SOUTH EASTERN EUROPE

asseco
SOUTH EASTERN EUROPE

Testiranje Java EE aplikacija s Arquillianom

Igor Vlahek
Software architect
Igor.vlahek@asseco-see.hr

Rovinj, 11.5.2017

asseco
SOUTH EASTERN EUROPE

Testiranje Java EE aplikacija s Arquillianom

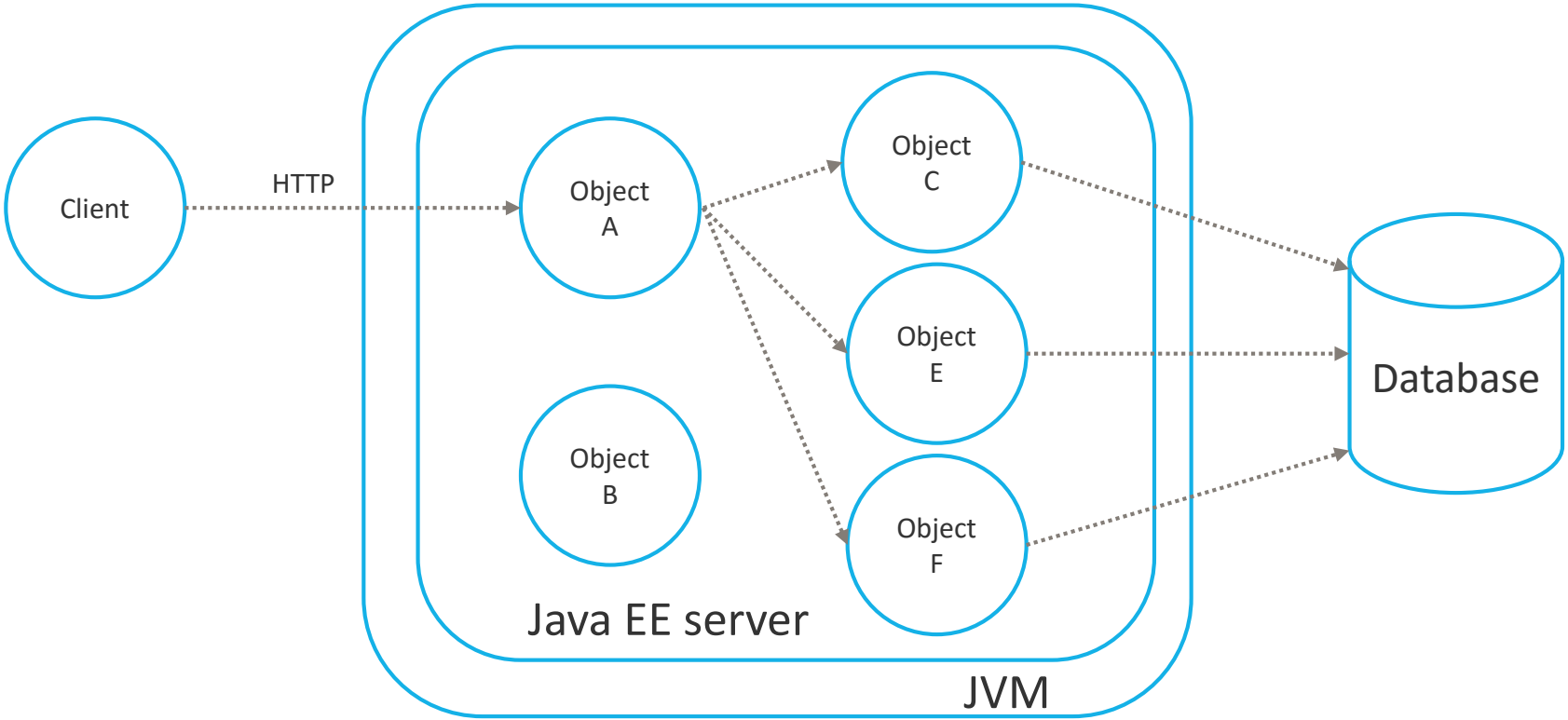
1. Java EE aplikacije
2. Testiranje Java EE aplikacije
3. Arquillian
4. Arquillian embedded container showcase
5. Arquillian remote container showcase

