

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

pag\_isep / EAPLI-201 x

Atlassian, Inc. [US] https://bitbucket.org/pag\_isep/eapli-2016-ecafeteria/src

BitbucketTeamsProjectsRepositoriesSnippets



EAPLI-2016-eCafeteria

ACTIONS

Clone

Create branch

Create pull request

Compare

Fork

NAVIGATION

Overview

Source

Commits

Branches

Pull requests

Downloads

Settings

Paulo Gandra de Sousa / EAPLI-2016-eCafeteria

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

=====

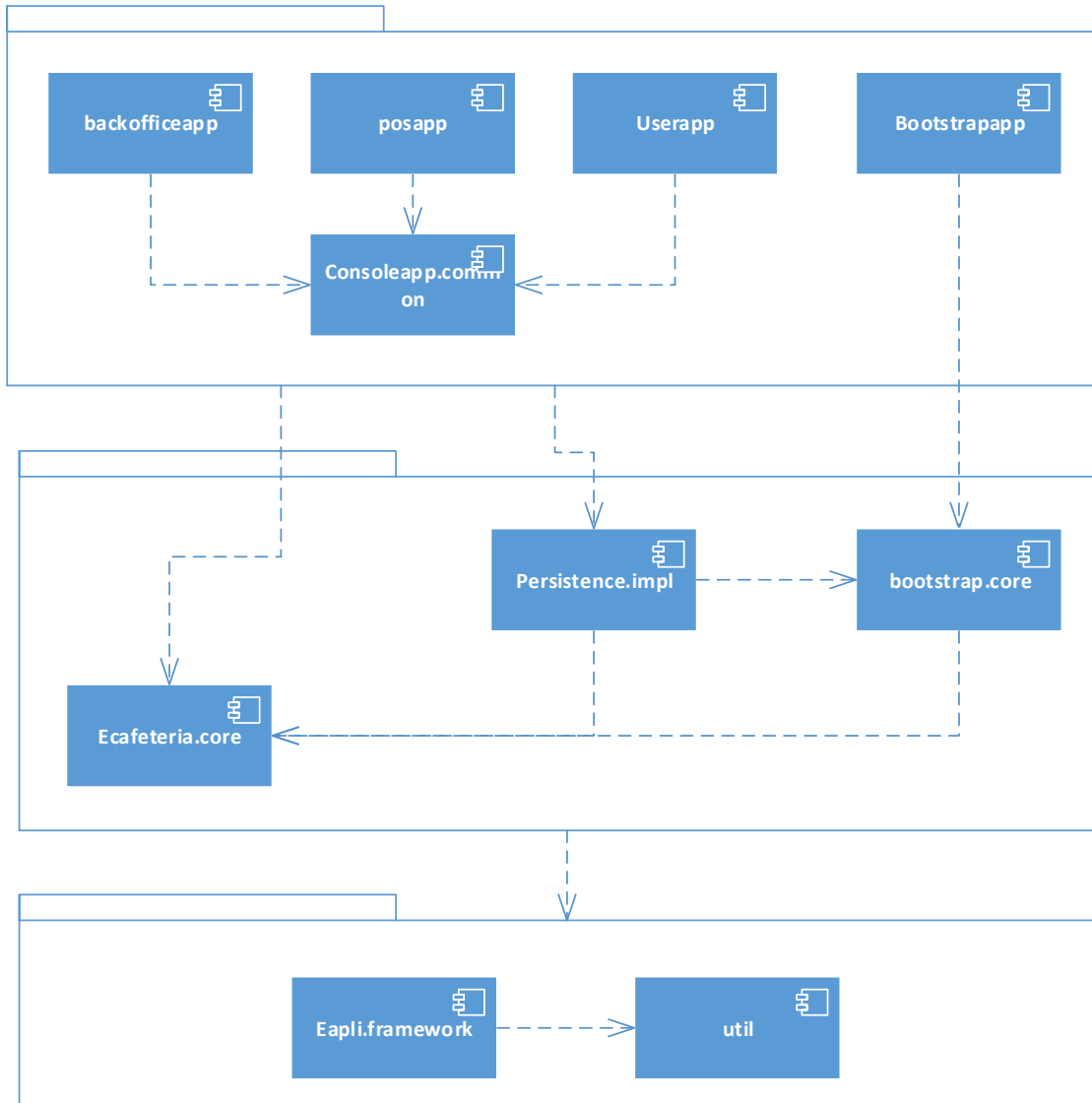
This application is part of the lab project for the course EAPLI.

