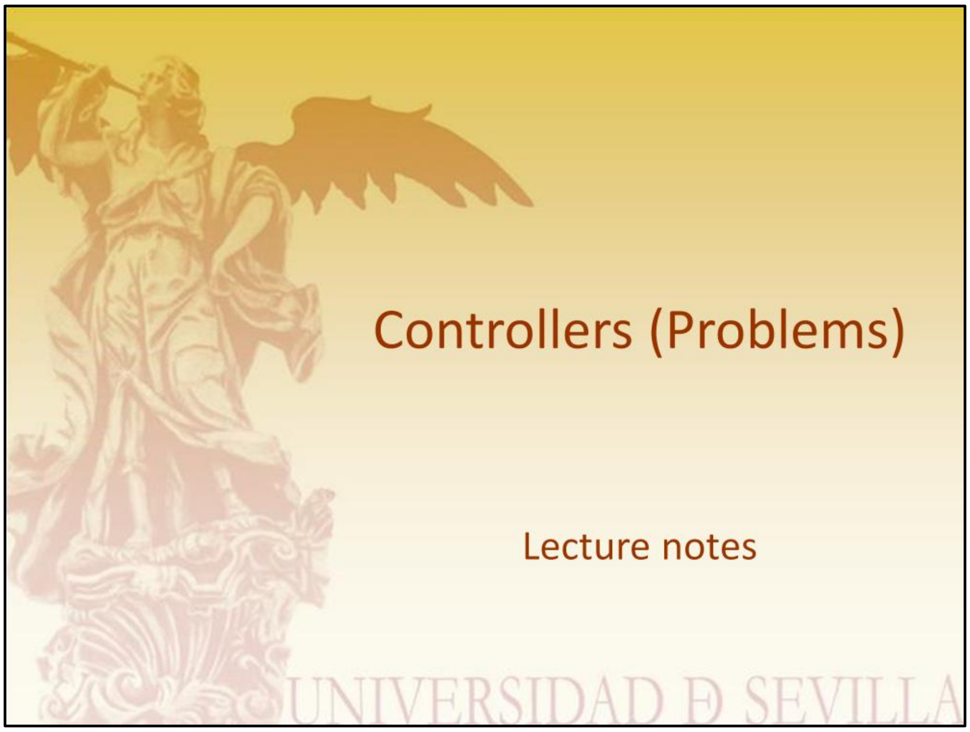


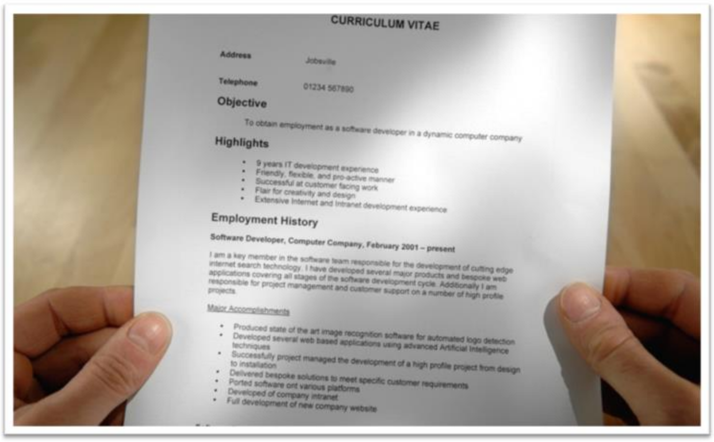
Controllers (Problems)

Lecture notes

Welcome to this lecture! Today, we'll start putting our knowledge on controllers to practice.

--

The problem: a curriculum manager



CURRICULUM VITAE

Address Jobsville

Telephone 01234 567890

Objective

To obtain employment as a software developer in a dynamic computer company

Highlights

- 9 years IT development experience
- Friendly, flexible, and pro-active manner
- Successful at customer facing work
- Flair for creativity and design
- Extensive Internet and Intranet development experience

Employment History

Software Developer, Computer Company, February 2001 – present

I am a key member in the software team responsible for the development of cutting edge internet search technology. I have developed several major products and bespoke web applications covering all stages of the software development cycle. Additionally I am responsible for project management and customer support on a number of high profile projects.

Major Accomplishments

- Produced state of the art image recognition software for automated logo detection
- Developed several web based applications using advanced Artificial Intelligence techniques
- Successfully project managed the development of a high profile project from design to installation
- Delivered bespoke solutions to meet specific customer requirements
- Ported software ont various platforms
- Developed of company intranet
- Full development of new company website

UNIVERSIDAD Ð SEVILLA

We’re going to work on a very simple curriculum manager.

Ready to start riding?

UNIVERSIDAD Ð SEVILLA

This problem's like riding this bicycle. It's a little more difficult than the previous problems because writing a controller requires you to pay attention to a variety of details; it's like removing one of the small back wheels, but not very difficult.

You're constrained!

