

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
16 September 2004 (16.09.2004)

PCT

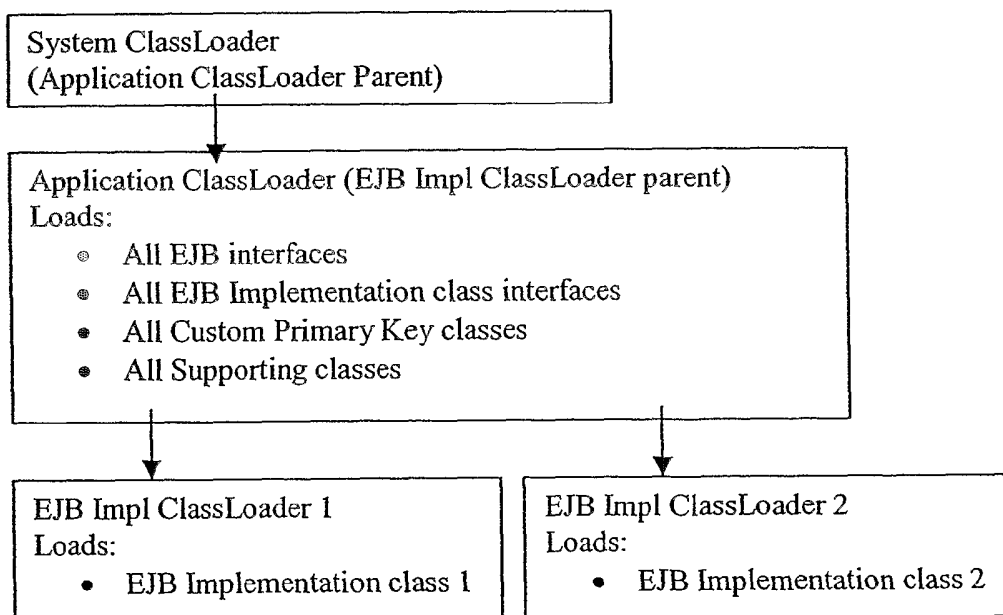
(10) International Publication Number
WO 2004/079527 A2

- (51) International Patent Classification⁷: **G06F**
- (21) International Application Number:
PCT/US2004/006214
- (22) International Filing Date: 27 February 2004 (27.02.2004)
- (25) Filing Language: English
- (26) Publication Language: English
- (30) Priority Data:
60/450,779 28 February 2003 (28.02.2003) US
10/787,706 26 February 2004 (26.02.2004) US
- (71) Applicant (for all designated States except US): **BEA SYSTEMS INC.** [US/US]; 2315 North First Street, San Jose, California 95131 (US).
- (72) Inventors; and
- (75) Inventors/Applicants (for US only): **SHINN, Matthew** [US/US]; 714 Hayes Street, San Francisco, California 94102 (US). **WHITE, Seth** [US/US]; 176 Escolta Way, San Francisco, California 94116 (US).
- (74) Agents: **MEYER, Sheldon, R.** et al.; FLIESLER MEYER LLP, Four Embarcadero Center, Fourth Floor, San Francisco, California 94111 (US).
- (81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.
- (84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: SYSTEM AND METHOD FOR EJB CLASSLOADING

New classloading scheme:



(57) Abstract: A system and method for EJB classloading. By changing the EJB classloading scheme so that the EJB impl class is loaded in a child ClassLoader, the system allows the old EJB impl class to be replaced with a new one without disturbing the rest of the deployment. The key to doing this is to remove all direct references to the EJB impl class from classes loaded in the parent ClassLoaders. Removing this dependency allows the old EJB impl class to be replaced at any time with a new impl class.

WO 2004/079527 A2

**Declarations under Rule 4.17:**

- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IT,

LU, MC, NL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)

- as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations

Published:

- without international search report and to be republished upon receipt of that report

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SYSTEM AND METHOD FOR EJB CLASSLOADING**COPYRIGHT NOTICE**

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

Claim of Priority:

[0001] This application claims priority to U.S. Provisional Patent Application 60/450,779, filed February 28, 2003, entitled "SYSTEM AND METHOD FOR EJB CLASSLOADING" (Atty. Docket No. BEAS-01451US0); and U.S. Patent Application No. _____, filed February 26, 2004, entitled "SYSTEM AND METHOD FOR EJB CLASSLOADING", (Atty. Docket No. BEAS-01451US1) both of which are incorporated herein by reference.

Field of the Invention:

[0002] The invention relates generally to application servers and software development systems, and particularly to a system and method for EJB classloading.

Background:

[0003] The architecture of the traditional EJB classloading scheme is such that any changes to the EJB implementation (impl) class forces the developer to go through a full redeployment (undeploy and redeploy cycle) for the changes to take effect. This results in unnecessary work being done, since much of the EJB deployment code that had to execute has no direct

-2-

dependency on the EJB impl class. One of the downsides is that it takes longer than necessary to have the new EJB impl changes take effect.

