NACIONĀLAIS ATTĪSTĪBAS PLĀNS 2020
EIROPAS SAVIENĪBA
Eiropas Sociālais fonds
IEGULDĪJUMS TAVĀ NĀKOTNĒ

Java programmēšanas pamati

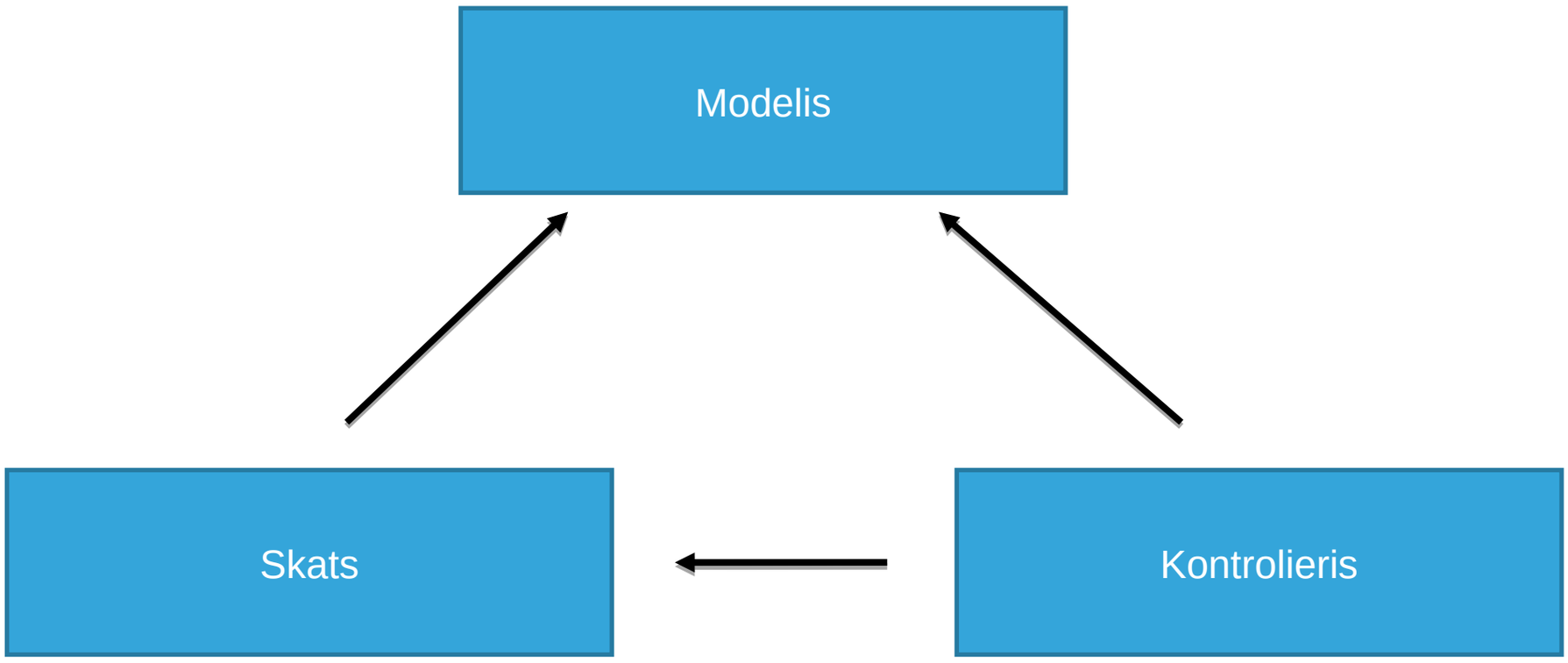
18. NODARBĪBA

MODELIS SKATS KONTROLIERIS

MODELIS – SKATS – KONTROLIERIS (MVC)

- MVC ir programmas arhitektūras veidošanas paterns
- MVC ļauj atdalīt lietojumprogrammu trīs loģiskās komponentēs:
  - modelis
  - skats
  - kontrolieris

MODELIS – SKATS – KONTROLIERIS (MVC)



MODELIS – SKATS – KONTROLIERIS (MVC)