UNIVERSIDAD D SEVILLA

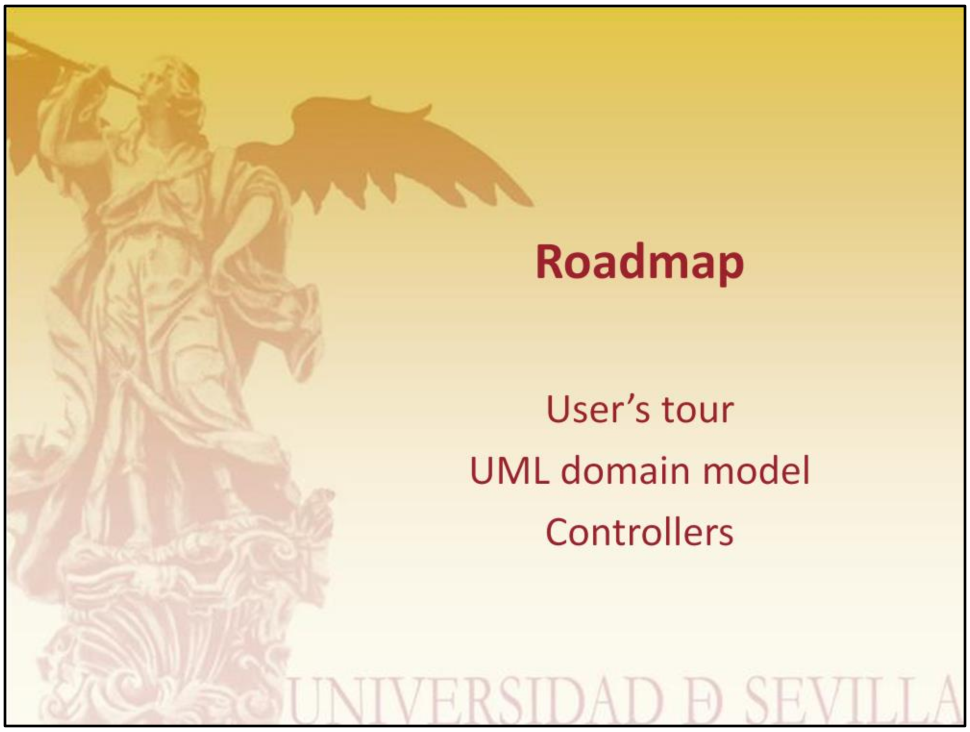
Note, however, that, as usual, you're totally constrained. Your only duty in this project is to implement a controller; you aren't allowed to change anything else. Please, take this very seriously: the other parts of the project were implemented by other people; you can't change the artefacts they've produced and hope this doesn't mess anything up!

Your first day as a software engineer

UNIVERSIDAD Ð SEVILLA

5

Please, realise that this problem's very similar to the problems that you'll have to solve on your first day as a software engineer: your boss will task you with a very simple task like implementing a controller building on a specification that will be very similar to ours.



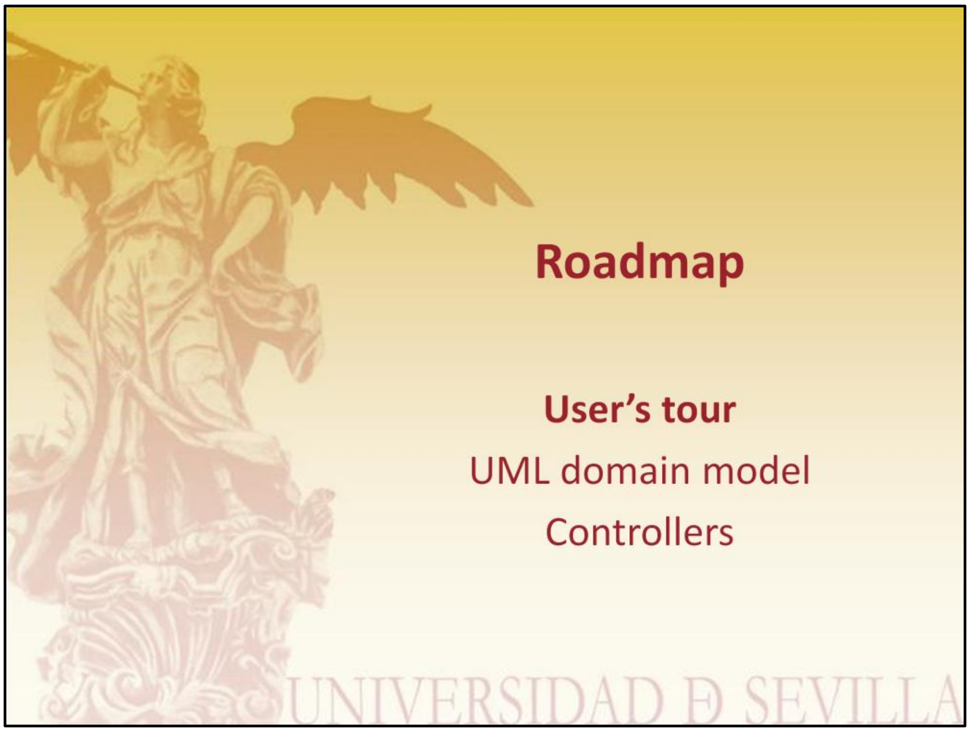
Roadmap

User's tour

UML domain model

Controllers

This is our roadmap for today. It won't be too long since the goal's that you can start working on your project as soon as possible. We'll start with a user's tour; then we'll present the UML domain model, and then the controllers that you have to implement.



Roadmap

User's tour

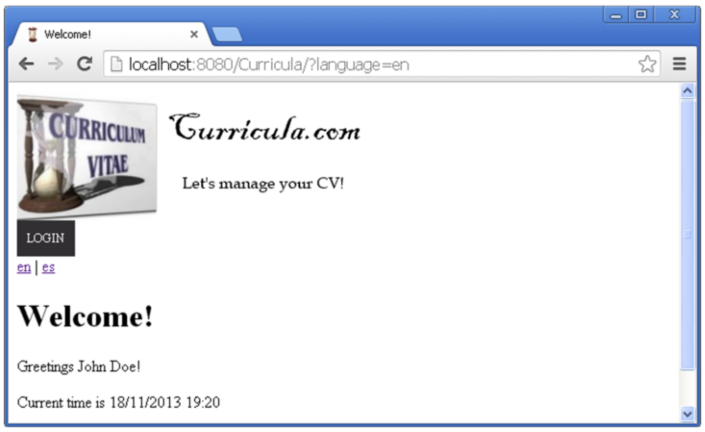
UML domain model

Controllers

UNIVERSIDAD Ð SEVILLA

Let's start with the user's tour. In this section, our goal's to present a number of screenshots so that you can have a better understanding of what we expect from this project.