Java EE

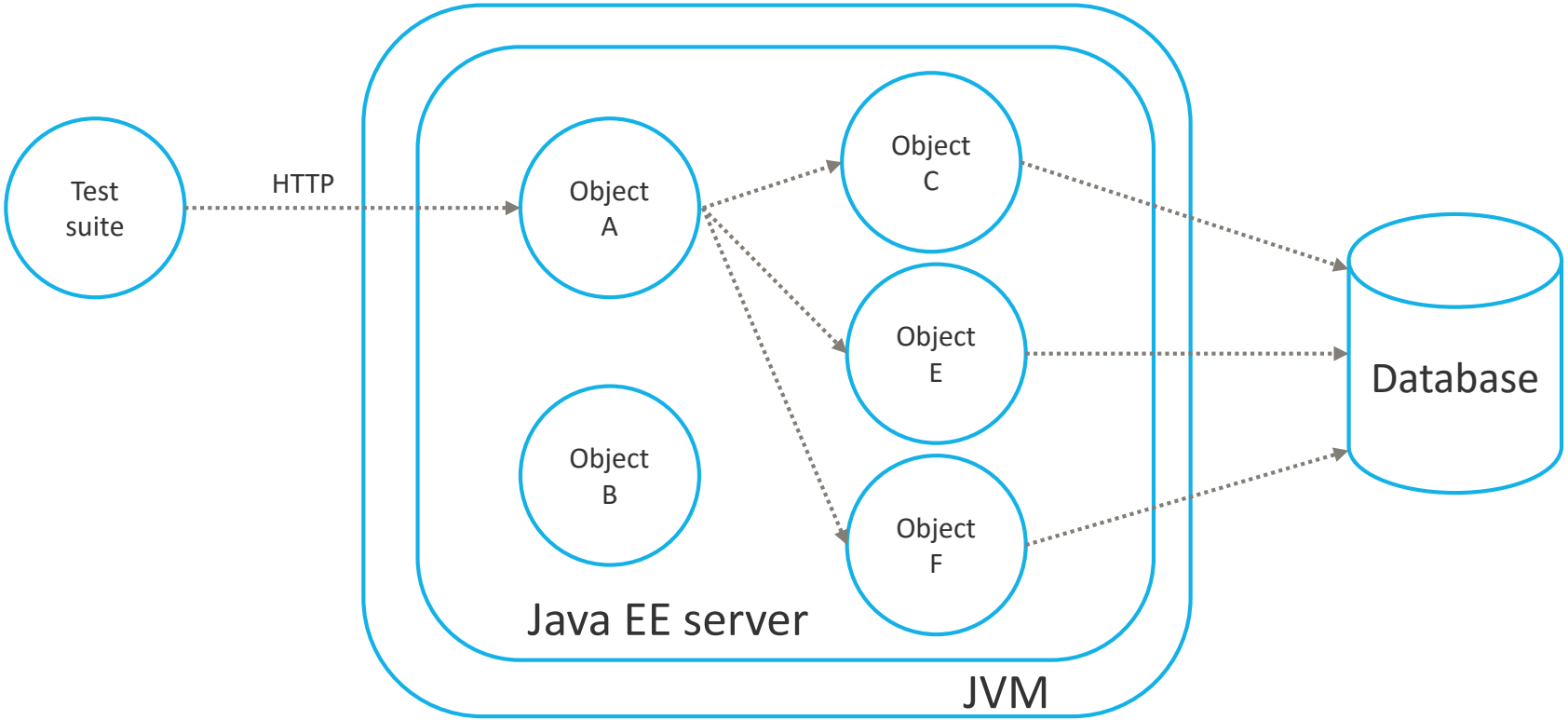
Skup specifikacija

Enterprise Application Technologies			
Batch Applications for the Java Platform	JSR 352	Download spec	
Concurrency Utilities for Java EE 1.0	JSR 236	Download spec	
Contexts and Dependency Injection for Java 1.1	JSR 346	Download spec	✓
Dependency Injection for Java 1.0	JSR 330	Download spec	✓
Bean Validation 1.1	JSR 349	Download spec	✓
Enterprise JavaBeans 3.2	JSR 345	Download spec	✓
Interceptors 1.2 (Maintenance Release covered under JSR 318)	JSR 318	Download spec	✓
Java EE Connector Architecture 1.7	JSR 322	Download spec	
Java Persistence 2.1	JSR 338	Download spec	✓
Common Annotations for the Java Platform 1.2	JSR 250	Download spec	✓
Java Message Service API 2.0	JSR 343	Download spec	
Java Transaction API (JTA) 1.2	JSR 907	Download spec	✓
JavaMail 1.5	JSR 919	Download spec	
Web Services Technologies			
Java API for RESTful Web Services (JAX-RS) 2.0	JSR 339	Download spec	✓
Implementing Enterprise Web Services 1.3	JSR 109	Download spec	
Java API for XML-Based Web Services (JAX-WS) 2.2	JSR 224	Download spec	
Web Services Metadata for the Java Platform	JSR 181	Download spec	
