**Lesson 6: Object Repository Administration & Maintenance**

**What is an Object Repository in UFT?**

Object Repository is a collection of object and properties with which UFT will be able to recognize the objects and act on it.

When a user records a test, the objects and its properties are captured by default. Without understanding objects and its properties, UFT will NOT be able to play back the scripts.

During test design phase, objects in Application Under Test (AUT) are recorded and stored by UFT. During test execution phase, a UFT script is replayed where the run-time objects are compared against the stored test objects.

The objects are represented in a tree view.

The object repository can be accessed by the following path:

1. Resources🡪Object Repository
2. Click on the object repository toolbar button
3. Keyboard keys (Ctrl+R)

Even when steps containing a test object are deleted from the test or component, the objects remain in the object repository.

**Types of Object Repository**

**Local Object Repository**: As soon as you create a new action in UFT, a corresponding object repository is created along with it. This repository is known as local object repository. Any objects that are recorded in this repository are only available for the corresponding action and are not available for other actions in a test. Local object repository is present inside the corresponding action folder and always have the name ObjectRepository.bdb

**Shared Object Repository**: A shared object repository is an object repository that exists independently of an individual test or action. As the name suggests, the test objects present inside shared object repositories can be used by multiple tests or actions. Shared object repositories can be managed using Object Repository manager. We will discuss more about OR manager in the later advanced tutorials. Shared object repository can have any user defined name with an extension tsr.

Example: FlightReservations.tsr

**NOTE**: While recording/learning objects inside an Object repository it is recommended to store only the required objects inside it. Any extra objects that are recorded inadvertently should be deleted from the object repository. This will keep the size of repository small, help in better maintenance and performance optimization.

**Following are the major difference between Local and shared ORs.**

|  |  |
| --- | --- |
| **Local Object Repository (LOR)** | **Shared Object Repository (SOR)** |
| This Object Repository is available; one for each action. | This type of OR is available for multiple tests and for multiple actions. |
| This is the OR that is available for each tests, by default. | This type of OR is usually used in frameworks considering reusability and maintainability. |
| Local Object Repository is editable in Object Repository. | Shared Object Repository is read-only by default but can be edited in Object Repository Manager. |
| It is NOT a standalone file that can edited. | SOR is a standalone file that can be edited easily. |
| It is saved with an Extension. bdb | SOR is saved with an Extension. tsr |
| It should be used when not many tests are working on the same screens of the application under test. | SOR Should be used when there are different scripts interacting with the same type of the object. |

**When to use local object repository and when to use shared object repository?**

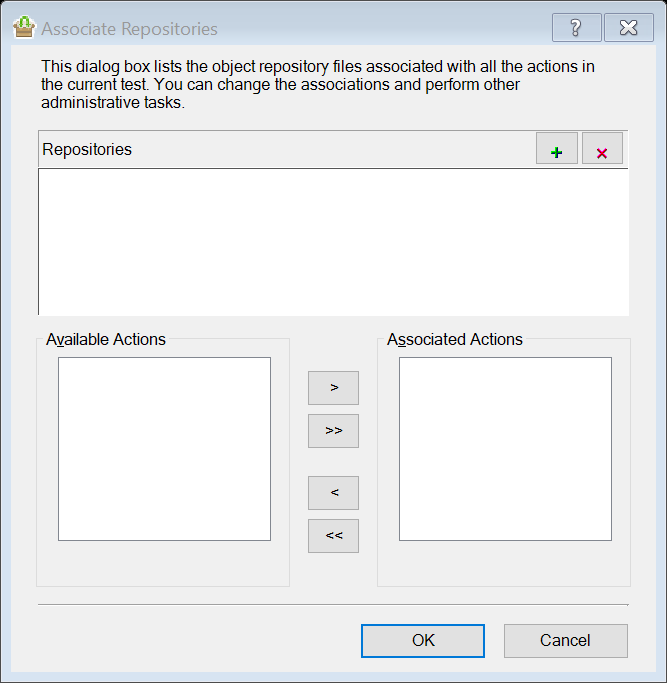
1. If you are new to UFT, use local object repository. Since any new object goes in a local object repository by default, you need not bother about creating or modifying shared object repositories.
2. If you are working on simple applications where you expect the object properties won’t change frequently, you may go for local object repository.
3. If you work on complex applications, where you expect the object properties can change from time to time you may go for shared object repository.
4. If you work on multi action tests, where you regularly use Call to Copy of Action, you may want to go for shared object repository.
5. The biggest advantage of a shared object repository lies in test objects maintenance. In case a test object needs to be updated, a single update in a shared object repository will reflect across all tests/actions using that shared object repository.

