In this Chapter, We will discuss how to develop a Spring Framework project to work with Neo4j Database.

**Spring DATA Neo4j Module Annotations**

We are going to use the following Spring Framework Annotations to develop this application.

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Spring DATA Neo4j Annotation</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>@GraphEntity</td>
<td>To define domain class a Neo4j Entity</td>
</tr>
<tr>
<td>2.</td>
<td>@GraphID</td>
<td>To define Node or Relationship id</td>
</tr>
<tr>
<td>3.</td>
<td>@GraphProperty</td>
<td>To define Node or Relationship properties</td>
</tr>
</tbody>
</table>

Before developing application, please refer "Neo4j Spring DATA Environment Setup" chapter to setup Maven Eclipse IDE project.

Brief Steps to implement Spring DATA Neo4j Application -

- Develop Graph Entity or Domain or POJO Classes
- Develop DAO or Repositories
- Develop Service Layer `@required`
- Spring DATA Neo4j XML Configurations

We now demonstrate how to develop a Maven based Spring DATA application in Eclipse IDE to perform Neo4j DB Operations.

**Develop Graph Entity or Domain or POJO Classes**

We should implement equals and hashCode methods.

It is not required provide setter method for "id" property because Neo4j will take care of assigning this property.

```java
package com.tp.springdata.neo4j.domain;

import org.springframework.data.neo4j.annotation.GraphId;
import org.springframework.data.neo4j.annotation.NodeEntity;

@NodeEntity
public class GoogleProfile {

    @GraphId Long id;
    private String name;
    private String address;
    private String sex;
    private String dob;

    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }
}
public void setName(String name) {
    this.name = name;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getSex() {
    return sex;
}

public void setSex(String sex) {
    this.sex = sex;
}

public String getDob() {
    return dob;
}

public void setDob(String dob) {
    this.dob = dob;
}

public boolean equals(Object other) {
    if (this == other)
        return true;
    if (id == null)
        return false;
    if (! (other instanceof GoogleProfile))
        return false;

    return id.equals(((GoogleProfile) other).id);
}

public int hashCode() {
    return id == null ? System.identityHashCode(this) : id.hashCode();
}

public String toString() {
    return "Profile[id:" + id + ",name:" + name + ",sex:" + sex + ",address:" + address + ",dob:" + dob + "]";
}
}

@GraphProperty is optional so we can omit this. Above Entity is same as below.

package com.tp.springdata.neo4j.domain;

import org.springframework.data.neo4j.annotation.GraphId;
import org.springframework.data.neo4j.annotation.NodeEntity;

@NodeEntity
public class GoogleProfile {
    @GraphId
    private Long id;

    private String name;
    private String address;
    private String sex;
    private String dob;
}
// Getter for id
// Setters and Getters for rest of properties
// implement equals() and hashCode() methods

Develop Spring DATA Neo4j Repository.

As we discussed earlier, we need to develop only interface by extending Spring DATA Neo4j API interface "GraphRepository" and not required its implementation.

Spring DATA Neo4j will take care of providing implementation for this interface internally.

Define a Repository or DAO interface for our Domain class : GoogleProfile

```java
package com.tp.springdata.neo4j.dao;

import org.springframework.data.neo4j.repository.GraphRepository;

public interface GoogleProfileRepository extends GraphRepository<GoogleProfile> {
}
```

It is very easy and simple to implement Spring DAT Neo4j repository. We just need to define an interface by extending GraphRepository by specifying our domain class as a parameter.

Develop Service layer Artifacts: Interface and Implementation.

It is good practice to provide a Service layer in our application on top of DAO layer.

Service Artifacts

Service Component Interface:

```java
package com.tp.springdata.neo4j.service;

import org.springframework.data.neo4j.conversion.Result;
import com.tp.springdata.neo4j.domain.GoogleProfile;

public interface GoogleProfileService {
    GoogleProfile create(GoogleProfile profile);
    void delete(GoogleProfile profile);
    GoogleProfile findById(long id);
    Result<GoogleProfile> findAll();
}
```

Service Component Implementation:

```java
package com.tp.springdata.neo4j.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.neo4j.conversion.Result;
import org.springframework.stereotype.Service;

import com.tp.springdata.neo4j.dao.GoogleProfileRepository;
import com.tp.springdata.neo4j.domain.GoogleProfile;

@Service("googleProfileService")
public class GoogleProfileServiceImpl implements GoogleProfileService {
    @Autowired
    private GoogleProfileRepository googleProfileRepository;

