

Creating an execution engine listener

The following example shows how to create an engine listener

```
public class MyEngineListener implements EngineListener {

    private Project project;

    public MyEngineListener(Project project) {
        this.project = project;
    }

    @Override
    public void elementActivated(Element element, Collection<?> values) {
        System.out.println("--Activate Element-- : element name = " + ((NamedElement)element).getName());
    }

    @Override
    public void elementDeactivated(Element element, Collection<?> values) {
        System.out.println("--Deactivate Element-- : element name = " + ((NamedElement)element).
getName());
    }

    @Override
    public void eventTriggered(String eventID) {
        NamedElement element = (NamedElement)project.getElementByID(eventID);
        System.out.println("--Event Trigger-- : event name = " + ((NamedElement)element).getName());
    }

    @Override
    public void executionTerminated() {
        System.out.println("--Engine Terminated--");
    }
}
```

Once you have created an `EngineListener`, you can register it to the specified `ExecutionEngine` if you want to receive events occurring in each execution engine. All engine listeners of a specific engine will be activated under the conditions as follows

- An element is activated.
- An element is deactivated.
- An event is triggered.
- An engine is terminated.

```
public interface EngineListener {
    void elementActivated(Element element, Collection<?> values);
    void elementDeactivated(Element element, Collection<?> values);
    void eventTriggered(String eventID);
    void executionTerminated();
}
```

See [Creating a new execution engine](#) for more information about adding execution engine listeners to the `MyExecutionEngine` class.



Note

You can add more than one execution engine listener to an execution engine.