[0004] Another problem is that a full redeployment can be very disruptive, since it invalidates all references to the EJB and forces clients to obtain those references again to continue to invoke on the EJB.

Summary:

[0005] The invention provides a system and method for EJB classloading. By changing the EJB classloading scheme so that the EJB impl class is loaded in a child ClassLoader, the system makes it possible to replace the old EJB impl class with a new one without disturbing the rest of the deployment. The key to doing this is to remove all direct references to the EJB impl class from classes loaded in the parent ClassLoaders. Removing this dependency allows the old EJB impl class to be replaced at any time with a new impl class.

Brief Description of the Figures:

[0006] Figure 1 shows a classloading scheme used in a typical application server.

[0007] Figure 2 shows a classloading scheme as used in accordance with an embodiment of the invention.

Detailed Description:

[0008] The invention provides a system and method for EJB classloading. The EJB classloading scheme is changed so that the EJB impl class* is loaded in a child ClassLoader. This allows us to replace the old EJB impl class with a new one without disturbing the rest of the deployment. (* by referring to EJB implementation class, this means the implementation class and

-3-

all super classes).

[0009] The architecture of the traditional EJB classloading scheme is such that any changes to the EJB implementation (impl) class forces the developer to go through a full redeployment (undeploy and redeploy cycle) for the changes to take effect. This results in unnecessary work being done, since much of the EJB deployment code that had to execute has no direct dependency on the EJB impl class. One of the downsides is that it takes longer than necessary to have the new EJB impl changes take effect.

[0010] By changing the EJB classloading scheme so that the EJB impl class is loaded in a child ClassLoader, it is possible to replace the old EJB impl class with a new one without disturbing the rest of the deployment. The key to doing this is to remove all direct references to the EJB impl class from classes loaded in the parent ClassLoaders. Removing this dependency allows the old EJB impl class to be replaced at any time with a new impl class.

[0011] Implementing this, however, presents some unique challenges. First, one must ensure the EJB impl class is actually loaded in the child and not from the parent ClassLoader. This is achieved via a custom ClassLoader implementation in which the EJB impl class is specifically disallowed from being loaded in the parent ClassLoader. This is necessary because the parent ClassLoader has access to the EJB impl class along with the rest of the EJB classes since they are all packaged together.

[0012] Another challenge is ensuring none of the classes loaded in the parent ClassLoader have direct references to the classes loaded in the child ClassLoader. This is achieved by generating a custom interface for the EJB impl class, declaring all methods that are invoked on the EJB impl by the classes loaded in the parent ClassLoader. The classes in the parent ClassLoader then refer to the bean class interface instead of the bean class, and the direct dependency is removed.

-4-

[0013] **Figure 1** shows a classloading scheme used in a typical application server.

[0014] **Figure 2** shows a classloading scheme as used in accordance with an embodiment of the invention. Note that this example assumes the application only contains two EJBs - the techniques described herein can also be extended to support multiple-EJB applications. As shown in **Figure 2**, each EJB impl class is loaded in a child ClassLoader. This makes it possible to replace the old EJB impl class with a new one without disturbing the rest of the deployment. All direct references to the EJB impl class from classes loaded in the parent ClassLoaders are removed.

[0015] The present invention may be conveniently implemented using a conventional general purpose or a specialized digital computer or microprocessor programmed according to the teachings of the present disclosure. Appropriate software coding can readily be prepared by skilled programmers based on the teachings of the present disclosure, as will be apparent to those skilled in the software art.

[0016] In some embodiments, the present invention includes a computer program product which is a storage medium (media) having instructions stored thereon/in which can be used to program a computer to perform any of the processes of the present invention. The storage medium can include, but is not limited to, any type of disk including floppy disks, optical discs, DVD, CD-ROMs, microdrive, and magneto-optical disks, ROMs, RAMs, EPROMs, EEPROMs, DRAMs, VRAMs, flash memory devices, magnetic or optical cards, nanosystems (including molecular memory ICs), or any type of media or device suitable for storing instructions and/or data.

[0017] The foregoing description of the present invention has been provided for the purposes of illustration and description. It is not intended to be

-5-

exhaustive or to limit the invention to the precise forms disclosed. Many modifications and variations will be apparent to the practitioner skilled in the art. Particularly, it will be evident that while the examples described herein illustrate how the invention may be used in a WebLogic environment, other application servers, servers, and computing environments, may use and benefit from the invention. The code examples given are presented for purposes of illustration. It will be evident that the techniques described herein may be applied using other code languages, and with different code.

[0018] The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, thereby enabling others skilled in the art to understand the invention for various embodiments and with various modifications that are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the following claims and their equivalence.

