



CS544 EA

Applications

Spring MVC: Data Input

Request Input

We've seen how path variables can be used for input

GET /cars/1

```
@RequestMapping(value="/cars/{id}", method=RequestMethod.GET)
public String get(@PathVariable int id, Model model) {
    model.addAttribute("car", carDao.get(id));
    return "carDetail";
}
```

The same can of course be done with normal request params

Params specification is optional!

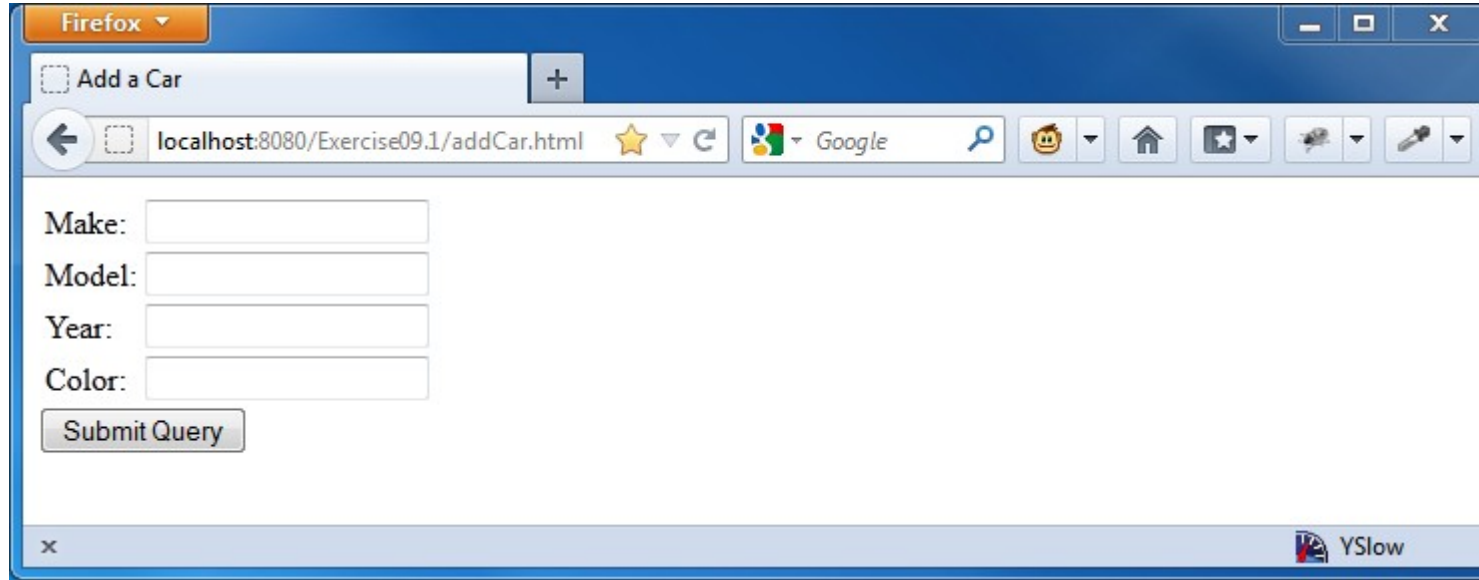
GET /cars?id=1

```
@RequestMapping(value="/cars", params="id", method=RequestMethod.GET)
public String getParam(@RequestParam int id, Model model) {
    model.addAttribute("car", carDao.get(id));
    return "carDetail";
}
```

@RequestParam is optional

Method param name has to match request param name

Many Parameters



The screenshot shows a Firefox browser window with a single tab titled 'Add a Car'. The address bar displays 'localhost:8080/Exercise09.1/addCar.html'. The page content includes four text input fields labeled 'Make:', 'Model:', 'Year:', and 'Color:', each followed by a small rectangular input box. Below these fields is a 'Submit Query' button. The browser's status bar at the bottom right shows a 'YSlow' icon.

```
public class Car {  
    private int id;  
    private String make;  
    private String model;  
    private int year;  
    private String color;  
}
```

Do Less Accomplish More

```
@RequestMapping(value="/cars", method=RequestMethod.POST)  
public String addParams(String make, String model, int year, String color) {  
    Car car = new Car(make, model, year, color);  
    carDao.add(car);  
    return "redirect:/cars";  
}
```

You can receive the from params
and then combine them into a Car

But you can also have Spring
do all the work for you

```
@RequestMapping(value="/cars", method=RequestMethod.POST)  
public String add(Car car) {  
    carDao.add(car);  
    return "redirect:/cars";  
}
```

Car class does have to adhere
to JavaBean standard

@RequestBody

- Web Services usually need to process the entire incoming request body (instead of parts)

```
@RestController
public class WebService {
    @Resource
    private CarService carService;

    @GetMapping(value="/cars", produces="application/json")
    public List<Car> getAll() {
        return carService.getAll();
    }

    @PostMapping(value="/addCar", consumes="application/json")
    public void addCar(@RequestBody Car car) {
        carService.add(car);
    }
}
```

Additional Parameters

- The following objects can be passed into Methods:

@PathVariable
@RequestParam
@RequestHeader
@RequestBody
@RequestPart (file upload)
Map / Model / ModelMap
BindingResult / Errors
SessionStatus
RedirectAttributes

HttpServletRequest
HttpServletResponse
HttpSession
InputStream
OutputStream
Reader
Writer
Principal (security)
Locale (internationalization)

- You can also define your own custom injectors
 - See Spring documentation