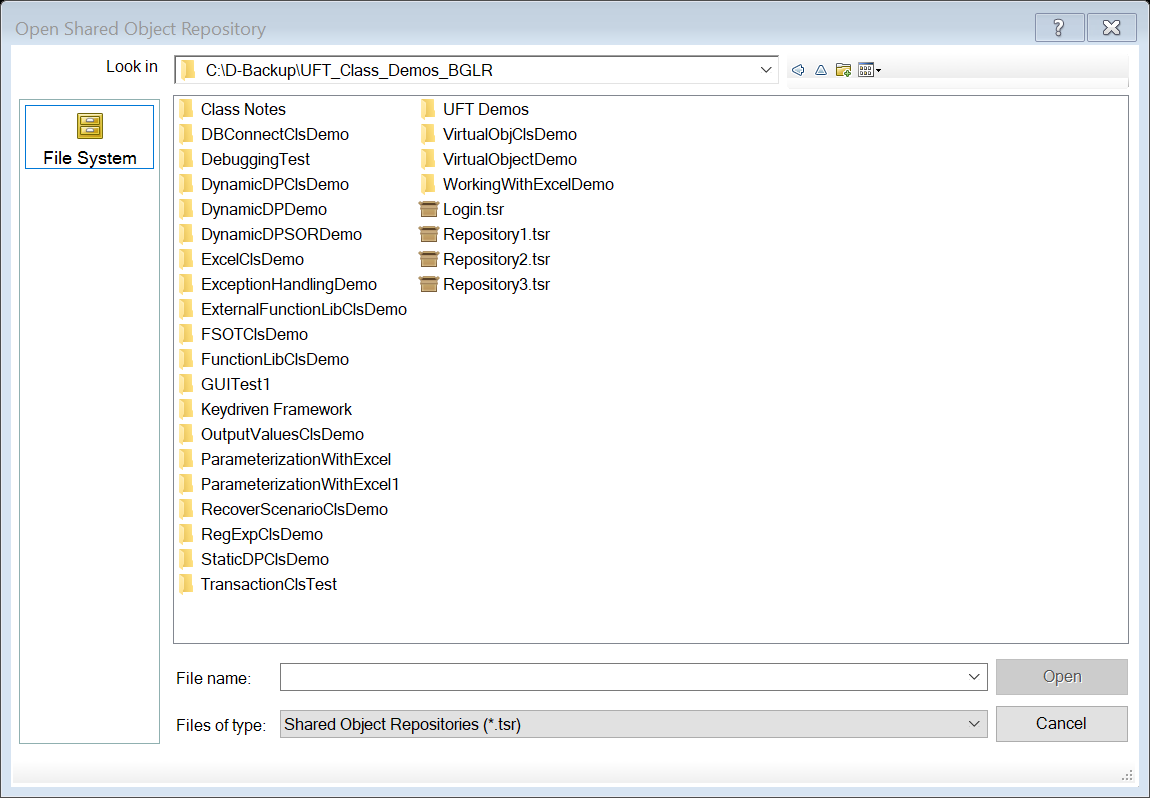
**Demo: Creating Shared Object Repository**

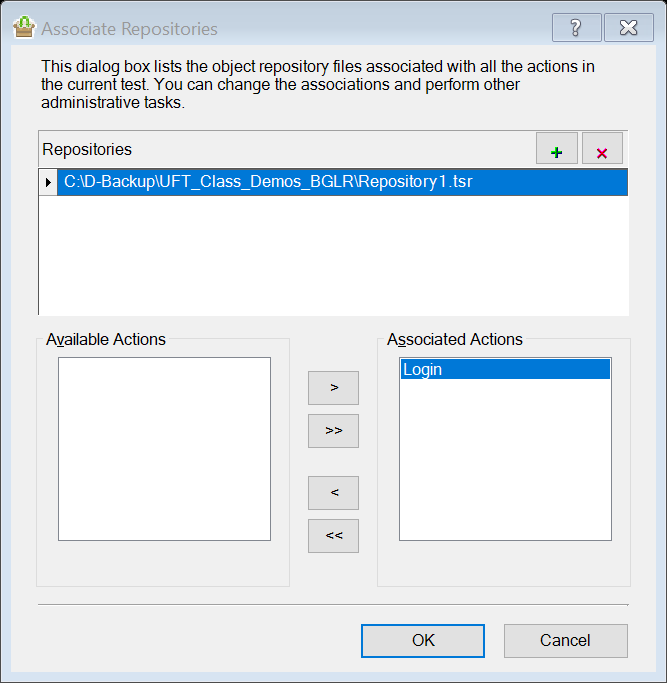
**Steps**: To create a shared object repository, click on Resources🡪Object Repository🡪File🡪Export Local Objects and save it at the desired location. It will be saved with the extension .tsr.

