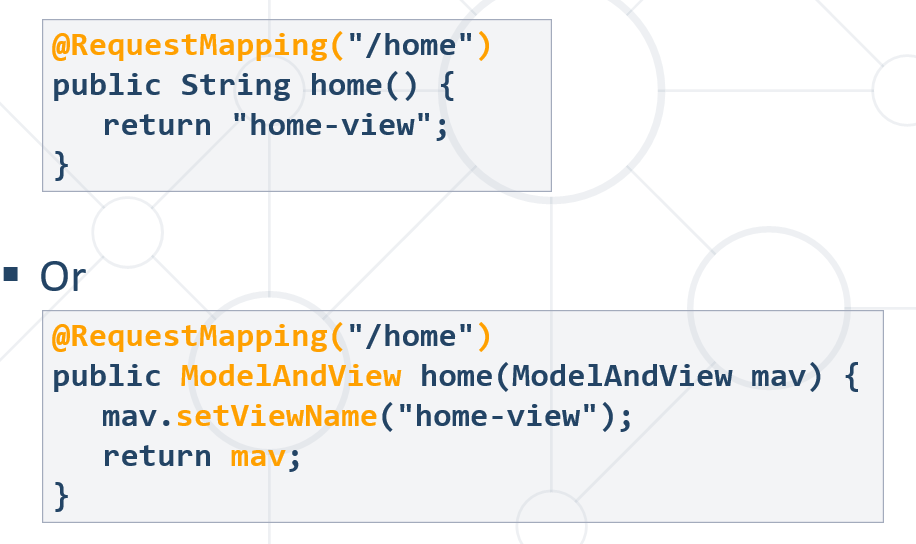
1. **Controller в Spring**

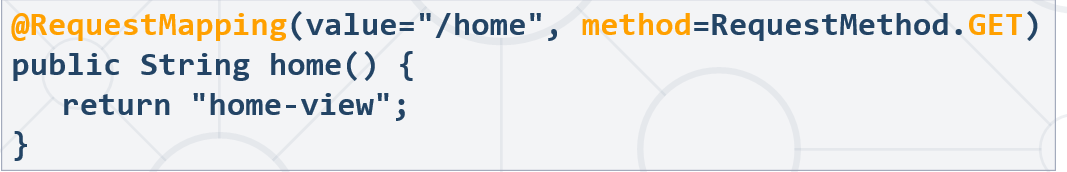
* Поставя се анотация @Controller



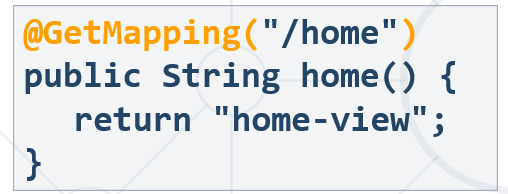
1. **Mapping annotations**

**-@RequestMapping –** по-добре е да се използва GetMapping и PostMapping. Request Mapping може да се изпозлва на класа, за да се посочи root-a на Url адреса





**-@GetMapping** - за Get заявки



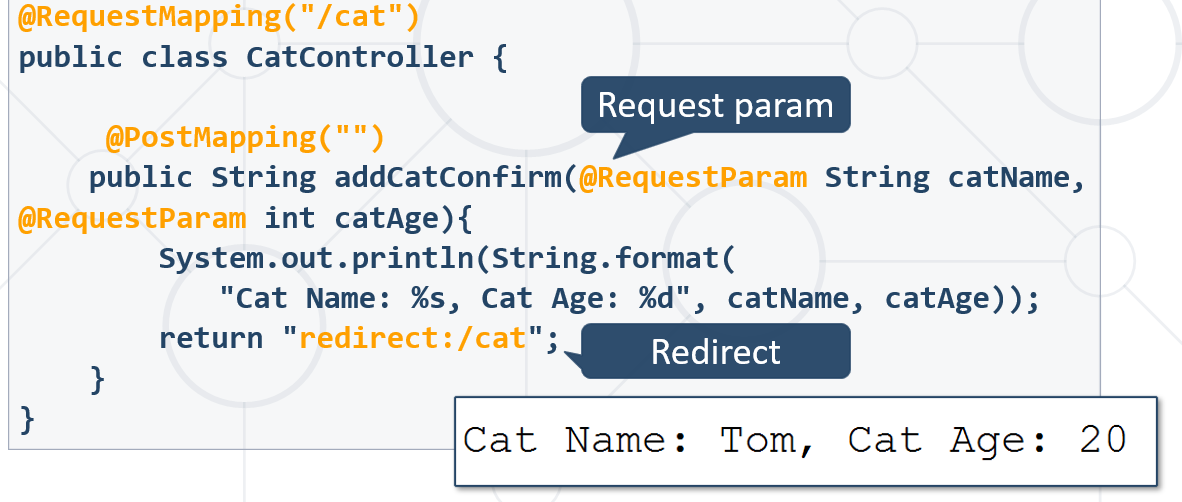
* Чрез анотацията **@ResponseBody** може да се връща цял обект (“game”) към клиента под формата на JSON body.



