EAPLI

Projeto PL Base eCafeteria

eCafeteria

- Cafeteria management
- Users, Kitchen and Menu managers, Cashiers
- User app
- Backoffice app
  - Kitchen management
  - Menu management
  - Delivery station

Scope of this lesson

- Project structure
- Architecture (same as proposed in classes)
- No business discussion
- Overview of existing codebase
- Focus on “User” and “Dish Type” domain concepts
- Two use cases
  - Register a new user
  - List all dish types

EAPLI-2016-eCafeteria

ACTIONS

- Clone
- Create branch
- Create pull request
- Compare
- Fork

NAVIGATION

- Overview
- Source
- Commits
- Branches
- Pull requests
- Downloads
- Settings

Source

master | EAPLI-2016-eCafeteria /

.idea			
backoffice.consoleapp			
consoleapp.common			
documentation			
ecafeteria.bootstrapapp			
ecafeteria.core			
framework			
utente.consoleapp			
util			
.gitignore	2.4 KB	3 days ago	added structure for menus of User App
Notes.txt	7.1 KB	2016-03-14	added notes documentation
README.md	378 B	2016-03-14	added notes documentation
build-console.bat	89 B	2016-03-14	updated script files templates
pom.xml	1.1 KB	3 hours ago	Comment tests ensureRoleTypeListIsNotEmpty, ensureInvalidAccessWithEmptyMemoryDatabase and ensureAuth
run-console.bat	455 B	2016-03-14	updated script files templates

Personal Expenses

Polythecnic of Porto, School of Engineering

EAPLI

==========

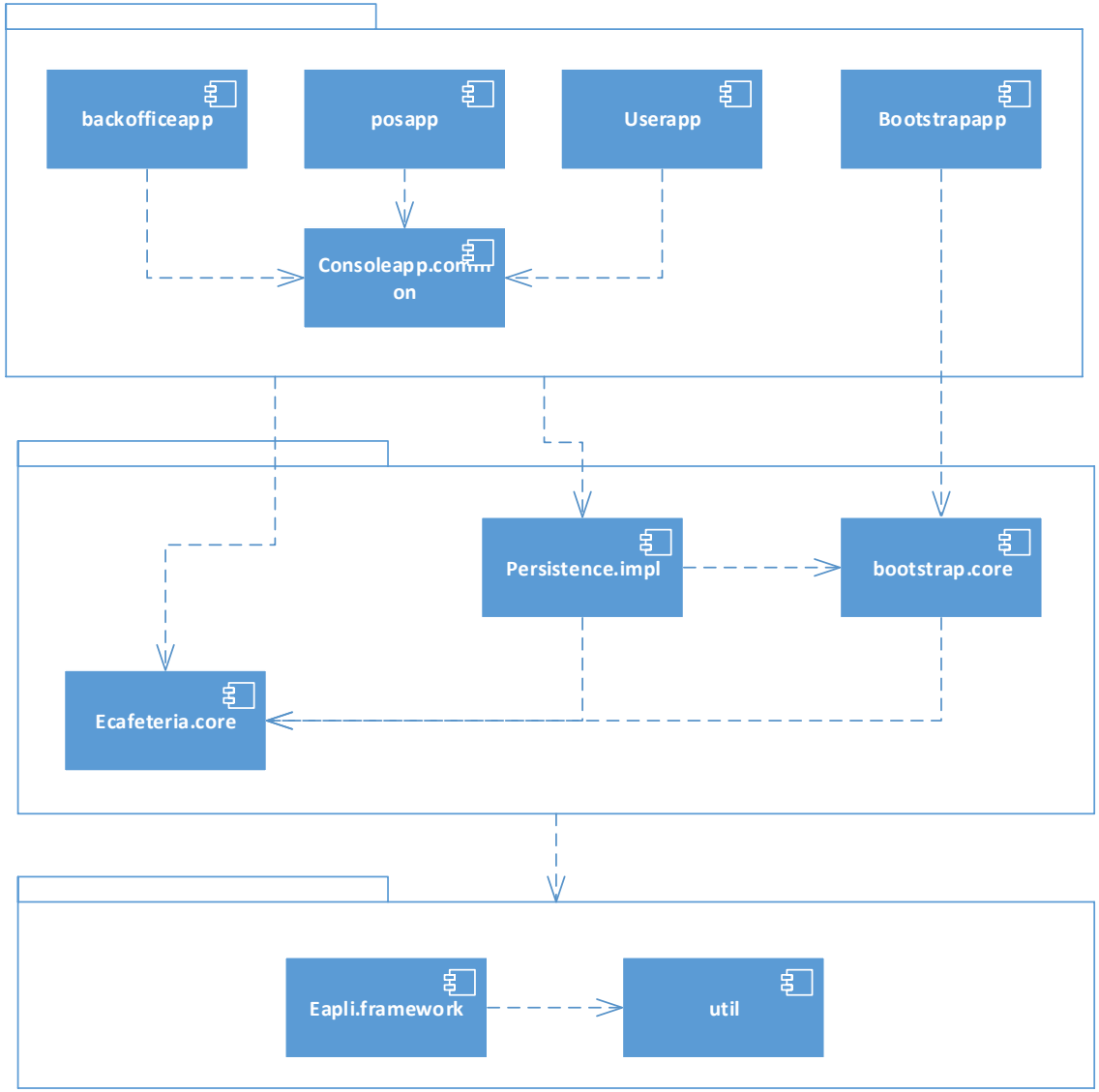
This application is part of the lab project for the course EAPLI.

Please check notes.txt for a discussion on design decissions.

BUILD

make sure JDK 1.7 is on the path

Components (a.k.a. projects)



- Backofficeapp, userapp, pos
- Core, console.common
- Persistence.impl
- bootstrap

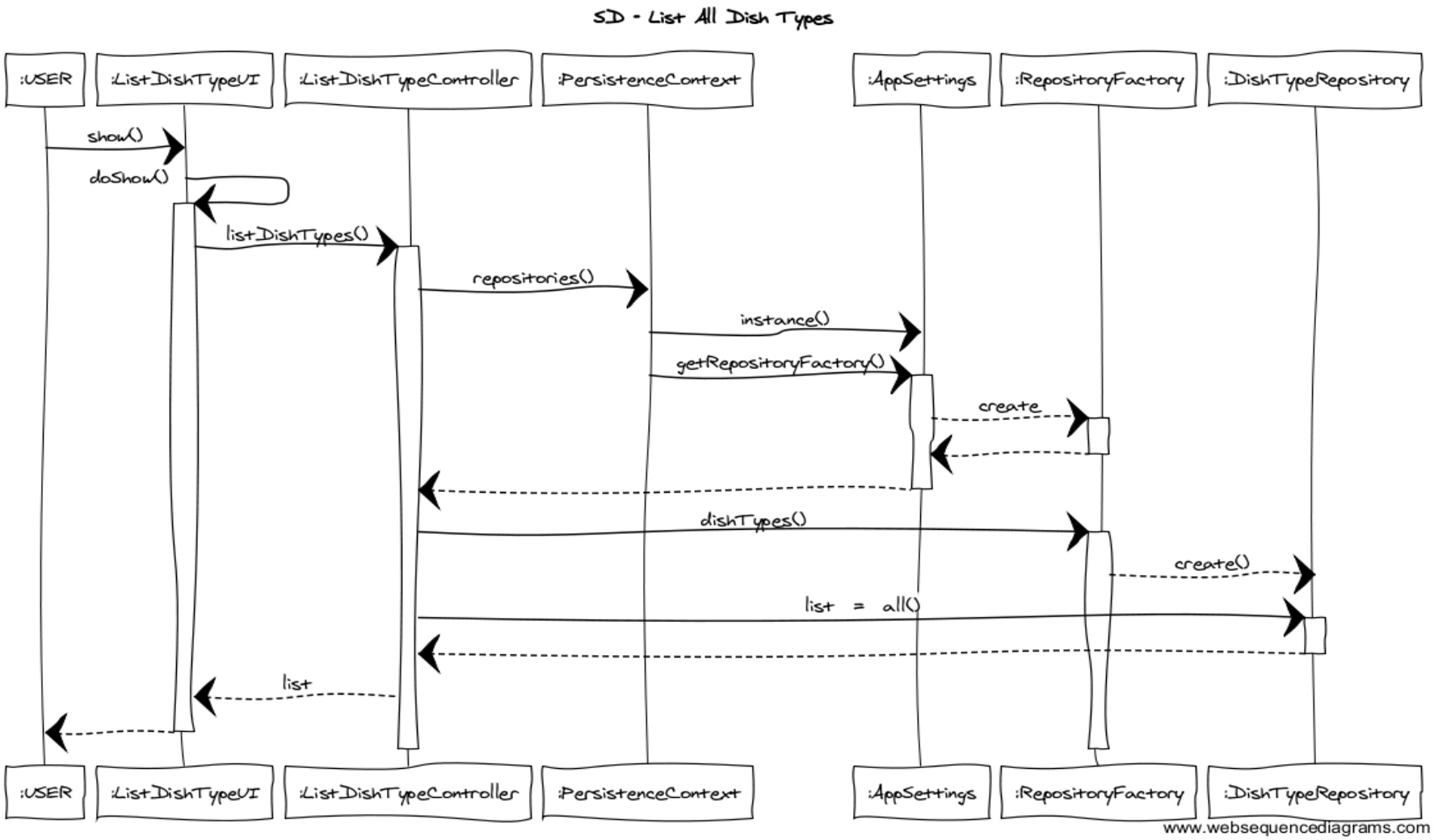
- Framework
  - Utility classes for DDD applications with JPA in EAPLI context
- Util
  - Generic utility classes