The welcome screen



Curricula.com

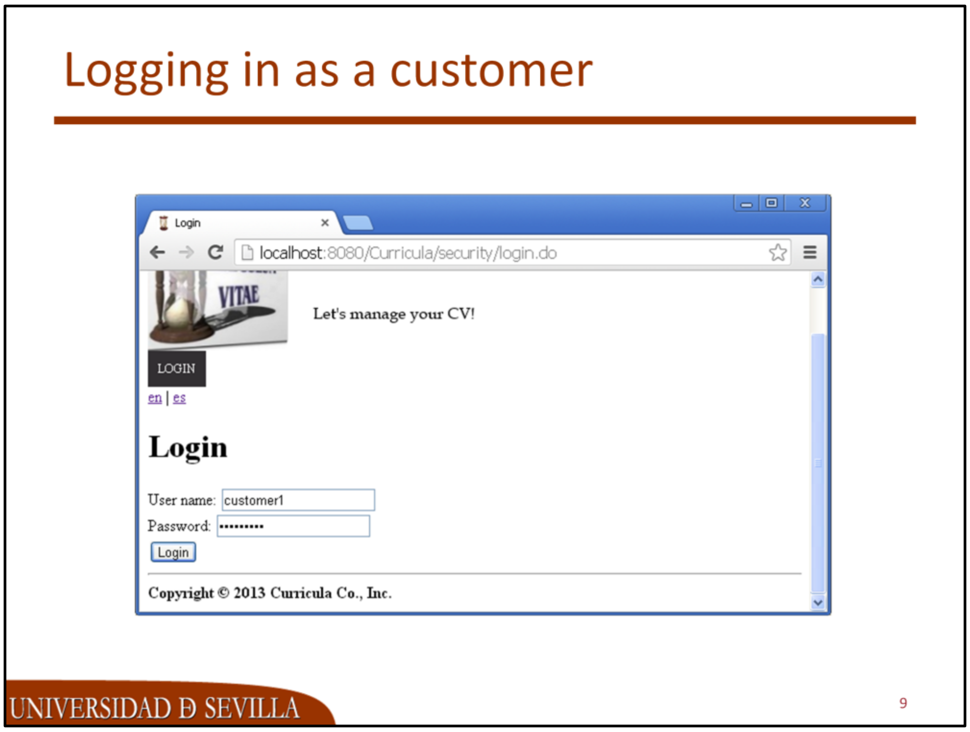
Let's manage your CV!

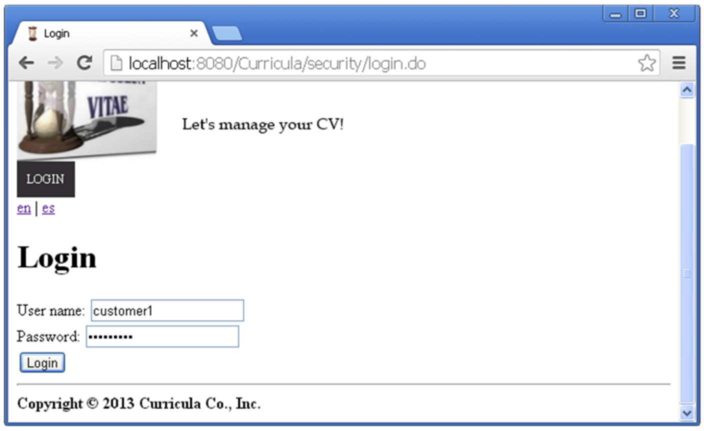
Welcome!

Greetings John Doe!

Current time is 18/11/2013 19:20

This is the welcome screen. By now, you should be very familiar with it. It has a logo, a simple menu that just displays “login”, a language bar, and a welcome message.


Logging in as a customer


Login

localhost:8080/Curricula/security/login.do

Let's manage your CV!

LOGIN

en | es

Login

User name: customer1

Password: ••••••••••

Login

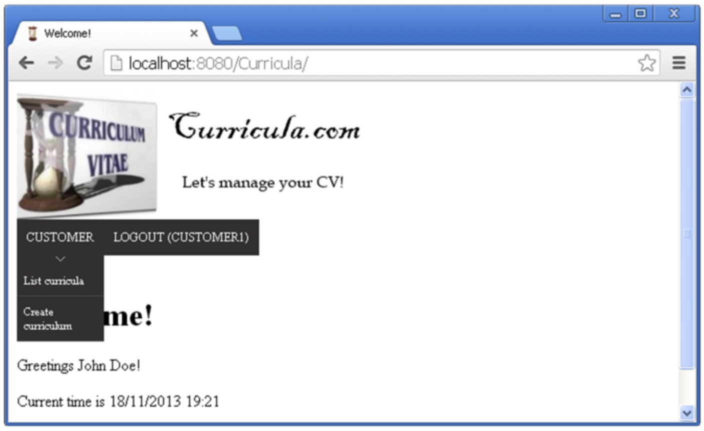
Copyright © 2013 Curricula Co., Inc.

UNIVERSIDAD D SEVILLA

9

You may log in as a customer using two pre-defined user accounts: "customer1/customer1" and "customer2/customer2".

The main menu



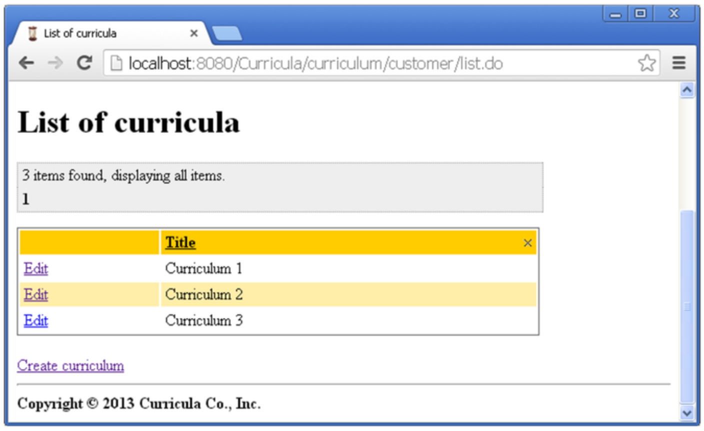
Greetings John Doe!

