S2 Submission

Haiwei He

February 28, 2024

Contents

1 Spring Boot Tech Report

Overview

Spring Boot is a framework built on top of Spring.
It's designed to simplify development and deployment of standalone Spring-based applications with simplified dependency injection, auto-configuration and built-in production-ready features.

About Spring

Spring framework was designed to support developing enterprise-grade Java applications with comprehensive infrastructure.

It's built on 5 key principles:

1. Inversion of Control
   - Outsource the construction and management of objects to the IoC container which create, assemble, wire together and manage objects.
   - Decouples tasks from their implementation, allowing them to be swapped out.
   - Register an object *(Called "Bean" in Spring)* once on how to provide it, then it will be injected into other objects that depend on it.
   - Objects can be denoted with annotations that helps the IoC container decide which one to use for a type in different contexts.
2. Dependency Injection
   - A form of IoC where components receive dependency via constructor arguments, factory methods and properties.
3. Aspect-Oriented Programming
   - Modularizes cross-cutting concerns which are parts of program that must affect many other parts. E.g. Logging or Validation.
   - Aspects are implementation of cross-cutting concerns and advice are methods within an aspect defined with a pointcut.
   - Pointcut are expressions that matches a joincut and joincut is a point in execution of the program.
   - Proxy objects are created when aspects are weaved into other parts. Which wraps the original object, but performs advices when a joincut matches an advice's pointcut.
4. Convention Over Configuration
   - Use default configuration that works for majority of use cases than the developer nit picking the perfect configuration
   - Reduce number of decisions requires which results in fewer errors.
5. Portability
   - Framework's responsible for adapting application components to the hosting requirement. Which allows support for older platforms.

Spring Boot Additions

Spring Boot improves on Spring with 4 features:

1. Simplified Dependency Injection
   - Provides starters which are sets of convenient dependency descriptors that include related dependencies.
2. Auto-Configuration
   - Automatically provide default configure application based on dependencies present in the classpath.
3. Standalone
   - Come with embedded server, don't need to be deployed on to an external web server.
4. Actuator Module
   - Provides out of the box production-ready features.

Our Back-End

Spring Boot provides the back-end consisting of 4 layers:

1. API Layer *(Specifics of this layer will be covered by team member Rhys)*
   - Exposes API endpoints for the front-end or external systems to communicate with.
   - Process request and pass it onto the service layer.
2. Service Layer
   - Validates data from the request, performs business logic and invoke additional relevant service calls.
   - Pass the data to the repository layer if CRUD operation is required.
3. Repository Layer *(Specifics of this layer will be covered by team member Garance)*
   - Performs CRUD operation on the database.

An Example Request

Walkthrough of a POST request to API endpoint `/api/register`.

API Layer

It will be handled by the `registerAccount` method in `AccountResource` controller.

```
@RestController
@RequestMapping("/api")
public class AccountResource {
...
```

```
@PostMapping("/register")
@ResponseStatus(HttpStatus.CREATED)
public void registerAccount(@Valid @RequestBody ManagedUserVM managedUserVM) {
    if (isPasswordLengthInvalid(managedUserVM.getPassword())) {
        throw new InvalidPasswordException();
    }
    User user = userService.registerUser(managedUserVM, managedUserVM.getPassword());
    mailService.sendActivationEmail(user);
  }
}
```

`@RequestBody` annotation maps the request body data into `managedUserVM` parameter.

`@Valid` annotation validates the data using constraints set in the `ManagedUserVM` class.

```
public class ManagedUserVM extends AdminUserDTO {
...
@Size(min = PASSWORD_MIN_LENGTH, max = PASSWORD_MAX_LENGTH)
private String password;
```

It then pass the data onto the service layer for it to perform the register user business logic using `userService` and `mailService`.

Service Layer

For the sake of the conciseness of the walkthrough `mailService` will be omitted.

