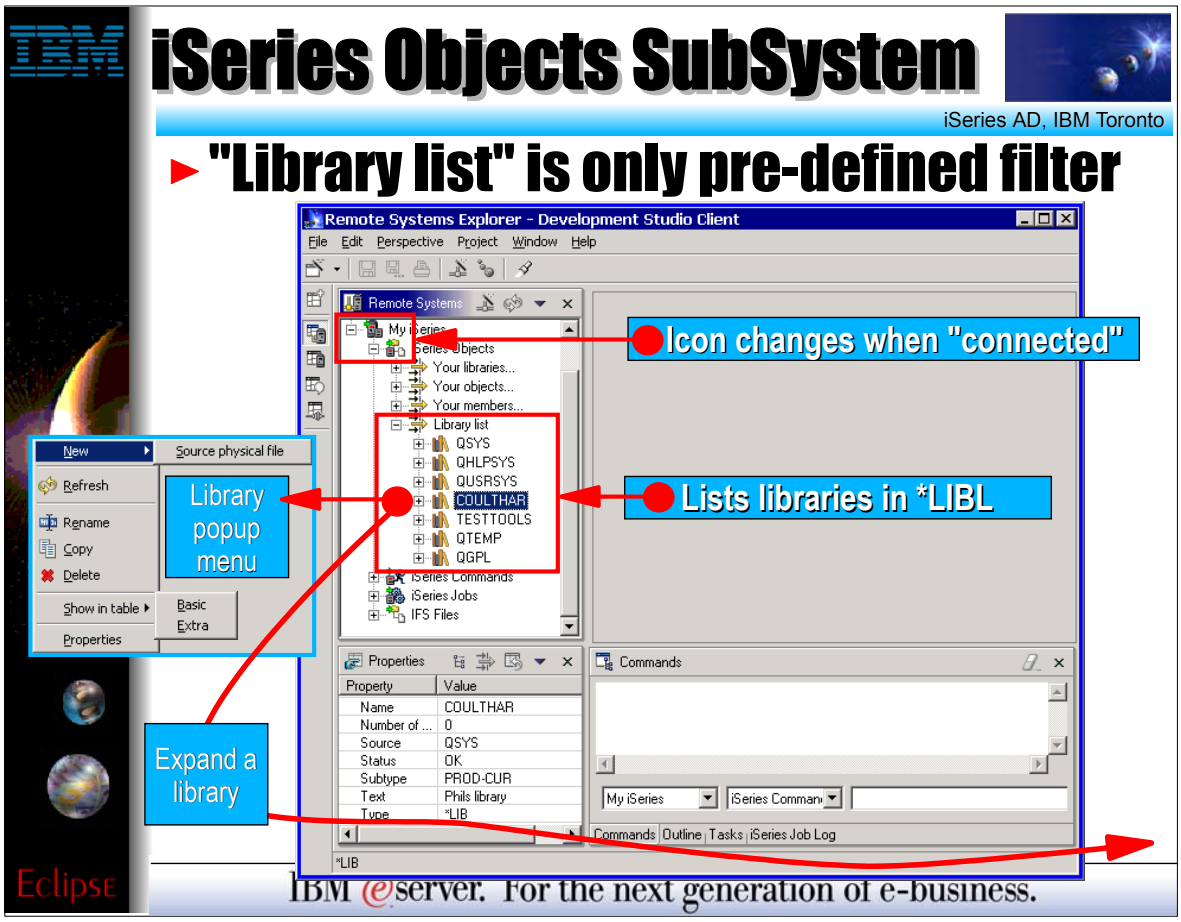
- Once connections are defined they can be expanded within the Remote Systems Explorer
- On expansion, the user sees subsystems, which are merely a functional grouping of the various types of remote resources that can be explored in the remote system
- For iSeries connections, there are four subsystems:
- 1. iSeries Objects is the PDM-like grouping, allowing access to libraries, objects and members
- 2. iSeries Commands allows developers to predefine command sets each of which contain one or more often used commands. When run, all commands in a command set are sent to the remote system and executed, and the results are logged in the Commands view
- 3. iSeries Jobs allows developers to see various jobs, subsettable by job attributes, and to perform a limited number of operations on those jobs
- 4. IFS Files allows developers to explore folders and files in the Integrated File System of the remote iSeries system



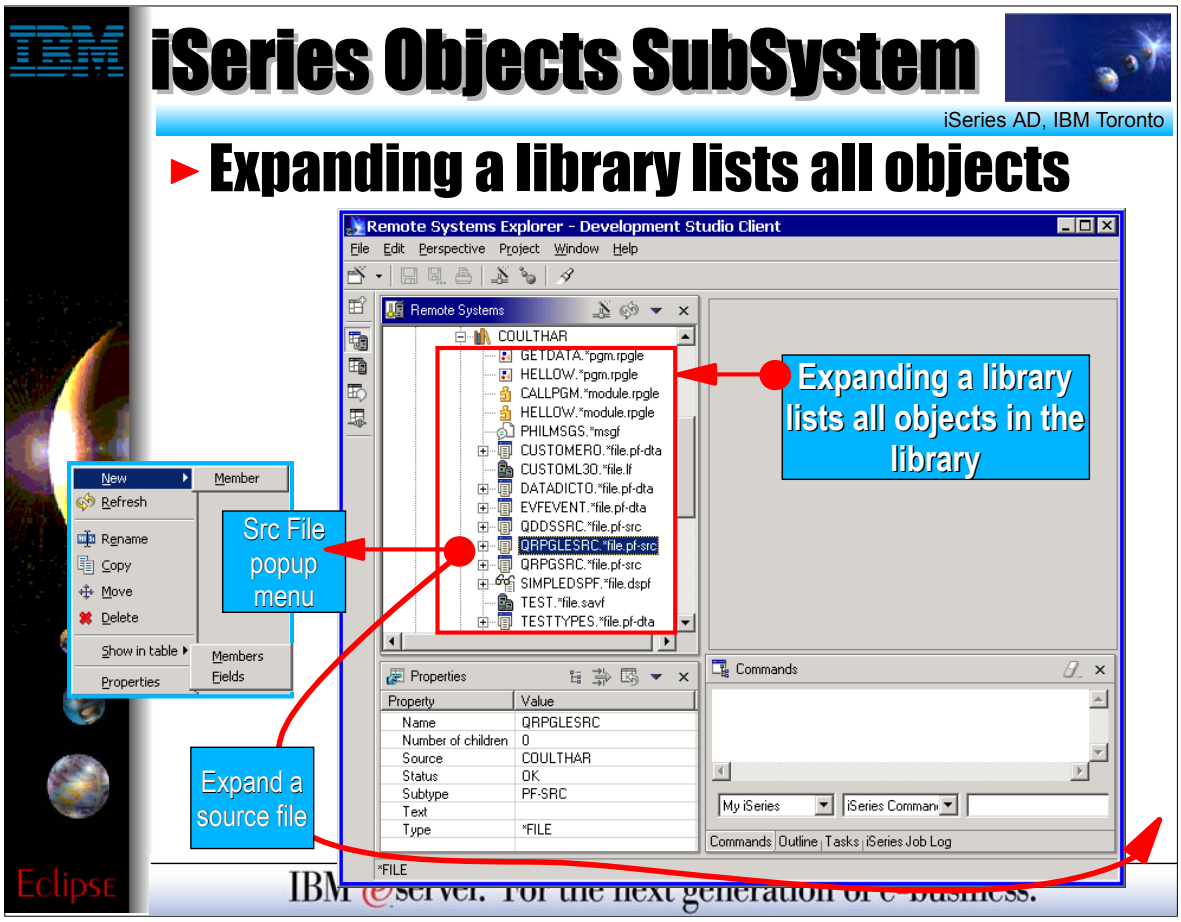
- When a connection is used to connect to a remote iSeries, your initial program as specified in your user ID is not honored. You can overcome this by using the Properties popup menu item from a selected iSeries Commands subsystem under a connnection.
- On this Properties dialog you can specify libraries to **add to the library list**, **specify a current library**, and **specify an initial command to run**. The initial command must not be interactive!
- When the connection is used to connect to the iSeries, the RSE will execute the appropriate ADDLIBLE, CHGCURLIB commands, and call your initial command.



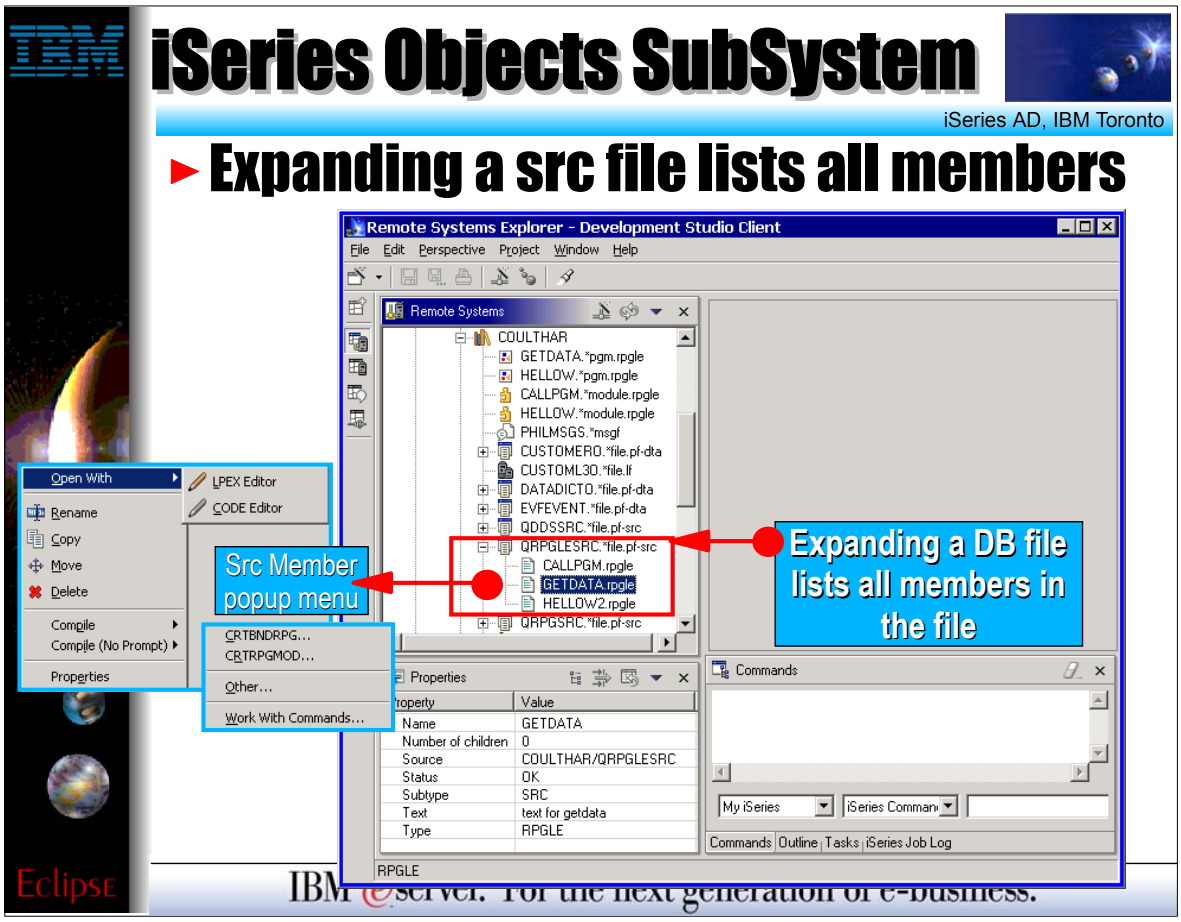
- We now drill down into the iSeries Objects subsystem. This is the subsystem you will use most often! It is very similar to PDM, in that it allows you to access objects in the QSYS file system, and perform actions on those objects.
- The three child items at the top of the list are for creating filters, much like in PDM:
- **Your libraries...** prompts you for a simple or generic library name, and lists all matching libraries. It is similar to **WRKLIBPDM**.
- **Your objects...** prompts you for a simple or generic library name and simple or generic object name, as well one or more object type and attribute pairs. It lists all matching objects in all matching libraries. it is similar to **WRKOBJPDM**.
- **Your members...** prompts you for a simple or generic library name, simple or generic file name, and simple or generic member name, as well as one or more member types which can also be generic. It lists all matching members in all matching files in all matching libraries. It is similar to **WRKMBRPDM**.
- Unlike PDM, the filters you create are permanently remembered and displayed in this list for easy re-use. We will have more to say about filters.
- To simulate STRPDM's option 12, you can start with the pre-defined **Library list** filter, that when expanded lists all libraries in your library list.
- With any filter, once it is expanded you can subsequently expand a library to see all objects in the library, and expand files to see all members in the file.
- When you expand your first filter, such as the pre-defined Library List filter, you are prompted for your password and then connected to the remote iSeries. Then, the results of resolving the filter are shown...



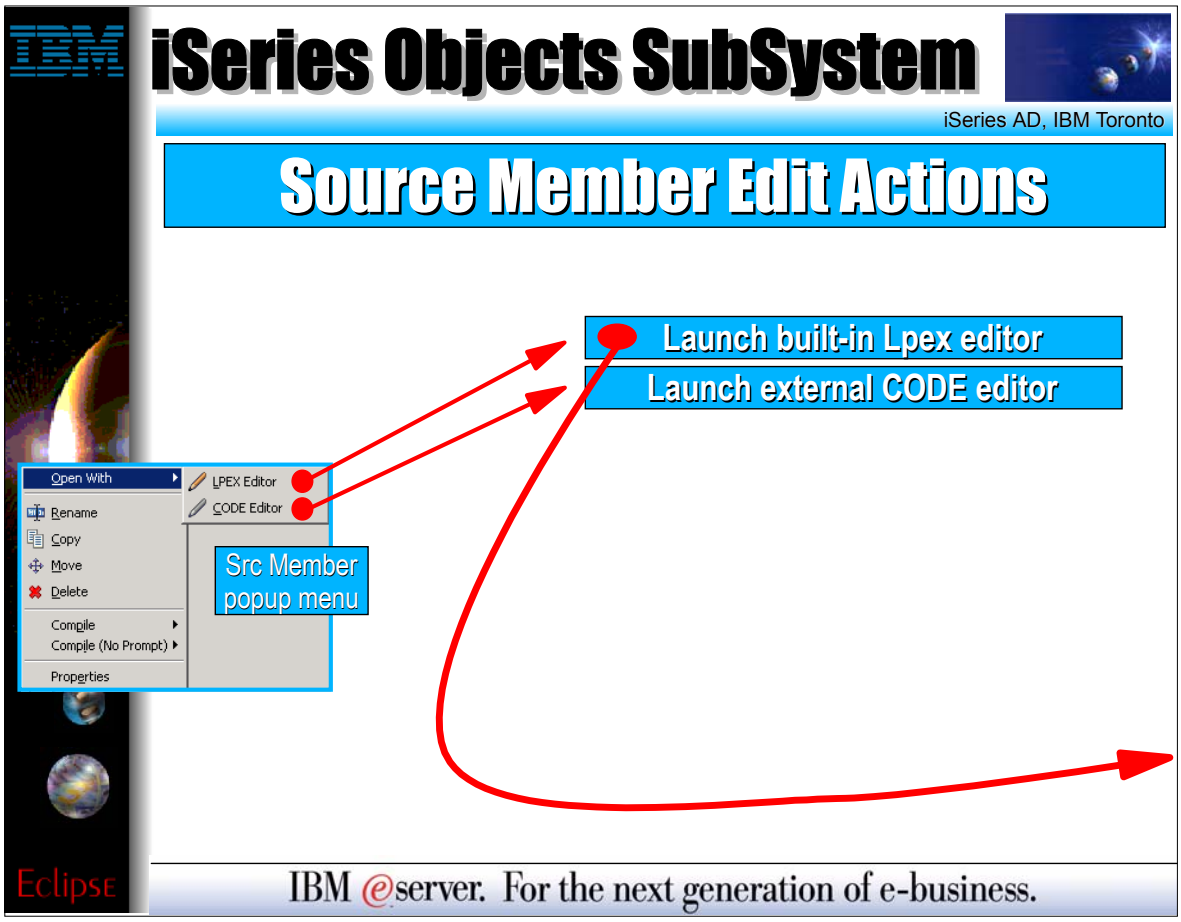
- When you are connected to the remote iSeries, the icons for the connection and its subsystems change to include a small green arrow, so as to indicate the connection status.
- When the pre-defined library list filter is expanded, and the connection is successful, you will see the libraries on your library list.
- For each library, you can right-click and select from a number of useful actions. There is an action to create a new source file within the selected library, to refresh the contents of the library if it is expanded, to rename the library, copy the library or delete the library. These last three actions remotely run the appropriate iSeries command and you will see it logged in the Commands view. There are also actions to open a multiple-column table view showing the contents of the library, similar again to PDM. There are two levels of details to choose from for the table: Basic and Extra. Extra includes additional columns of information, but will take a bit longer to present.
- If you expand a library, you will see all the objects in that library...



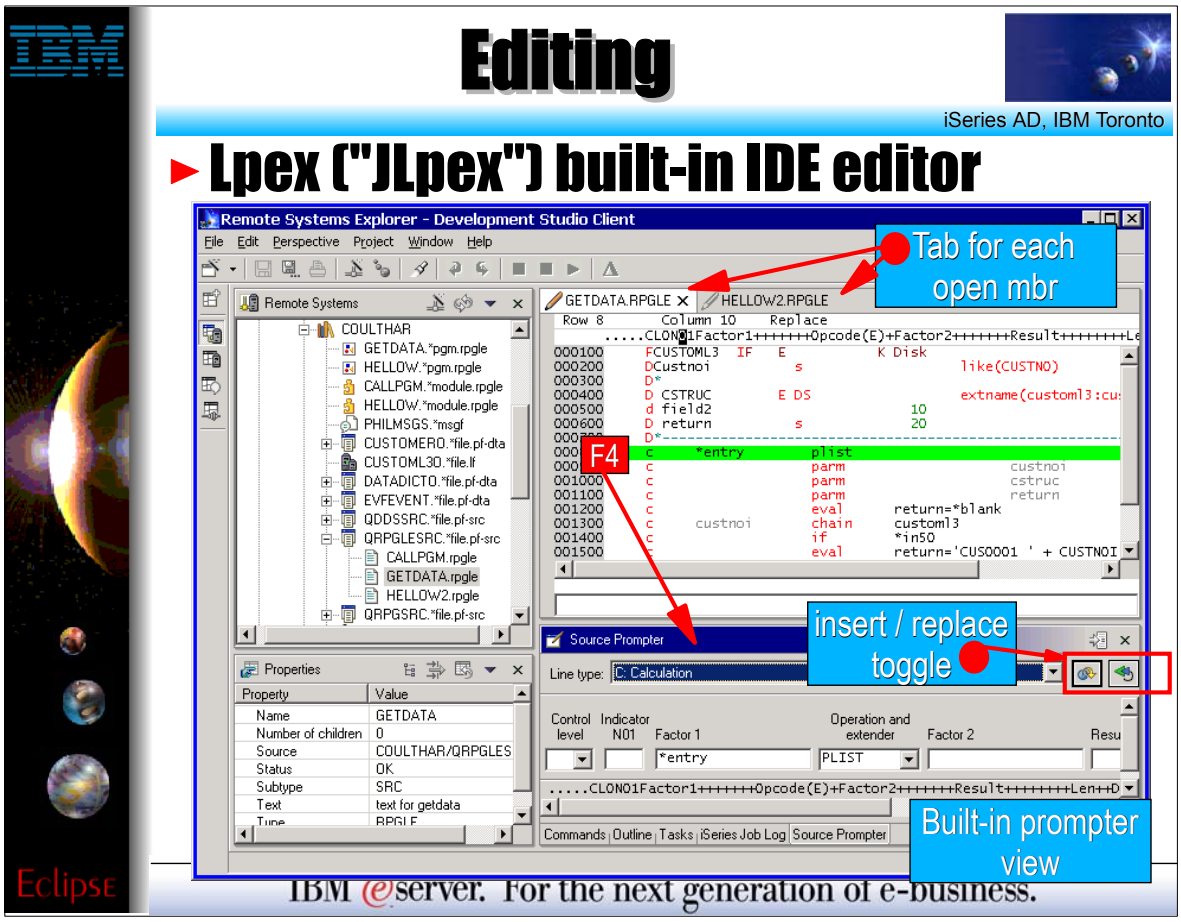
- When a library is expanded in the Remote Systems tree view, all the objects within that library are listed underneath the library.
- For each object, you can right-click and select from a number of useful actions. The exact list of actions will depend on the type of object you select, and whether you selected one or multiple objects.
- For a source file, the popup menu has an action to create a new member within the selected file, to refresh the contents of the file if it is expanded, to rename the file, copy the file, move the file and delete the file. These actions remotely run the appropriate iSeries command and you will see it logged in the Commands view. For both data and source files, there are also actions to open a multiple-column table view showing the contents of the file. You can choose to list the members, similar to PDM, or list the fields.
- If you expand a file, you will see all the members in that file...



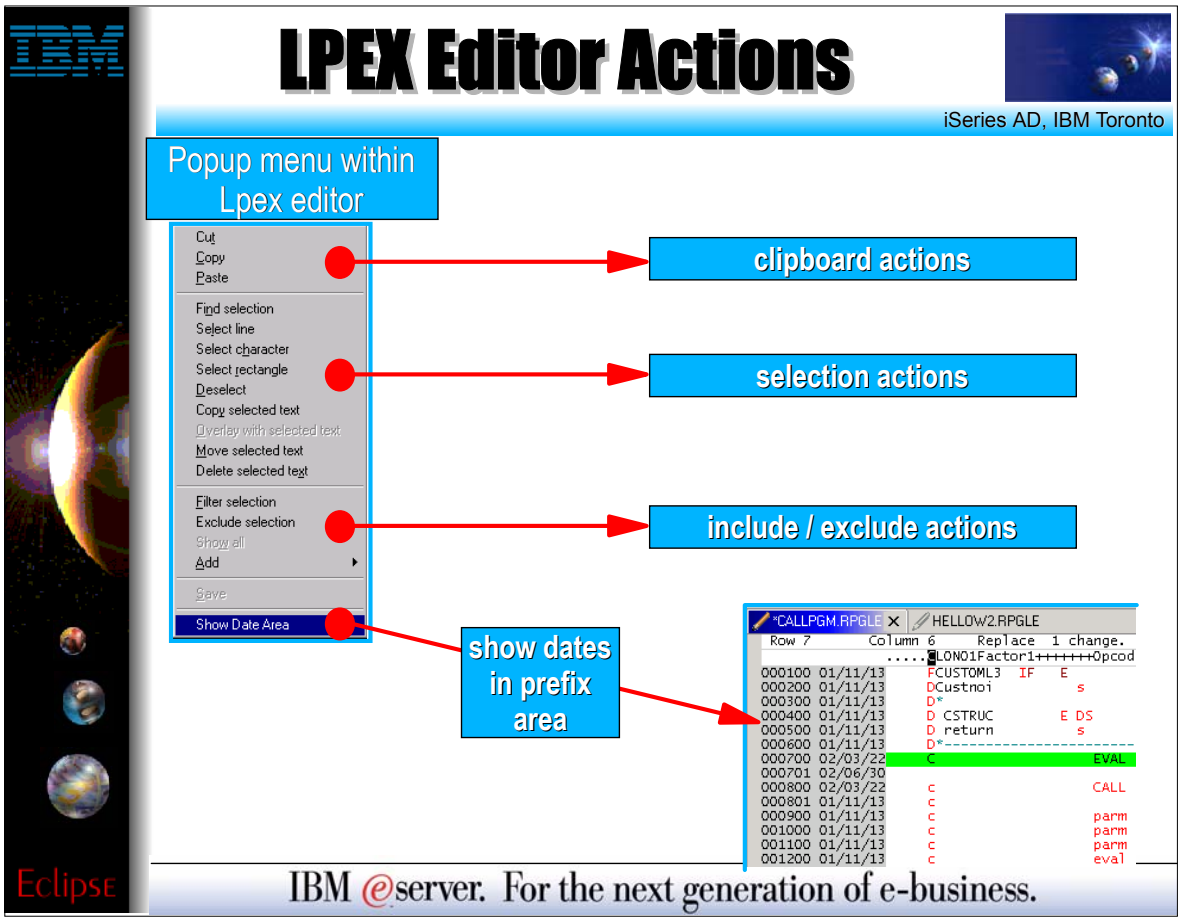
- When a data or source file is expanded in the Remote Systems tree view, all the members within that file are listed underneath the file.
- For each member, you can right-click and select from a number of useful actions. The exact list of actions will depend on whether the member is a data file or a source, and whether you selected one or multiple members.
- For a source file, the popup menu hasctions for editing, renaming, copying, moving, deleting and compiling the source member. We will have more to say about the editing and compiling actions in the next slides...



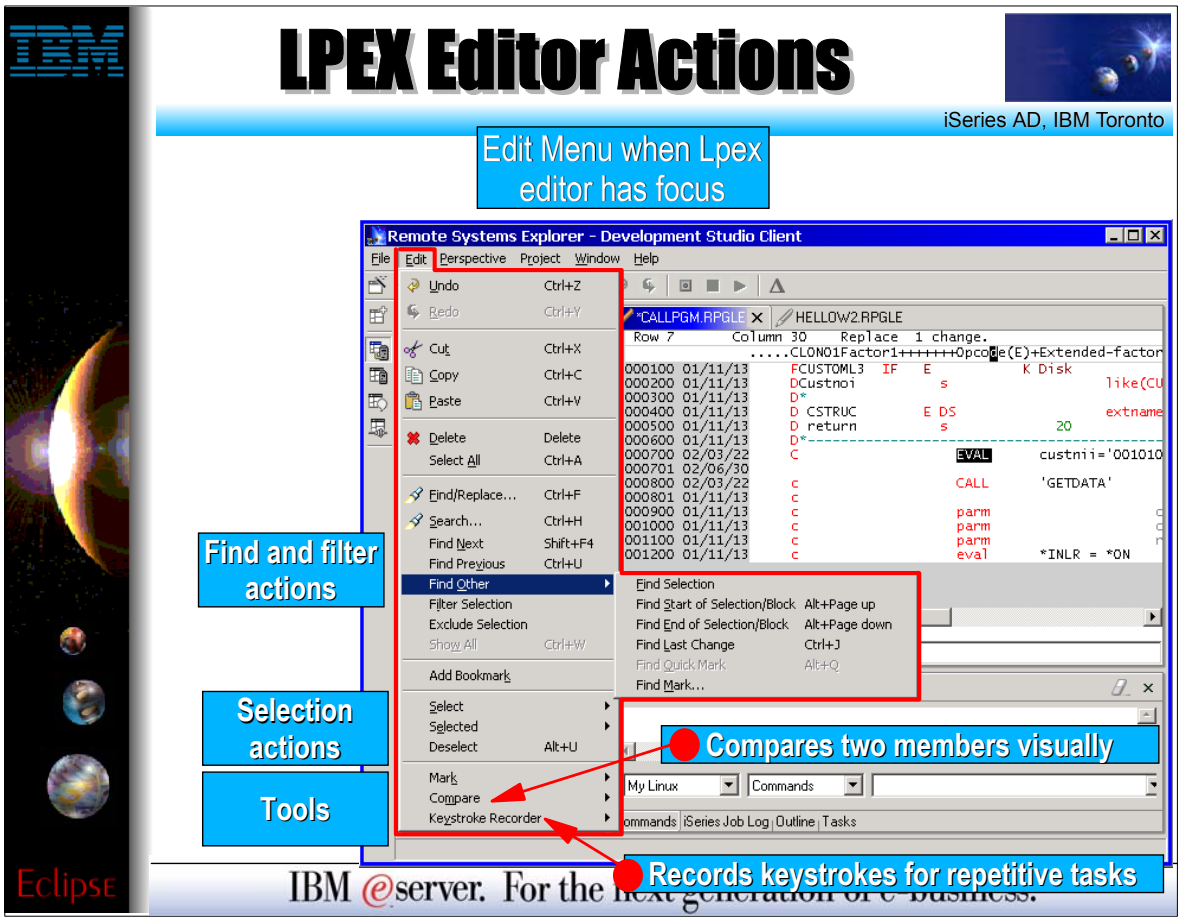
- For a source member, there are two options for editing:
- **1. Lpex Editor**. This is the new editor, written all in Java, that is built-in to the IDE. It is a re-write of the original CODE Editor, but as you will see has a subset of the functionality in CODE at this point.
- **2. CODE Editor**. This is the classic full-functioned CODE editor, which is offered as an alternative until the Lpex editor catches up to the functionality of the CODE editor. This launches the CODE Editor in a separate window.
- We will cover the Lpex editor next in a bit more detail...



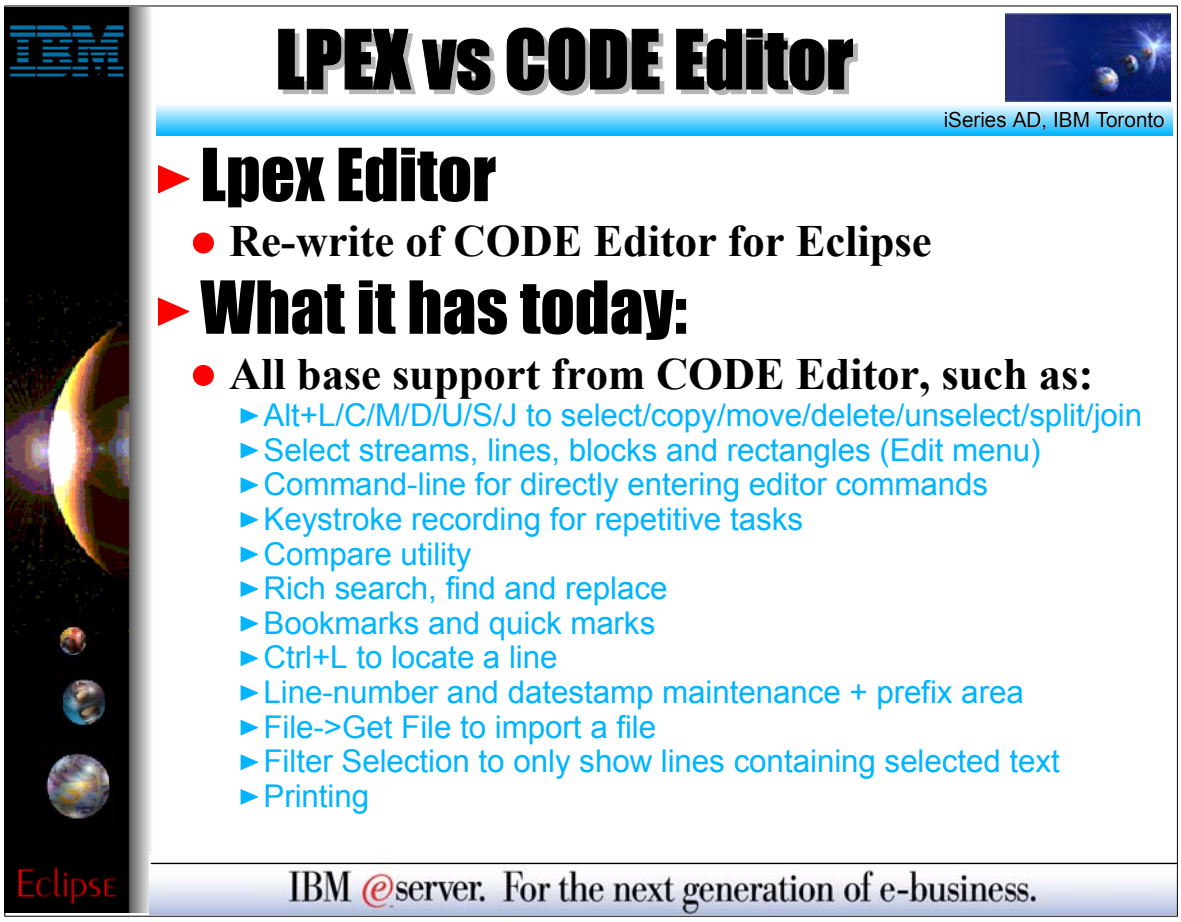
- The Lpex editor is built-in, so it shows up in a pane within the IDE.
- You can open multiple members for editing, and each will be shown in the editor area with a tab that when selected brings that member to the foreground.
- You can double click on a tab to expand that member's edit window to full size.
- When a tab shows an asterisk in it, that indicates there are pending changes that should be saved.
- For RPG (both III and IV) you will notice there is color highlighting and familiar F4 support to prompt for the current line. The prompter sits is a view that doesn't overlap the editor. When done filling in the prompt, you can press one of two buttons to replace the current line or insert a new line.



