Quick answers to common problems

SAP ABAP Advanced Cookbook

Over 80 advanced recipes with excellent programming techniques that focus on the Netweaver 7.0 EHP2 and above

Rehan Zaidi

[PACKT] enterprise
PUBLISHING

SAP ABAP Advanced Cookbook

Over 80 advanced recipes with excellent programming techniques that focus on the Netweaver 7.0 EHP2 and above

Rehan Zaidi

[PACKT] PUBLISHING enterprise professional expertise distilled

BIRMINGHAM - MUMBAI

SAP ABAP Advanced Cookbook

First published: December 2012

Production Reference: 1191212

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-488-0

www.packtpub.com

Cover Image by Artie Ng (artherng@yahoo.com.au)

Credits

Author
Rehan Zaidi

Reviewers
Steffen Macke
Alvaro Tejada Galindo
Alexey Tveritinov
Eric Wildenstein

Acquisition Editor
Rukhsana Khambatta

Lead Technical Editor
Susmita Panda

Technical Editors
Kaustubh S. Mayekar
Kirti Pujari

Copy Editor
Laxmi Subramanian

Project Coordinator
Arshad Sopariwala

Proofreaders
Clyde Jenkins
Lydia May Morris
Kevin McGowen
Stephen Swaney

Indexer
Rekha Nair

Graphics
Aditi Gajjar

Production Coordinator
Shantanu Zagade

Cover Work
Shantanu Zagade

About the Author

Rehan Zaidi has more than 13 years of SAP experience and has been writing about SAP topics since 2001. He co-authored an ABAP programming training manual for a course taught in North America and has written a number of SAP books and articles about ABAP, workflow, HR functional and technical users, and SAP user experiences. Rehan has carried out support and implementation projects involving various areas of ABAP and workflow, and has worked in technical and functional areas of SAP ERP HCM. He holds bachelor and master's degrees in computer science. You may reach Rehan via e-mail at `erpdomain@gmail.com`.

I am very thankful to my parents, especially my mother, whose prayers are with me all the time. I am grateful to the many friends and well-wishers who have supported and encouraged me both through the duration of this project and throughout my life as a whole.

In the preparation of the book, I would like to thank Rukhsana Khambatta for turning a book idea (that began in my mind) into reality. In addition, I am indebted to the entire team at Packt Publishing, including Susmita Panda, Sai Gamare, Arshad, and others. Last but not least, my thanks to those who reviewed this book and provided me with feedback, especially Steffen Macke for his invaluable suggestions.

I apologize to anyone whom I have failed to mention. There are many people who have helped me in this process and who have encouraged the creation of this book. To all of you, I extend my most heartfelt thanks.

About the Reviewers

Steffen Macke is a Civil Engineer and Software Developer. After several years of work on water supply projects in the Middle East, he's now back in Germany and has joined the software industry.

Maps and Geographic Information Systems (GIS) played a key role in his hydraulic analysis and customer database activities. They served him as an entry point to the world of programming, relational databases, version management systems, and web technology. The complexity of the projects he encountered made him embrace diversity, active communities, and practical approaches. That's why he doesn't have a favorite programming language, operating system, or database management system.

Steffen is actively involved in a number of open source projects, among which the general purpose drawing software Dia is the most popular (`http://dia-installer.de`). His passion for open source does not mean that he's ignorant to the advantages of commercial software development models, he believes that they're great to make a living. If you're interested in Steffen's views and projects, make sure that you visit his website `http://sdteffen.de`.

Alvaro Tejada Galindo worked as a Senior ABAP Consultant for 11 years, then he moved to SAP Labs in Montreal where he works as a Development Expert. Besides his SAP background, Alvaro is very proficient in scripting languages like PHP, Python, Ruby, and R and considers himself to be a regular expressions hero.

Alvaro has worked in Peru and Canada for some of the best consultant companies, namely Stefanini IT Solutions, ActualiSap, and Beyond Technologies. Presently, he is working for SAP.

Alvaro has published several programming books on `http://www.lulu.com/spotlight/blag`.

I would like to thank my wife Milly and my daughter Kiara for all their support while I was doing this book's review.

Alexey Tveritinov graduated from Moscow State University of Informatics and Craftsmanship in 2008. After that he was hired by NVIDIA in a GPU and driver testing team as Junior Software Engineer, where he undertook development of various tools for tests automation and performance measurement. After spending one year at NVIDIA he left the company as his work on the software had finished, and he wasn't involved in other developments.

