Application Process

Chief Architect - Java - $60/hr

STEP 1

Email Verification

STEP 2

Technical Screen

STEP 3

Profile Completion

STEP 4

Project Evaluation

STEP 5

Phone Screen

STEP 4

Project Evaluation

Submission ends on Saturday November 28 2015 at 12:53 PM

Need an extension?

Time left:

2 days	23 hours	58 min	39 sec

Only file with type **'.zip, application/zip'** is acceptable. Make sure the size of your delivery file does not exceed **30 MB**

SELECT FILE UPLOAD

Instructions

Your challenge has started.

Please read through your assignment very carefully before beginning your solution. Try to complete as much of the solution as possible in the given timeframe. To upload your solution, click the "SELECT FILE" button above. After selecting your solution, click "UPLOAD". You will be able to review your submission after uploading it.

If you have any questions, submit a Support Request (mailto:tech-trial@crossover.com). However, your questions should pertain to the test-taking procedure rather than the content of the assignment. If you had any problems and need an extension, please feel free to Request an Extension. We will determine the extension period based on the reason of the request.

Assignment

CROSS|OVER Tech Trial

Congratulations on making it to this stage of the Java Chief Software Architect evaluation!

You have to be very talented to make it this far, you've already proven yourself better than most! With the insane amount of applications most companies receive, it's almost impossible to find the great candidates (that's you!) What we provide is a way for you to show your skills, and differentiate yourself from all the other applicants. By doing well on this "real world" assignment, you'll go from being just another resume in a pile of 1000, to being one of the 5 candidates we are showing to companies as the "best of the best". This is the biggest, and most important part of our process. After you complete the assignment we will setup a quick interview for you to be able to discuss what you've built, (and let you brag a little). That's it, then you're done!

We've tried to scope the project to be an appropriate size so you can complete in a reasonable amount of time, while still being able to demonstrate your enterprise-class development skills. Though this is a fictitious example, this scenario is very similar to what you may encounter in your Chief architect job.

Important note: there are two assignments in this trial. Make sure you submit each assignment deliverables into a separate folder and all should be zipped into a single file.

Assignment 1

Overall Objectives

Chief architects are responsible for setting best practices for project teams and in many cases this means leading by example. For this assignment, you will design a new system for loading properties for a code base that currently loads dozens of properties from the classpath, System.getProperties and remote URLs that return JSON.

Your new system will serve as the foundation for all new system configuration moving forward. After brainstorming with the team, you've collected a set of characteristics that everyone desires in the new property management mechanism. You have also pulled together a set of sample property files from the existing system. You'll now create a prototype implementation which you will distribute to the team for additional feedback.

Technical Characteristics:

1. Implement your code within the following project skeleton, please maintain the original structure of the project and don't make harmful changes in the package hierarchy and the pom file, please download it from here (http://techtrial.s3.amazonaws.com/System/DevF/properties-dist.zip)
2. Assume the solution will need to scale to about 100 properties
3. Once all URIs are loaded, all properties should be set. Any unset properties are considered an error
4. Type safety is important and having a type safe accessor to properties is a requirement
5. The easier and more centralized the addition / removal of properties the better
6. Initialization should support loading and combining properties from multiple URIs
7. Property files will specified by a uri such as file://, http:// or the custom classpath:resources/
8. The URI suffix will determine the file format - .properties and .json will be supported initially
9. If a property is defined more than once, the last source to set that property value "wins" and is considered the final property value

Example Property Files

Your system should support all the properties defined in the example properties files below and should consider all properties required. You should use your professional experience to determine a type.

classpath:resources/jdbc.properties

```
JDBC_DRIVER=com.mysql.jdbc.Driver
JDBC_URL=jdbc:mysql://localhost/test
JDBC_USERNAME=username123
JDBC_PASSWORD=password123
hibernate.generate_statistics=true
hibernate.show_sql=true
jpa.showSql=true
```

file:///tmp/aws.properties

```
aws_access_key=AKIAJSF6XRIJNJTTTL3Q
aws_secret_key=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
aws_account_id=12345678
aws_region_id=us-east-1
```

http://localhost/global/config.json

```
{
   "auth.endpoint.uri": "https://authserver/v1/auth",
   "job.timeout": 3600,
   "job.maxretry": 4,
   "sns.broadcast.topic_name": "broadcast",
   "sns.broadcast.visibility_timeout": 30,
   "score.factor": 2.4,
   "jpa.showSql": false
}
```

Solution Requirements

You may create as many interfaces and classes as you wish, but there are a few requirements on solution structure to assist us in our evaluation.