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

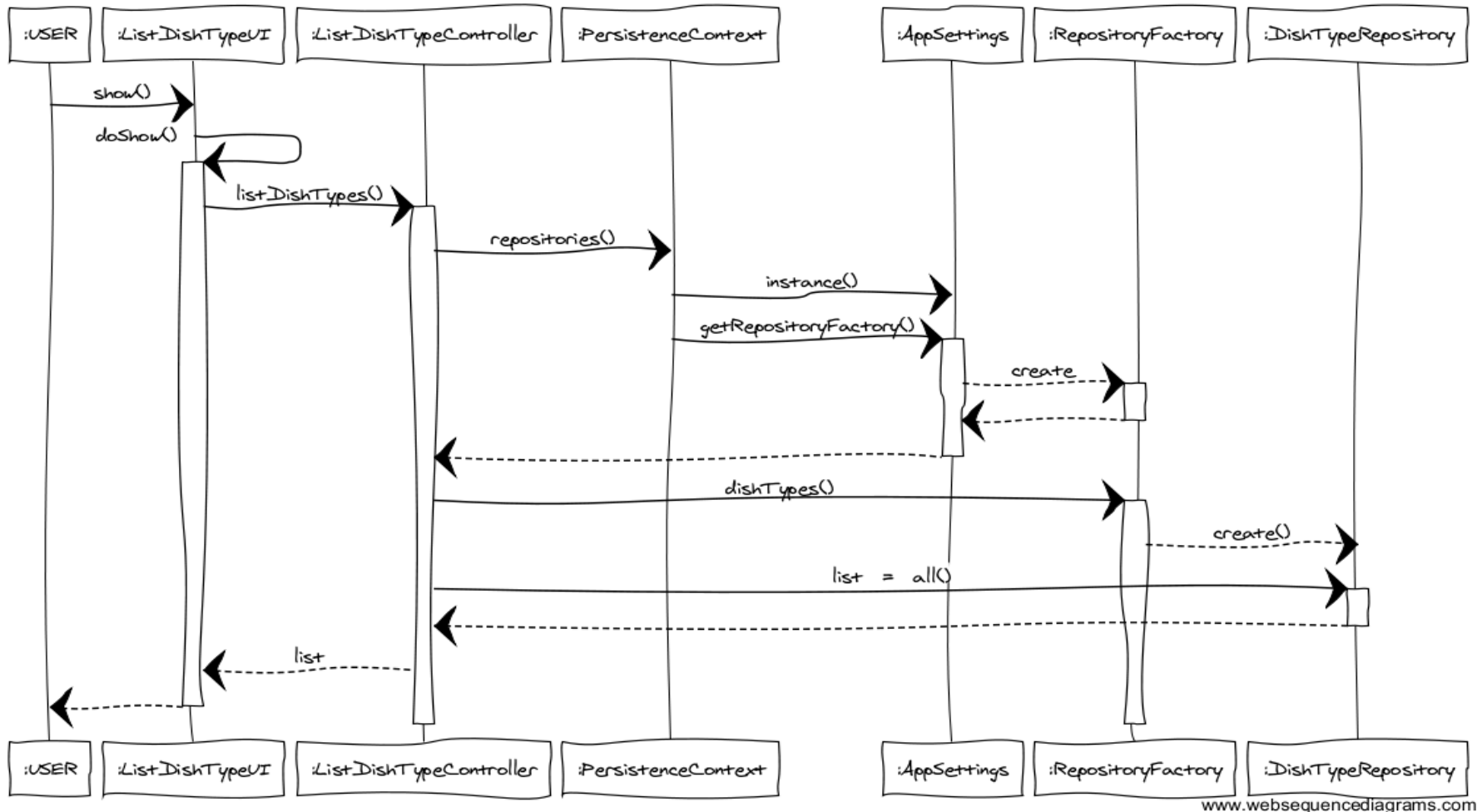
# eCafeteria design decisions

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

