TopCoder Direct API
Deployment Guide

Revision History

Author	Revision Number	Date
Ghost_141	1.0	2014/7/21
TCSASSEMBLER	2.0	2014/8/25

Deployment Instructions.....3
1.Organization of Submission.....3
2.Application Setup.....3
5.Database Setup.....4
6.Application Server Setup.....4
7.Maven setup.....4
8.Install tech.core.api.....5
9.Deployment Instructions.....5
10.Deploy tc api.....5
11.Starting.....5
12.Verification.....6
13.Resource Contact List.....8

Deployment Instructions

1. Organization of Submission

direct-api
: Contains the direct api source code and pom.xml which is used by maven.

doc
: Contain this document.

test_files
: Contains the test data sql files that will be used for test purpose.

tomcat
: Contains the updated tomcat configuration files.

2. Application Setup

```
JDK 1.6
Informix 11.5
Maven 3.x
Tomcat 7.0.54
Git 1.8+
Node 0.10.20+
NPM 1.3+
```

3. Application Properties File

<submission>/direct-api/src/main/resources/app.properties

Property Name	Description	Example Setting
oauthClientId	The client id of auth0 account	CMaBuwSnY0Vu68PLrWatvvu3iIiGPh7t
oauthClientSecret	The secret of auth0 account	ZEEIRf_aLhvbYymAMTFefoEJ_8y7ELrUaboMTmE5fQoJXEo7sxxyg8IW6gtbyKuT

Since we need nodejs version tc-api to get the jwt token for this challenge(I will explain reason later) these two properties need to be consist with value defined here: https://github.com/cloudspokes/tc-api/blob/master/config/tc-config.js#L78 and here https://github.com/cloudspokes/tc-api/blob/master/config/tc-config.js#L80

<submission>/direct-api/src/test/resources/test.properties

Property Name	Description	Example Setting
oauthClientId	The client id of auth0 account	CMaBuwSnY0Vu68PLrWatvvu3iIiGPh7t
oauthClientSecret	The secret of auth0 account	ZEEIRf_aLhvbYymAMTFefoEJ_8y7ELrUaboMTmE5fQoJXEo7sxxyg8IW6gtbyKuT
heffanToken	The token for user heffan	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.xxxx
testerToken	The token for user dok_tester	EyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.xxxxx
superToken	The token for user super	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.xxxx
jdbc-server	The server ip address that hold topcoder informix database	54.197.180.131

The oauthClientId and oauthClientSecret should have same value as oauthClientId and oauthSecret in app.properties file.

4. Build Scripts Setup

The following maven lifecycle or plugin command will be used in this challenge:

- clean - Clean up after the build.
- install - Install the built artifact into the local repository.
- test - Run all the test files.
- tomcat7:deploy-only - Deploy a WAR to Tomcat without forking the package lifecycle.

5. Database Setup

The database svn revision I'm working on is 85095
Following steps will help you setup the database for test purpose.

1. Upload my submission.zip to vm server /mnt/shared/temp_files.
   ```
   scp submission.zip tc@xxxx:/mnt/shared/temp_files
   ```
2. Login server as 'tc' account
   ```
   ssh tc@xxx.xxx.xxx.xxx
   ```
3. Switch to 'informix' account (The password is 1nf0rm1x)
   ```
   su informix -l
   ```
4. Execute following commands to clean the data and insert test data
   ```
   cd /mnt/shared/temp_files
   unzip submission.zip
   cd submission/test_files
   dbaccess - clean.sql
   dbaccess - test_data.sql
   ```

Now your database is ready for test.

6. Application Server Setup

The tomcat application server doesn't support JTA. So in order to use it for transaction management in direct api application, we need to modify something for it.
Following these steps:

1. Download and install the `tomcat 7.0.54`
2. Make sure you add `CATALINA_HOME` in your class path.
   If you don't know how to do it, please follow the steps in this page
   http://tomcat.apache.org/tomcat-7.0-doc/setup.html
3. Unpack my submission.zip
   ```
   unzip submission.zip
   ```
4. Copy all files under <submission>/tomcat folder to your tomcat server path
   ```
   cp -r <submission>/tomcat/* $CATALINA_HOME/
   ```
5. Update the `SERVER_IP` in `CATALINA_HOME/conf/server.xml` to your vm ip address
   This step is very important. The server will failed to startup if you miss this step.
   Tips: You need to Update `SERVER_IP` in three places.