Business logic for registering will be performed by the method `registerUser` in `userService`

```
@Service
@Transactional
public class UserService {
...
public User registerUser(AdminUserDTO userDTO, String password) {
    userRepository
        .findOneByLogin(userDTO.getLogin().toLowerCase())
        .ifPresent(existingUser -> {
            boolean removed = removeNonActivatedUser(existingUser);
            if (!removed) {
                throw new UsernameAlreadyUsedException();
            }
        });
    ... //more validation
    User newUser = new User();
    ... //new user business logic
    authorityRepository.findById(AuthoritiesConstants.USER).ifPresent(authorities::add);
    newUser.setAuthorities(authorities);
    userRepository.save(newUser);
    this.clearUserCaches(newUser);
    log.debug("Created Information for User: {}", newUser);
    return newUser;
  }
}
```

`@Service` annotation indicates to Spring Boot the class is responsible for business logic and made it available for `accountResource` and other parts via dependency injection.
`@Transactional` is used to mark class or methods as transactional and will trigger a rollback if exceptions are thrown to ensure data integrity of the database.

In the code above `userRepository` from the repository layer is employed to perform CRUD operation in order to validate and create new user.

> **Aop example**
>
> In the code above the advice `logAfterThrowing` from `LoggingAspect` will be triggered when an exception is thrown.
> As It's pointcut matches any point in execution when an exception is thrown.

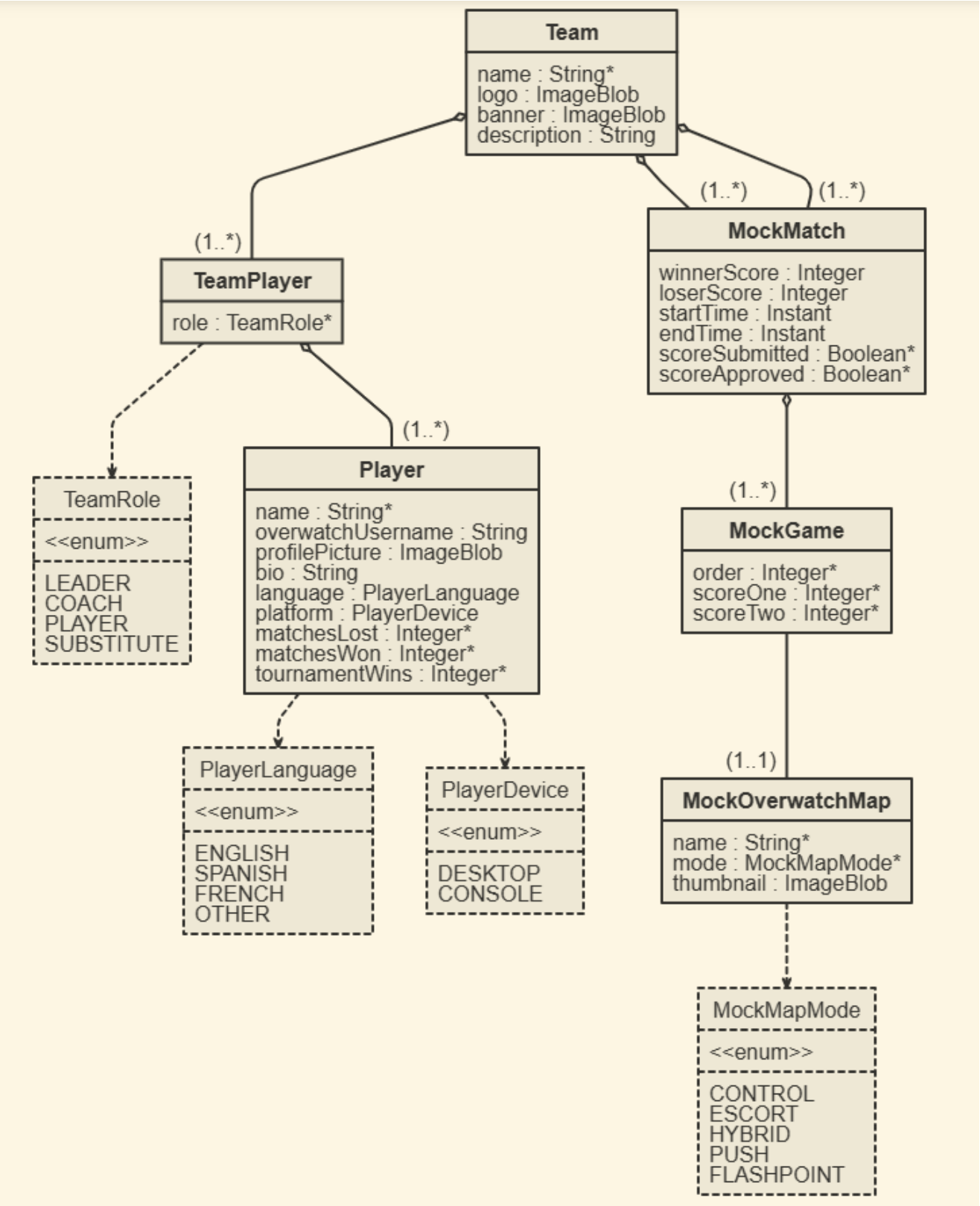
Repository Layer

In the repository layer `UserRepository` uses JPA to interface the database and perform operations.

```
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
```