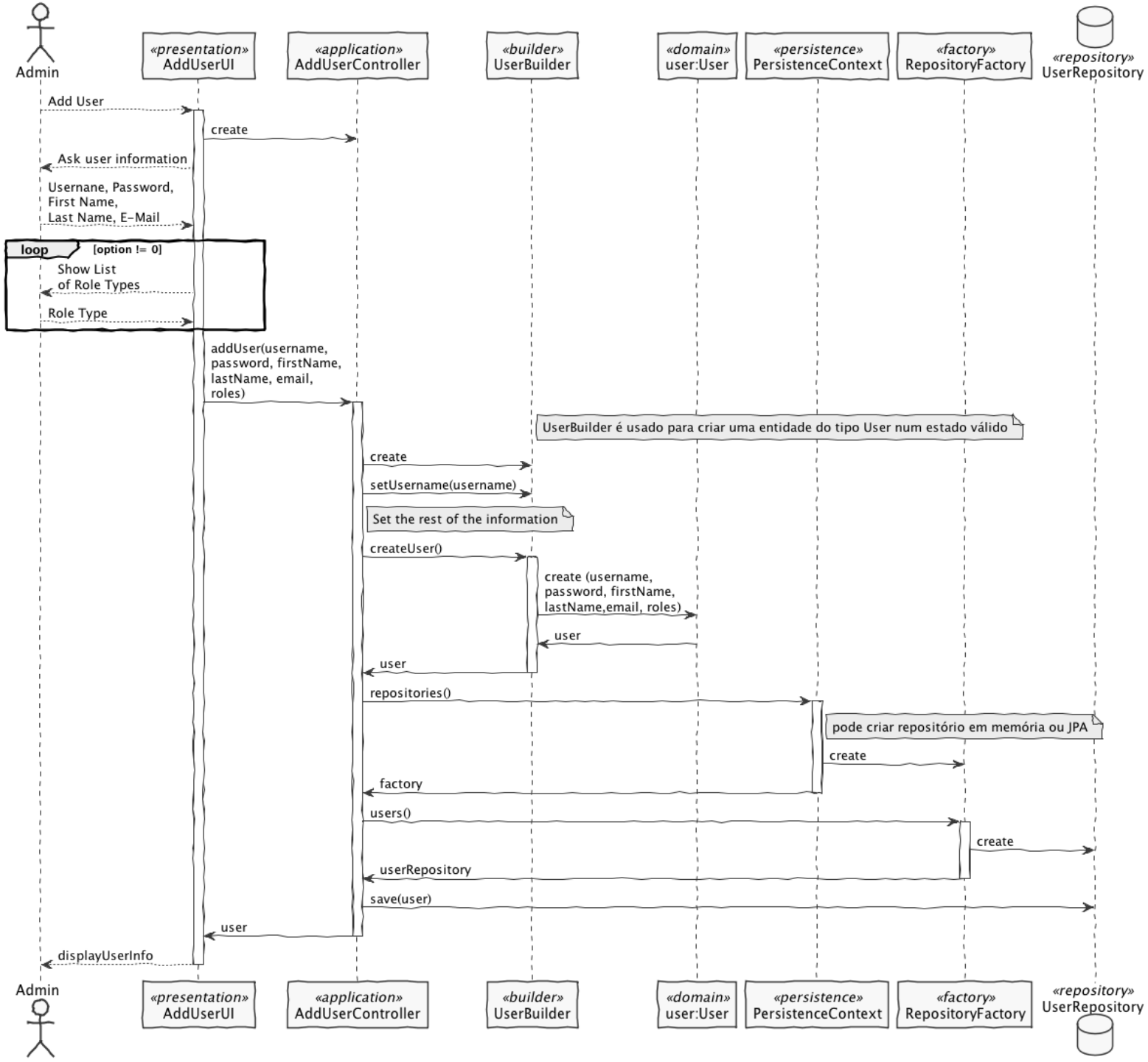
More on this  
next lesson

# List Dish Types

SD - List All Dish Types

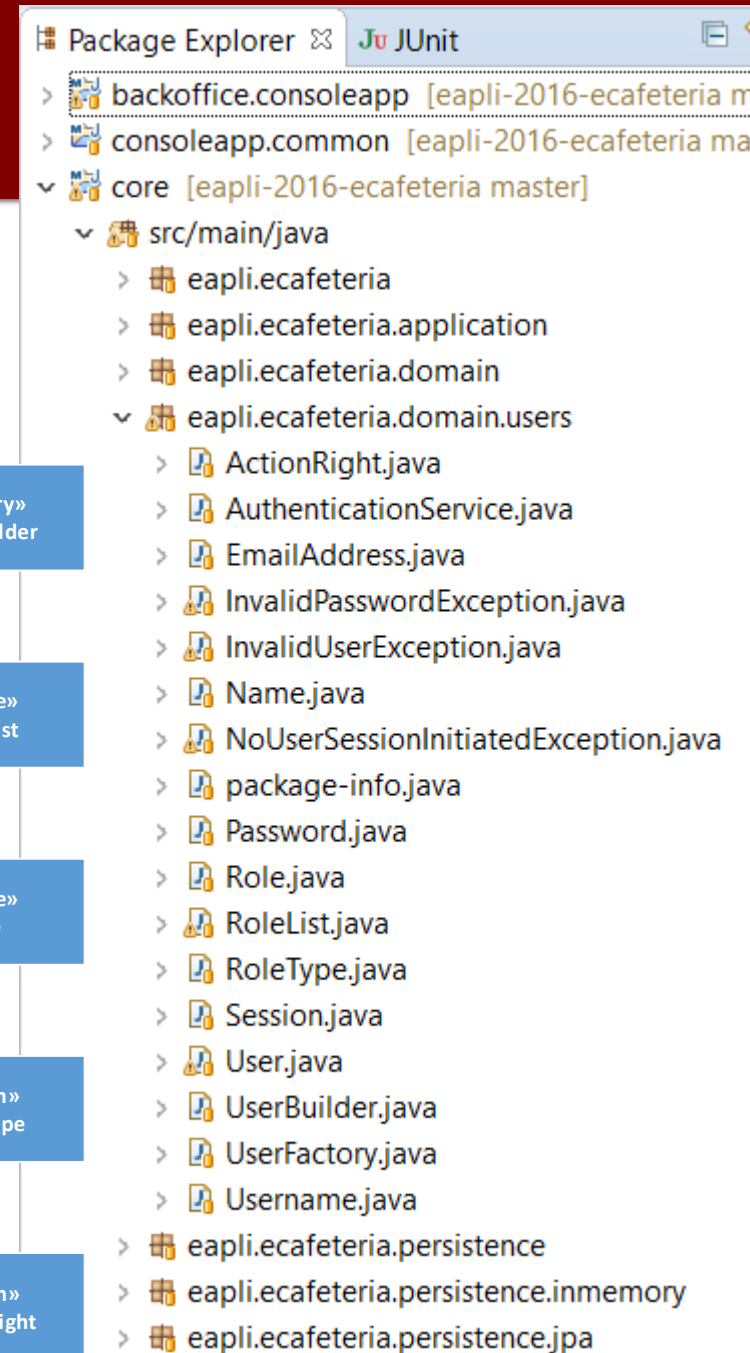
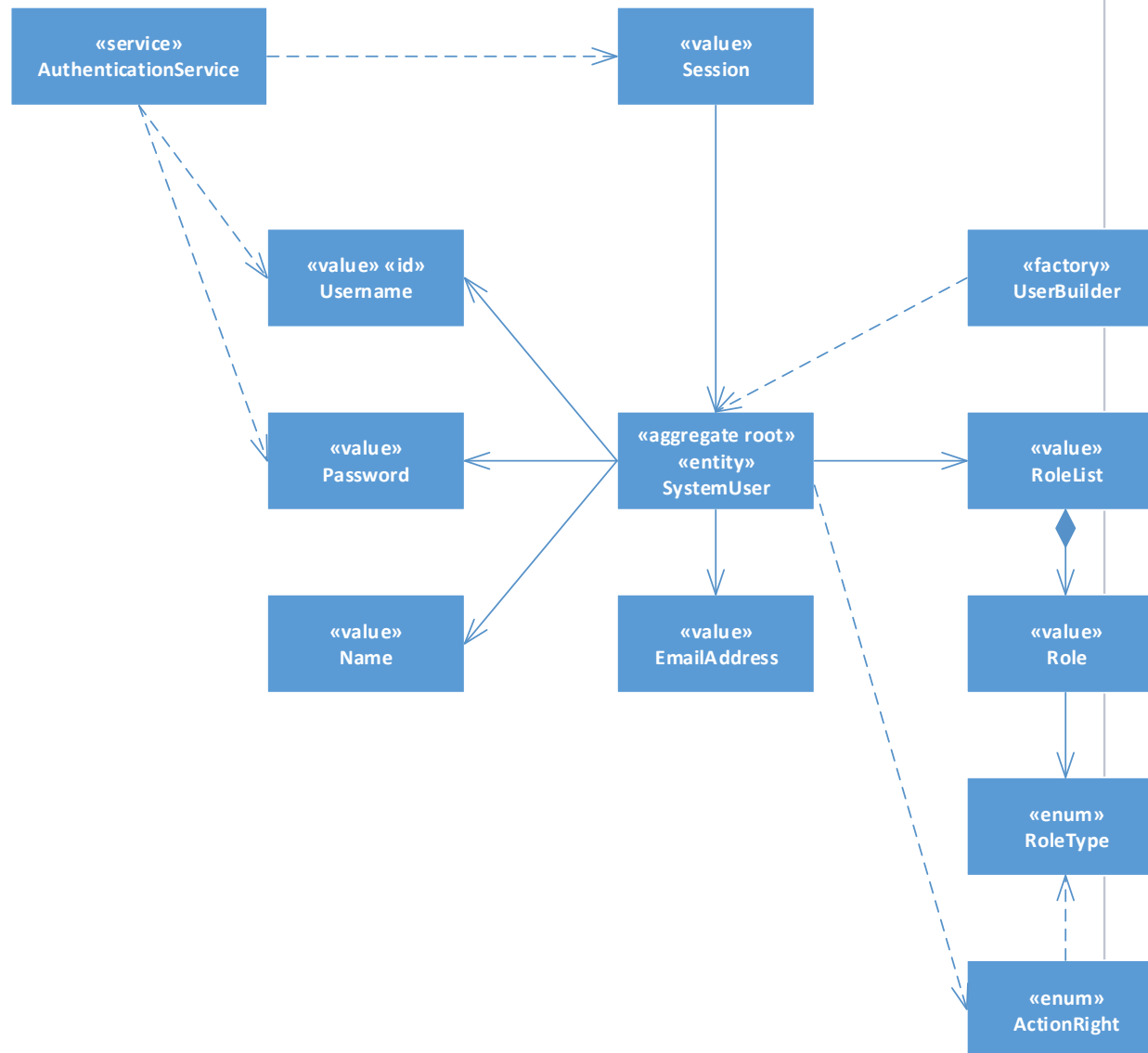


