Developing a high performance API for IBM Cognos integration using WebSphere DynaCache

Ravindran Yelchur (ryelchur@us.ibm.com)
Software Engineer
IBM

Li Xie (lixie@us.ibm.com)
Software Engineer
IBM

Performance is a key measurement for an API, especially when the API is integrated with services that may not respond quickly. One example of such a service is a complex IBM® Cognos® Business Intelligence (BI) report. This tutorial describes a solution for a RESTful API design that uses WebSphere® DynaCache to help improve response times for performance sensitive API clients, such as a user interface. The tutorial also discusses a few critical factors to effectively build APIs with performance in mind, such as cache key configuration and invalidation rules.

Introduction

In today's API economy, there is a high acceptance rate for adopting APIs into business solutions and architecture designs. However, not much attention is paid to the quality factors of an API. If you are serious about the quality of your business solutions, you need to be serious about the scalability, accessibility, performance, and security of an API that you are building.

Speaking about performance, there are scenarios where APIs can integrate with all types of data sources and services and depend on a back-end service that may not return the response within seconds. This back-end service can become a bottleneck for your API client, and therefore can impact the entire business solution.

For example, in digital marketing space, a rich user interface (UI) invokes an API that pulls data from IBM Cognos reports. Due to the complexity of the data process as well as business intelligence logic, the execution time can be in minutes. This, by any standard, is not acceptable for an end user facing business critical applications.

Can you build an API that can out-perform its back-end services? The answer is "yes". You can achieve this goal by leveraging WebSphere DynaCache into the API design and make the asset
highly reusable for similar business scenarios as a high performance API integrated with slow back-end services.

**Overview of RESTful APIs for Cognos BI reports**

Cognos Business Intelligence (BI) can create reports and analyze data from a variety of data sources. The Cognos BI module consists of many components. The Cognos Report Studio is used to author reports and to publish them to the Cognos BI server. You can access these reports through the Cognos portals.

Cognos BI also allows published reports to be accessed programmatically using the Cognos Mashup Services via REST and SOAP.

As illustrated in Figure 1, the Cognos Mashup Service (CMS) provides basic accessibility for individuals to consume reports remotely. However, for a high demand business application serving thousands of users, you will need to address accessibility, scalability, and data quality, in addition to the service provided by CMS out-of-the-box.

**Figure 1. Overview of Cognos Mashup Service interface**

In the following sections, we will discuss how to build an API for an existing Cognos report using REST.

Each report that is authored in the Cognos Report Studio is identified by a unique name that the user provides to identify a report. Also, the Cognos Framework Manager generates a unique ID for each report, which can also be used to identify the report. The client application needs this unique search path or a unique ID to invoke the report.

The following is a sample URL structure to invoke a report using the report name and its path:

http://webservername:portnumber/<cognos contextpath>/rds/reportData/path/<report name with the complete path>?reportOption1=value1&reportOption2=value2&p_parameter1=value1&p_parameter2=value2...

The following is a sample URL structure to invoke a report using the report ID:
Developing a high performance API for IBM Cognos integration using WebSphere DynaCache

The reports take various report options, one such option is `fmt=DataSet`.

This report option allows the result of the Cognos report to be provided in an XML format that contains only the data, but not the report layout. Refer to the Cognos mashup documentation for the various report options.

**Accessing Cognos reports from your Java application**

In this tutorial, we will show how to access the Cognos report using a simple HTTPClient without having any Cognos SDKs as dependencies on the application. However, you may use the Cognos SDK for more control and for complex manipulations. Refer to the Cognos documentation on using the Cognos SDK.

Listing 1 shows the sample application to invoke the Cognos report URL via an Apache HTTPClient library.

**Listing 1. Invoking the Cognos report via HTTP**

```java
HttpClient client = new HttpClient();
client.getHostConfiguration().setHost(<hostname>, <port number>, "http");

// parameter strings are application specific parameters separated by &

GetMethod reportUrl = new GetMethod(cognosReportUrl);
client.executeMethod(reportUrl);
String xmlReturnString = getResponseBodyAsString(redirect.getResponseBodyAsStream());
reportUrl.releaseConnection();
```

If the Cognos server is secured, then instead of using `http`, use `https` and make multiple HTTP calls for the login to execute the report, and then finally to logoff. For the logon and logoff to the Cognos URL, you can use the following format: `https://webservername:portnumber/<cognos context path>/rds/auth/logon`.