7. Maven setup

In order to compile and deploy the direct application to tomcat server we need to add something in the maven setting file.
(For maven installation, please refer to its official site: http://maven.apache.org/)
Edit the `settings.xml` in your maven home folder in your environment.(Add one if you don't have it)
Add following content in it

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0
http://maven.apache.org/xsd/settings-1.1.0.xsd">

  <servers>
```

```
    <server>
      <id>direct-tomcat</id>
      <username>direct</username>
      <password>password</password>
    </server>
  </servers>
</settings>
```

The key part is the <server> tag and its subtag. We need it for tomcat deployment.

Note: Make sure the user exists in TOMCAT_HOME/conf/tomcat-users.xml and has the necessary roles(eg. manager-script)

8. Install tech.core.api

The tech.core.api framework needs to be installed in your maven repository.

- clone from github https://github.com/cloudspokes/tech-core
- run “mvn install” on each module

Note: you must install the tech.core.deps.* first as it is referenced by the api and sample.basic modules.

9. Deployment Instructions

Note: deploy/undeploy goals require tomcat to be started first.

1. Execute following commands
   ```
   cd <submission>/direct-api
   mvn clean
   mvn install
   mvn tomcat7:undeploy
   mvn tomcat7:deploy-only
   ```
2. Run `mvn test` to run the integration test. In this step you don’t need to setup application server.

10. Deploy tc api

Since java-jwt is not complete and it can’t provide enough feature for us, we need tc api to help us do something instead.
Following these steps to setup tc api

1. Checkout the tc api from github repository to some place.
   ```
   git checkout github.com/cloudspokes/tc-api <tc-api>
   ```
2. Install Node.JS and npm. Please refer the official website about how to set these things up.
   Node.JS: http://nodejs.org/
   NPM: http://npmjs.org/
3. Install the dependency for tc api application
   ```
   cd <tc-api>
   npm install
   ```

11. Starting

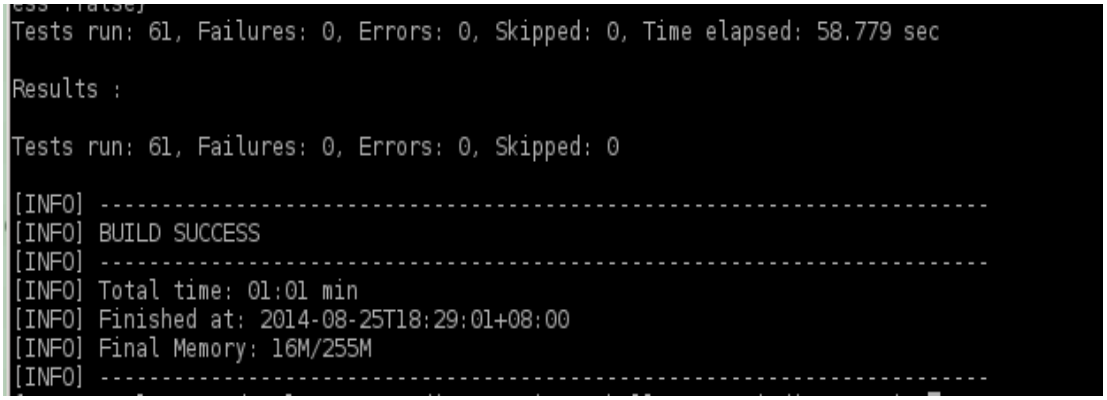
Note: deploy/undeploy goals require tomcat to be started first.

Run these commands under <submission>/direct-api

```
mvn clean
mvn install
mvn tcomat7:undeploy
mvn tomcat7:deploy-only
```

12. Verification

1. Verify by integration test
   Run the `mvn test` under <submission>/direct-api
   After the test you should see this:



2. Verify by calling from http tool

ENSURE you have inserted the test data, see “Database Setup” for details

2.1 Since we need jwt token so the api can authenticate who is calling, we need to generate jwt token first.
The jwt token can’t be generated by using java-jwt library, so we have to use tc api to do this job.
Execute following command
`cd <tc-api>`
`node test/helpers/manualJwt.js “ad|132456”`
This will get jwt token for user “heffan”

```
$ node test/helpers/manualJwt.js "ad|132456"
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZHwxMzI0NTYiLCJleHAiOjE0MDYwMTY3MTksImF1ZCI6IkNNYUJ1d1NuWTBWdTY4UExyV2F0dnZ1M2lJaUdQaDd0IiwiaWF0IjoxNDA1OTU2NzE5fQ.5zatOuGRLhXWi53Jf6TlrsfXMjZe8MqTVbCm9kpnpfg
```

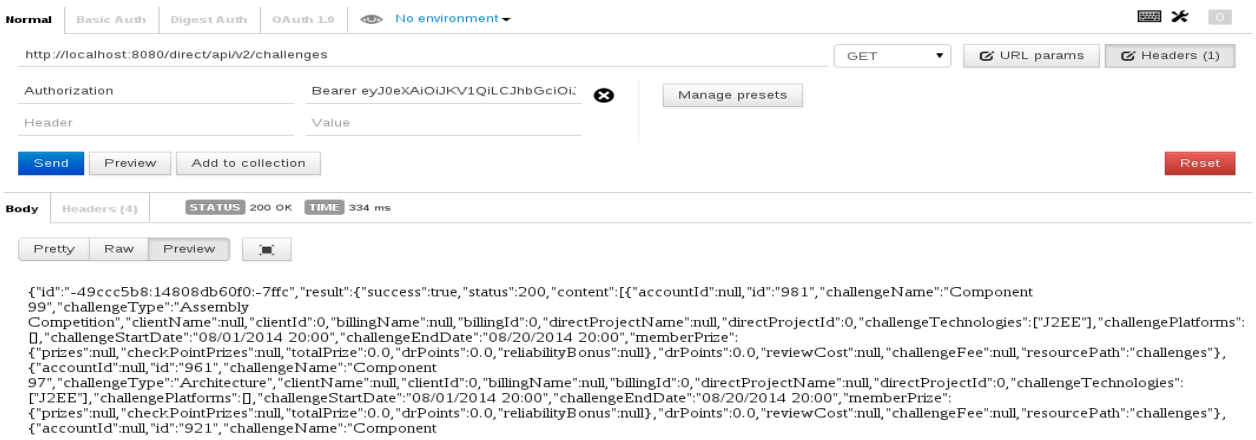
`node test/helpers/manualJwt.js “ad|132457”`
This will get jwt token for user “super”