Current time is 18/11/2013 19:21

UNIVERSIDAD D SEVILLA

As a customer you can list your curricula and create new ones.

Listing curricula


List of curricula

localhost:8080/Curricula/curriculum/customer/list.do

List of curricula

3 items found, displaying all items.
1

	Title
Edit	Curriculum 1
Edit	Curriculum 2
Edit	Curriculum 3

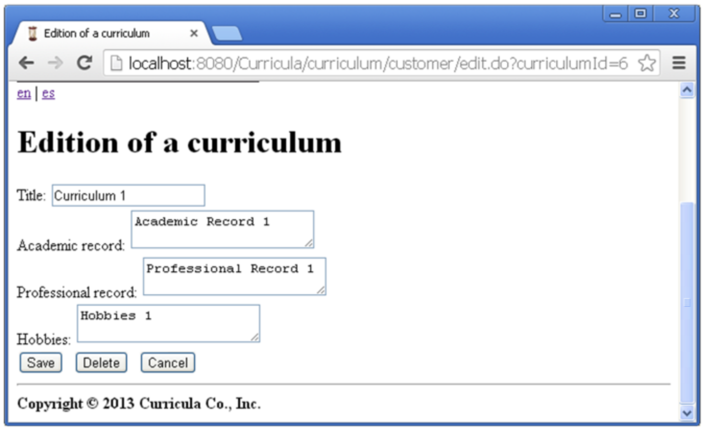
Create curriculum

Copyright © 2013 Curricula Co., Inc.

UNIVERSIDAD D SEVILLA

This is how a listing of curricula looks like. There's an "Edit" link on each curriculum and a "Create curriculum" link at the bottom of the listing, as usual.

Editing a curriculum


Edition of a curriculum

localhost:8080/Curricula/curriculum/customer/edit.do?curriculumId=6

en | es

Edition of a curriculum

Title: Curriculum 1

Academic record: Academic Record 1

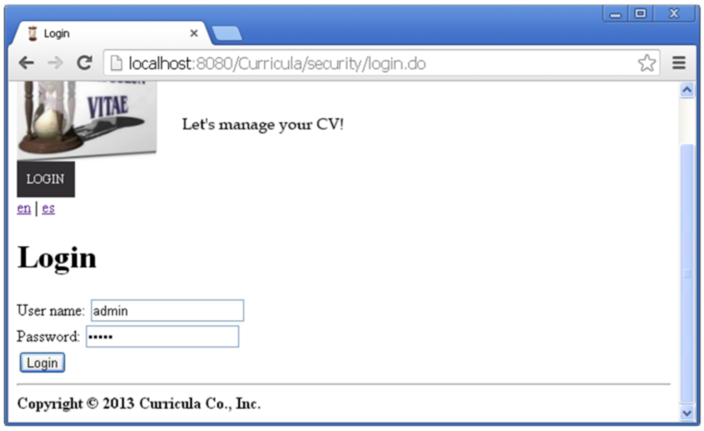
Professional record: Professional Record 1

Hobbies: Hobbies 1

Copyright © 2013 Curricula Co., Inc.

This slide shows the edition form for a curriculum. As usual it has a “Save” button to save the changes, a “Delete” button to delete a curriculum, an a “Cancel” button to return to the listing.

Logging in as an administrator


Login

localhost:8080/Curricula/security/login.do

Let's manage your CV!

LOGIN

en | es

Login

User name: admin

Password: •••••

Login

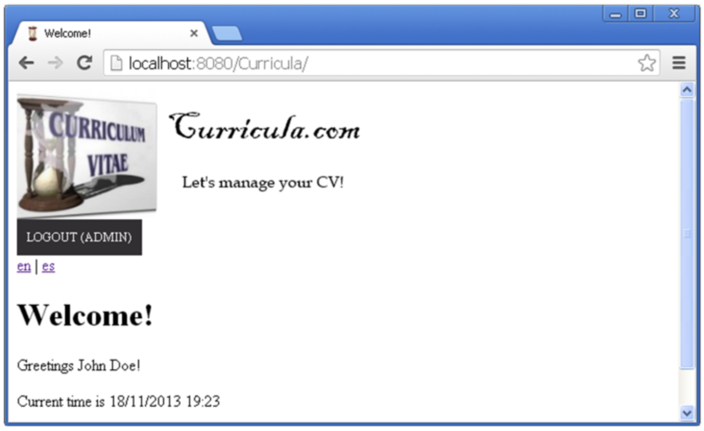
Copyright © 2013 Curricula Co., Inc.

UNIVERSIDAD Ð SEVILLA

13

You can also log in as an administrator using account “admin”/”admin”.