After that he was hired by a medical company named Trackpore Technology where he developed embedded software for plasmapheresis medical units using Linux and C++.

In 2011, he was hired by SAP CIS as Developer Associate and started to work on implementing the framework for XML reports according to specifications of legal units of Russia, Ukraine, and other CIS countries, without the limitations of DMEE.

I would like to thank Vasily Kovalsky, a teacher at the SAP training center, for his patience and knowledge. In addition, I would like to thank my managers Vadim and Juri for the trust in me and my skills. Also I would like to thank all developers in the GS unit of SAP, who were open to share their knowledge and experience. Also, I would like to thank my girlfriend Olga Tupikina for her patience and understanding while I was working on several projects and had little time to share with her.

Eric Wildenstein is a SAP independent Consultant, who has been working on ERP implementations for blue chip companies in Western Europe and North Africa regions since 1997. He mainly specializes in ABAP Object programming, NetWeaver XI/PI and SAP Business Workflow, providing technical expertise across the core business modules of SAP. Prior to being self-employed in 2000, he worked as an in-house Programmer Analyst on behalf of PricewaterhouseCoopers, U.K. and Andersen Consulting, France, on both SAP R/3 and C/S architectures.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit `www.PacktPub.com` for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at `www.PacktPub.com` and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at `service@packtpub.com` for more details.

At `www.PacktPub.com`, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.

`http://PacktLib.PacktPub.com`

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at `www.PacktPub.com`, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following `@PacktEnterprise` on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface

Advanced Business Application Programming (ABAP) is SAP's proprietary 4th Generation Language (4GL). SAP core is written almost entirely in ABAP. ABAP is a high level programming language used in SAP for development and other customization processes. This book covers advanced SAP programming applications with ABAP. It teaches you to enhance SAP applications by developing custom reports and interfaces with ABAP programming. This cookbook has quick and advanced real world recipes for programming ABAP.

It begins with the applications of ABAP objects and ALV tips and tricks. It then covers design patterns and dynamic programming in detail. You will also learn the usage of quality improvement tools such as transaction SAT, SQL Trace, and the code inspector. Simple transformations and its application in Excel downloading will also be discussed, as well as the newest topics surrounding Adobe Interactive Forms and the consumption and creation of Web services. The book comes to an end by covering advanced usage of Web Dynpro for ABAP and the latest advancement in Floorplan Manager.

What this book covers

Chapter 1, *ABAP Objects*, introduces useful recipes related to the object-oriented programming. This will include useful design patterns, the shared memory, and the persistent object concept.

Chapter 2, *Dynamic Programming*, covers facets of dynamic programming as applied in ABAP, such as Dynamic Open SQL and usage of field symbols and references.

Chapter 3, *ALV Tricks*, shows how you can get the most out of ALV programs. Starting with a simple ALV program, we will add code in recipes to fulfill a variety of user requirements.

Chapter 4, *Regular Expressions*, guides you on how you can embed regex programming in your ABAP programs and solve complicated problems in the least possible time and with minimal code.

Chapter 5, *Optimizing Programs*, shows the newer feature of secondary indexes and the transaction SAT (runtime analyzer) along with valuable program optimization tips.

Chapter 6, *Doing More with Selection Screens*, discusses recipes based on less frequently applied functionality within ABAP programs' selection screens, such as the addition of tabstrips and placement of buttons on toolbar. In addition, we will see how to take folder and file names as input, followed by a recipe for writing code in search help exits.

Chapter 7, *Smart Forms – Tips and Tricks*, introduces various recipes based on Smart forms and fulfilling user's form printing requirements in the least possible time.

Chapter 8, *Working with SQL Trace*, provides lesser-known tricks related to the SQL Trace tool. This will include the performance optimization usage of the SQL trace tool as well as the use of finding data source of screen fields.

Chapter 9, *Code Inspector*, shows how to check the quality of custom programs using standard checks, along with the procedure for creating your own checks.

Chapter 10, *Simple Transformations*, discusses in detail the Simple Transformation language and the representation of data variables in it, the application for Excel download format will also be shown.

