Data Access

Requirements

Lab Requirements

Purpose of this lab

- Get familiar with Spring Boot, JDBC and in-memory data persistence
- Demonstrate how easy it is to switch from an in-memory database to a MySQL database
- Estimated time: 30 minutes

JDBC in-memory data project

1. Create a project with the following attributes using these directions.
   - **Group**: `io.pivotal.workshop`
   - **ArtifactId**: `code-snippet-manager-jdbc`
   - **Dependencies**: `web, jdbc, h2, devtools`
2. Create a `SnippetRecord` class in the `io.pivotal.workshop` package.

```
package io.pivotal.workshop;

import java.time.LocalDate;

public class SnippetRecord {

    public final String id;
    public final String title;
    public final String code;
    public final LocalDate created;
    public final LocalDate modified;

    public SnippetRecord(String id, String title,
String code, LocalDate created, LocalDate modified)
{
        this.id = id;
        this.title = title;
        this.code = code;
        this.created = created;
        this.modified = modified;
    }
}
```

3. Create a `NewSnippetFields` class in the `io.pivotal.workshop` package. This class will be used to encapsulate the title and code fields on snippet creation.

```
package io.pivotal.workshop;

public class NewSnippetFields {

    public final String title;
    public final String code;

    public NewSnippetFields(String title, String co
de) {
        this.title = title;
        this.code = code;
    }

    // Make jackson happy when parsing JSON into th
is class
    private NewSnippetFields() {
        this(null, null);
    }
}
```

4. Create a `SnippetInfo` class in the `io.pivotal.workshop` package. This class will be used to generate the JSON produced by our controller.

```
package io.pivotal.workshop;

public class SnippetInfo {

    public final String id;
    public final String title;
    public final String code;
    public final String created;
    public final String modified;

    public SnippetInfo(String id, String title,
String code, String created, String modified) {
        this.id = id;
        this.title = title;
        this.code = code;
        this.created = created;
        this.modified = modified;
    }
}
```

5. Create a `SnippetPresenter` class in the `io.pivotal.workshop` package. The responsibility of this class is to convert our `SnippetRecord` to a `SnippetInfo`.

```
package io.pivotal.workshop;

import org.springframework.stereotype.Component;

import java.time.format.DateTimeFormatter;

@Component
public class SnippetPresenter {

    DateTimeFormatter formatter = DateTimeFormatter
.ISO_DATE;

    public SnippetInfo present(SnippetRecord record
) {
        return new SnippetInfo(
            record.id,
            record.title,
            record.code,
            record.created.format(formatter),
            record.modified.format(formatter)
        );
    }
}
```

6. Create a `SnippetRepository` class in the `io.pivotal.workshop` package that is constructed with a `JdbcTemplate`. This class will save a snippet, find a snippet, and list all snippets.

```
package io.pivotal.workshop;

import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

import java.sql.*;
import java.util.List;

import static java.util.UUID.randomUUID;

@Repository
public class SnippetRepository {

    private final JdbcTemplate jdbcTemplate;
    private final RowMapper<SnippetRecord>
rowMapper = (ResultSet rs, int row) -> new SnippetR
ecord(
        rs.getString("id"),
        rs.getString("title"),
        rs.getString("code"),
        rs.getDate("created").toLocalDate(),
        rs.getDate("modified").toLocalDate()
    );

    public SnippetRepository(JdbcTemplate jdbcTempl
ate) {
        this.jdbcTemplate = jdbcTemplate;
    }


    private final String SQL_INSERT = "insert into
snippet (id, title, code, created, modified)" +
        " values(?, ?, ?, now(), now())";

    public SnippetRecord save(NewSnippetFields newS
nippetFields) {
        String newId = randomUUID().toString();

        jdbcTemplate.update(SQL_INSERT, newId,
newSnippetFields.title, newSnippetFields.code);

        return findOne(newId);
    }


    private final String SQL_QUERY_ALL = "select *
from snippet";

    public List<SnippetRecord> findAll() {
        return jdbcTemplate.query(SQL_QUERY_ALL,
rowMapper);
    }


    private final String SQL_QUERY_BY_ID = "select
* from snippet where id = ?";

    public SnippetRecord findOne(String id) {
        return jdbcTemplate.queryForObject(SQL_QUER
Y_BY_ID, new Object[]{id}, rowMapper);
    }
}
```

7. Spring Data can initialize your database at start up by reading specially named files in the `src/main/resources` directory. Create the following files:
   - `src/main/resources/schema.sql`

```
DROP TABLE IF EXISTS snippet;

CREATE TABLE snippet
(
    id varchar(36) NOT NULL,
    title varchar(200) NOT NULL,
    code varchar(500) DEFAULT NULL,
    created date NOT NULL,
    modified date NOT NULL,
    PRIMARY KEY (id)
);
```