eCafeteria design decisions

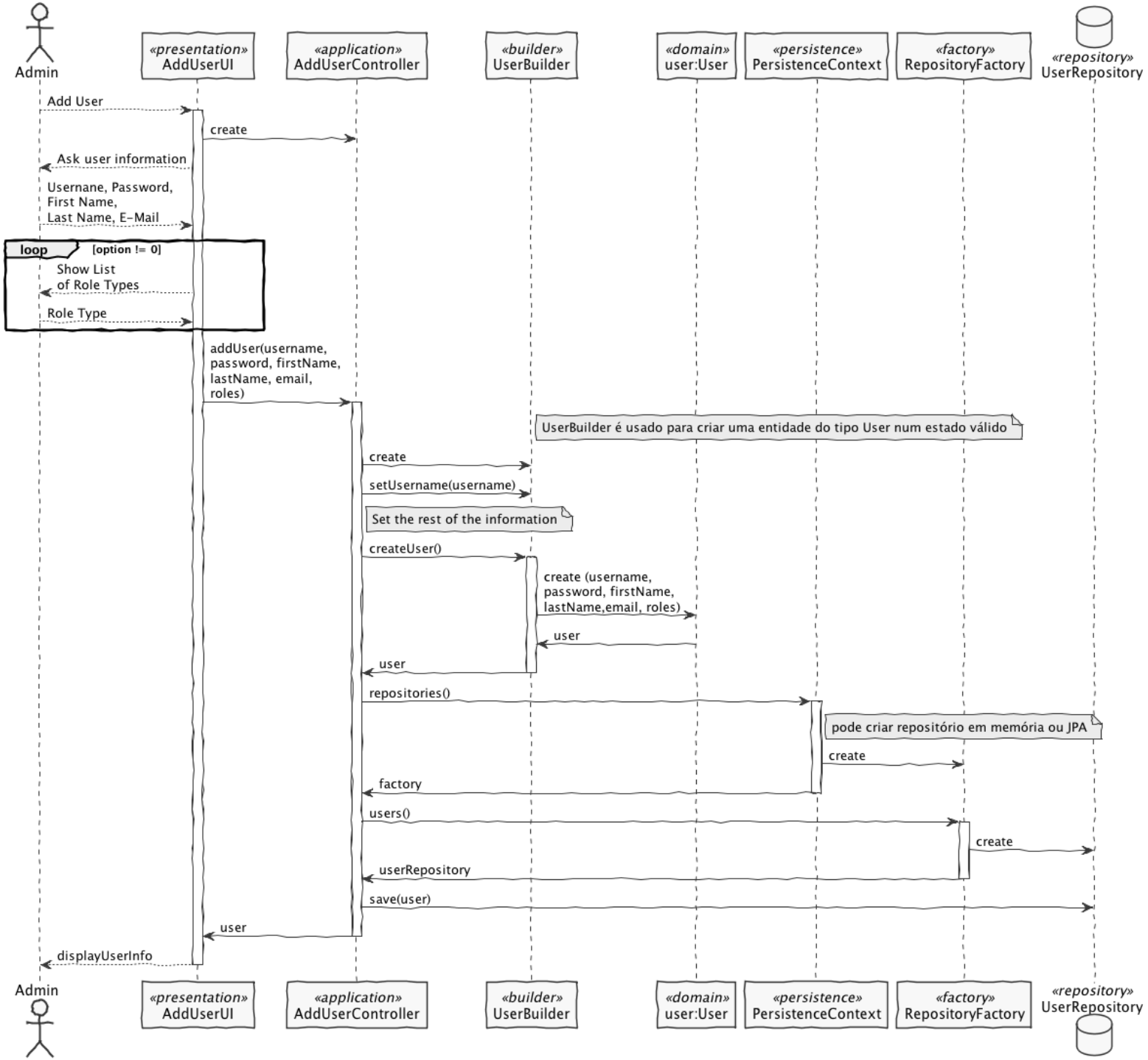
- Layers
  - Presentation
  - Application
  - Domain
  - Persistence
- Domain objects travel to UI for output