The credentials for the logon are passed as an XML string. Refer to the Cognos Mashup Service documentation for more details. Figure 2 shows the interactions from API clients to Cognos reports with various data source connections.
Using the WebSphere DynaCache framework

The DynaCache framework is the service that comes with WebSphere by default. This is a lightweight caching solution for distributed applications to utilize your caching needs.

Dynamic cache works within an application server Java™ Virtual Machine (JVM), intercepting calls to cacheable objects. For example, it intercepts calls through a servlet service method or a command execute method. It either stores the output of the object to the cache or serves the content of the object from dynamic cache.

In this tutorial, we will cache the objects using cachespec.xml. First, you need to enable dynamic cache in WebSphere. Then, you can create a cachespec.xml file and place it in WEB-INF of the web module.

There is a default cache monitor Enterprise Application (EAR) that is packaged with the WebSphere installation. You can deploy and configure this EAR as needed. It provides a simple UI for users to monitor the cache objects and invalidates them from the cache. Figure 3 shows an example of cached objects and their attributes at runtime via the WebSphere Cache Monitor. Refer to the cacheable objects documentation on setting up dynamic cache using cachespec.xml.
Putting it all together

Now you can retrieve the data from the Cognos reports and integrate it with your application. The next question is how to enhance performance.

In many cases, the Cognos reports might be building analytics by getting complex data across distributed systems, therefore the report could take a few minutes to run. If these reports are being integrated in the applications, keeping the user waiting that long will not enhance the user’s interaction. The next section discusses how to integrate the Cognos report with the user application to perform at a much faster rate.

From the previous sections, you have seen how you can integrate the data from the Cognos reports in the user application. We also briefly discussed about DynaCache offered by WebSphere.

Dynamic caching is highly effective for the reports that take long time to run. In addition to regular on-demand API services, you can schedule the selected APIs to run during off-peak hours in order to pre-load those high-demand and long-running report objects into dynamic cache.

When the user requests the application to load the data from the report, the application looks up DynaCache and loads the content from the cache, thus instantly rendering the data to the user. You can set up the cache to have a life cycle on when the data gets obsolete and to discard it. Figure 4 shows how DynaCache works to enhance the performance of a Cognos API.

Figure 4. Enhance application performance caching report data using WebSphere DynaCache
A CognosReportHelper POJO is created to encapsulate the logic and invoke the Cognos report using the HTTP client. A CognosReportCacher class that extends com.ibm.wbephere.command.CacheableCommandImpl is then implemented. This class contains the report data in an XML string format and is the one that gets cached in DynaCache for each unique report instance. The two methods, isReadyToCallExecuted and performExecute, are critical to implement in this class. The performExecute method is where the call to the Cognos report is made by calling the exposed method on the CognosReportHelper class (see Figure 6).

The user application code will then call the execute method on CognosReportCacher (see Figure 5), which is inherited from CacheableCommandImpl.

**Note:** The sample codes in this tutorial are shown for illustration purposes. Further tailoring and testing are recommended for your production usage.

**Figure 5. Sample code for CognosReportCacher**

```java
public class CognosReportCacher extends CacheableCommandImpl {
    private String _uri = null;
    private String _reportXMLStr = null;

    public CognosReportCacher(String uri) {
        _uri = uri;
    }

    public String getReportXMLStr() {
        return _reportXMLStr;
    }

    public boolean isReadyToCallExecute() {
        boolean retValue = false;
        if (com.ibm.websphere.cache.DynamicCacheAccessor.isCachingEnabled()) {
            if (!StringUtils.isEmpty(_uri))
                retValue = true;
        }
        return retValue;

    }

    public void performExecute() throws HttpException, IOException, Exception {
        CognosReportHelper reportHelper = new CognosReportHelper(_uri);
        String reportXMLStr = reportHelper.executeReportOnCognos(_uri);
        _reportXMLStr = reportXMLStr;
    }

    public String getReportKey() {
        return _uri;
    }
}
```
Developing a high performance API for IBM Cognos integration using WebSphere DynaCache

Figure 6. Sample code for CognosReportHelper

```java
public class CognosReportHelper {
    public String executeUsingHttpProtocol(String url)
        throws HttpException, IOException {
        String response = "";
        HttpClient client = new HttpClient();
        GetMethod method = new GetMethod(url);
        client.executeMethod(method);
        response = getResponseBodyAsString(method.getResponseBodyAsStream());
        return response;
    }

    private String getResponseBodyAsString(InputStream inStream)
        throws IOException, UnsupportedEncodingException {
        ByteArrayOutputStream outputStream = new ByteArrayOutputStream();
        byte[] readBytes = new byte[512];
        int byteCount = 0;
        String responseStr = null;
        while ((byteCount = inStream.read(readBytes, 0, readBytes.length)) > 0) {
            outputStream.write(readBytes, 0, byteCount);
            responseStr = new String(outputStream.toByteArray(), "UTF-8");
        }
        return responseStr;
    }
}
```

In the user application, the Cognos report is invoked as shown in Figure 7.

