TerraMeta Software, Inc.

Plasma Quick Start (POJO)

1 Introduction

This step-by-step guide shows how to build a Maven project which generates a simple MySql data model with 2 tables, inserts data and reads data using only Java objects (POJO's) as the source of metadata. It requires basic knowledge of the Java programing language and Apache Maven and assumes the following prerequisites.

- Java JDK 1.7 or Above
- Maven 3.x or Above

2 Plasma Quick Start (POJO)

2.1 Add Build Time Plasma Dependencies

Add the following dependencies to your Maven project, including plasma, an RDBMS provider, MySql client and a connection pooling library, DBCP.

```
<dependency>
      <groupId>org.terrameta</groupId>
      <artifactId>plasma-core</artifactId>
      <version>2.0.0</version>
</dependency>
```

2.2 Create POJOs

Create 3 java enumeration classes annotated with Plasma annotations. Enumerations rather than Java classes are annotated to facilitate reuse across multiple code generation and metadata integration contexts. The annotations capture structural metadata elements, as below.

- Entity Type and Property Names and Aliases. See Type, Alias.
- Data Types. See DataProperty, ReferenceProperty.
- Cardinalities, Nullability and Visibility. See DataProperty, ReferenceProperty.
- Constraints. See ValueConstraint, EnumerationConstraint.
- Inheritance Relationships (multiple inheritance is supported). See Type.
- Associations Between Entities. See ReferenceProperty.

The below example enumerations which create a classic data model "Person-Organization" with a common superclass entity "Party". It is intended to illustrate several of the available Plasma annotations, but the model may be easily simplified, for example use just a single entity.

```
@Type(name = "Party", isAbstract = true)
public enum Party {
  @Alias(physicalName = "CRTD_DT")
  @DataProperty(dataType = DataType.Date, isNullable = false)
  createdDate
}

@Alias(physicalName = "PERSON")
@Type(superTypes = { Party.class })
public enum Person {
  @Key(type = KeyType.primary)
  @ValueConstraint(maxLength = "36")
  @Alias(physicalName = "FN")
  @DataProperty(dataType = DataType.String, isNullable = false)
  firstName,
```

```
  @Key(type = KeyType.primary)
  @ValueConstraint(maxLength = "36")
  @Alias(physicalName = "LN")
  @DataProperty(dataType = DataType.String, isNullable = false)
  lastName,
  @ValueConstraint(totalDigits = "3")
  @Alias(physicalName = "AGE")
  @DataProperty(dataType = DataType.Int)
  age,
  @Alias(physicalName = "DOB")
  @DataProperty(dataType = DataType.Date)
  dateOfBirth,
  @Alias(physicalName = "EMP")
  @ReferenceProperty(targetClass = Organization.class, targetProperty =
"employee")
  employer;
}

@Alias(physicalName = "ORG")
@Type(superTypes = { Party.class })
public enum Organization {
  @Key(type = KeyType.primary)
  @ValueConstraint(maxLength = "36")
  @Alias(physicalName = "NAME")
  @DataProperty(dataType = DataType.String, isNullable = false)
  name,
  @EnumConstraint(targetEnum = OrgCat.class)
  @Alias(physicalName = "ORG_CAT")
  @DataProperty(dataType = DataType.String, isNullable = false)
  category,
  @Alias(physicalName = "PARENT")
  @ReferenceProperty(isNullable = true, isMany = false, targetClass =
Organization.class, targetProperty = "child")
  parent,
  @Alias(physicalName = "CHILD")
  @ReferenceProperty(isNullable = true, isMany = true, targetClass =
Organization.class, targetProperty = "parent")
  child,
  @Alias(physicalName = "EMPLOYEE")
  @ReferenceProperty(isNullable = true, isMany = true, targetClass =
Person.class, targetProperty = "employer")
  employee;
}
```

