HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

05 - AOP & testing

Managed components are powerful… but how do we test them?

AMT 2019

Olivier Liechti

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

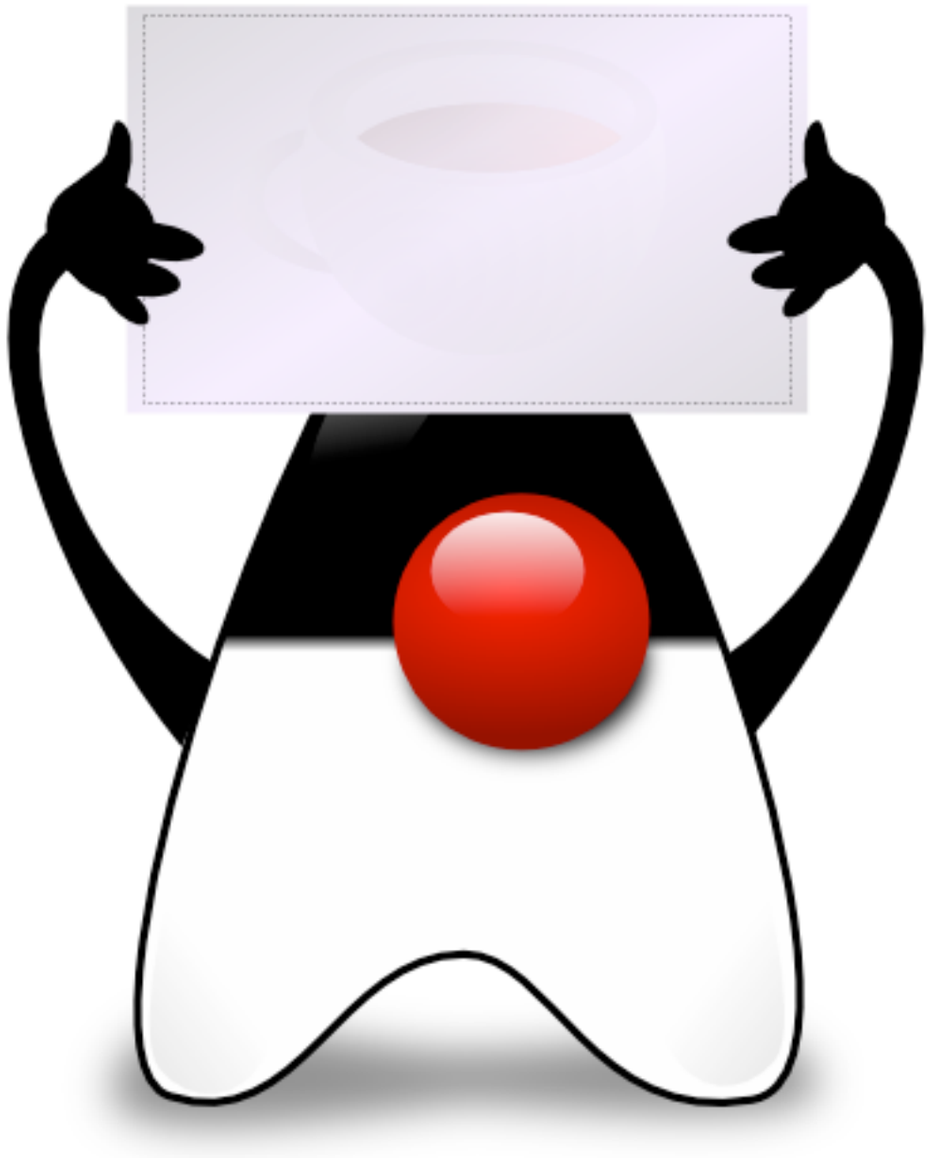
Quick recap…
The Power of Aspect Oriented Programming

Break

The Curse of testing in managed environments

Project time

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Quick recap

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is inversion of control? (IoC)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is dependency injection? (DI)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is resource pooling and why is it useful?

Explain with 2 examples

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is the goal of the Data Access Object pattern?

(DAO)

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Why do we have to be careful when using HttpSession?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Is a servlet thread-safe? What about a stateless session bean?

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is the difference between a stateless session bean and a singleton EJB?

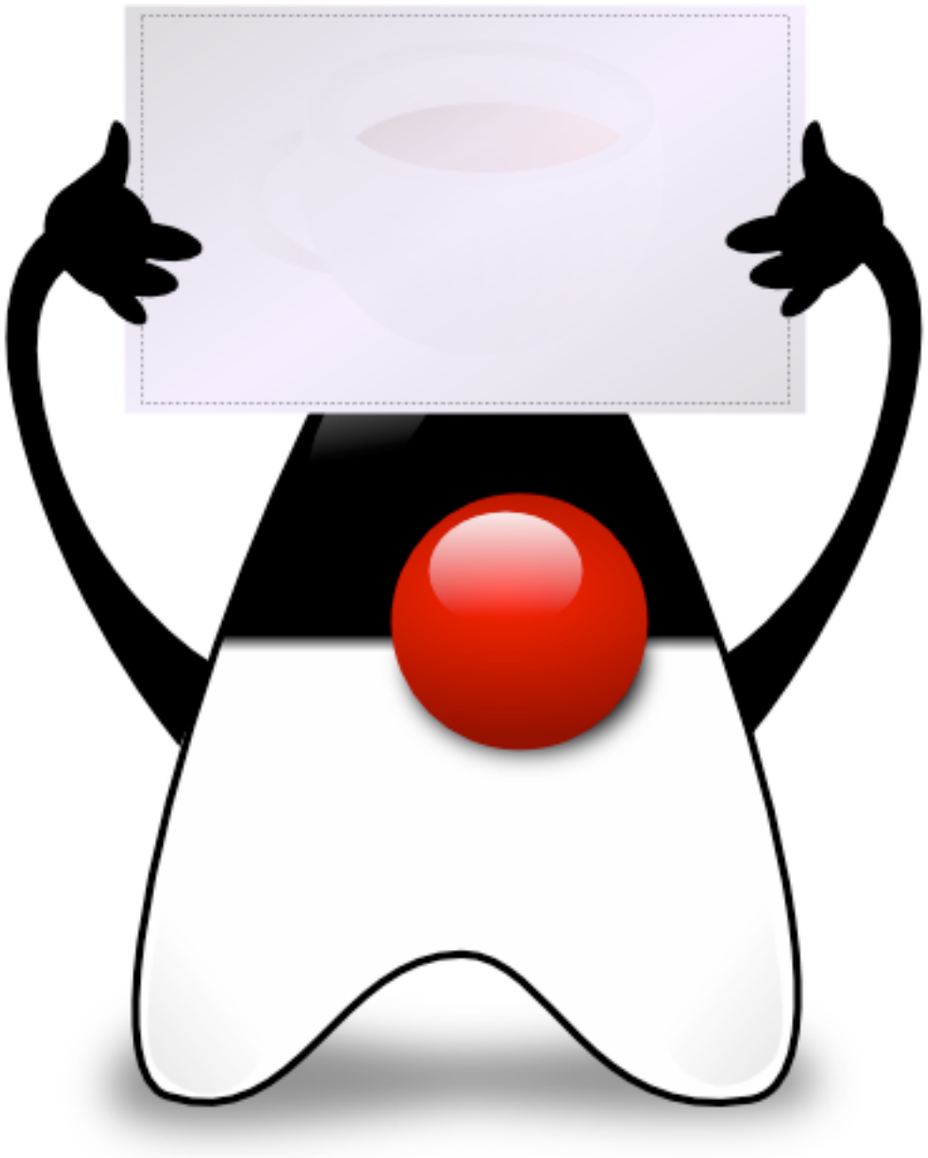
heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

