

**Napredne tehnike i principi razvoja softvera**  
**Razvoj web servisa i serverske strane web aplikacija**

TEMA 2:

# REST KONTROLERI

---

Vladimir Dimitrieski, PhD, [dimitrieski@uns.ac.rs](mailto:dimitrieski@uns.ac.rs)

Milan Čeliković, PhD, [milancel@uns.ac.rs](mailto:milancel@uns.ac.rs)



# Pregled

- REST (step-by-step)
  - Napredni elementi kreiranja REST servisa u Springu
  - Metode
  - Parametri
  - Postman
- Debugging
  - Breakpoints
- Zadaci

# REST KONTROLERI

---

Napredni elementi kreiranja REST servisa u Spring-u

Metode

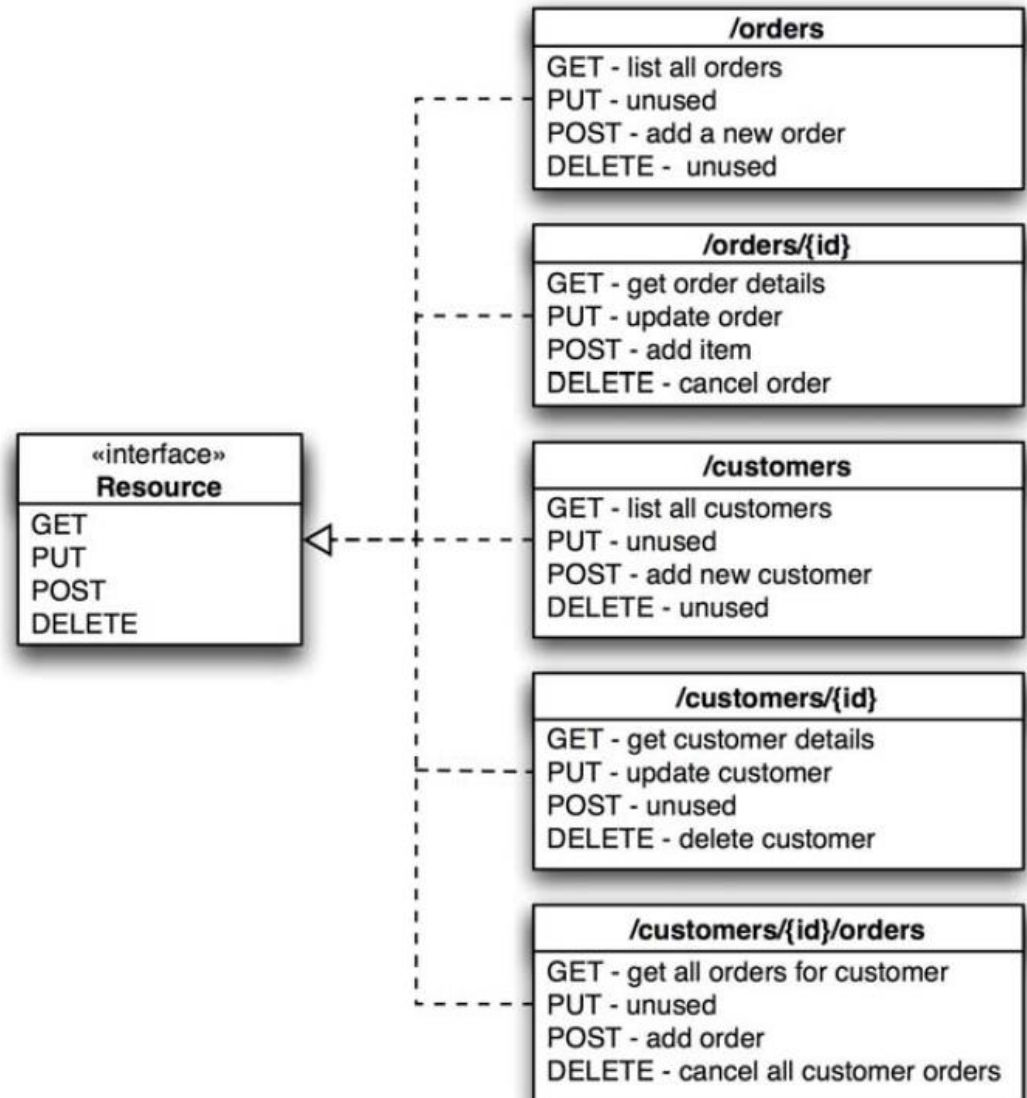
Parametri

# REST – standardne metode

- Koristiti isključivo standardne HTTP metode za rukovanje resursima
  - GET, POST, PUT, DELETE ...
  - bez uvođenja novih metoda specifičnih za trenutni projekat
  - generički klijenti mogu samo da implementiraju standardne HTTP metode i da poznaju URI resursa
- Primeri:
  - dobavljanje klijenta sa ID-jem 1234
    - GET na <http://example.com/customers/1234>
  - brisanje klijenta sa ID-jem 1234
    - DELETE na <http://example.com/customers/1234>
  - dodavanje novog klijenta sa ID-jem 4321
    - POST na <http://example.com/customers/4321>
    - u telu zahteva specificirati vrednosti za sva polja klijenta