Chapter 11, *Sending E-mail Using BCS Classes*, covers the classes of the Business Communication Service (BCS) for e-mail generation. This chapter will cover everything from simple e-mails for SAP users to Internet e-mail addresses, and also the procedure for adding attachments of various formats.

Chapter 12, *Creating and Consuming Web Services*, covers the step-by-step procedure for the creation of Web services based on an ABAP function module using the Inside-Out approach. The steps required to create a consumer of the Web service will also be shown.

Chapter 13, *SAP Interactive Forms by Adobe*, shows how to create both print and interactive forms using the SAP Interactive forms technology. A number of scenarios such as Offline form processing will also be covered.

Chapter 14, *Web Dynpro for ABAP*, shows how to create simple and advanced Web Dynpro for ABAP (WD4A) applications. The advanced topics related to the Web Dynpro components will also be covered.

Chapter 15, *Floorplan Manager*, covers newer features of the Floorplan Manager design used for creating Web Dynpro applications quickly. Both the configuration and coding for useful Floorplans will also be covered.

What you need for this book

ECC 6 system with Netweaver 7.02 or higher. A trial version of ABAP Netweaver 7.02 or higher will also suffice.

Who this book is for

SAP Developers and Consultants who have at least a basic knowledge of ABAP.

Conventions

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "clicking the **Next** button moves you to the next screen".

Warnings or important notes appear in a box like this.

Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book—what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to `feedback@packtpub.com`, and mention the book title in the subject of your message.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on `www.packtpub.com/authors`.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books—maybe a mistake in the text or the code—we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting `http://www.packtpub.com/support`, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded to our website, or added to any list of existing errata, under the Errata section of that title.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at `copyright@packtpub.com` with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at `questions@packtpub.com` if you are having a problem with any aspect of the book, and we will do our best to address it.

1
ABAP Objects

In this chapter, we start with recipes for ABAP objects. This chapter is designed to provide useful recipes related to the storage of ABAP objects in shared memory and the database (persistent objects), as well as some useful design patterns. In this chapter, we will look at ways of:

- Creating a shared memory object
- Creating a persistent object
- Creating classes based on factory methods
- Creating classes based on singleton design pattern
- Creating classes based on adapter pattern

Introduction

This chapter explores recipes related to ABAP objects. Two useful features of the object-oriented ABAP are storage options in the shared memory as shared objects, and in the database as objects of persistent classes. The details about both the prerequisites as well as the necessary steps needed to created shared memory-enabled objects and persistent objects will be discussed later in this chapter.

Moreover, design patterns are very important in object-oriented programming. In this chapter, we will see how to implement three of them using ABAP objects, namely the adapter, `singleton`, and the factory design. We will create a class with a `factory` method design. Later, we will show how this class may be modified in order to behave like a singleton class. Finally, we will see how an object of one class may be converted to that of another using an adapter class. The examples are kept simple in order to emphasize on the design pattern concept.

For this chapter, we assume that the reader has basic knowledge of the ABAP objects, and is familiar with the class-builder transaction.

Creating a shared memory object

This recipe shows how to store the instances of your classes in the shared memory of the application server. A number of programs may access these objects that reside on the application server shared memory.

Two classes are necessary for shared memory, namely the `area` class and the `area root` class. The `root` class is necessary for storing (encapsulating) the data that are to be stored in the shared memory. An `area` class may comprise of various instances that may consist of a number of versions.

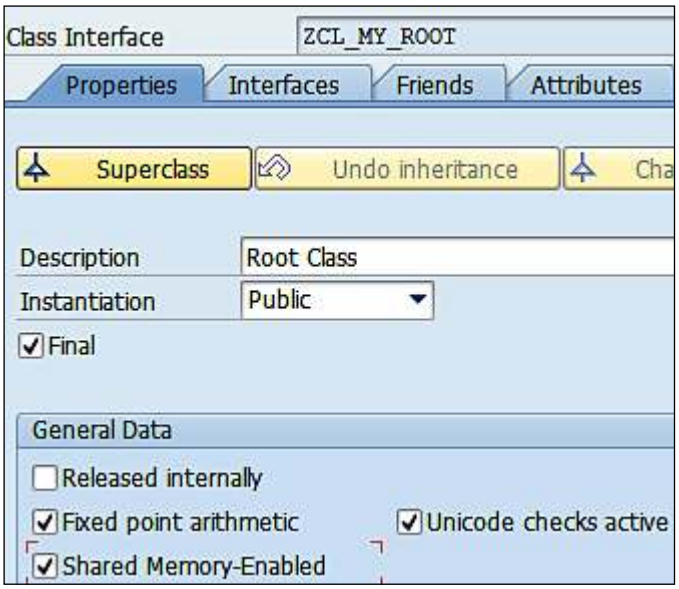
An important concept shown in this recipe is the `CREATE OBJECT` statement with the addition `AREA HANDLE`. This will create the object in the application server that is shared memory pointed to by the area handle `myarea`.