More on this next lesson

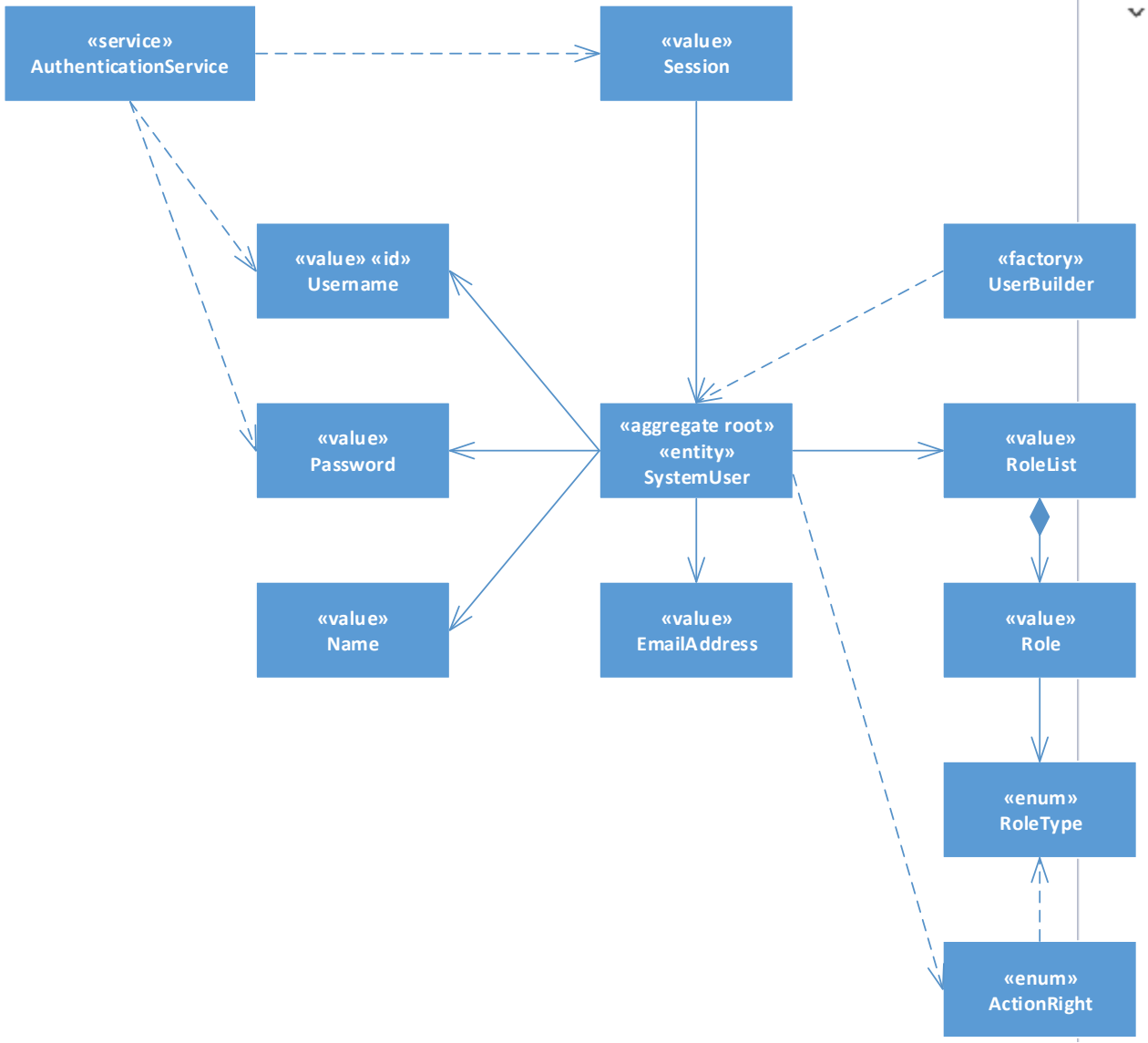
List Dish Types

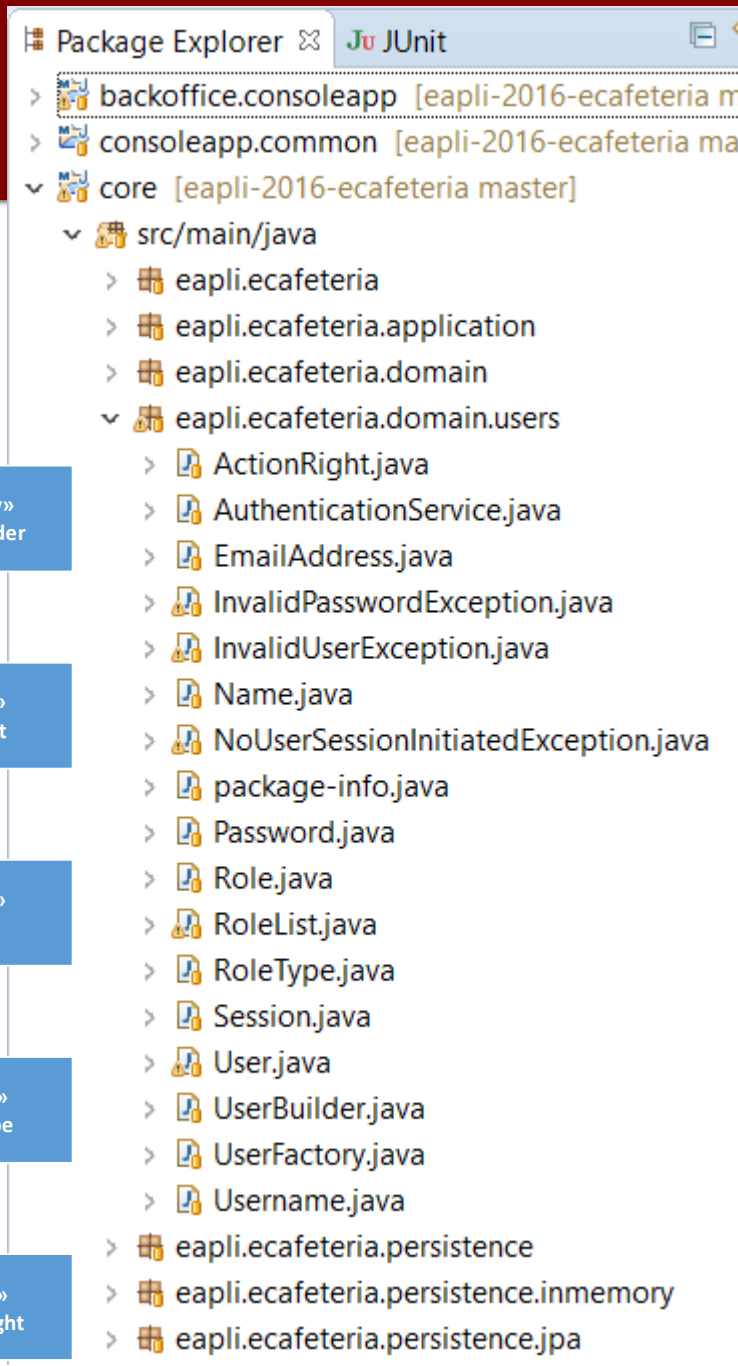


Register New User

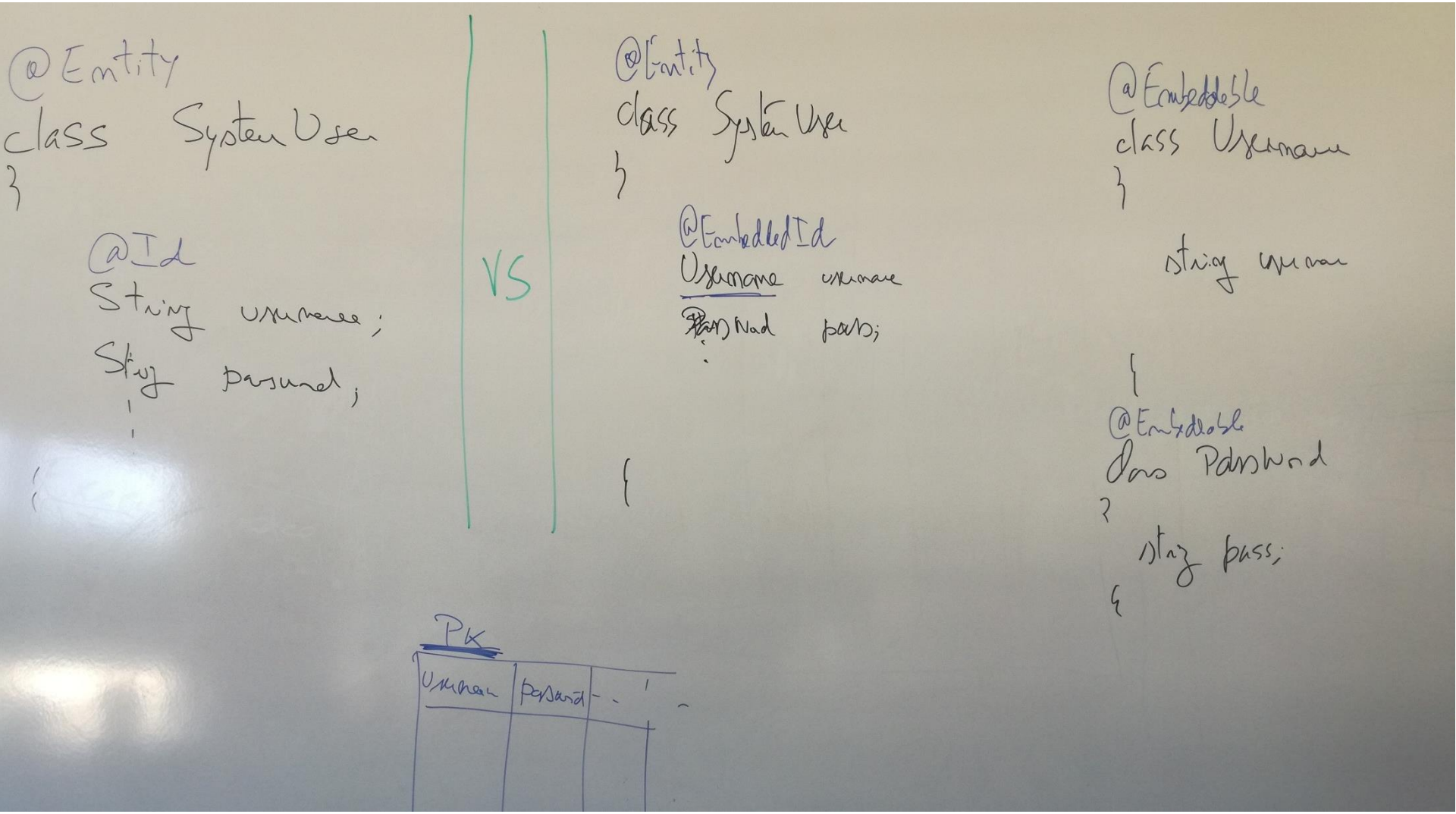


User model


«service»
AuthenticationService
«value»
Session
«value» «id»
Username
«factory»
UserBuilder
«value»
Password
«aggregate root»
«entity»
SystemUser
«value»
RoleList
«value»
Name
«value»
EmailAddress
«value»
Role
«enum»
RoleType
«enum»
ActionRight


Package Explorer
JUnit
backoffice.consoleapp [eapli-2016-ecafeteria m
consoleapp.common [eapli-2016-ecafeteria ma
core [eapli-2016-ecafeteria master]
src/main/java
eapli.ecafeteria
eapli.ecafeteria.application
eapli.ecafeteria.domain
eapli.ecafeteria.domain.users
ActionRight.java
AuthenticationService.java
EmailAddress.java
InvalidPasswordException.java
InvalidUserException.java
Name.java
NoUserSessionInitiatedException.java
package-info.java
Password.java
Role.java
RoleList.java
RoleType.java
Session.java
User.java
UserBuilder.java
UserFactory.java
Username.java
eapli.ecafeteria.persistence
eapli.ecafeteria.persistence.inmemory
eapli.ecafeteria.persistence.jpa

Embeddable vs regular fields



Domain invariants

```
@Test
public void ensurePasswordHasAtLeastOneDigitAnd6CharactersLong()
{
	new Password("abcdefgh1");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsSmallerThan6CharactersAreNotAllowed()
{
	new Password("ab1c");
}

@Test(expected = IllegalArgumentException.class)
public void ensurePasswordsWithoutDigitsAreNotAllowed() {
	new Password("abcdefgh");
}
```

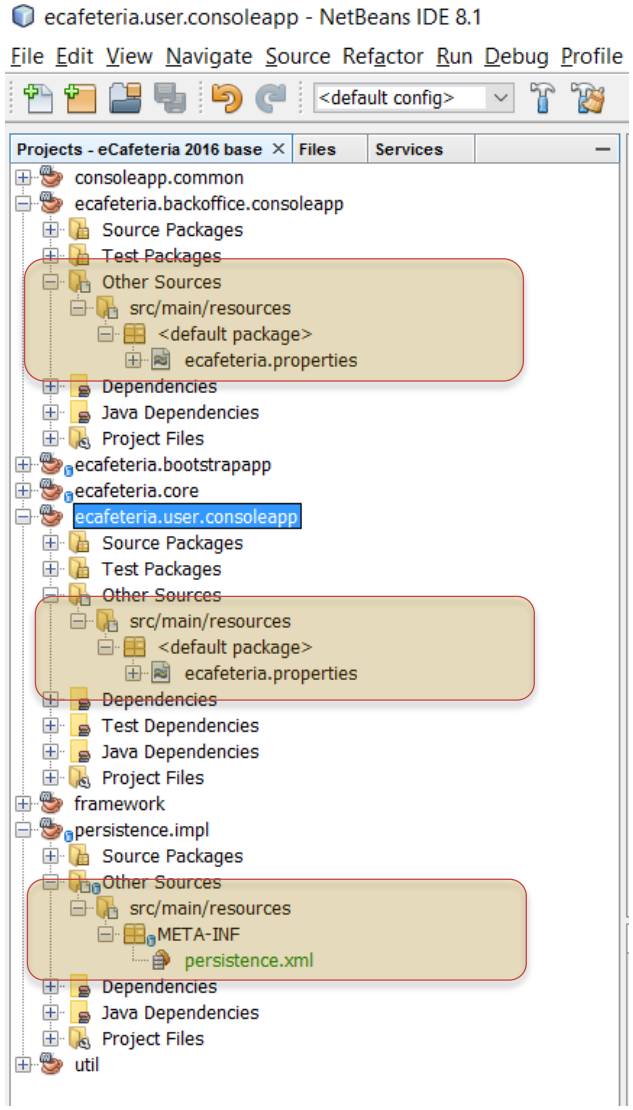
Domain

```
public class Password {

…

public Password(String password) {
        if (!meetsMinimumRequirements(password)) {
            throw new IllegalStateException();
        }
        thePassword = password;
}

private boolean meetsMinimumRequirements(String password) {
        if (Strings.isNullOrEmpty(password)
        || password.length() < 6)
        || !Strings.containsDigit(password))
        {
            return false;
        }

        return true;
}
```

