



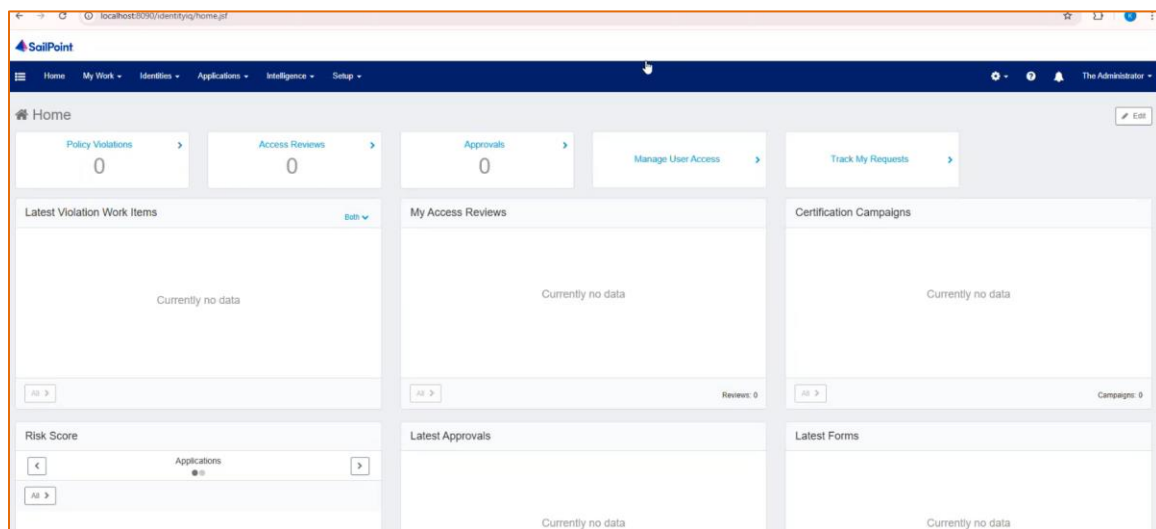
Knowledge Article: Hands-On with SailPoint IIQ (Java Intro)

This document summarizes key learnings and steps from the SailPoint IIQ Java Intro session. It includes practical takeaways

Introduction to SailPoint & IAM

Identity and Access Management (IAM) ensures employees have the right access to applications. SailPoint IIQ is a leading IAM tool, built on Java, and rules form the backbone of its customization.

Step 1 –SailPoint UI



Step 2 – Accessing Debug Page

Navigate to `identityiq/debug`. This debug console acts as a lightweight IDE where developers can write, test, and execute Beanshell/Java rules directly within SailPoint.