# REST – standardne metode

- Standardni interfejs
  - npr. četiri metode
  - implementiraju ga svi servisi



# STS projekat

- Kreirati novi spring boot projekat u STS
  - File -> New -> Spring starter project
  - **na prvom ekranu popuniti kao što je dato na slici**
    - nakon popunjavanja pritisnuti dugme Next
  - **na drugom ekranu sve ostaviti kako jeste**
    - nisu potrebne dodatne biblioteke
    - pritisnuti dugme Finish
  - **Dodati u pom.xml zavisnost od spring-boot-starter-web biblioteke**

The screenshot shows the 'New Spring Starter Project' dialog in the Spring Tool Suite (STS). The dialog is titled 'New Spring Starter Project' and features a green power button icon in the top right corner. The fields are as follows:

- Service URL: `http://start.spring.io`
- Name: `rest_examples`
- ☒ Use default location
- Location: `E:\preobuka20017\wsgit\rest_examples` (with a 'Browse' button)
- Type: `Maven`
- Packaging: `Jar`
- Java Version: `1.8`
- Language: `Java`
- Group: `com.iktpreobuka`
- Artifact: `rest_examples`
- Version: `0.0.1-SNAPSHOT`
- Description: `Demo project for Spring Rest examples`
- Package: `com.iktpreobuka.restexamples`

At the bottom, there is a 'Working sets' section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below this is a 'Working sets:' dropdown menu and a 'Select...' button. At the very bottom, there are four buttons: '?', '< Back', 'Next >' (highlighted with a blue border), 'Finish', and 'Cancel'.

# STS projekat – REST Kontroler

- kreirati novi paket
  - desni klik na postojeći paket -> new -> package
    - nazvati ga *com.iktpreobuka.restexamples.controllers*
- kreirati novu klasu u paketu
  - desni klik na paket -> new -> class
    - nazvati je *BankClientRestController*
- **@RestController**
  - označava da je klasa Spring REST kontroler
    - moguće je slati zahteve ka metodama klase
- **@RequestMapping("/")**
  - definiše putanju poziva REST metode
  - koja je implementirana pomoću odgovarajuće Java metode u klasi

# STS projekat – REST kontroler

```
package com.iktpreobuka.restexamples.controllers;

import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/bankclients")
public class BankClientRestController {

}
```



# Primer 1

- Kreirati REST *endpoint*
  - Koji vraća listu koja sadrži klijente banke
    - putanja **/bankclients**
- kreirati novi paket
  - desni klik na postojeći paket -> new -> package
    - nazvati ga *com.iktpreobuka.restexamples.entities*
- kreirati novu klasu u paketu entities
  - desni klik na paket -> new -> class
    - nazvati je *BankClientBean*
  - Klasa *BankClientBean* sadrži attribute:
    - *Id, name, surname, email*

# Primer 1

```
package com.iktpreobuka.restexamples.entities;
```

```
public class BankClientBean {
```

```
    protected Integer id;
```

```
    protected String name;
```

```
    protected String surname;
```

```
    protected String email;
```

```
}
```

