ML Draughts

Design Patterns

`> version: beta`

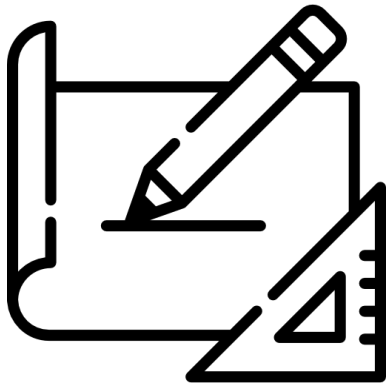
Lukas Treffner
Martin Wechsler

Graz University of Technology

12.12.2023 Winter-Term

https://mld.p4s3r0.com

Architecture



Architecture

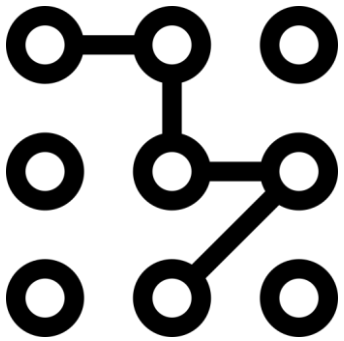
> Project Structure

- Server Client
  - Frontend: Vue.js
  - Backend: ASP.NET Core

- Communication Protocol
  - Sockets / RESTFul API
  - Custom communication interface

- Based on Docker

Design Patterns



Design Patterns

> Client/Server | > Factory Method | > Command | > Singleton

- First requirement: Multiplayer
- Backend: game logic
- Frontend: UI/UX
  - Progressive Web App
- Communication: (mostly) WebSockets
- Custom Messaging protocol:
  - CommandType
  - Parameter (e.g., game_id, client_id …)
  - Response: JSON

Design Patterns

> Client/Server | > Factory Method | > Command | > Singleton

- Create commands
- Command-Interface:
  - ICommand
- Factory is Singleton

```
public class CommandFactory : ICommandFactory {
    private readonly IGameCache _gameCache;
    public CommandFactory(IGameCache gameCache) {
        _gameCache = gameCache;
    }

    public ICommand CreateCommand(string
         socketMessage, WebSocket socket) {
        var commands = socketMessage.Split(';');
        switch(commands[0]) {
            case "init":
                return new InitCommand(…);
            …
        }
    }
}
```

Design Patterns

> Client/Server | > Factory Method | > Command | > Singleton

```
public interface ICommand{
        public Type CommandType {get; set;}
        public Boolean CommandValid {get;}
        public Response HandleCommand();
}
```

- Slim down main loop
  - One call to handle all requests
- Execute() = HandleCommand()
- Returns correct Response object

```
public class UnknownCommand : ICommand{
        …
        public Response HandleCommand() {
            return new Response(ResponseTypes.UnknownCommand);
        }
}
```

Design Patterns

> Client/Server | > Factory Method | > Command | > Singleton

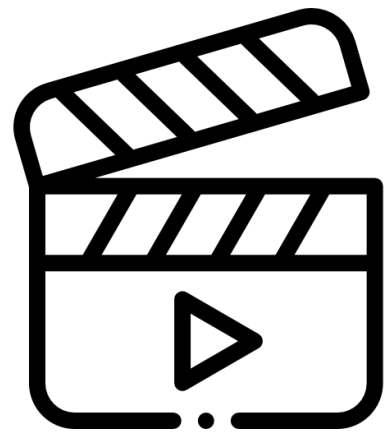
Program.cs:

```
…
builder.Services.AddSingleton<IGameCache, GameCache>();
builder.Services.AddSingleton<ICommandFactory, CommandFactory>();
…
```

- Only once
- GameCache
  - Game information
  - Multiplayer
- CommandFactory

```
public class GameCache : IGameCache {
        private readonly ConcurrentDictionary<string, Draughts> _gameCache
                = new ConcurrentDictionary<string, Draughts>();
        …
}
```

DEMO



Special thanks

- https://www.flaticon.com/ (for icons in this slideset)
- Christian Pasero (for hosting the game on his server)