**Demo: How to associate Shared Repository with Action.**

**Steps**: Resources🡪Associate Repositories🡪Click on + button🡪







**Demo**: How to edit Shared Repository?

By default, shared OR(Object repository) will be opened in read only mode. For editing the shared OR, We have to go the options ‘Resources –> Object Repository Manager’ and there we can open the shared OR from its location where it was saved. It will still show in read only mode. For making it in editable mode, we must click on one icon called ‘Enable Editing’. Icon is available on the tool bar in Object repository manager.

**Features of Object Repository:**

**A screenshot of a social media post

Description automatically generated**

1. Recorded or added objects with a parent and child hierarchy.
2. Button to add new objects.
3. Update the properties of existing objects from AUT. Select the object in OR and click on this button. Then hover on that object in AUT and select. It will pick up the updated properties.
4. It will add insight objects which are image based identification.
5. Define new object – We can create an object manually by providing Environment,properties and class name.
6. Highlight the selected object in AUT.
7. It helps to locate the object in object repository and will be useful when object repository is huge. Click on it ->select the required object from AUT and click OK . it will identify the object in object repository or give error message if object does not exist in OR.
8. Object Spy- used to spy over the objects in AUT and find out their properties. It can be useful while writing descriptive programming.
9. It will associate the shared repository to the particular action.
10. Logical name of the object- it can be user defined. When changes , same will be reflected in the script as well.
11. It tells which class objects belongs.
12. It tells if object repository is local or shared.
13. It is used to add more properties into test object properties pane if existing properties are not enough for object identification.
14. Remove the selected properties from test object details.
15. It will restore the mandatory properties set required for object identification.
16. Properties of the object which will help to identify the object at run time in AUT.

**NOTE**: One can change the object's name which will not have any effect on identifying the object uniquely. If there is a change in logical name, the same name should be used while scripting. Only the Object's Name can be changed and not its properties. Properties of any object must be unique so that UFT will be able to recognize the objects and act on it. If the object properties were same for two or more objects, then during execution an error would be thrown that "More than one object is matched for the specified properties".

**Object Identification Mechanism:**

From automation perspective, whatever you see in an Application like Browser, Page, button, Textbox, Link, Radio button, check box etc. are called as Objects.

Now, an automation engineer’s task is to perform some operation on these objects like enter data into text box, click on button etc. For that, these objects should be identified through our script. The approach used for identifying these objects is called **“Object Identification”**.

An object is identified on the basis of a set of properties and values which helps to recognized that object uniquely in AUT. Every object will have a combination of properties and values.

What is **Test Object** and **Run-time Object**?

**Test objects** are the stored representations of the actual objects in your application. As we have seen in the last tutorial, UFT identifies an object using a set of properties called Test object properties. UFT creates test objects by learning test object properties and values of the objects in your application. These test objects, their properties and corresponding values are then stored in UFT in OR. Using the objects and the properties stored inside object repository, UFT identifies object during run time.

**Runtime objects** are the actual objects available in AUT during the execution time. The property-value pairs of test objects AND run-time objects are compared for object identification purposes in UFT.

**UFT Identification Mechanism:**

1. **Learned Description:** In the first stage, UFT identifies an object using a combination of mandatory and assistive properties. This combination of mandatory and assistive properties is known as Learned description also sometimes referred as description properties or test object description.
   1. **Mandatory Properties:** Mandatory properties are the properties that are always recorded when UFT learns an object.
   2. **Assistive Properties**: are the optional properties that are recorded only when UFT is unable to create a unique object description using mandatory properties alone.