Java API for XML-Based RPC (JAX-RPC) 1.1 (Optional)	JSR 101	Download spec	
Java APIs for XML Messaging 1.3	JSR 67	Download spec	
Java API for XML Registries (JAXR) 1.0	JSR 93	Download spec	
Management and Security Technologies			
Java Authentication Service Provider Interface for Containers 1.1	JSR 196	Download spec	
Java Authorization Contract for Containers 1.5	JSR 115	Download spec	

Java EE aplikacija



Testiranje Java EE aplikacije



asseco
SOUTH EASTERN EUROPE

Testiranje Java EE aplikacije (2)

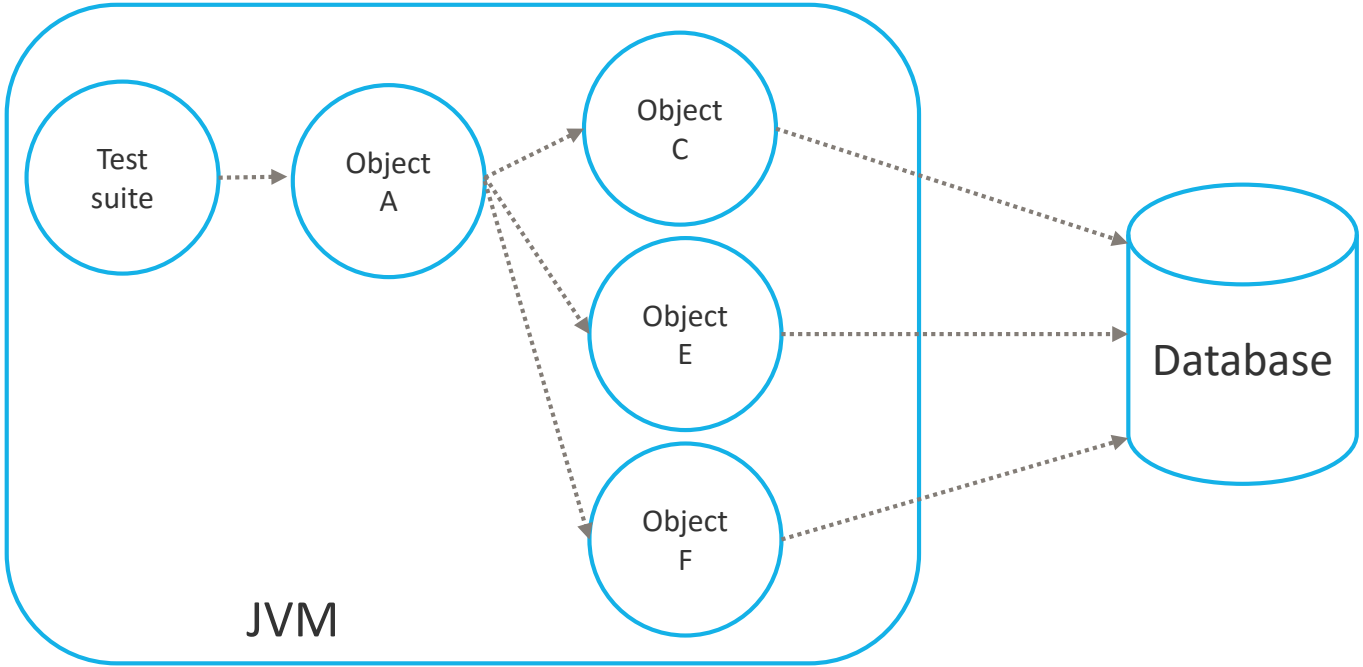
```
@Test
public void should_create_city() {
    CityDTO cityDTO = CityDTOBuilder.aCityDTO().withPostalCode("10090").withCountryId(country1Organization1.getId()).build();

    //OPERATE
    CityDTO data = restClient.postToResource(cityDTO, CityDTO.class);

    //CHECK
    City createdCity = cityRepository.findById(data.getId());
    assertThat(data.getPostalCode()).isEqualTo(createdCity.getPostalCode());
    assertThat(data.getName()).isEqualTo(createdCity.getName());
    assertThat(data.getCountry().getId()).isEqualTo(cityDTO.getCountry().getId());
}
```