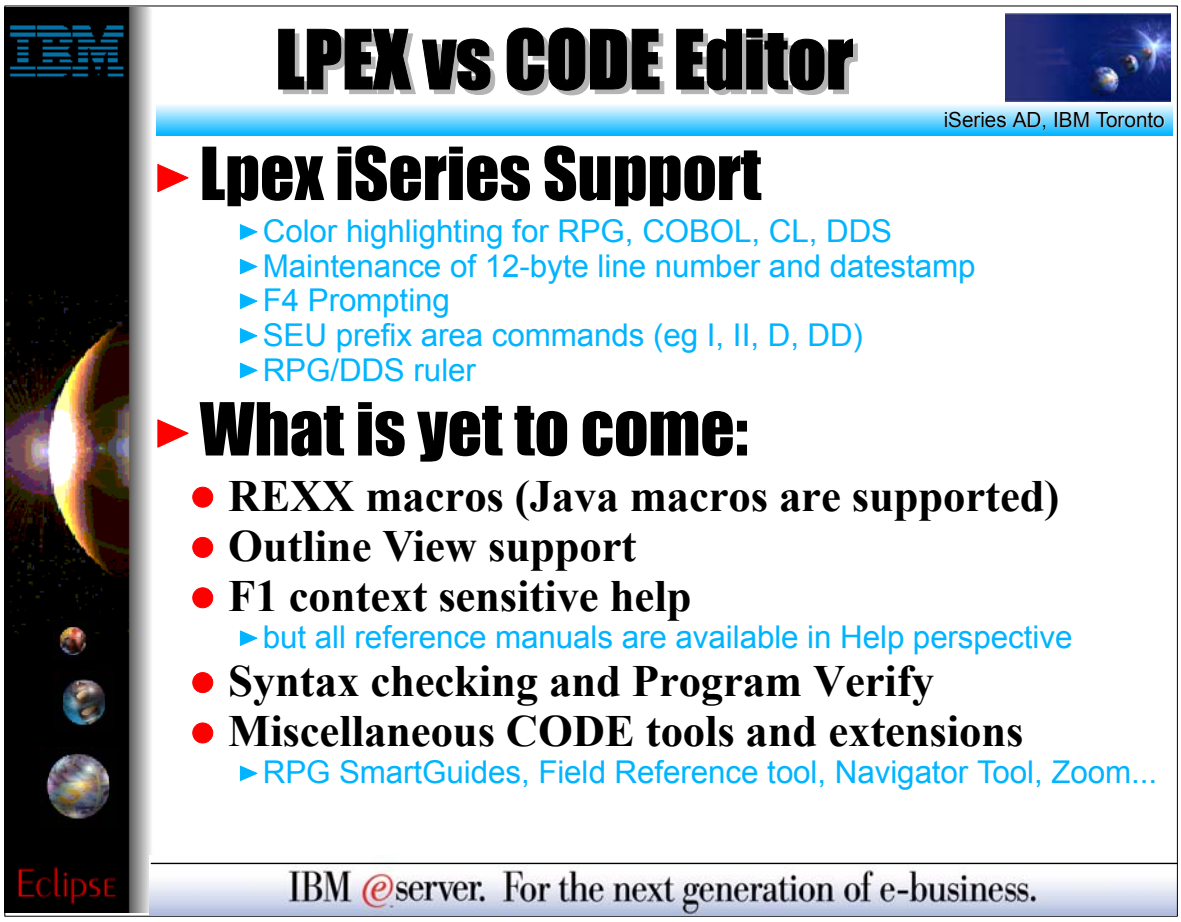
- Here we see the popup menu within the Lpex editor
- The editor is rich with function, and is iSeries-aware!



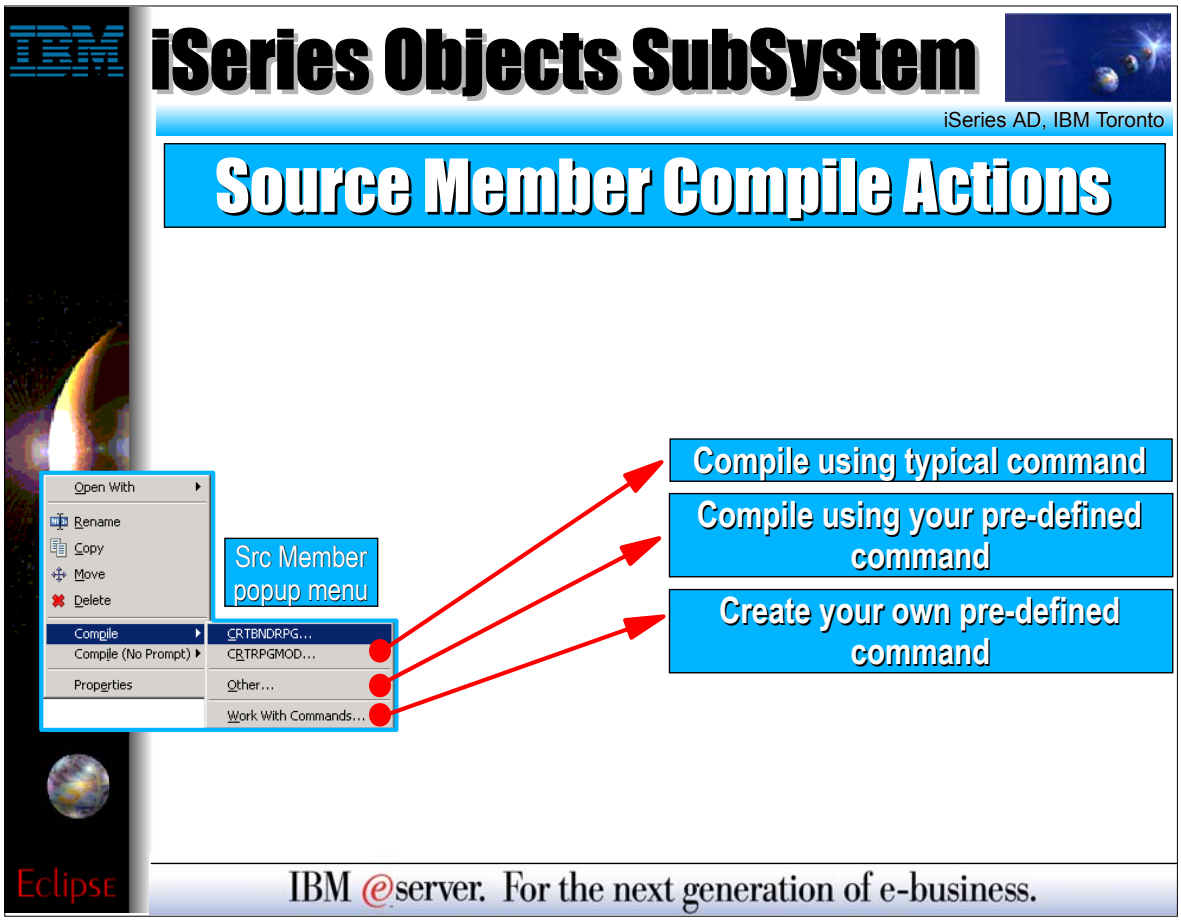
- Here we see the Edit pulldown menu when the Lpex editor is in focus.
- There are even more editor actions here, and there are a handful in the toolbar as well.
- The editor tools include a compare utility to visually compare and merge source members, and a keystroke recorder to record and playback keystrokes for repetitive tasks.



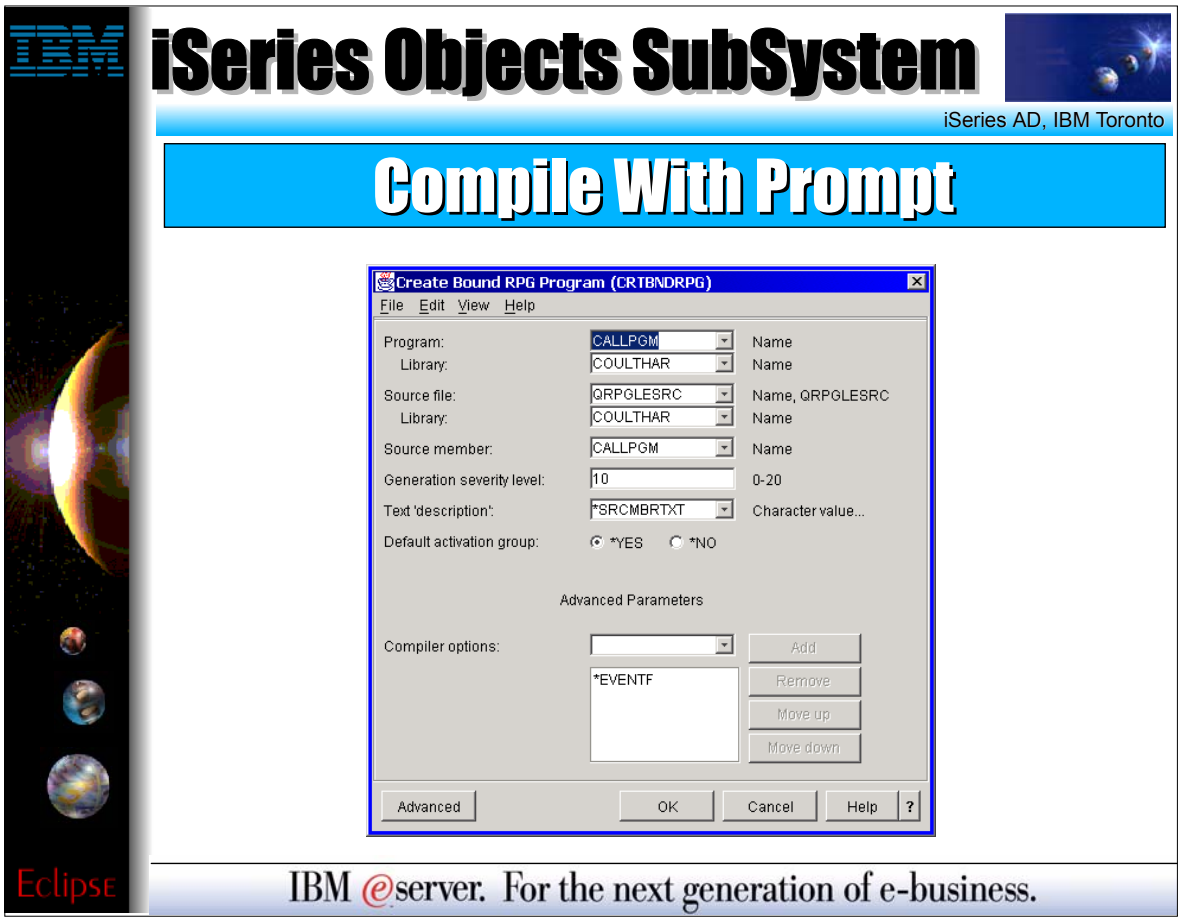
- The Lpex editor in WDSc is a Java rewrite of the classic CODE editor that is written in C++.
- The CODE editor has 10 years of evolution behind it, and not all of its functionality is available in Lpex today.
- However, the Lpex editor still does have a rich base of function, as shown here



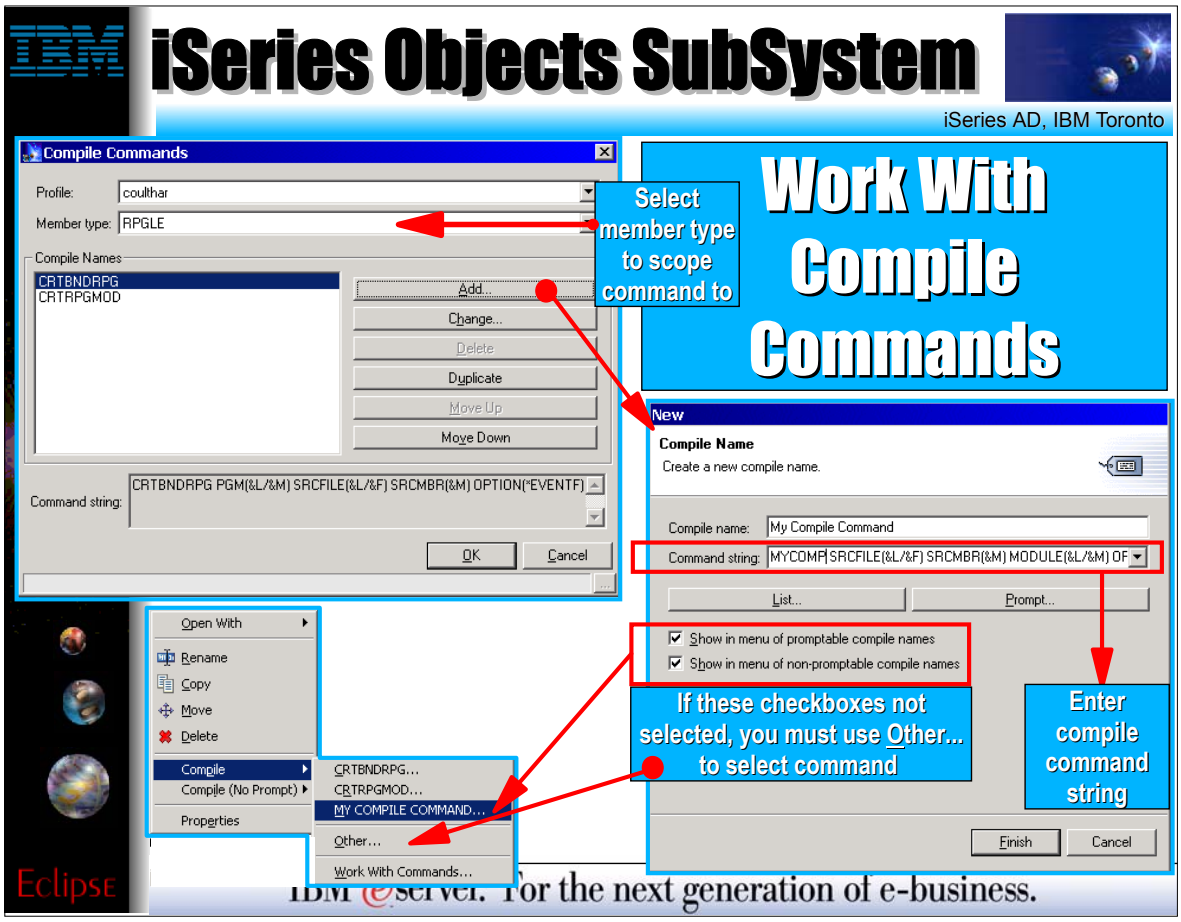
- In addition to the base editor function, there is additional support for each programming language supported by the editor.
- The language-specific functionality currently available in Lpex is listed here.
- However, there is still a number of functions in the CODE editor that are yet to come in Lpex. These too are listed here. Many of these will come in the next release.



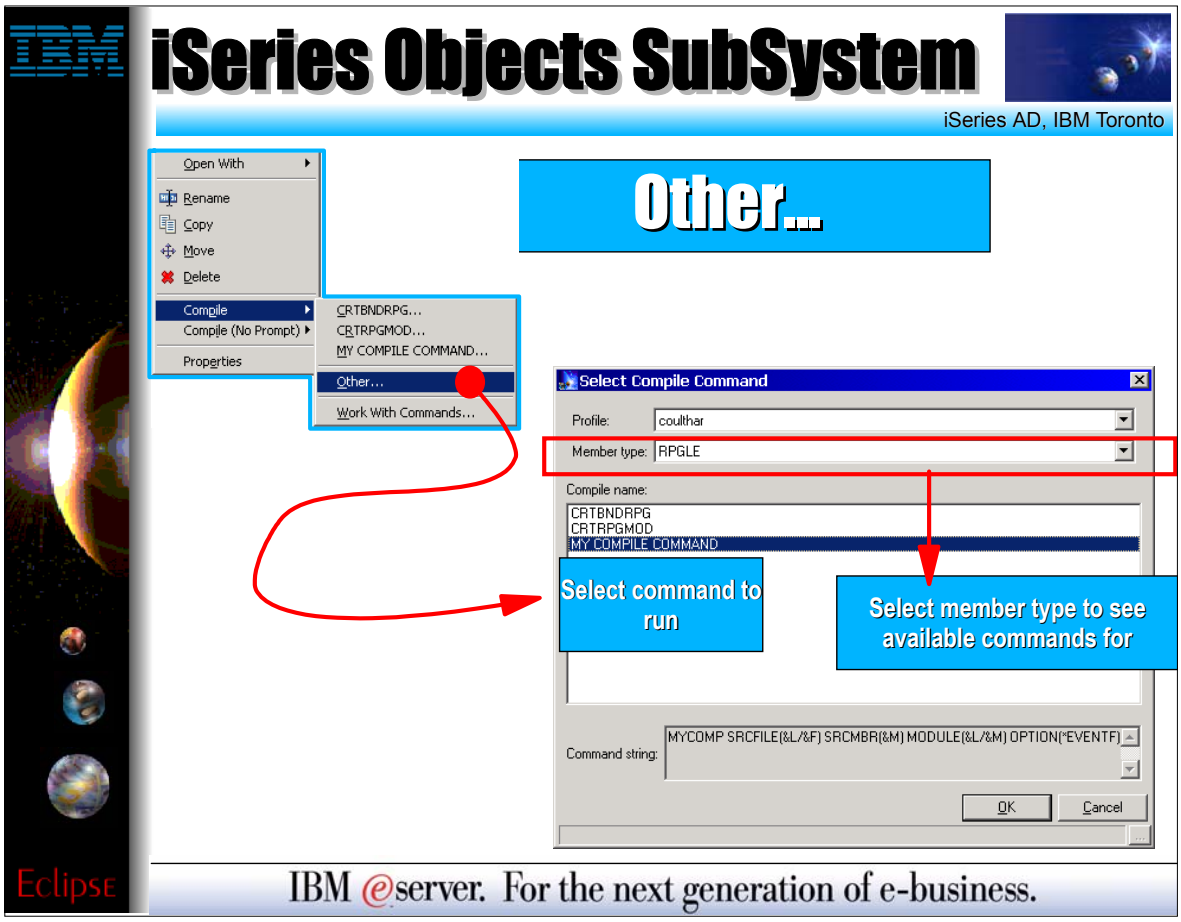
- For a source member, there are two primary options for compiling:
- **1. Compile** (prompt). This runs the selected compile command, and prompts you for parameters.
- **2. Compile (No Prompt)**. The runs the selected compile command, without prompting for the parameters.
- IBM pre-supplies some compile commands specific for the member type, or you can identify your own commands to use see and use, per member type



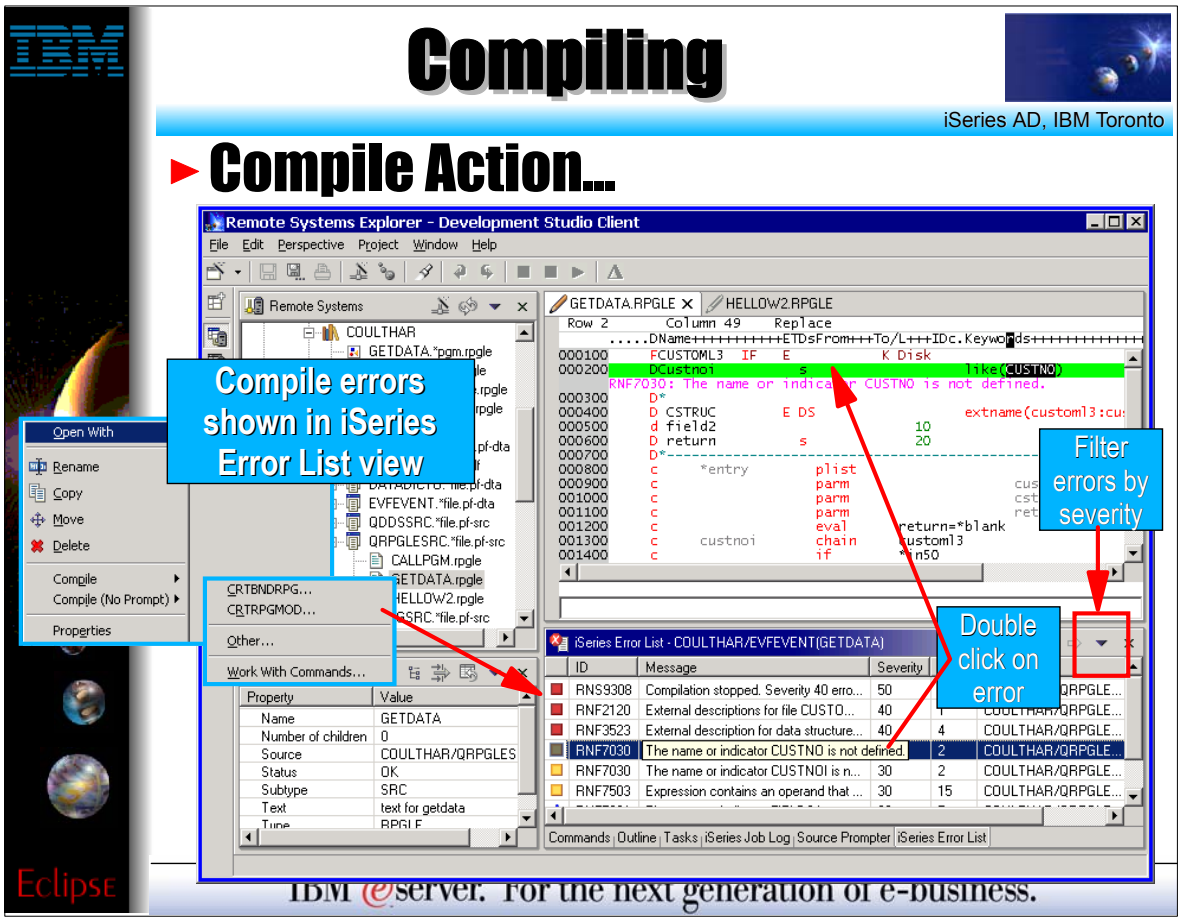
- When you select to prompt the compile command, the command prompt is converted to a GUI and displayed



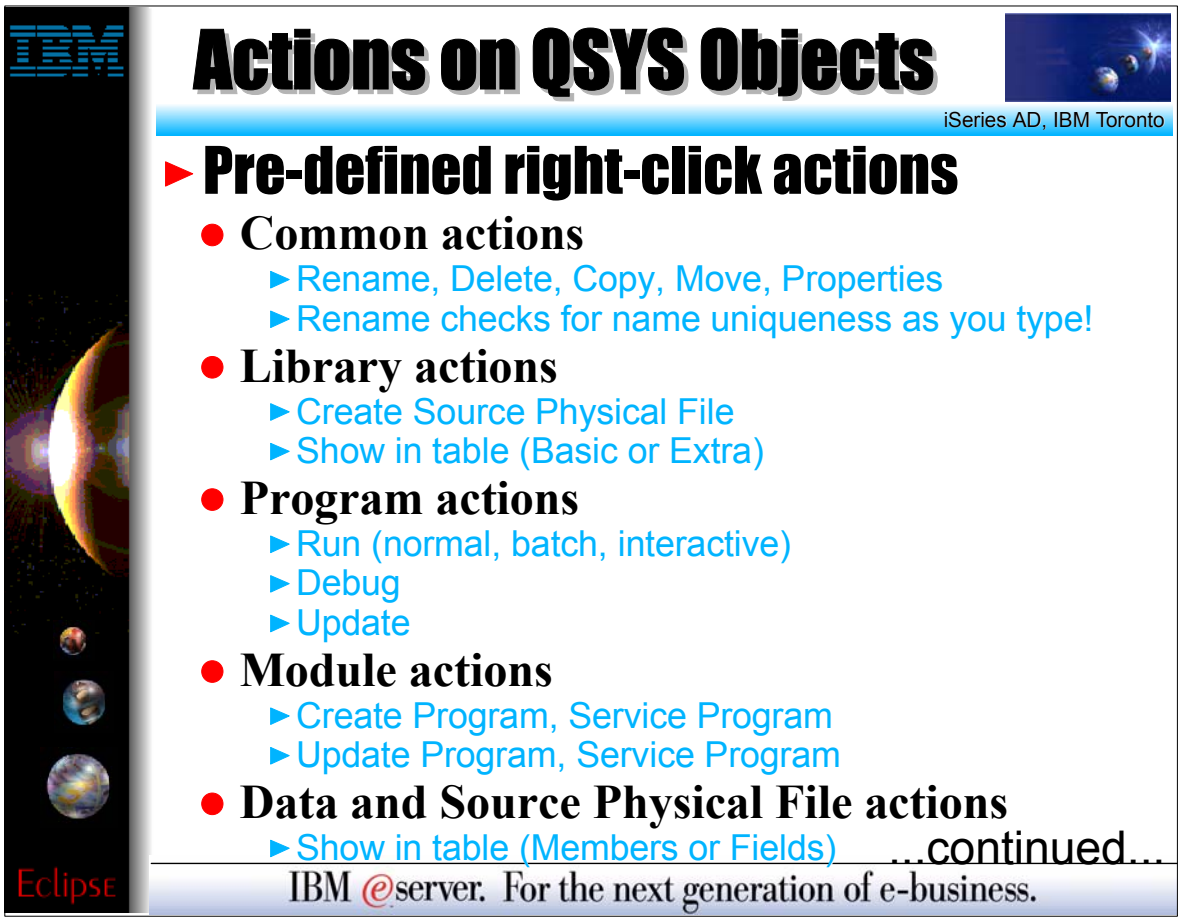
- To create your own compile actions, select the Work With Commands menu item from the cascading compile menu in the popup menu for a source member.
- Here, use the Add or Duplicate buttons to create the commands you want. These may be totally different commands, or the same commands but with different parameters.
- When adding new commands, be sure to select the member type to scope it to. It will only show up in the compile menu for members of this type.
- You can elect to have these commands show up in the compile menus. If you choose not to, then you must use Other... to select the command at compile time



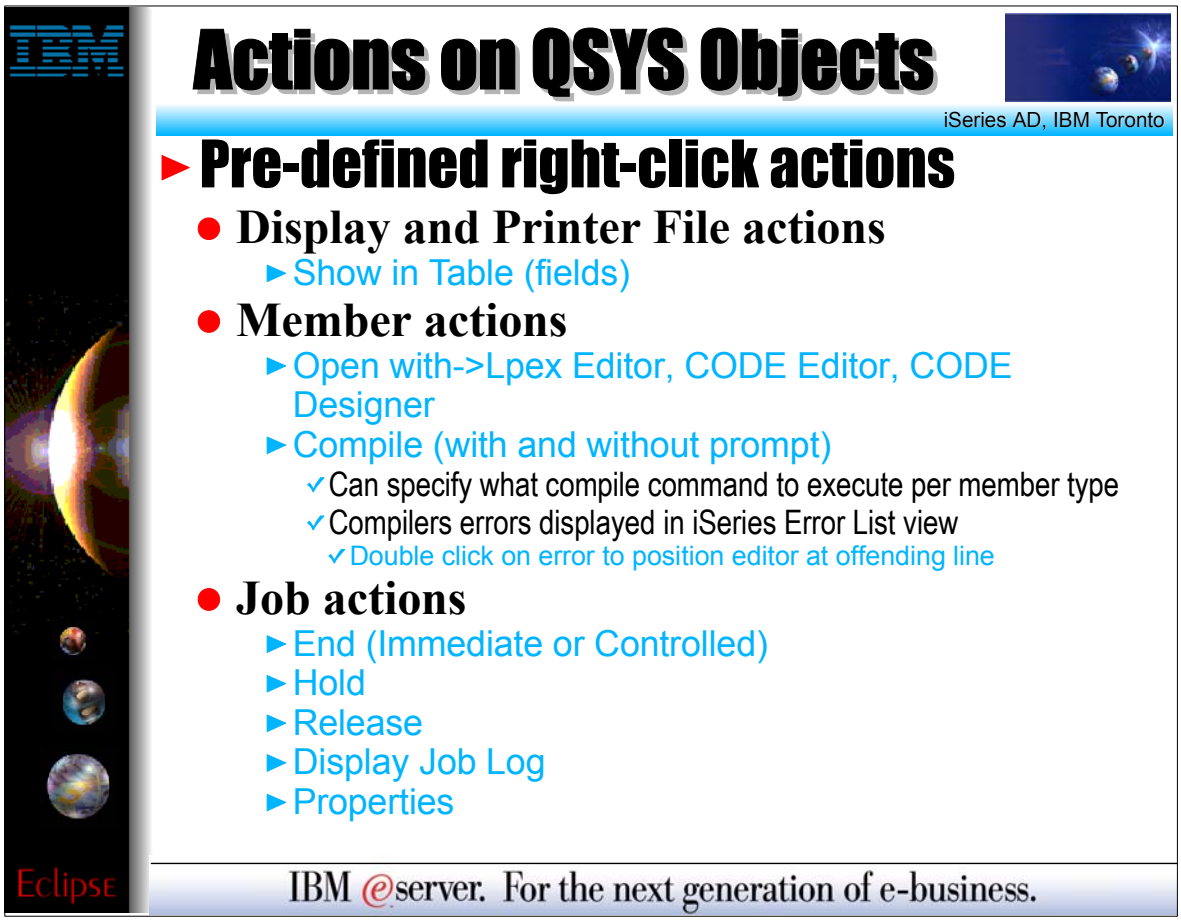
- You use Other... to reach commands not in the compile menu.