# Primer 1

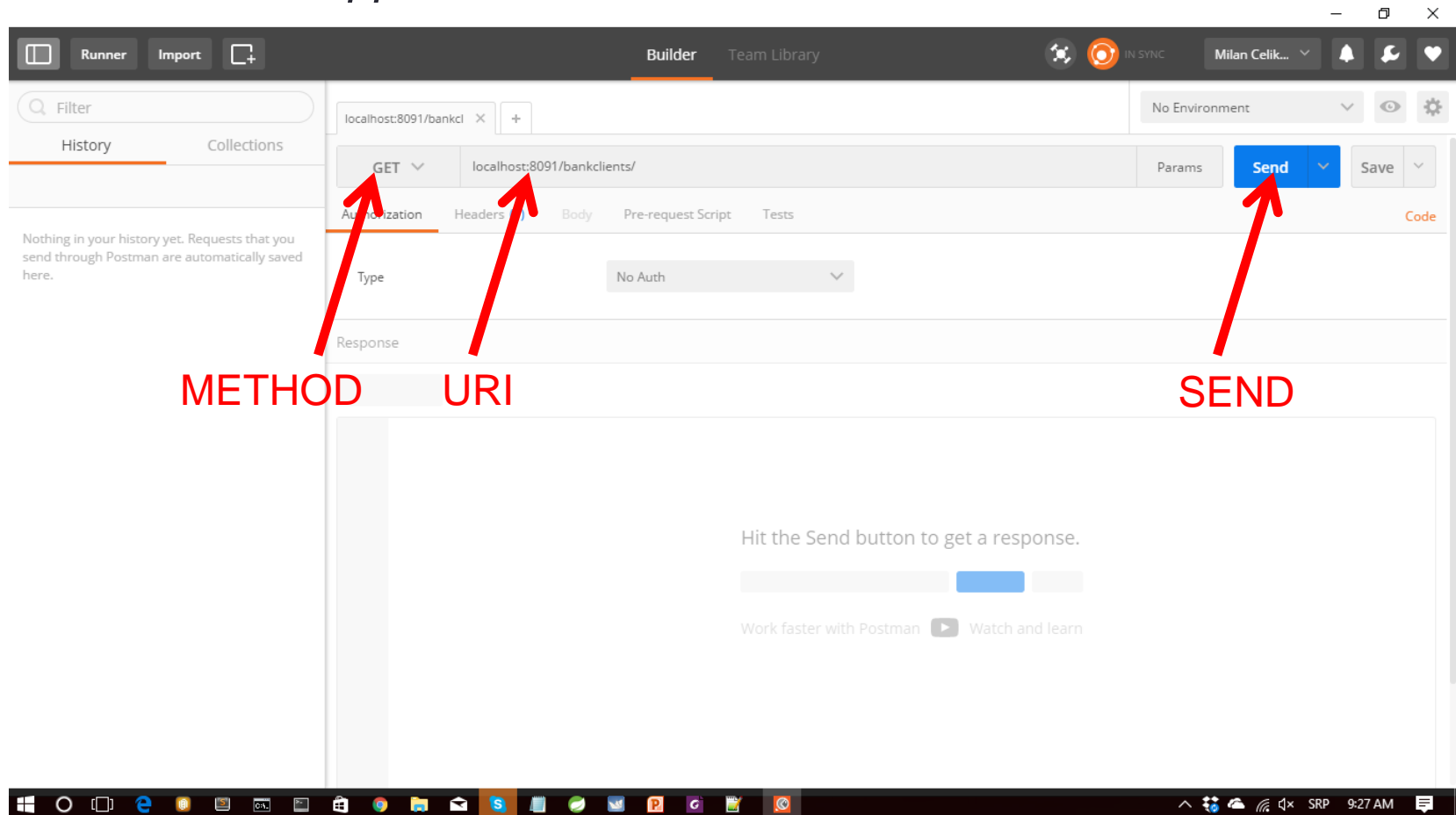
- U klasi *BankClientBean* kreirati get i set metode, prazan konstruktor i konstruktore sa parametrima
- U klasi *BankClientRestController* kreirati metodu `getAll()` koja vraća listu klijenata banke
- Parametar *method* anotacije *RequestMapping* postaviti na vrednost *RequestMethod.GET*

# Primer 1

```
@RequestMapping(method = RequestMethod.GET)  
public List<BankClientBean> getAll() {  
    List<BankClientBean> clients = new ArrayList<BankClientBean>();  
    clients.add(new BankClientBean(1, "Milan", "Celikovic",  
        "milancel@uns.ac.rs"));  
    clients.add(new BankClientBean(2, "Vladimir", "Dimitrieski",  
        "dimitrieski@uns.ac.rs"));  
    return clients;  
}
```

# Primer 1 - testiranje

- Pokrenuti *Postman* aplikaciju
  - *Chrome -> Apps -> Postman*



# Primer 2

- Kreirati REST *endpoint*
  - Koji vraća klijenta banke po vrednosti *id* klijenta
    - putanja **/bankclients/{clientId}**
    - ukoliko klijent sa prosleđenom *id* vrednosti ne postoji vratiti obejkat sa null vrednostima
- @PathVariable
  - označava da metoda prihvata parametar (*Path parameter*) koja je deo URI
- Parametar *value* anotacije *RequestMapping* postaviti na vrednost **/clientId**

# Primer 2

```
@RequestMapping(method = RequestMethod.GET, value =("/{clientId}")  
public BankClientBean getById(@PathVariable String clientId) {  
    if(clientId.equals("1"))  
        return new BankClientBean(1, "Milan", "Celikovic",  
            "milancel@uns.ac.rs");  
    else  
        return new BankClientBean();  
}
```

## Primer 2 - testiranje

The screenshot displays a REST client interface with a tab labeled 'localhost:8090/bankcl'. Below the tab, the request configuration is shown:

- Method:** GET (indicated by a red arrow labeled 'METHOD Type').
- URI:** localhost:8090/bankclients/1 (indicated by a red arrow labeled 'URI').
- Path Parameter:** 1 (highlighted with a red box and indicated by a red arrow labeled 'PATHPARAM').
- Auth:** No Auth (indicated by a red arrow labeled 'SEND').
- Buttons:** Send (blue button), Save (grey button).
- Tabs:** Auth, Headers (1), Body, Pre-req., Tests, Code.

Red arrows point from the labels 'METHOD Type', 'URI', 'PATHPARAM', and 'SEND' to their respective elements in the interface.