What is reflection in Java?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

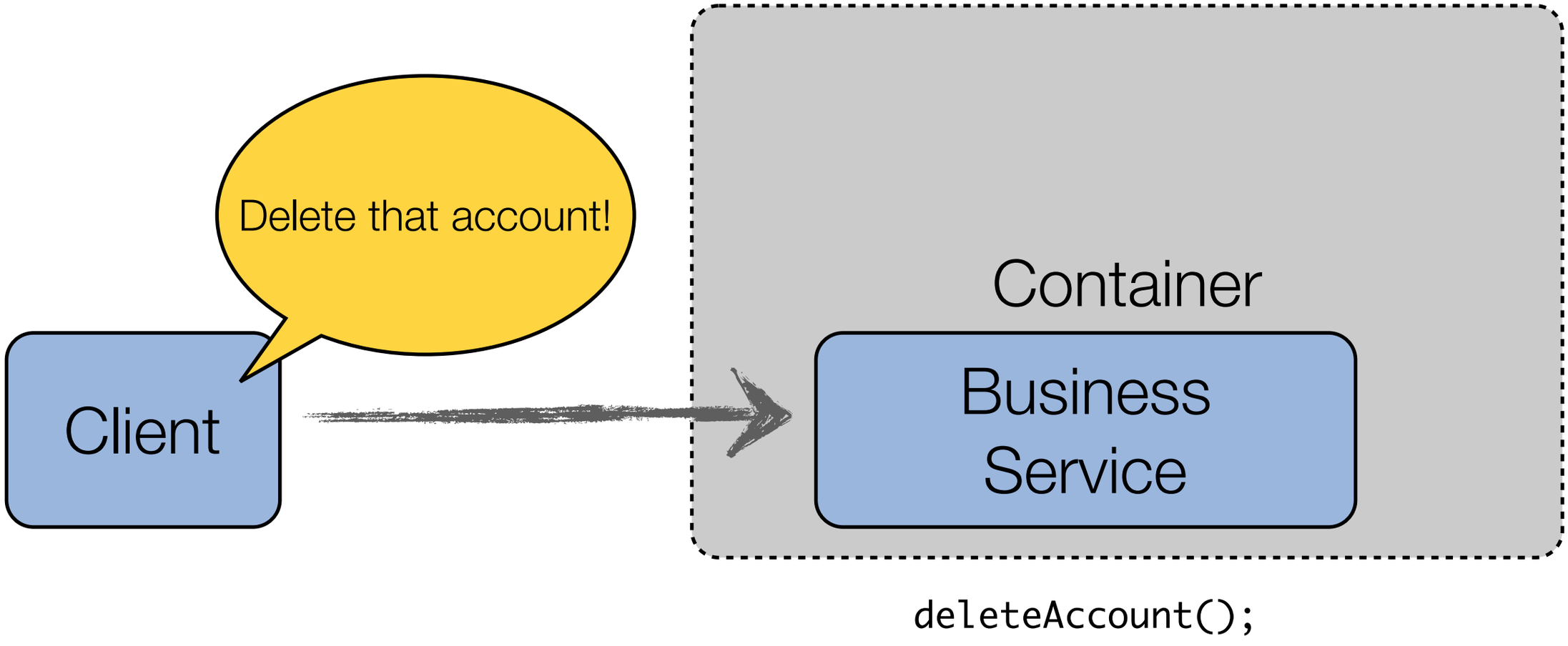
What do we mean by "managed component"?

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



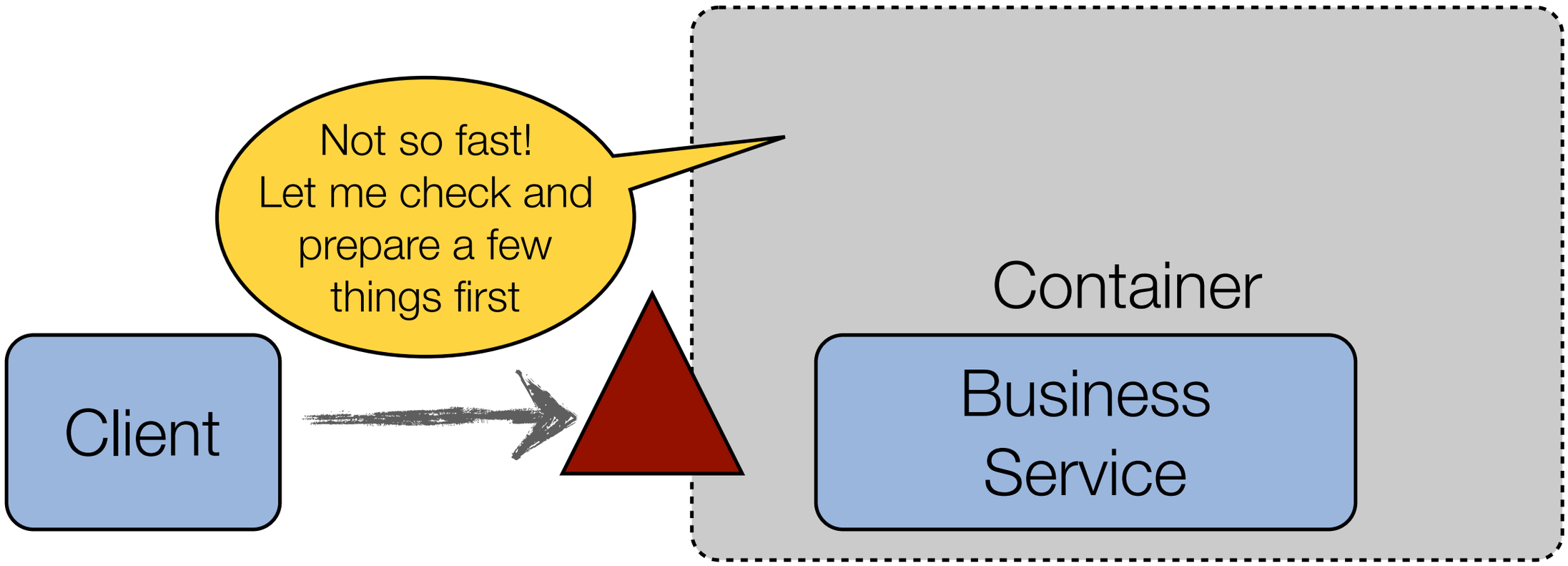
Introduction to AOP

The app server **mediates** the access between clients and EJBs. What does it mean?