2 UML

2.1 Diagram



2.2 JDL

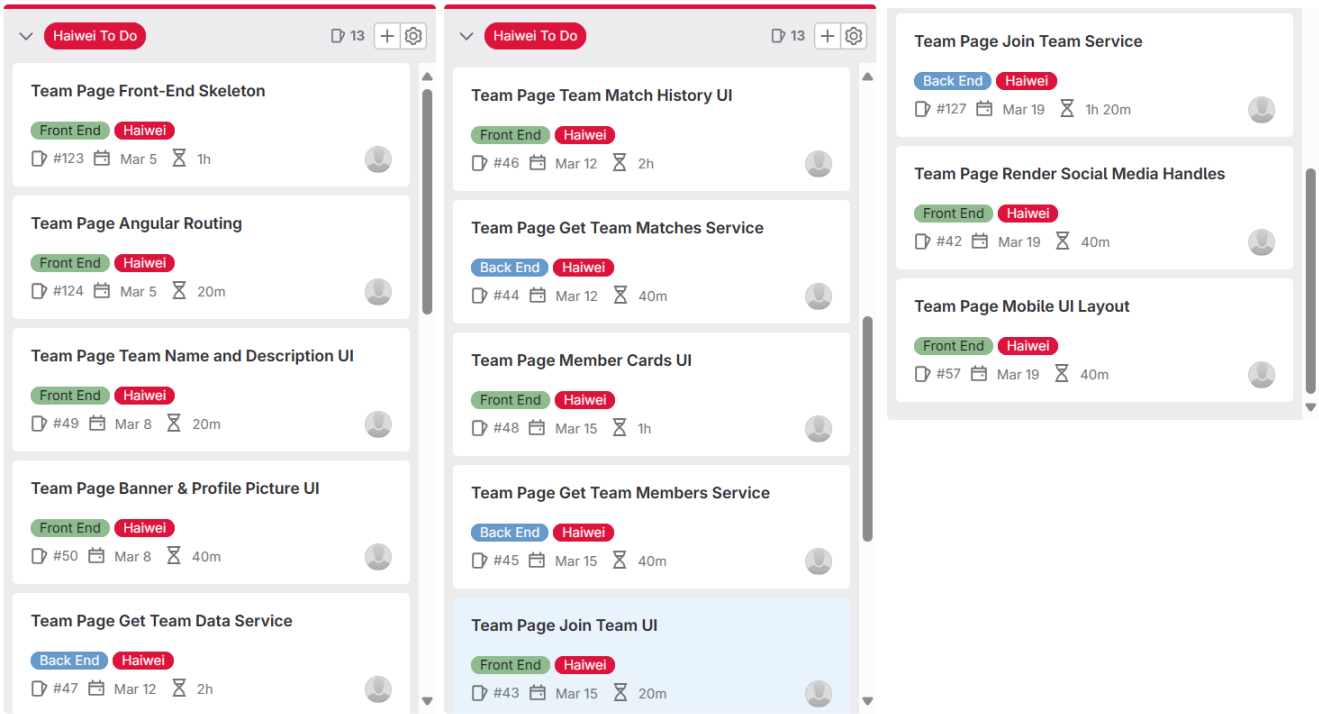
```
enum TeamRole {
    LEADER
    COACH
    PLAYER
    SUBSTITUTE
}
enum MockMapMode {
    CONTROL
    ESCORT
    HYBRID
    PUSH
    FLASHPOINT
}
enum PlayerLanguage {
    ENGLISH
    SPANISH
    FRENCH
    OTHER
}
enum PlayerDevice {
    DESKTOP
    CONSOLE
}
entity Team{
    name String required unique
    logo ImageBlob
    banner ImageBlob
    description String
}
entity MockMatch{
    winnerScore Integer min(0)
    loserScore Integer min(0)
    startTime Instant
    endTime Instant
   scoreSubmitted Boolean required
   scoreApproved Boolean required
}
entity TeamPlayer {
    role TeamRole required
}
entity Player{
    name String required minlength(3) maxlength(50)
    overwatchUsername String minlength(3) maxlength(40)
    profilePicture ImageBlob
    bio String maxlength(1000)
    language PlayerLanguage
    platform PlayerDevice
    matchesLost Integer required min(0)
    matchesWon Integer required min(0)
    tournamentWins Integer required min(0)
}
entity MockOverwatchMap{
    name String required
    mode MockMapMode required
    thumbnail ImageBlob
}
entity MockGame{
    order Integer required min(0)
    scoreOne Integer required
    scoreTwo Integer required
}
relationship OneToMany{
    Tournament to MockMatch
    Team{teamOne} to MockMatch{teamOne}
```

```
    Team{teamTwo} to MockMatch{teamTwo}
    Player to GamePlayer
    MockMatch to MockGame

    TeamPlayer{player} to Player{player}
    Team{team} to TeamPlayer{team}
}
relationship OneToOne{
    MockGame to MockOverwatchMap
}
```