Some additional design decisions

- Bootstrap data
- Support two repositories
  - In memory
  - Relational database
- Decide which repository implementation to use based on property file
- Simple main menu

Resources



- Persistence.impl has persistence.xml
- Application projects define the properties file

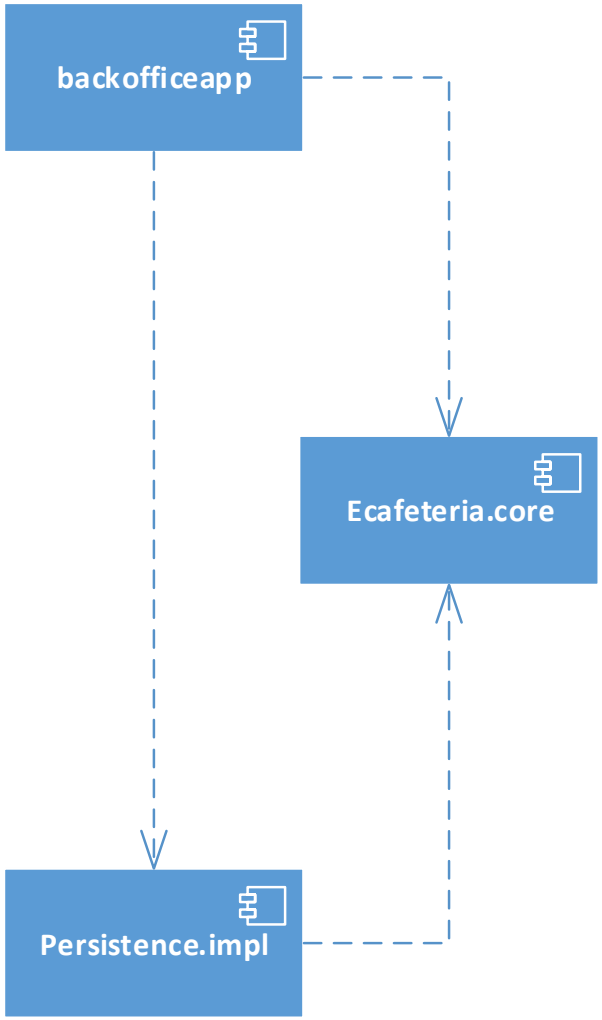
bootstrap

```
public class ECafeteriaBootstraper implements Action {

        @Override
        public boolean execute() {
                // declare bootstrap actions
                final Action[] actions = { new UsersBootstrap(), };
                // execute all bootstrapping
                boolean ret = false;
                for (final Action boot : actions) {
                        ret |= boot.execute();
                }
                return ret;
        }
}

public class UsersBootstrap implements Action {

        @Override
        public boolean execute() {
                registerAdmin();
                return false;
        }

        private void registerAdmin() {
                final String username = "admin";
                final String password = "admin";
                final String firstName = "John";
                final String lastName = "Doe";
                final String email = "john.doe@emai.l.com";
                final List<RoleType> roles = new ArrayList<RoleType>();
                roles.add(RoleType.Admin);

                final UserRegisterController userController = new UserRegisterController();
                userController.registerUser(username, password, firstName, lastName, email, roles);
        }
}
```

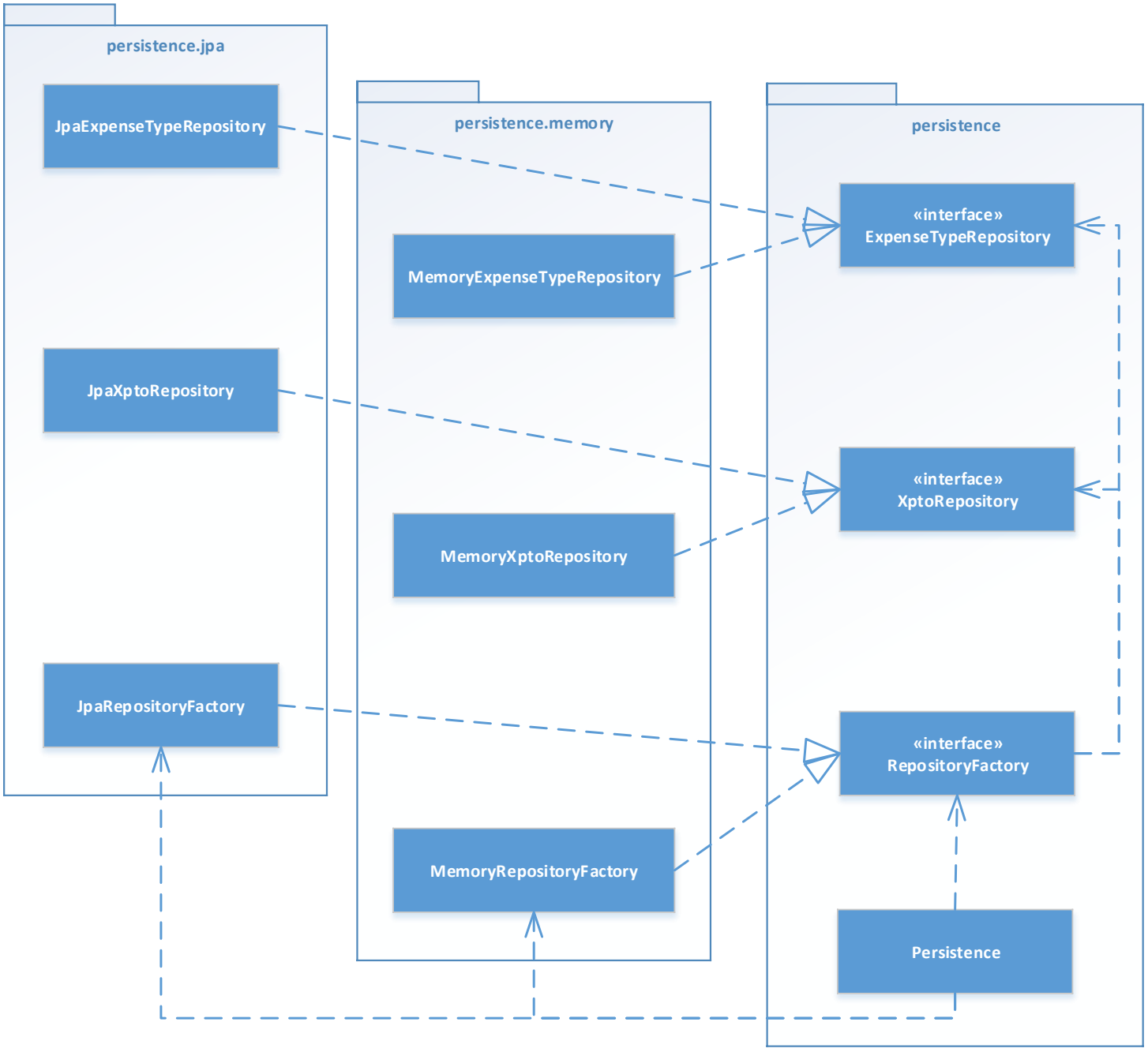
Persistence

- Controller needs to access the repositories
- But we want to have two repository implementations
  - In memory
  - Relational database
- Hide persistence details from rest of the code
  - Interfaces
  - Factories

Persistence



- Separate the definition of repositories (core) from the actual implementation (persistence.impl)
- Apply "Abstract Factory" GoF pattern


persistence.jpa
JpaExpenseTypeRepository
JpaXptoRepository
JpaRepositoryFactory
persistence.memory
MemoryExpenseTypeRepository
MemoryXptoRepository
MemoryRepositoryFactory
persistence
«interface»
ExpenseTypeRepository
«interface»
XptoRepository
«interface»
RepositoryFactory
Persistence

Example

```
11
12  public interface MaterialRepository extends DataRepository<Material, Long> {
13
14      Material findByAcronym(String acronym);
15  }
16
```

```
10  public class InMemoryMaterialRepository extends InMemoryRepositoryWithLongPK<Material>
11          implements MaterialRepository {
12
13      @Override
14      public Material findByAcronym(String acronym) {
15          return matchOne(e -> e.id().equals(acronym));
16      }
17  }
18
```

```
9   class JpaMaterialRepository extends CafeteriaJpaRepositoryBase<Material, Long>
10          implements MaterialRepository {
11
12      @Override
13      public Material findByAcronym(String acronym) {
14          return matchOne("e.acronym=:acronym", "acronym", acronym);
15      }
16  }
17
```