Getting ready

Prior to writing the code for storing objects in shared memory, an `area root` class must be created and a shared memory area be defined using transaction `SHMA`.

The steps required for creating a root class are:

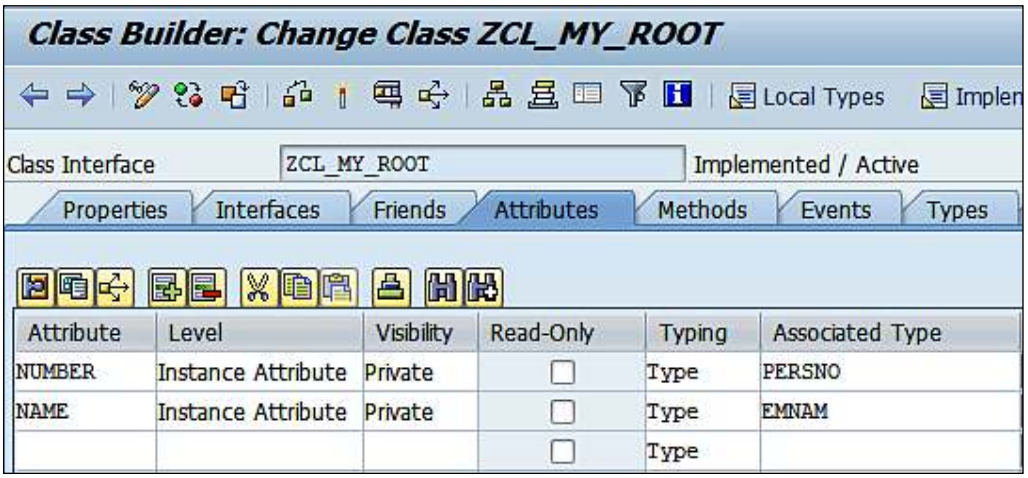
1. Call transaction `SE24`; enter a suitable name to your `root` class, as shown in the following screenshot. On the **Properties** tab, we need to make sure that the `Shared-Memory` checkbox is switched on.



Downloading the example code

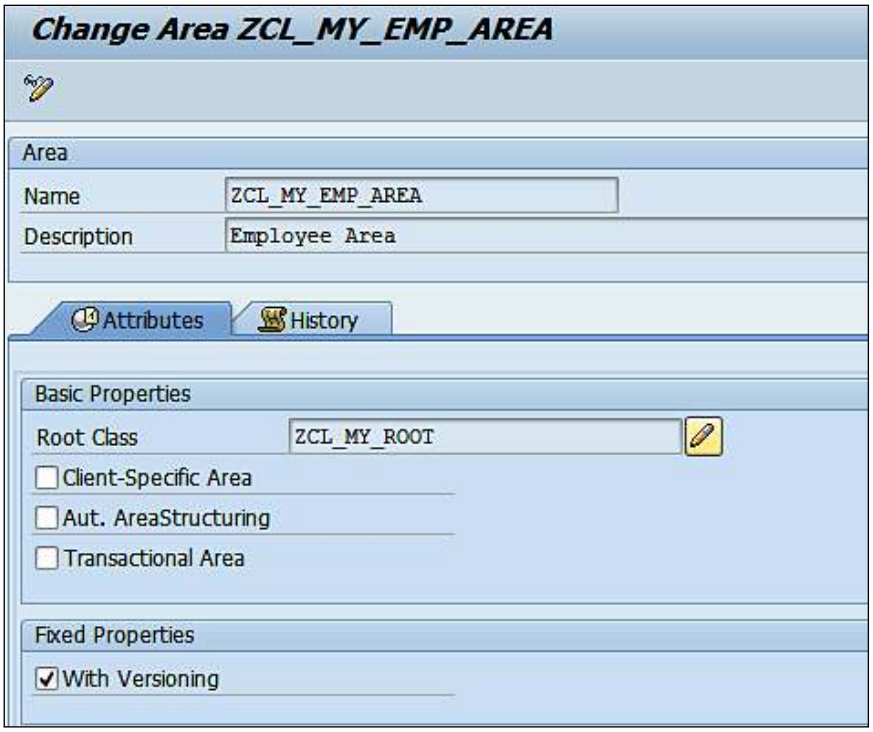
You can download the example code files for all Packt books you have purchased from your account at `http://www.packtpub.com`. If you purchased this book elsewhere, you can visit `http://www.packtpub.com/support` and register to have the files e-mailed directly to you