## Primer 2

- Kreirati metodu *getDB* unutar kontrolera *BankClientRestController* koja vraća listu klijenata banke.
  - Metoda treba da bude vidljiva samo unutar klase
- Izmeniti metodu *getById* tako da pronalazi klijenta u listi klijenata na osnovu prosleđene *id* vrednosti

## Primer 2

```
protected List<BankClientBean> getDB() {  
    List<BankClientBean> clients = new ArrayList<BankClientBean>();  
    clients.add(new BankClientBean(1, "Milan", "Celikovic",  
        "milancel@uns.ac.rs"));  
    clients.add(new BankClientBean(2, "Vladimir",  
        "Dimitrieski", "dimitrieski@uns.ac.rs"));  
    return clients;  
}  
  
@RequestMapping(method=RequestMethod.GET, value =("/{clientId}")  
public BankClientBean getById(@PathVariable String clientId) {  
    for(BankClientBean bcb : getDB())  
        if(bcb.getId().equals(Integer.parseInt(clientId)))  
            return bcb;  
    return new BankClientBean();  
}
```

# Primer 3

- Kreirati REST *endpoint*
  - Koji omogućuje dodavanje novog klijenta
    - putanja **/bankclients**
    - Metoda ispisuje ime i prezime klijenta koji je prosleđen i vraća poruku „*New client added*“
- @RequestBody
  - označava da metoda prihvata parametar koji je deo tela (*body*) web zahteva
- Parametar *method* anotacije *RequestMapping* postaviti na vrednost *RequestMethod.POST*

# Primer 3

```
@RequestMapping(method = RequestMethod.POST)  
public String add(@RequestBody BankClientBean client) {  
    System.out.println(client.getName().concat("  
        ").concat(client.getSurname())));  
    return "New client added";  
}
```

# Primer 3 - testiranje

The screenshot displays a REST client interface with the following components:

- URL Bar:** localhost:8091/bankcl
- Method:** POST
- Params:** localhost:8091/... Params
- Buttons:** Send, Save
- Status:** 200 OK, Time: 277 ms
- Environment:** No Environment
- Body Tab:** Selected, showing the request body.
- Body Type:** JSON (application/json)
- Body Content:**

```
1 {  
2   "id": 2,  
3   "name": "Milan",  
4   "surname": "Celikovic",  
5   "email": "milancel@uns.ac.rs"  
6 }
```
- Response Tab:** Selected, showing the response body.
- Response Content:**

```
1 New client added
```

Red arrows and labels highlight key elements:

- BODY:** Points to the 'Body' tab.
- JSON / TEXT:** Points to the 'JSON (application/json)' type.
- TYPE:** Points to the 'raw' type.

# Primer 4

- Kreirati REST *endpoint*
  - Koji omogućuje izmenu postojećeg klijenta
    - putanja **/bankclients/{clientId}**
    - Ukoliko je prosleđen id sa vrednošću 1 metoda vraća podatke o klijentu sa izmenjenim vrednostima. U suprotnom vraća null vrednost.
- Parametar *method* anotacije *RequestMapping* postaviti na vrednost *RequestMethod.PUT*

# Primer 4

```
@RequestMapping(method = RequestMethod.PUT, value =("/{clientId}")  
public BankClientBean modify(@PathVariable String clientId,  
@RequestBody BankClientBean client) {  
  
    BankClientBean bcb = new BankClientBean(1, "Milan", "Celikovic",  
        "milancel@uns.ac.rs");  
    if(clientId.equals("1")) {  
        bcb.setName(client.getName());  
        return bcb;  
    } else  
        return null;  
}
```

# Primer 5

- Kreirati REST *endpoint*
  - Koji omogućuje brisanje postojećeg klijenta
    - putanja **/bankclients/{clientId}**
    - Ukoliko je prosleđen id sa postojećom vrednosti metoda vraća podatke o klijentu koji je obrisao. U suprotnom vraća objekat sa null vrednostima.
- Parametar *method* anotacije *RequestMapping* postaviti na vrednost *RequestMethod.DELETE*



# Primer 5

```
@RequestMapping(method = RequestMethod.DELETE, value =  
"/{clientId}")  
public BankClientBean delete(@PathVariable String clientId) {  
    for(BankClientBean bcb : getDB())  
        if(bcb.getId().equals(Integer.parseInt(clientId))) {  
            getDB().remove(bcb);  
            return bcb;  
        }  
    return new BankClientBean();  
}
```

# Primer 6

- Kreirati REST *endpoint*
  - Koji vraća klijenta na osnovu vrednosti *name* i *surname*
    - putanja **/bankclients**
    - Ukoliko je prosleđen zahtev sa vašim imenom i prezimenom vratiti objekat koji sadrži vaše ime i prezime. U suprotnom vraća objekat sa null vrednostima.
- @RequestParam
  - označava da metoda prihvata parametar (*Query parameter*) koji je deo zahteva

# Primer 6

```
@RequestMapping(method = RequestMethod.GET, value = "/client")
public BankClientBean getByNameSurname(@RequestParam("name") String
name, @RequestParam("surname") String surname) {
    if(name.equals("Milan") && surname.equals("Celikovic"))
        return new BankClientBean(1, "Milan", "Celikovic",
            "milancel@uns.ac.rs");
    else
        return new BankClientBean();
}
```

# Primer 6 - testiranje

The screenshot shows a REST client interface with a GET request to `localhost:8091/bankcl`. The 'Params' tab is active, displaying a table of query parameters. Two parameters are listed: 'name' with value 'Milan' and 'surname' with value 'Celikovic'. Red arrows point to these parameters, with labels 'QUERYPARAM' and 'PARAMS' respectively.

