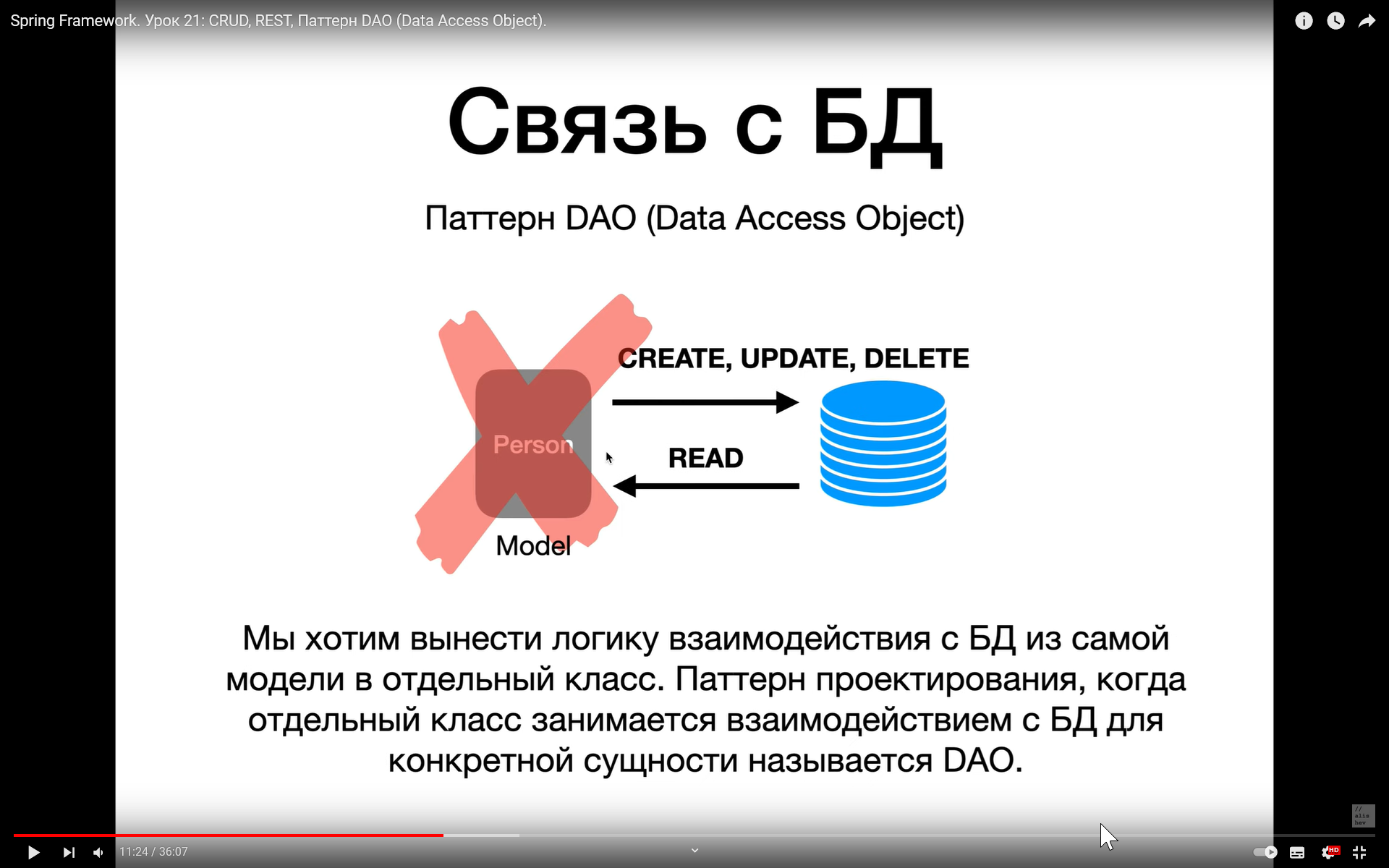


Если сущность Person в единственном числе, то контолллер должен быть во множественном числе, так как контроллер работает со множеством этих людей