2. We have named it `ZCL_MY_ROOT`. We will then define two **Instance Attributes**, **NUMBER** and **NAME**, having private visibility, as shown in the following screenshot:

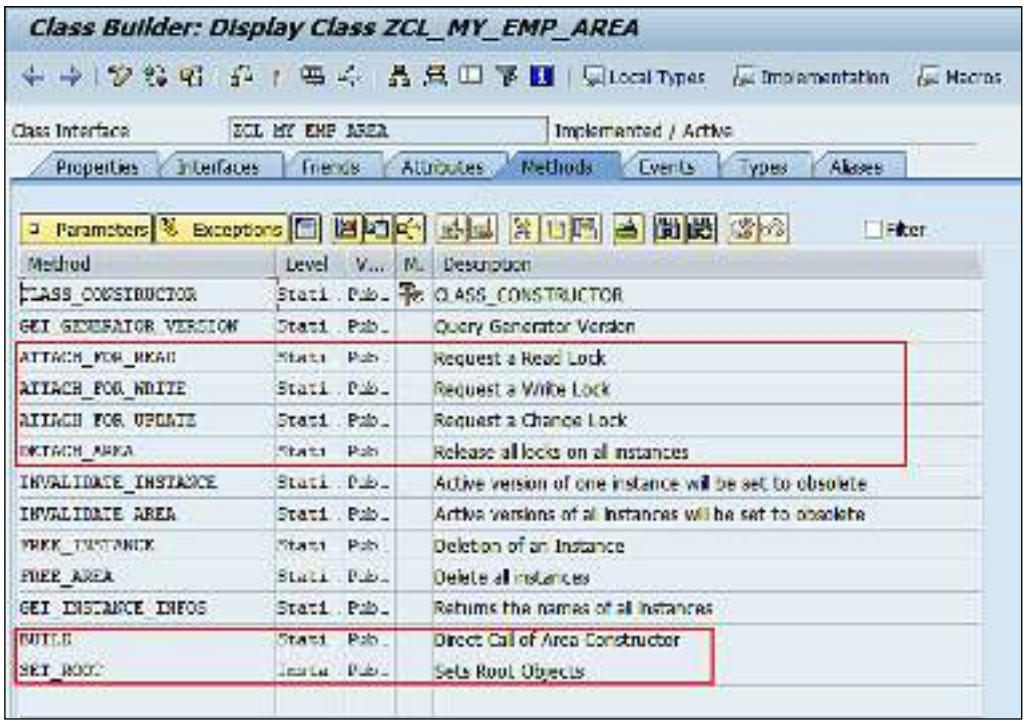


3. Two suitable methods, **SET_DATA** and **GET_DATA**, are also added to the class. The **SET_DATA** method contains code that imports number and name and assigns to the attributes **NUMBER** and **NAME** of the class. The **GET_DATA** method does just the opposite, that is, it exports the **NUMBER** and **NAME** attribute for a given shared memory object.
4. Next, the shared memory area should be created. This is done via transaction `SHMA`.

5. Enter a suitable name and click on the **Create** button. We have typed the name `ZCL_MY_EMP_AREA`. On the screen that appears, enter the description of the area. Also, enter the name of the `root` class created earlier in the **Root Class** field. You may leave the **Client-Specific Area** checkbox unchecked as it is not required for our recipe. Now, save your entries. Refer to the following screenshot:



6. This will also generate an `area` class by entering the same name `ZCL_MY_EMP_AREA`.



7. This area class will contain the necessary methods used for reading, changing, and creating the `area`, such as **ATTACH_FOR_UPDATE**, **ATTACH_FOR_READ**, and **ATTACH_FOR_WRITE**.