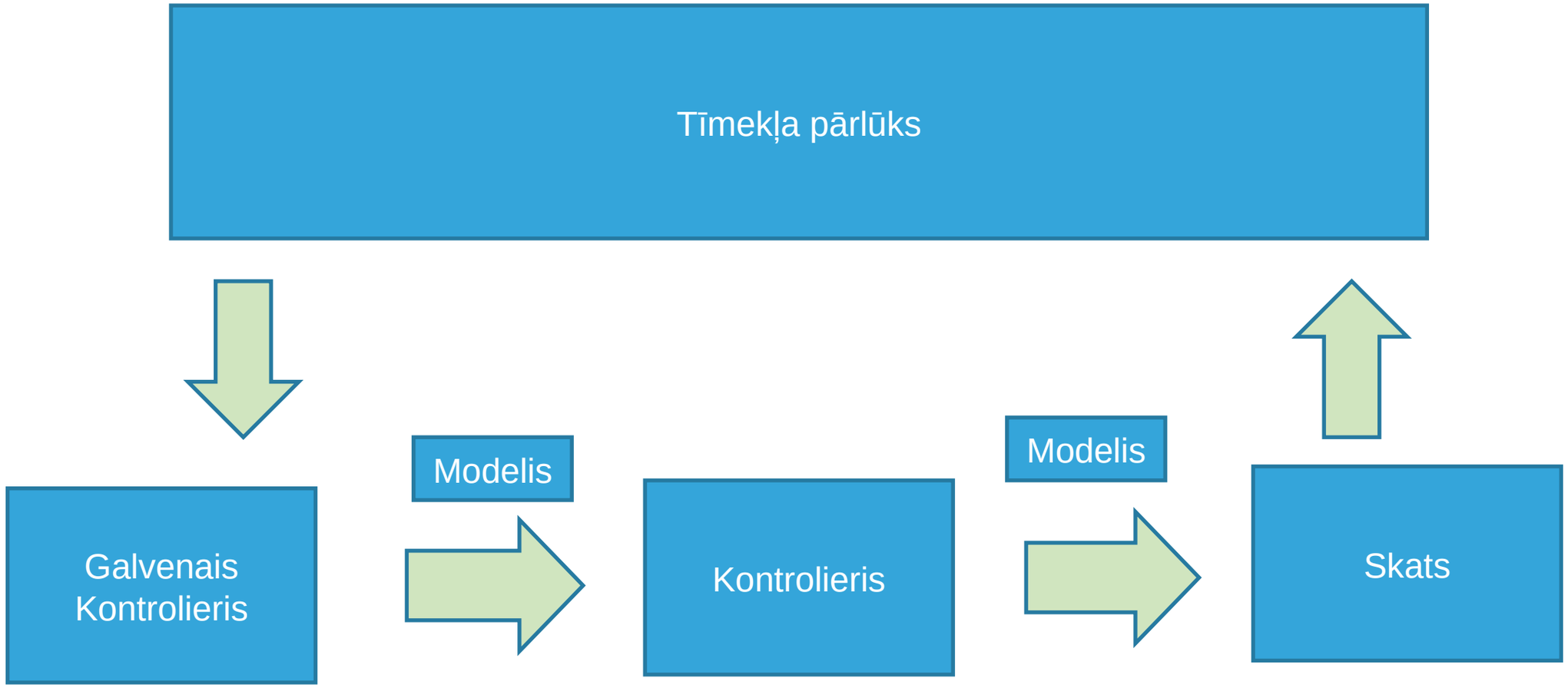
- **Modelis** – šai komponentei atbilst visa ar datiem saistītā loģika. Šī komponente atspoguļo datus, kas tiek nodoti starp **Skatu** un **Kontrolieri** vai arī jebkuri citi biznesa loģikas dati
- **Skats** – Šī komponente ir atbildīga par lietotāja saskarnes datiem, par to kā modeļa dati tiks lietotājam attēloti
- **Kontrolieris** – Šī komponente ir atbildīga par ienākošā pieprasījuma saņemšanu, biznesa loģikas izpildi un atbilstošas izvades radīšanu.

SPRNIG MVC

SPRING MVC

- Spring MVC ir Spring freimvorka modulis, kas ir īstenojis Model-View-Controller paternu
- Apkopo visas MVC paterna priekšrocības ar Spring vienkāršību
- Spring implementē MVC izmantojot «galveno kontrolieri» - DispatchServlet