    public GoogleProfile create(GoogleProfile profile) {
        return googleProfileRepository.save(profile);
    }
}
```
public void delete(GoogleProfile profile) {
    googleProfileRepository.delete(profile);
}

public GoogleProfile findById(long id) {
    return googleProfileRepository.findOne(id);
}

public Result<GoogleProfile> findAll() {
    return googleProfileRepository.findAll();
}

Spring DATA Neo4j XML Configurations

To run Spring based applications, we need to provide some XML configurations.

We need to provide the following details in Spring XML configuration file

Provide Spring Data Neo4j namespace

xmlns:neo4j=http://www.springframework.org/schema/data/neo4j

Provide Spring Data Neo4j Schema definition XSD file

xsi:schemaLocation="http://www.springframework.org/schema/data/neo4j
http://www.springframework.org/schema/data/neo4j/spring-neo4j.xsd"

spring-neo4j.xsd file contains all Spring Data Neo4j related XML tags

Provide our Neo4j Database location and our Graph Entities Domain or POJO classes base package

<neo4j:config storeDirectory="C:\TPNeo4jDB" base-package="com.tp.springdata.neo4j.domain"/>

Here storeDirectory="C:\TPNeo4jDB" specifies our Neo4j Database files are stored at C:\TPNeo4jDB location in our file system.

base-package="com.tp.springdata.neo4j.domain"

All our Graph Entities have com.tp.springdata.neo4j.domain as base package in our application classpath

Provide our Spring Data Neo4j Repositories DAO Interfaces base package.

<neo4j:repositories base-package="com.tp.springdata.neo4j.dao"/>

All our Spring Data Neo4j Repositories are available at com.tp.springdata.neo4j.dao package in our application classpath

Provide Spring configuration to register Annotated components with Spring IOC Container.

<context:component-scan base-package="com.tp.springdata.neo4j.service"/>

All our components or services are available at "com.tp.springdata.neo4j.service" package in our application classpath

Complete "googleprofile.xml"

<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<beans xmlns="http://www.springframework.org/schema/beans"
      xmlns:context="http://www.springframework.org/schema/context"
Develop Test program and test all operations

```java
import java.util.Iterator;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import org.springframework.data.neo4j.conversion.Result;
import com.tp.springdata.neo4j.service.GoogleProfileService;
import com.tp.springdata.neo4j.domain.GoogleProfile;

public class GoogleProfileTest {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("googleprofile.xml");
        GoogleProfileService service = (GoogleProfileService) context.getBean("googleProfileService");

        // Please uncomment one of the operation section
        // and comment remaining section to test only one operation at a time
        // Here I've uncommented CREATE operation and
        // commented other operations: FIND ONE, FIND ALL, DELETE
        // CREATE Operation
        GoogleProfile profile = createProfile();
        createProfile(service, profile);
        System.out.println("GoogleProfile created successfully.");

        // FIND ONE
        /*
        GoogleProfile profile = getOneProfileById(service,67515L);
        System.out.println(profile);
        */

        // FIND ALL
        /*
        getAllProfiles(service);
        */

        // DELETE
        /*
        GoogleProfile profile = createProfile();
        deleteProfile(service, profile);
        System.out.println("GoogleProfile deleted successfully.");
        */
    }
}
```
private static GoogleProfile createProfile(GoogleProfileService service, GoogleProfile profile){
    return service.create(profile);
}

private static void deleteProfile(GoogleProfileService service, GoogleProfile profile){
    service.delete(profile);
}

private static GoogleProfile getOneProfileById(GoogleProfileService service, Long id){
    return service.findById(id);
}

private static void getAllProfiles(GoogleProfileService service){
    Result<GoogleProfile> result = service.findAll();
    Iterator<GoogleProfile> iterator = result.iterator();
    while(iterator.hasNext()){
        System.out.println(iterator.next());
    }
}

private static GoogleProfile createProfile(){
    GoogleProfile profile = new GoogleProfile();
    profile.setName("Profile-2");
    profile.setAddress("Hyderabad");
    profile.setSex("Male");
    profile.setDob("02/02/1980");
    return profile;
}

I have uncommented only for CREATE operation and commented rest of 3 operations.
If you want to test other operations, just comment unwanted section and uncomment wanted section and test this application

**Final Eclipse Maven project structure**

```
<table>
<thead>
<tr>
<th>spring-data-neo4j</th>
</tr>
</thead>
<tbody>
<tr>
<td>com.tp.springdata.neo4j.dao</td>
</tr>
<tr>
<td>com.tp.springdata.neo4j.domain</td>
</tr>
<tr>
<td>com.tp.springdata.neo4j.service</td>
</tr>
<tr>
<td>src/main/resources</td>
</tr>
<tr>
<td>src/main/java</td>
</tr>
<tr>
<td>src/test/java</td>
</tr>
<tr>
<td>src/test/resources</td>
</tr>
<tr>
<td>jre System Libray [J2SE-1.5]</td>
</tr>
<tr>
<td>Maven Dependencies</td>
</tr>
<tr>
<td>src</td>
</tr>
<tr>
<td>target</td>
</tr>
<tr>
<td>pom.xml</td>
</tr>
</tbody>
</table>
```
NOTE:-

When we run this program, please make sure that our Neo4j Database server should be in SHUTDOWN state.

**Before executing this Java Program, check your Neo4j is in Shutdown mode or not. If not, please click on "Stop" button to show down it.**

![Neo4j Database Server Status](image1)

Run GoogleProfileTest.java application and see the results

When we run this CREATE operation in Eclipse IDE, we can observe the following messages in console.

```
<terminated> GoogleProfileTest [Java Application] C:\UDK1.7.0_60\bin\javaw.exe [19 Aug 2014 17:26:24]
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org_CODES.html#StaticLoggerBinder for further details.
GoogleProfile created successfully.
```

After executing some operations, then START the Neo4j Database server by clicking "Start" button

![Neo4j Database Server Status](image2)

Open Neo4j Data Browser by using http://localhost:7474/ url

![Neo4j Data Browser](image3)
Type the below command on Data Browser

```
MATCH (gp:GoogleProfile)
RETURN gp
```

Click on "Execute" button and observe the results
So far we have created 3 GoogleProfile nodes.