Testiranje Java EE aplikacije (3)

Prednosti i mane

- Prednosti
  - → Testiranje aplikacije u cijelosti
  - → Testiranje kako će se aplikacija ponašati na stvarnom serveru
  - → Neophodno kod razvijanja EE aplikacija
- Nedostaci
  - → Užasno sporo
  - → Interna arhitektura može patiti jer nema testiranja sloja po sloj

Testiranje Java EE aplikacije sloj po sloj (1)



Testi

```
@Test
public voi
    //OPE
    entity
    interr
    entity

    //CHE
    entity
    assert
    assert
    final
    assert
}
```

```
public InternalTokenService createInternalTokenService() throws Exception {
    InternalTokenService internalTokenService = new InternalTokenService();
    internalTokenService.setUserAuthPolicyDao(userAuthPolicyDao);
    internalTokenService.setArchiveReasonDao(archiveReasonDao);
    internalTokenService.setArchiveSlotDAO(archiveSlotDao);
    internalTokenService.setArchiveTokenDao(archiveTokenDao);
    internalTokenService.setArchiveTokenToServiceDao(archiveTokenToServiceDao);
    internalTokenService.setArchiveTokenToSxsUserDao(archiveTokenToSxsUserDao);
    internalTokenService.setPrintQueueDao(printQueueDao);
    internalTokenService.setPrintQueueHistoryDao(printQueueHistoryDao);
    internalTokenService.setTokenTypeToDocumentTypeDao(tokenTypeToDocumentTypeDao);
    internalTokenService.setTokenDao(tokenDao);
    internalTokenService.setTokenServiceDao(tokenServiceDao);
    internalTokenService.setSlotDAO(slotDao);
    internalTokenService.setTokenStatusDao(tokenStatusDao);
    internalTokenService.setMasker(createCapMaskerFacade());
    internalTokenService.setCrypto(createCryptoProvider());
    internalTokenService.setMapper(createModelMapper());
    internalTokenService.setSxsUserDao(sxsUserDao);
    internalTokenService.setAdminUserToLoginModeDao(adminUserToLoginModeDao);
    internalTokenService.setSession(createThreadSession());
    internalTokenService.setInternalMobileApplicationService(createInternalMobileApplicationService());
    return internalTokenService;
}
```