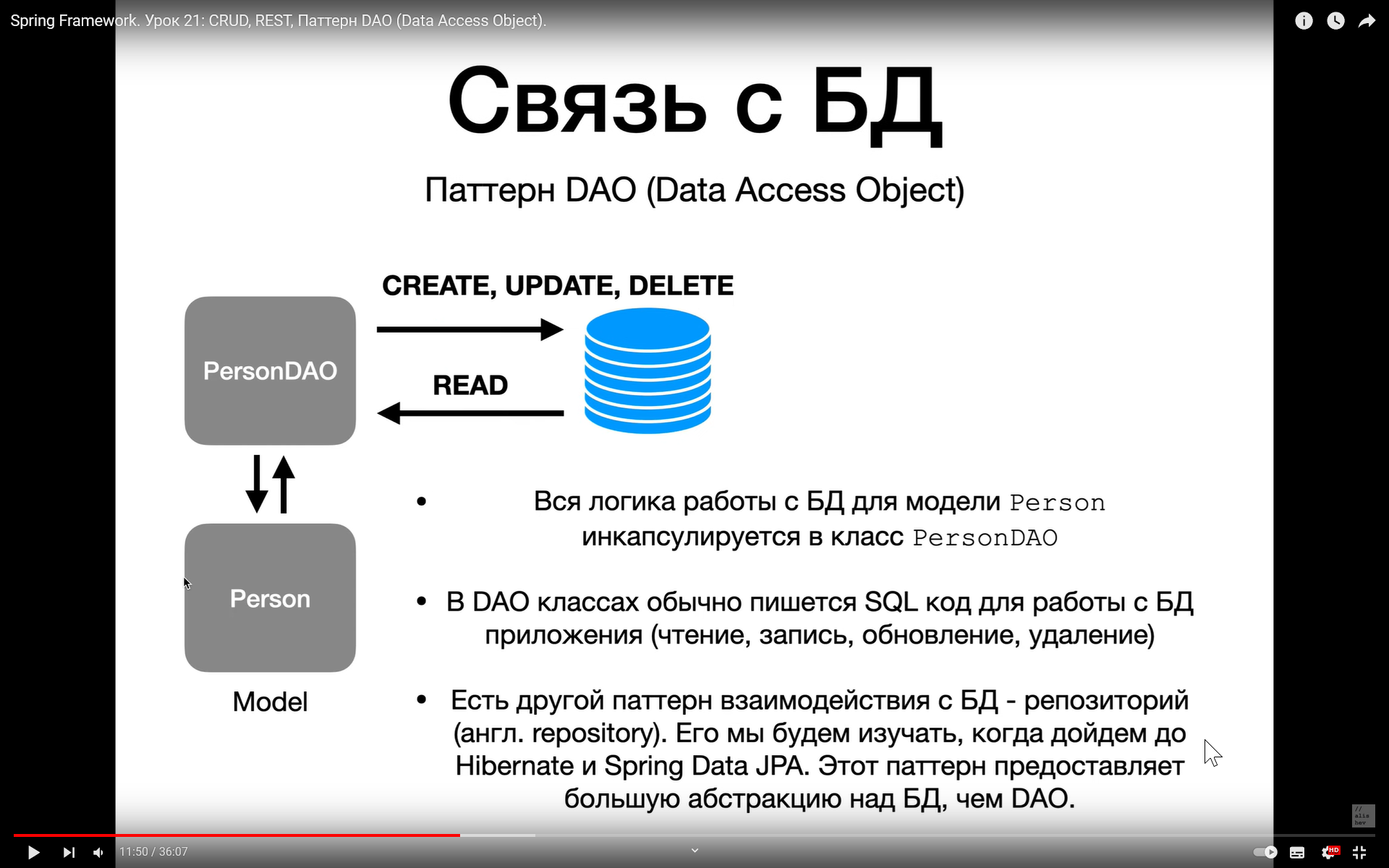
Get / people/ мы получим список из людей

