Phases of Adapter Processing

This chapter covers the following topics:

- SET/INVOKE/GET Phase - The Default Phases
- INIT Phase when Adapter is Constructed
- DESTROY Phase when Adapter is Deregistered

SET/INVOKE/GET Phase - The Default Phases

An adapter object is the logical representation of a page. The page runs inside the GUI client, the adapter runs inside the Java server.

The user changes information on the page (e.g. inputs some values into field controls) and operates some functions (e.g. chooses a button). Every time a function is invoked, a request is initiated from the client. The request contains all data that was changed on the client, and it contains the command (e.g. the method to be called; sometimes there is no explicit command but the request just is a "data transfer request", e.g. when having defined `FLUSH="server"` with a FIELD control).

The request is processed in three phases:

- (activate)
- SET phase
- INVOKE phase
- GET phase
- (passivate)

During the SET phase, Application Designer passes all changed property values into the property representations of the server side adapter object. In the following INVOKE phase, the method is called that is associated with the request. In the GET phase, Application Designer checks all property values if they have changed. If a change happened (i.e. during the INVOKE phase, some property values were changed), then the changes are communicated back as response to the client.

The SET and GET phases have dedicated methods which are called inside the adapter in order to signalize the start and end of the phases:

- **SET phase:**

```
reactOnDataTransferStart();
set...();
set...();
set...();
reactOnDataTransferEnd();
```

- **GET phase:**

```
reactOnDataCollectionStart();
set...();
set...();
set...();
reactOnDataCollectionEnd();
```

You can use the methods for diverse purposes:

- **reactOnDataTransferStart()**
  You may want to initialize certain internal data that needs to be initialized for each request processing.

- **reactOnDataTransferEnd()**
  You may want to check which properties actually have changed and which application checks have to be invoked.

- **reactOnDataCollectionStart()**
  You may want to build up some interim objects for complex data structures that allow a faster response for the following get calls.

- **reactOnDataCollectionEnd()**
  You may want to set certain data to initial values - in case they were changed during request processing.

INIT Phase when Adapter is Constructed

The INIT phase is processed only once per adapter instance - at the point of time when it is constructed.

The INIT phase is internally processed in two steps:

- Creation of the adapter object via the `new` operator (without any parameters).
- `init()`

Application Designer first creates an instance of an adapter object and then calls the `init()` method of the object.

Important:
Many functions inside your adapter class (extended from Application Designer's `Adapter` class) are only available after having constructed the object. Application Designer first creates the instance of the object and then internally registers the object inside its internal data structures (session management, etc.). The `init()` method is called after the internal registration. All functions that require the adapter object to be correctly registered will fail when being called inside the constructor, but will not fail if called in the `init()` method.

Best practice: use the `init()` method as constructor for an adapter object.

DESTROY Phase when Adapter is Deregistered

The DESTROY phase is processed only once per adapter instance - when Application Designer has internally deregistered the adapter instance:

The DESTROY phase consists of one method that is called inside the adapter:

- `destroy()`

Application Designer’s session management deregisters all adapters when a subsession or session is closed. All associated adapters are internally deregistered so that they are available for garbage collection. In addition, each instance receives a destroy signal so that it can internally pass back resources that it used.