1. Your submission must include a pom.xml at the top level. You should assume we will build, test and run using the latest released version of maven 3, JDK 1.8 and JUnit 4.x on the linux platform
2. You must define a main method in com.crossover.trial.properties.Main
3. Your main should accept as arguments the URIs containing the properties to load
4. Your system should support any combination of file: uri: and classpath: URIs
5. You should output to stdout the names of all properties in case insensitive alphabetical order, in the following format:

```
propertyName, propertyType, propertyValue
```

Here is partial output for the properties files provided:

```
aws_access_key, java.lang.String, AKIAJSF6XRIJNJTTTL3Q
aws_account_id, java.lang.Integer, 12345678
aws_region_id, com.amazonaws.regions.Regions, us-east-1
aws_secret_key, java.lang.String, pmqnweEYvdiw7cvCdTOES48sOUvK1rGvvctBsgsa
jpa.showSql, java.lang.Boolean, false
```

Scoring

1. Completeness (Auto-grader)
    1. Your submission will first be auto-graded for compliance and then checked manually.
    2. The auto-grader grades your submission using mvn 3.3.3 or newer and the latest available JDK1.8.
    3. The commands executed are provided as reference to understand how your code is executed / tested.
    (note: you are not given access to the grader)

```
tar zcf firstname.lastname.properties.tgz -C trial

pushd trial/code; mvn clean package exec:java -Dexec.mainClass="com.crossover.trial.properties.Main" -Dexec.args="classpath:resou
    file:///tmp/system.properties
    http://localhost:8080/global.properties" > output.txt
```

d. output.txt will be fed to the auto grader, which will product a score between 0 and 10

2. Code quality (Human Grader) - CrossOver screens candidates that work well in high caliber teams. This means creating code that is to read by the dozens of people that will likely use it is just as important as the code's functionality. 50% of your score on the trial is derived from the readability and long term maintainability of your code. A completely correct solution that is very difficult to read and maintained will be failed. Here are the criteria we use to judge the quality and workmanship of your code.
    1. Code organization, encapsulation and visibility are all important - you'll be penalized for code that is poorly designed
    2. Following industry conventions on styling. Both Google and JDK Java Style are appropriate
    3. Code documentation - balance the level of comments with the readability of your code, neither too many comments nor too few comments should be used.
    4. Testing - Unit testing and coverage of both success and failure paths
    5. Exception handling - logging errors and handling exceptions as needed.

Delivery / What to submit

Please, read and follow this section carefully. Any delivery that does not follow this section will be scored lower or won't be evaluated.

You will be submitting your deliverable through the Crossover Candidate Portal. The delivery for this assignment should consist of a zip file named CA_Java_Prop-<your_name>.zip containing the following:

./readme.txt - A *simple* readme explaining assumptions you made or feedback on the assignment

./pom.xml - folder holds your top level pom.xml and your main and test directories

./main/java/* and ./test/java/* - the conventional folder structure for maven projects

Thank you and we look forward to seeing your work in this 'real scenario' work assignment!

Assignment 2

Overview

Chief architects are responsible for setting best practices for project teams and in many cases this means leading by example. For this assignment, you'll be taking a project that was originally created as an intern project, and improving it to professional quality.

The Airport Weather App (AWA) (http://techtrial.s3.amazonaws.com/System/DevF/weather-dist.zip) is a fictional REST service that collects and distributes airport weather information. The system holds all data in memory both for simplicity and performance. This application has a series of rest endpoints that provide data to dependent systems.

Unfortunately, AWA suffers from a number of issues. The code was originally written by an intern that is long gone. The intern had limited professional development experience and this shows through in the code. There are issues with performance, threading and encapsulation. The AWA concept is a good one and the v1 code has served it's purpose, but now the code needs to be improved to professional quality.

v1 Service Details:

The v1 of the AWA app has a set of REST endpoints:

- GET /collect/ping
- POST /collect/weather/{iata}/{pointType}
- POST /collect/airport/{iata}/{lat}/{long} (not implemented)
- DELETE /collect/airport/{iata} (not implemented)
- GET /query/ping
- GET /query/weather/{iata}/{radius}

Objectives:

Your goal, as an experienced software developer, is to apply java development best practices and convention to this code base (https://techtrial.s3.amazonaws.com/System/DevF/weather-dist.zip) (https://drive.google.com/open?id=0B6k_oHEMrW8PRkpFNWVBOEktNEE). You should modify the code to follow the patterns in the java world, restructure the code so that it's easier to work with. You should break this up into three parts.