The main menu



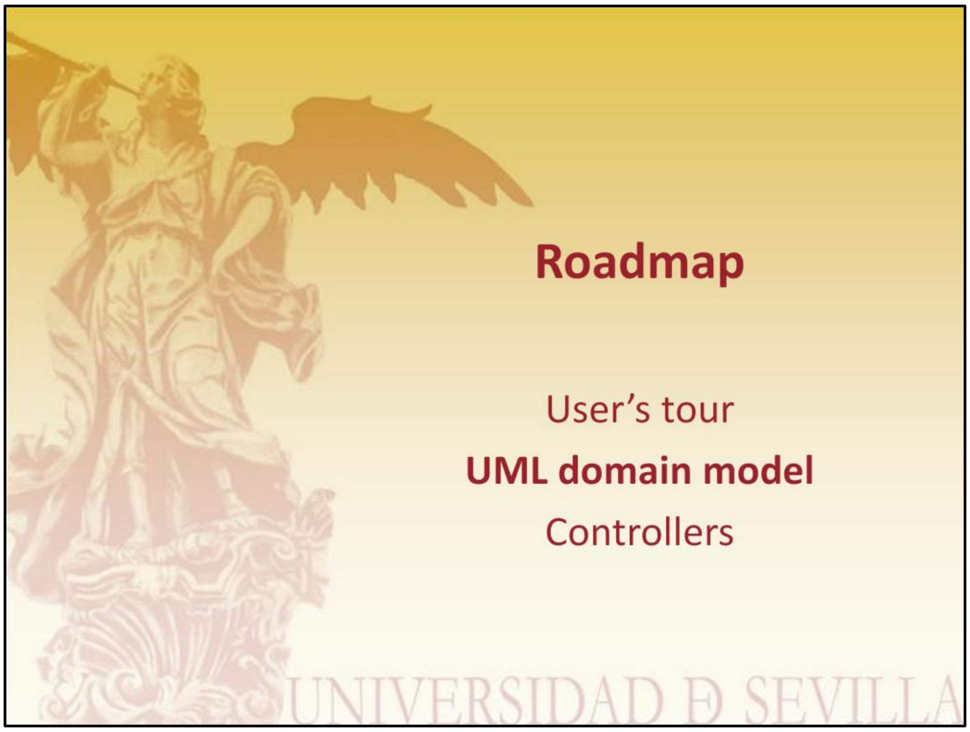
Welcome!

Greetings John Doe!

Current time is 18/11/2013 19:23

UNIVERSIDAD Đ SEVILLA

In this case, the main menu doesn't offer any options, except for logout. In other words, we won't implement any administrator's functionalities.



Roadmap

User’s tour

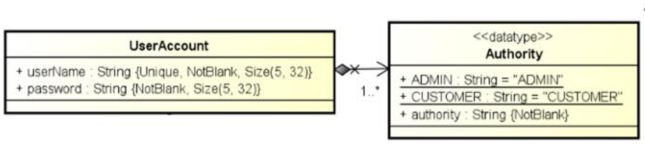
UML domain model

Controllers

UNIVERSIDAD Ð SEVILLA

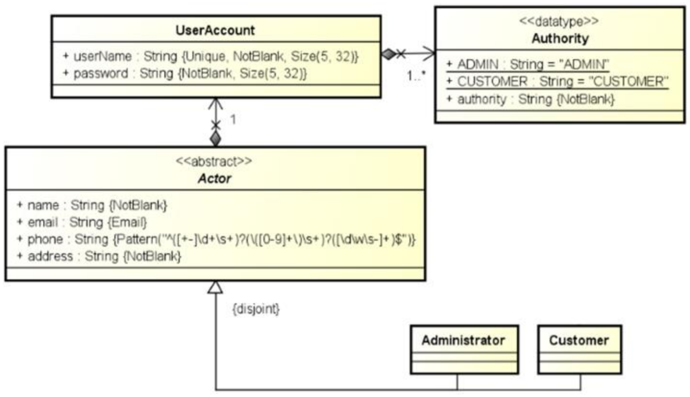
Let’s now present the UML domain model.

The model (I)


UserAccount
+ userName : String {Unique, NotBlank, Size(5, 32)}
+ password : String {NotBlank, Size(5, 32)}
1..*
<<datatype>>
Authority
+ ADMIN : String = "ADMIN"
+ CUSTOMER : String = "CUSTOMER"
+ authority : String {NotBlank}

As usual, we start with the account-related classes. In this problem, we have only two authorities, namely: administrators and customers.

The model (II)

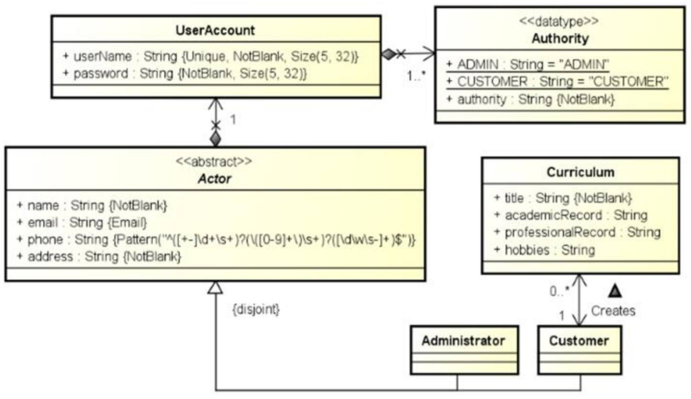

UserAccount
+ userName : String {Unique, NotBlank, Size(5, 32)}
+ password : String {NotBlank, Size(5, 32)}
1..*
<<datatype>>
Authority
+ ADMIN : String = "ADMIN"
+ CUSTOMER : String = "CUSTOMER"
+ authority : String {NotBlank}
1
<<abstract>>
Actor
+ name : String {NotBlank}
+ email : String {Email}
+ phone : String {Pattern("^([+-]\d+\s+)?(\([0-9]+\)\s+)?([\d\w\s-]+)$")}
+ address : String {NotBlank}
{disjoint}
Administrator
Customer

