Lab: JQL

Estimated time: 30 minutes

In this lab, you will:

1. Create a basic search and view the JQL query.
2. Create JQL queries with the help of autocomplete and column sorting.
3. Use functions as values.
4. Use time unit qualifiers.
5. Use various operators.
6. Use Boolean operators.

> *Note: These instructions assume that you have projects from the previous labs. If you have other projects, you can modify the queries to make them work for your projects.*

1: Create a basic search and view the JQL query.

1. Open basic search by clicking on **Filters**. If needed, click on the **Switch to basic** link to view basic search.
2. In basic search, search for all issues of `projectA`.
3. Click on the **Switch to JQL** link to enter advanced/JQL search. View the JQL query associated with the basic search. You can use this technique of switching from basic to advanced/JQL search to help "write" JQL queries. This is helpful because you might need to copy the JQL to other parts of the Jira interface (or to other products like Confluence).
4. Click on column headers in list view to sort the results. Notice the changes to the query.
5. Experiment with creating other basic searches and viewing the resulting JQL query.

> *Congratulations, you have created a basic search and viewed the resulting JQL query.*

2: Create JQL queries with the help of autocomplete and column sorting.

1. Enter advanced/JQL search (if necessary).
2. Clear the current JQL query.
3. Create and execute a query that finds all issues in the `projectA` project:
   - With the JQL textbox selected, press `p` and select `project` from the autocomplete dropdown.
   - Press the space bar to view operator autocomplete.
   - Select the equals (=) operator.
   - Press the space bar to view value autocomplete.
   - Select `projectA`.
   - Press **Enter** to execute the query.
4. Add an ORDER BY clause to the query by clicking on the `Summary` heading in list view.
5. Experiment with creating other JQL queries.

> *Congratulations, you have created JQL queries with the help of autocomplete and column sorting.*

3: Use functions as values.

1. In advanced/JQL search, use autocomplete to find all issues assigned to you using the currentUser() function. `assignee = currentUser()`
2. Find all issues that were created since the `startOfWeek()` . You will use the `>` operator.
3. In a separate browser window, perform a web search for `Jira advanced searching functions reference` . Click on the Atlassian documentation link. Click on **Cloud** in the upper right. View the available advanced/JQL searching functions.
4. Back in Jira, experiment with other searches that use functions as values.

> *Congratulations, you have used functions as values.*

4: Use time unit qualifiers.

1. In advanced/JQL search, enter the query `updated > -2d` to find issues that were updated in the past 48 hours.
2. Modify the previous query to find issues updated in the past two hours.
3. Using basic search (you may need to clear the existing advanced/JQL search and press **Enter** to enable the basic search link), find all issues that were updated in the past hour. Switch to advanced/JQL search and notice that a time unit qualifier is used in the query. This is a helpful way to write queries with time unit qualifiers.
4. In advanced/JQL search, find all issues that were updated yesterday or today. (Hint: use the startOfDay() function with an argument of a time unit qualifier of `-1d` .)
5. Experiment with other searches that use time unit qualifiers.

> *Congratulations, you have used time unit qualifiers.*

5: Use various operators.

1. In advanced/JQL search, enter `assignee` and press the space bar to view the available operators.
2. Select the `is` operator and press the space bar. Notice that the only valid value is `EMPTY` . Select it.
3. Execute the query to find all unassigned issues.
4. Execute the query `text ~ "item 1 NOT 2"` to find issues with text fields that contain `item` and `1` but not `2` .
5. Modify the previous query to capitalize `ITEM` and verify that text strings are not case-sensitive.
6. Modify the previous query to change `not` to lowercase and verify that the query results are different. The NOT keyword in text field searches is case-sensitive. This query is the same as `item 1 2` and probably returns no issues because no issues have a 1 and a 2.
7. In a separate browser window, perform a web search for `Jira advanced searching operators reference` . Select the Atlassian documentation. Click on **Cloud** in the upper right. Explore the reference.

8. Back in Jira, experiment with using other operators.

> *Congratulations, you have used various operators.*

6: Use Boolean operators.

1. In advanced/JQL search, use a Boolean operator to find issues with an `assignee` of `currentUser()` and a `status` of `Done`.
2. Use the `NOT` Boolean operator to find issues that do not have a `status` of `Backlog`. Verify that this query is equivalent to `status != Backlog`.
3. Use the `OR` operator to find issues with a `status` of `Selected for Development` or `In Progress`.
4. Create a query that is equivalent to the previous query using the `in` operator. (See answer at the end of lab.)
5. Create a query containing multiple Boolean operators that returns different results depending on if you use parentheses in the query. Examples: `NOT (project = projectA OR project = projectB)` `(status = Done OR status = "To Do") AND summary ~ "item 1"`.
6. Experiment with other Boolean operators.

> *Congratulations, you have used Boolean operators and completed this lab.*

Answer to 6.4: `status in ("Selected for Development", "In Progress")`