# Register New User





# User model



# Embeddable vs regular fields

```
@Entity
class SystemUser
{
```

```
    @Id
    String username;
    String password;
    ...
}
```

VS

```
@Entity
class SystemUser
{
```

```
    @EmbeddedId
    Username username;
    String Password password;
    ...
}
```

```
@Embeddable
class Username
{
```

```
    String username;
```

```
    {
        @Embeddable
        class Password
        {
            String password;
        }
    }
}
```

PK		
username	password	- - -

# 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;  
    }  
}
```

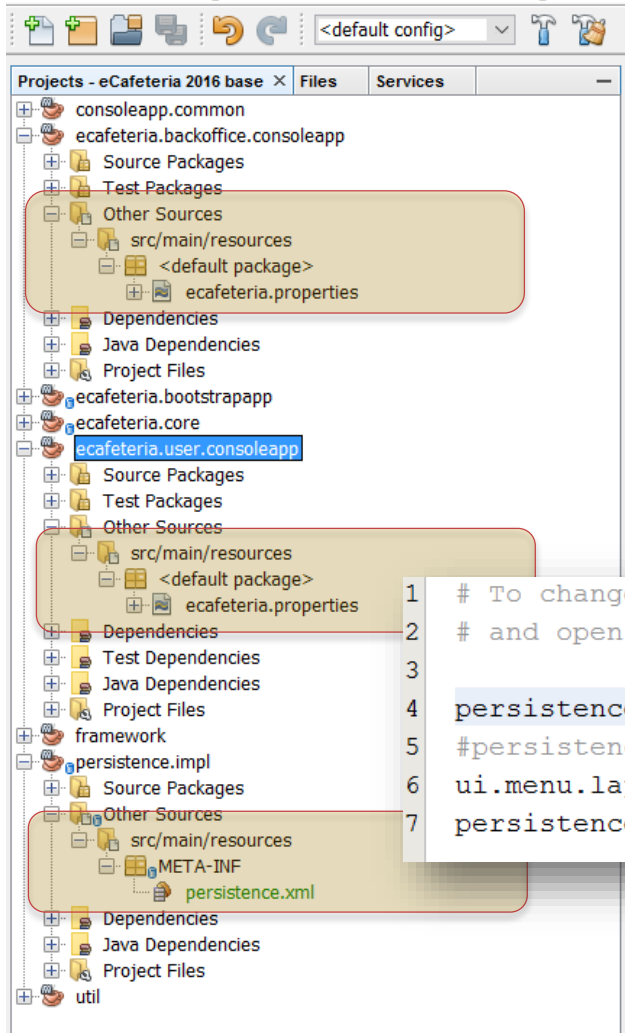
# Some additional design decisions

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

# Resources

ecafeteria.user.consoleapp - NetBeans IDE 8.1

File Edit View Navigate Source Refactor Run Debug Profile



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