2.3 Create Namespace

In the same package as the above POJOs, create a file called package_info.java with the below annotated package.

```
@Alias(physicalName = "HR")
@Namespace(uri = "http://plasma-quickstart-pojo/humanresources")
@NamespaceProvisioning(rootPackageName = "quickstart.pojo.model")
@NamespaceService(storeType = DataStoreType.RDBMS,
providerName = DataAccessProviderName.JDBC,
properties = {
      "org.plasma.sdo.access.provider.jdbc.ConnectionURL=jdbc:mysql://localhost:
3306/hr?autoReconnect=true",
      "org.plasma.sdo.access.provider.jdbc.ConnectionUserName=root",
```

```
      "org.plasma.sdo.access.provider.jdbc.ConnectionPassword=yourpassword",
      "org.plasma.sdo.access.provider.jdbc.ConnectionDriverName=com.mysql.jdbc.D
river",
      "org.plasma.sdo.access.provider.jdbc.ConnectionProviderName=examples.quick
start.connect.DBCPConnectionPoolProvider",
      "org.plasma.sdo.access.provider.jdbc.ConnectionPoolMinSize=1",
      "org.plasma.sdo.access.provider.jdbc.ConnectionPoolMaxSize=10",
      "org.apache.commons.dbcp.validationQuery=SELECT COUNT(*) FROM person",
      "org.apache.commons.dbcp.testOnBorrow=false",
      "org.apache.commons.dbcp.testOnReturn=false",
      "org.apache.commons.dbcp.maxWait=30000",
      "org.apache.commons.dbcp.testWhileIdle=false",
      "org.apache.commons.dbcp.timeBetweenEvictionRunsMillis=30000",
      "org.apache.commons.dbcp.minEvictableIdleTimeMillis=40000"
})
package examples.quickstart.pojo;
import org.plasma.config.annotation.NamespaceService;
import org.plasma.config.annotation.NamespaceProvisioning;
import org.plasma.sdo.annotation.Namespace;
import org.plasma.sdo.annotation.Alias;
import org.plasma.config.DataAccessProviderName;
import org.plasma.config.DataStoreType;
```

2.4 Add Plasma Maven Plugin

Add the Plasma Maven Plugin with 3 executions which generate data access and query (DSL) classes as well as a schema for MySql. See below Plasma Maven Plugin Configuration for complete listing. After adding the plugin and executions type:

```
maven generate-sources
```

The source code and MySql DDL should appear under target/generated-sources.

2.5 Add Run Time Dependencies

Add the following dependencies to your Maven project, including an RDBMS service provider Cloudgraph RDB, MySql client and a connection pooling library, DBCP.

```
<dependency>
    <groupId>org.cloudgraph</groupId>
    <artifactId>cloudgraph-rdb</artifactId>
    <version>1.0.7</version>
</dependency>
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.23</version>
</dependency>
<dependency>
    <groupId>commons-dbcp</groupId>
    <artifactId>commons-dbcp</artifactId>
    <version>1.4</version>
</dependency>
```

3 Plasma Maven Plugin Configuration

```
<plugin>
	<groupId>org.terrameta</groupId>
	<artifactId>plasma-maven-plugin</artifactId>
	<version>2.0.0</version>
	<dependencies>
		<dependency>
			<groupId>org.terrameta</groupId>
			<artifactId>plasma-core</artifactId>
			<version>2.0.0</version>
		</dependency>
		<dependency>
			<groupId>org.cloudgraph</groupId>
			<artifactId>cloudgraph-rdb</artifactId>
			<version>1.0.7</version>
		</dependency>
	</dependencies>
	<executions>
		<execution>
			<id>sdo-create</id>
			<configuration>
				<action>create</action>
				<dialect>java</dialect>
				<additionalClasspathElements>
					<param>${basedir}/target/classes</param>
				</additionalClasspathElements>
				<outputDirectory>${basedir}/target/generated-
sources/java</outputDirectory>
			</configuration>
			<goals>
				<goal>sdo</goal>
			</goals>
		</execution>
		<execution>
			<id>dsl-create</id>
			<configuration>
				<action>create</action>
				<dialect>java</dialect>
				<additionalClasspathElements>
					<param>${basedir}/target/classes</param>
				</additionalClasspathElements>
				<outputDirectory>${basedir}/target/generated-
sources/java</outputDirectory>
			</configuration>
			<goals>
				<goal>dsl</goal>
			</goals>
		</execution>
		<execution>
			<id>ddl-create-mysql</id>
			<configuration>
				<action>create</action>
				<dialect>mysql</dialect>
				<additionalClasspathElements>
					<param>${basedir}/target/classes</param>
				</additionalClasspathElements>
				<outputDirectory>${basedir}/target/ddl</outputDirectory>
				<outputFile>mysql-create.sql</outputFile>
```

```
                </configuration>
                <goals>
                      <goal>rdb</goal>
                </goals>
          </execution>
     </executions>
</plugin>
```
.