-6-

Claims:

What is claimed is:

1. A system for EJB classloading comprising:
a computer including an EJB stored thereon, said EJB having an EJB implementation class associated therewith; and,
wherein the EJB implementation class is loaded in a child classloader such that all direct references to the EJB implementation class from classes loaded in the parent classloaders are removed.
2. The system of claim 1 wherein the custom classloader implementation used for the EJB implementation class is specifically disallowed from being loaded in the parent classloader.
3. The system of claim 1 wherein a custom interface is generated for the EJB implementation class, wherein said custom interface declares all methods that are invoked on the EJB implementation class by the classes loaded in the parent classloader.
4. The system of claim 3 wherein the classes in the parent classloader refer to the custom bean class interface.
5. A method for EJB classloading comprising the steps of:
providing a computer including an EJB stored thereon, said EJB having an EJB implementation class associated therewith; and,
loading the EJB implementation class in a child classloader such that all direct references to the EJB implementation class from classes loaded in the

-7-

parent classloaders are removed.

6. The method of claim 5 wherein the custom classloader implementation used for the EJB implementation class is specifically disallowed from being loaded in the parent classloader.

7. The method of claim 5 wherein a custom interface is generated for the EJB implementation class, wherein said custom interface declares all methods that are invoked on the EJB implementation class by the classes loaded in the parent classloader.

8. The method of claim 7 wherein the classes in the parent classloader refer to the custom bean class interface.

9. A computer readable medium including instructions stored thereon which when executed cause the computer to perform the steps of:

providing a computer including an EJB stored thereon, said EJB having an EJB implementation class associated therewith; and,

loading the EJB implementation class in a child classloader such that all direct references to the EJB implementation class from classes loaded in the parent classloaders are removed.

10. The computer readable medium of claim 9 wherein the custom classloader implementation used for the EJB implementation class is specifically disallowed from being loaded in the parent classloader.

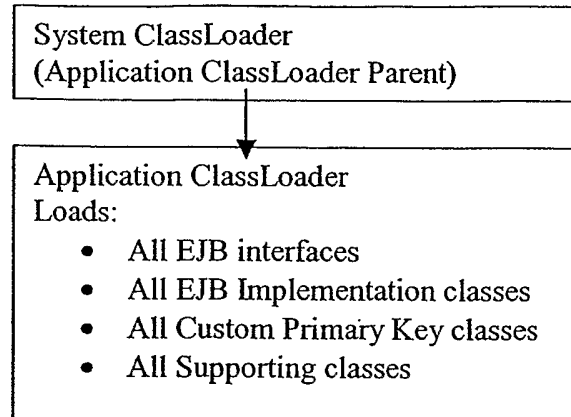
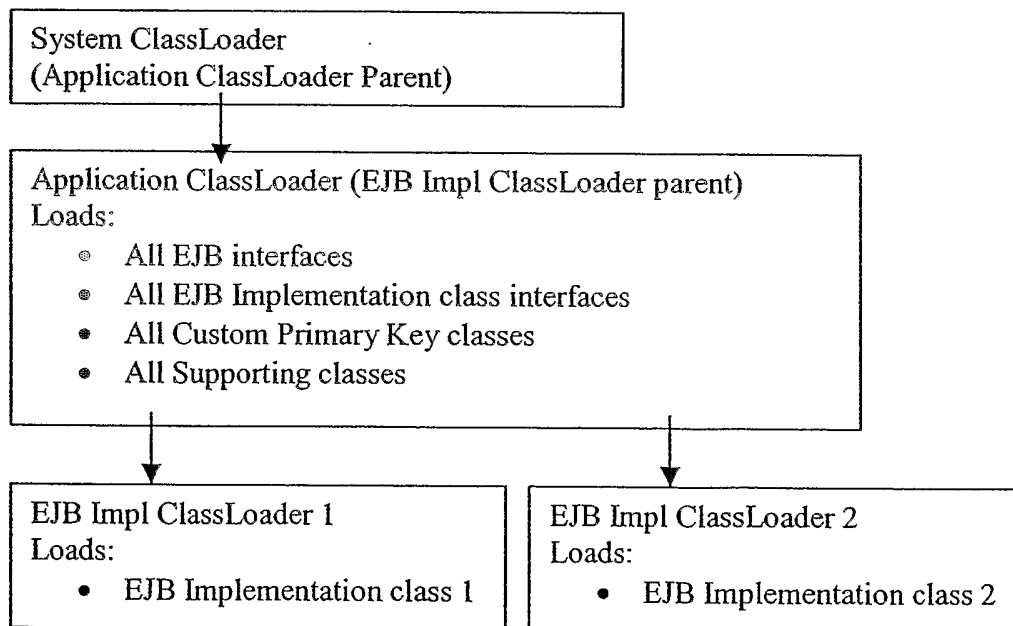
11. The computer readable medium of claim 9 wherein a custom interface is generated for the EJB implementation class, wherein said custom interface

-8-

declares all methods that are invoked on the EJB implementation class by the classes loaded in the parent classloader.

12. The computer readable medium of claim 11 wherein the classes in the parent classloader refer to the custom bean class interface.

1/1

Old classloading scheme:**Figure 1****New classloading scheme:****Figure 2**