```
1 # To change this template, choose Tools | Templates
2 # and open the template in the editor.
3
4 persistence.repositoryFactory=eapli.ecafeteria.persistence.jpa.JpaRepositoryFactory
5 #persistence.repositoryFactory=eapli.ecafeteria.persistence.inmemory.InMemoryRepository
6 ui.menu.layout=horizontal
7 persistence.persistenceUnit=eapli.eCafeteriaPU
```

# 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@email.l.com";
        final List<RoleType> roles = new ArrayList<RoleType>();
        roles.add(RoleType.Admin);

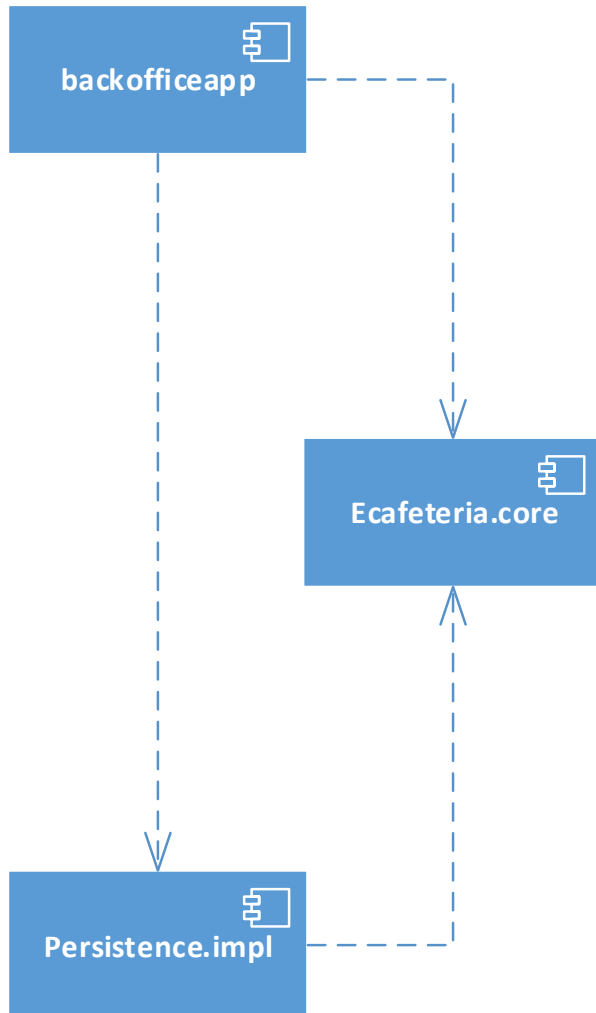
        final UserRegisterController userController = new UserRegisterController();
        userController.registerUser(username, password, firstName, lastName, email, roles);
    }
}
```

# Persistence

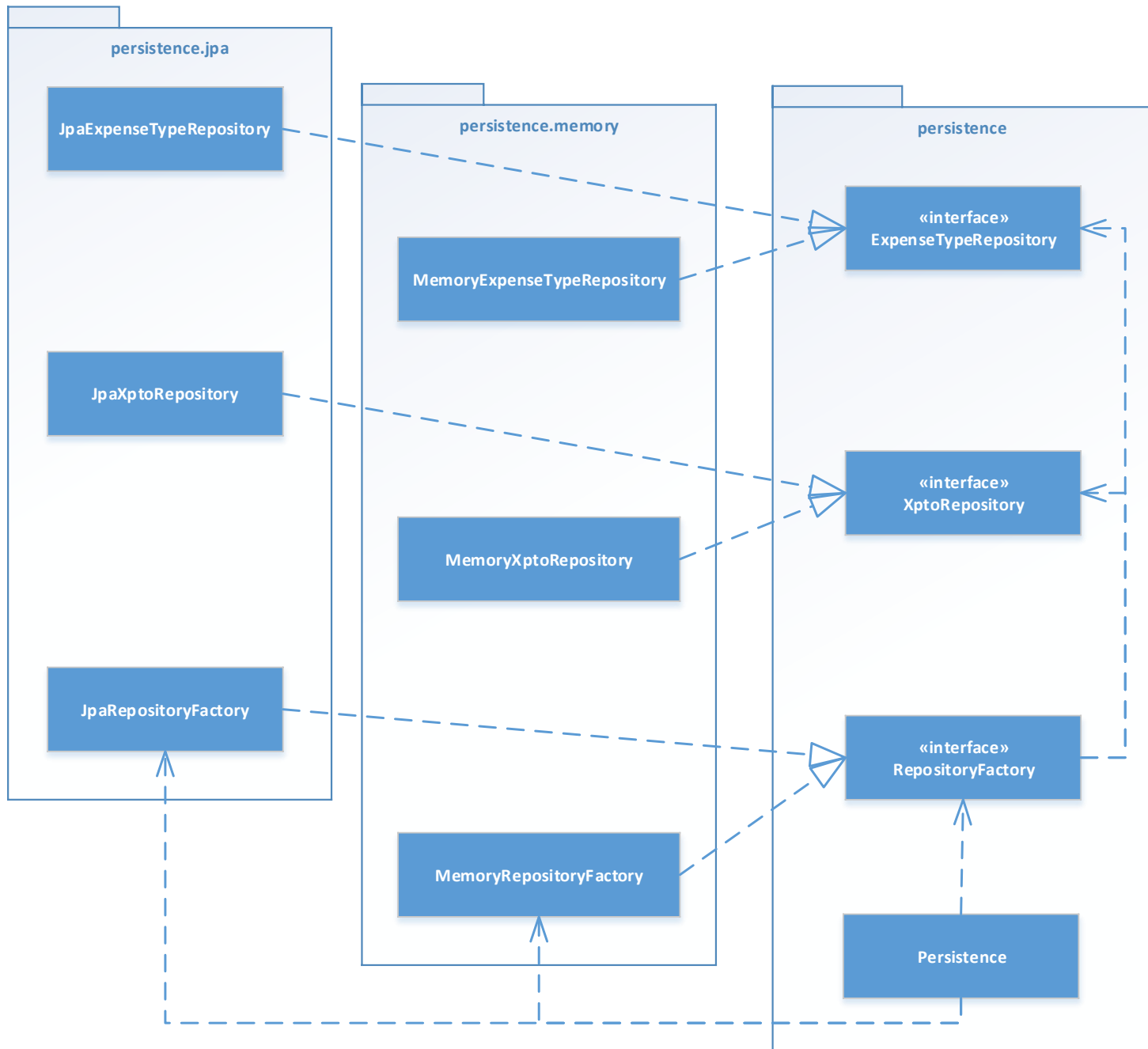
- Controller needs to access the repositories
- But we want to have two repository implementations
  - In memory
  - Relational database
- Hide persistence details from rest of the code
  - Interfaces
  - Factories