1. Code Review - you should treat the code as if you were asked to conduct a code review. You should add your own comments to the prefixed with // CR: on *each* line of the your comments. As a point of feedback, be careful to not write *too much* - it's a code review, not a blog post. After you've entered your commends, create firstname.lastname.awa.part1.zip (20% weight)
2. Next, you'll apply your own feedback and modify the code. The code is now *your* code - change it to meet professional coding standards. Once complete, create firstname.lastname.awa.part2.zip. (50% weight)
3. Now that the code is cleaned up, it's time to add a feature. The feature to implement, which you may have noticed in your code review, is that airports are hard coded. You should implement the [POST|DELETE] /collect/airport endpoints. You should also create an AirportLoader which will read airports.txt and call your endpoint. Once complete, create firstname.lastname.awa.part3.zip. (30% weight.) A sample airports.txt (http://techtrial.s3.amazonaws.com/System/DevF/airports.txt) (https://techtrial.s3.amazonaws.com/System/DevF/airports.txt) (https://drive.google.com/open?id=0B6k_oHEMrW8PbFJmTTA3RkxqT3M) is included with the trial, but a more complete airport.txt will be used to evaluate your work. The airports.txt (http://techtrial.s3.amazonaws.com/System/DevF/airports.txt) (https://techtrial.s3.amazonaws.com/System/DevF/airports.txt) (https://drive.google.com/open?id=0B6k_oHEMrW8PbFJmTTA3RkxqT3M) is a comma separated file with the following headers:

City	Main city served by airport. May be spelled differently from name.
Country	Country or territory where airport is located. IATA/FAA 3-letter FAA code or IATA code (blank or "" if not assigned)
ICAO	4-letter ICAO code (blank or "" if not assigned) Latitude Decimal degrees, up to 6 significant digits. Negative is South, positive is North.
Longitude	Decimal degrees, up to 6 significant digits. Negative is West, positive is East.
Altitude	In feet
Timezone	Hours offset from UTC. Fractional hours are expressed as decimals. (e.g. India is 5.5)
DST	One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U (Unknown)

Solution Requirements

Your assignment is part computer graded and part human graded. If you fail to add // CR: to your comments, change the structure of the zip file or your code does not compile with maven the computer grader may not be able to generate a score. Please check your submissions for correctness.

Create a zip of firstname.lastname.awa.part?.zip and call it CA_Java_AWA-<your_name>.zip. This is your submission artifact. Upon extraction, your submission should create:

- firstname.lastname.awa.part1.zip
- firstname.lastname.awa.part2.zip
- firstname.lastname.awa.part3.zip

Each part should extract to a directory (the name will be ignored) containing a pom.xml. The auto grader will run mvn package to build your package. It will then apply its own unit tests and run REST API tests against WeatherServer.

Scoring

1. Completeness (Auto-grader)
    1. Your submission will first be auto-graded for compliance and then checked manually.
    2. If you fail to add '// CR:' to your comments, change the structure of the zip file or your code does not compile with maven, the computer grader may not be able to generate a score.
    3. The auto-grader grades your submission using mvn 3.3.3 or newer and the latest available JDK1.8.
2. Code quality (Human Grader) - CrossOver screens candidates that work well in high caliber teams. This means creating code that is to read by the dozens of people that will likely use it is just as important as the code's functionality. A significant portion of your score on the trial is derived from the readability and long term maintainability of your code. A completely correct solution that doesn't follow conventional best practices for java development will be failed. Here are the criteria we use to judge the quality and workmanship of your code.
    1. Code organization, encapsulation and visibility are all important - you'll be penalized for code that is poorly designed
    2. Following industry conventions on styling. Both Google and JDK style guides are appropriate
    3. Code documentation - balance the level of comments with the readability of your code, neither too many comments nor too few comments should be used.
    4. Testing - Unit testing and coverage of both success and failure paths
    5. Exception handling - logging errors and handling exceptions as needed.

Delivery / What to submit

Please, read and follow this section carefully. Any delivery that does not follow this section will be scored lower or won't be evaluated.

You will be submitting your deliverable through the Crossover Candidate Portal. The delivery for this assignment should consist of a zip file named CA_Java_AWA-<your_name>.zip containing the following:

./firstname.lastname.awa.part1.zip - The code read portion of your assignment

./firstname.lastname.awa.part2.zip - The refactoring portion of your assignment

./firstname.lastname.awa.part3.zip - The enhancement / new code portion of your assignment.

Each of these zip archives must contain their own pom.xml and maven structure to build and test the code.

Thank you and we look forward to seeing your work in this 'real scenario' work assignment!

How do we evaluate our candidates?

We evaluate your delivery against the following three criteria:

COMPLETENESS:

The amount of the given task that you completed.

DESIGN QUALITY:

The quality of your architecture or design for the given task.

CODE QUALITY:

The quality of the implementation of your design.

What are the possible outcomes of the evaluation?

INVITED BUT NOT STARTED:

If you are invited but do not start your assignment within the given invitation period, don't worry - you can apply again whenever you have the time.

INVITED, STARTED, AND NOT DELIVERED:

If you start your assignment but do not deliver it, you will not be eligible for a second chance until six months after your initial evaluation.

INVITED, STARTED, DELIVERED, AND NOT SELECTED:

If you started and delivered your assignment solution but are not selected to move on, you may have a second chance! We will assess your score as well as the effort that you put into the assignment. If you qualify for a second chance, we will let you know by email and re-invite you to the project evaluation after one month.