Step 3 – Creating Rules

Best practice: duplicate an existing rule, rename it, and remove the ID. SailPoint automatically generates a new ID.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <DOCTYPE Rule PUBLIC "sailpoint.dtd" "sailpoint.dtd">
3 <Rule created="175845275074" id="f1" language="beanshell" modified="175845264239" name="Main-Test1">
4   <Description>Allows all objects to be selected (no filtering)</Description>
5   <Signature returnType="sailpoint.object.Filter">
6     <Inputs>
7       <Argument name="log">
8         <Description>
9           The log object associated with the SailPointContext.
10        </Description>
11      </Argument>
12      <Argument name="context">
13        <Description>
14          A sailpoint.api.SailPointContext object that can be used to query the database if necessary.
15        </Description>
16      </Argument>
17      <Argument name="requestor" type="sailpoint.object.Identity">
18        <Description>
19          Identity that is making the Life Cycle Manager request.
20        </Description>
21      </Argument>
22      <Argument name="requestee" type="sailpoint.object.Identity">
23        <Description>
24          Identity on whose behalf the Life Cycle Manager request is being made. In the case of bulk requests,
25          this argument will be set to null when determining the rules that are visible to the requestor.
26          It will be provided once a selection has been made in order to determine whether or not the given requestee
27          should have access to the selected role.
28        </Description>
29      </Argument>
30    </Inputs>
31    <Returns>
32      <Argument name="filter">
33        <Description>
34          A Filter object that will be used to search for accessible request objects.
35        </Description>
36      </Argument>
37    </Returns>
38  </Signature>
39  <Source>import sailpoint.object.QueryInfo;
40
41  return new QueryInfo(null, false);</Source>
42 </Rule>
43
```

Step 4 – Writing Your First Rule

Example logic:

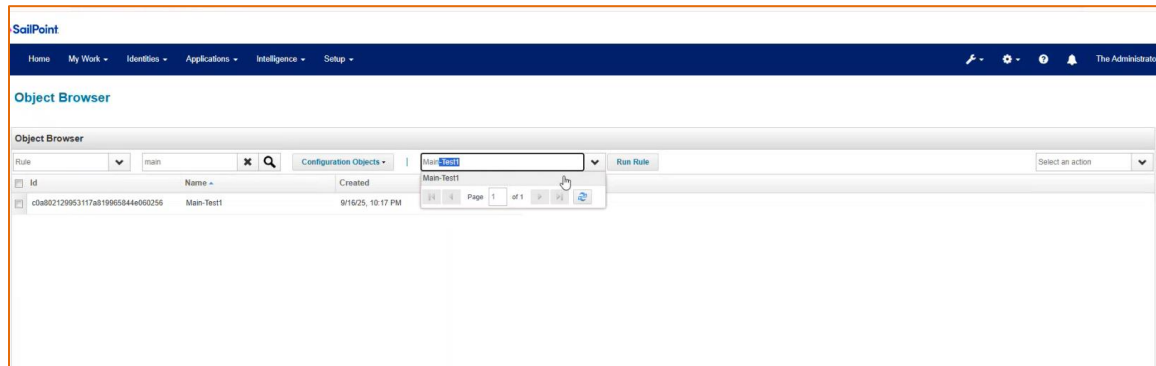
System.out.println("Hello World")

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <DOCTYPE Rule PUBLIC "sailpoint.dtd" "sailpoint.dtd">
3 <Rule created="175845275074" id="f1" language="beanshell" modified="175845264239" name="Main-Test1" significantModified="175845264239">
4   <Description>Allows all objects to be selected (no filtering)</Description>
5   <Signature returnType="sailpoint.object.Filter">
6     <Inputs>
7       <Argument name="log">
8         <Description>
9           The log object associated with the SailPointContext.
10        </Description>
11      </Argument>
12      <Argument name="context">
13        <Description>
14          A sailpoint.api.SailPointContext object that can be used to query the database if necessary.
15        </Description>
16      </Argument>
17      <Argument name="requestor" type="sailpoint.object.Identity">
18        <Description>
19          Identity that is making the Life Cycle Manager request.
20        </Description>
21      </Argument>
22      <Argument name="requestee" type="sailpoint.object.Identity">
23        <Description>
24          Identity on whose behalf the Life Cycle Manager request is being made. In the case of bulk requests,
25          this argument will be set to null when determining the rules that are visible to the requestor.
26          It will be provided once a selection has been made in order to determine whether or not the given requestee
27          should have access to the selected role.
28        </Description>
29      </Argument>
30    </Inputs>
31    <Returns>
32      <Argument name="filter">
33        <Description>
34          A Filter object that will be used to search for accessible request objects.
35        </Description>
36      </Argument>
37    </Returns>
38  </Signature>
39  <Source>import sailpoint.object.QueryInfo;
40
41  return new QueryInfo(null, false);</Source>
42 </Rule>
43
```



Step 5 - Search for Your Rule

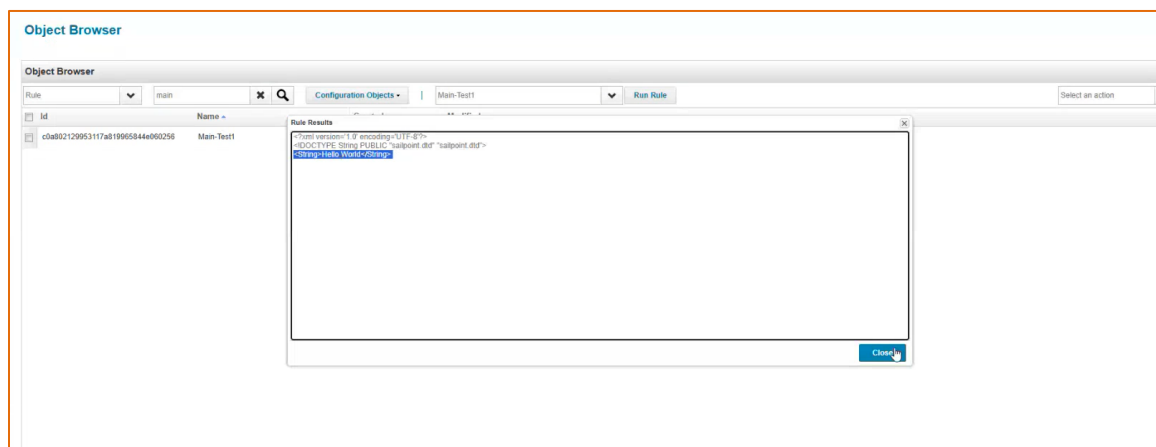
In the Object Browser, select Rule as the object type. Search by the name of your rule (e.g., 'main-test1') and open it to verify the logic.



Step 6 – Quick Output with Return

To display results instantly in the Debug console:

return "Hello World";



Key Programming Basics

The session also covered Java/Beanshell fundamentals relevant for IIQ development:

- Data Types – int, string, boolean
- String operations – concatenation (useful for building display names)
- Importance of return statements
- Difference between system logs vs console outputs



Key Takeaways

- Rules are the core of SailPoint IIQ development.
- Use Debug page to test and prototype safely.
- Duplicate rules instead of creating from scratch.
- `System.out.println` → server logs, `return` → console output.
- Java/Beanshell basics are essential for SailPoint customizations.