# Persistence



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



# Example

```
public interface MaterialRepository extends DataRepository<Material, Long> {  
    Material findByAcronym(String acronym);  
}
```

```
public class InMemoryMaterialRepository extends InMemoryRepositoryWithLongPK<Material>  
    implements MaterialRepository {  
  
    @Override  
    public Material findByAcronym(String acronym) {  
        return matchOne(e -> e.id().equals(acronym));  
    }  
}
```

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

# 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
```

```
1 # To change this template, choose Tools | Templates
2 # and open the template in the editor.
3
4 persistence.repositoryFactory=eapli.ecafeteria.persistence.jpa.JpaRepositoryFactory
5 #persistence.repositoryFactory=eapli.ecafeteria.persistence.inmemory.InMemoryRepositoryFactory
6 ui.menu.layout=horizontal
7 persistence.persistenceUnit=eapli.eCafeteriaPU
```

# 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>
10      implements MaterialRepository {
11
12      @Override
13      public Material findByAcronym(String acronym) {
14          return matchOne("e.acronym=:acronym", "acronym", acronym);
15      }
16  }
```

```
17  class CafeteriaJpaRepositoryBase<T, K extends Serializable>
18      extends JpaTransactionalRepository<T, K> {
19
20      CafeteriaJpaRepositoryBase(String persistenceUnitName) {
21          super(persistenceUnitName);
22      }
23
24      CafeteriaJpaRepositoryBase() {
25          super(Application.settings().getPersistenceUnitName());
26      }
27  }
```

# 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>
14     implements UserRepository {
15
16     public JpaUserRepository(boolean autoTx) {
17         super(Application.settings().getPersistenceUnitName(), autoTx);
18     }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
103
```

# 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
44         = 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, IntelliJ, Netbeans
- Pom.xml



AddRoleType... AddUserActio... AddUserUI.java ECafeteriaBo... ECafeteriaBa... ECafeteriaU... UsersBootst... MainMenu.java AbstractUIJa

```
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>
```



# 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



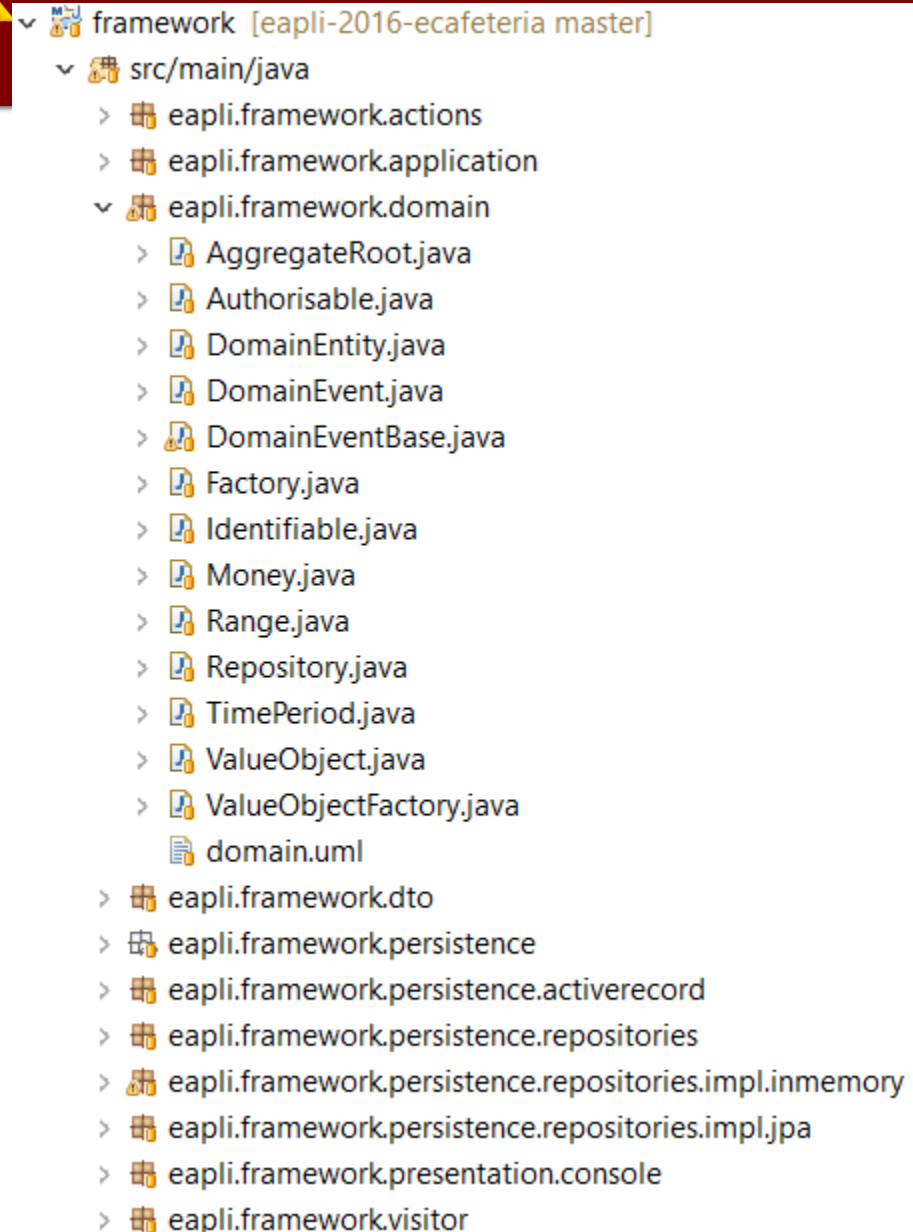
# EAPLI Framework

# 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