Persistence Context

```
16  public class PersistenceContext {
17
18      private PersistenceContext() {
19      }
20
21      public static RepositoryFactory repositories() {
22          final String factoryClassName = Application.settings().getRepositoryFactory();
23          try {
24              return (RepositoryFactory) Class.forName(factoryClassName).newInstance();
25          } catch (ClassNotFoundException | IllegalAccessException | InstantiationException ex) {
26              // FIXME handle exception properly
27              Logger.getLogger(PersistenceContext.class.getName()).log(Level.SEVERE, null, ex);
28              return null;
29          }
30      }
31  }
32
```

Persistence Context Usage

```
public class RegisterMaterialController implements Controller {

    private final MaterialRepository repository = PersistenceContext.repositories().materials();

    public Material registerMaterial(String acronym, String description)
            throws DataIntegrityViolationException, DataConcurrencyException {
        Application.ensurePermissionOfLoggedInUser(ActionRight.MANAGE_KITCHEN);

        final Material mat = new Material(acronym, description);
        return this.repository.save(mat);
    }
}
```

JPA Repositories (framework)

- JpaBaseRepository
  - Generic repository implementation that expects the entity manager factory to be injected by a container, e.g., web server
- JpaNotRunningInContainerBaseRepository
  - For scenarios where the code is not running in a container but transaction is managed by the outside, e.g., controller
- JpaTransactionalBaseRepository
  - For scenarios not running in a container but transactions are created and committed by each repository method; the connection is also closed automatically in each method.
- JpaAutoTxRepository
  - Dual behaviour to either have outside transactional control or explicit transaction in each method

Full transaction control by the repository

```
9   class JpaMaterialRepository extends CafeteriaJpaRepositoryBase<Material, Long>
            implements MaterialRepository {
11
12      @Override
        public Material findByAcronym(String acronym) {
14          return matchOne("e.acronym=:acronym", "acronym", acronym);
15      }
16  }
17
```

```
    class CafeteriaJpaRepositoryBase<T, K extends Serializable>
            extends JpaTransactionalRepository<T, K> {
17
18      CafeteriaJpaRepositoryBase(String persistenceUnitName) {
19          super(persistenceUnitName);
20      }
21
22      CafeteriaJpaRepositoryBase() {
23          super(Application.settings().getPersistenceUnitName());
24      }
25  }
26
```

Transaction control (1)

- Accepting a signup request needs to
  - Create a system user
  - Create a cafeteria user
  - Change the status of the signup request
- Three different aggregates!

Transaction control (2): use JpaAutoTxRepository

```
13  class JpaUserRepository extends JpaAutoTxRepository<SystemUser, Username>
        implements UserRepository {
15
16      public JpaUserRepository(boolean autoTx) {
17          super(Application.settings().getPersistenceUnitName(), autoTx);
18      }
19  }
```

```
14  class JpaCafeteriaUserRepository
15          extends JpaAutoTxRepository<CafeteriaUser, MecanographicNumber>
            implements CafeteriaUserRepository {
17
```

```
13      class JpaSignupRequestRepository
14              extends JpaAutoTxRepository<SignupRequest, Username>
15              implements SignupRequestRepository {
16
17          public JpaSignupRequestRepository(boolean autoTx) {
18              super(Application.settings().getPersistenceUnitName(), auto
19          }
20
21          @Override
            public Iterable<SignupRequest> pendingSignupRequests() {
23              return repo.match("e.approvalStatus=eapli.ecafeteria.domain
```

Transaction control (3): explicit control by the controller

```
37  public class AcceptRefuseSignupRequestController implements Controller {
38
39      private final UserRepository userRepository
40              = PersistenceContext.repositories().users(false);
41      private final CafeteriaUserRepository cafeteriaUserRepository
42              = PersistenceContext.repositories().cafeteriaUsers(false);
43      private final SignupRequestRepository signupRequestsRepository
            = PersistenceContext.repositories().signupRequests(false);
45
```

Transaction control (3): explicit control by the controller

```
46      public SignupRequest acceptSignupRequest(SignupRequest theSignupRequest)
47              throws DataIntegrityViolationException, DataConcurrencyException {
48          Application.ensurePermissionOfLoggedInUser(ActionRight.ADMINISTER);
49
50          if (theSignupRequest == null) {
51              throw new IllegalStateException();
52          }
53
54          // explicitly begin a transaction
55          userRepository.beginTransaction();
56
57          SystemUser newUser = createSystemUserForCafeteriaUser(theSignupRequest);
58          createCafeteriaUser(theSignupRequest, newUser);
59          theSignupRequest = acceptTheSignupRequest(theSignupRequest);
60
61          // explicitly commit the transaction
62          userRepository.commit();
63
64          return theSignupRequest;
65      }
66
```

Build

- Maven
  - Dependency manager
  - Artifact (jar) repository
  - Build automation
  - Other tasks, e.g., deploy, run
- Works for Eclipse, InteliJ, Netbeans
- Pom.xml

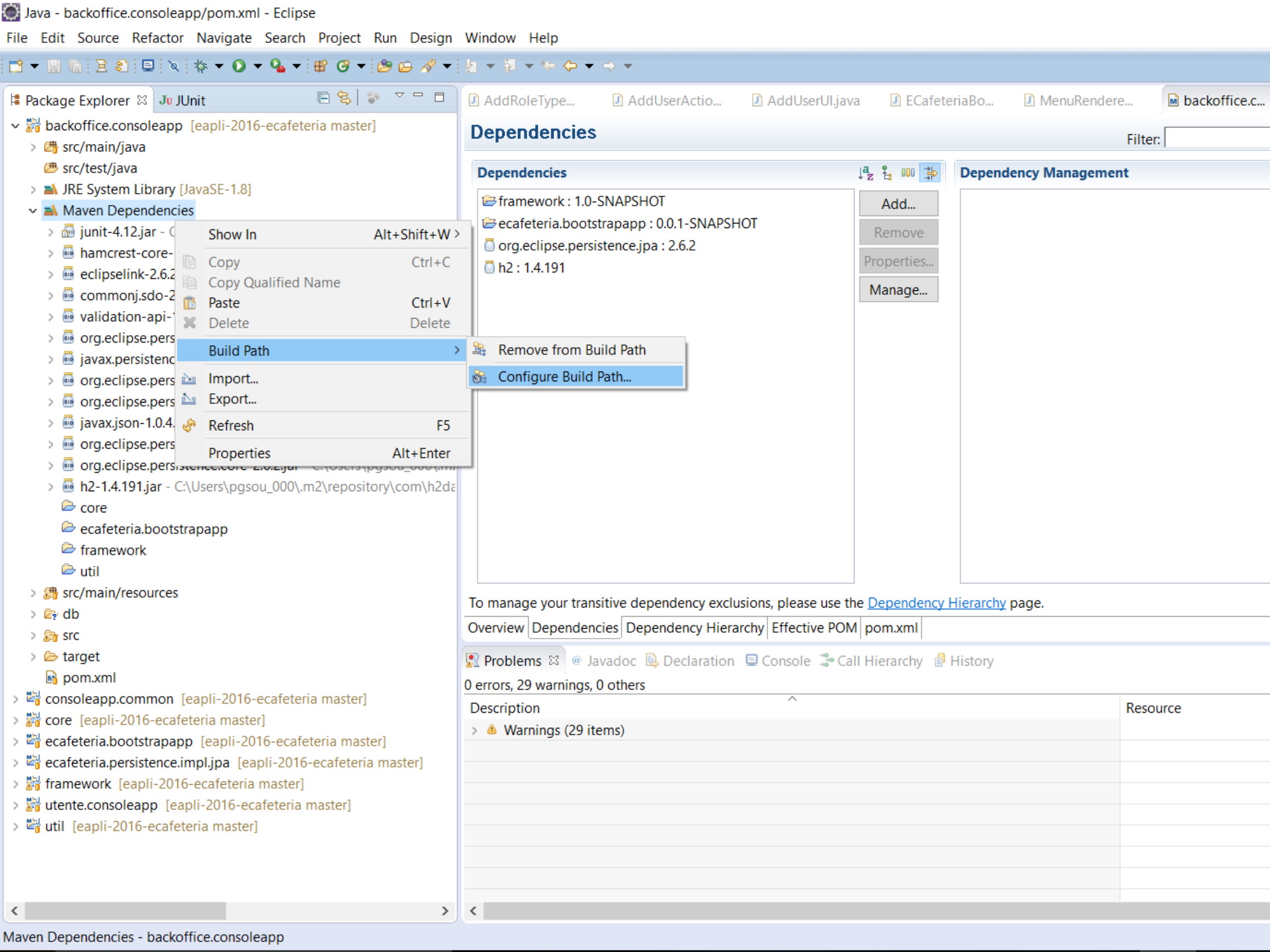
Java - backoffice.consoleapp/pom.xml - Eclipse

File Edit Source Navigate Search Project Run Window Help

AddRoleType... AddUserActio... AddUserUI.java ECafeteriaBo... ECafeteriaBa... ECafeteriaU... UsersBootst... MainMenu.java AbstractUI.ja