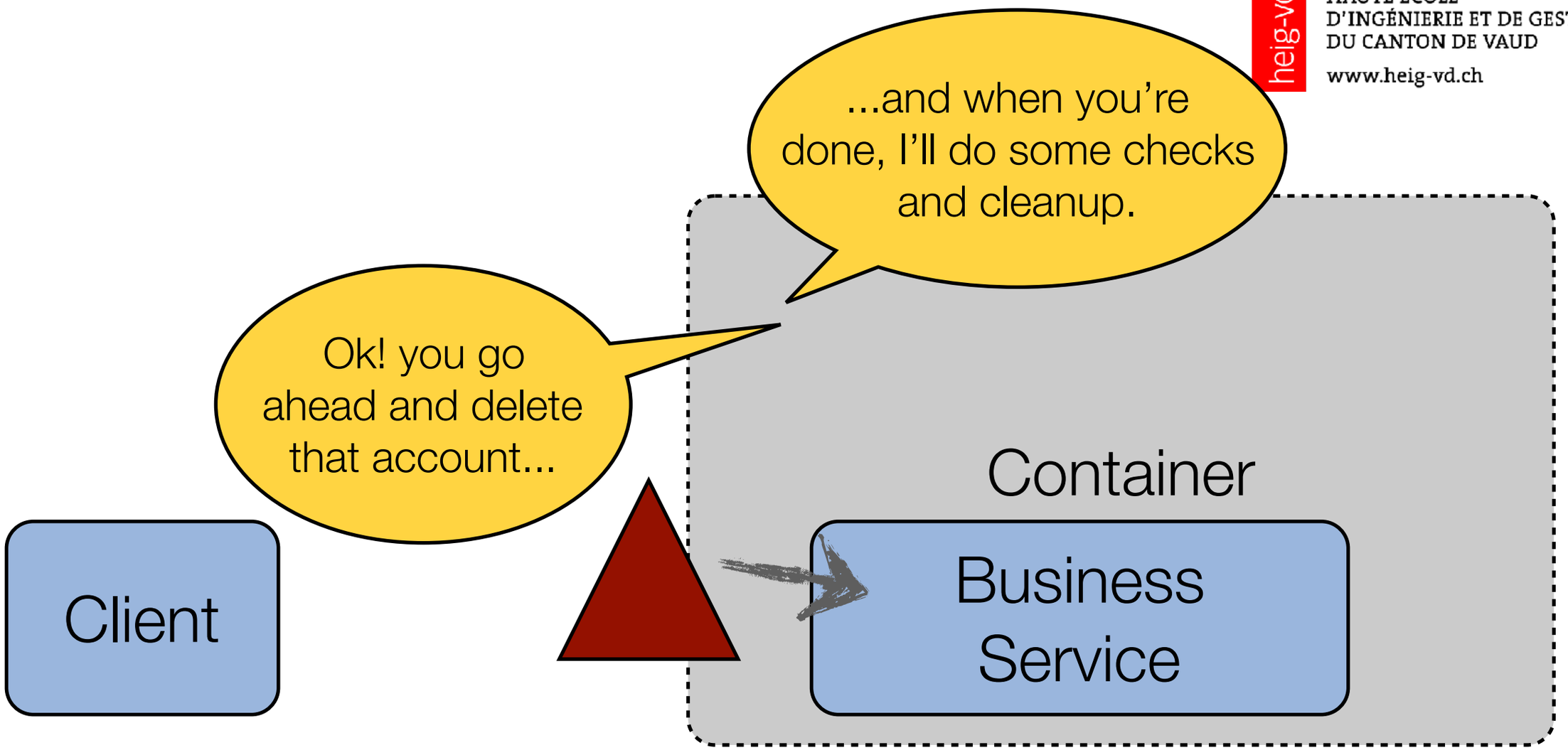

```
deleteAccount();
```

The business service, implemented as a Stateless Session Bean, is a **managed component**.
The client ***thinks*** that he has a direct reference to a Java object.
He is ***wrong***.




```
deleteAccount();
```

In reality, when the client invokes the deleteAccount() methods, the call is going **through the container**.
The container is in a position to **perform various tasks** (security checks, transaction demarcation, etc.)



When done, the container can forward the method call to the business service (your implementation).
On the way back, the response also goes back **via the container**.

heig-vd

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD

www.heig-vd.ch

How is that possible?
How does it work?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

App Server
Naming Service

lookup();

I need a service for managing customer accounts... Give me a reference!

Client

Your EJB
Interface

deleteAccount();

proxy generated
by the app serv.

deleteAccount();

Your EJB
Implementation

deleteAccount();

Your service implementation implements your interface.
The container dynamically generates a class, which implements the same interface. This class performs the technical tasks and invokes your class (proxy).

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Aha! it's a mechanism we can use to implement Aspect Oriented Programming, right?

Spring Framework - AOP

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **Aspect Oriented Programming**
  - Separation of concerns, cross-cutting concerns
  - Terminology
  - AOP frameworks
- **Putting Aspect Oriented Programming in practice**
  - Interceptors in Java EE
  - AOP in the Spring framework

Aspect Oriented Programming (AOP)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- In all applications, there are "things" that need to be done over and over and that are **orthogonal** to **business logic**.
- Examples:
  - Logging and auditing
  - Security checks (authorization)
  - Transaction management
- In traditional object-oriented design, the common approach is to implement the pure business logic and these orthogonal functions **at the same place** (in class methods).

***Separation of concerns**: business logic vs. other "aspects"*

AOP Frameworks

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- AspectJ created at Xerox PARC in 2001 (Gregor Kiczales)
- Several other frameworks and projects have been developed (e.g. AspectWerkz), for different languages.
- Java EE was built to achieve the goal of AOP (separation of concerns). It makes the concepts and terminology visible with Interceptors.
- The Spring Framework makes it possible to use AOP concepts and relies itself on AOP for some of its features.

aspectj *crosscutting objects for better modularity*

http://eclipse.org/aspectj/

Aspect Oriented Programming (AOP)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Where is my business logic? It's hard to find... What do I have to bother with all these infrastructure concerns?
- How can I get a global view for security management in my application?
- What if I need to change the way I do the auditing? I will have to go in every single method...
- *What a nightmare!!*