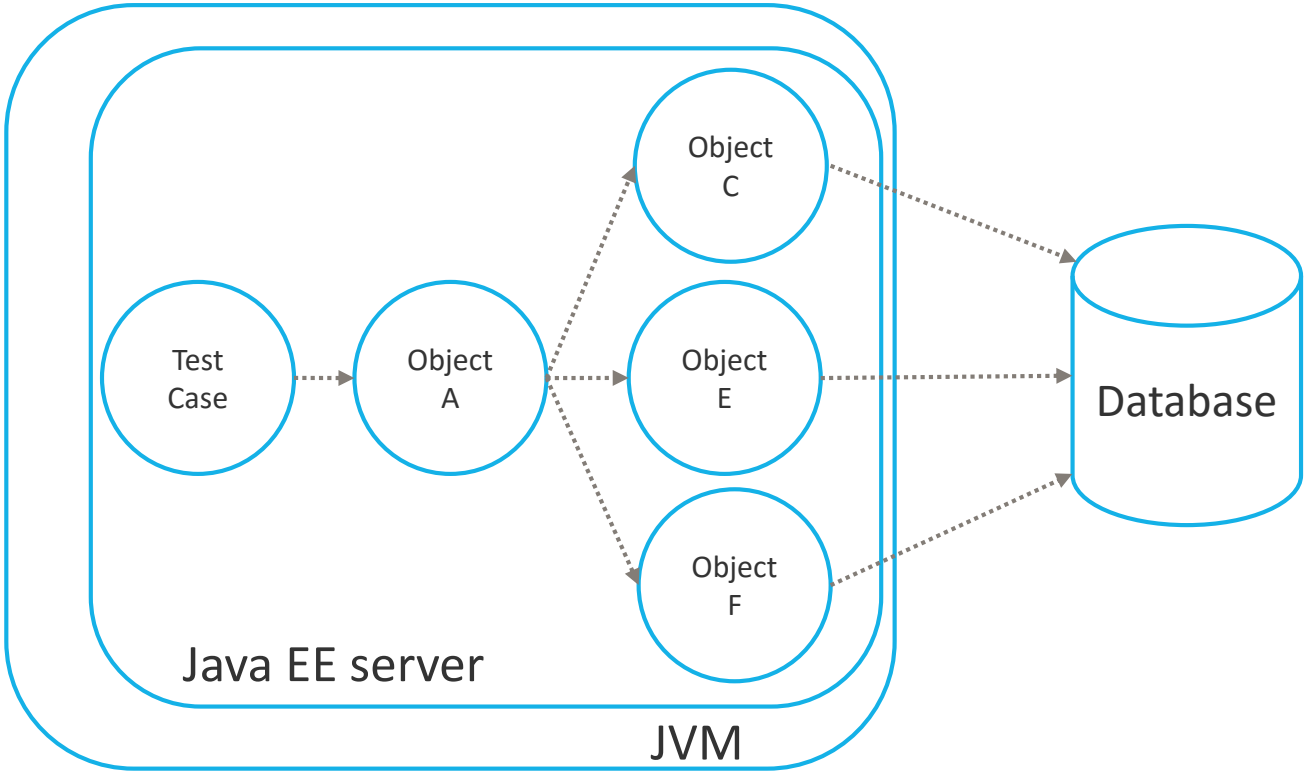
```
i());
```

Testiranje Java EE aplikacije sloj po sloj (3)

Prednosti i mane

- Prednosti
  - → Testiranje aplikacije sloj po sloj
  - → Brže jer se ne pakira cijela aplikacija
  - → Brže jer nije potreban deploy aplikacije na Java EE server
- Nedostaci
  - → Simuliramo mogućnosti Java EE servera
  - → Ne znamo da li će se naša aplikacija tako ponašati kada ju postavimo na Java EE server
  - → Simuliranje mogućnosti Java EE servera zahtjeva mnogo dodatnog posla

In-container testiranje (1)