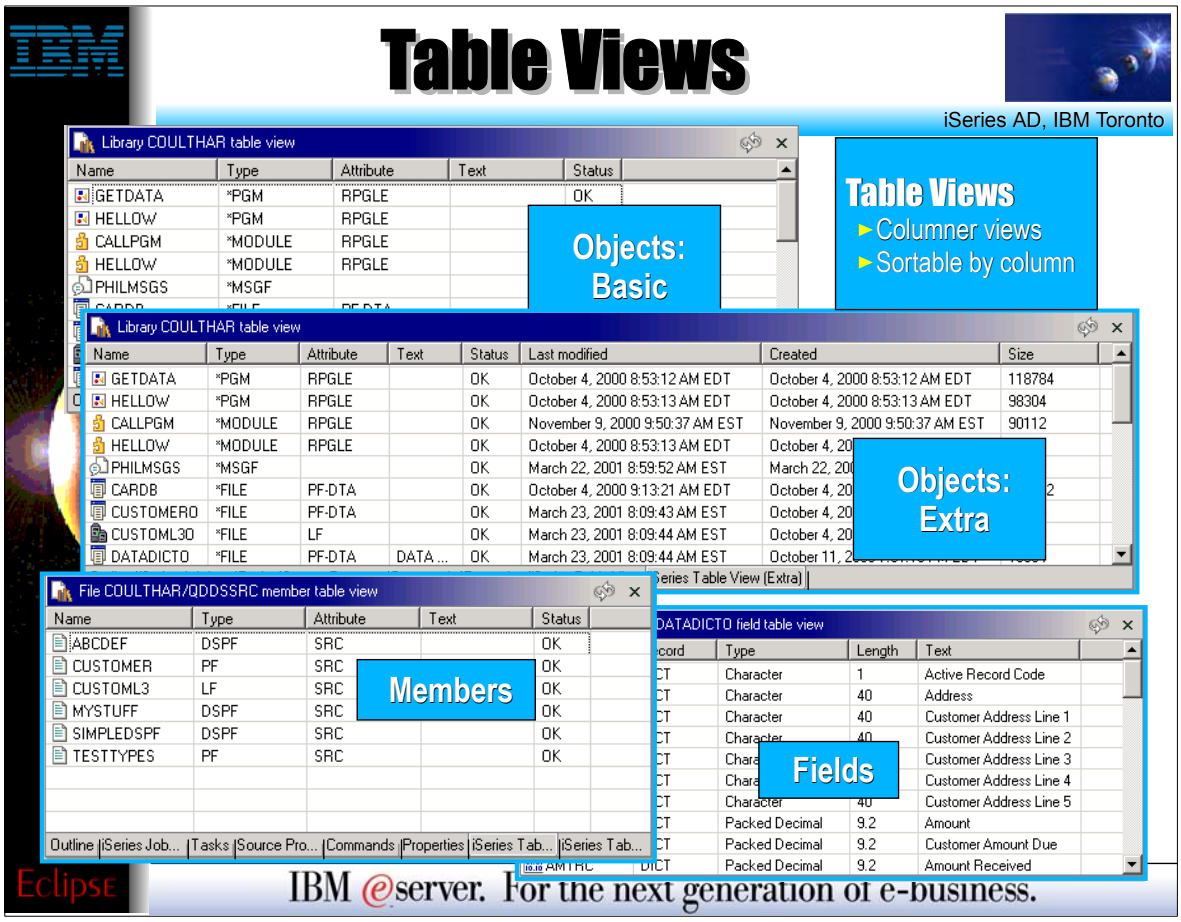
- When you select the compile command to run, the command is submitted to batch (you can change this via Windows->Preferences).
- When the compile is done, there errors are returned and displayed in the **iSeries Error List** view.
- You can double-click on an error to position you in the editor at the line of code causing the error. The error message is also inserted under that line, for context.
- The iSeries Error List view has a pulldown menu to the right of its title bar that can be used to filter out messages by severity.



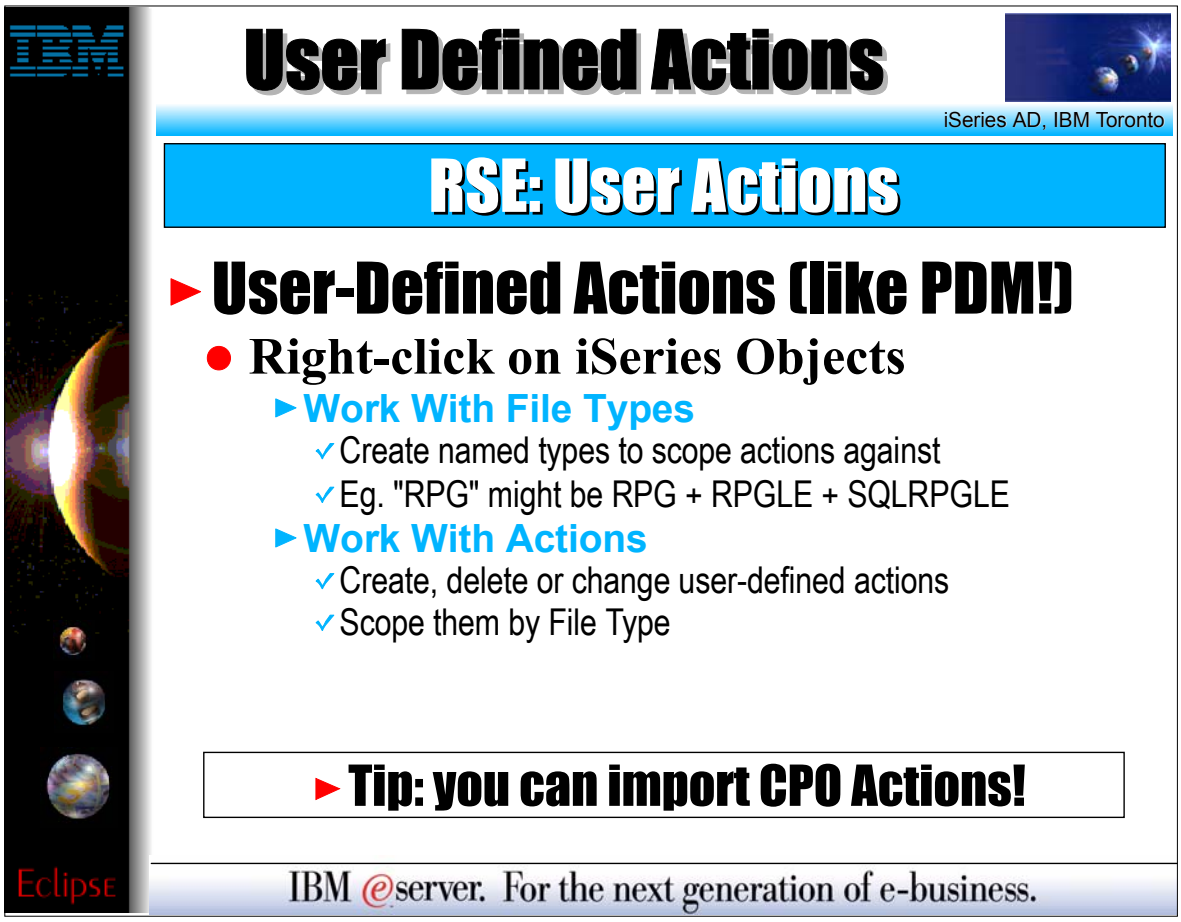
- At this point, a number of the popup menu actions have been shown.
- This slide summarizes the popup menu actions for remote libraries and objects.



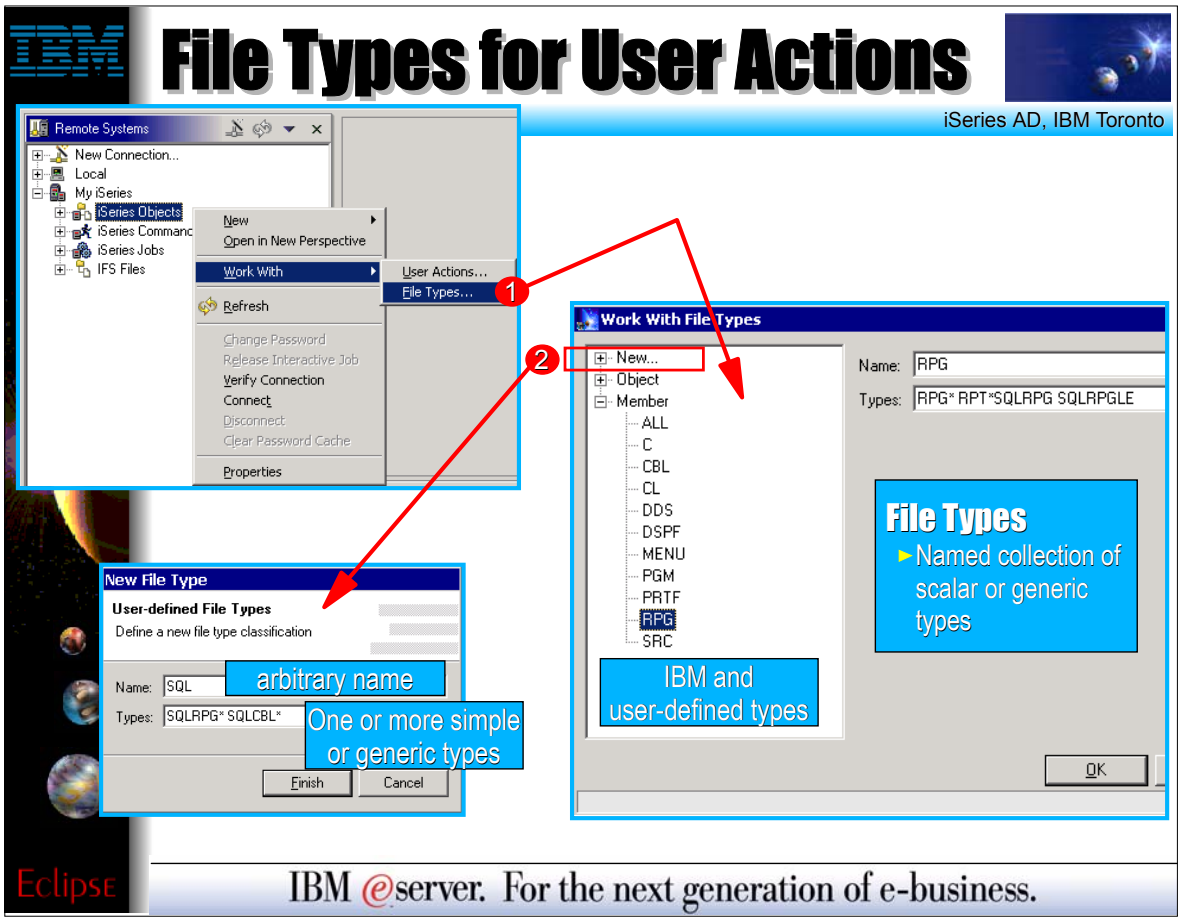
- This slide summarizes the popup menu actions for remote device files, members and jobs (when using the **iSeries Jobs** subsystem).



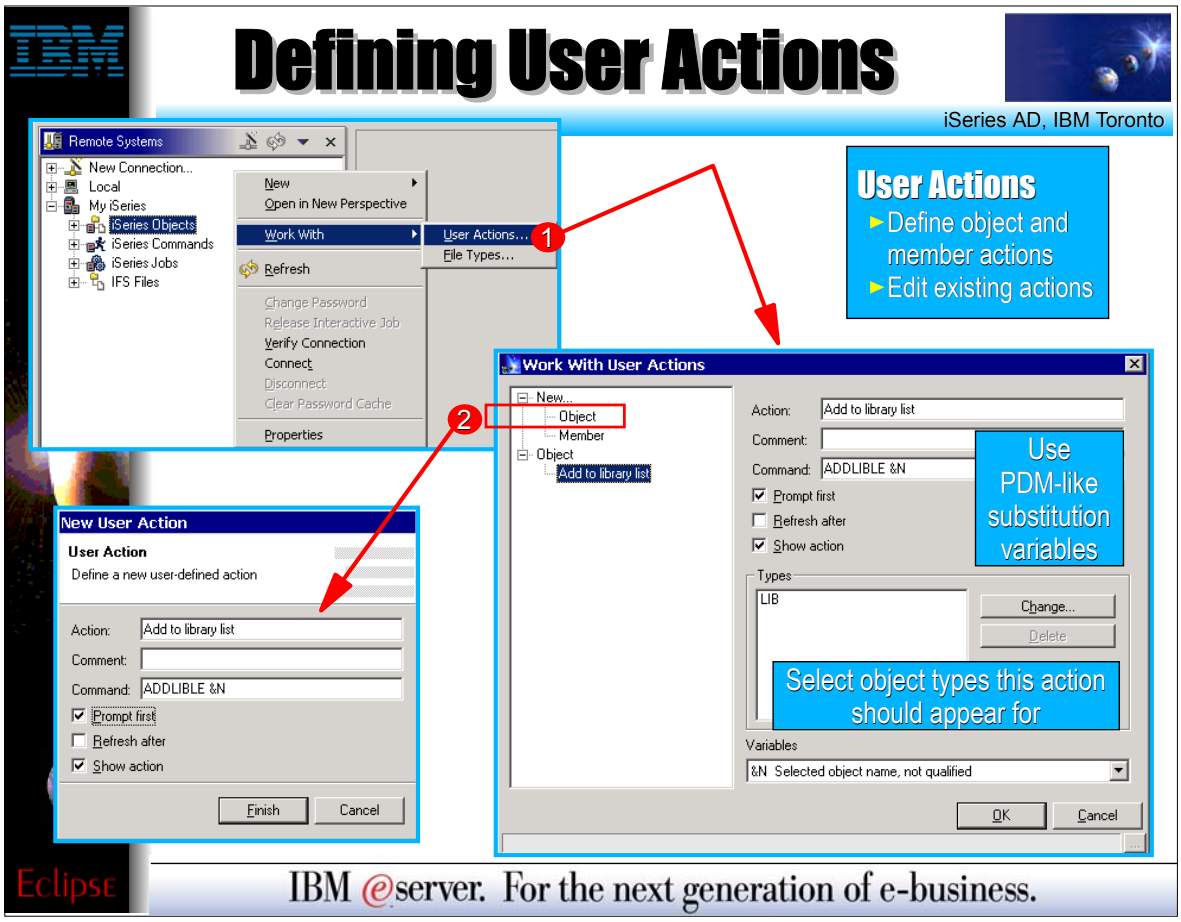
- For libraries and files we can use a popup menu action to open table views to see the contents of the library or file.
- Here we see what those views look like.
- In all these table, the columns are attributes for the object or member in each row. The table can be sorted by an attribute by simply clicking on the column heading.
- The first two tables are listing library contents, first in Basic mode and then in Extra mode.
- The third table is listing members in a database file, and the fourth table is listing fields in a device file or record format.



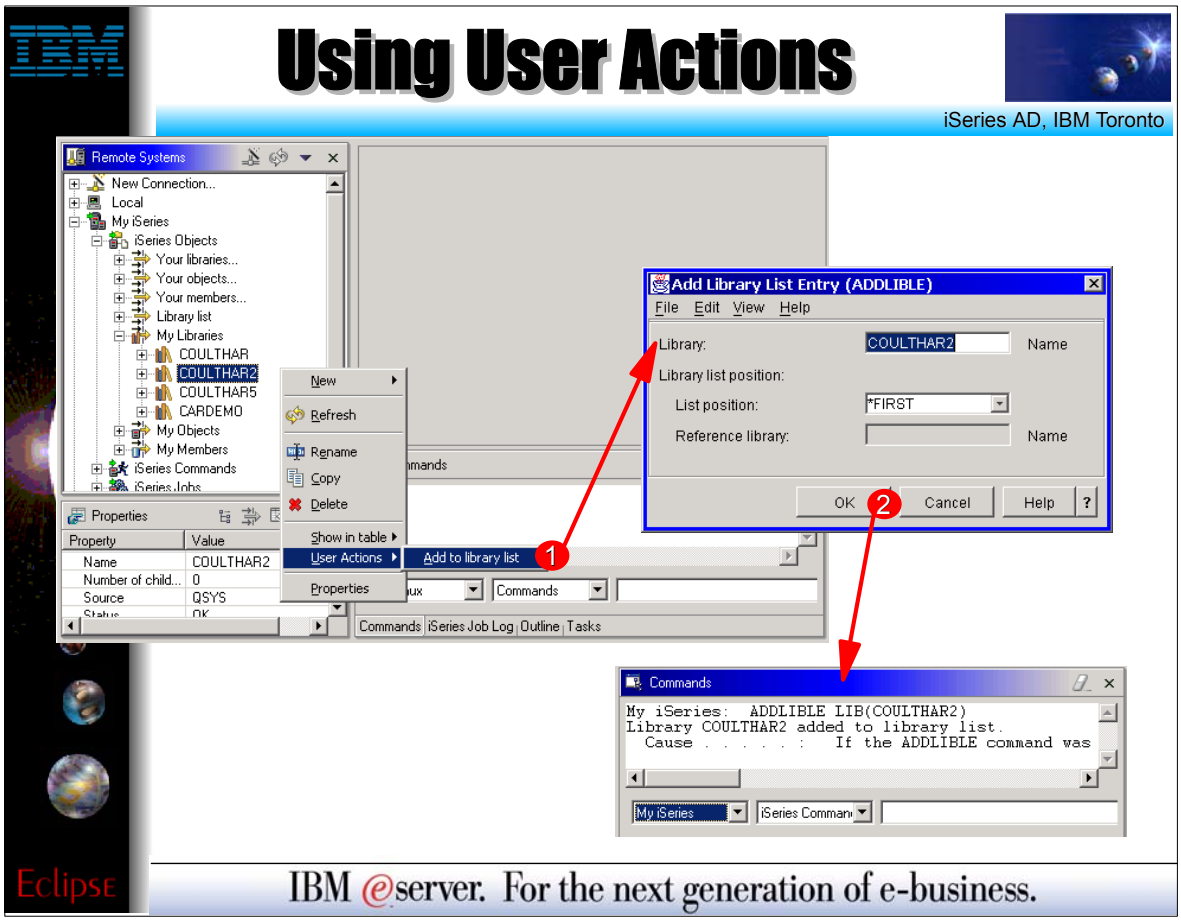
- While IBM supplies a number of useful actions for remote iSeries objects, it is not possible to supply them all.
- Like PDM, you can easily define your own actions.
- To create your own actions, use the Work With actions in the popup menu for iSeries Objects.
- These user-defined actions will appear in the popup menus for remote resources. To avoid seeing all actions in all popup menus, you scope each action to a one or more object or member types.
- You first define named collections of object or member types, then you create your actions and scope them to one of these named collections of types. You actions will then only appear for object or members that match one of the types in the collection.
- If you are a CODE user, you can use File->Import to import existing actions from CODE Project Organizer.



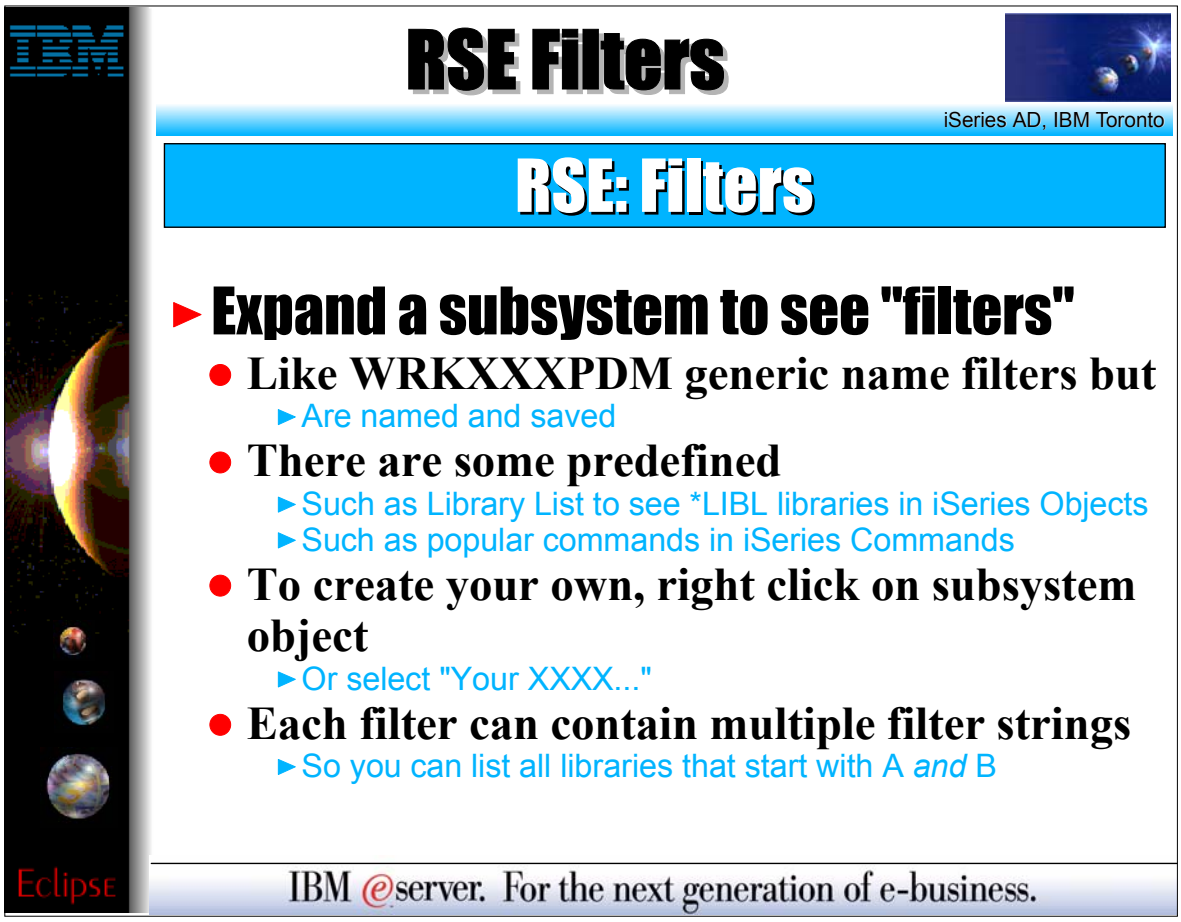
- To define named collections of types for action scoping, use the Work With -> File Types popup menu action for the iSeries Objects subsystem.
- Select New... to define a new collection: give the collection a name and type in one or more simple or generic types.
- There are two types of named collections: one for objects and one for members.
- Once you have defined a type, it shows up in the dialog, and can be edited by simply selecting it on the left, and changig the information on the right.
- There are many pre-defined types supplied by IBM



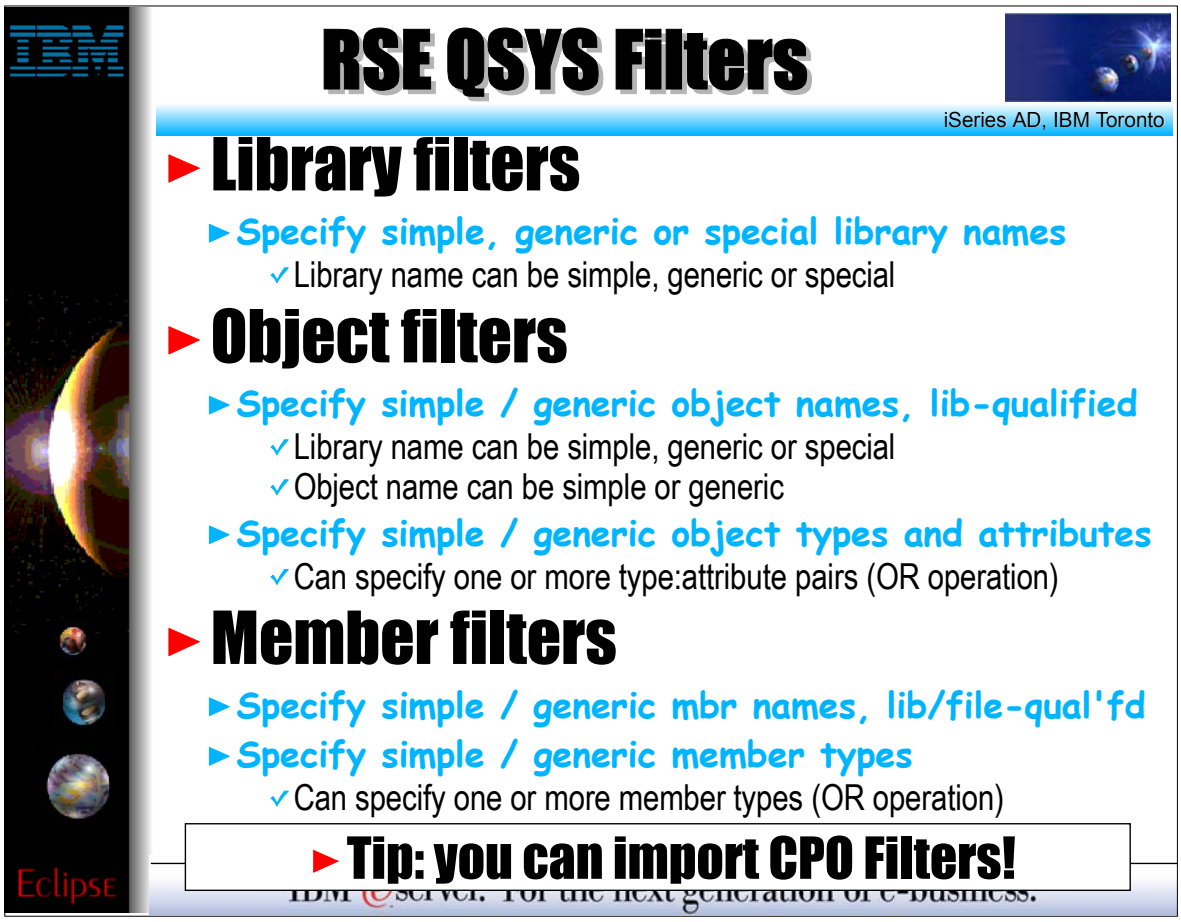
- Once you have defined a type to scope your action by, use Work With -> User Actions to define an object or member action.
- Object actions only appear in the popup menus for libraries or objects.
- Member actions only appear in the popup menus for members.
- When you define the action, you initially specify a label to show in the popup menu and a non-interactive iSeries command to run when that action is selected. This command can use substitution variables that are identical to those in PDM.
- After defining the command, select it in the list and select one or more types to scope it by.



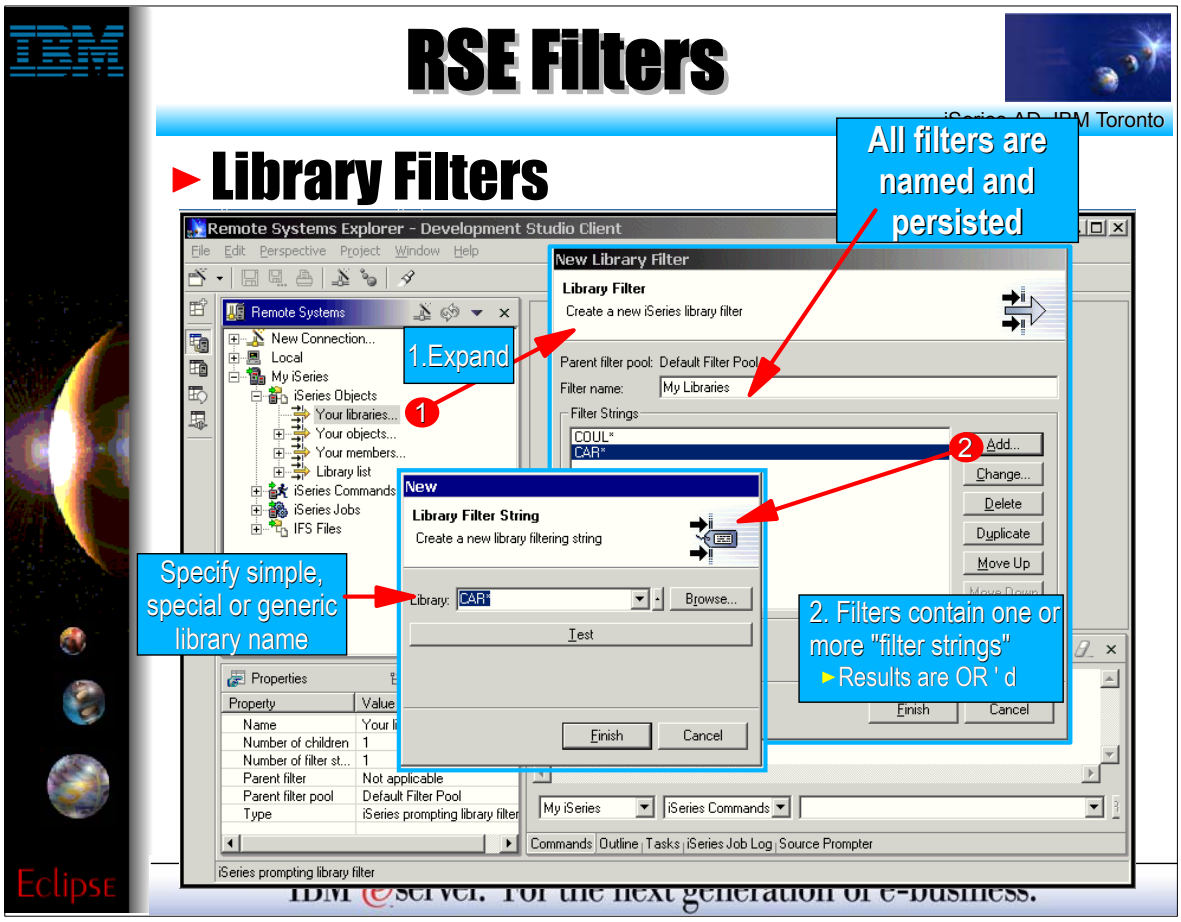
- Once your action is defined, you can use it.
- Right click on an object matching one of the types you specified, and expand the **User Actions** menu in the popup.
- Your action appears in the menu. Select it. If you chose to prompt the command, you will see the GUI prompt for the command.
- When the command has finished running, it results are logged in the Commands view.



- The Library List filter IBM supplies is useful, but it won't be long before you want to create your own filter to subset the list for performance and productivity reasons.
- Filters in RSE are like WRKXXXPDM commands in that you specify simple or generic names, and object or member type criteria, to generate a subsetted list. Unlike PDM however, the filters in RSE are saved between sessions, so you need only specify the filtering criteria once.
- All filters in RSE are named collections of filter strings. It is the filter strings themselves that hold the filtering criteria. By allowing multiple filter strings for each filter, you can complete flexibility in what objects are shown when that filter is expanded. For example, sometimes you cannot use a single generic name to capture all the objects or members for a particular task.