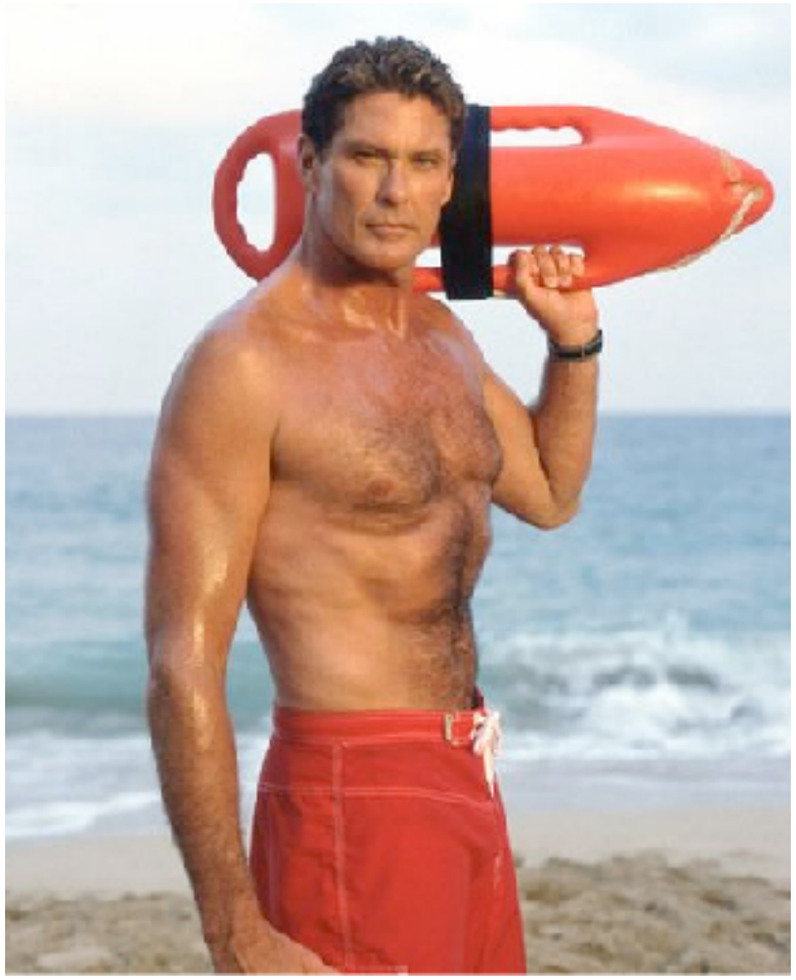

```
public void addProduct(Product p) {
  // check if the user is authenticated and authorized
  ...
  // start transaction
  ...
  // finally, some business logic
  ...
  // commit transaction
  ...
  // leave a trace in the audit trail
  ...
}
```

```
public void removeProdcut(Product p) {
  // check if the user is authenticated and authorized
  ...
  // start transaction
  ...
  // finally, some business logic
  ...
  // commit transaction
  ...
  // leave a trace in the audit trail
  ...
}
```

AOP to the rescue

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **AOP supports the separation of concerns**. In other words, it gives a way to split the implementation of the business logic from the implementation of system-level functions.
- **Terminology**
  - An **aspect** or **cross-cutting concern** refers to **something** that needs to be done throughout the application code. Security, logging and transaction management are examples of cross-cutting concerns.
  - An **advice** is the **orthogonal logic** that is executed when a certain join point is executed (advice can be executed **before**, **after** or **around** the join point).
  - A **pointcut** is an **expression** used to define a set of join points. With a pointcut, one can specify which join points (i.e. which methods)
  - A **join point** defines **when** the orthogonal logic could be executed. For instance, the execution of a `processOrder()` **method** is a join point.



AOP to the rescue

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- **AOP supports the separation of concerns**. In other words, it gives a way to split the implementation of the business logic from the implementaiton of system-level functions.
- **Terminology**
  - An **aspect** or **cross-cutting concern** refers to **something** that needs to be done throughout the application code (e.g. security, transaction management)
  - An **advice** is the code that is executed for a certain join point and some **pointcut** (i.e. **before**, **after** or **around**)
  - A **pointcut** is an **expression** used to define a set of join points. With a pointcut, one can specify which join points (i.e. which methods)
  - A **join point** defines **when** the orthogonal logic can be executed. For instance, the execution of a `processOrder()` **method** is a join point.

All methods that start with "`find`" in the `ch.heigvd.amt` package

the `findSea` in the `ch.heigvd.amt.BayWatch` class

How Can it Work?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- There are different ways to implement AOP.
- Remember that we want to "**combine**" two pieces of orthogonal code - located in two different artifacts (a "business" class and an "advice class").
- One possibility is to use a **special compilation process**. This is called "**weaving**", since the aspect code is weaved into the main business logic. As an alternative, it is possible to do the weaving as an **after-compilation** process. "Weaving" is what the AspectJ framework and toolset is doing.
- Another approach is to use **proxies** that are **dynamically generated**. This is something we can do we reflection.

AOP in Java EE

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Interceptors can be added **globally** (in the XML deployment descriptor) at the **class level** (apply to all methods in the class) or at the **method level**.

Interceptor Metadata Annotation	Description
`javax.interceptor.AroundConstruct`	Designates the method as an interceptor method that receives a callback after the target class is constructed
`javax.interceptor.AroundInvoke`	Designates the method as an interceptor method
`javax.interceptor.AroundTimeout`	Designates the method as a timeout interceptor for interposing on timeout methods for enterprise bean timers
`javax.annotation.PostConstruct`	Designates the method as an interceptor method for post-construct lifecycle events
`javax.annotation.PreDestroy`	Designates the method as an interceptor method for pre-destroy lifecycle events

*"Interceptors are used in conjunction with Java EE managed classes to allow developers to invoke interceptor methods on an associated target class, in conjunction with **method invocations** or **lifecycle events**. Common uses of interceptors are logging, auditing, and profiling."*

https://javaee.github.io/tutorial/interceptors.html

Examples

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Binding interceptors at the **class level**:

```
@Stateless
@Interceptors({PrimaryInterceptor.class, SecondaryInterceptor.class})
public class OrderBean {

    public void placeOrder(Order order) { ... }

}
```

Binding interceptors at the **method level**:

```
@Stateless
public class OrderBean {

    @Interceptors({PrimaryInterceptor.class, SecondaryInterceptor.class})
    public void placeOrder(Order order) { ... }

}
```

Examples

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Implementing an interceptor:

```
@AroundInvoke
public Object modifyGreeting(InvocationContext ctx) throws Exception {
    Object[] parameters = ctx.getParameters();
    String param = (String) parameters[0];
    param = param.toLowerCase();
    parameters[0] = param;
    ctx.setParameters(parameters);
    try {
        return ctx.proceed();
    } catch (Exception e) {
        logger.warning("Error calling ctx.proceed in modifyGreeting()");
        return null;
    }
}
```

Applying the interceptor:

```
@Interceptors(HelloInterceptor.class)
public void setName(String name) {
    this.name = name;
}
```

AOP in the Spring Framework

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- AOP is used in the Spring Framework to:
  - provide **declarative** enterprise services, especially as a replacement for EJB declarative services. The most important such service is declarative **transaction management**
  - allow users to implement **custom aspects**, complementing their use of OOP with AOP

"If you are interested only in generic declarative services or other pre-packaged declarative middleware services such as pooling, you do not need to work directly with Spring AOP, and can skip most of this chapter."