In-container testiranje (2)

```java
/**
 * Created by ivlahek on 23.6.2016..
 */
public class CountryRepositoryTest extends ArquillianBaseClass {

    @Inject
    protected CountryRepository countryRepository;

    @Test
    public void should_create_country() {
        //BUILD
        Country country = CountryBuilder.aCountry().build();

        //OPERATE
        countryRepository.persist(country);

        //CHECK
        assertThat(country.getId()).isNotNull();
    }
```

In-container testiranje (3)

Prednosti i mane

- Prednosti
  - → Testiranje aplikacije sloj po sloj
  - → Brže jer se ne pakira cijela aplikacija
  - → Testiramo aplikaciju u pravom Java EE containeru
  - → Test postaje dio aplikacije, test ima pristup svim Java EE resursima
- Nedostaci
  - → Postavljanje testa (pakiranje) zahtjeva ponekad mnogo posla
  - → Inicijalno postavljanje u mavenu zna mnogo vremena oduzet

asseco
SOUTH EASTERN EUROPE

Arquillian testing framework

„Arquillian is all about testing your code inside a container”

- Skup biblioteka koji omogućuju izvršavanje testova unutar Jave EE container-a
- Velik skup dostupnih adapter-a prilagođen svakom containeru
- Velik broj extenzija (Jacoco)
- Omogućuje pisanje automatiziranih Junit testova
- Arquillian – container interakcija
  - → **Embedded containers**
  - → Managed containers
  - → **Remote containes**

Pisanje Java testova

```java
@RunWith(Arquillian.class)
public class GreeterTest {

  @Inject
  Greeter greeter;

  @Deployment
  public static WebArchive createDeployment() {
    WebArchive jar = ShrinkWrap.create(WebArchive.class)
        .addClasses(Greeter.class, PhraseBuilder.class)
        .addAsWebInfResource(EmptyAsset.INSTANCE, "beans.xml");
    System.out.println(jar.toString(true));
    return jar;
  }

  @Test
  public void should_create_greeting() {
    Assert.assertEquals("Hello, Earthling!", greeter.createGreeting("Earthling"));
    greeter.greet(System.out, "Earthling");
  }
}
```