UNIVERSIDAD D SEVILLA

17

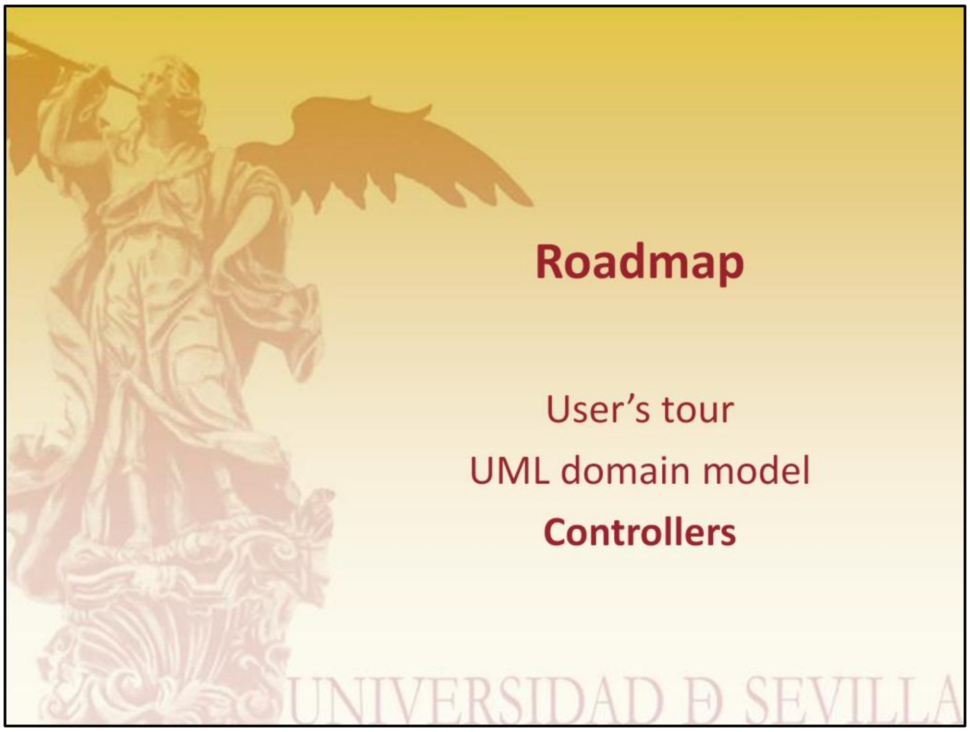
This slide shows the actor hierarchy; as usual there's a class to represent each of the authorities.

The model (III)


UserAccount
+ userName : String {Unique, NotBlank, Size(5, 32)}
+ password : String {NotBlank, Size(5, 32)}
1..*
<<datatype>>
Authority
+ ADMIN : String = "ADMIN"
+ CUSTOMER : String = "CUSTOMER"
+ authority : String {NotBlank}
1
<<abstract>>
Actor
+ name : String {NotBlank}
+ email : String {Email}
+ phone : String {Pattern("^([+-]\d+\s+)?(\([0-9]+\)\s+)?([\d\w\s-]+)$")}
+ address : String {NotBlank}
Curriculum
+ title : String {NotBlank}
+ academicRecord : String
+ professionalRecord : String
+ hobbies : String
0..*
1
Creates
{disjoint}
Administrator
Customer

UNIVERSIDAD Ð SEVILLA

This slide shows the model regarding curricula: each customer may create an arbitrary number of curricula, each of which has a title (which cannot be blank), an academic record, a professional record, and a hobbies section. Just realise that the only attribute with a constraint's "title"; in other words: you may save a curriculum with a null academic record, which allows you to save drafts and return to them later.



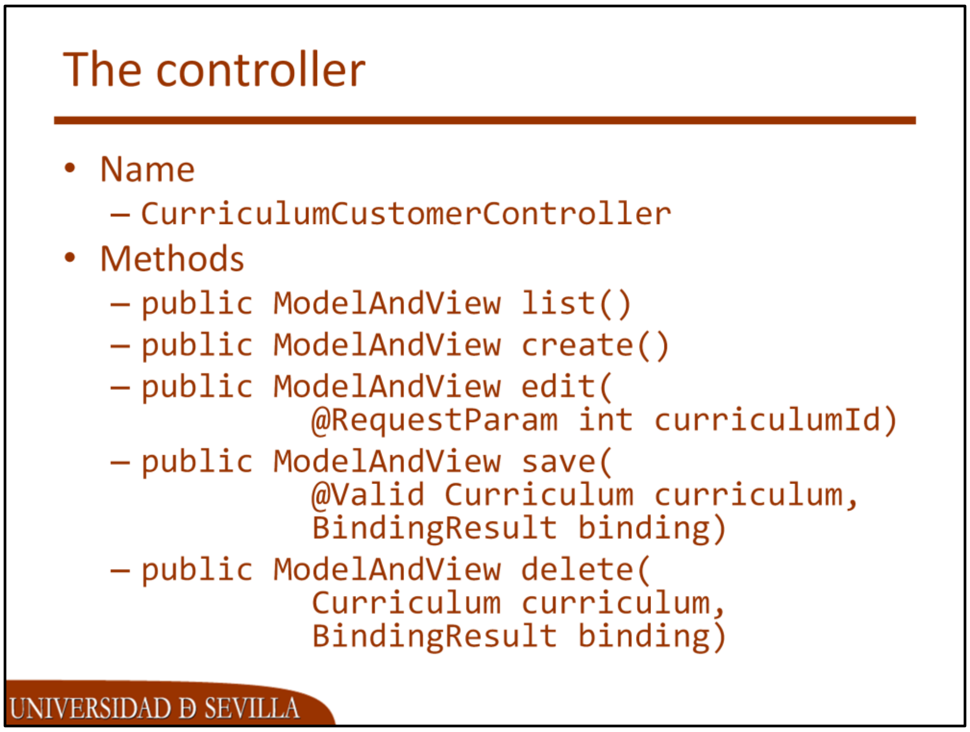
Roadmap

User’s tour

UML domain model

Controllers

The UML domain model was simple, wasn’t it? Let’s now provide a little more information on the controllers that you have to implement.