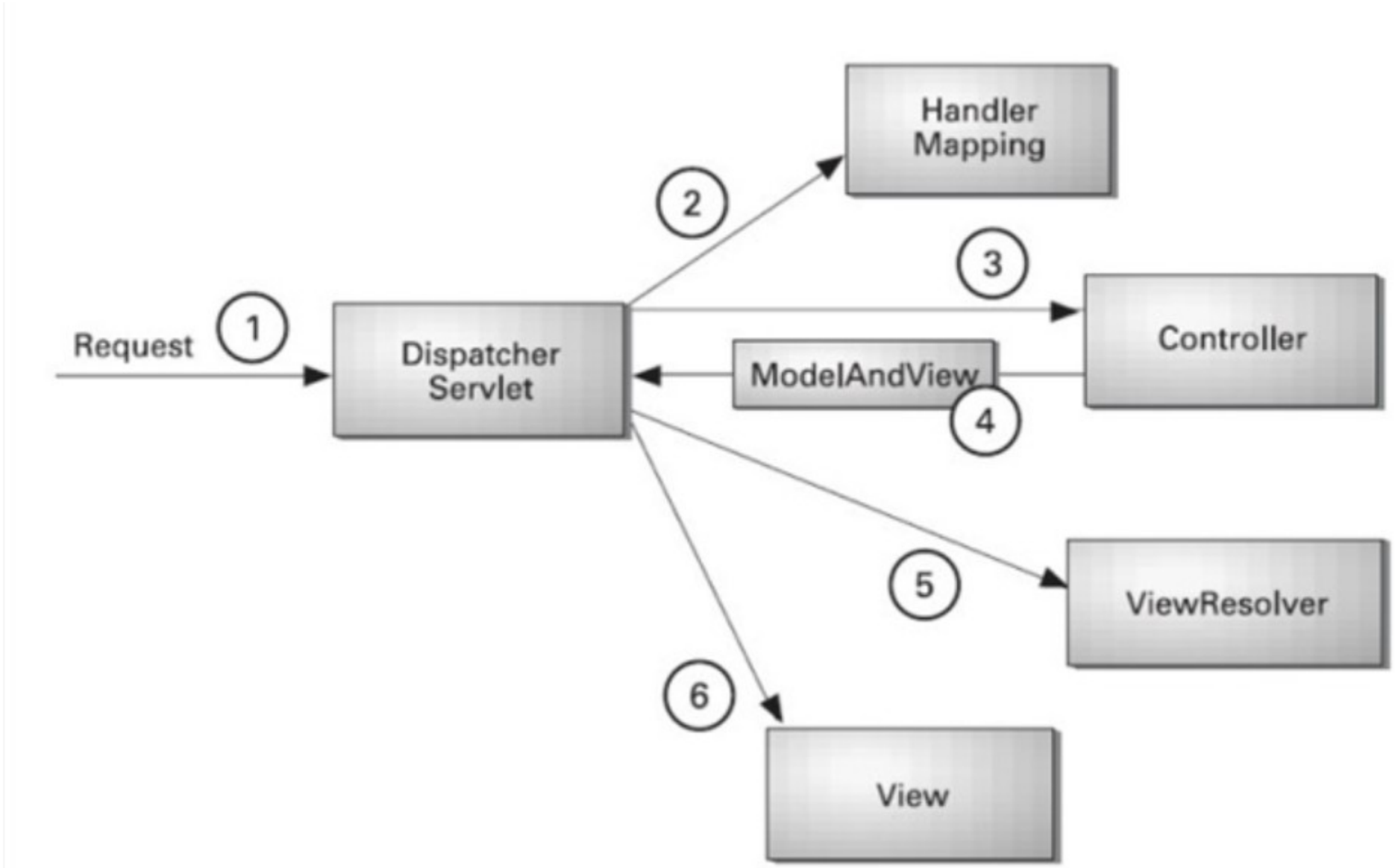
SPRING MVC SHĒMA



SPRING MVC

- **Model** – modelis satur lietojumprogrammas datus. Dati var būt viens objekts vai arī objektu masīvs
- **Controller** – kontrolieris satur lietojumprogrammas biznesa loģiku. Tā ir klase, kura apzīmēta ar @Controller anotāciju
- **View** – skats raksturo veidu kā informācija tiek atspoguļota lietotājam. Tiek izmantotas tehnoloģijas kā JSP, JSF u.c., lai izveidotu datu attēlošanu html veidā.
- **Front Controller** – ir Spring MVC – DispatchServletClass. Ir atbildīgs par visas Spring MVC aplikācijas plūsmas pārvaldību

SPRING PLŪSMA



SPRING PLŪSMA

- Visi ienākošie pieprasījumi tiek pārtverti ar DispatchServlet
- Tā kā DispatchServlet satur visus «handlerus», tad tas nodod pieprasījumu atbilstošajam kontrolierim
- Kontrolieris atgriež ModelAndView objketu
- DispatchServlet atrod atbilstošo skata «resolveri» un nodod tam objektu datu attēlošanai