AOP with Spring: Pointcuts

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Pointcuts can be declared with an annotation (or with XML...)

```
@PointCut(expression)
private void aNameForThisSetOfMethods {}
```

The **expression** is based on the AspectJ pointcut language. Here are some examples:

the execution of any public method:

```
execution(public * *(..))
```

the execution of any method with a name beginning with "set":

```
execution(* set*(..))
```

the execution of any method defined by the AccountService interface:

```
execution(* com.xyz.service.AccountService.*(..))
```

the execution of any method on a Spring bean named 'tradeService':

```
bean(tradeService)
```

the execution of any method on a Spring bean with a name matching the wildcard expression

```
bean(*Service)
```

```
execution(modifiers-pattern? ret-type-pattern declaring-type-pattern? name-pattern(param-pattern) throws-pattern?)
```

http://static.springsource.org/spring/docs/2.5.6/reference/aop.html#aop-pointcuts

Defining a Pointcut “Inline”

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
@Aspect
public class MyFirstAspect {

    @Before("execution(public * ch.heigvd.osf..*.*(..))")
    public void myMethod(JoinPoint jp) {
     System.out.println("My advice has been applied...");
     System.out.println("target: " + jp.getTarget());
     System.out.println("this: " + jp.getThis());
     System.out.println("signature: " + jp.getSignature());
    }
}
```

```
    <aop:aspectj-autoproxy/>

    <bean id="myFirstAspect" class="ch.heigvd.osf.hellospringaop.aspects.MyFirstAspect">
    </bean>
```

Notes:

- `myMethod` will be executed before **any public method** in **any class** in the `ch.heigvd.osf package` (or in a **sub-package**) is called.
- `myMethod` has access to runtime information

Using an @Aspect to Define Pointcuts

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
package ch.heigvd.osf.system;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Pointcut;

@Aspect
public class SystemPointCuts {

  @Pointcut("execution(* create*(..))")
  public void createMethods() {}

  @Pointcut("execution(* update*(..))")
  public void updateMethods() {}

  @Pointcut("execution(* delete*(..))")
  public void deleteMethods() {}

  @Pointcut("createMethods() && updateMethods() && deleteMethods()")
  public void allCRUDMethods() {}

}
```

Using an @Aspect to Implement Advices

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

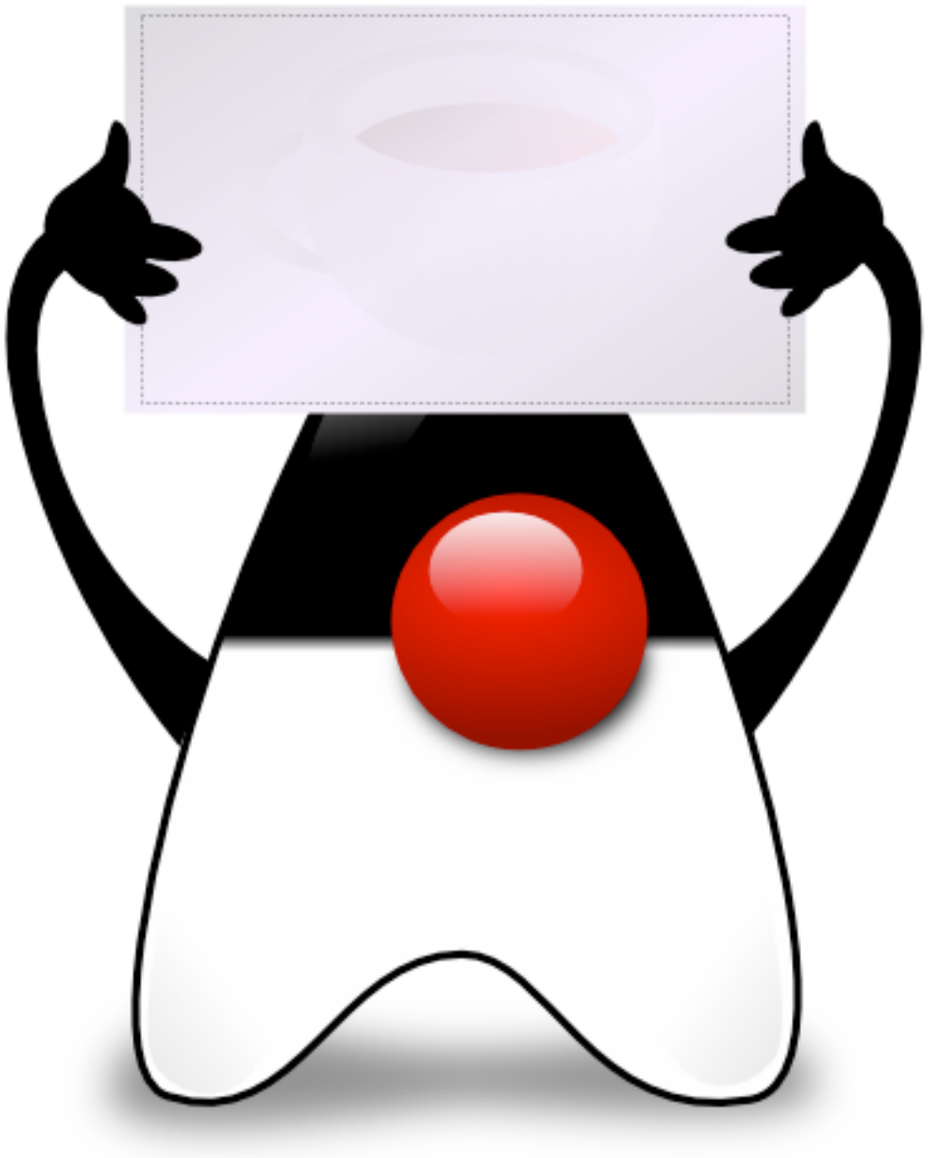
```
package ch.heigvd.osf.system.logging;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;

@Aspect
public class MyLoggingAspect {
  @Before("ch.heigvd.osf.system.SystemPointCuts.allCRUDMethods()")
  public void doLogOperation() {
    log.info(“About to call a CRUD method....”);
  }
}
```

Here, we work with:

- one pointcut, which is defined in the SystemPointCuts aspect (see previous slide)
- this pointcut defines a set of several join points: all the methods with a name starting with either create, update or delete
- one advice, which states that before every execution of the join points matching the pointcut, we will execute the doLogOperation

heig-vd
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Testing managed components

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
git@github.com:SoftEng-HEIGVD/Teaching-HEIGVD-AMT-Example-Notes.git

branch: step-003-datastore
```

How do I test this?

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

```
@WebServlet(name = "MyServlet", urlPatterns = “/hello")
public class MyServlet extends HttpServlet {
  @EJB
  IUsersDAO usersDAO;

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    User user = User.builder()
      .username(“oliechti”).firstName(“Olivier”).lastName(“Liechti”)
      .build();
    try {
      usersDAO.create(user);
      resp.getWriter().println("created user");
    } catch (Exception e) {
      resp.getWriter().println("Could not create user: " + e.getMessage());
    }
  }
}
```

If I write a unit test, I could instantiate a **TestServlet**… but how do I get **HttpServletRequest** and **HttpServletResponse** objects? And how do I make an **IUsersDAO** available to the test?

How do I test this?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
@Stateless
public class UsersDAO implements IUsersDAO {

  @Resource(lookup = "jdbc/notes")
  DataSource dataSource;

  @EJB
  IAuthenticationService authenticationService;

 public User create(User entity) throws DuplicateKeyException {
    Connection con = null;
    try {
      con = dataSource.getConnection();
      PreparedStatement statement = con.prepareStatement("INSERT INTO amt_users (USERNAME, FIRST_NAME, LAST_NAME,
EMAIL, HASHED_PW) VALUES (?, ?, ?, ?, ?)");
      statement.setString(1, entity.getUsername());
      statement.setString(2, entity.getFirstName());
      statement.setString(3, entity.getLastName());
      statement.setString(4, entity.getEmail());
      statement.setString(5, authenticationService.hashPassword(entity.getPassword()));
      statement.execute();
      return entity;
    } catch (SQLException e) {
      e.printStackTrace();
      throw new Error(e);
    } finally {
      try {
        con.close();
      } catch (SQLException e) {
        e.printStackTrace();
      }
    }
  }
}
```

If I write a unit test, I could instantiate a **UsersDAO**…
but how do I get a **DataSource** and an
IAuthenticationService objects?