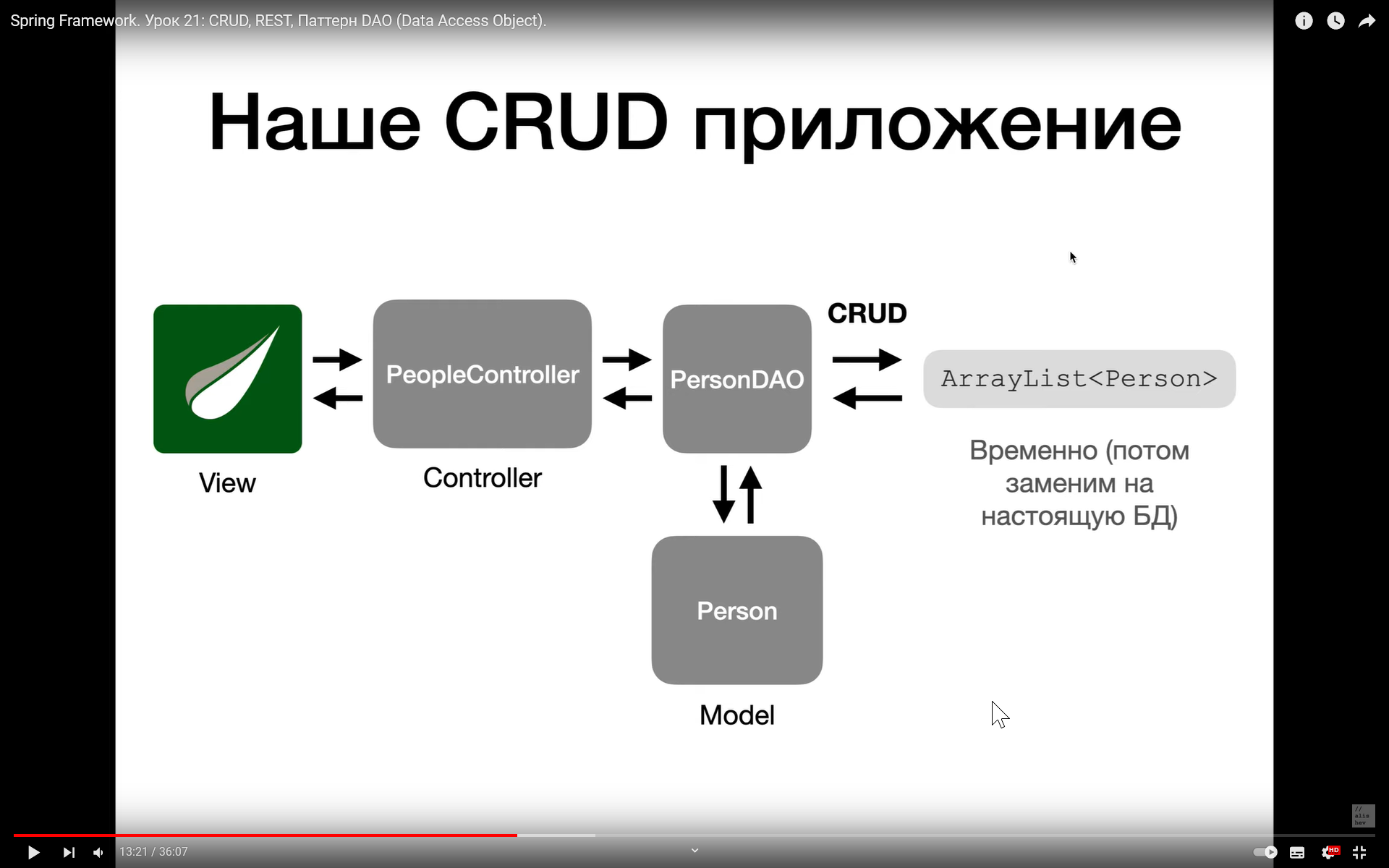
Get / people/:id получим какого-то конкретного человека



Мы не будем иметь ни какой логики по взаимодействию с базой данных в нашей моделе, у нас будет отдельный класс PersonDAO, который инкапсулирует всю работу с базой данных для этой модели Person

PersonDAO будет обращаться к базе данных и будет или читать или создавать новую запись, обновлять или удалять.





PersonDAO будет общаться с нашим списком, а PeopleController, контроллер, который работает с сущностью Person, будет обращаться к этому PersonDAO, будет либо читать с него данные, либо вызывать методы в этом DAO, чтобы