The controller

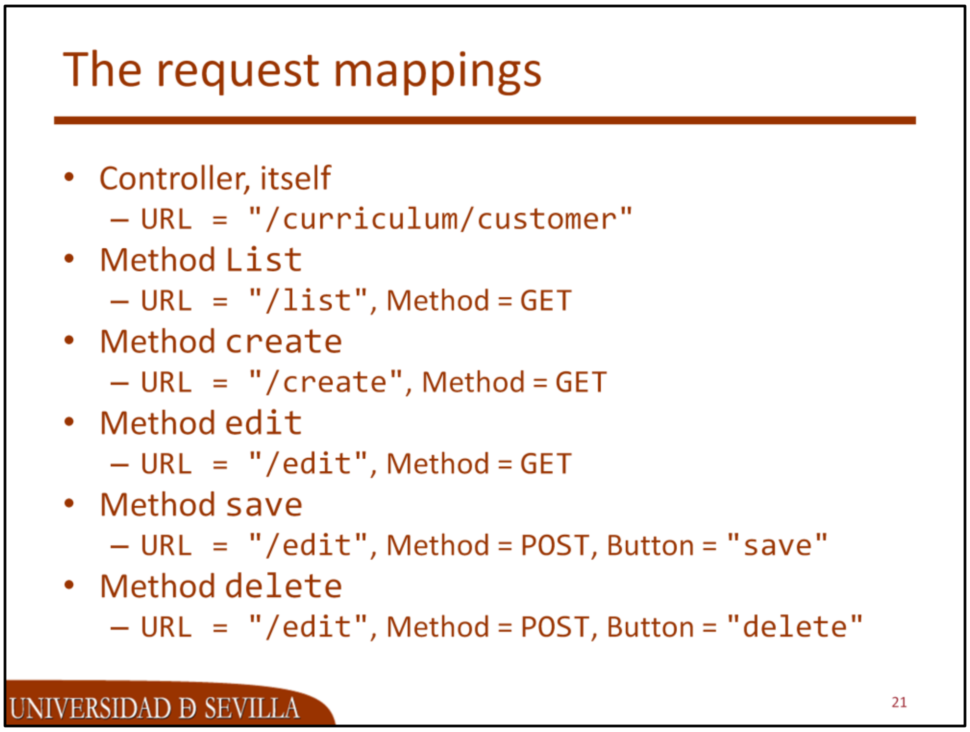
- Name
  - `CurriculumCustomerController`
- Methods

```
– public ModelAndView list()
– public ModelAndView create()
– public ModelAndView edit(
          @RequestParam int curriculumId)
– public ModelAndView save(
          @Valid Curriculum curriculum,
          BindingResult binding)
– public ModelAndView delete(
          Curriculum curriculum,
          BindingResult binding)
```

UNIVERSIDAD Ð SEVILLA

Ok, we talk about controllers in plural, but, in reality, you only have to implement a controller called “CurriculumCustomerController”. Below you may find an intuitive description of each method:

- “list”: it must return a model and a view to render the listing of curricula that were created by the principal.
- “create”: it must return a model and a view to edit an empty curriculum.
- “edit”: it gets the identifier of a curriculum as a parameter; it must return a model and a view to edit it. If the curriculum doesn’t exist or was created by a customer other than the principal, an exception is expected to occur.
- “save”: it gets a curriculum and a binding result as parameters; if the curriculum has validation errors, it’s expected to return a model and a view in which the errors are highlighted; otherwise, it’s expected to save the curriculum to the database and return to the listing; if an exception occurs while saving the curriculum, then it’s expected to return a model and a view so that the user can re-edit it and try to save it again. Note that no user but the customer who created a curriculum must be allowed to save it.
- “delete”: it gets a curriculum and a binding result as parameter; if the curriculum can be removed, then it’s expected to return to the listing; otherwise, an appropriate error message must be shown on the edition form. Note that no user but the customer who created a curriculum must be allowed to delete it.



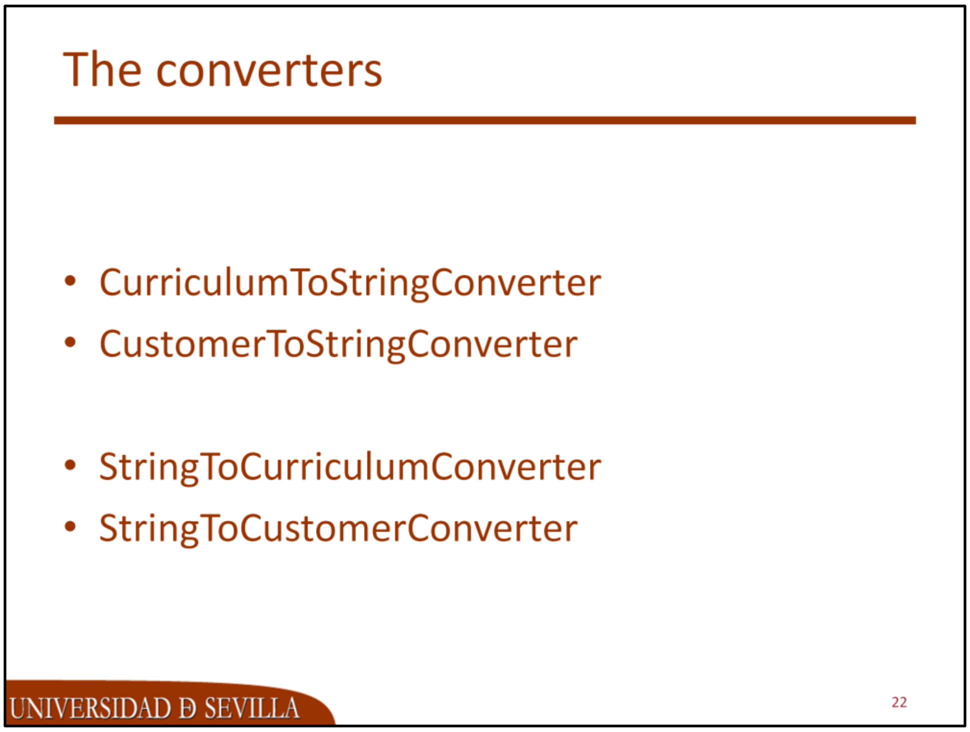
The request mappings

- Controller, itself
  - URL = `"/curriculum/customer"`
- Method `List`
  - URL = `"/list"`, Method = GET
- Method `create`
  - URL = `"/create"`, Method = GET
- Method `edit`
  - URL = `"/edit"`, Method = GET
- Method `save`
  - URL = `"/edit"`, Method = POST, Button = `"save"`
- Method `delete`
  - URL = `"/edit"`, Method = POST, Button = `"delete"`

UNIVERSIDAD D SEVILLA

21

This slide shows the request mappings for the controller, itself, and each of its methods.



