# Managed Beans 1.0 Specification

Please send comments to: javaee-spec-feedback@sun.com

Final Release - 11/5/09 Roberto Chinnici, Bill Shannon

# Contents

**Managed Beans 1.0 Specification . . . . . . . . . . . . . . . . . . . . . . i**

**MB.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 1**

MB.1.1 What Are Managed Beans? . . . . . . . . . . . . . . . . . . . . . . 1

MB.1.2 Why Managed Beans? . . . . . . . . . . . . . . . . . . . . . . . . . 1

MB.1.3 Acknowledgements . . . . . . . . . . . . . . . . . . . . . . . . . . 2

**MB.2 Managed Beans Definition . . . . . . . . . . . . . . . . . . . . . . . 5**

MB.2.1 Basic Model . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5

MB.2.1.1 Component Definition . . . . . . . . . . . . . . . . . . . . . . . 5

MB.2.1.2 Naming . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

MB.2.1.3 Lifecycle and Resource Injection . . . . . . . . . . . . . . . . . 6

MB.2.1.4 Threading . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

MB.2.1.5 Interceptors . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

MB.2.2 Extensions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

MB.2.2.1 Component Definition . . . . . . . . . . . . . . . . . . . . . . . 7

MB.2.2.2 Naming . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7

MB.2.2.3 Lifecycle and Resource Injection . . . . . . . . . . . . . . . . . 8

MB.2.2.4 Threading . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

MB.2.2.5 Interceptors . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8

**Appendix MB.A: Revision History . . . . . . . . . . . . . . . . . . . . . 9**

MB.A.1 Changes in Proposed Final Draft . . . . . . . . . . . . . . . . . . . 9

**Appendix MB.B: Related Documents . . . . . . . . . . . . . . . . . . . 11**

# CHAPTER MB.1 Introduction

This specification defines Managed Beans for the Java Platform, Enterprise Edition.

## MB.1.1 What Are Managed Beans?

Managed Beans are container-managed objects with minimal requirements, otherwise known under the acronym “POJOs” (Plain Old Java Objects). They support a small set of basic services, such as resource injection, lifecycle callbacks and interceptors. Other, more advanced, aspects will be introduced in companion specifications, so as to keep the basic model as simple and as universally useful as possible.

## MB.1.2 Why Managed Beans?

Managed Beans offer a lightweight component model aligned with the rest of the Java EE Platform.

By supporting the common resource injection and lifecycle services, Managed Beans fit well into the Java EE programming model. At the same time, their lightweight nature makes them a natural starting point to encapsulate application functionality, with the knowledge that they can be morphed into more powerful components if and when the need occurs. In this sense, they can be seen as a Java EE platform-enhanced version of the JavaBeans component model found on the Java SE platform.

It won’t be missed by the reader that Managed Beans have a precursor in the homonymous facility found in the JavaServer Faces (JSF) technology. Indeed, the web tier has seen ample use of lightweight components, tied together with a variety of mechanisms. Managed Beans as defined in this specification represent a generalization of those found in JSF; in particular, Managed Beans can be used anywhere in a Java EE application, not just in web modules.

In introducing Managed Beans, we also have a longer-term goal: to provide a common foundation for the different kinds of components that exist in the Java EE platform, allowing us to align them better and reconcile their differences as much as possible.

Many of the distinctions that exist between component types in Java EE are historical in nature. In hindsight, the platform might have adopted a more uniform model where components start their existence as undistinguished Java objects and grow into more powerful entities by drawing on container-provided services. The annotation-based programming model introduced in version 5 of the Java EE Platform naturally lends itself to such an interpretation.

Managed Beans offers us a way to carry out such a refactoring of the existing components over time while offering developers some genuinely useful functionality in the short term.

