22. Managed BeansHibernate supports consuming many of its extension points as "managed beans". A bean being managed simply means that its creation and lifecycle are managed by a container of some sort. The main contract for managed beans is org.hibernate.resource.beans.spi.ManagedBeanRegistry Often these beans are managed by an external service, such as CDI. The contract org.hibernate.resource.beans.container.spi.BeanContainer is used to integrate the external container. ManagedBeanRegistry integrates support for a BeanContainer if one is specified. By default, Hibernate creates references to the beans and links their lifecycle to the SessionFactory. It supports a number of ways to influence how this process works. 22.1. Manageable Beans Jakarta Persistence defines support for resolving AttributeConverter and "entity listener" classes as managed beans. Additionally, Hibernate supports resolving the following integrations as managed beans: org.hibernate.type.descriptor.jdbc.JdbcType org.hibernate.type.descriptor.java.BasicJavaType org.hibernate.type.descriptor.java.MutabilityPlan org.hibernate.usertype.UserType org.hibernate.usertype.UserCollectionType org.hibernate.metamodel.EmbeddableInstantiator org.hibernate.envers.RevisionListener org.hibernate.id.IdentifierGenerator At the moment, when using either delayed or extended CDI access, resolving these Hibernate integrations as managed beans is disabled. 22.2. CDI BeanContainer Hibernate provides built-in support for using a CDI BeanManager as the BeanContainer. Jakarta Persistence indicates that the setting jakarta.persistence.bean.manager be used to pass along a CDI BeanManager to use, so Hibernate follows that approach. 22.2.1. CDI BeanManager - default By default, Hibernate follows the Jakarta Persistence requirements for using CDI BeanManager. Most importantly, this means accessing beans from the BeanManager immediately during bootstrap. In many cases this can cause circularity problems as CDI is often a consumer of persistence as well. In such cases, delayed or extended access should be used 22.2.2. CDI BeanManager - delayed Rather than accessing the CDI managed beans immediately, Hibernate can be configured to delay accessing the beans until first needed using hibernate.delay\_cdi\_access. Note however that this has some limitations{fn-cdi-availability} 22.2.3. CDI BeanManager - extended Sometimes the actual BeanManager instance is not known until after Hibernate has been bootstrapped. For such cases, Hibernate provides the org.hibernate.resource.beans.container.spi.ExtendedBeanManager contract, which is basically a promise or future for a BeanManager reference. An instance of ExtendedBeanManager passed as jakarta.persistence.bean.manager triggers this behavior. The ExtendedBeanManager implementation accepts the LifecycleListener passed to its #registerLifecycleListener method. It will call LifecycleListener#beanManagerInitialized and LifecycleListener#beforeBeanManagerDestroyed as lifecycle callbacks for the real BeanManager. Hibernate uses the LifecycleListener#beanManagerInitialized callback to get access to the real BeanManager. When used in WildFly, this is all automatically set up by the server 22.3. Custom BeanContainer Other containers (Spring, e.g.) can also be used and integrated by implementing BeanContainer and declaring it using hibernate.resource.beans.container.