Create Query components

No SQL query components

Apstrata developers can persist their data inside NoSQL key/value pair structures called **documents,** which are saved into their application **store**. To retrieve the data that you have persisted in your documents, you can define sophisticated queries to search for document matching your criteria, using the Query API.
While you can invoke the Query API from your client side and pass it all the required parameters to execute your query, you will most of the time need to reuse this same logic across clients or event across server-side components. Since Apstrata always has modularity and reusability in mind, it allows you to encapsulate your query logic into components, called "Saved Queries" using the SaveQuery API.
Back to the map **Next station**: Geospatial queries

Example 1: load my saved game

You are developing this cool game that runs on mobile devices. Although your game is addictive, your players will stop playing from time to time, therefore, you decide to persist the games as Apstrata documents. When your players get back to the game, you need to load its data back to the device, i.e. get the corresponding document using its **document key** (in our example, we assume our document has the following key F0BBBBEC7EE9EF59EAC220599BADFD08).

The simplest way for you to create a "Saved Query" is to log in to the Apstrata workbench, click on "Manage App > Saved Queries > New". This opens a query editor within which a predefined template is already loaded.

```
<query>
        <executeACL>creator</executeACL> <!-- This query only authorizes the retrieval of a document
created by the caller of the query -->
        <store>DefaultStore</store> <!-- Put the name of your store here. "DefaultStore" is the default
value -->
        <condition><![CDATA[ apsdb.documentKey<string> = {docKey}]]></condition> <!-- {docKey} is a
mandatory parameter to pass to the query -->
        <returnedFields> <!-- The list of document fields we want to return -->
                <field>player</field>
                <field>level</field>
                <field>lives</field>
                <field>score</field>
        </returnedFields>
</query>
```

Try it!

From the "Saved Queries" editor, click "Save" to save your query then "Run". The workbench opens a form where you have to enter the "docKey" field with a value that matches one of your game document keys. Once done, click "Go" to execute the saved query.

Invoke the saved query

Navitabs License Error

The number of licensed users is incompatible with the number of licensed users of Confluence. Please contact your system administrator.

License Details

Example 2: get top 10 highest scores, in descending order, grouped by level

In your game, you would like to display the 10 best scores of all times, as well as the corresponding level player name. Using Apstrata, you can simply create a query that returns no more than 10 results and such as "score > 0", also specifying that the results should be sorted in descending order.

Once again, let us use the Apstrata workbench to create a saved query:

```
 <query>
        <executeACL>creator</executeACL><!-- This query only authorizes the retrieval of a document created
by the caller of the query -->
        <store>DefaultStore</store>
        <condition><![CDATA[ score<numeric> > 0]]></condition><!-- we search for documents that have a
strictly positive score -->
        <returnedFields> <!-- list of fields to return -->
            <field>player</field>
                <field>level</field>
                <field>lives</field>
                <field>score</field>
        </returnedFields>
    <aggregate>
        <expression>Max($score)</expression> <!-- we query by Max score -->
                <global>true</global> <!-- the aggregation is applied to all the documents in the store
that match the condition -->
        <groupBy> <!-- group results by level -->
            <field>
                <name>level</name>
                <type>numeric</type>
            </field>
        </groupBy>
    </aggregate>
    <resultsPerPage>10</resultsPerPage> <!-- only return 10 results per page. since we do not mention a
page number, we will get the top 10 scores -->
</query>
```

Invoke the saved query

Navitabs License Error

The number of licensed users is incompatible with the number of licensed users of Confluence. Please contact your system administrator.

License Details

Dig deeper

- Saved Queries API

Related tutorials

Apply validation	Create Query components	Persist your content	Query your content