Key	Value	Description
<input checked="" type="checkbox"/> name	Milan	
<input checked="" type="checkbox"/> surname	Celikovic	
New key	Value	Description

Below the 'Params' tab, the 'Headers' tab is active, showing a table of headers. One header is listed: 'Content-Type' with value 'application/json'. A red arrow points to this header, with the label 'QUERYPARAM'.

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
New key	Value	Description

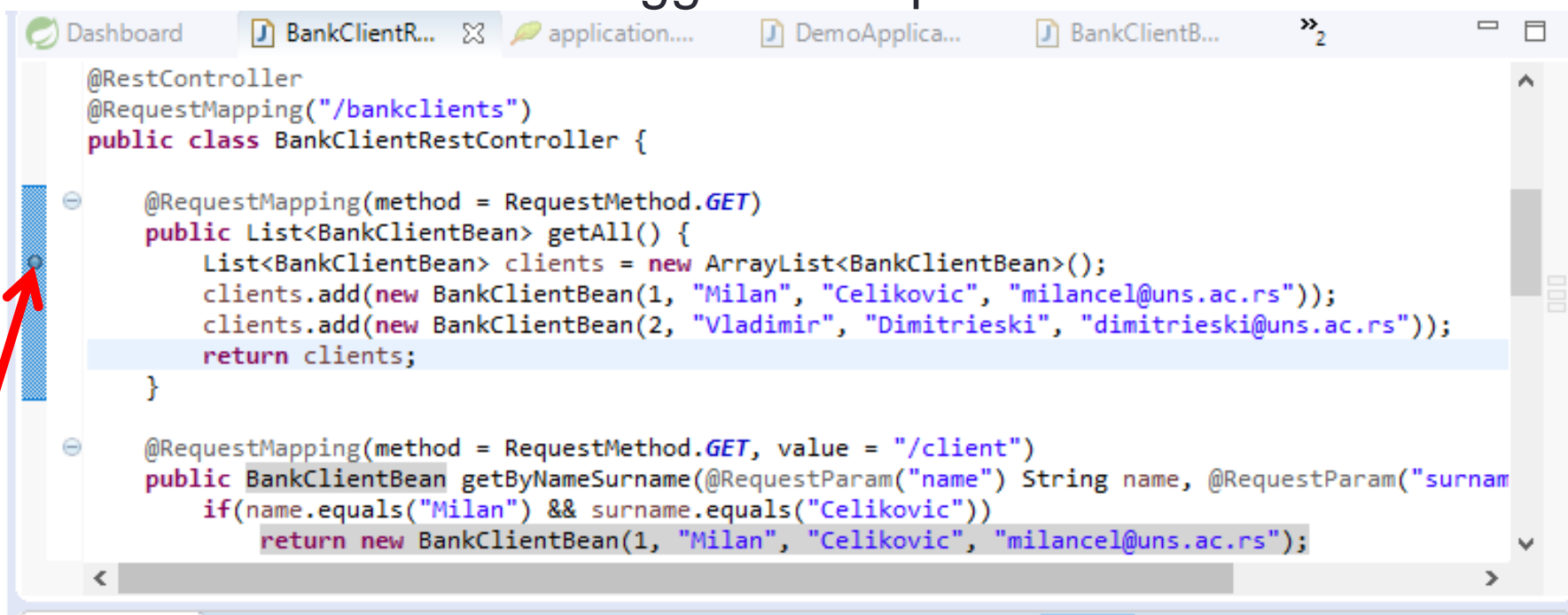
# DEBUGGING

---

Breakpoints

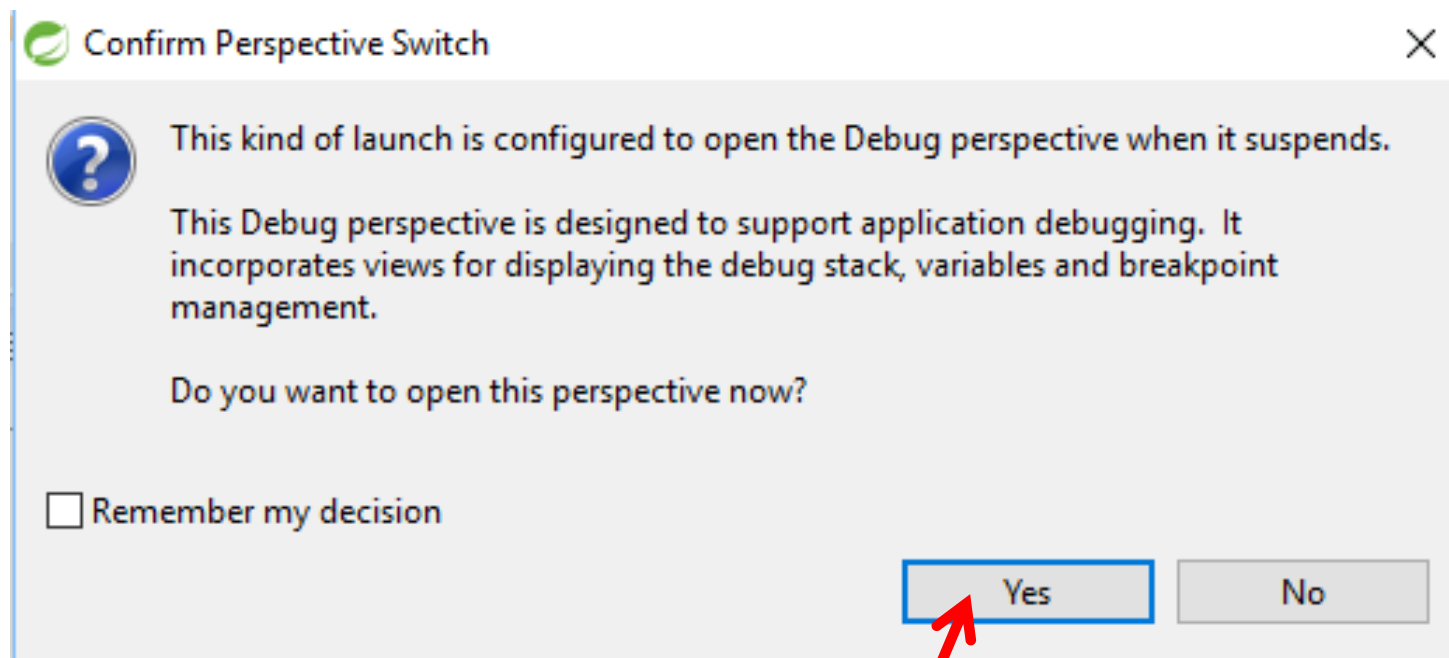
# Tačka prekida

- Zadati tačku prekida na odgovorajućem mestu u programu
- Desni klik mišem -> Toggle Breakpoint



# Debugging

- Pokrenuti STS projekat
  - **Debug As** -> Spring Boot App
  - Potvrditi promenu u Debug perspektivu



# Debugging

**Debug Console:**

- Daemon Thread [http-nio-8091-exec-9] (Suspended (breakpoint at line 21 in BankClientRestController))
  - owns: NioEndpoint\$NioSocketWrapper (id=86)
  - BankClientRestController.getAll() line: 21
  - NativeMethodAccessorImpl.invoke0(Method, Object, Object[]) line: not available [native method]