Figure 7. Sample code for user application

```java
public class MyReportService {
    // the cognosReportURL is an url that contains the information on how
    // to reach the report along with report options and parameters
    // that are needed for the report to run.

    public void getReportData(String cognosReportURL)
        throws Exception {
        CognosReportCacher reportCache = new CognosReportCacher(cognosReportURL);
        reportCache.execute();
        String xmlStr = reportCache.getReportXMLStr();
        // the application now can parse the returned xml string
        // and use the data to enhance and transform to ones need.
    }
}
```

Figure 8 illustrates the sequence diagram for the sample code.
Figure 8. Sequence diagram

Listing 2 shows the cachespec.xml definition.

Listing 2. cachespec.xml definition

```xml
<?xml version="1.0"?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
  <cache-entry>
    <class>command</class>
    <sharing-policy>not-shared</sharing-policy>
    <name>com.myservice.cognos.reports.CognosReportCacher</name>
    <cache-id>
      <component type="method" id="getReportKey">
        <required>true</required>
      </component>
      <timeout>604800</timeout>
      <inactivity>604800</inactivity>
    </cache-id>
  </cache-entry>
</cache>
```

getReportKey() is a callback method required by cachespec.xml for the DynaCache framework. This functions as the cache ID to provide a unique key for a report object in the cache, where you can have the repeated API calls routed to that cache. If the same report is invoked by different parameters, the parameters are added to the key to make it unique. As you can see in the sample code shown in Figure 5, getReportKey() returns a full URL of the caller's request. The timeout parameter value is in seconds, after which the object is discarded from the cache.

Another important consideration in the solution design is the cache invalidation rule. This rule specifies how you want the cached objects to expire so that the cache is refreshed properly. This should be a business-driven task, depending on how frequent the data source changes and how significant it is to synch up the API return with the data source.

In this example, the data source gets refreshed once a week. Therefore, a simple rule by a cache timeout (604800) is made in its place. For more complex scenarios, you can invalidate the cache
by rules or design trigger events to clear the cache programmatically. This is significant for high performance APIs with production strength. In other words, you want to have a balance of high performance and data consistency according to your business needs.

**Promoting to WebSphere eXtreme Scale**

The dynamic cache engine is the default cache provider for the basic dynamic cache capability for your APIs. However, if you desire extended capabilities, such as transactional support, improved scalability, and high availability, you can, with additional cost, configure WebSphere eXtreme Scale as a cache provider.

The good news is that you do not need to rewrite your API code to work with eXtreme Scale. The steps to do so are:

1. Install the eXtreme Scale client or the eXtreme Scale client and server packages in your application server for the remote server and other topologies, respectively.
2. Configure the eXtreme Scale dynamic cache provider as your cache provider on WebSphere Application Server.
3. Restart the server.

**Conclusion**

This tutorial showed how to integrate your application with data from an existing Cognos report using the Cognos Mashup Service via REST. In addition, it showed how you can cache the report data by using WebSphere DynaCache for better performance.

You can extend this solution to other integration scenarios with slow back-end services. The DynaCache configurations, once in place, can stay for the same integration patterns, regardless of the back-end service. For more capacity with caching, we recommend that you replace WebSphere DynaCache, the default cache provider, with WebSphere eXtreme Scale as a pluggable service provider to avoid rewriting your API for this switch.
Resources

- WebSphere Application Server Knowledge Center: Configuring cacheable objects with the cachespec.xml file
- WebSphere Application Server Knowledge Center: Configuring dynamic cache (DynaCache) to use the WebSphere eXtreme Scale dynamic cache provider
- WebSphere eXtreme Scale
- IBM Cognos Mashup Service
- developerWorks WebSphere zone
About the authors

Ravindran Yelchur

Ravi Yelchur is an experienced IBM Software Services Consultant with 20+ years of experience. His area of expertise is in WebSphere and J2EE technologies. He is also an IBM certified Cognos report author.

Li Xie

Li Xie is a certified IT Specialist (Expert Level) and an IBM Software Services Consultant with 20+ years of experience. He has co-authored an IBM Redbook for WebSphere Commerce and other articles in the area of application integration. He is now working on Business Intelligence APIs for IBM Data Driven Marketing.

© Copyright IBM Corporation 2015
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)