An important point to note here is that, during a run session, UFT looks for objects that match all properties present in the learned description, it does not distinguish between properties that were learned as mandatory properties and those that were learned as assistive properties.

**How UFT uses Mandatory and assistive properties?**

**While Recording:**

When we try to add an object to the object repository, UFT follows the following procedure:

1. Fetch all the mandatory properties of the object and check if the unique object description can be created. If not then, continue to step 2.
2. Add one assistive property at a time to the object repository and try to create a unique object description.
3. If unique description cannot be created then, UFT will use the ordinal identifiers for creating the unique identification description.
4. **Visual Relation Identifier:** Visual Relation Identifiers allow you to identify fields in your application based on other objects that are always near them. For example, Username label text always appears to the left of the textbox for entering Username.
5. **Smart Identification:** If UFT is unable to identify any object that matches the learned object description, or if it finds more than one object that fits the description, then UFT ignores the learned description, and uses Visual Relation Identifiers (VRI) to identify an object uniquely. If VRI is not defined or VRI is not able to identify the object uniquely, UFT invokes the Smart Identification mechanism and try to identify the object. The Smart Identification mechanism uses following two types of properties**.**
   1. **Base Filter Properties**: The most fundamental properties of a particular test object class; those whose values cannot be changed without changing the essence of the original object. For example, if a Web link’s tag was changed from a to any other value, you could no longer call it the same object.
   2. **Optional Filter Properties**: Other properties that can help identify objects of a particular class. These properties are unlikely to change on a regular basis, but can be ignored if they are no longer applicable.
6. **Ordinal Identifiers:** Ordinal identifiers are used by UFT to create a unique identification description, if unique description cannot be created using mandatory or assistive properties.

**Ordinal identifier are of 3 types:**

1. **Index**: Index tells the position of the objects in the application code. Index start from zero(0). Suppose, there are 4 identical text boxes on any page which are having exact same properties and we need to enter some text into 3rd one. Then index will be useful as their index will be assigned to them like 0,1,2,3 respectively. So, in our case, 3rd text box will be identified with the index 2.

A screenshot of a cell phone

Description automatically generated

The value of index for the group of text boxes will be as follows −

|  |  |
| --- | --- |
| **Object Name** | **Index Value** |
| TextBox 1 | 0 |
| TextBox 4 | 1 |
| TextBox 2 | 2 |
| TextBox 5 | 3 |
| TextBox 3 | 4 |
| TextBox 6 | 5 |

1. **Location**: Location will also tell the position of the object but relative to the parent window.It also start with zero(0).

|  |  |
| --- | --- |
| **Object Name** | **Index Value** |
| TextBox 1 | 0 |
| TextBox 2 | 1 |
| TextBox 3 | 2 |
| TextBox 4 | 3 |
| TextBox 5 | 4 |
| TextBox 6 | 5 |

1. **CreationTime**: This is used only for browser identification. Suppose, we have 4 browser opened with the same application. Consider Gmail application. Now, we need to enter the gmail username in 2nd browser. In this case, we have to use ‘CreationTime’ as 1 to work with the second browser.

**Merge Repositories:**

**Steps:**

1. Go to ‘Resources –> Object Repository Manager’
2. In object repository manager, Go to Tools –> Object Repository Merger tool’
3. We will get 2 browse options to open 2 shared repositories for merging
4. When we click ‘OK’, it shows a window containing the details about ‘conflicting object’ and ‘Merge summary’
5. We can take action or modify object as per our need, if required close the window
6. When we click on save button, it will be saved as a third repository with all the merged objects

**Compare Object Repositories:**

**Steps:**

1. Go to ‘Resources –> Object Repository Manager’
2. In object repository manager, Go to Tools –> Object Repository comparison tool’
3. Same way as above, select the first file (OR1) and second file (OR2) and click OK
4. It will open a window with the details of the unique objects (Highlight in blue color) and conflicting objects (Highlight in Red color)
5. We can see first OR in left pane and second OR in right pane.

**Object Repository Parameter**: Object repository parameter are used to parametrized the object whose properties are changing dynamically in the Application. Suppose, there is button ‘Submit’ in the localized application and button’s text property gets changed as the language of the application changes.Now, in this case we can parametrized the text property of the button so user can provide the property value according to the language.