  - NativeMethodAccessorImpl.invoke(Object, Object[]) line: not available
  - DelegatingMethodAccessorImpl.invoke(Object, Object[]) line: not available
  - Method.invoke(Object, Object...) line: not available
  - ServletHandlerMethod(HandlerMethod).doInvoke(Object...) line: 205
  - ServletHandlerMethod(HandlerMethod).invokeForRequest(NativeWebRequest, Model...

**Variables:**

Name	Value
this	BankClientRestController (id=87)

**Source Code:**

```
@RestController
@RequestMapping("/bankclients")
public class BankClientRestController {

    @RequestMapping(method = RequestMethod.GET)
    public List<BankClientBean> getAll() {
        List<BankClientBean> clients = new ArrayList<BankClientBean>();
        clients.add(new BankClientBean(1, "Milan", "Celikovic", "milancel@uns.ac.rs"));
        clients.add(new BankClientBean(2, "Vladimir", "Dimitrieski", "dimitrieski@uns.ac.rs"));
        return clients;
    }

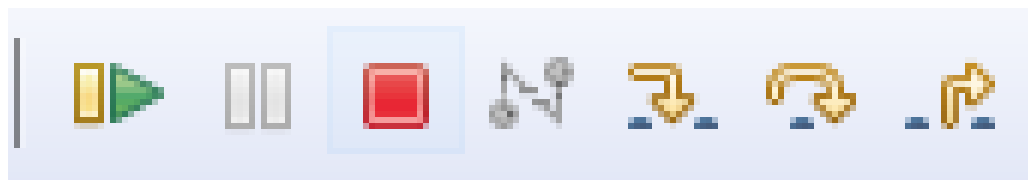
    @RequestMapping(method = RequestMethod.GET, value = "/client")
    public BankClientBean getByNameSurname(@RequestParam("name") String name, @RequestParam("surname") String surname) {
```