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3           xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <groupId>eapli</groupId>
6      <artifactId>ecafeteria.backoffice.consoleapp</artifactId>
7      <version>1.0-SNAPSHOT</version>
8      <packaging>jar</packaging>
9
10     <properties>
11         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
12         <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
13         <maven.compiler.source>1.8</maven.compiler.source>
14         <maven.compiler.target>1.8</maven.compiler.target>
15     </properties>
16
17     <name>ecafeteria.backoffice.consoleapp</name>
18
19     <dependencies>
20         <dependency>
21             <groupId>eapli</groupId>
22             <artifactId>framework</artifactId>
23             <version>1.0-SNAPSHOT</version>
24         </dependency>
25         <dependency>
26             <groupId>eapli</groupId>
27             <artifactId>ecafeteria.bootstrapapp</artifactId>
28             <version>0.0.1-SNAPSHOT</version>
29         </dependency>
30         <dependency>
31             <groupId>org.eclipse.persistence</groupId>
32             <artifactId>org.eclipse.persistence.jpa</artifactId>
33             <version>2.6.2</version>
34         </dependency>
35         <dependency>
36             <groupId>com.h2database</groupId>
37             <artifactId>h2</artifactId>
38             <version>1.4.191</version>
39         </dependency>
40     </dependencies>
41 </project>
```

Overview | Dependencies | Dependency Hierarchy | Effective POM | pom.xml


Java - backoffice.consoleapp/pom.xml - Eclipse
File Edit Source Refactor Navigate Search Project Run Design Window Help
Package Explorer
JUnit
backoffice.consoleapp [eapli-2016-ecafeteria master]
src/main/java
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
junit-4.12.jar
hamcrest-core-
eclipselink-2.6.2
commonj.sdo-2
validation-api-
org.eclipse.pers
javax.persistenc
org.eclipse.pers
org.eclipse.pers
javax.json-1.0.4.
org.eclipse.pers
org.eclipse.pers
h2-1.4.191.jar - C:\Users\pgsou_000\.m2\repository\com\h2da
core
ecafeteria.bootstrapapp
framework
util
src/main/resources
db
src
target
pom.xml
consoleapp.common [eapli-2016-ecafeteria master]
core [eapli-2016-ecafeteria master]
ecafeteria.bootstrapapp [eapli-2016-ecafeteria master]
ecafeteria.persistence.impl.jpa [eapli-2016-ecafeteria master]
framework [eapli-2016-ecafeteria master]
utente.consoleapp [eapli-2016-ecafeteria master]
util [eapli-2016-ecafeteria master]
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Build Path
Import...
Export...
Refresh F5
Properties Alt+Enter
Remove from Build Path
Configure Build Path...
AddRoleType...
AddUserActio...
AddUserUI.java
ECafeteriaBo...
MenuRendere...
backoffice.c...
Dependencies
Filter:
Dependencies
framework : 1.0-SNAPSHOT
ecafeteria.bootstrapapp : 0.0.1-SNAPSHOT
org.eclipse.persistence.jpa : 2.6.2
h2 : 1.4.191
Add...
Remove
Properties...
Manage...
Dependency Management
To manage your transitive dependency exclusions, please use the Dependency Hierarchy page.
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Problems
Javadoc
Declaration
Console
Call Hierarchy
History
0 errors, 29 warnings, 0 others
Description
Resource
Warnings (29 items)
Maven Dependencies - backoffice.consoleapp

Use cases implemented in base project

- Add user
- List users