- There are three types of filters you can create in the iSeries Objects subsystem.
- 1. **Library filters**. These list libraries when expanded.
- 2. **Object filters**. These list objects when expanded.
- 3. **Member filters**. These list members when expanded.
- We will see an example of creating each of these.
- If you are an existing CODE user, you can use File->Import to import filters from CODE Project Organizer.

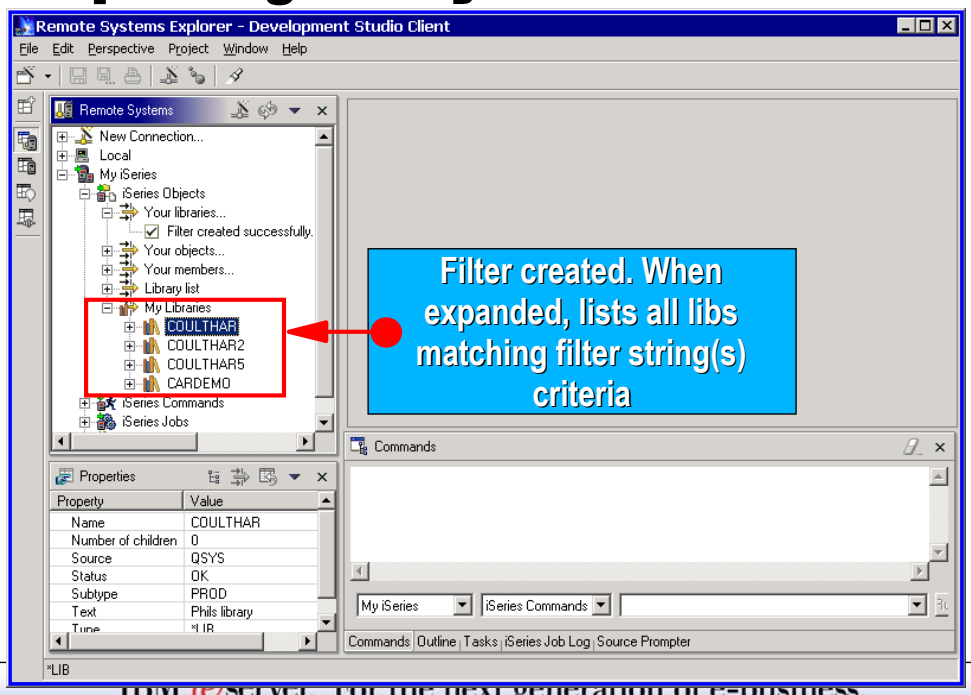


- Here we use the **Your libraries...** prompt to create a library filter. This prompt creates a filter, and then immediately expands it for you. Alternatively, you can also right click on the iSeries Objects subsystem and use a popup menu action to create a new library filter.
- To create a library filter, type in a name to give the filter. This is what will appear in the tree under iSeries Objects.
- Press **Add...** once or more to add library filter strings. Each is a simple or generic library name.

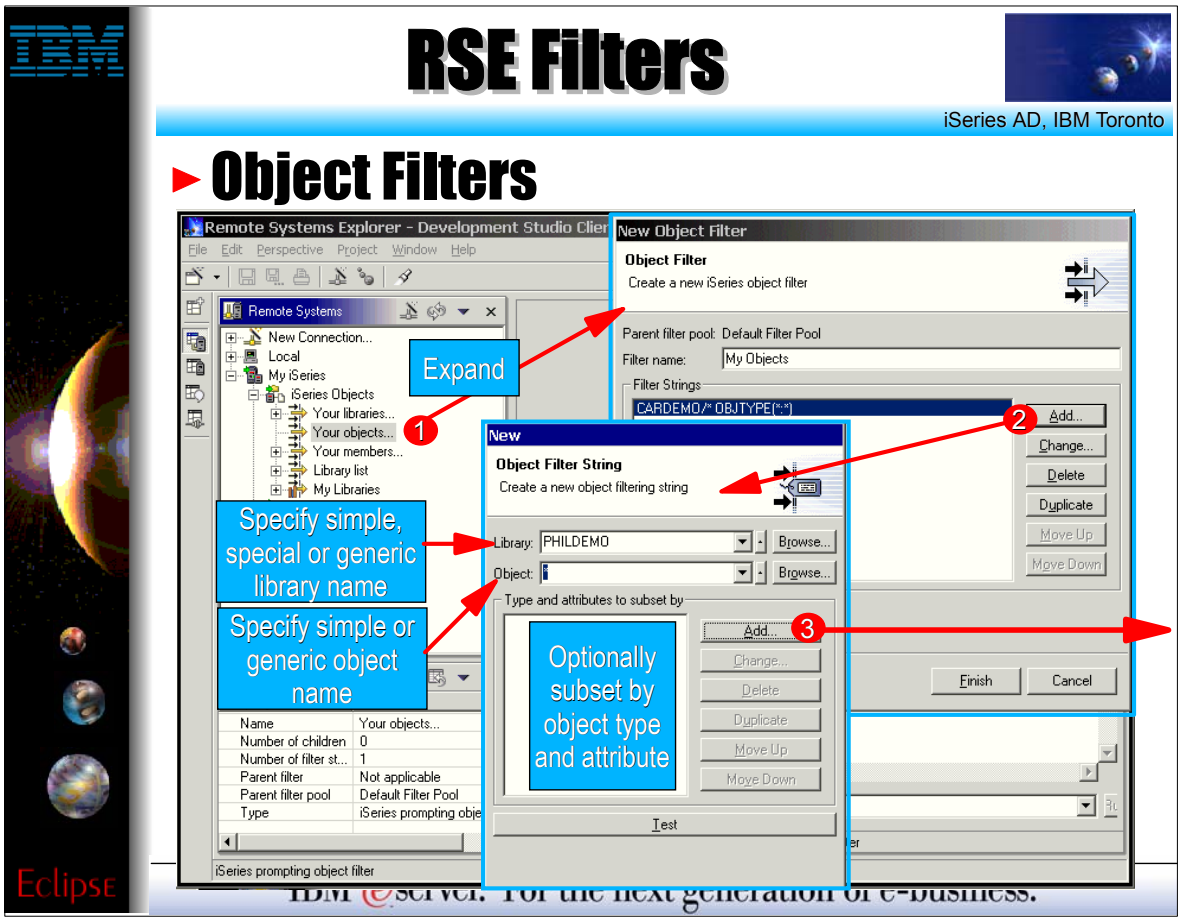
RSE Filters

iSeries AD, IBM Toronto

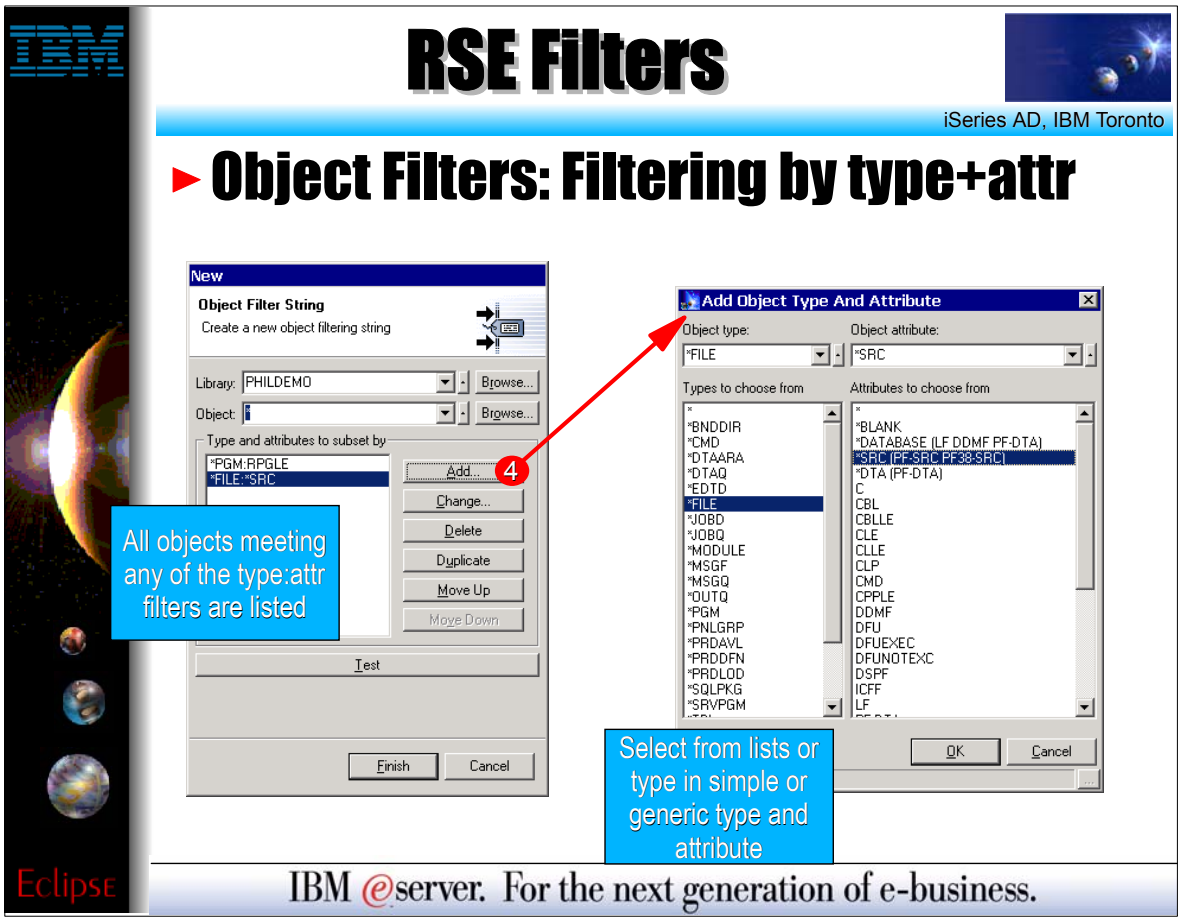
Expanding Library Filters



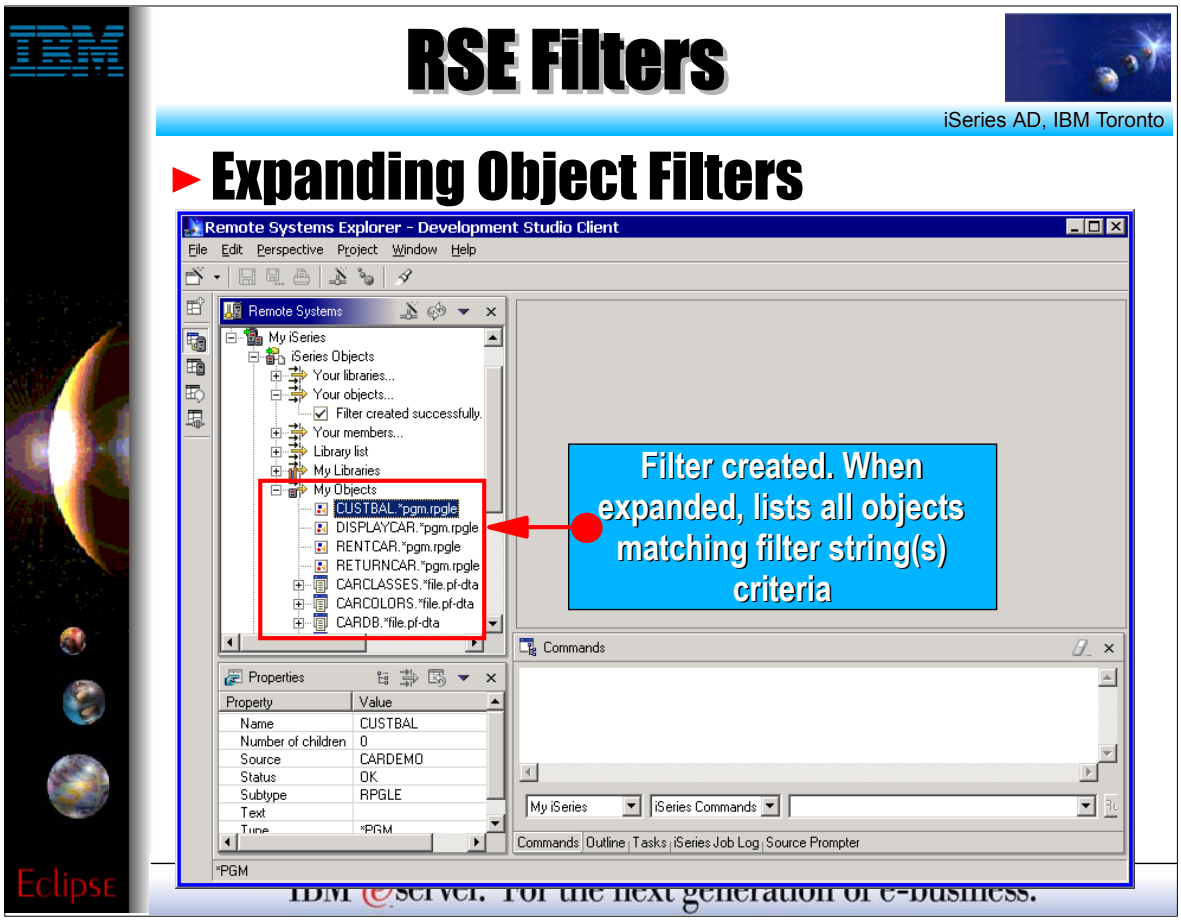
- When a library filter is created, it shows up in the list of existing filters for iSeries Objects.
- You can edit the filter by right clicking on it and selecting the Change... action from its popup menu.
- When a library filter is expanded, all libraries matching one or more of the filter's filter strings are listed underneath the filter.
- Note that these libraries can subsequently be expanded, just as they can for the Library List filter.



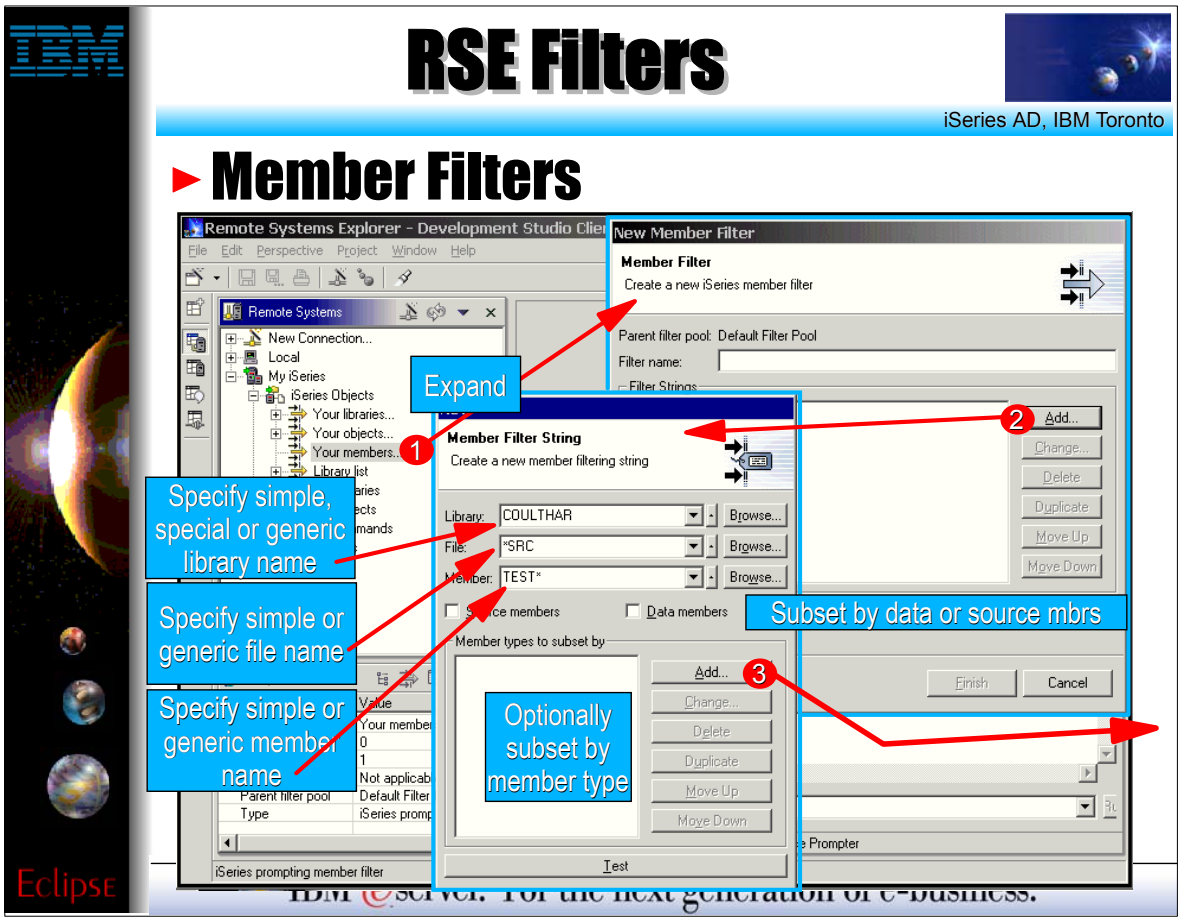
- Here we use the **Your Objects...** prompt to create a library filter. This prompt creates a filter, and then immediately expands it for you. Alternatively, you can also right click on the iSeries Objects subsystem and use a popup menu action to create a new object filter.
- To create an object filter, type in a name to give the filter. This is what will appear in the tree under iSeries Objects.
- Press **Add...** once or more to add object filter strings.
- Each object filter string identifies objects to list. The library and object name can be simple or generic.
- To further subset the list to only objects of particular types or attributes, press **Add...** from the filter string wizard...



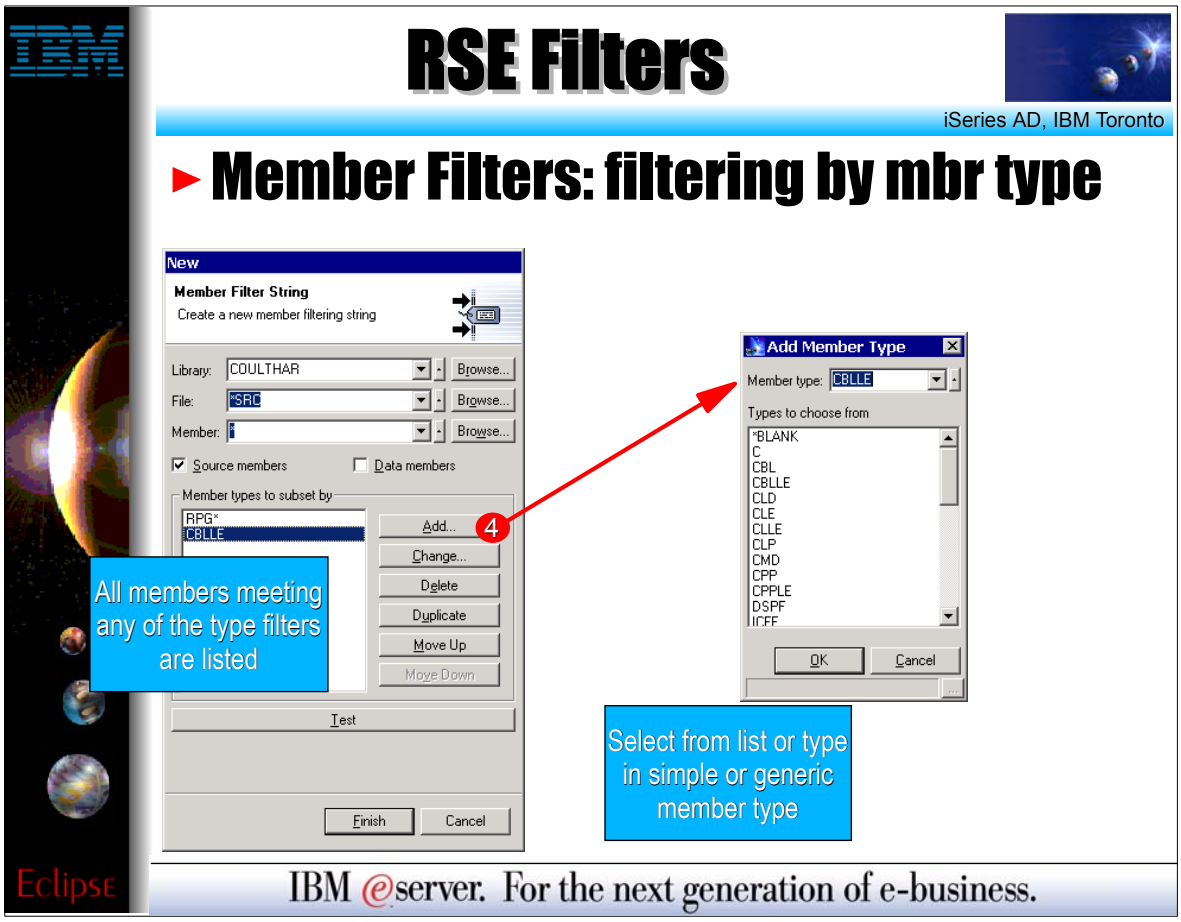
- The Add button in the Object Filter String wizard allows you to specify one or more object type and object attribute pairs. The object type can be any valid iSeries object type. The object attribute can be * to match on all objects of that type, or a simple or generic attribute to restrict to only objects of that type that match the attribute.
- You can add as many object type and attribute pairs as you need to refine your list. All objects that match on any of these type and attribute pairs will be listed, when the filter is expanded.



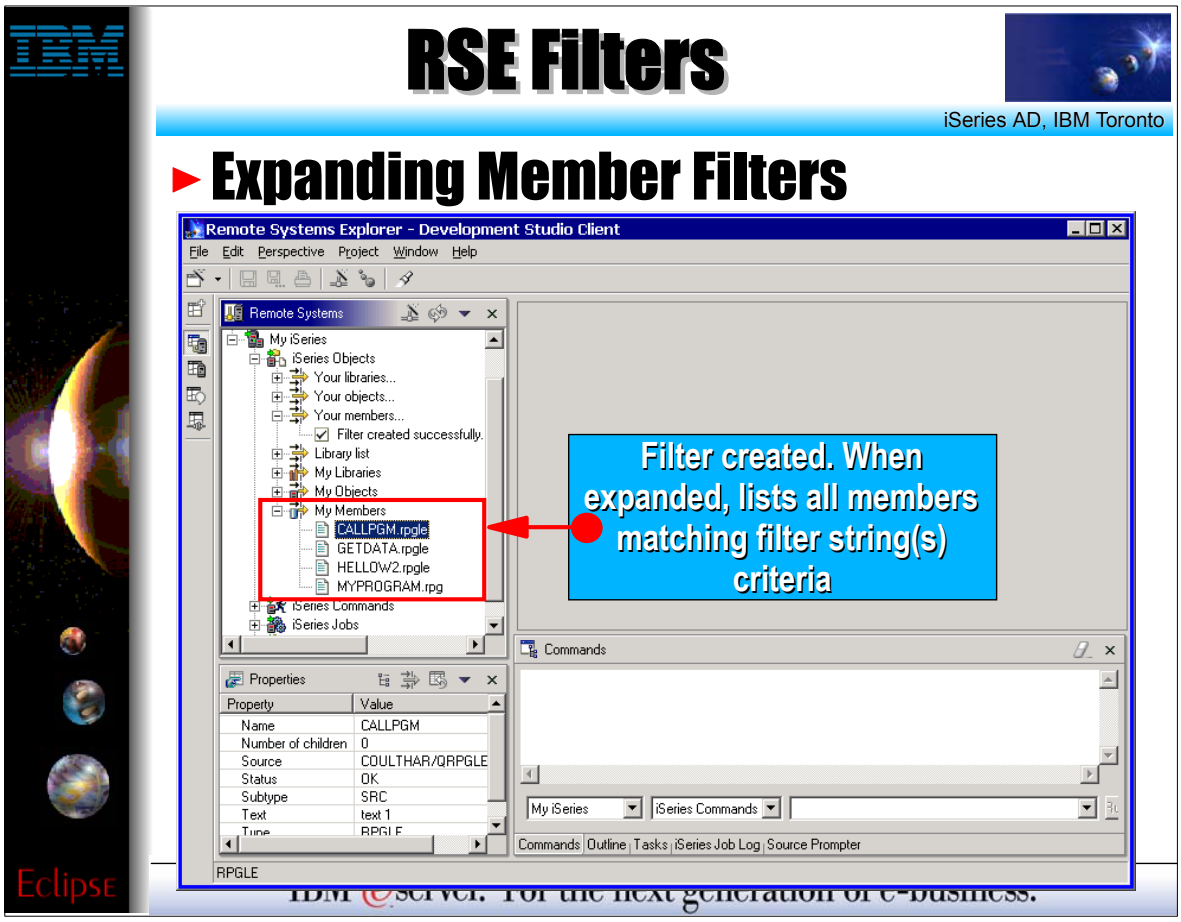
- When an object filter is created, it shows up in the list of existing filters for iSeries Objects.
- You can edit the filter by right clicking on it and selecting the Change... action from its popup menu.
- When an object filter is expanded, all objects matching one or more of the filter's filter strings are listed underneath the filter.
- Note that these objects can subsequently be expanded, if they are files, just as they can from the Library List filter.



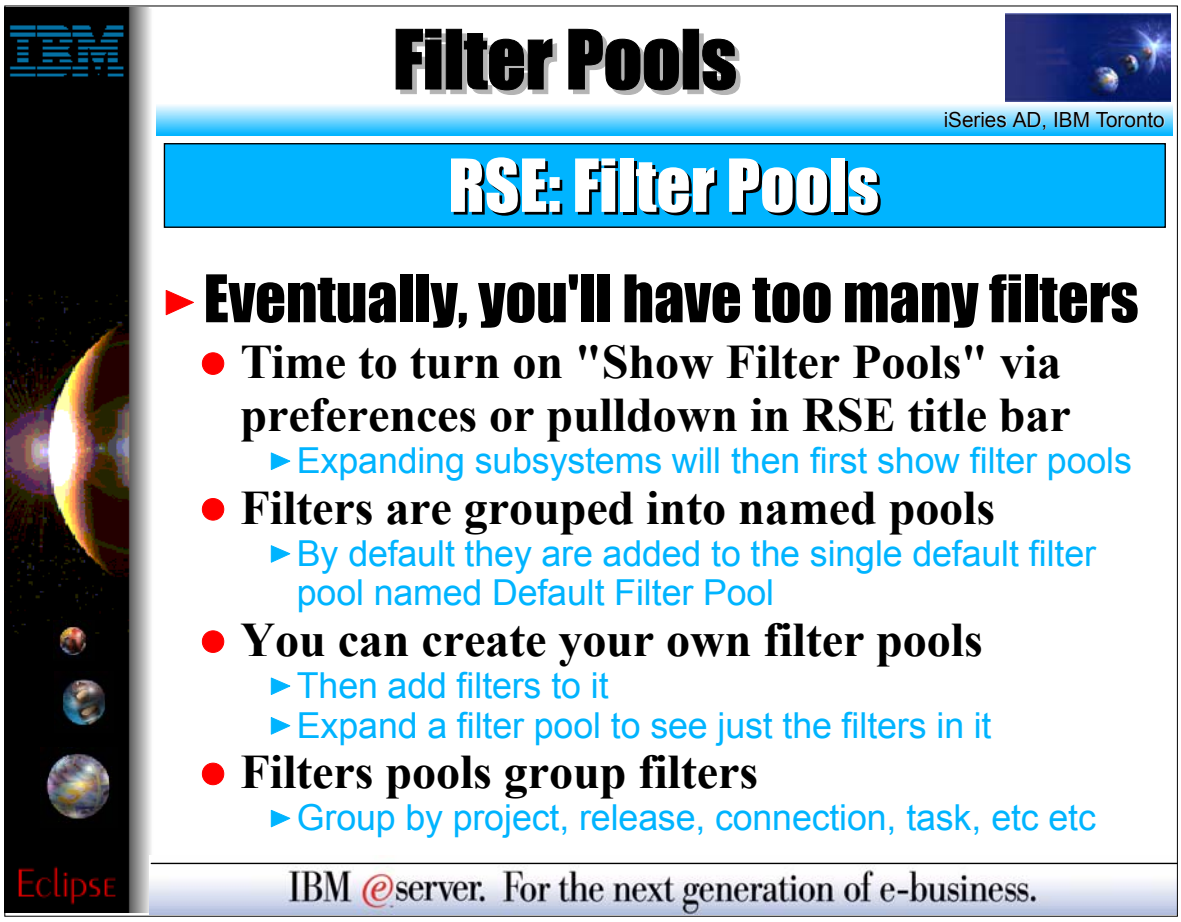
- Here we use the **Your Members...** prompt to create a member filter. This prompt creates a filter, and then immediately expands it for you. Alternatively, you can also right click on the iSeries Objects subsystem and use a popup menu action to create a new member filter.
- To create a member filter, type in a name to give the filter. This is what will appear in the tree under iSeries Objects.
- Press **Add...** once or more to add member filter strings.
- Each member filter string identifies members to list. The library, file and member name can be simple or generic.
- To further subset the list to only members of particular types press **Add...** from the filter string wizard...



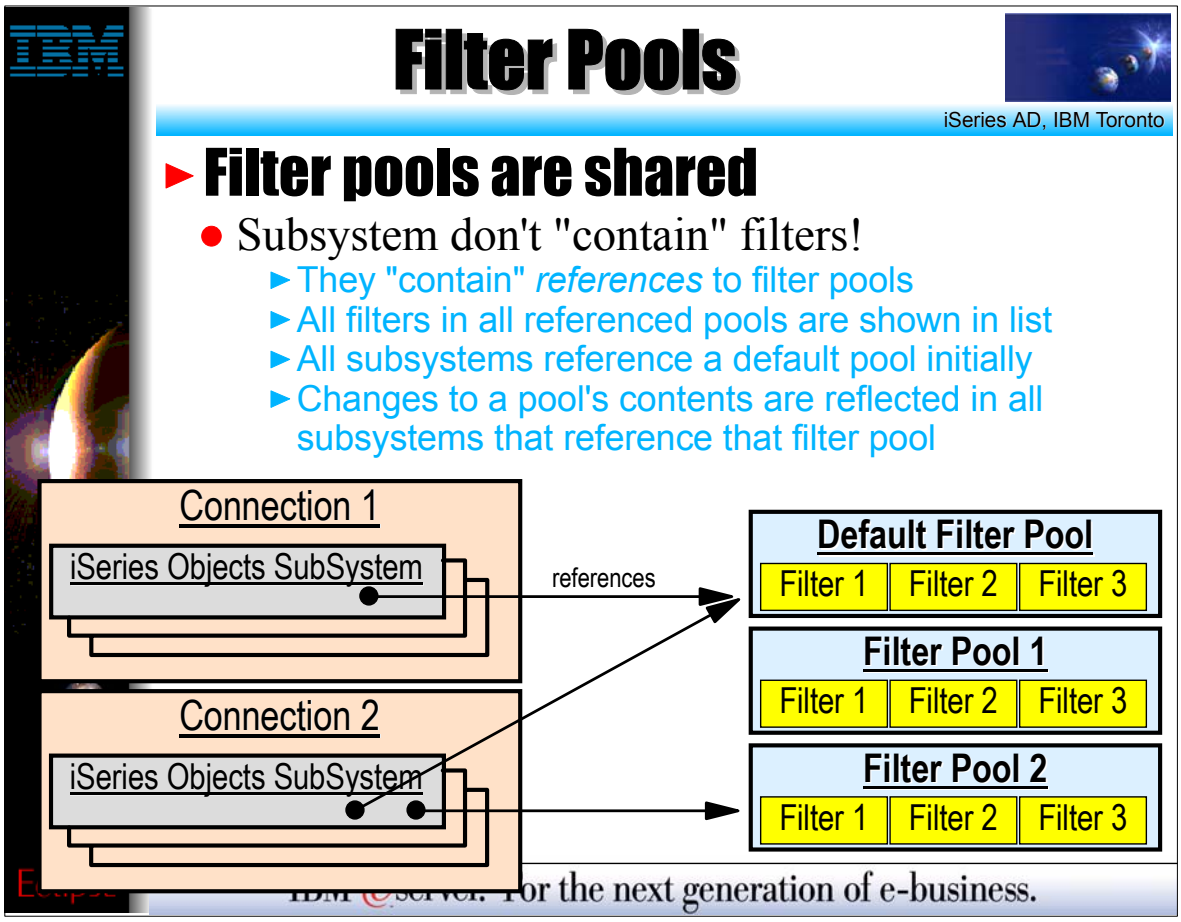
- The Add button in the Member Filter String wizard allows you to specify one or more member types. Each member type can be a simple member type, or a generic member type as in RPG*.
- You can add as many member types as you need to refine your list. All members that match on any of these type will be listed, when the filter is expanded.