- создать

-обновить

-или удалить человека

14.40 тайм спем

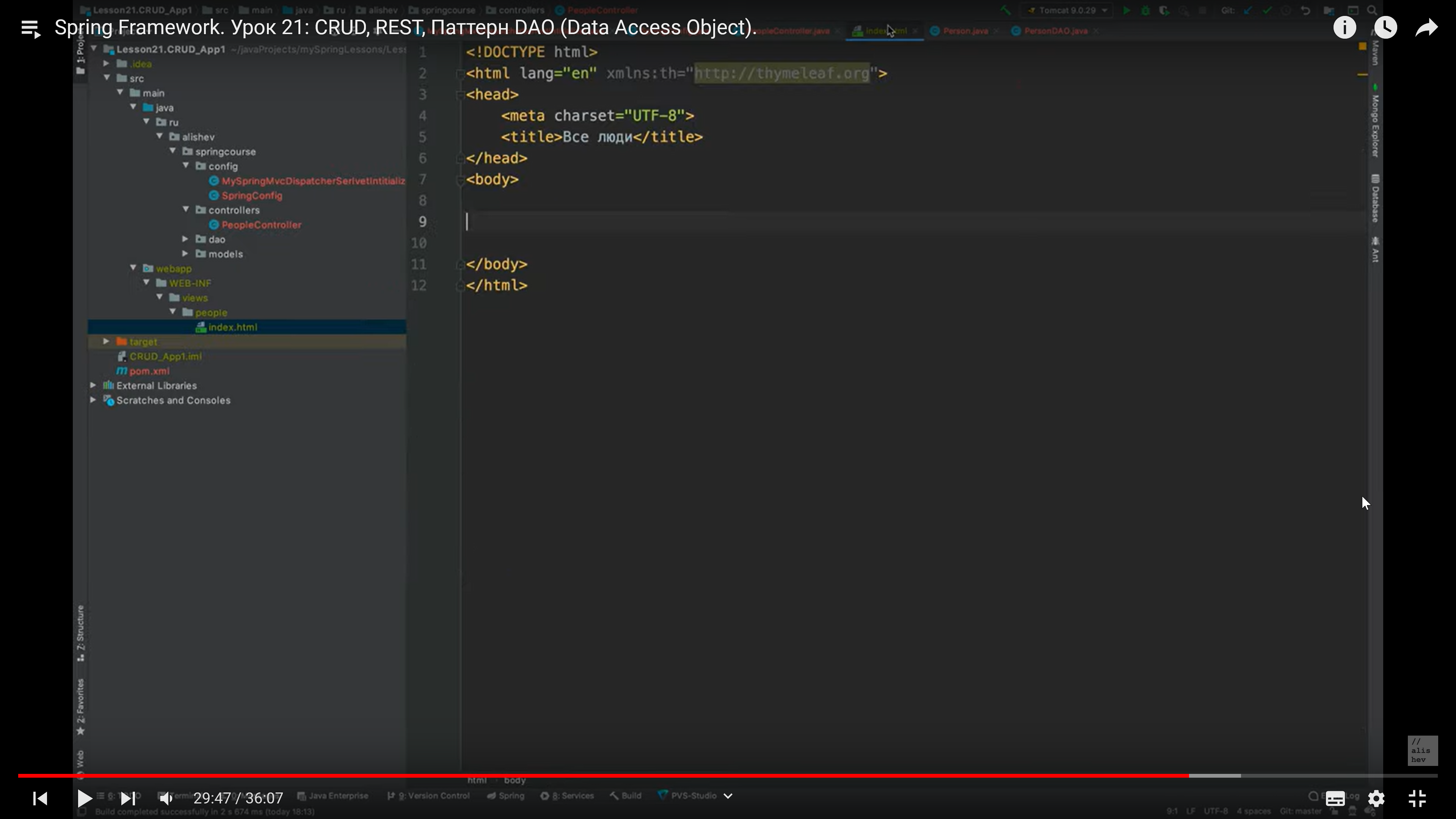
И все, что мы получаем от PersonDAO, список всех людей или конкретного человека, мы будем отображать на странице в браузере с помощью шаблонизатора таймлиф и с помощью таймлифа мы также создадим формы, мы будем каких-то новых людей создавать и будем делать post запросы, patch запросы и delete запросы к контроллеру, соответственно контроллер будет вызывать какой-то метод DAO и будет добавлять, удалять или обновлять человека.