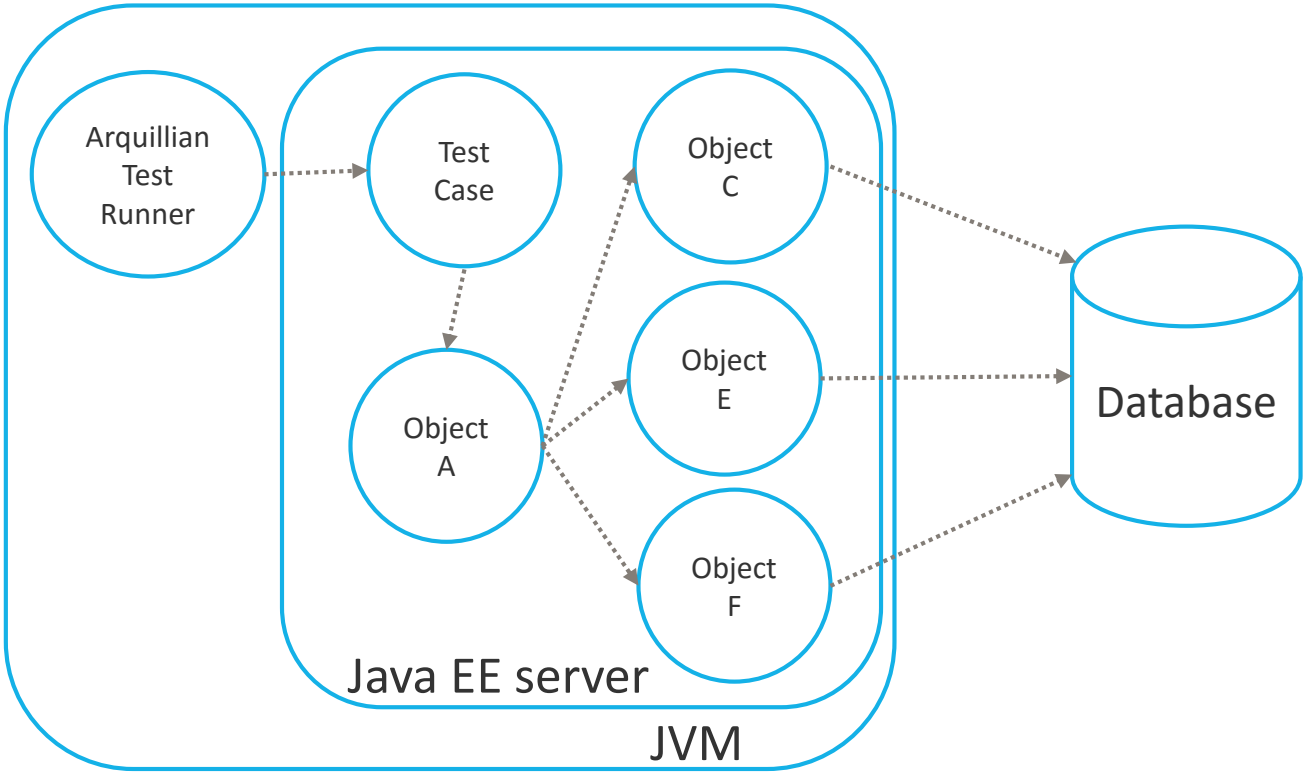
Postavljanje Arquillian-a

```xml
<profile>
    <id>wildfly</id>
    <activation>
        <activeByDefault>true</activeByDefault>
    </activation>
    <dependencies>
        <!--OPTIONAL DEPENDENCIES - DEPENDS OF THE CONTAINER ON WHICH WE WISH TO RUN TESTS-->
        <dependency>
            <groupId>org.wildfly</groupId>
            <artifactId>wildfly-arquillian-container-remote</artifactId>
            <version>8.2.0.Final</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.jboss.arquillian.protocol</groupId>
            <artifactId>arquillian-protocol-servlet</artifactId>
            <version>1.1.13.Final</version>
        </dependency>
        <!--OPTIONAL DEPENDENCIES - DEPENDS OF THE CONTAINER ON WHICH WE WISH TO RUN TESTS-->

        <dependency>
            <groupId>org.jboss.shrinkwrap.resolver</groupId>
            <artifactId>shrinkwrap-resolver-impl-maven</artifactId>
            <version>2.2.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.jboss.arquillian.junit</groupId>
            <artifactId>arquillian-junit-container</artifactId>
            <version>1.1.2.Final</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</profile>
```

```xml
<profile>
    <id>weld</id>
    <!--<activation>-->
    <!--<activeByDefault>true</activeByDefault>-->
    <!--</activation>-->
    <dependencies>
        <!--OPTIONAL DEPENDENCIES - DEPENDS OF THE CONTAINER ON WHICH WE WISH TO RUN TESTS-->
        <dependency>
            <groupId>org.jboss.arquillian.container</groupId>
            <artifactId>arquillian-weld-ee-embedded-1.1</artifactId>
            <version>1.0.0.CR9</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.jboss.weld</groupId>
            <artifactId>weld-core</artifactId>
            <version>1.1.33.Final</version>
            <scope>test</scope>
        </dependency>
        <!--OPTIONAL DEPENDENCIES - DEPENDS OF THE CONTAINER ON WHICH WE WISH TO RUN TESTS-->

        <!--MANDATORY DEPENDENCIES - MUST BE INCLUDED IN EVERY PROJECT USING ARQUILLIAN-->
        <!--SHRINKWRAP IS USED TO BUILD ARTIFACT THAT IS GOING TO BE DEPLOYED IN THE CONTAINER-->
        <dependency>
            <groupId>org.jboss.shrinkwrap.resolver</groupId>
            <artifactId>shrinkwrap-resolver-impl-maven</artifactId>
            <version>2.2.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.jboss.arquillian.junit</groupId>
            <artifactId>arquillian-junit-container</artifactId>
            <version>1.1.2.Final</version>
            <scope>test</scope>
        </dependency>
        <!--MANDATORY DEPENDENCIES - MUST BE INCLUDED IN EVERY PROJECT USING ARQUILLIAN-->
    </dependencies>
</profile>
```