- Deactivate user
- Check permissions
- Signup
- Approve new user
- Add dish type
- Edit dish type
- Deactivate dish type
- List dish types
- Register organic unit
- Add dish
- Add Material

Next steps

1. Read project description
2. Discuss and clear assumptions in PL
3. Clone class' repository
   - One for each PL class
4. Study base code
5. Analyse – design – code – test – document

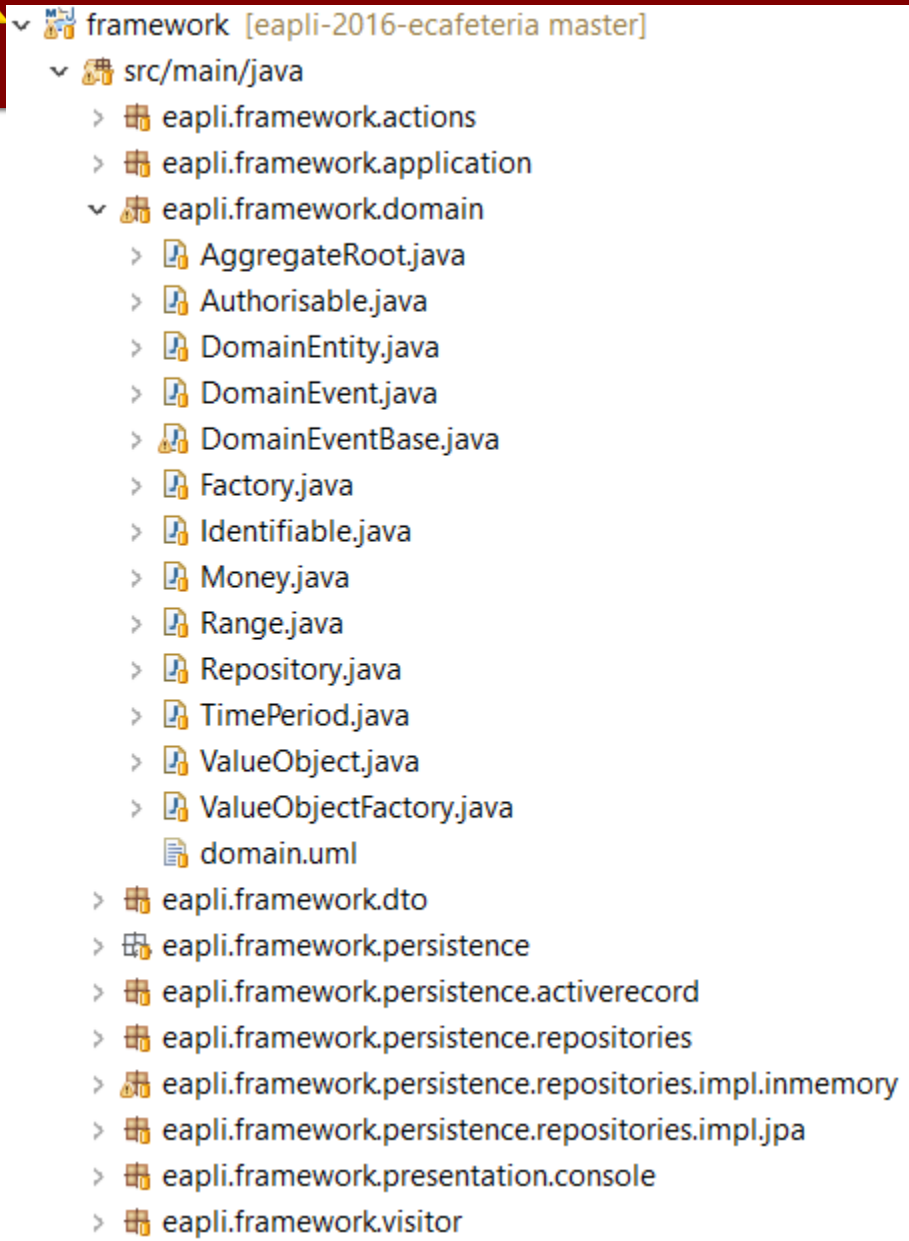
EAPLI Framework

Eapli.Util

- Console
  - Helper console reading functions
- DateTime
  - Simplifies manipulation of dates and times thru java Calendar
- Files
  - File manipulation helper
- Math
  - Sample math utility
- RomanNumeral
  - Represents a decimal number as a Roman numeral
- Strings
  - String manipulation

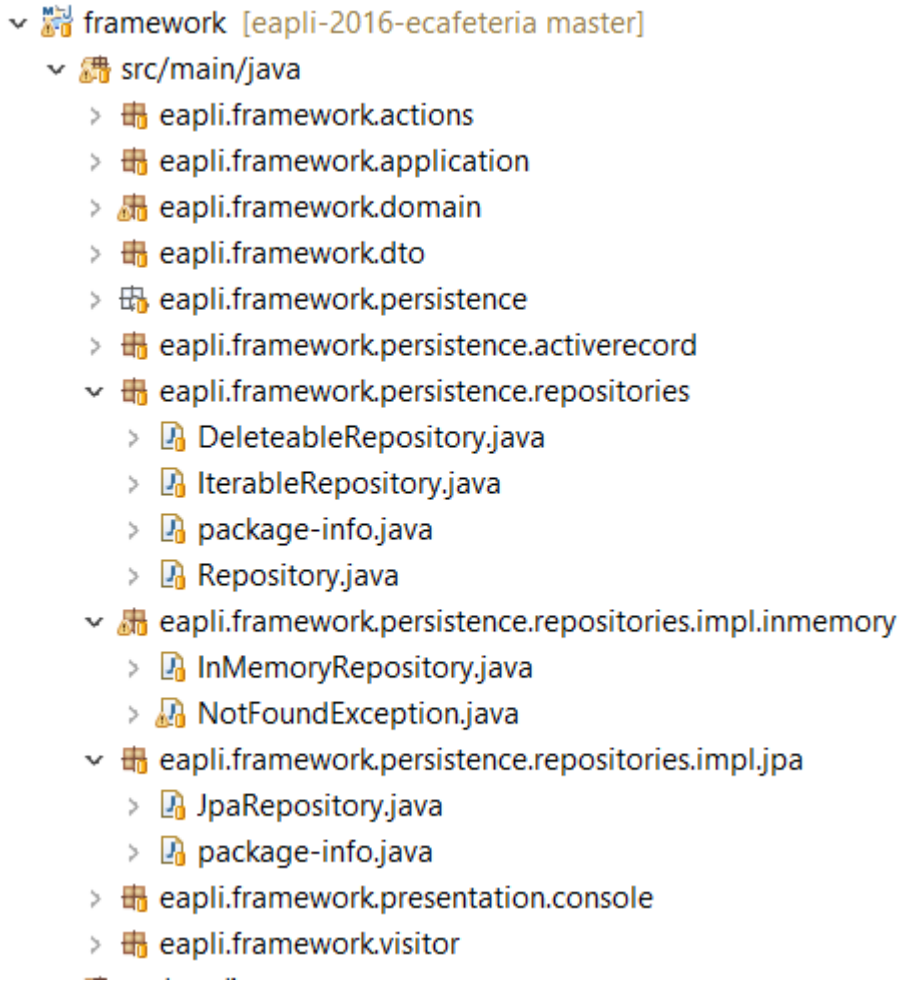
Eapli.Framework

- Domain objects
  - Money
  - Range
  - Time period
- DDD pattern interfaces
  - ValueObject
  - DomainEntity
  - AggregateRoot



Eapli.Framework

- Persistence
  - Repository interfaces
- Implementations
  - JPA
  - InMemory list



Repositories

- DataRepository
- TransactionalContext

- Interfaces for describing repository functionalities and transactions

In memory Repositories

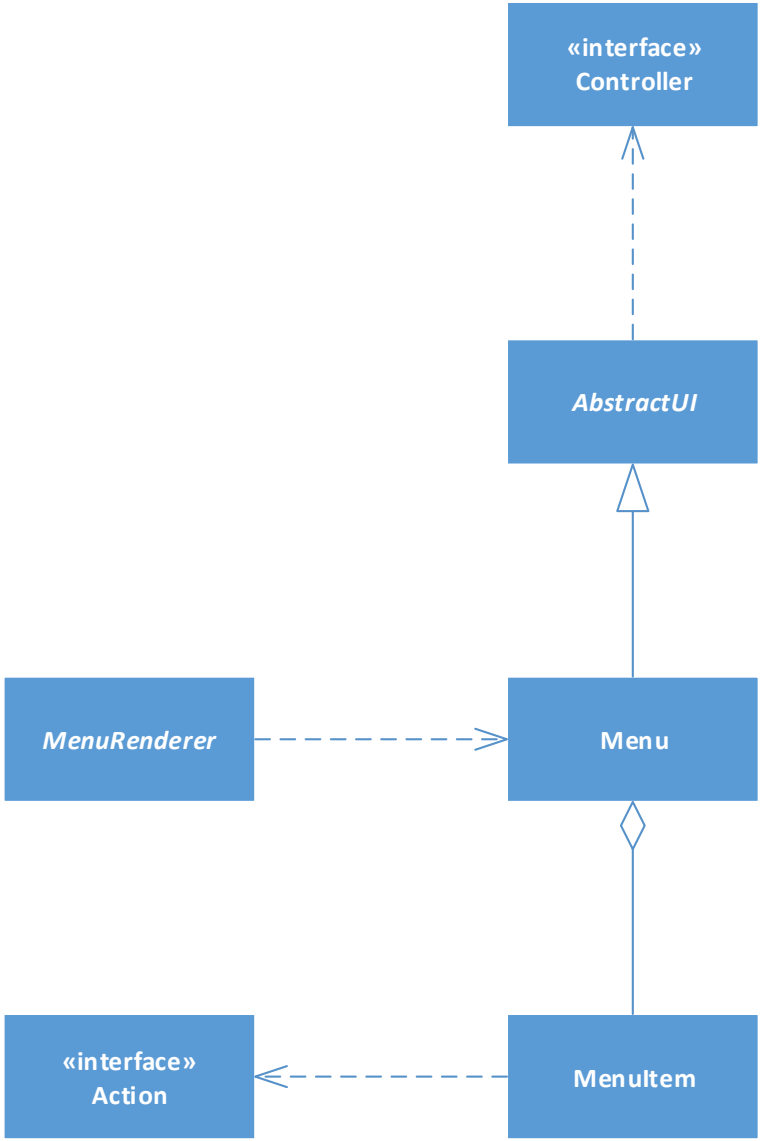
- InMemoryBaseRepository
- InMemoryBaseRepositorywithLongPK

- For pedagogical testing purposes only!

JPA Repositories

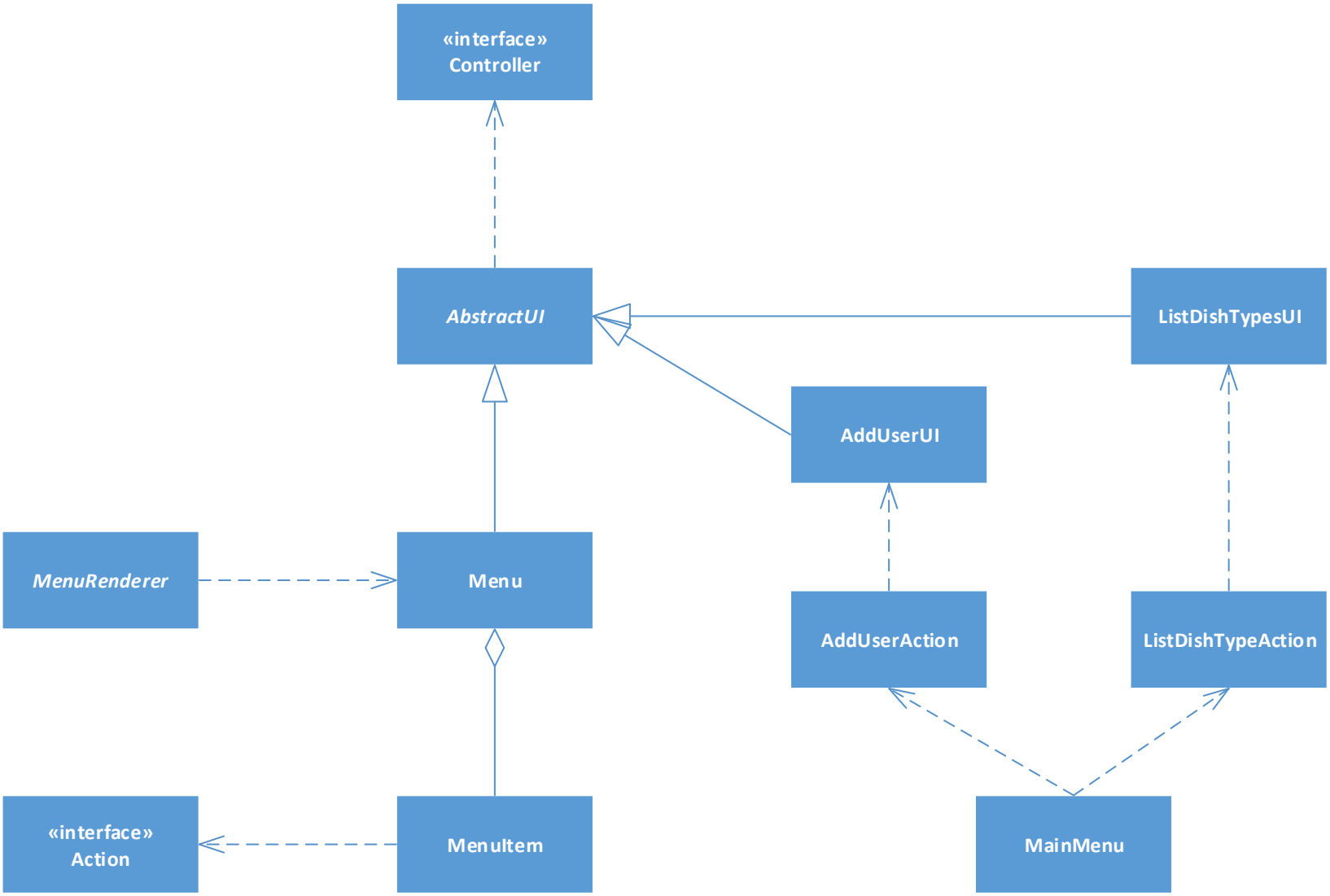
- JpaBaseRepository
  - Generic repository implementation that expects the entity manager factory to be injected by a container, e.g., web server
- JpaNotRunningInContainerBaseRepository
  - For scenarios where the code is not running in a container but transaction is managed by the outside, e.g., controller
- JpaTransactionalBaseRepository
  - For scenarios not running in a container but transactions are created and committed by each repository method; the connection is also closed automatically in each method.
- JpaAutoTxRepository
  - Dual behaviour to either have outside transactional control or explicit transaction in each method

Presentation



- src/main/java
  - eapli.framework.actions
    - Action.java
    - CompoundAction.java
    - ExitAction.java
    - IfThenAction.java
    - NullAction.java
    - ReturnAction.java
    - ShowMessageAction.java
  - eapli.framework.application
  - eapli.framework.domain
  - eapli.framework.dto
  - eapli.framework.persistence
  - eapli.framework.persistence.activerecord
  - eapli.framework.persistence.repositories
  - eapli.framework.persistence.repositories.impl.inmemory
  - eapli.framework.persistence.repositories.impl.jpa
  - eapli.framework.presentation.console
    - AbstractUI.java
    - HorizontalMenuRenderer.java
    - ListWidget.java
    - Menu.java
    - MenuItem.java
    - MenuRenderer.java
    - SelectWidget.java
    - ShowUiAction.java
    - ShowVerticalSubMenuAction.java
    - SubMenu.java
    - VerticalMenuRenderer.java
    - VerticalSeparator.java
  - eapli.framework.visitor

presentation



Java - framework/src/main/java/eapli/framework/presentation/console/AbstractUI.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

AddRoleType2List.java | AddUserAction.java | AddUserUI.java | ECafeteriaBootstrap.java | ECafeteriaBackoffice.java | ECafeteriaUtenteApp.java | UsersBootstrap.jav

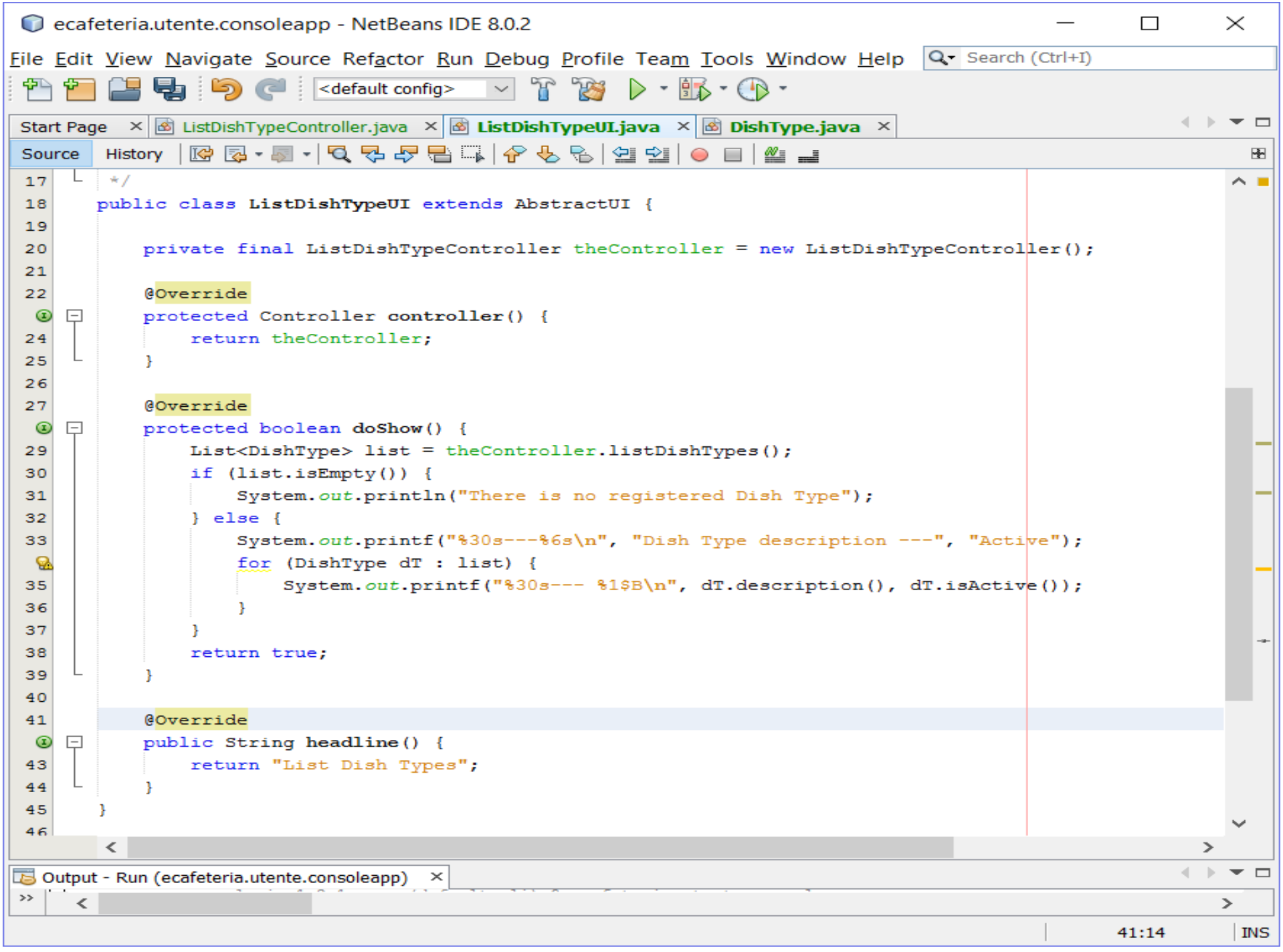
```
 */
public abstract class AbstractUI {