```
▼ framework [eapli-2016-ecafeteria master]
  ▼ src/main/java
    > eapli.framework.actions
    > eapli.framework.application
    > eapli.framework.domain
    > eapli.framework.dto
    > eapli.framework.persistence
    > eapli.framework.persistence.activerecord
    ▼ eapli.framework.persistence.repositories
      > DeleteableRepository.java
      > IterableRepository.java
      > package-info.java
      > Repository.java
    ▼ eapli.framework.persistence.repositories.impl.inmemory
      > InMemoryRepository.java
      > NotFoundException.java
    ▼ eapli.framework.persistence.repositories.impl.jpa
      > JpaRepository.java
      > package-info.java
    > eapli.framework.presentation.console
    > eapli.framework.visitor
```

# Repositories

- DataRepository
- TransactionalContext
- Interfaces for describing repository functionalities and transactions

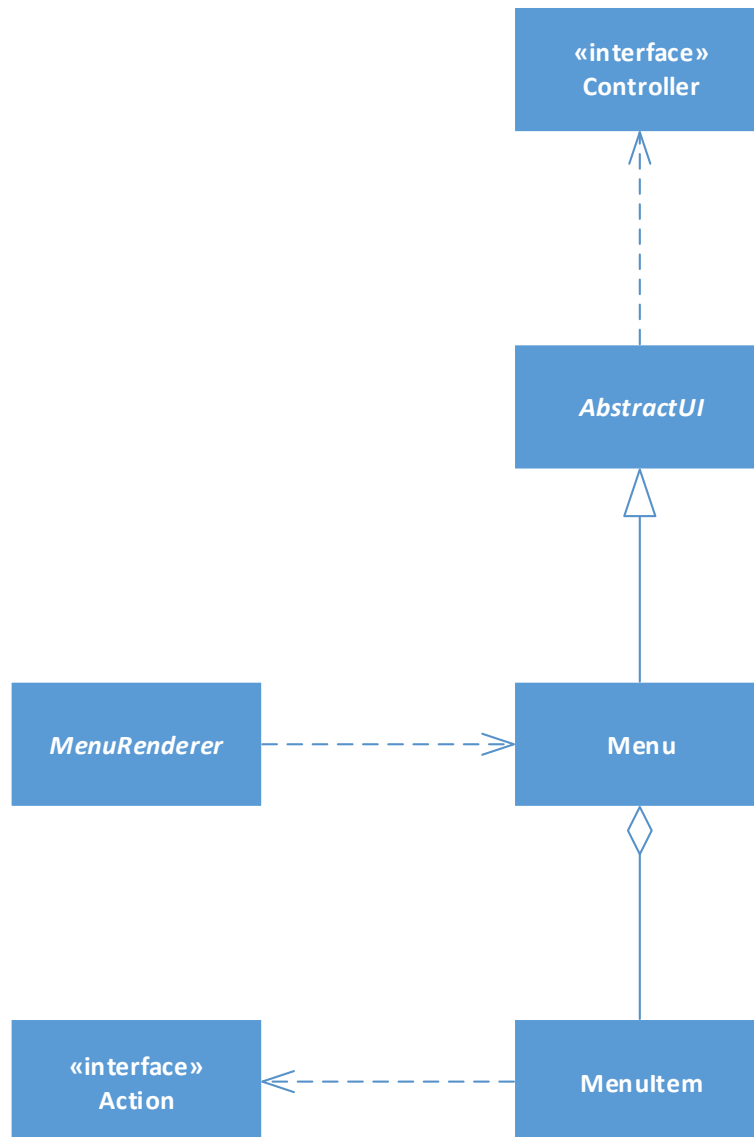
# In memory Repositories

- 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

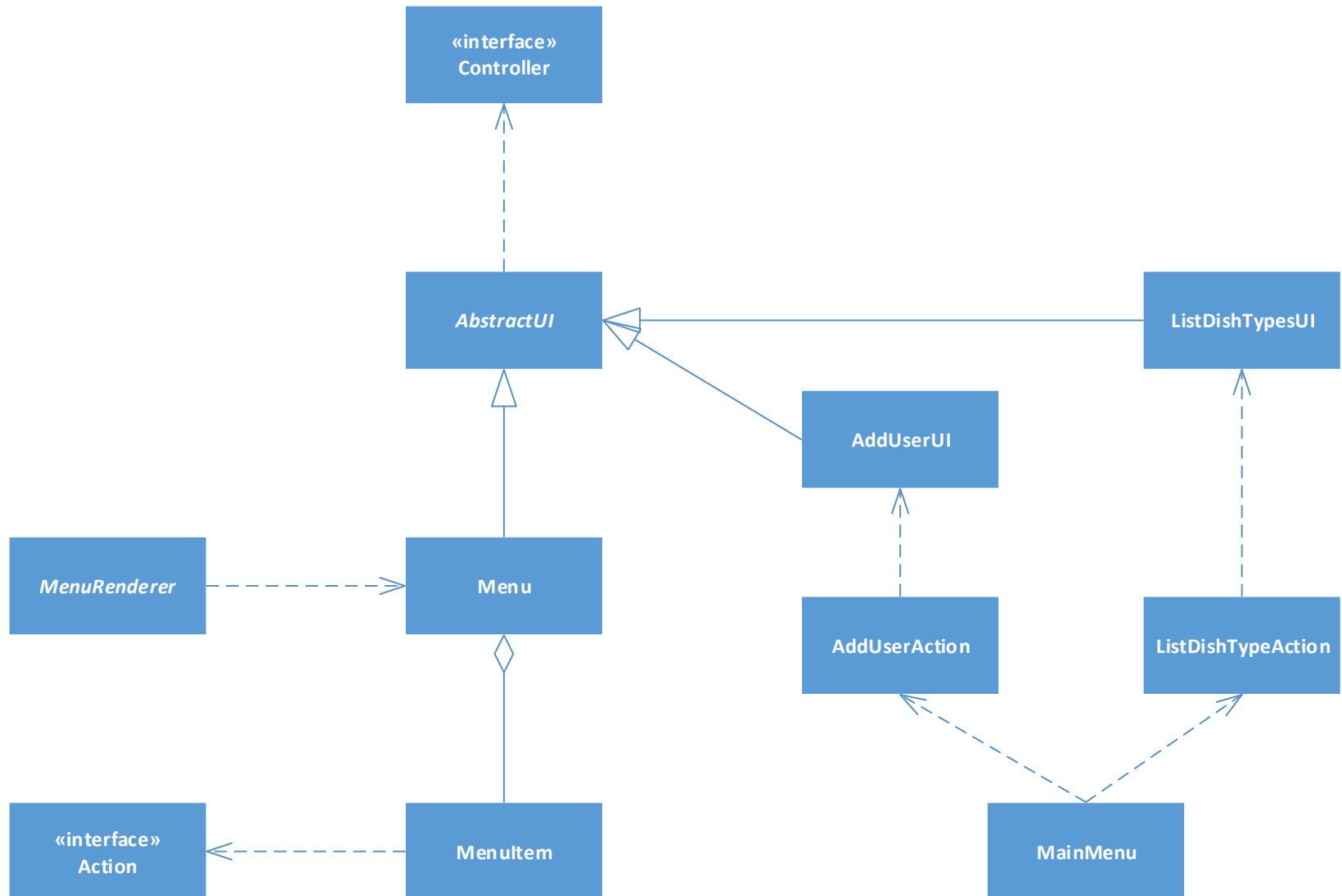
# Presentation



- 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