Approach 1: mocking (unit testing)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Instead of using "real" objects that are normally provided by application, we generate "**fake**" objects that look like them.
- The "**fake**" or "**mock**" objects are like puppets: we can **tell them in advance how they should react to method calls** (i.e. when an object calls your method x, answer that).
- Typically, we create a "mock" object by providing the interface that it should implement.

Mockito is one of the most popular mocking framworks in Java

Approach 1: mocking (unit testing)

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

now you can verify interactions

```
import static org.mockito.Mockito.*;

// mock creation
List mockedList = mock(List.class);

// using mock object - it does not throw any "unexpected interaction" exception
mockedList.add("one");
mockedList.clear();

// selective, explicit, highly readable verification
verify(mockedList).add("one");
verify(mockedList).clear();
```

and stub method calls

```
// you can mock concrete classes, not only interfaces
LinkedList mockedList = mock(LinkedList.class);

// stubbing appears before the actual execution
when(mockedList.get(0)).thenReturn("first");

// the following prints "first"
System.out.println(mockedList.get(0));

// the following prints "null" because get(999) was not stubbed
System.out.println(mockedList.get(999));
```

Going back to our servlet…

HAUTE ÉCOLE D'INGÉNIERIE ET DE GESTION DU CANTON DE VAUD
www.heig-vd.ch

```
@WebServlet(name = "MyServlet", urlPatterns = “/hello")
public class MyServlet extends HttpServlet {
  @EJB
  IUsersDAO usersDAO;

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    User user = User.builder()
      .username(“oliechti”).firstName(“Olivier”).lastName(“Liechti”)
      .build();
    try {
      usersDAO.create(user);
      resp.getWriter().println("created user");
    } catch (Exception e) {
      resp.getWriter().println("Could not create user: " + e.getMessage());
    }
  }
}
```

If I write a unit test, I could instantiate a **TestServlet**… but how do I get **HttpServletRequest** and **HttpServletResponse** objects? And how do I make an **IUsersDAO** available to the test?

How do I test this?

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
package ch.heigvd.amt.notes.presentation;

@ExtendWith(MockitoExtension.class)
class TestServletTest {

  @Mock
  HttpServletRequest request;

  @Mock
  HttpServletResponse response;

  @Mock
  IUsersDAO usersDAO;

  @Mock
  PrintWriter responseWriter;

  TestServlet servlet;

  @BeforeEach
  public void setup() throws IOException {
    servlet = new TestServlet();
    servlet.usersDAO = usersDAO;
    when(response.getWriter()).thenReturn(responseWriter);
  }