3 Kanban Cards

3.1 TO-DO List



Cards are ordered by importance and have been assigned time estimates and due dates. Every card has been given description of it's requirements and resources that may help.

3.2 Example Card Descriptions

Team Page Front-End Skeleton

Open Issue created 1 hour ago by **Haiwei He**

Generate angular component and create skeleton HTML and SCSS

> Refer to Mark's Angular tech report

Team Page Team Name and Description UI

Open Issue created 3 weeks ago by **Haiwei He**

Write front-end code to:

- query team name and description from get team data service.
- update the page content accordingly.

> Refer to Rhys' API tech report and Mark's Angular tech report

Edited just now by Haiwei He

Team Page Get Team Matches Service

Open Issue created 3 weeks ago by **Haiwei He**

Write back-end code to:

- listen to API request for match result.
- service to query the repository layer for data.
- calculate statistics.
- return response object.

> Refer to Rhys' API tech report, Haiwei's Spring Boot tech report and Garance's JPA tech report

Edited just now by Haiwei He

4 Timesheets

Team sheet Number/ID: heh04

Team member name: Haiwei He

Team representative (secrata Talha

Team meeting sign off date: 20.02.2024

Date from: 13.02.2024

Date until: 18.02.2024

Task	Date	Start time	End time	Total Hours
Team Meeting with Tutor - Discuss S2 / M2	13.02.2024	2:00 PM	2:40 PM	0:40
Team Meeting - Progress Check + plan for M2	16.02.2024	12:30 PM	1:30 PM	1:00
Research Spring Boot	18.02.2024	5:00 PM	6:00 PM	1:00
Write Tech Report	20.02.2024	3:00 AM	6:00 AM	3:00
				0:00
				0:00
				0:00

Total Hours **5:40**

Team sheet Number/ID: heh05

Team member name: Haiwei He

Team representative (secrata Ogieltaziba

Team meeting sign off date: 27.02.2024

Date from: 20.02.2024

Date until: 27.02.2024

Task	Date	Start time	End time	Total Hours
Team Meeting with Tutor - Discuss S2 / M2	20.02.2024	2:00 PM	3:23 PM	1:23
Improve Tech Report	23.02.2024	12:00 AM	4:22 AM	4:22
Create class diagram	23.02.2024	4:22 AM	4:39 AM	0:17
Team Meeting - JDL	23.02.2024	12:00 PM	2:50 PM	2:50
				0:00
				0:00
				0:00

Total Hours **8:52**

Team sheet Number/ID: heh06

Team member name: Haiwei He

Team representative (secrata Mark

Team meeting sign off date:

Date from: 27.02.2024

Date until: 05.03.2024

Task	Date	Start time	End time	Total Hours
Team Meeting with Tutor - Discuss S2 / M2	27.02.2024	2:30 PM	4:17 PM	1:47
Work on detailed Kanban cards and compile S2 submission	27.02.2024	9:20 PM	11:56 PM	2:36
				0:00
				0:00
				0:00
				0:00
				0:00

Total Hours **4:23**