Лучше внедрить через конструктор а не через аннотацию @Autowired над полем и таким образом зависимость внедряется в наш контроллер. Таким образом мы можем получать людей из нашей условной базы данных.

private final PersonDAO personDAO;  
  
public PeopleController(PersonDAO personDAO) {  
 this.personDAO = personDAO;  
}

Мы отправляем в шаблон список из людей под ключем people

@GetMapping()  
public String index(Model model){  
 //Получим всех людей из DAO и положим их в модель и передадим на отображение в представление  
 // тут под ключем people будет лежать список из людей  
 // динамический массив ArrayList из объектов класса Person  
 model.addAttribute("people", personDAO.index());  
 return "people/index";  
}



## Вопросы

http://localhost:8080/controllers/templates/people/1

по ссылке переходит на

http://localhost:8080/people/1

а надо на

http://localhost:8080/controllers/people/1

а если вставить controllers

переходит на

<http://localhost:8080/controllers/people/controllers/people/1>

шаблонизатор с ошибкой

Файл index.html

<!DOCTYPE html>

<html lang="en" xmlns:th="http://thymeleaf.org">

<head>

<meta charset="UTF-8" >

<title>All people</title>

</head>

<body>

<!--мы должны пройтись по списку из людей и вывести каждого человека-->

<!--в таймлифе есть подобие цикла.-->

<!--мы передаем на вход что-то на подобии цикла for each-->

<!--мы указываем здесь название коллекции ${views} и указываем-->

<!--временную переменную person, которая будет указывать-->

<!--на каждый элемент этой коллекции-->

<!--обращаемся к коллекции с помощью доллара и фигурных скобок-->

<div th:each="person : ${people}"></div>

<!--Тут будет лежать имя человека-->

<!--user будет вставляться таймлифом динамически-->

<!--При нажатии мы должны переходить на страницу каждого человека-->

<!--В круглых скобках указываем значение для этого id-->

<!--id равняется тому что лежит в person.getId(), то есть мы получаем id каждого человека-->

<!--мы получаем id каждого человека-->

<!--мы его кладем во временную переменную id-->

<!--эта временная переменная используется здесь, чтобы сформировать динамически правильный адрес-->

<!--на который будет переходить пользователь при нажатии на эту ссылку-->

<!--для первого будет-->

<!--/views/1-->

<!--для второго-->

<!--/views/2-->

<!--Таймлиф будет на динамически генерировать html code-->

<!--с правильными ссылками-->

<!--При нажатии на каждое из этих имен, на ссылку мы будет переходить

//(на страницу каждого человека) на указанный ниже метод контроллера-->

<!--@GetMapping("/{id}")-->

<!--public String show(@PathVariable("id") int id, Model model){-->

<!--// Получим одного человека по его id из DAO и передадим на отображение в представление-->

<!--// и должны положить его в модель и вернуть необходимый шаблон-->

<!--// тут будет лежать, то что пришло из DAO то что пришло по id-->

<!--model.addAttribute("person", personDAO.show(id));-->

<!--// тут вы возвращаем из этого метода название того шаблона, где будет показываться этот человек-->

<!--// под шаблоном имеется в виду views, то есть index.html-->

<!--return "views/show";-->

<!--будем передавать id в адресе-->

<!--}-->

<!--<a th:href="@{people/{id}(id=${person.getId()})}" th:text="${person.getName()}">user</a>-->

<a th:href="@{/people/{id}(id=${person.getId()})}" th:text="${person.getName()}">user</a>

</body>

</html>

Шаблонизатор show.index

Show.html

<!--Вместо VALUE таймлиф вставит динамически значение-->  
<!--Поля этого человека будут доступны в нашем шаблоне и их поместить вместо значения value-->  
  
  
<!--Человек получается от контроллера-->  
<!--Под ключ Person мы помещаем правильного человека-->  
<!--Поля этого человека будут доступны в нашем шаблоне-->  
<!--И мы их можем получить вместо значений VALUE-->  
  
<!--@GetMapping("/{id}")-->  
<!--public String show(@PathVariable("id") int id, Model model){-->  
<!--// Получим одного человека по его id из DAO и передадим на отображение в представление-->  
<!--// и должны положить его в модель и вернуть необходимый шаблон-->  
<!--// тут будет лежать, то что пришло из DAO то что пришло по id-->  
<!--model.addAttribute("person", personDAO.show(id));-->  
<!--// тут вы возвращаем из этого метода название того шаблона, где будет показываться этот человек-->  
<!--// под шаблоном имеется в виду views, то есть index.html-->  
<!--return "views/show";-->  
<!--}-->

Вопрос 2

при вводе адреса

<http://localhost:8080/controllers/people/>

создается ссылка   
people/1, а должна создаваться controllers/1

при аргументе

<a th:href="@{controllers/people/{id}(id=${person.getId()})}" th:text="${person.getName()}">user</a>

Создается ссылка   
controllers/people1

При переходе на нее

Мы переходим на

http://localhost:8080/controllers/people/controllers/people/1