SPRNIGBOOT


spring
BOOT

PĀRSKATS

- Spring-Boot dod iespēju viegli un ātri izveidot produkcijas vides kvalitātes Spring bāzētu lietojumprogrammu
- Spring-Boot lietojumprogrammām Spring konfigurācija ir ļoti minimāla

IEZĪMES

- Izveido patstāvīgu (stand-alone) Spring lietojumprogrammu
- Iekļauts Tomcat, Jetty, vai Undertow aplikāciju serveris
- Piedāvā «starter» atkarības, lai vienkāršotu jūsu «build» konfigurāciju
- Automātiski nokonfigurē Spring un 3ās puses bibliotēkas
- Sagatavo produkcijai gatavas iezīmes – metrikas, veselības pārbaude un ārēja konfigurācija
- Izslēdz nepieciešamību pēc XML konfigurācijas

IESĀC AR SPRING INITUALIZR

- Izvēlies vēlamo konfigurāciju savam projektam no https://start.spring.io/
- Uzģenerē savu projekta mapi
- Importē to savā IDEA

BUILD.GRADLE

```
plugins {
	id 'org.springframework.boot' version '2.4.3'
	id 'io.spring.   ' version '1.0.11.RELEASE'
	id 'java'
}

group = 'com.example'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '1.8'

repositories {
	mavenCentral()
}

dependencies {
	implementation 'org.springframework.boot:spring-boot-starter-web'
	testImplementation 'org.springframework.boot:spring-boot-starter-test'
}

test {
	useJUnitPlatform()
}
```

SPRING PALAIŠANA

- Metodē «main» tiek izsaukt SpringApplication metode run
- SpringApplication piestartē mūsu programmu un automātiski konfigurēto aplikāciju serveri (Tomcat)

```
package com.example.demo;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class DemoApplication {

	public static void main(String[] args) {
		SpringApplication.run(DemoApplication.class, args);
	}

}
```

CLASSPATH ATKARĪBAS

- SpringBoot nodrošina vairākus «Startētājus» (starter), kas pievieno nepieciešamās bibliotēkas Jūsu classpath
- spring-boot-starter-web iekļauj sevī komponentes, kas ir nepieciešamas tīmekļa programmai, piemēram Tomcat serveris
- spring-boot-starter-test iekļauj nepieciešamās komponentes tīmekļa aplikācijas testēšanai
- Spraudnis dependency-management nodrošina šo atkarību pārvaldību

ATKARĪBU PĀRVALDĪBA

- Katrā Spring Boot laidienā ir apkopots to atbalstīto atkarību saraksts
- Praksē nav nepieciešams norādīt precīzu atkarības versiju, jo SpringBoot to pārvalda Jūsu vietā
- Kad tiek atjaunota SpringBoot versija, tad tiek atjaunotas arī atkarību versijas
- Bet ir atļauts arī norādīt sev vēlamo versiju atkarībai, ja tas ir nepieciešams

REST

REST

- REST ir akronīms no **RE**presentational **S**tate **T**ransfer – Reprezentatīva stāvokļa nodošana
- REST ir programmas arhitektūras stils, kas nosaka kā API ir jāizskatās
  - API nosaka kā ar Jūsu programmai ir iespējams apmainīties ar datiem
- REST nosaka 6 ierobežojumus

5 REST DIZAINA PRINCIPI

- Klients-Serveris – atdali lietotāja interfeisu no datu bāzes
- Bezstāvokļa (stateless) – katrs pieprasījum no klienta satur visu nepieciešamo informāciju tā izpildei. Serveris netiek glabāts stāvoklis.
- Kešojams (cachable) – pieprasījuma vai atbildes dati var tikt kešoti
- Vienota saskarne (uniform interface) – saskarnei ir jānodrošina standartizēts un vienots komunikācijas veids starp klientu un serveri
- Slāņveida sistēma (Layered system) – slāņveida sistēmā komponentēm ir ierobežota redzamība, tā lai tās neredz tālāk par savu darbības lauku.

ATSAUCES

- https://www.javatpoint.com/spring-mvc-tutorial
- https://www.mulesoft.com/resources/api/what-is-rest-api-design
- https://florimond.dev/en/posts/2018/08/restful-api-design-13-best-practices-to-make-your-users-happy/
- https://spring.io

LATVIJAS UNIVERSITATE
ANNO 1919

VADĪBAS UN UZŅĒMĒJDARBĪBAS