  @Test
  void doGet() throws ServletException, IOException, DuplicateKeyException, SQLException {
    servlet.doGet(request, response);
    verify(usersDAO, atLeastOnce()).create(any());
  }
}
```

Important: when we mock, we only test one part of the system (we write unit tests). We don't use a real IUsersDAO.

It's good, because we don't need the container, the database, etc. It makes our test execution fast.

But it's not enough: we will also need to write tests to validate the end-to-end behavior (integration tests)

Approach 2: in-container testing

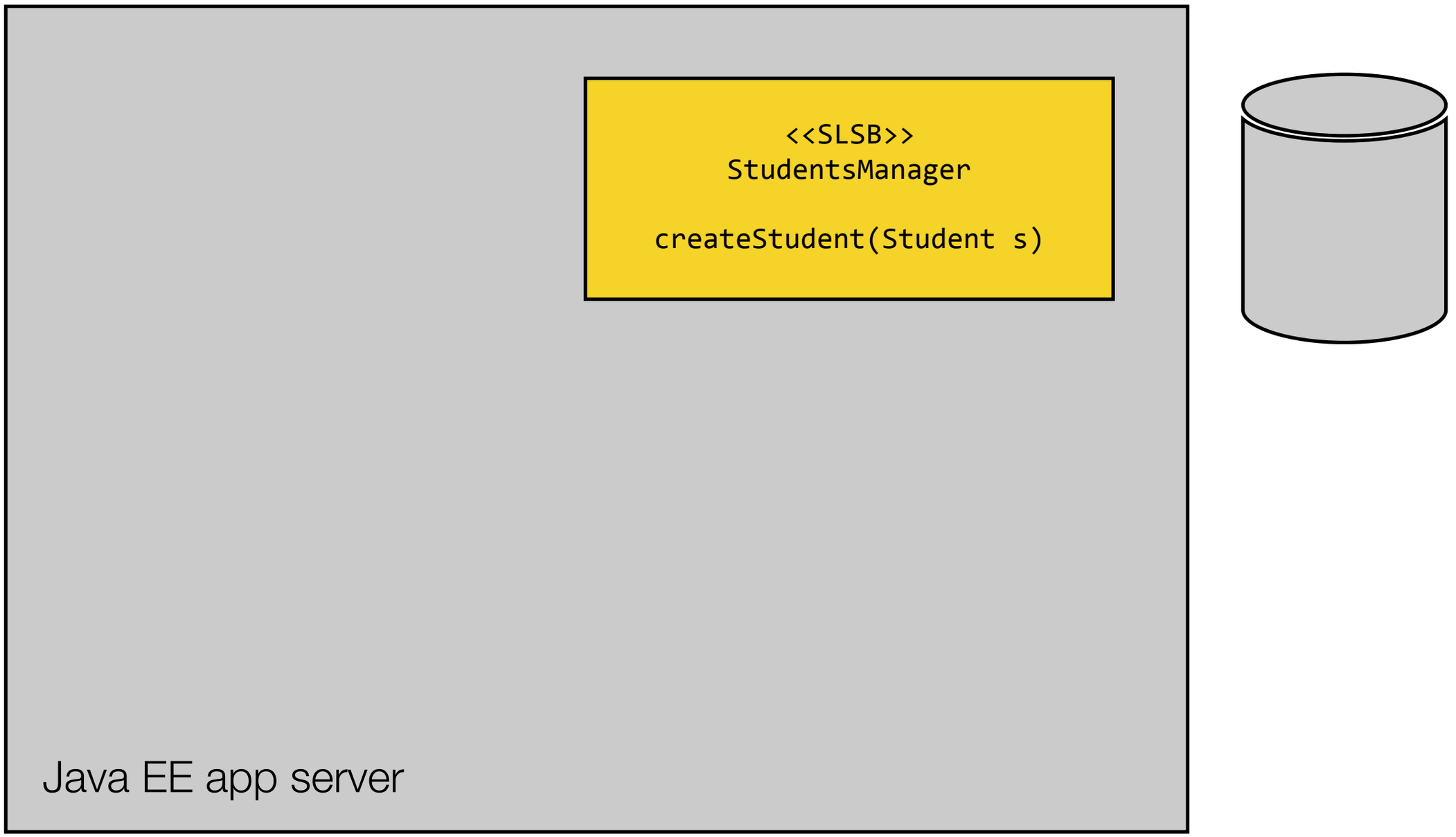
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- The components that we want to test are managed by the application server.
- If we run unit tests on the client, then we don't have these components ready to use.
- Idea: can we package and send our tests into an application server, execute them there, and receive the results?
- Idea: can we get a framework to do that for us, and not have to worry about all this plumbing?

Arquillian is a (set of) tool(s) to "easily" deploy tests in Java EE containers, to enable integration tests.

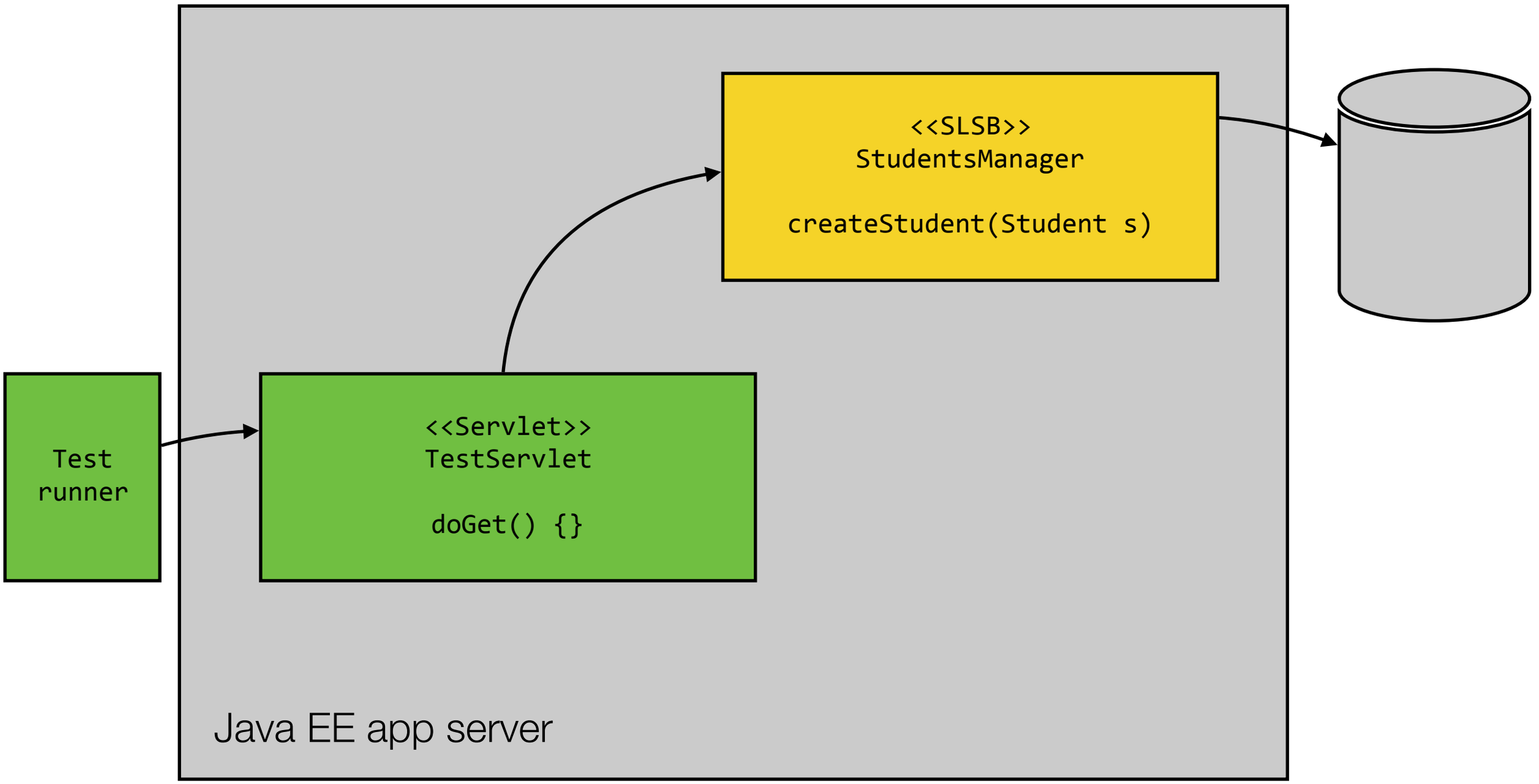
Approach 2: in-container testing

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



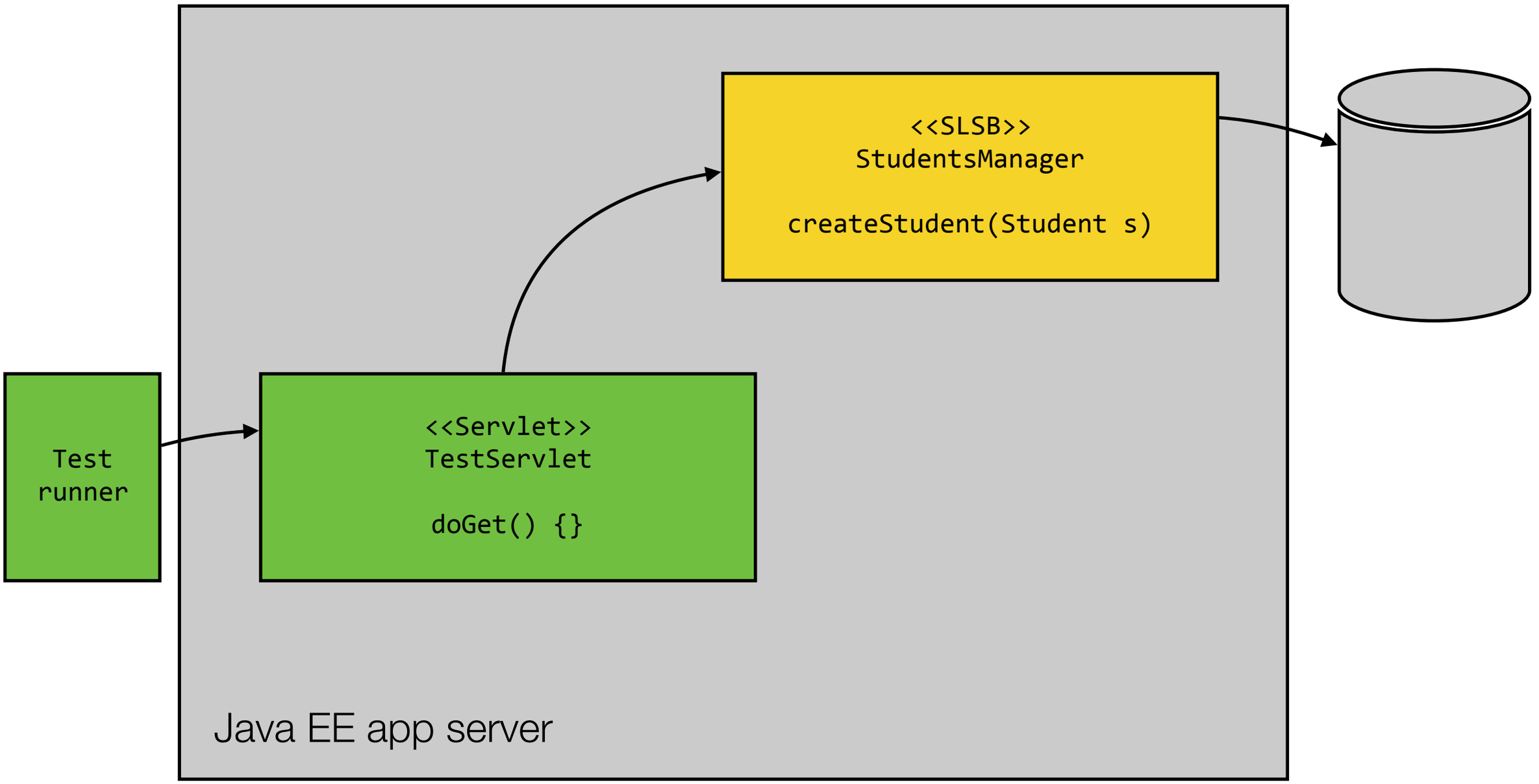
Approach 2: in-container testing

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



Approach 2: in-container testing

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



The Arquillian jungle…

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

Findings…

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

- Arquillian guides on the website have last been updated in November 2017…
- More information in GitHub repos, especially in issues
- Arquillian works with JUnit 4, not with JUnit 5.
- There are hundreds of ways to combine the arquillian framework and extensions.
- It took me a solid week to find a combination that worked for me.
- But once I had it running, it allowed me to implement and test a DAO in less than 1 hour. It also allowed me to find a memory leak.