The converters

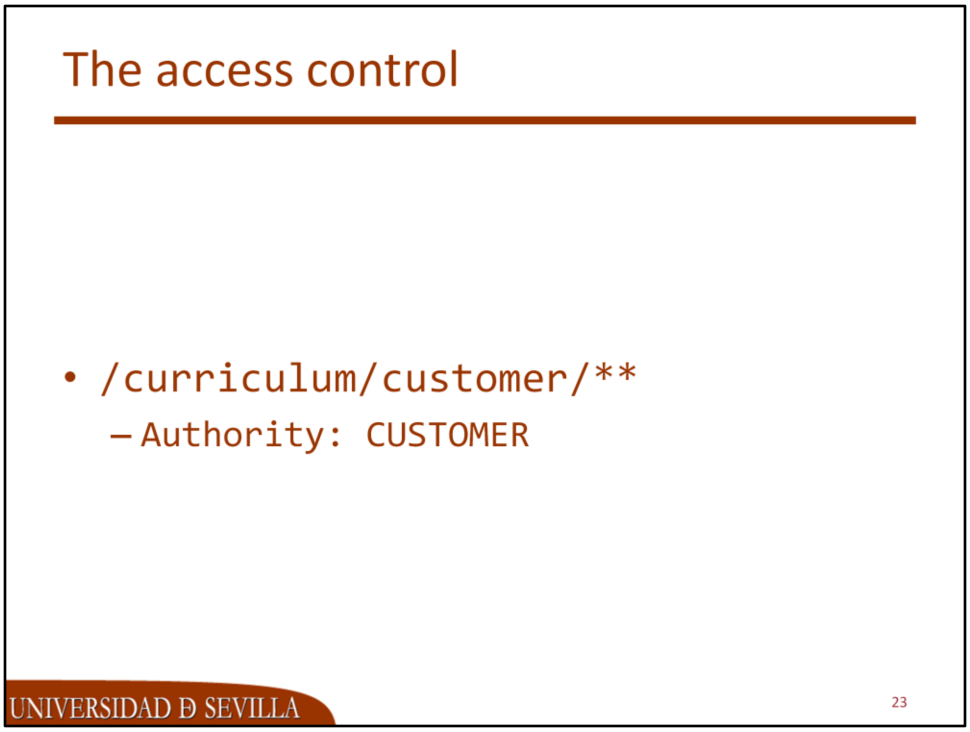
- CurriculumToStringConverter
- CustomerToStringConverter

- StringToCurriculumConverter
- StringToCustomerConverter

UNIVERSIDAD D SEVILLA

22

Please, note that the controllers are just a part of the picture. In order to complete the project, you also have to implement the converters that are presented in this slide, and you have to configure the project template so that it can work with them.



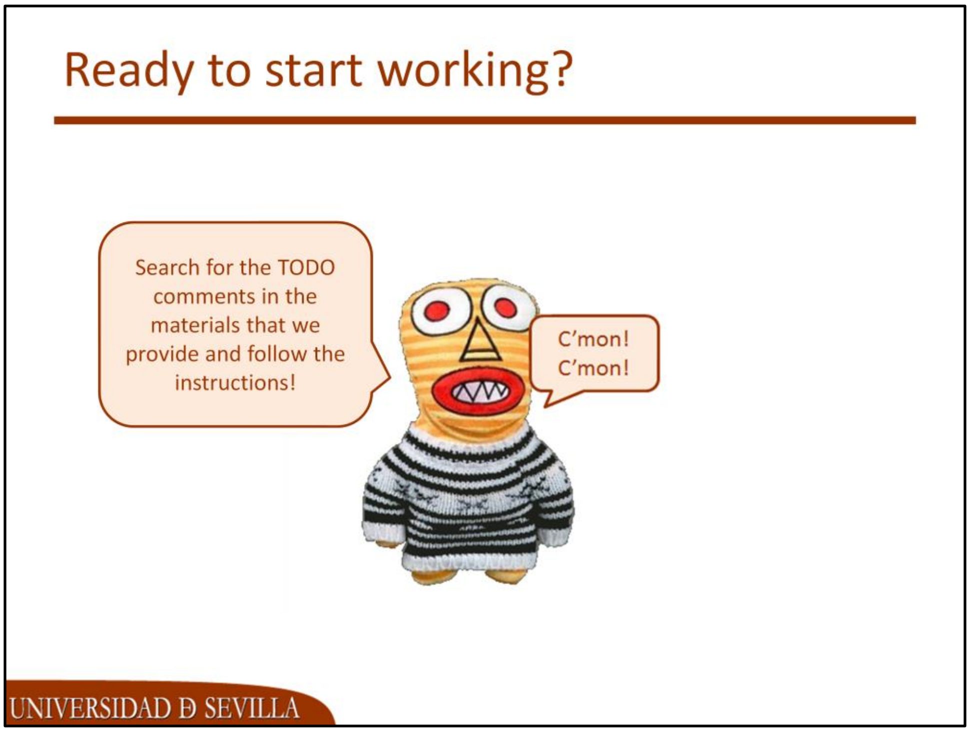
The access control

- /curriculum/customer/**
  - Authority: CUSTOMER

UNIVERSIDAD D SEVILLA

23

Furthermore, you have to configure your URL access control. Please, review the URLs of this web information system carefully, and configure the roles required to have access to them. It's a little cumbersome in general, but not difficult at all in this problem.



Ready to start working?

Search for the TODO comments in the materials that we provide and follow the instructions!
C'mon! C'mon!
UNIVERSIDAD Ð SEVILLA

Ready to start working? Just search for the TODO comments in the materials that we provide and follow the instructions. C'mon!

Thanks!

Sea breeze blows ahead.
And the sun glows.
Perfect summer day.

Thanks for attending this lecture! See you next day!