**Outline:**

- com.iktpreobuka.myfirstproject.controllers
  - BankClientRestController
    - getAll() : List<BankClientBean>
    - getByNameSurname(String, String) : BankClientBean
    - getById(String) : BankClientBean
    - add(BankClientBean) : String
    - delete(String) : BankClientBean
    - put(String, BankClientBean) : BankClientBean



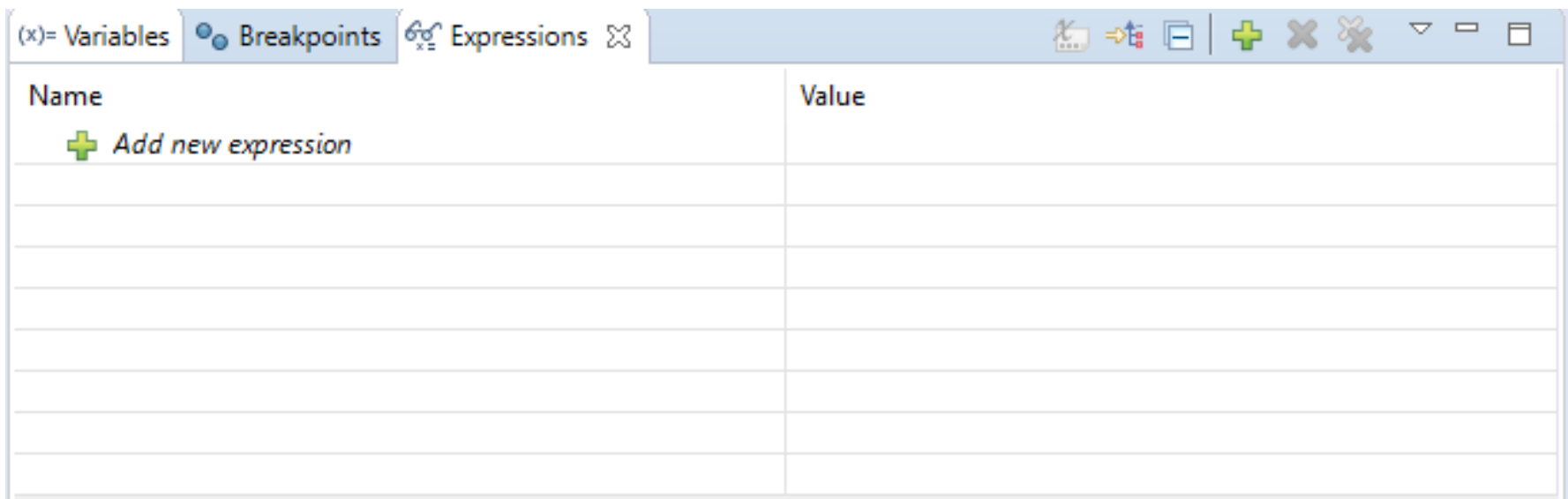
# Debugging



- Nastavi izvršavanje programa
- Pauziraj izvršavanje programa
- Zaustavi izvršavanje programa
- Uđi u pozivajuću metodu
- Pređi na narednu naredbu u programu
- Povratak u metodu iz koje je pozvana

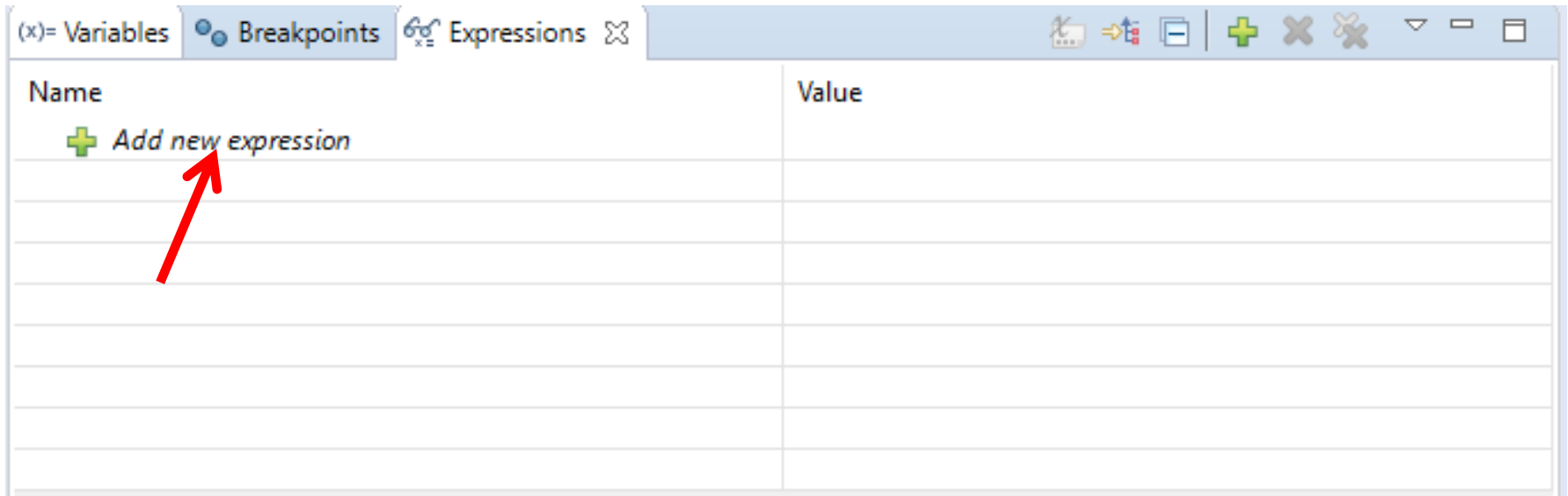
# Debugging

- Po potrebi dodati prozor za nadgledanje vrednosti promenljivih
  - Window -> Show View -> Expressions