```
12  */
13  public abstract class AbstractUI {
14
15      public static final String SEPARATOR = "+-----+";
16      public static final String BORDER    = "+=====+";
17
18      * derived classes should provide the Controller object. an example of the
19      protected abstract Controller controller();
20
21      /**
22       * derived classes should override this method to perform the actual
23       * rendering of the UI. follows the Template Method pattern
24       *
25       * @return true if the user wants to leave this UI
26       */
27      protected abstract boolean doShow();
28
29      * derived classes should override this method to provide the title of the
30      public abstract String headline();
31
32      public void mainLoop() {
33          boolean wantsToExit;
34          do {
35              wantsToExit = show();
36          } while (!wantsToExit);
37      }
38
39      /**
40       *
41       * @return true if the user wants to leave this UI
42       */
43      public boolean show() {
44          drawFormTitle();
45          final boolean wantsToExit = doShow();
46          drawFormBorder();
47          // Console.waitForKey("Press any key.");
48
49          return wantsToExit;
50      }
51
52      protected void drawFormTitle() {
53          System.out.println();
54          drawFormTitle(headline());
55      }
56  }
```



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,
```

Start Page x ListDishTypeController.java x ListDishTypeUI.java x DishType.java x

Source History

```
17  */
18  public class ListDishTypeUI extends AbstractUI {
19
20      private final ListDishTypeController theController = new ListDishTypeController();
21
22      @Override
23      protected Controller controller() {
24          return theController;
25      }
26
27      @Override
28      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");
34              for (DishType dT : list) {
35                  System.out.printf("%30s--- %1sB\n", dT.description(), dT.isActive());
36              }
37          }
38          return true;
39      }
40
41      @Override
42      public String headline() {
43          return "List Dish Types";
44      }
45  }
46
```

Output - Run (ecafeteria.utente.consoleapp) x