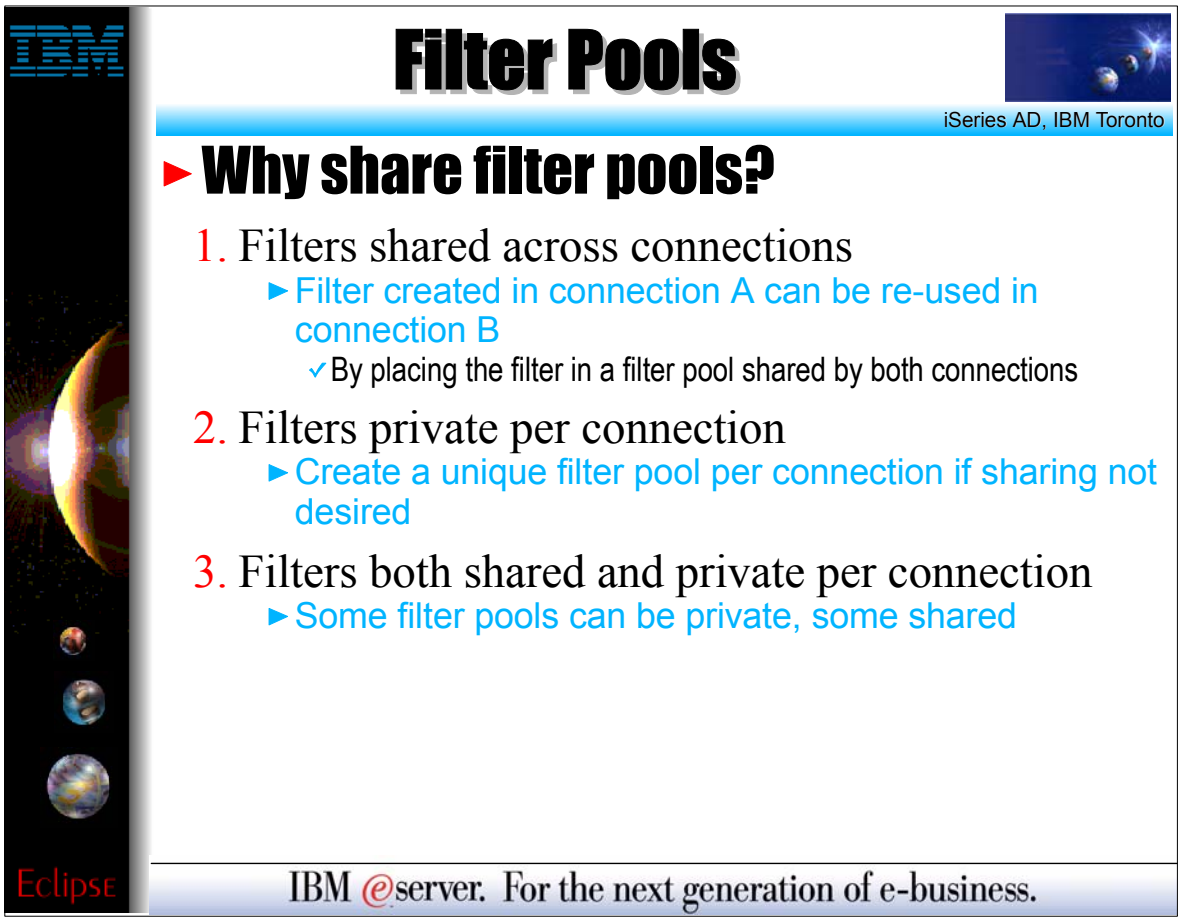
- When a member filter is created, it shows up in the list of existing filters for iSeries Objects.
- You can edit the filter by right clicking on it and selecting the Change... action from its popup menu.
- When a member filter is expanded, all members matching one or more of the filter's filter strings are listed underneath the filter.



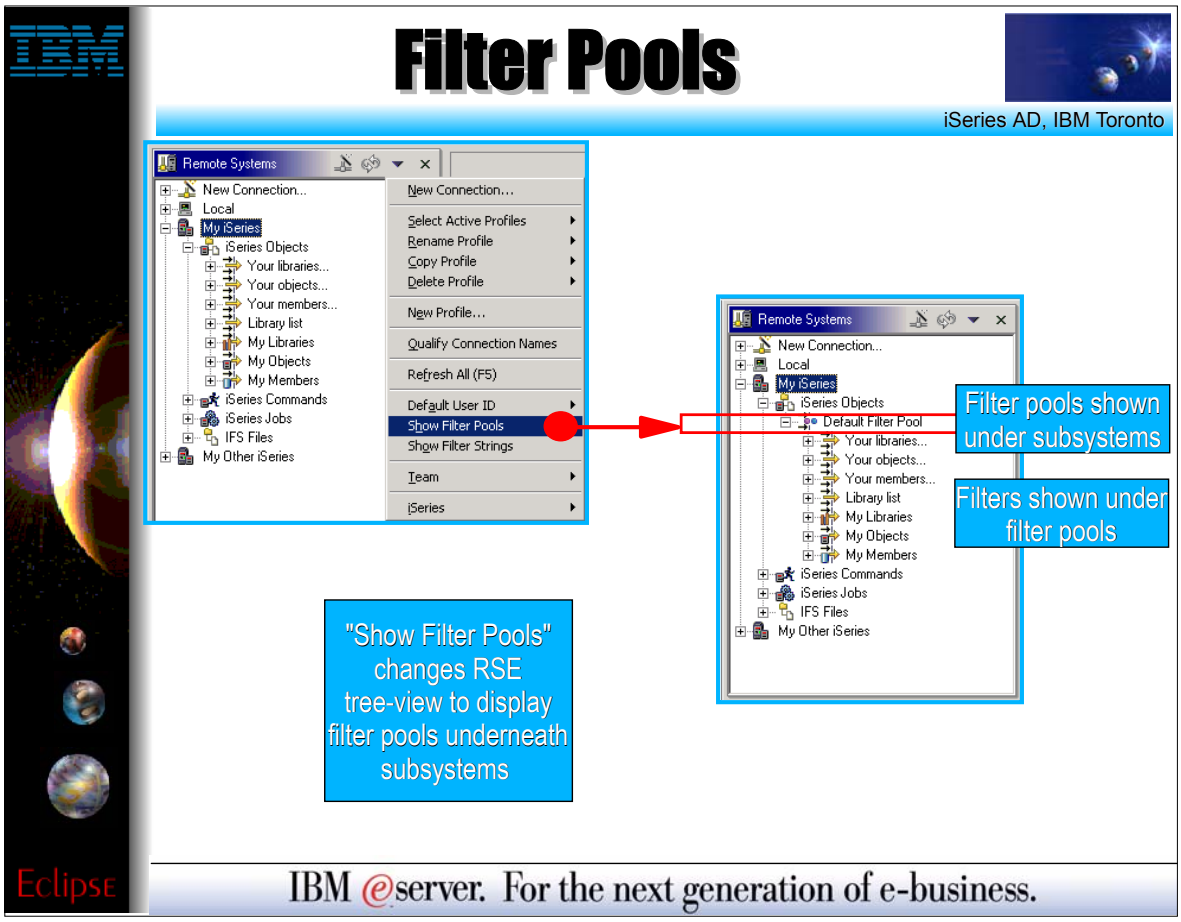
- There is an advanced feature in the RSE for partitioning filters into named collections, called **filter pools**.
- To enable this, you must select the "**Show Filter Pools**" preference
- Actually, all filters are contained in a filter pool. By default, they all go into one IBM supplied filter pool, named **Default Filter Pool**.
- By turning on the Show Filter Pools preference, you will see these filter pools when you expand a subsystem. When you expand a filter pool, you will then see the filters.
- Filter pools can be used to effectively group filters by task, or project, or release, or developer, or whatever you like.



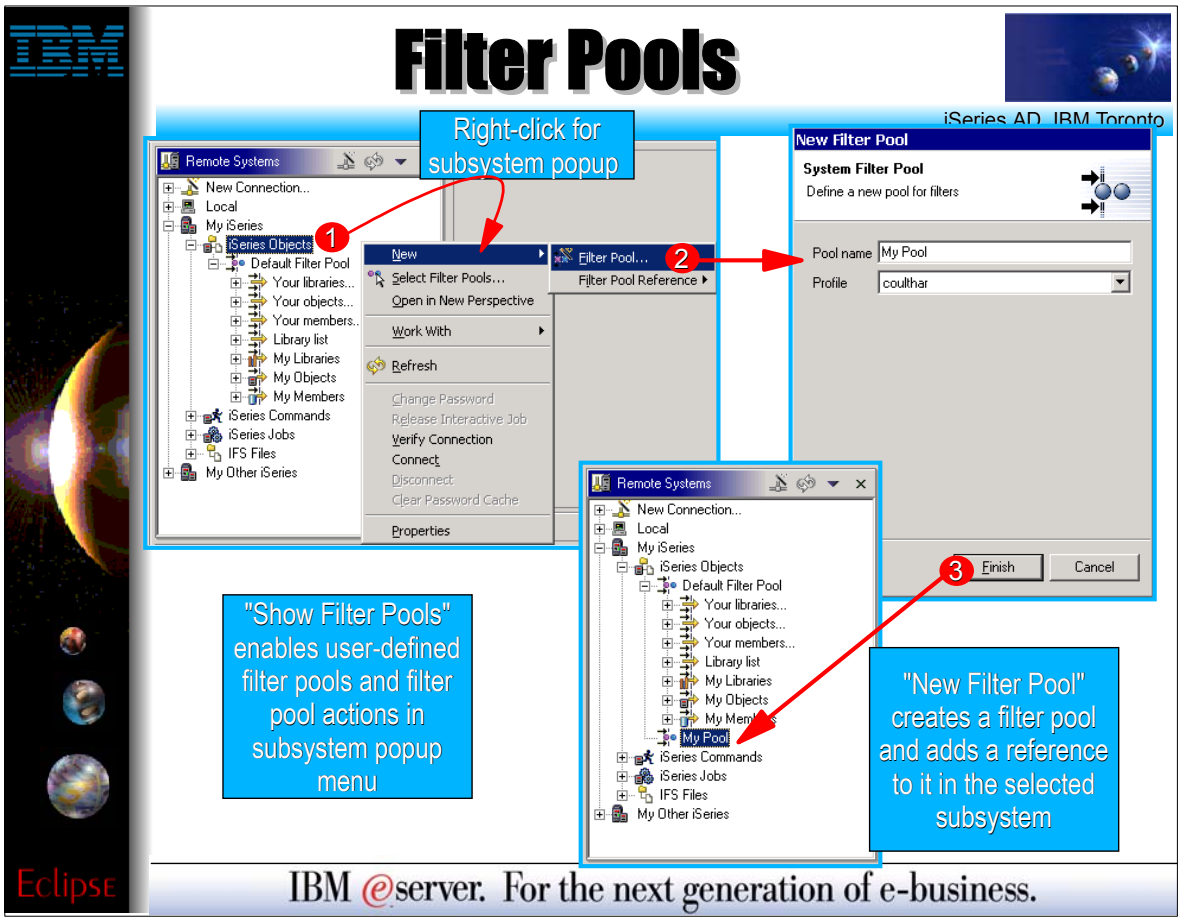
- Subsystems within each connection do not actually contain filter pools per se. They contain references to filter pools.
- Every connection you create is supplied with a single reference to the default filter pool.
- You can easily add references to other filter pools for any connection.
- When filters are added, changed or deleted within a filter pool, these changes are reflected in all connections that reference the filter pool. Or more accurately, these changes are reflected in the subsystem that references the filter pool within all connections. It is important to know that each connection has a unique subsystem instance.
- The diagram shows two connections and three filter pools. The first connection contains only the IBM supplied reference to the default filter pool. The second connection also contains a reference to another filter pool. Filters in Filter Pool 2 only appear for Connection 2 in this example.



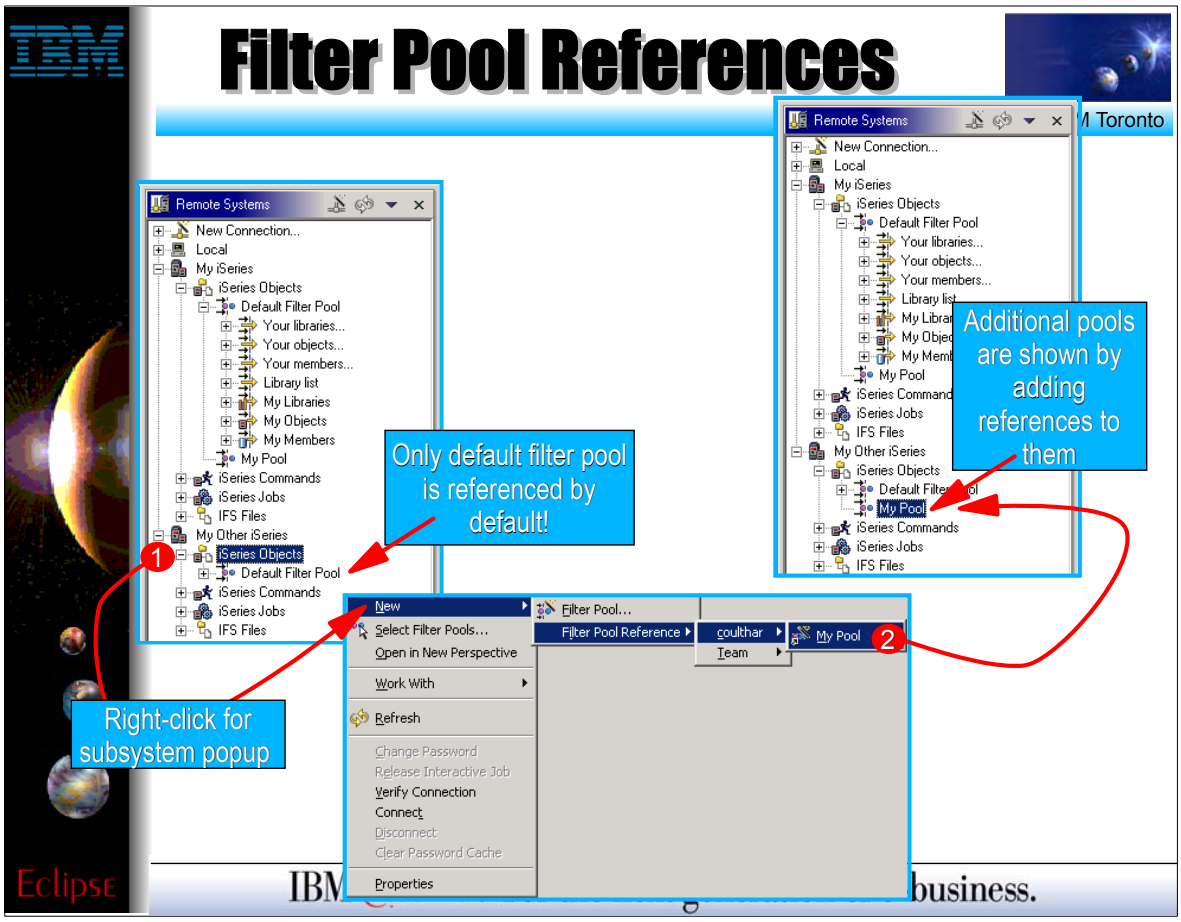
- There are good reasons for storing filter pool references, versus actual filter pools, within each connection.
- Consider creating a connection, then adding a number of filters to it:
- If you delete the connection, would you expect the filters to be deleted too?
- If you create a second connection, would you expect to be able to see the filters from the first connection?
- If you do see the filters in two connections, if you change a filter in one connection, do you expect it to be changed in the other connection?
- By using filter pools and filter pool references, we get the most flexibility:
- 1. You can share filters across connections by referencing the same pool in multiple connections.
- 2. You can have filters unique to a connection by having a unique filter pool only referenced by that connection
- 3. You can choose to reference multiple filter pools in one connection, so some filters can be shared between connections while others are unique to a connection. This is most powerful when used with team support that we describe later.
- In all cases, there is only a single copy of each filter, and when changed all connections referencing its parent filter pool see that change.
- Filters are never deleted implicitly, for example when a connection is deleted. You must explicitly delete the filter, or its filter pool.



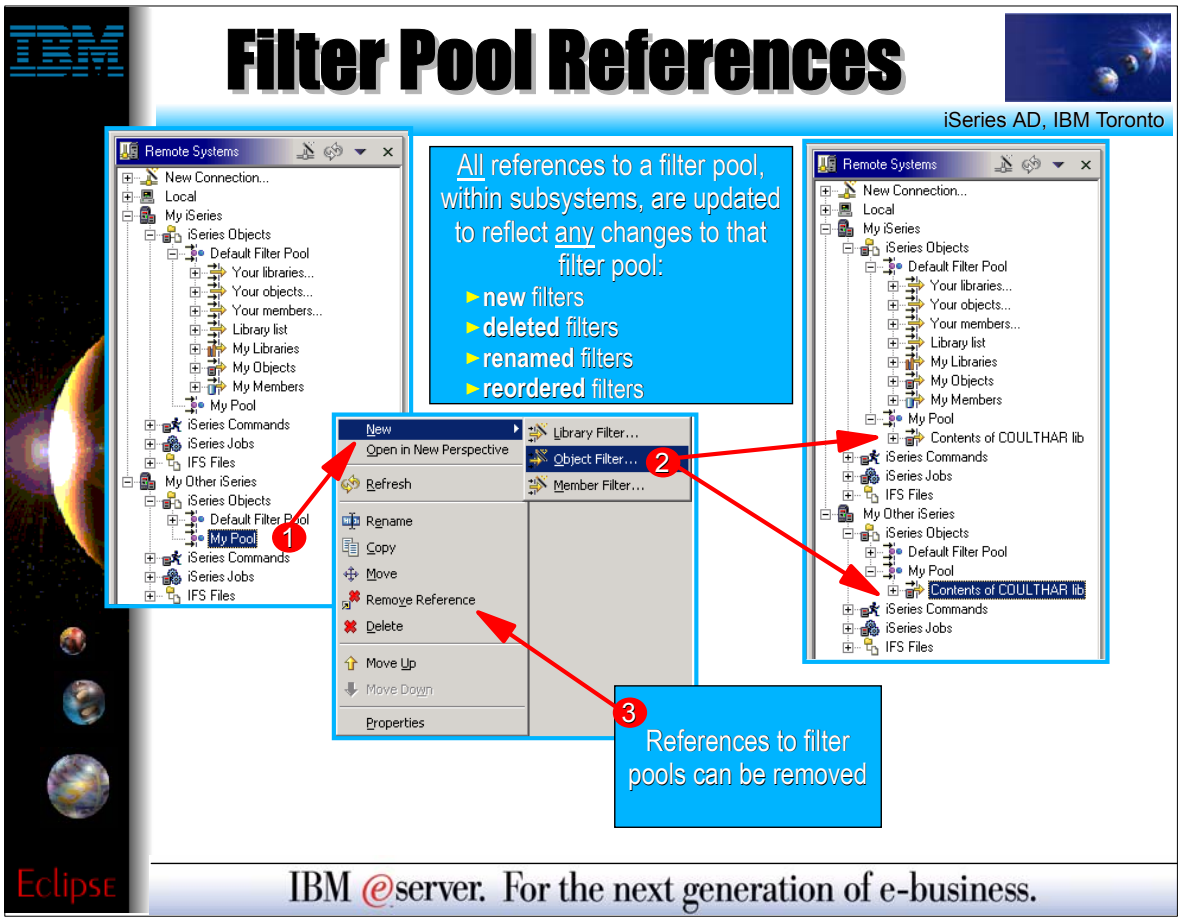
- Here we see how to turn on the Show Filter Pools preference.
- Once enabled, when a subsystem is expanded you see not the filters, but first the filter pools referenced by ***this*** subsystem in *this* connection.
- When a filter pool is expanded, then you see the filters within that filter pool.



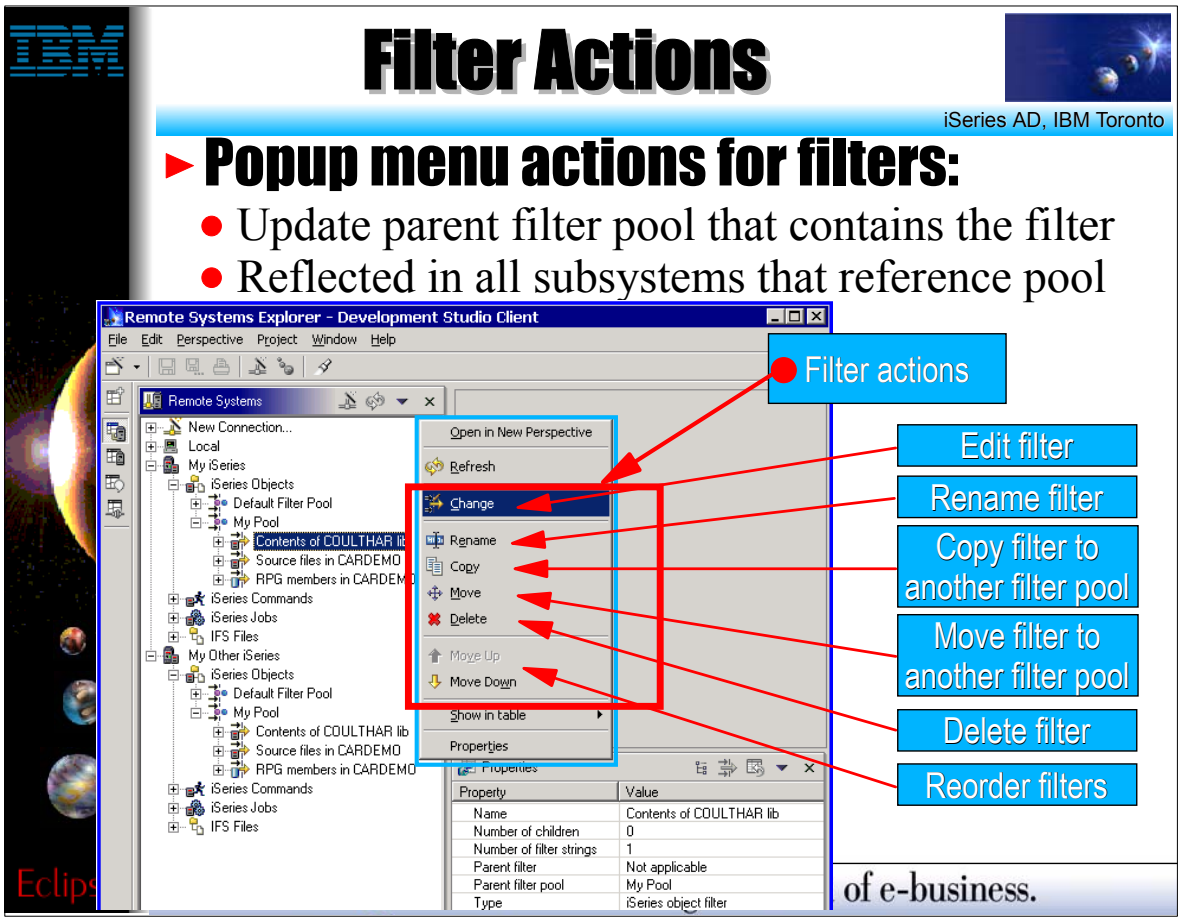
- To create a new filter pool, right click on the subsystem, then select **New -> Filter Pool...**
- All you need to specify for your new filter pool is a unique name. This is what you see when the subsystem is expanded. Never mind the Profile prompt for now, we will explain that later. It is defaulted so you need not think about it.
- When Finish is pressed, the filter pool is created, AND a reference to it is added to this subsystem in this connection.
- No other connection has a reference to this new filter pool yet!



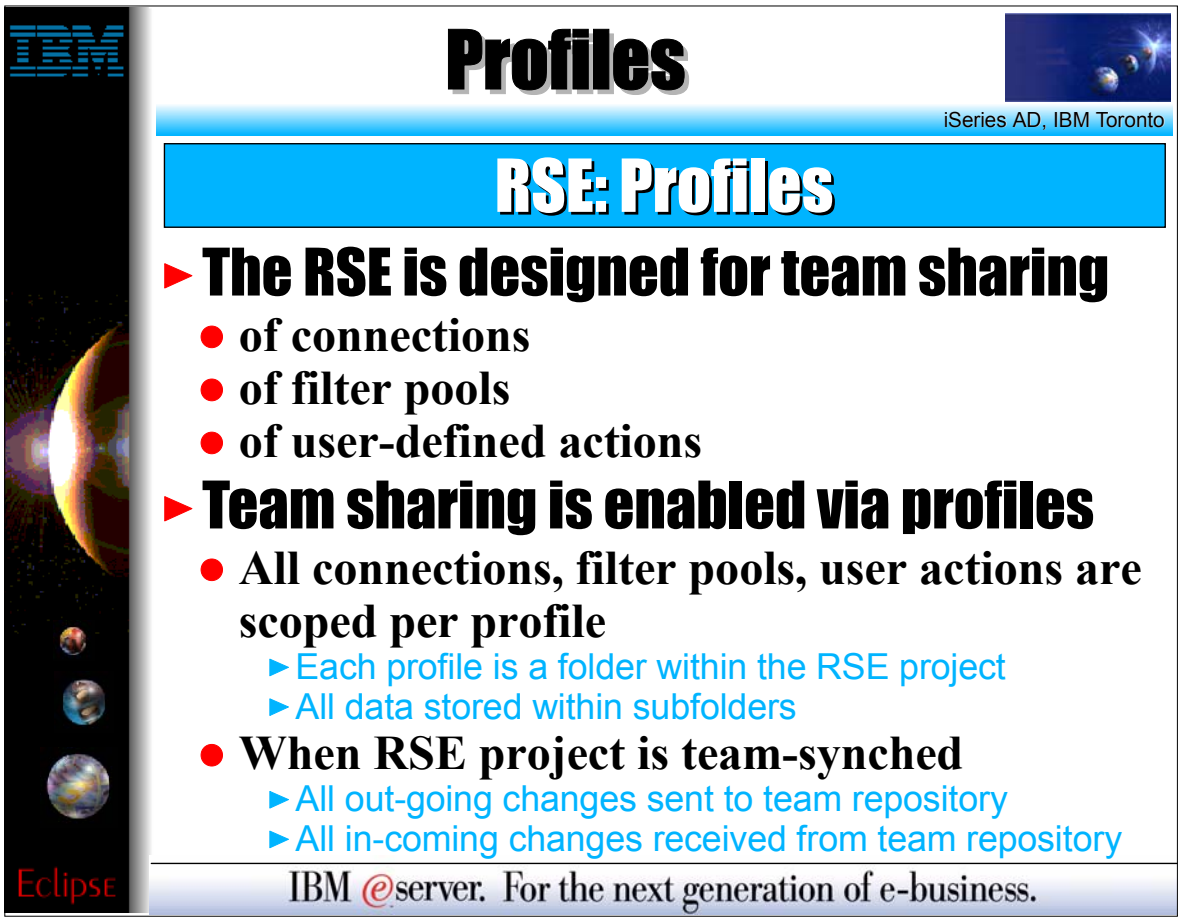
- To enable another connection to see the filters we will add to this new filter pool, you must explicitly add a reference to this filter pool in that other connection.
- Right click on the iSeries Objects subsystem for the other connection, and select the **New -> Filter Pool Reference** menu. Select the filter pool, within the profile it was created in (by default it will be the profile with your workstation's hostname).
- Once a reference is added, this pool will now show up under the subsystem for this other connection too.
- Now let's add some filters to this new filter pool...