   - `src/main/resources/data.sql`

```
insert into snippet (id, title, code, created,
modified) values ('66921076-ed1d-458b-9d7d-ce9a
227d64a5', 'JavaScript: Hello World', 'console.
log("Hello World!");', '2016-07-31', '2016-07-3
1');
insert into snippet (id, title, code, created,
modified) values ('4465afe0-4e8f-4779-90e3-e6b9
71c4cc7d', 'HTML: Hello World', '<html><body><h
1>Hello World</h1></body></html>', '2016-07-31'
, '2016-07-31');
insert into snippet (id, title, code, created,
modified) values ('842c4bd0-32a0-4c6e-8f7a-74c4
5c23ddf1', 'Bash: Hello World', 'echo "Hello Wo
rld"', '2016-07-31', '2016-07-31');
insert into snippet (id, title, code, created,
modified) values ('7d1bce20-799b-49b0-8306-6f55
8aac4dd9', 'Python: Hello World', 'print "Hello
World"', '2016-07-31', '2016-07-31');
```

8. Create a `SnippetController` class in the `io.pivotal.workshop` package.

```
package io.pivotal.workshop;

import org.springframework.http.*;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.Serv
letUriComponentsBuilder;

import java.net.URI;
import java.util.List;

import static java.util.stream.Collectors.toList;

@RestController
@RequestMapping("/snippets")
public class SnippetController {

    private final SnippetRepository snippetReposito
ry;
    private final SnippetPresenter snippetPresenter
;

    public SnippetController(SnippetRepository snip
petRepository, SnippetPresenter snippetPresenter) {
        this.snippetRepository = snippetRepository;
        this.snippetPresenter = snippetPresenter;
    }


    @GetMapping
    public List<SnippetInfo> snippets() {
        return snippetRepository.findAll()
            .stream()
            .map(snippetPresenter::present)
            .collect(toList());
    }

    @GetMapping("/{id}")
    public SnippetInfo snippet(@PathVariable("id")
String id) {
        SnippetRecord record = snippetRepository.fi
ndOne(id);
        return snippetPresenter.present(record);
    }

    @PostMapping
    public ResponseEntity<SnippetInfo> add(@Request
Body NewSnippetFields newSnippetFields) {
        SnippetRecord savedSnippetRecord =
snippetRepository.save(newSnippetFields);
        SnippetInfo savedSnippetInfo =
snippetPresenter.present(savedSnippetRecord);

        HttpHeaders httpHeaders = new HttpHeaders()
;
        httpHeaders.setLocation(buildSnippetUri(sav
edSnippetInfo));

        return new ResponseEntity<>(savedSnippetInf
o, httpHeaders, HttpStatus.CREATED);
    }


    private URI buildSnippetUri(SnippetInfo snippet
) {
        return ServletUriComponentsBuilder
            .fromCurrentRequest().path("/" +
snippet.id)
            .buildAndExpand().toUri();
    }
}
```

9. Run the application.
10. The H2 library comes with an embedded SQL browser that you can access at localhost:8080/h2-console. The default database name is *testdb* and the complete JDBC URL is `jdbc:h2:mem:testdb` Try running a query through the web UI.
11. Open a terminal window and run the following command:

```
curl -s localhost:8080/snippets
```

12. Use `curl` to add a new snippet code and review the results in the h2-console and via the `/snippets` endpoint.

JDBC MySQL project

1. You will start this section with the code from the previous section.
2. Add a mysql user and create a database:

```
mysql -uroot
```

```
create user 'springboot'@'localhost' identified by
'workshop';
create database testdb;
grant all privileges on testdb.* to 'springboot'@'l
ocalhost';
```

3. Open your `build.gradle` file and replace the *h2* dependency with a `mysql` dependency.

```
runtime('mysql:mysql-connector-java')
```

4. Configure your datasource to use MySQL in `src/main/resources/application.properties`:

```
spring.datasource.driver-class-name=com.mysql.jdbc.Dr
iver
spring.datasource.url=jdbc:mysql://localhost:3306/tes
tdb?useSSL=false
spring.datasource.username=springboot
spring.datasource.password=workshop
```

5. Run the application.
6. Open another terminal window and run the following command: `curl -s localhost:8080/snippets`
7. Using `curl` add a new snippet and review the results in the `/snippets` endpoint and through the `mysql` commandline client.

Challenges

1. Modify the `SnippetRepository` class by adding a function that allows searching between snippet creation Dates. Create a custom SQL query and modify the `SnippetController` to support this feature.