```
$ node test/helpers/manualJwt.js "ad|132457"
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZHwxMzI0NTciLCJleHAiOjE0MDYwMTY5NDgsImF1ZCI6IkNNYUJ1d1NuWTBWdTY4UExyV2F0dnZ1M2lJaUdQaDd0IiwiaWF0IjoxNDA1OTU2OTQ4fQ.H9hFKJ7Zbi8wjWR4hI491W8CI34AgwjdQgj4RcQiIy4
```

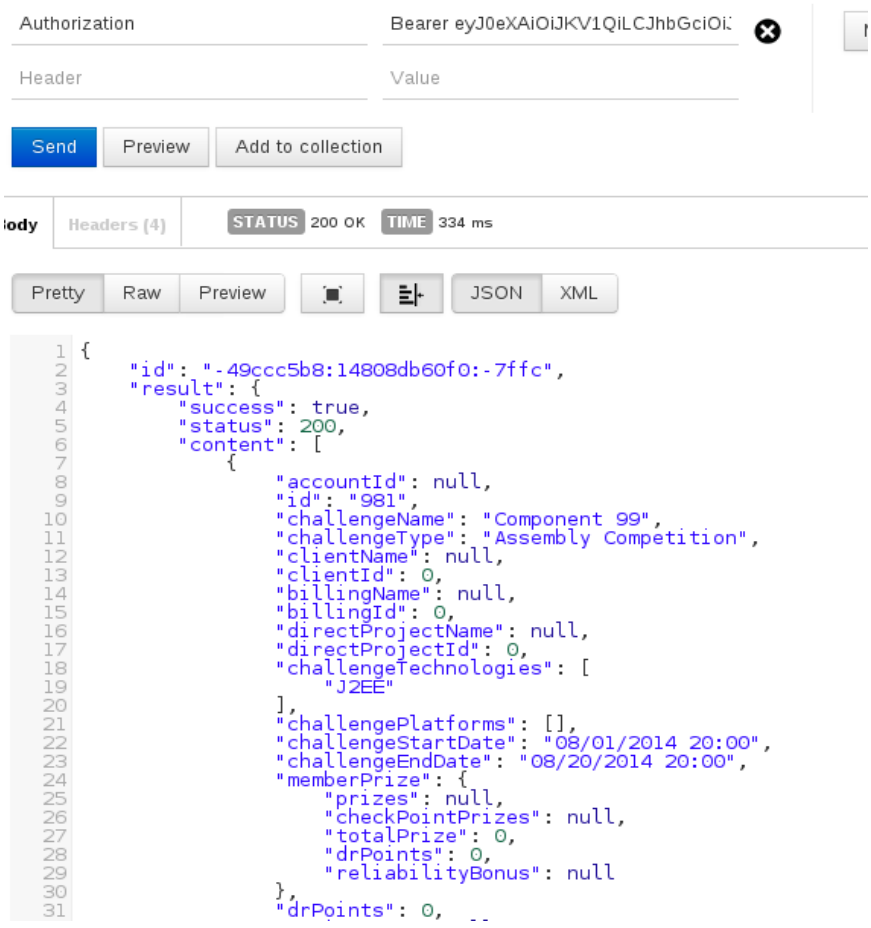
`node test/helpers/manualJwt.js “ad|20”`
This will get jwt token for user “dok_tester”

Note: tests will fail if there are expired tokens for any of the three above.

2.2 Prepare a http tool for calling api.
I recommend this one:
https://chrome.google.com/webstore/detail/fdmmgilgnpjigdojojpjoooidkmcomcm
2.3 Call api like this with a http GET method:



Be aware that we need to set Authorization header with value “Bearer ” + jwt token.
You should get results like this:



For the other conditions, please see MyChallengesTest.java.

13. Resource Contact List

Name	Resource Email