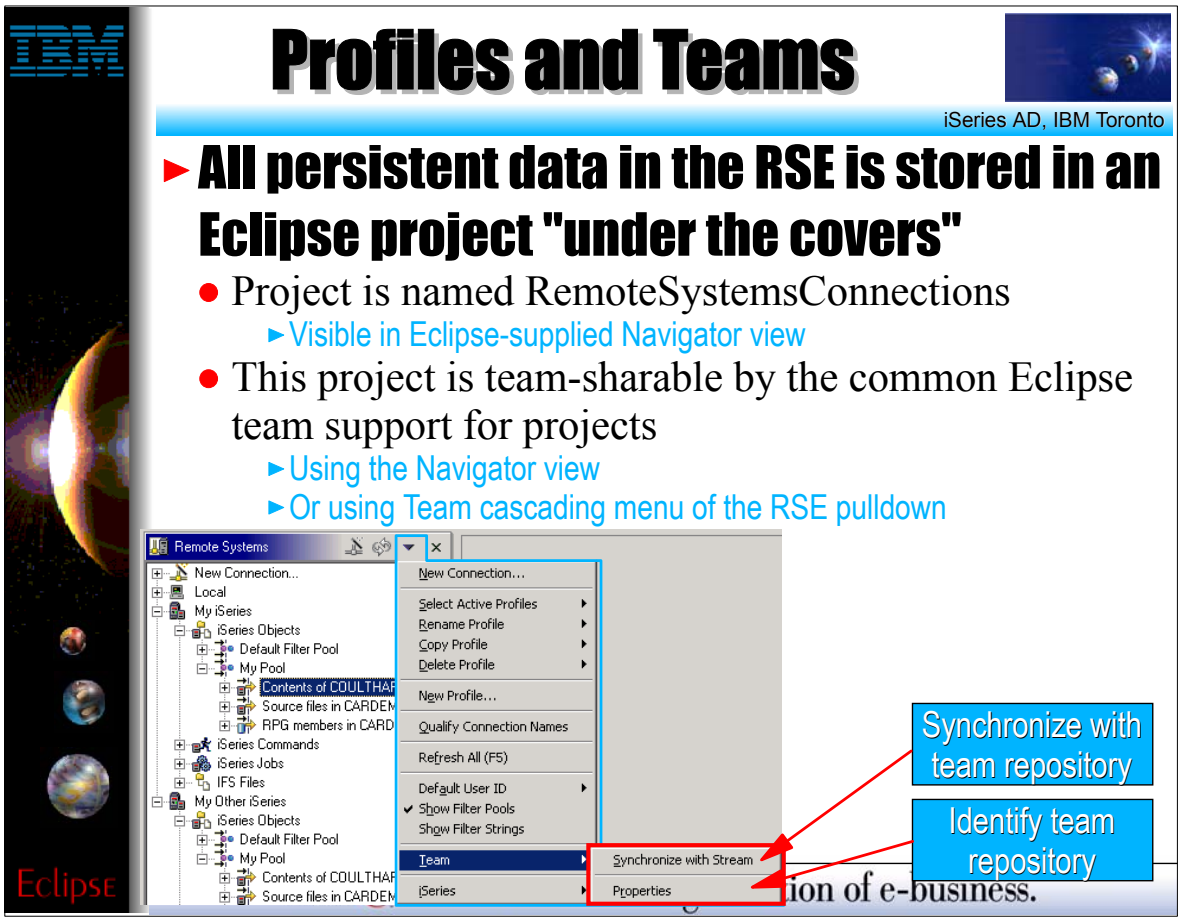
- To populate a filter pool with filters, right click on it and select **New->XXX Filter...**, choosing the type of filter you want to create.
- The New Filter wizard you get is the same as we already described for the Your Libraries, Your Objects and Your Members prompts. Those are simply shortcuts to these menu items.
- As you create filters in the filter pool, they appear in the RSE tree underneatch each reference to this filter pool.



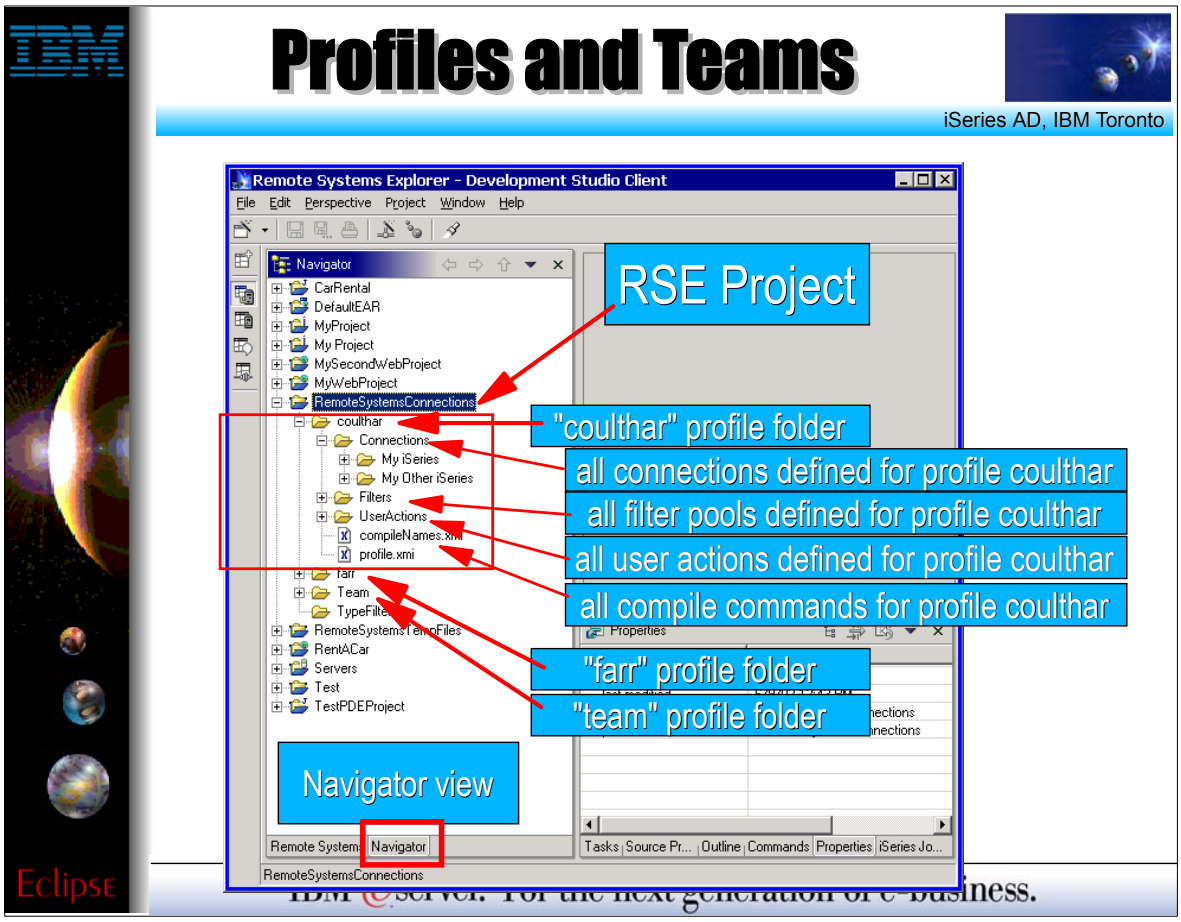
- You can do much with filters by using the right click popup menu.
- For example, you can edit, rename, copy, move, delete and reorder filters.
- All these actions affect the parent filter pool for the selected filters, and all references to that filter pool will be updated automatically to reflect the changes.



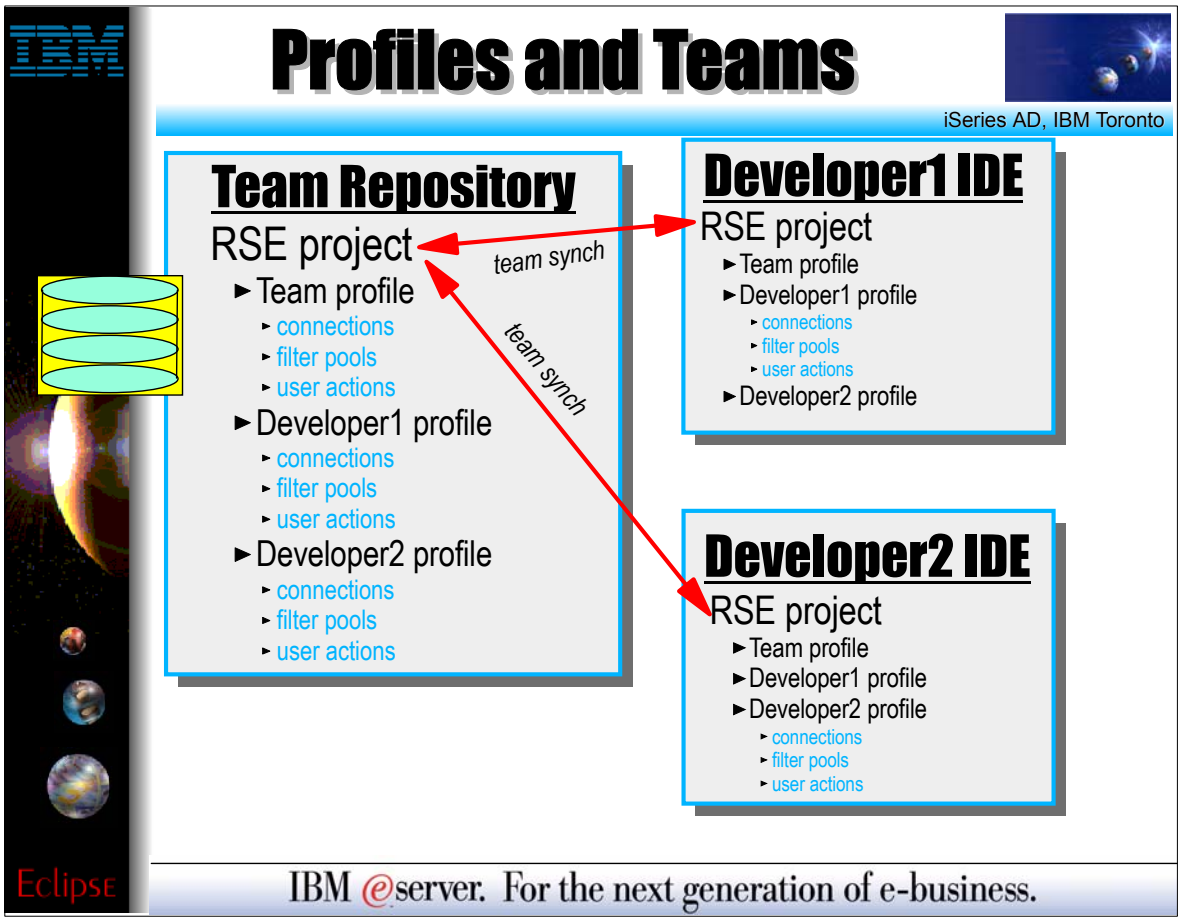
- We see that with the RSE you will be creating connections, filter pools, user defined actions and compile commands.
- In a team environment, we might wish to share some or all of this information to save the effort of each team member having to redundantly create them. This is especially for team members working on a shared task, such as mainenance of an application.
- The RSE leverages the Eclipse team support to enable this.
- The first order of business for effectively enable team support is to allow delineation between "team" information and "private" information per developer.
- The RSE enables this by using profiles. Every bit of information you create in the RSE is owned by a specific profile, and each developer decides which profiles they wish to see the information of.
- Team sharing is done using the Eclipse team support, which uses a central repository, and a "synchronize" action to synchronize individual developer's information with the repository.



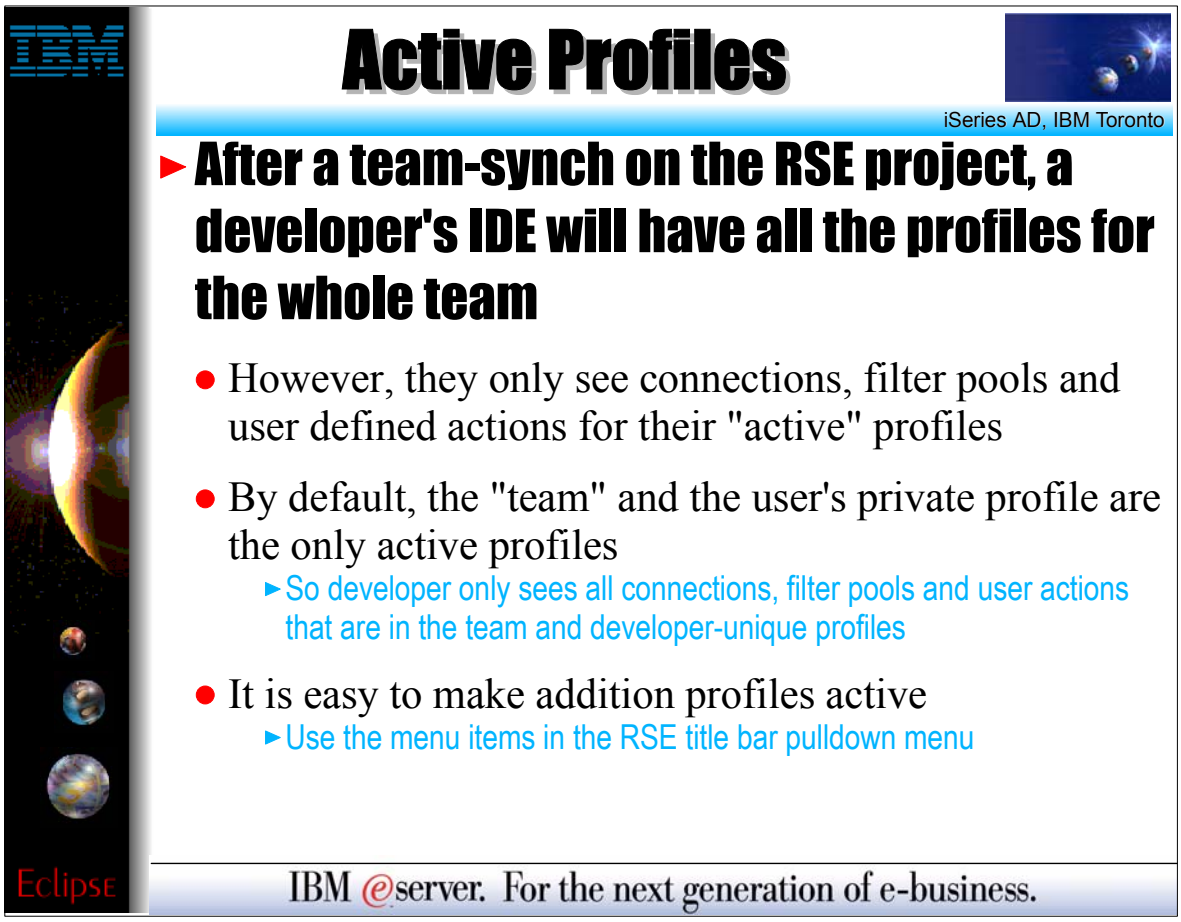
- The RSE uses a single Eclipse project to hold all of the data created within it by a user.
- This project inherits the team support that all Eclipse projects have, which includes the ability to be associated with a central repository (**Team->Properties**) and to synchronized with that repository (**Team->Synchronize with Stream**).



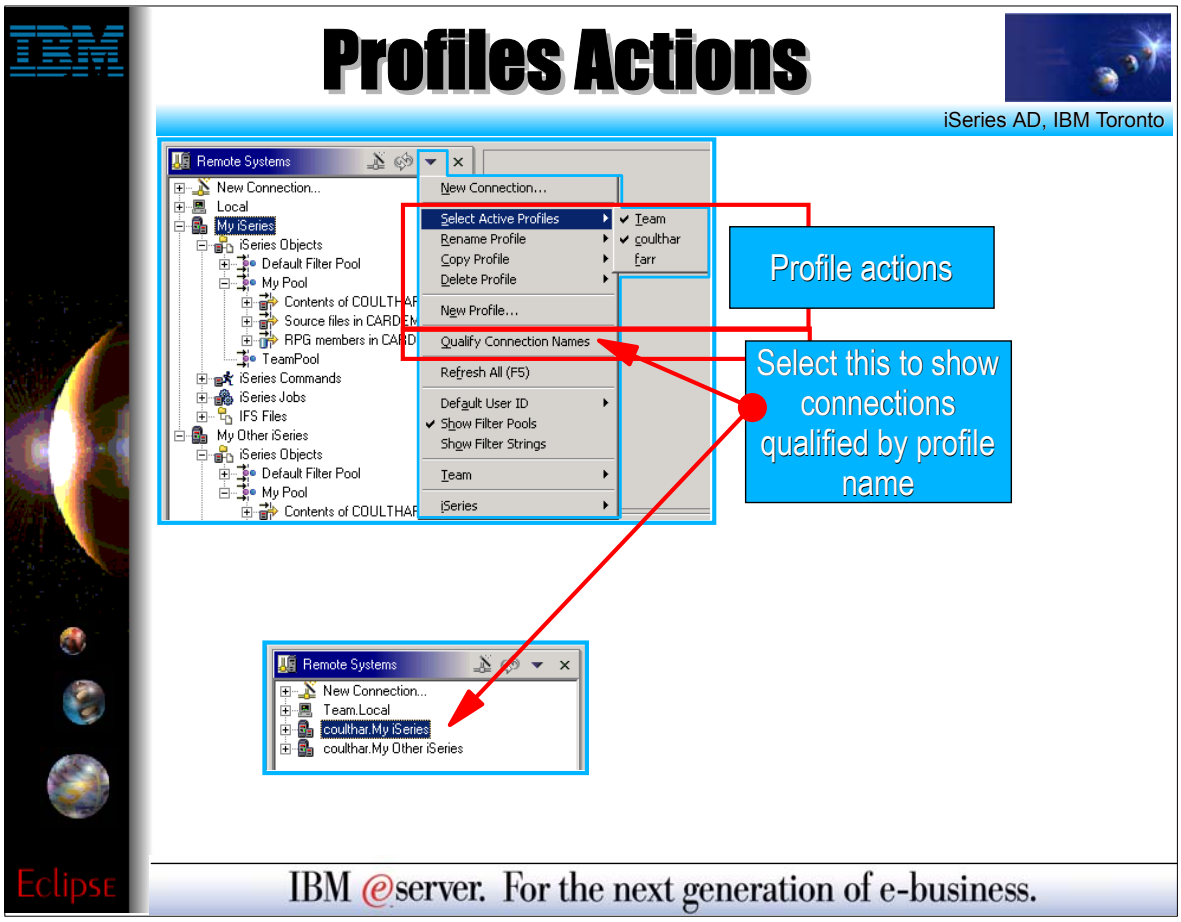
- While you should never have to do this, you can get an understanding of where the RSE stores the information you create in it, by switching to the Eclipse-supplied navigator view and exploring this RemoteSystemsConnections project.
- The most important thing to notice is that the primary folders in the project are the profile folders, and everything else is scoped underneatch those, including connections, filter pools, user actions and compile commands.



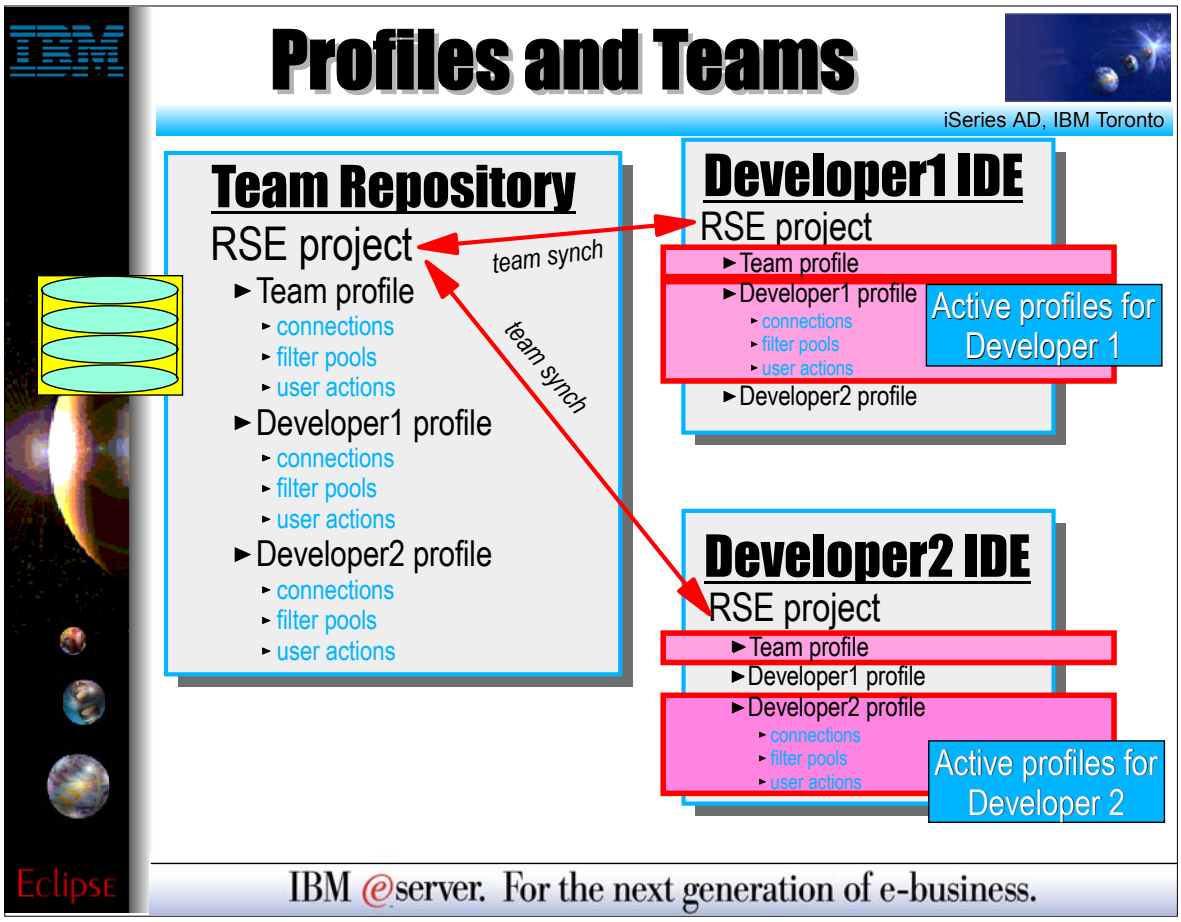
- Here we see pictorially how a central repository relates to each developer's IDE, with respect to the RSE information the team will share.
- You should notice that every developer actually has all the information for every profile within the team, when team support is used. We will see why this is and how it is handled...



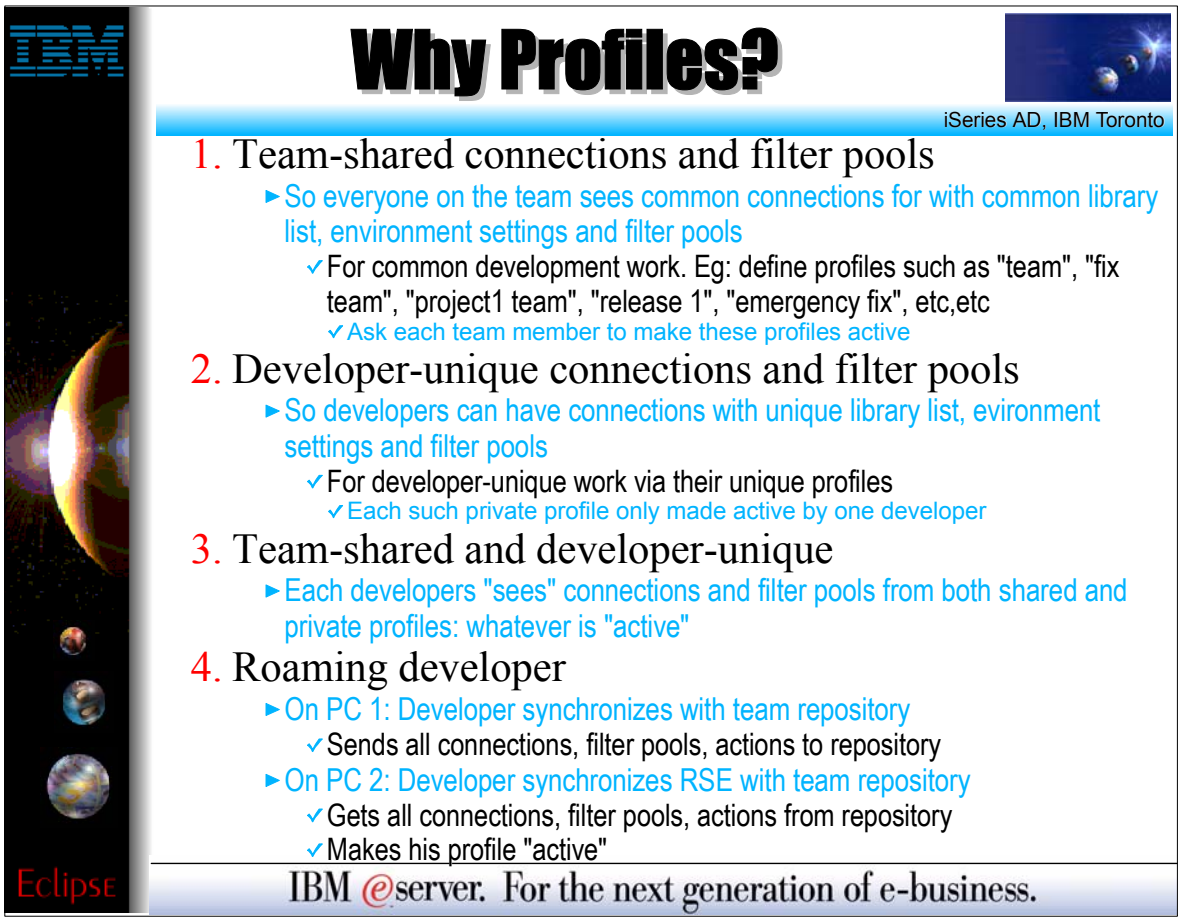
- Because a team-synchronize action copies all the contents of the project from the central repository to the target developer's IDE (and vice versa), after such an action the developer will have all the profiles for all his team members in his/her IDE.
- This could be overwhelming to see connections and filter pools and user actions and compile commands for the whole team.
- So, by default the RSE only shows a subset of these. It shows only the information for those profiles which are "active", which by default is the "team" profile and the unique profile for that developer.
- Should there be a desire to see information from other profiles, this is easily enabled by making those profiles active too. We will see how to do this...



- The pulldown menu from the title bar of the Remote Systems view contains all the profile menu actions.
- From here, you can choose which profiles are active, and you can create, rename, copy and delete profiles.
- A very important preference setting in this menu is "Qualify Connection Names". When toggled on, this changes the Remote Systems view to show each connection prefixed by the profile it comes from. This can be handy when working with both shared and private connections.



- Here we see how active profiles keep team members from being flooded with the information from all other team members.
- While each workstation has all the connections, filter pools, user actions and compile commands for all team members, by default each developer only sees their own and those that were created in the team profile and hence intended to be seen and shared by all.

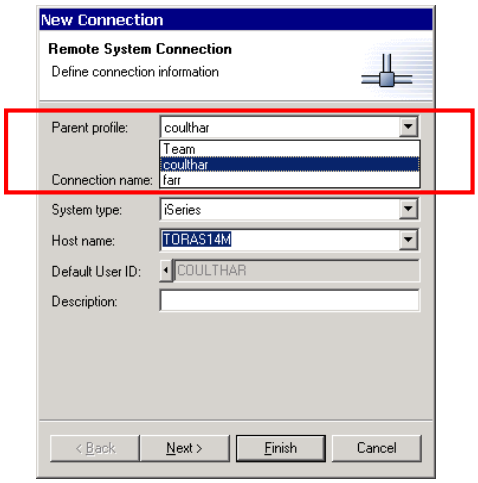


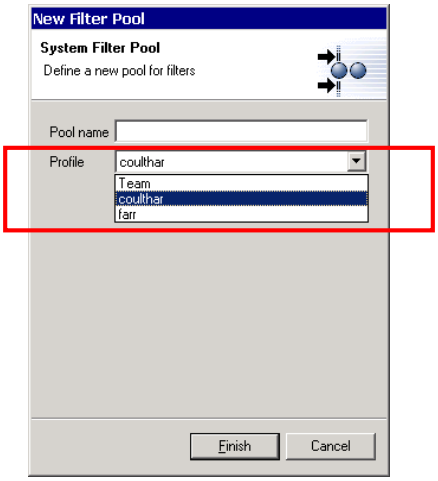
- There are many good reasons for the use of profiles within the Remote Systems explorer, which can be netted out to:
- a. They allow developers to see both team-shared and private information, combined as one set of information.
- b. They allow developers to roam from PC to PC, knowing they can always access their information by simply synchronizing with the team repository and setting their profile as active.

Active Profiles

iSeries AD, IBM Toronto

- When creating connections and filter pools you must specify the active profile to contain the new connection or profile

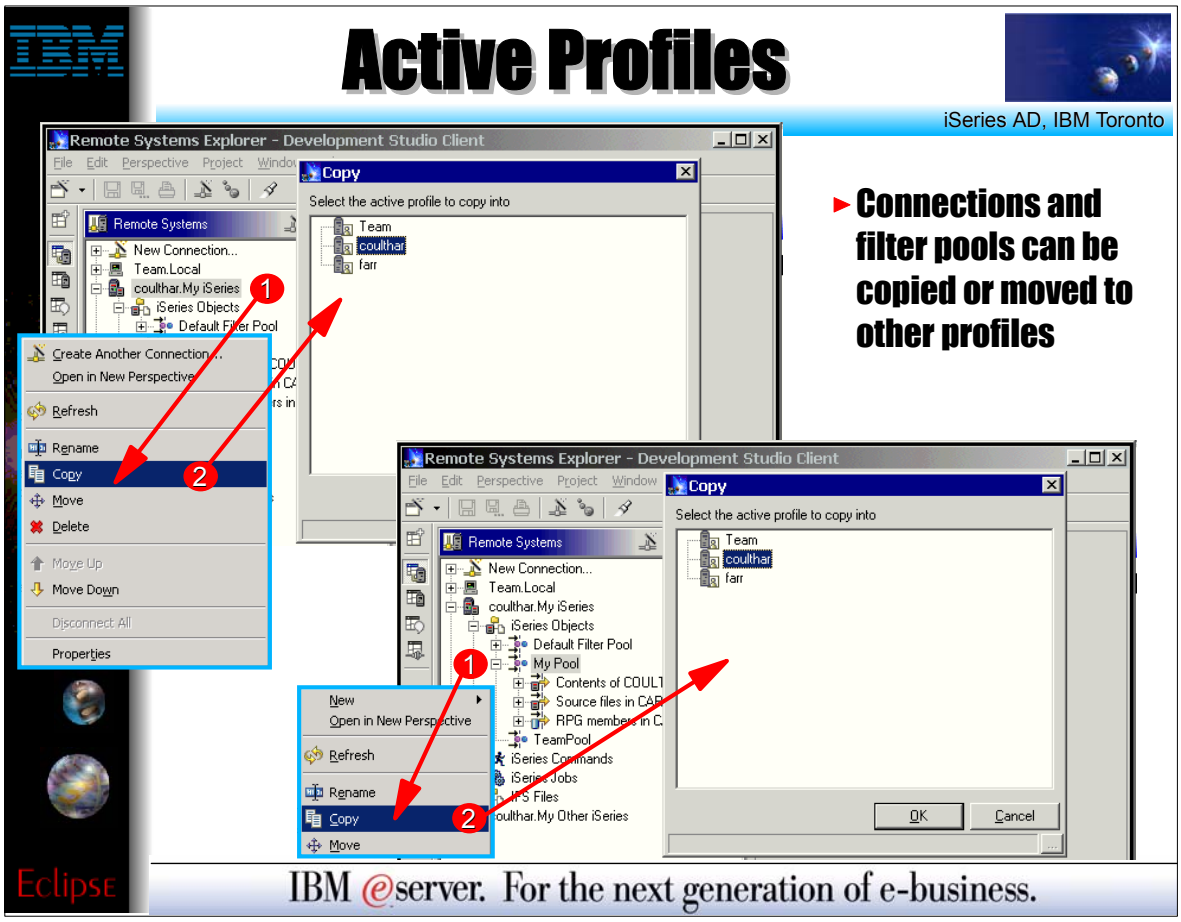




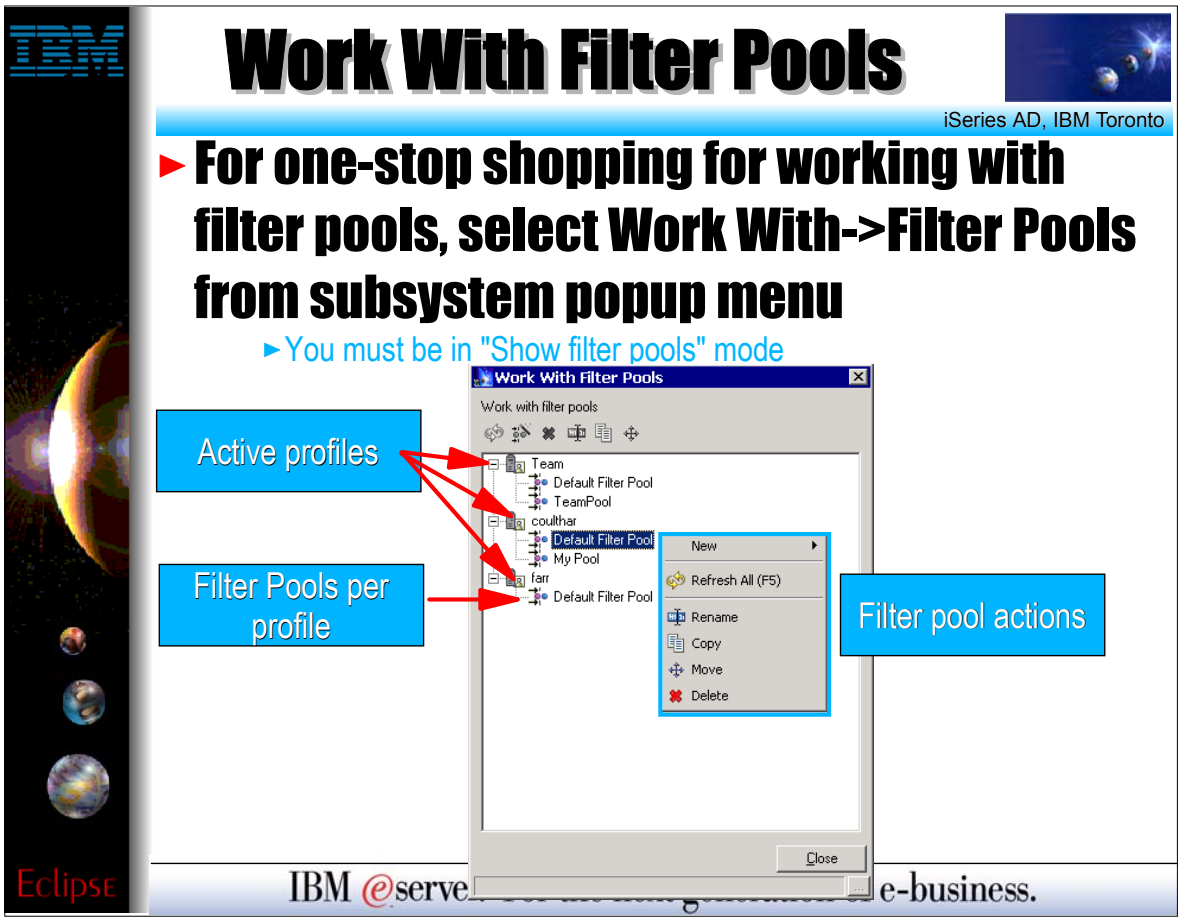
Eclipse

IBM @server. For the next generation of e-business.

