Google Guice

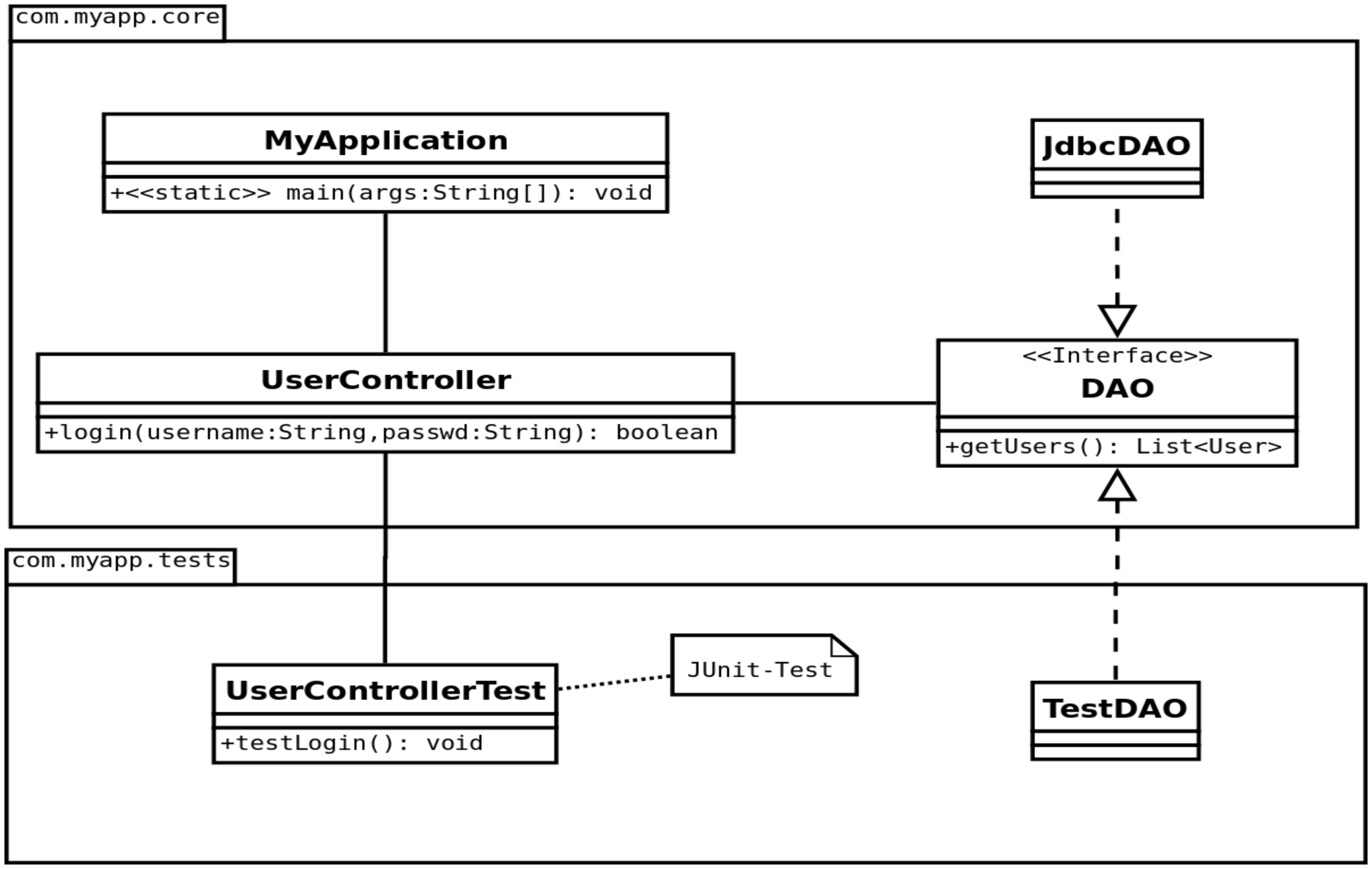
Depencency injection framework

Basic facts

- Pure Java
  - no external configuration files (compared to Spring)
  - bindings configuration in modules (Java classes)
- Uses annotations and generics
  - type safe
  - easy refactoring in your IDE
  - requires Java 5
- Open source under Apache License 2.0
- Integration with Servlets, Struts2 ans Spring possible

Simple example

Simple application with database and JUnit tests



Simple example

- Class UserController
    - subject of JUnit tests
    - requires a database

- Interface DAO - abstraction of all database operations
    - Class JdbcDAO - "real" databse
    - Class TestDAO - mockup implementation for JUnit tests

- How does UserController get an instance of DAO?

UserController

```
class UserController {
    private DAO dao;


    public UserController() {

    }

    public void someMethod {
        List<User> users = dao.getUsers();
        ....
    }
}
```

UserController - with Guice

```
class UserController {
    private DAO dao;

    @Inject
    public UserController(DAO dao) {
        this.dao = dao;
    }

    public void someMethod {
        List<User> users = dao.getUsers();
        ....
    }
}
```

UserController - with Guice (alternative)

```
class UserController {
    @Inject
    private DAO dao;

    public UserController() {

    }

    public void someMethod {
        List<User> users = dao.getUsers();
        ....
    }
}
```

Configuration - JUnit Tests

```
public class TestModule extends AbstractModule
{

    public void configure() {
        bind(DAO.class).to(TestDAO.class);
    }

}
```

Configuration - Application

```
public class AppModule extends AbstractModule
{

    public void configure() {
        bind(DAO.class).to(JdbcDAO.class);

    }

}
```

Bind to ...

- `bind(DAO.class).to(TestDAO.class);`

-
  ```
  bind(DAO.class).to(TestDAO.class)
      .in(Scopes.SINGLETON);
  ```
- `bind(DAO.class).toInstance(new TestDAO());`
- `bind(DAO.class).toProvider(DAOFactory.class);`
- Default implementation: `@ImplementedBy`

MyApplication

```
public static void main(String[] args) {
    ...
    Injector injector =
        Guice.createInjector(new AppModule());

    UserController userController =
        injector.
        getInstance(UserController.class);
}
```

JUnit Test

```
@Before
public void setup() {
    Injector injector =
        Guice.createInjector(new TestModule());

    UserController userController =
        injector.
        getInstance(UserController.class);
}
```

Much more...

- Annotated with:

```
@Inject
public SomeConstructor(@Blue DAO dao);
...
bind(DAO.class).
    annotatedWith(Blue.class).
    to(....);
```

- Annotations with attributes
- `ServletScopes.SESSION` bzw. `REQUEST`
- `SringIntegration`