Način rada

Embedded containers



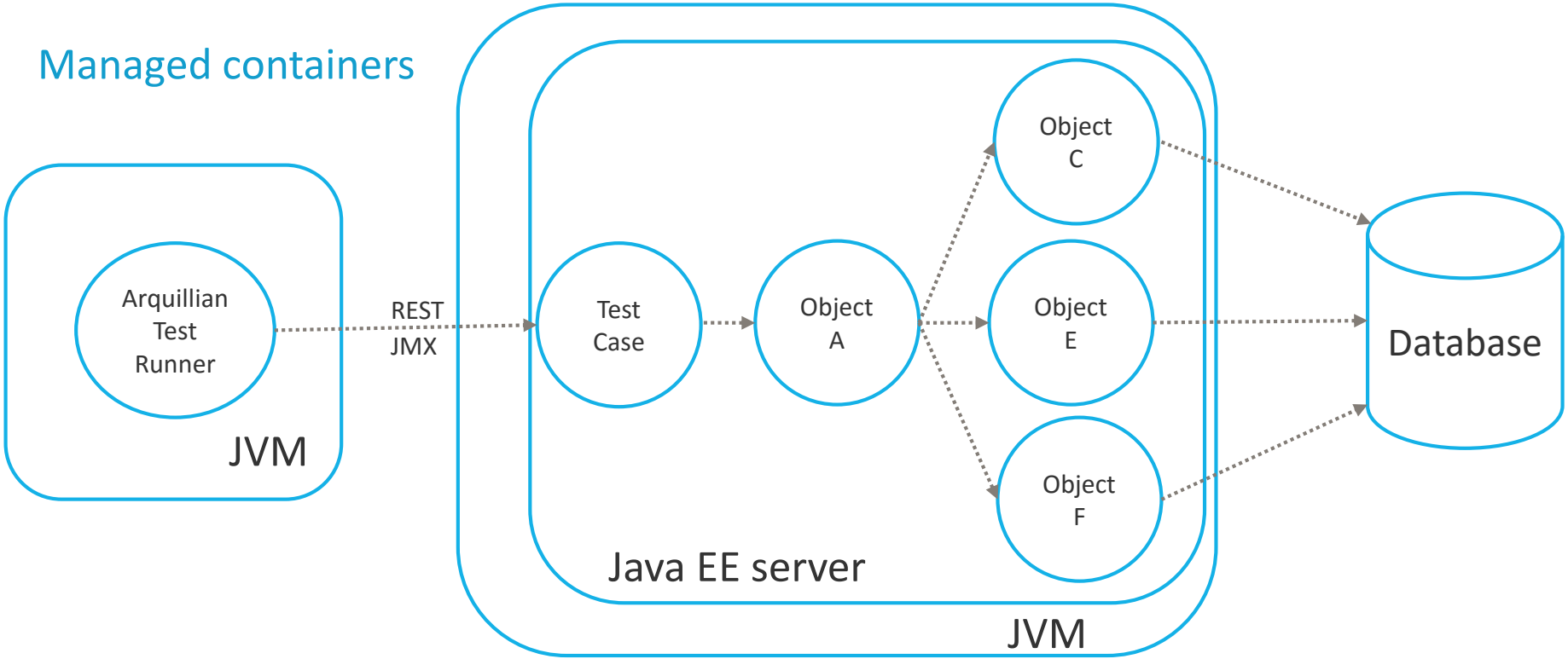
asseco
SOUTH EASTERN EUROPE

Showcase embedded containers

https://github.com/ivlahek/JavaCRO2017ArquillianWeld

Način rada

Managed containers



asseco
SOUTH EASTERN EUROPE

Showcase managed containers

https://github.com/ivlahek/JavaCRO2017ArquillianWildfly

Solutions for demanding business.