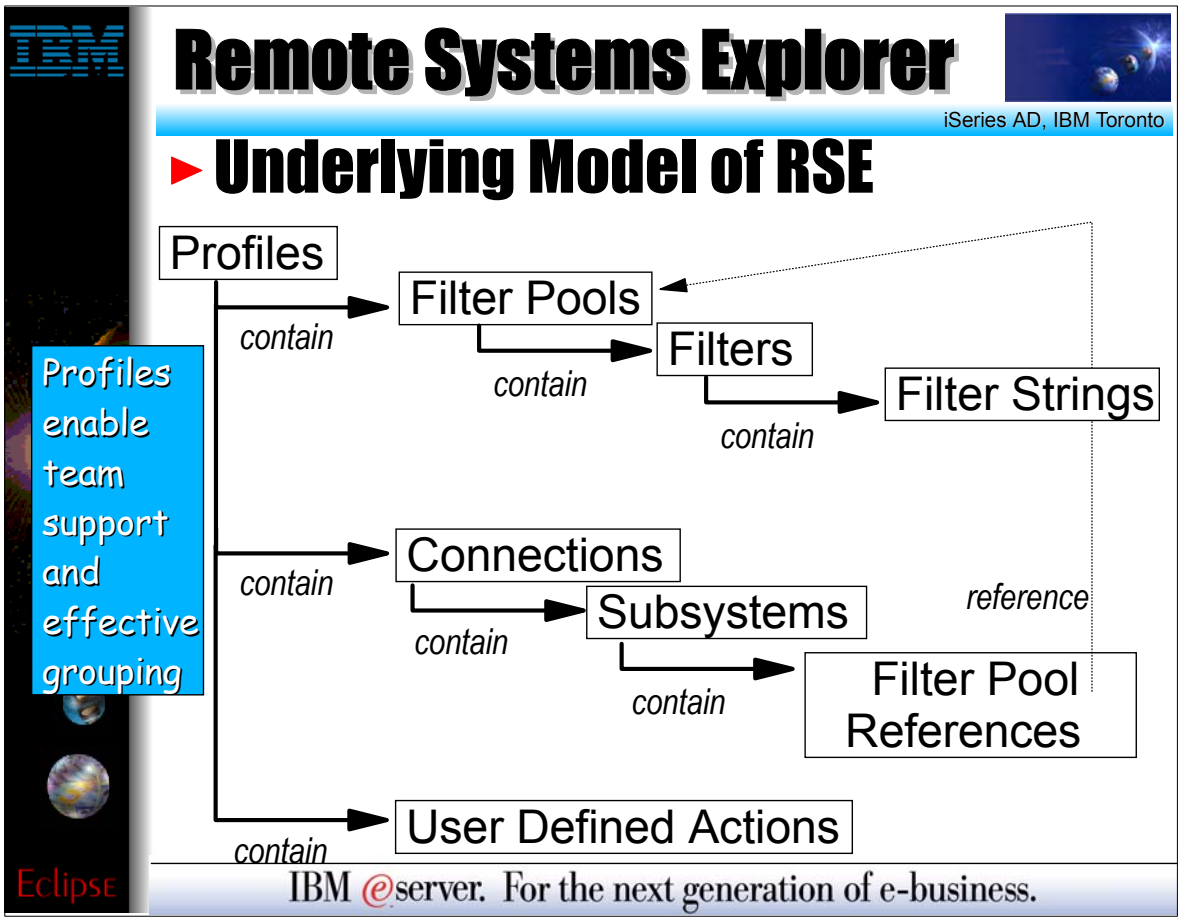
- We see now why both the connection and the filter pool wizards contain a dropdown for selecting the target profile.
- These both default to the first non-team active profile.



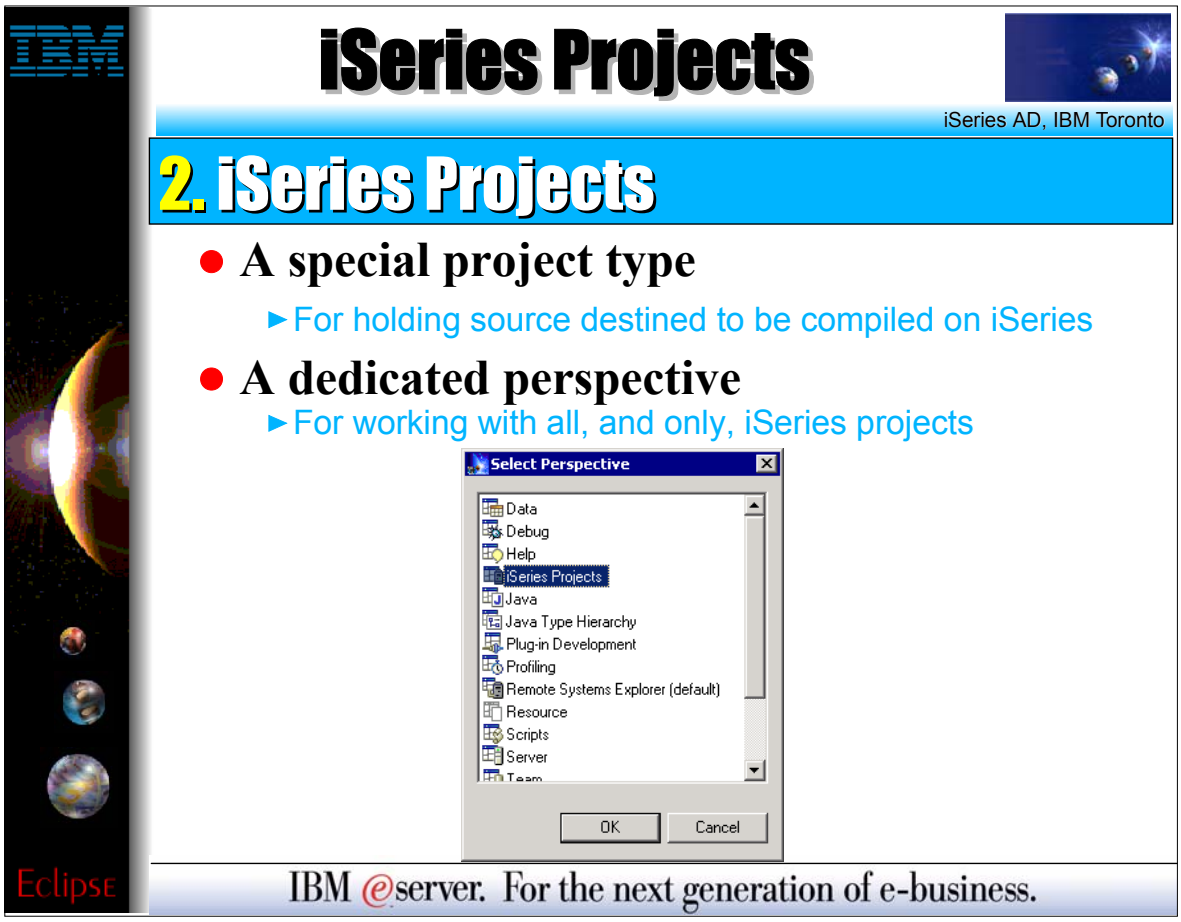
- Here we seen how we can use the copy and move actions to move information between profiles.



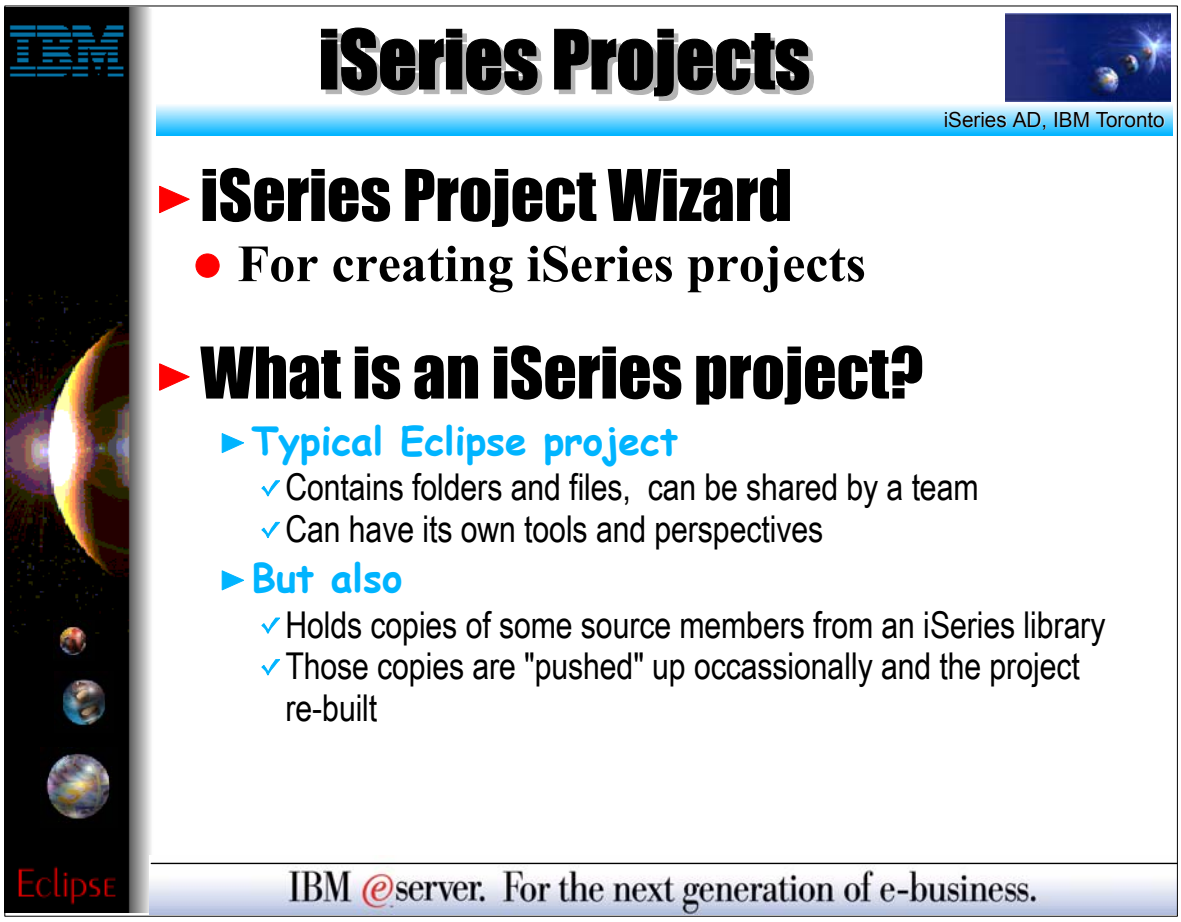
- Finally, now we can return to the topic of Filter Pools to finish it.
- There is a handy Work With Filter Pools action per subsystem that allows working with all the filters pools in all the active profiles, from one dialog.
- In this dialog, you will see all the filters pools within each active profile. From here you can create, rename, copy, move and delete filter pools.



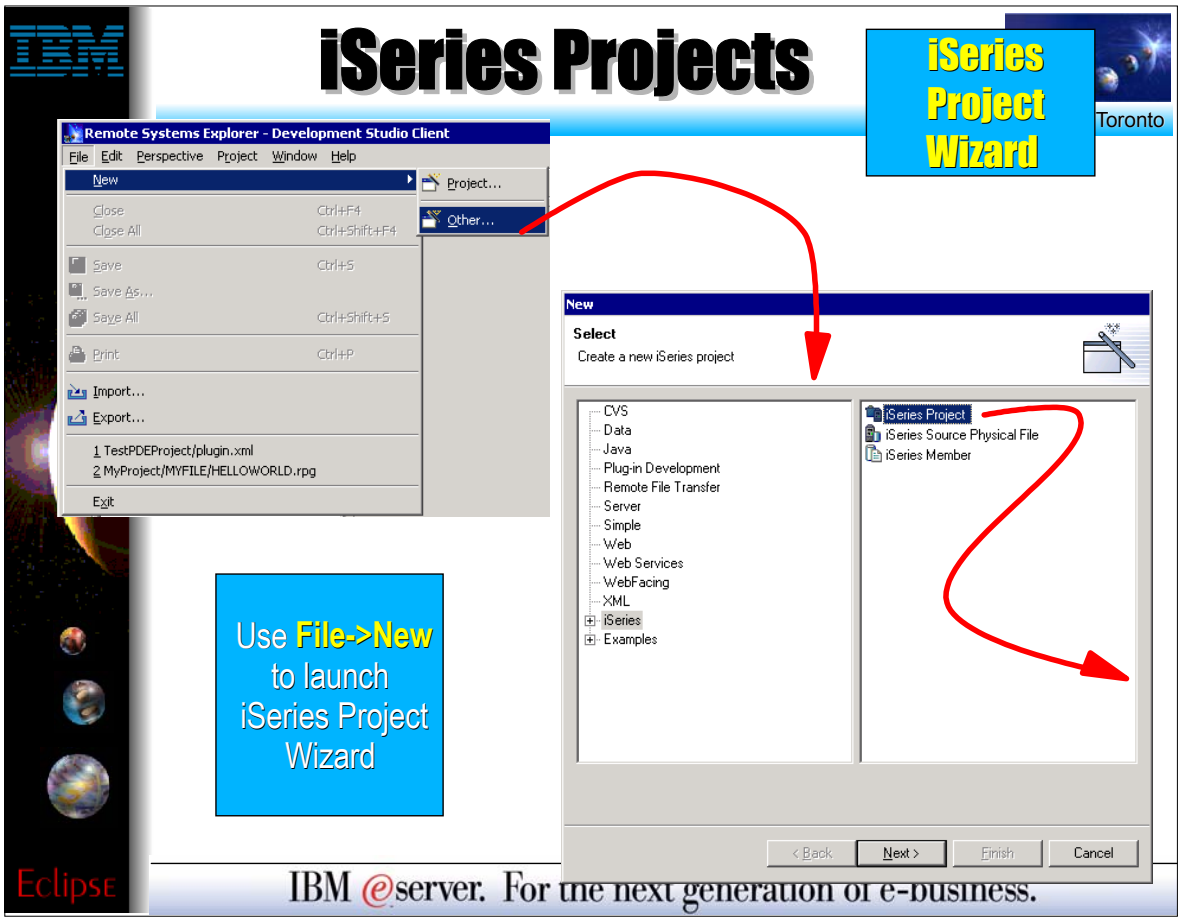
- This is an architecture chart showing the underlying model of all the information that the RSE maintains.
- For those with an architectural bent, this may help you to visualize all the pieces we have been talking about.



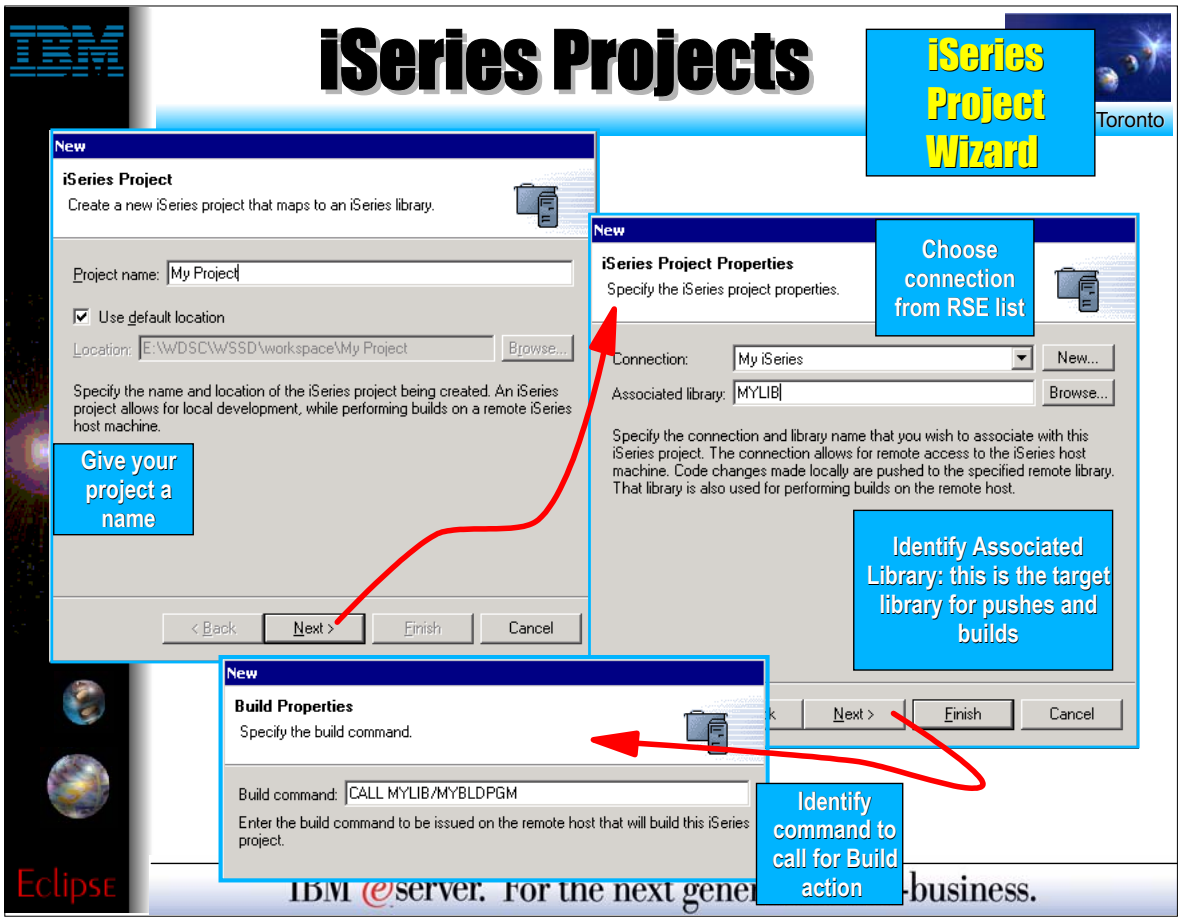
- This concludes the RSE part of the presentation.
- Now it is time to turn our attention to the second major area of support for RPG and COBOL programmers in WDSc.
- This is the iSeries projects support.
- There is a special project type designed to hold source that is destined to be compiled and run on an iSeries. There is a wizard for creating a new iSeries Project.
- There is a special perspective for working with iSeries projects.



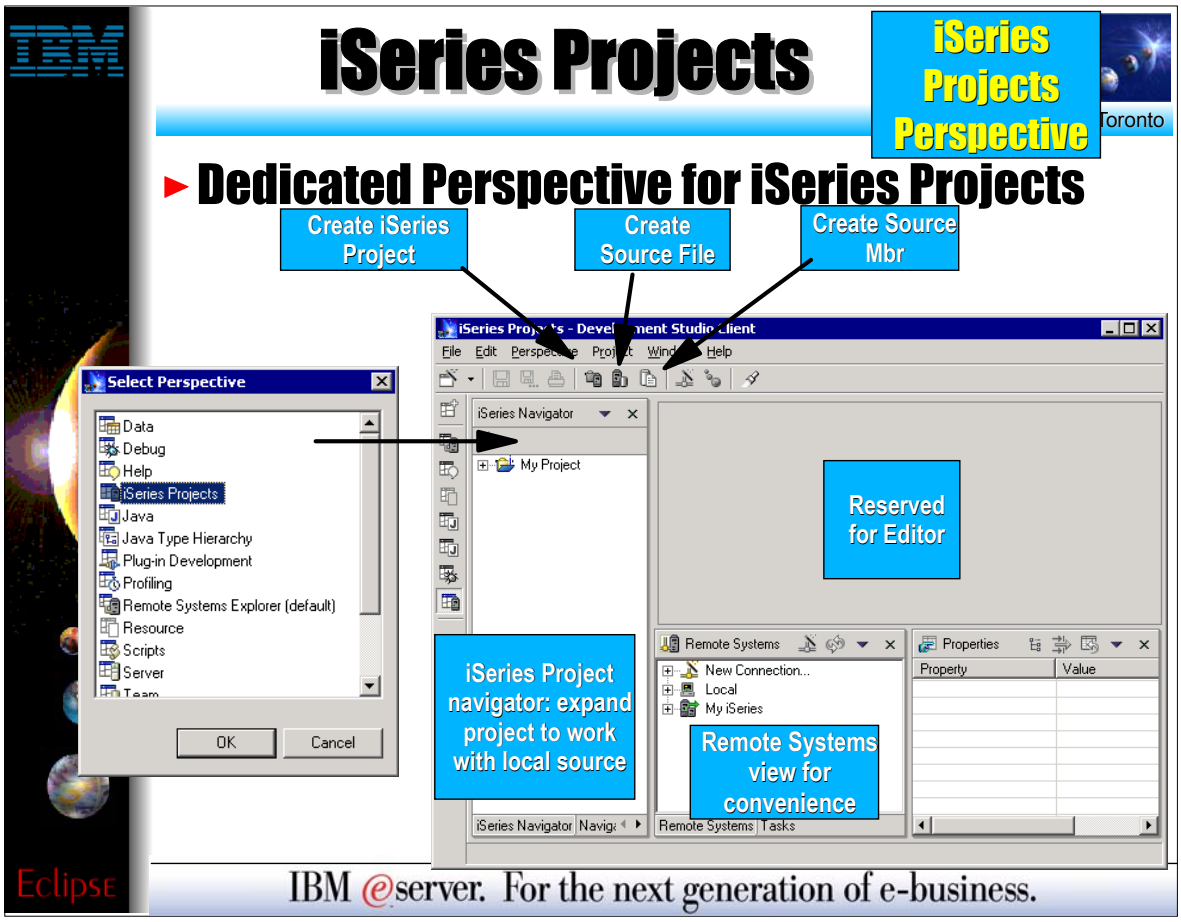
- An iSeries Project is a typical Eclipse project. This means it contains folders and files that are in the local file system of the IDE, but these can be synchronized with a central repository for team support.
- Since the files are local, there is actions to support "pushing" those files up to an iSeries library, where they become source members. This is done prior to compiling or "building" the source for the purpose of testing.



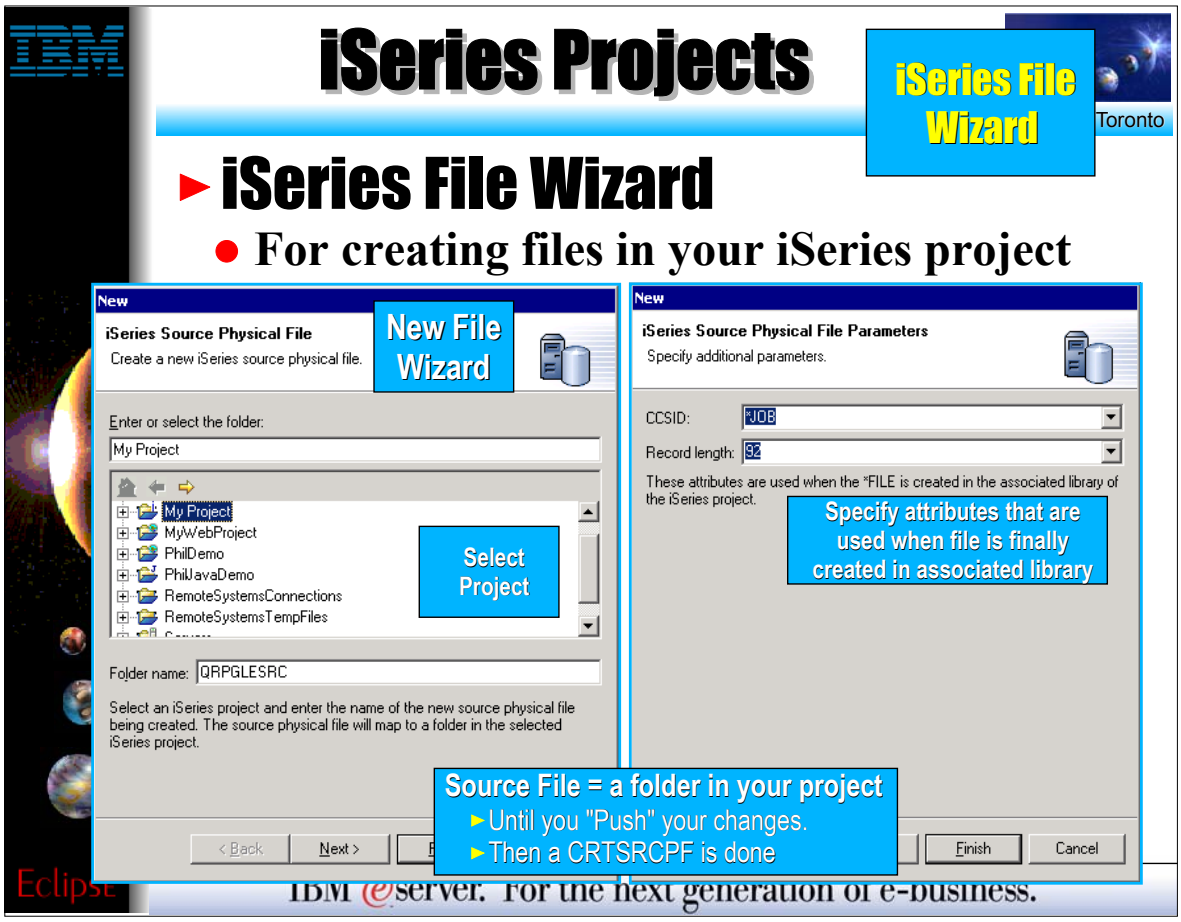
- Here we see how to create a new iSeries project, using the File menu's New menu item. Select iSeries and then iSeries Project, and press Next.
- This can be launched from any perspective, but will switch you to the iSeries Projects perspective after creating the project.



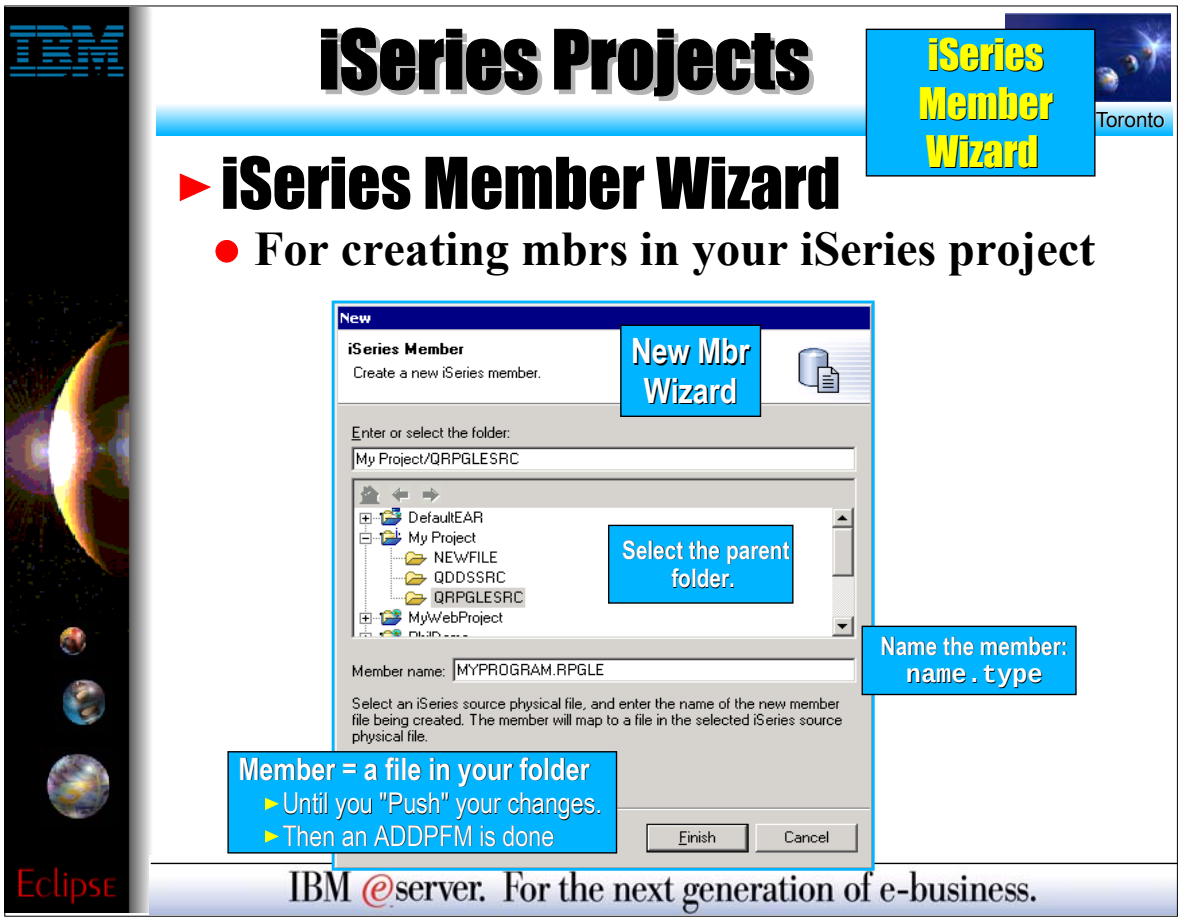
- Here we see the rest of the iSeries Project wizard.
- First, give the project a name. Any name you want!
- Then select or create a connection (from the RSE!) that identifies the iSeries with which this project is associated. Also select the library on that iSeries where the contents of this project will be pushed to.
- Also, specify a CL command to call for the "Build" action. If you don't have such a CL command, enter a dummy value or leave it blank.