    public static final String SEPARATOR = "+--------------------------------------------------------------------------------+";
    public static final String BORDER    = "+================================================================================+";

     * derived classes should provide the Controller object. an example of the
    protected abstract Controller controller();

    /**
     * derived classes should override this method to perform the actual
     * rendering of the UI. follows the Template Method pattern
     *
     * @return true if the user wants to leave this UI
     */
    protected abstract boolean doShow();

     * derived classes should override this method to provide the title of the
    public abstract String headline();

    public void mainLoop() {
        boolean wantsToExit;
        do {
            wantsToExit = show();
        } while (!wantsToExit);
    }

    /**
     *
     * @return true if the user wants to leave this UI
     */
    public boolean show() {
        drawFormTitle();
        final boolean wantsToExit = doShow();
        drawFormBorder();
        // Console.waitForKey("Press any key.");

        return wantsToExit;
    }

    protected void drawFormTitle() {
        System.out.println();
```

Java - backoffice.consoleapp/src/main/java/eapli/ecafeteria/backoffice/consoleapp/presentation/MainMenu.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

AddRoleType2List.java | AddUserAction.java | AddUserUI.java | ECafeteriaBootstrap.java | ECafeteriaBackoffice.java | ECafeteriaUtenteApp.java | UsersBoots

```
21  */
22  public class MainMenu extends AbstractUI {
23
24      /**
25       * @return true if the user selected the exit option
26       */
27      @Override
28      public boolean doShow() {
29          final Menu menu = buildMainMenu();
30          final MenuRenderer renderer = new VerticalMenuRenderer(menu);
31          return renderer.show();
32      }
33
34      @Override
35      public String headline() {
36          return "eCAFETERIA [@" + AppSettings.instance().session().authenticatedUser().id() + "]";
37      }
38
39      private Menu buildMyUserMenu() {
40          final Menu myUserMenu = new Menu("My account >");
41
42          myUserMenu.add(
43                  new MenuItem(CHANGE_PASSWORD_OPTION, "Change password", new ShowMessageAction("Not implemented yet")));
44          myUserMenu.add(new MenuItem(LOGIN_OPTION, "Change user (Login)", new LoginAction()));
45          myUserMenu.add(new MenuItem(LOGOUT_OPTION, "Logout", new LogoutAction()));
46
47          return myUserMenu;
48      }
49
50      private Menu buildMainMenu() {
51          final Menu mainMenu = new Menu();
52
53          final Menu myUserMenu = buildMyUserMenu();
54          mainMenu.add(new SubMenu(MY_USER_OPTION, myUserMenu, new ShowVerticalSubMenuAction(myUserMenu)));
55
56          mainMenu.add(new VerticalSeparator());
57
58          if (AppSettings.instance().session().authenticatedUser().isAuthorizedTo(ActionRight.Administer)) {
59              final Menu usersMenu = buildUsersMenu();
60              mainMenu.add(new SubMenu(USERS_OPTION, usersMenu, new ShowVerticalSubMenuAction(usersMenu)));
61
62              final Menu organicUnitsMenu = buildOrganicUnitsMenu();
63              mainMenu.add(new SubMenu(ORGANIC_UNITS_OPTION, organicUnitsMenu,
```


ecafeteria.utente.consoleapp - NetBeans IDE 8.0.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Start Page
ListDishTypeController.java
ListDishTypeUI.java
DishType.java
Source
History
17 */
18 public class ListDishTypeUI extends AbstractUI {
19
20 private final ListDishTypeController theController = new ListDishTypeController();
21
22 @Override
protected Controller controller() {
24 return theController;
25 }
26
27 @Override
protected boolean doShow() {
29 List<DishType> list = theController.listDishTypes();
30 if (list.isEmpty()) {
31 System.out.println("There is no registered Dish Type");
32 } else {
33 System.out.printf("%30s---%6s\n", "Dish Type description ---", "Active");
for (DishType dT : list) {
35 System.out.printf("%30s--- %1$B\n", dT.description(), dT.isActive());
36 }
37 }
38 return true;
39 }
40
41 @Override
public String headline() {
43 return "List Dish Types";
44 }
45 }
46
Output - Run (ecafeteria.utente.consoleapp)
41:14
INS