# Debugging

- Po potrebi dodati prozor za nadgledanje vrednosti promenljivih
  - Window -> Show View -> Expressions
  - Dodati promenjive / izraze pomoću *Add new expression*



# Debugging

(x)= Variables Breakpoints Expressions

Name	Value
▼ $x+y$ "clients"	(id=93)
▼ ▲ elementData	Object[10] (id=106)
▼ ▲ [0]	BankClientBean (id=108)
> ▲ email	"milancel@uns.ac.rs" (id=118)
> ▲ id	Integer (id=122)
> ▲ name	"Milan" (id=124)
> ▲ surname	"Celikovic" (id=126)
▲ [1]	null
▲ [2]	null
▲ [3]	null
▲ [4]	null
▲ [5]	null
▲ [6]	null
▲ [7]	null
▲ [8]	null
▲ [9]	null
▲ modCount	1
▲ size	1

# ZADACI

---

# Zadatak 1

- Kreirati sledeće REST *endpoints*
  - 1.1 endpoint koji iz liste klijenata banke uzima samo email adrese svih klijenata i vraća listu email adresa
    - putanja **/emails**
  - 1.2 endpoint koji vraća listu koja sadrži imena klijenata, čije ime počinje na slovo koje je prosleđeno kao parametar
    - putanja **/clients/{firstLetter}**
  - 1.3 endpoint koji vraća listu koja sadrži imena i prezimena klijenata, čije ime počinje na slovo koje je prosleđeno kao parametar i čije prezime počinje na slovo koje je prosleđeno kao parametar
    - putanja **/clients/firstLetters**
  - 1.4 endpoint koji vraća listu koja sadrži imena klijenata, koja su sortirana u redosledu koji je prosleđen kao parameter
    - putanja **/clients/sort/{order}**

# Zadatak 2

- Kreirati sledeće REST *endpointe*
  - 2.1 endpoint koji u listi klijenata banke, svakom klijentu, postavlja polje bonitet na 'P' (pozitivan) ako je klijent mlađi od 65 godina ili 'N' negativan ako je klijent stariji od 65 godina
    - putanja **/clients/bonitet**
    - u klasu BankClientBean dodati atribut datum rođenja i bonitet
  - 2.2 endpoint koji briše klijenta iz liste klijenta ukoliko klijent nema jednu od vrednosti: ime, prezime, email
    - putanja **/clients/delete**
  - 2.3 endpoint koji vraća ukupan broj klijenata u listi klijenata koji imaju manje od broja godina koje je prosleđeno kao parametar
    - putanja **/clients/countLess/{years}**
  - 2.4 endpoint koji prosečan broj godina klijenata iz liste klijenata
    - putanja **/clients/averageYears**

# Zadatak 3

- Kreirati sledeće REST *endpointe*
  - 3.1 endpoint koji omogućuje izmenu mesta stanovanja klijenta
    - putanja **/clients/changelocation/{clientId}**
    - u klasu BankClientBean dodati atribut grad
    - novu vrednost mesta stanovanja proslediti kao QueryParameter
  - 3.2 endpoint koji vraća klijente banke koji žive u gradu koji je prosleđen kao parametar
    - putanja **/clients/from/{city}**
  - 3.3 endpoint koji vraća klijente banke koji žive u gradu koji je prosleđen kao parametar i čiji je broj godina ispod broja prosleđenog kao drugi parametar
    - putanja **/clients/findByCityAndAge**



# Zadatak 4

- Izmeniti zadatke 2.1, 3.4 i 3.5 sa prethodnih vežbi tako da korisnik prosleđuje odgovarajuće parametre
  - 4.1 endpoint koji vraća „Hello *yourName*!“, gde *yourName* prosleđeno kao parametar
    - putanja **/greetings/{name}**
  - 4.2 endpoint koji vraća sumu prvih n brojeva
    - putanja **/sumaNiza/{n}**
  - 4.3 endpoint koji predstavlja englesko-srpski rečnik i koji za reč na srpskom vrati odgovarajući prevod na engleski jezik
    - putanja **/recnik/{trazena\_rec}**
    - **DODATNO:** ukoliko za traženu reč ne postoji prevod, tada ispisati „Rec *trazena\_rec* ne postoji u rečniku.“