## MB.1.3 Acknowledgements

This specification was created under the Java Community Process as JSR-316. The spec leads for the JSR-316 Expert Group were Bill Shannon (Sun Microsystems, Inc.) and Roberto Chinnici (Sun Microsystems, Inc.). The expert group included the following members: Florent Benoit (Inria), Adam Bien (Individual), David Blevins (Individual), Bill Burke (Red Hat Middleware LLC), Larry Cable (BEA Systems), Bongjae Chang (Tmax Soft, Inc.), Rejeev Divakaran (Individual), Francois Exertier (Inria), Jeff Genender (Individual), Antonio Goncalves (Individual), Jason Greene (Red Hat Middleware LLC), Gang Huang (Peking University), Rod Johnson (SpringSource), Werner Keil (Individual), Michael Keith (Oracle), Wonseok Kim (Tmax Soft, Inc.), Jim Knutson (IBM), Elika S. Kohen (Individual), Peter Kristiansson (Ericsson AB), Changshin Lee (NCsoft Corporation), Felipe Leme (Individual), Ming Li (TongTech Ltd.), Vladimir Pavlov (SAP AG), Dhanji R. Prasanna (Google), Reza Rahman (Individual), Rajiv Shivane (Pramati Technologies), Hani Suleiman (Individual).

We’d also like to acknowledge the work done by the JSF expert group over the years to define a notion of “managed beans” in that specification.

# CHAPTER MB.2

# Managed Beans Definition

This chapter defines the Managed Beans component model.

The presentation is organized in two sections. The first one describes the basic component model for Managed Beans. This is the minimal set of requirements for Managed Beans implementations. The second section describes how specifications that build on this one may extend the basic model to support more advanced functionality.

For example, in the basic component model, Managed Beans must provide a no-argument constructor, but a specification that builds on Managed Beans, such as CDI (JSR-299), can relax that requirement and allow Managed Beans to provide constructors with more complex signatures, as long as they follow some well-defined rules. Similarly, in the basic model, a Managed Bean component must be declared using the `ManagedBean` annotation, but other specifications are allowed to alter this requirement, e.g. to provide a purely XML-based way to turn a class into a Managed Bean.

## MB.2.1 Basic Model

### MB.2.1.1 Component Definition

A Managed Bean can be declared by annotating its class with the `javax.annotation.ManagedBean` annotation.

A Managed Bean must not be: a final class, an abstract class, a non-static inner class.

A Managed Bean may not be serializable, unlike a regular JavaBean component.

Managed Bean implementations must support Managed Beans that have a no-argument constructor.

### MB.2.1.2 Naming

A Managed Bean may optionally have a name, a `String`.
The name can be specified using an element of the `ManagedBean` annotation.

```
@ManagedBean(“cart“)
public class ShoppingCart { ... }
```

Managed Bean names must be unique within a Java EE module. It is an error if a Java EE module contains an EJB component and a Managed Bean with the same name.

For each named Managed Bean, Java EE containers must make available the following entries in JNDI, using the same naming scheme used for EJB components.

In the application namespace:
```
java:app/<module-name>/<bean-name>
```

In the module namespace of the module containing the Managed Bean:
```
java:module/<bean-name>
```

Java EE applications may obtain a new instance of a Managed Bean by looking up the corresponding names in JNDI or by using resource injection. See Chapter EE.5 of the Java EE Platform specification for more details.

### MB.2.1.3 Lifecycle and Resource Injection

Managed Beans may use the `javax.annotation.PostConstruct` and `javax.annotation.PreDestroy` annotations to identify methods to be called back by the container at the appropriate points in the bean’s lifecycle.

In a Java EE implementation, a Managed Bean may use any of the resource injection functionality laid out in Chapter EE.5 of the Java EE Platform specification, “Resources, Naming and Injection“. A Managed Bean does not have its own component-scoped “`java:comp`” namespace. For this reason, Managed Beans should define resources in the “`java:module`” namespace or above. JNDI lookup operations from a method defined on a Managed Bean will use the naming context of that method's caller.

#### MB.2.1.4 Threading

Method invocations on a Managed Bean execute in the same thread as the caller.

#### MB.2.1.5 Interceptors

A Managed Bean may use interceptors as defined in the Interceptor specification.

