Hibernate Envers automatically integrates with Hibernate ORM and provides a powerful and easy to use solution to document all changes performed on the audited entities.

But implementing a conditional audit requires more work. By default, Hibernate Envers registers a set of event listeners which are triggered by Hibernate ORM. You need to replace these listeners to customize Envers' audit capabilities.

Customize Envers' Event Listeners

Hibernate Envers' provides a set of listeners which are triggered by the following event types:

- *EventType.POST_INSERT*
- *EventType.PRE_UPDATE*
- *EventType.POST_UPDATE*
- *EventType.POST_DELETE*
- *EventType.POST_COLLECTION_RECREATE*
- *EventType.PRE_COLLECTION_REMOVE*
- *EventType.PRE_COLLECTION_UPDATE*

You can extend or replace these listeners to customize your audit log.

In this example, I want to ignore all updates of books which are not published. These are all *Book* entities which's *publishingDate* attribute is null. So, I will replace the existing listeners for events of *EventType.PRE_UPDATE* and *EventType.POST_UPDATE*.

Customize the Handling of *EventType.PRE_UPDATE* Events

Hibernate Envers provides the *EnversPreUpdateEventListenerImpl*. It already implements all the required logic to write the audit information. The only thing you need to do is to extend this class and ignore all update operations which you don't want to document in the audit log.

I do that in the *MyEnversPreUpdateEventListenerImpl* class. It extends Envers' *EnversPreUpdateEventListenerImpl* and overrides the *onPreUpdate* method.

Within that method, I check if the event was triggered for a *Book* entity and if the *publishingDate* is null. If that's the case, I ignore the event and in all other cases, I just call the method on the superclass.

```
public class MyEnversPreUpdateEventListenerImpl extends
        EnversPreUpdateEventListenerImpl {


    …


    @Override
    public boolean onPreUpdate(PreUpdateEvent event) {
        if (event.getEntity() instanceof Book &&
          ((Book) event.getEntity()).getPublishingDate() == null) {
            log.debug("Ignore all books that are not published.");
            return false;
        }

        return super.onPreUpdate(event);
    }
}
```

Customize the Handling of EventType.POST_UPDATE Events

You can replace the listener for the *EventType.POST_ UPDATE* event in the same way. The only difference is that you now need to extend the *EnversPostUpdateEventListenerImpl* class. I did that in the following code snippet.

```
public class MyEnversPostUpdateEventListenerImpl extends
        EnversPostUpdateEventListenerImpl {

    …

    @Override
    public void onPostUpdate(PostUpdateEvent event) {
        if (event.getEntity() instanceof Book &&
          ((Book) event.getEntity()).getPublishingDate() == null) {
            log.debug("Ignore all books that are not published.");
            return;
        }

        super.onPostUpdate(event);
    }
}
```

Register your Listener Implementations

You need to provide your implementation of Hibernate's *Integrator* interface to register your listener implementations.

The easiest way to do that is to copy and adapt the *EnversIntegrator* class. Hibernate Envers uses this class by default. All event listeners are registered in the *integrate()* method.

```
public class MyEnversIntegrator implements Integrator {

    @Override
    public void integrate(Metadata metadata,
            SessionFactoryImplementor sessionFactory,
            SessionFactoryServiceRegistry serviceRegistry) {

        ...

        if (enversService.getEntitiesConfigurations()
                .hasAuditedEntities()) {
            …

            listenerRegistry.appendListeners(
                    EventType.PRE_UPDATE,
                    new MyEnversPreUpdateEventListenerImpl(
                            enversService )
            );
            listenerRegistry.appendListeners(
                    EventType.POST_UPDATE,
                    new MyEnversPostUpdateEventListenerImpl(
                            enversService )
            );
        }
    }
    ...
}
```

The last thing you need to do to use your custom event listeners is to add the fully qualified name of your *Integrator* implementation in the *META-INF/services/org.hibernate.integrator.spi.Integrator* file.

```
org.thoughts.on.java.envers.MyEnversIntegrator
```

OK, that's all. Hibernate Envers will now use your custom event listeners.

Learn more

You can learn more about Hibernate Envers in the following posts:

- Getting Started with Hibernate Envers
- Query Data From Your Audit Log with Hibernate Envers
- Extend Hibernate Envers' Standard Revision