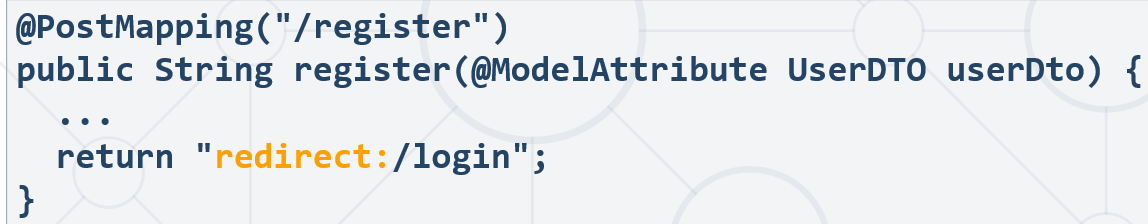
**-@PostMapping –** за Post заявки

* Getting a parameter from the query string

**-**чрез **@RequestParam** се assign-ват параметрите в URL адреса след „?“

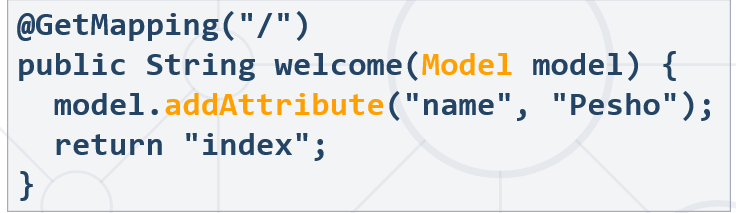


* **-@ModelAttribute – filling objects with a form data**

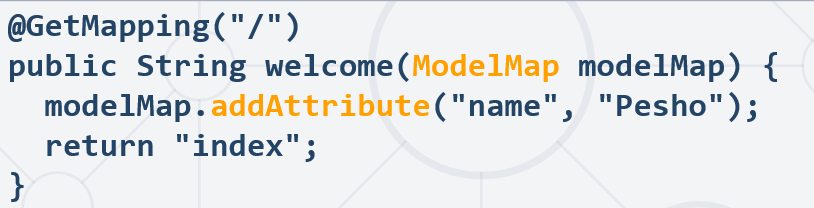


1. **Passing Attributes to View**

* Предават се параемтрите към HTML формата
* **Model**



* **ModelMap**



* **ModelAndView**



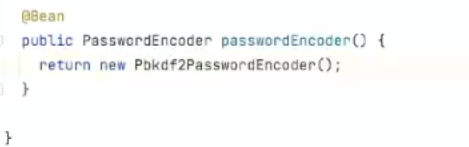
1. **Service and Binding model –**

- различни ДТО service и binding, които се извличат от HTTP Request и които се подават на Service

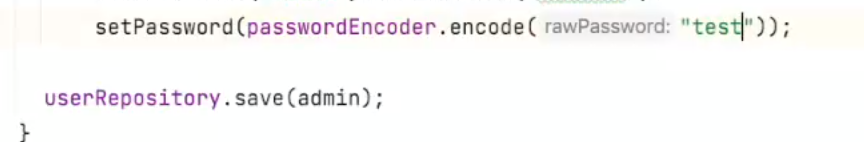
1. **Spring security crypto**

* добавя се депендънсито
* Inject PasswordEncoder – с метод matches() се сравнява raw password и encoded password.





* Метод encode() хешира паролата преди да се запази обекта в DB



1. **Logger**

* С метод .info() се използва логера за логване на данни в контролера

private static Logger *LOGGER* = LoggerFactory.*getLogger*(UserController.class);

*LOGGER*.info("User tried to login. User name {}, user password {}",   
 userLoginBindingDto.getUsername(),  
 userLoginBindingDto.getPassword());

1. **Import SQL file on startup**

* Добавя се в application файла

Spring:sql:init:mode:always



* След това се зъдава на същото ниво както application.yml файла data.sql

1. **@PathVariable and dynamic URL**

* ID предварително се конфигурира от динамичния th:href в HTML
* После при извикване на get заявка се предава ID към метода като PathVariable