### MB.2.2 Extensions

Specifications that build on the present one (called here an "extension specification") may modify some of the aspects of the basic model, as detailed in the rest of this section.

#### MB.2.2.1 Component Definition

An extension specification may provide ways to declare a Managed Bean that go beyond those in Section MB.2.1.1, "Component Definition".

An extension specification may allow a Managed Bean to declare constructors with complex signatures, thus dropping the requirement that a no-argument constructor be present.

#### MB.2.2.2 Naming

An extension specification may offer alternative ways to name a Managed Bean, e.g. as a side-effect of placing some other annotation on the bean class, but, if specified, the `ManagedBean("...")` annotation takes priority, and with it the rules in Section MB.2.1.2, "Naming".

Of course an extension specification may also introduce one or more additional namespaces in which some or all Managed Beans get registered, either with the Managed Bean name defined in Section MB.2.1.2, "Naming" or with an independently defined name.

#### MB.2.2.3 Lifecycle and Resource Injection

An extension specification may define its own lifecycle model, adding e.g. pooling, sharing of instances, etc., beyond the basic model described in Section MB.2.1.3, “Lifecycle and Resource Injection”.

An extension specification may allow Managed Beans to have their own “`java:comp`” namespace.

#### MB.2.2.4 Threading

An extension specification may add its own threading requirements, overriding any requirements set in Section MB.2.1.4, “Threading”.

For example, invocations on a [proxy for] a Managed Bean may be performed using a different thread than the caller’s.

#### MB.2.2.5 Interceptors

An extension specification may add its own interceptor-like facilities to the predefined one.

For example, an extension specification may allow declaring type-safe interceptors, defined using a different set of APIs than those in the `javax.interceptor` package.

# APPENDIX MB.A

# Revision History

## MB.A.1 Changes in Proposed Final Draft

### MB.A.1.1 Additional Requirements

- First draft

### MB.A.1.2 Removed Requirements

- First draft

### MB.A.1.3 Editorial Changes

- First draft

# APPENDIX MB.B

# Related Documents

This specification refers to the following documents. The terms used to refer to the documents in this specification are included in parentheses.

*Java™ Platform, Enterprise Edition Specification Version 6*. Available at `http://java.sun.com/javaee`.

*Java™ Platform, Standard Edition, v6 API Specification* (Java SE specification). Available at `http://java.sun.com/javase/6/docs/api/index.html`.

*Enterprise JavaBeans™ Specification, Version 3.1* (EJB specification). Available at `http://java.sun.com/products/ejb`.

*JavaServer Faces 2.0* (JSF specification). Available at `http://jcp.org/en/jsr/detail?id=314`.

*Common Annotations for the Java™ Platform* (Common Annotations specification). Available at `http://jcp.org/en/jsr/detail?id=250`.

*Contexts and Dependency Injection for the Java EE Platform 1.0* (CDI specification). Available at `http://jcp.org/en/jsr/detail?id=299`.

Sun® microsystems

We make the net work.

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, California 95054, U.S.A.
650 960-1300

For U.S. Sales Office locations, call:
800 821-4643
In California:
800 821-4642

Australia: (02) 844 5000
Belgium: 32 2 716 7911
Canada: 416 477-6745
Finland: +358-0-525561
France: (1) 30 67 50 00
Germany: (0) 89-46 00 8-0
Hong Kong: 852 802 4188
Italy: 039 60551
Japan: (03) 5717-5000
Korea: 822-563-8700
Latin America: 650 688-9464
The Netherlands: 033 501234
New Zealand: (04) 499 2344
Nordic Countries: +46 (0) 8 623 90 00
PRC: 861-849 2828
Singapore: 224 3388
Spain: (91) 5551648
Switzerland: (1) 825 71 11
Taiwan: 2-514-0567
UK: 0276 20444

Elsewhere in the world,
call Corporate Headquarters:
650 960-1300
Intercontinental Sales: 650 688-9000