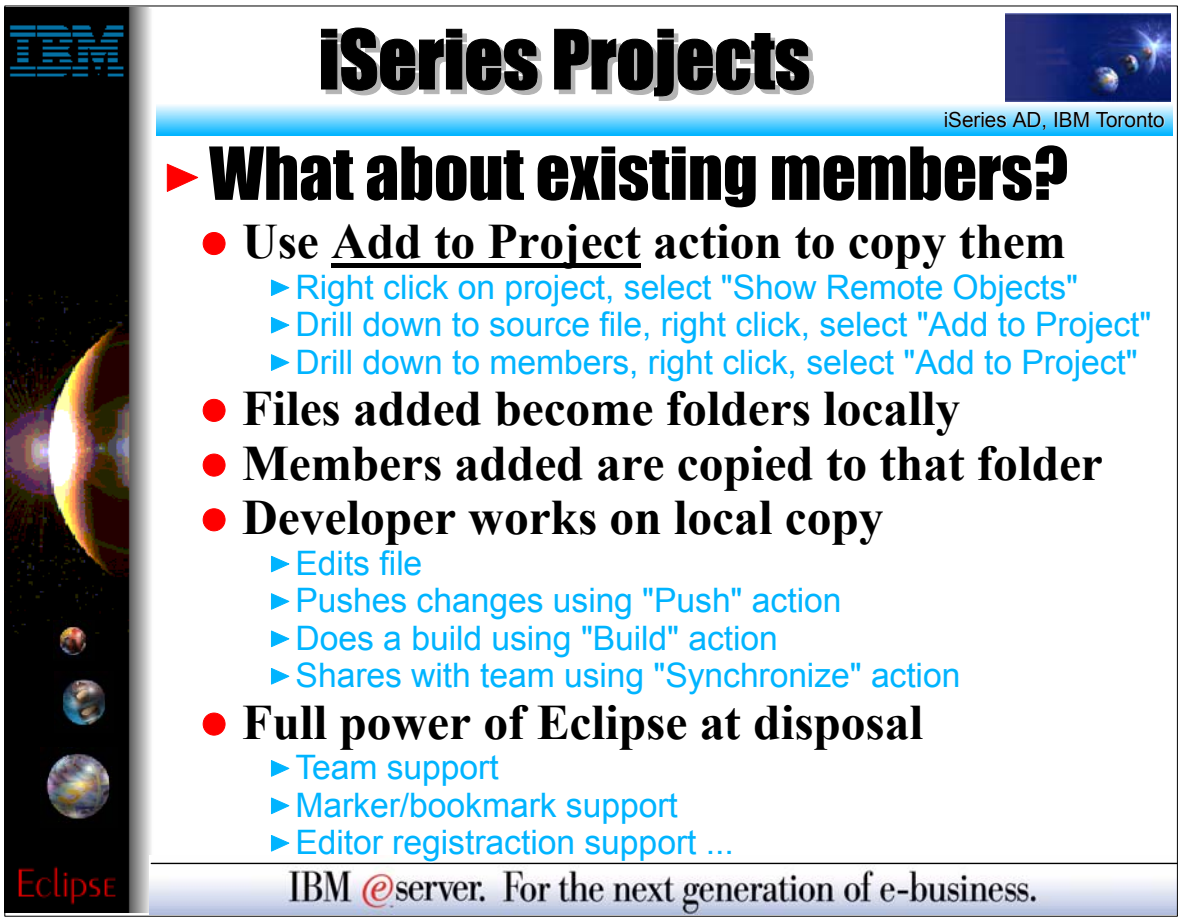
- Here we see the iSeries Projects perspective, which you can explicitly open.
- The primary view is the iSeries Navigator that allows exploration of all existing iSeries Projects.
- There are wizards lauchable from the toolbar for creating a new iSeries Project, or creating source physical files and source members within an existing project.
- The iSeries Projects perspective includes a full copy of the Remote Systems Explorer, as a view, for your convenience. This allows you to easily work with the contents of the associated library while simultaneously working with the local copies of the source within the project.



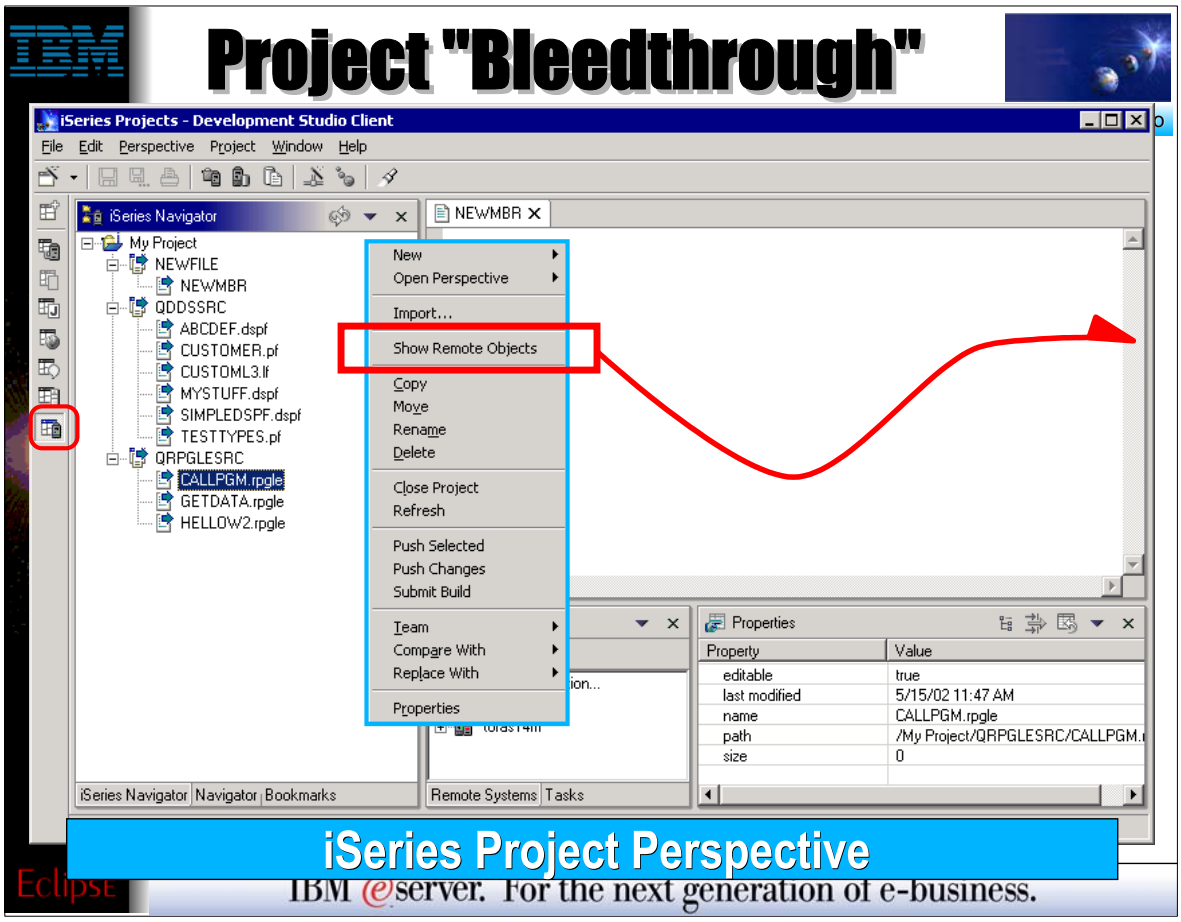
- Here we see the wizard for creating a new iSeries source physical file. Within the project, this is actually a folder. When the project is pushed to its associated library, this result in a CRTSRCPF command being run to create a file with the attributes specified in this wizard.



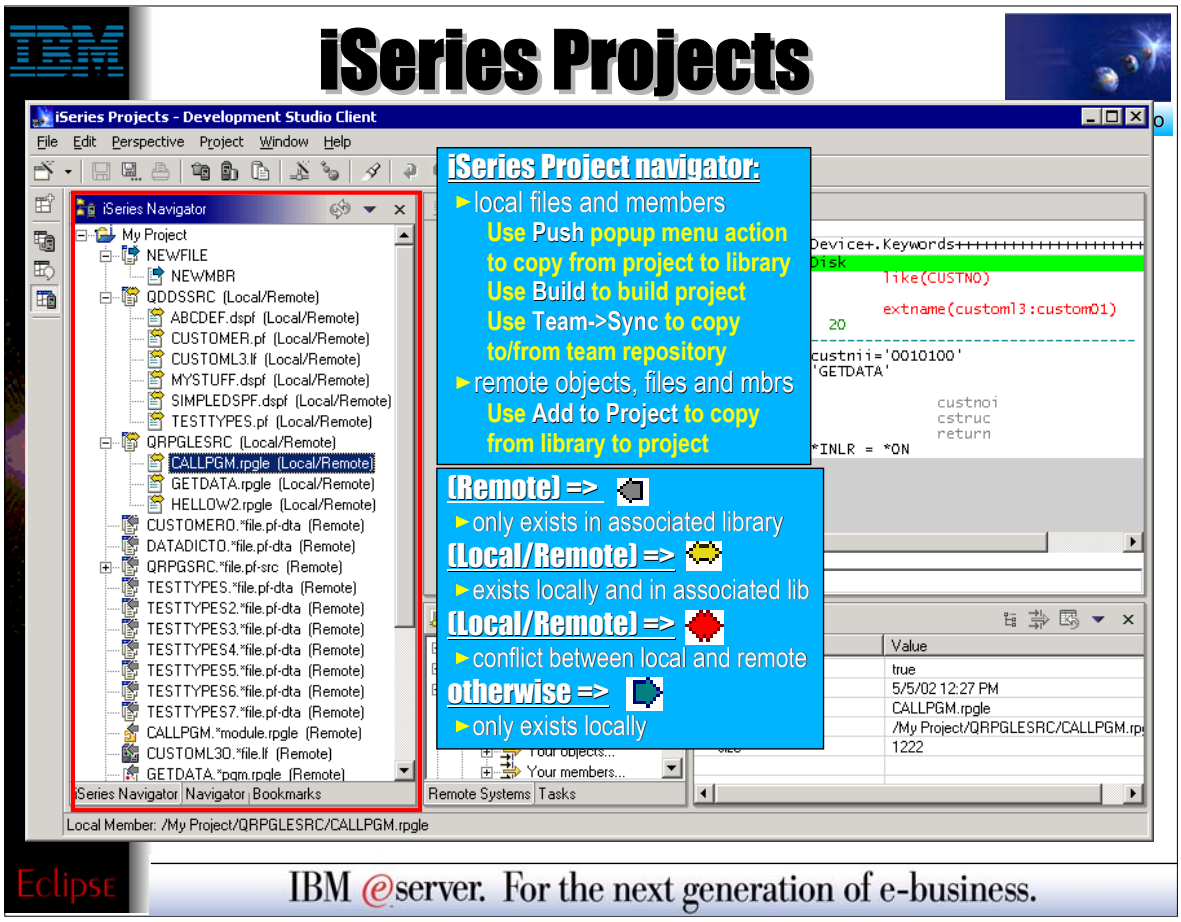
- Here we see the wizard for creating a new iSeries source member within an iSeries project. Actually, it is within a source phyical file (aka folder) within an iSeries project.
- Locally, a member is really a file on disk. The file's extension is the member's type, as in ABC.RPGLE.
- When this project is pushed to its associated library, this will result in an ADDPFM command being run to create the file with the name and type, in its parent file.



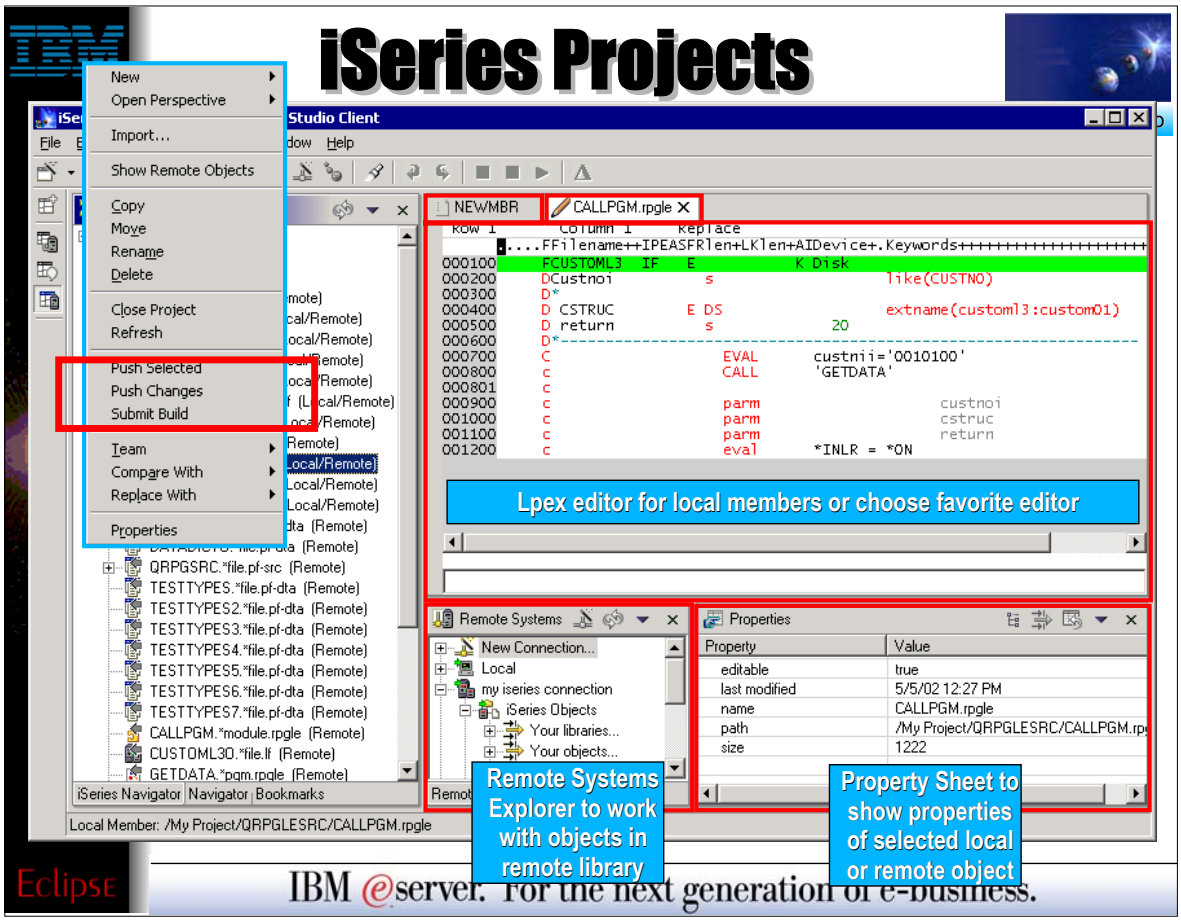
- Often, you won't be creating new files and members from scratch, but rather working with existing files and members.
- You can import these from your associated library into your iSeries project. The files become local folders, while the members become local files within a local folder.
- To import, you have to first select the "Show Remote Objects" menu item from the project's popup menu. This will show the contents of the project's associated library in the iSeries Navigator view. You can select files or members, and right click on them to select their "Add to Project" action. This will copy them down to the local project.
- Once files and members have local copies in the project, you can edit and work with them locally, and when ready push them back to the library where they can be compiled and run.
- An iSeries project inherits all the capabilities of an Eclipse project, includes support for teams, for persistent markers and bookmarks, and for registering your favourite editor per file extension. The latter can't be done for the Remote Systems Explorer, as only editors capable of working on remote source are valid. But for iSeries projects, the source is a local file so any editor can be used.



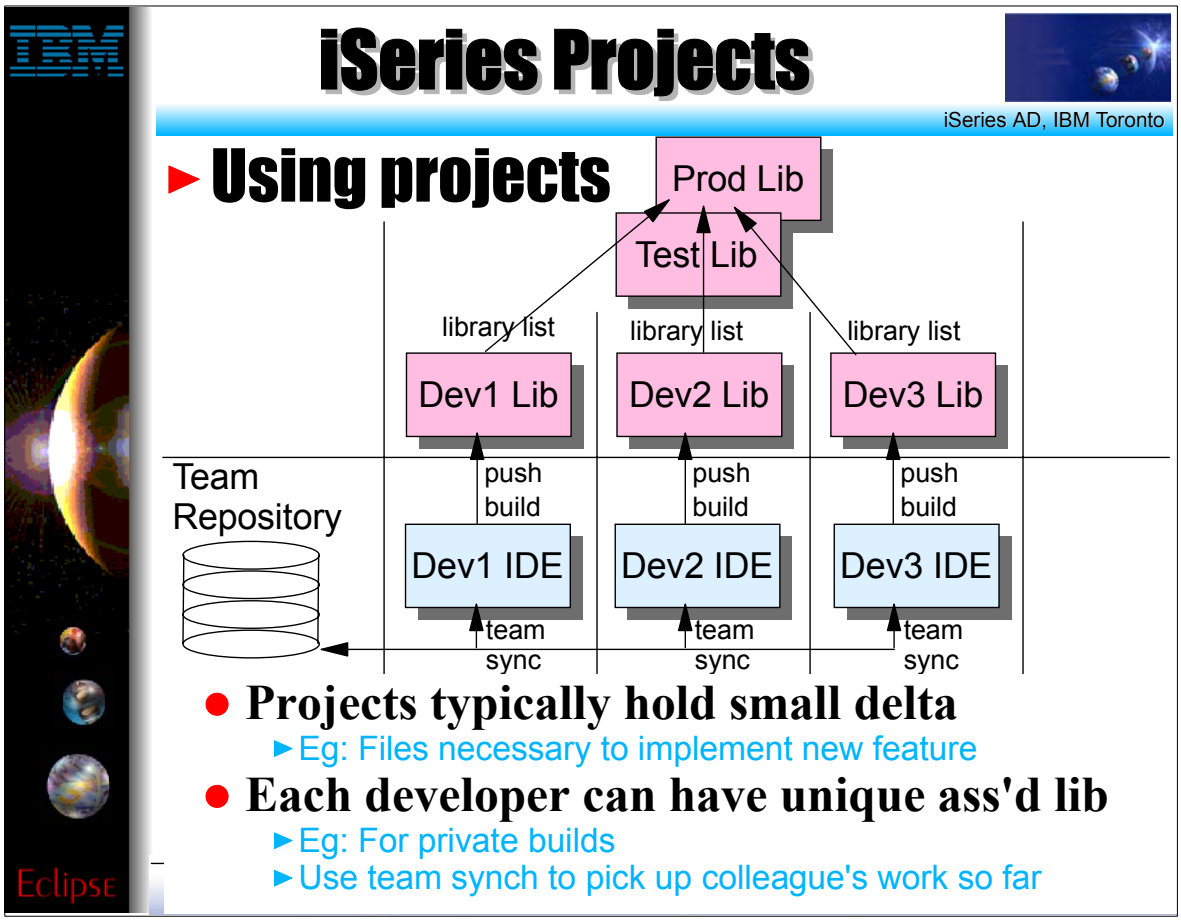
- Here we see an iSeries project in the iSeries Navigator, within the iSeries Projects perspective, that has a number of local files and members.
- Right-clicking on the project or anything in the project, gives a popup menu with the all-important "Show Remote Objects" menu item....



- Once "Show Remote Objects" is enabled (it is a toggle) all the objects in the associated library for this project are shown in the iSeries Navigator.
- This navigator is called a "bleed through" view because it lists both local files/members and remote objects, files and members. However, if a file or member exists both locally and remotely, it is not shown twice. Rather, it is only shown once, and its icon and bracketed text identify it as existing in both places.
- The icons and text enable you to see at a glance which files/members exist only locally, only remotely, or both locally and remotely. In the latter case, the color of the icon indicates if the two are in synch or not. If not, it is an indication you need to "push" your local source to the associated library to get it in synch.

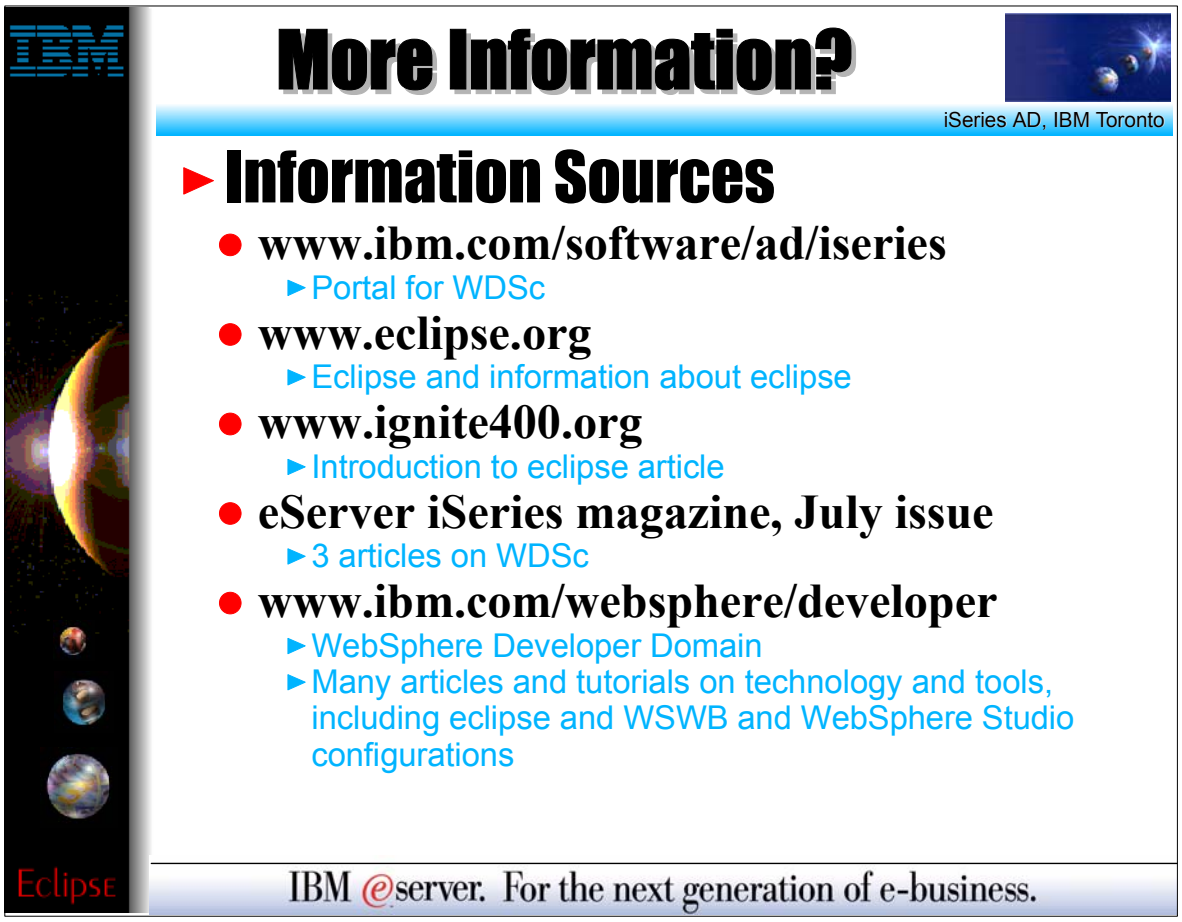


- With the local copies of members, you can edit them using the Lpex or CODE editor, or editor of your choice. When a local or remote object is selected, you see important properties of it in the Properties view.
- When editing is complete, right clicking on the project or anything in the project offers actions to push the changed source members to the associated library, and run the build command to compile those changes (if you supplied such a command)
- To work on remote objects, use the Remote Systems view that is also in the iSeries Projects perspective. For example, if you do not have a build command, then use the RSE to compile your remote source member after pushing it.

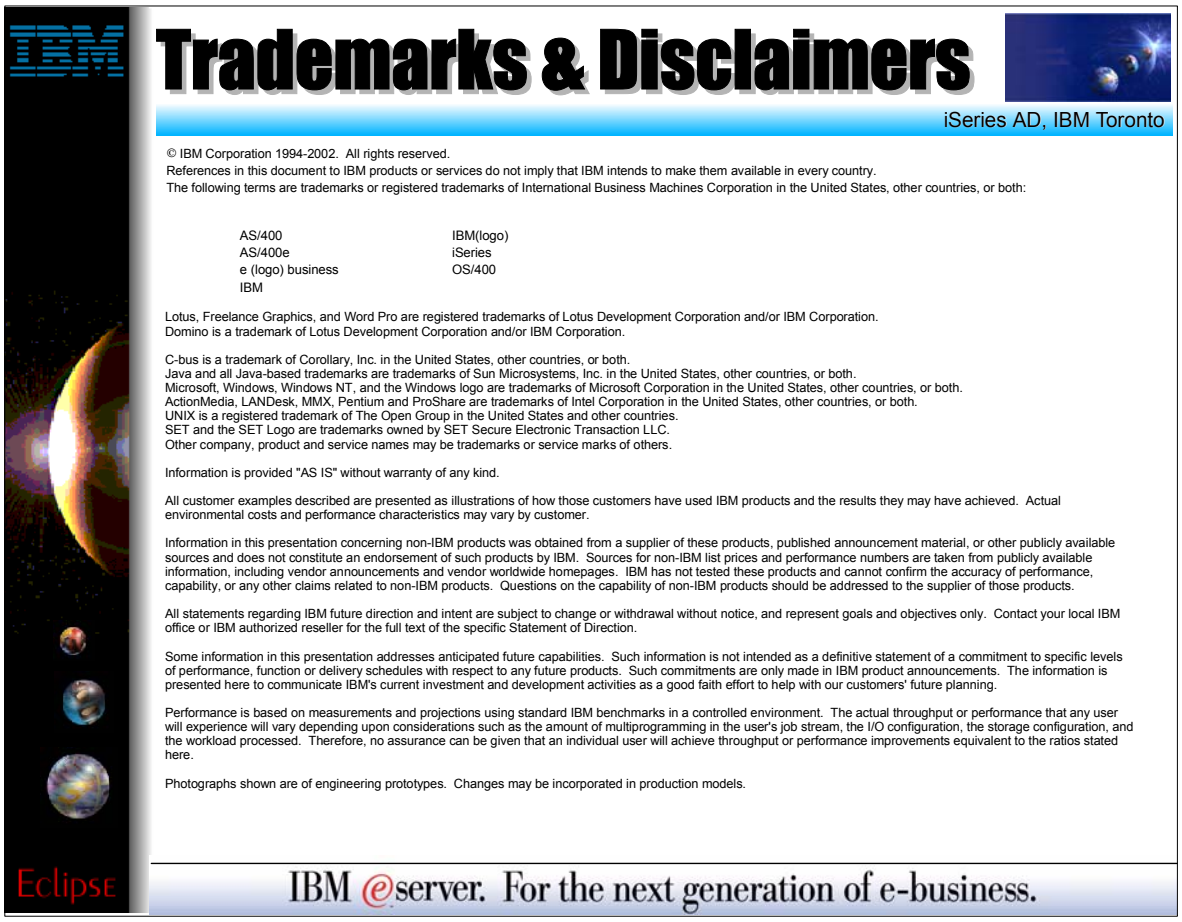


- The typical usage scenario for iSeries projects is to use them for tasks, such as adding a feature to an existing application.
- Rarely will you use an iSeries project to hold all the source for an application... use an iSeries change management vendor for that.
- Consider a change that involves 3 developers changing a number of files...
- The lead developer will create the iSeries project and synchronize it with the repository. The other developers will use the Eclipse team support to add that project to their workspace. Each will change the properties of the project to associate it with their own private library. The master library containing all the source will be on their library list, so compiles will work, but their private library will only hold the members they work on. If they use an SCM product, then they will use it to check source members out to their private library. These members and files will then be imported into their project, where they will edit them.
- After editing, they will "push" their changes to their library and do a "Build" or a compile, and run and test their unique changes.
- Every so often, they will do a team synchronization to give their changes to their fellow team members, and pick up the changes of those team members. After a synch, they will push their colleaque's files to their own associated library and test the changes all work together.
- At the end of the cycle, someone will pick one of the libraries, and use the SCM product to promote their changes back into the test and production stages.

- iSeries projects are young technology!



- Where to get more information


Trademarks & Disclaimers
iSeries AD, IBM Toronto
© IBM Corporation 1994-2002. All rights reserved.
References in this document to IBM products or services do not imply that IBM intends to make them available in every country.
The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:
AS/400
AS/400e
e (logo) business
IBM
IBM(logo)
iSeries
OS/400
Lotus, Freelance Graphics, and Word Pro are registered trademarks of Lotus Development Corporation and/or IBM Corporation.
Domino is a trademark of Lotus Development Corporation and/or IBM Corporation.
C-bus is a trademark of Corollary, Inc. in the United States, other countries, or both.
Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.
Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.
ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.
UNIX is a registered trademark of The Open Group in the United States and other countries.
SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.
Other company, product and service names may be trademarks or service marks of others.
Information is provided "AS IS" without warranty of any kind.
All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.
Information in this presentation concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.
All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.
Some information in this presentation addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements. The information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.
Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.
Photographs shown are of engineering prototypes. Changes may be incorporated in production models.
Eclipse
IBM @server. For the next generation of e-business.