```
import
org.springframework.web.util.HtmlUtils;
import com.tangosol.net.CacheFactory;
import com.tangosol.net.NamedCache;
import com.tangosol.util.Filter;
import com.tangosol.util.QueryHelper;
import com.tangosol.util.filter.AlwaysFilter;
import
com.tangosol.util.filter.ContinuousQueryCa
che;

import java.util.ArrayList;
import java.util.Collections;

public class QueryTab {

    // Cache name to be used in queries
    private String cacheName;

    // Getter for cacheName
    public String getCacheName() {
```

```
        return cacheName;
    }

    // Setter for cacheName with input validation and sanitization
    public void setCacheName(String cacheName) {
        if (!isValidCacheName(cacheName)) {
            throw new IllegalArgumentException("Invalid cache name provided.");
        }
        // Sanitize cache name
        this.cacheName = HtmlUtils.htmlEscape(cacheName);
    }

    // Executes a query on the cache
    public Object executeQuery(String query) {
        // Validate and decode the query
```

```
        String decodedQuery =
decodeQueryString(query);

        // Sanitize the cache name before
accessing the cache
        final String sanitizedCacheName =
HtmlUtils.htmlEscape(getCacheName());

        // Access the cache safely
        NamedCache cache =
CacheFactory.getCache(sanitizedCacheNa
me);
        Filter filter =
QueryHelper.createFilter(decodedQuery);

        // Apply the filter to the cache
        ContinuousQueryCache queryResults
= new ContinuousQueryCache(cache,
filter);
        ArrayList<String> cacheKeyList = new
ArrayList<>();
```

```
        // Process cache keys from the query
results
        for (Object cacheKey :
queryResults.keySet()) {

cacheKeyList.add(cacheKey.toString());
        }

        // Log the number of keys found
        LOGGER.debug("Found " +
cacheKeyList.size() + " keys for the query: "
+ query);

        // Sort the result list and return
        Collections.sort(cacheKeyList);
        return cacheKeyList;
    }

    // Decodes and sanitizes the query string
    private String decodeQueryString(String
```

```
query) {
        if (query == null ||
query.trim().isEmpty()) {
            throw new
IllegalArgumentException("Query string
cannot be null or empty.");
        }

        String decodedQuery = query;

        // Replace backslashes and slashes
safely
        if (decodedQuery.contains("\\")) {
            decodedQuery =
decodedQuery.replace("\\", "\\\\");
        }
        if (decodedQuery.contains("/")) {
            decodedQuery =
decodedQuery.replace("/", "\\/");
        }
```

```
        // Sanitize the decoded query
        return
HtmlUtils.htmlEscape(decodedQuery);
    }

    // Validates the cache name for allowed
characters
    private boolean
isValidCacheName(String cacheName) {
        // Allow only alphanumeric,
underscore, and hyphen in cache names
        return cacheName != null &&
cacheName.matches("^[a-zA-Z0-9_-]+$");
    }
}
```