How to do it...

For creating the set of code that writes object's contents to the shared memory, follow these steps:

1. Two object references `my_handle` and `my_root` are defined, one for `area` class and the other for `root` class.
2. The static method `attach_for_write` of the `area` class `zcl_my_emp_area` is called.
3. The `CREATE OBJECT` with the area handle, `my_handle` must then be called.
4. The root and the created area instance must be linked using the `set_root` method of the handle.
5. The `set_data` method is called with the relevant number and name.

6. The `detach_commit` method of the `area` class is then called.

```
data : my_handle  type ref to zcl_my_emp_area  .
data : my_root type ref to zcl_my_root.

try .
    CALL METHOD zcl_my_emp_area=>attach_for_write
      EXPORTING
        inst_name = 'INST_NAME'
      RECEIVING
        handle    = my_handle.

    CREATE OBJECT my_root area handle my_handle.

    CALL METHOD my_handle->set_root
      EXPORTING
        root = my_root.

    CALL METHOD my_root->set_data
      EXPORTING
        number = '00000024'
        name   = 'John Reed'.

    CALL METHOD my_handle->detach_commit.

  catch cx_shm_attach_error.
    write :/ 'Error in Writing to Area' .
endtry.
```

How it works...

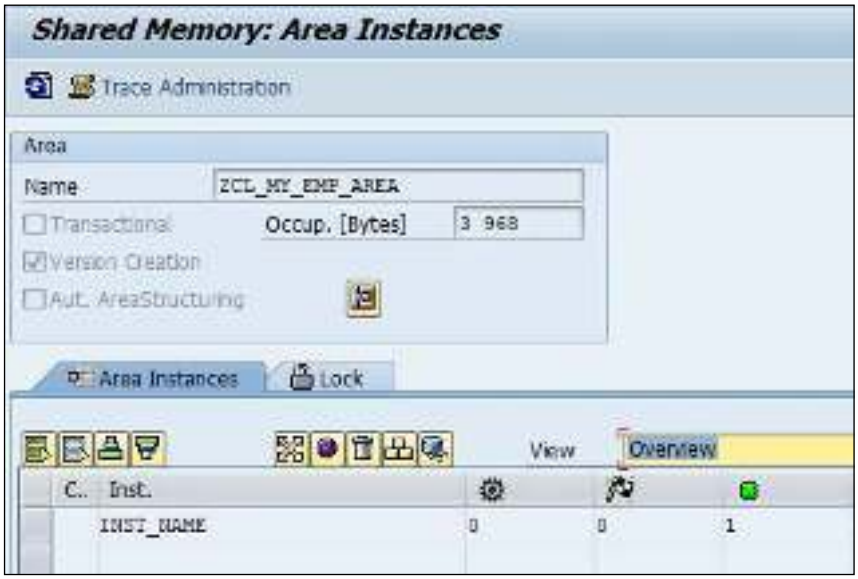
In the shared memory-writing program, the statements collectively make the writing of object in the shared memory. Let us see how the program code works.

An area instance version needs to be created before any data may be written in the shared memory on the application server. The `attach_for_write` static method is used for this purpose and returns a handle to the area instance created in the application server memory. This imposes `write` lock on the version.

The `CREATE OBJECT` statement is then called with the name of the created handle. This creates a `root` object in the `area` instance of the shared memory. The link between the `area` instance and the `root` class is created using the `set_root` method. The `set_data` method is then called for the root reference `my_root` and supplied with the name and number of the employee, which are then stored in the shared area. Finally, the `detach_commit` method is called and the `write` lock is released.

Once the program has run successfully, you may see the created object in the shared memory using the shared memory transaction `SHMM`. This will appear as your **area** class name **ZCL_MY_EMP_AREA**. Refer to the following screenshot:

Double-click on the name of area to view the details, as shown in the following screenshot:



There's more...

The read program is somewhat similar. However, instead of the `attach_for_write` method used earlier, we will use `attach_for_read`. The same instance name is passed and the handle is received. The method imposes a `read` lock on the area instance. Then, the `get_data` method of the `root` object is called using the area handle, `my_handle`. This returns the employee name and number stored earlier into the variables name and number respectively.