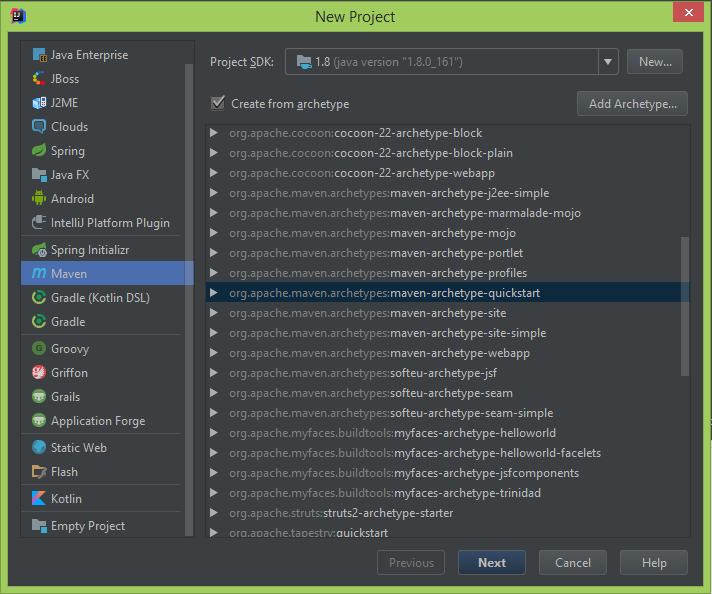
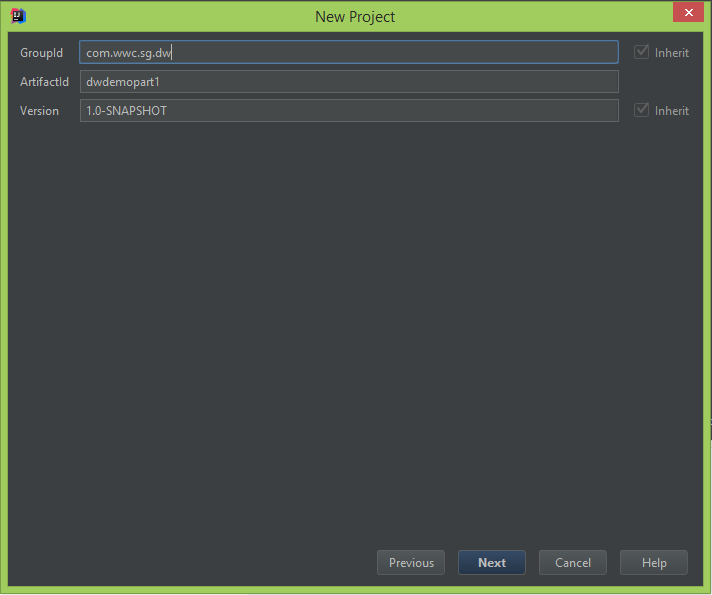
Dropwizard Workshop Instructions

Step 1: Create new Maven project

Open IDE and then click Create New Project. Choose the maven-archetype-quickstart, then enter ***Group Id*** and ***Artifact Id***, and finally press ***Finish***.





## 

## Step 2: Update the pom.xml

Now update the pom.xml file to include the dropwizard-core maven dependency. We will also update the Maven project to use Java version 1.8. After updating the pom.xml file, update the Maven Project(Right Click on pom.xml > Maven > Reimport)

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.wwc.sg.dw</groupId>  
 <artifactId>dw-demo-part1</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <dropwizard.version>1.2.2</dropwizard.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>io.dropwizard</groupId>  
 <artifactId>dropwizard-core</artifactId>  
 <version>${dropwizard.version}</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

## Step 3: Create Configuration class

A dropwizard application has one configuration class which specifies the environment specific parameters. This maps to the config yml file.   
This class extends the **io.dropwizard.Configuration** class.

package com.wwc.sg.dw;  
  
import io.dropwizard.Configuration;  
  
public class DWDemoAppConfiguration extends Configuration {  
  
 private String application;  
 private String environment;  
 private String version;  
 private String template;  
  
 public String getApplication() {  
 return application;  
 }  
  
 public void setApplication(String application) {  
 this.application = application;  
 }  
  
 public String getEnvironment() {  
 return environment;  
 }  
  
 public void setEnvironment(String environment) {  
 this.environment = environment;  
 }  
  
 public String getVersion() {  
 return version;  
 }  
  
 public void setVersion(String version) {  
 this.version = version;  
 }  
  
 public String getTemplate() {  
 return template;  
 }  
  
 public void setTemplate(String template) {  
 this.template = template;  
 }  
  
}

## Step 4: Create Application class

This is the main entry point for any dropwizard application. This class extends the ***io.dropwizard.Application*** class and implement the initialize(Bootstrap<Configuration>) and run(Configuration, Environment) methods. They prepare the runtime environment of the application.

package com.wwc.sg.dw;  
  
import com.wwc.sg.dw.resources.IndexResource;  
import io.dropwizard.Application;  
import io.dropwizard.configuration.ResourceConfigurationSourceProvider;  
import io.dropwizard.setup.Bootstrap;  
import io.dropwizard.setup.Environment;  
  
public class DWDemoApp extends Application<DWDemoAppConfiguration> {  
  
 public static void main(String[] args) throws Exception {  
 new DWDemoApp().run(new String[] { "server", "DWDemoApp.yml"});  
 }  
  
 @Override  
 public void initialize(Bootstrap<DWDemoAppConfiguration> bootstrap) {  
 bootstrap.setConfigurationSourceProvider(  
 new ResourceConfigurationSourceProvider());  
 }  
  
 @Override  
 public void run(DWDemoAppConfiguration configuration, Environment environment) throws Exception {  
 final IndexResource resource = new IndexResource(  
 configuration.getApplication(),  
 configuration.getEnvironment(),  
 configuration.getVersion(),  
 configuration.getTemplate()  
 );  
 environment.jersey().register(resource);  
 }  
}

## Step 5: Create Resource class

Resource classes exposes our rest endpoints.

package com.wwc.sg.dw.resources;  
  
import javax.ws.rs.GET;  
import javax.ws.rs.Path;  
import javax.ws.rs.PathParam;  
import javax.ws.rs.Produces;  
import javax.ws.rs.core.MediaType;  
import javax.ws.rs.core.Response;  
  
@Path("/")  
@Produces(MediaType.APPLICATION\_JSON)  
public class IndexResource {  
  
 private String name;  
 private String environment;  
 private String version;  
 private String template;  
  
 public IndexResource(String name, String environment, String version, String template){  
 this.name = name;  
 this.environment = environment;  
 this.version = version;  
 this.template = template;  
 }  
  
 @GET  
 public Response getVersion() {  
 StringBuffer sb = new StringBuffer();  
 sb.append("Application:" + name + "\n");  
 sb.append("Environment:" + environment + "\n");  
 sb.append("Version:" + version + "\n");  
 return Response.ok(sb.toString()).build();  
 }  
  
 @GET  
 @Path("hello/{name}")  
 public Response printGreeting(@PathParam("name") String name) {  
 final String greeting = String.format(template, name);  
 return Response.ok(greeting).build();  
 }  
}