***Logging***

You may need something extra logging, specially to validate the autowiring when failind.

File ***application.properties***

Add:

logging.level.org.springframework = DEBUG

***Database & JDBC***

Good collection of code snippet to access databases using Java JDBC+Spring

<https://mkyong.com/spring/spring-jdbctemplate-querying-examples/>

***Needed:***

***1 -A Spring Config Class***

@Configuration

@ComponentScan("ca.gc.cbsa.usstates")

**public** **class** SpringJdbcConfig {

@Bean

**public** DataSource postgreSQLlDataSource() {

DriverManagerDataSource dataSource = **new** DriverManagerDataSource();

dataSource.setDriverClassName("org.postgresql.Driver");

dataSource.setUrl("jdbc:postgresql://localhost:5432/USElection");

dataSource.setUsername("postgres");

dataSource.setPassword("D@njo2020");

**return** dataSource;

}

}

***2 - A Row Rapper Class (convenient, not mandatory)***

**public** **class** StateRowMapper **implements** RowMapper<State> {

@Override

**public** State mapRow(ResultSet rs, **int** rowNum) **throws** SQLException {

State state = **new** State();

state.setId(rs.getInt("id"));

state.setName(rs.getString("NAME"));

state.setAbreviation(rs.getString("abreviation"));

state.setElectoralVotes(rs.getInt("electoralVotes"));

**return** state;

}

}

}

}

***3 - A JdbcTemplate field and invocation on it***

In the calling class:

@Autowired

JdbcTemplate jdbcTemplate;

**public** State findStateById(**int** id) {

String sql = "SELECT \* FROM \"USState\" WHERE id = ?";

**return** jdbcTemplate.queryForObject(sql,

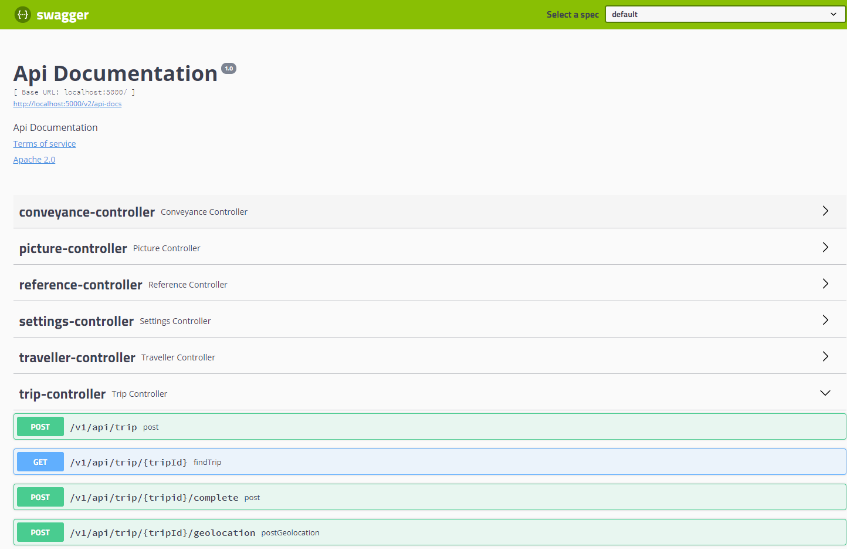
**new** Object[]{id},

**new** StateRowMapper());

}

***REST***

You can have a web display of all the URL exposed by a Spring Boot REST server, using swagger:



A simple, but fully working example can be found here:

<https://www.javainuse.com/spring/boot_swagger>

And you can query the boot REST api using a link like:

<http://localhost:8080/swagger-ui.html>