- I tried **embedded**, **managed** and **remote** configurations. In the end, I believe that the remote configuration is the best.
- I had to add the payara self-signed certificate into my Java keystore.
- I have a setup where I can run Wildfly and Payara in Docker containers. I deploy the full .war in these containers. I can do remote debugging from IntelliJ.

Defining the containers

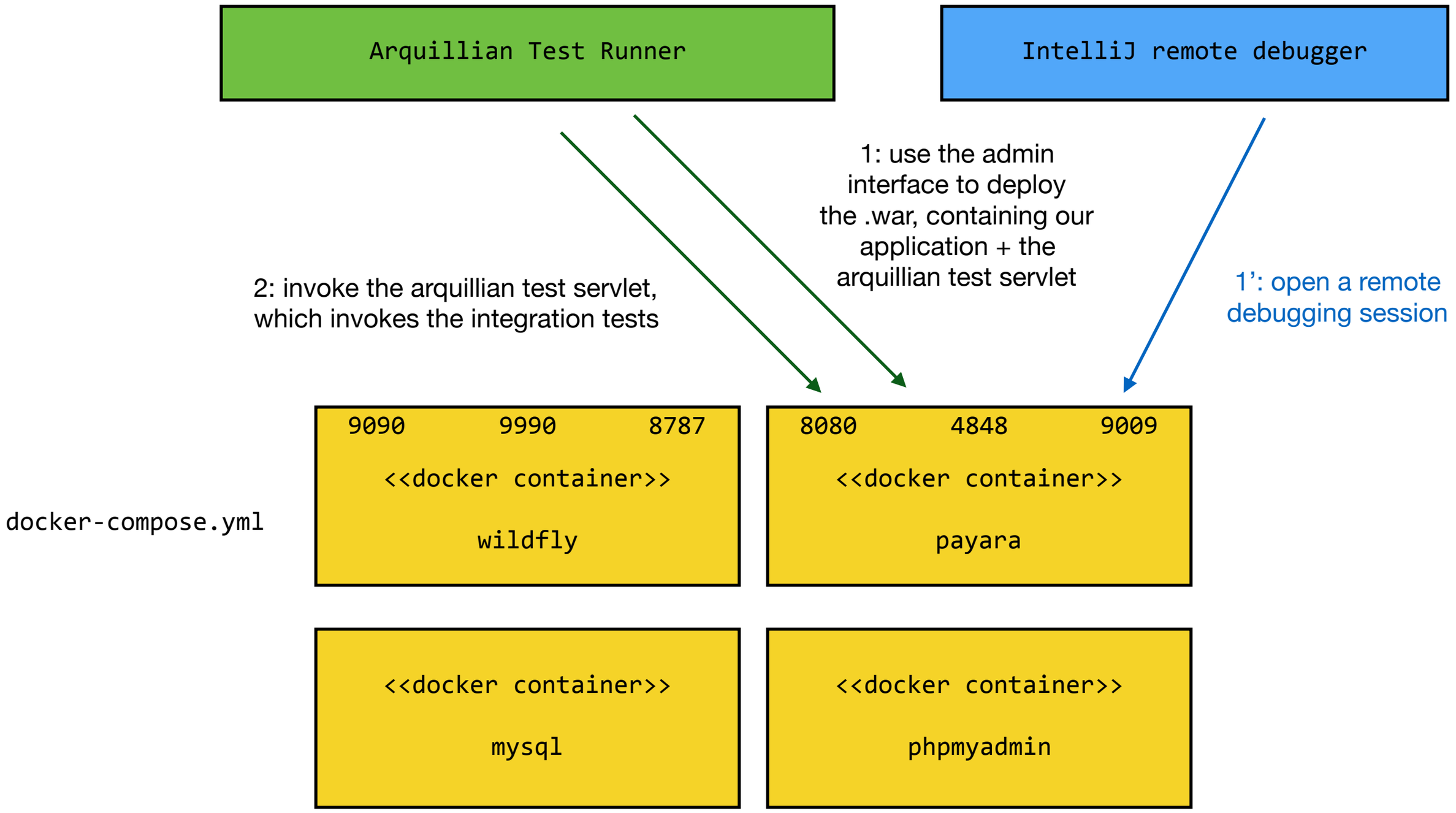
HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
<arquillian xmlns="http://jboss.org/schema/arquillian"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="
        http://jboss.org/schema/arquillian
        http://jboss.org/schema/arquillian/arquillian_1_0.xsd">

    <container qualifier="payara-remote" default="true">
        <configuration>
            <property name="chameleonTarget">payara:5.182:remote</property>
            <property name="adminHost">localhost</property>
            <property name="adminPort">4848</property>
            <property name="adminUser">admin</property>
            <property name="adminPassword">admin</property>
            <property name="adminHttps">true</property>
        </configuration>
    </container>
    <container qualifier="wildfly15-remote" default="false">
        <configuration>
            <property name="chameleonTarget">wildfly:15.0.1.Final:remote</property>
            <property name="managementAddress">localhost</property>
            <property name="managementPort">9990</property>
            <property name="username">admin</property>
            <property name="password">admin</property>
        </configuration>
    </container>
</arquillian>
```

Decouple deploy and debug

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch



docker-compose.yml

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

```
version: '3.7'
services:
  payara:
    build: ../../images/payara
    environment:
      PAYARA_ARGS: "--debug"
    ports:
      - 8080:8080
      - 4848:4848
      - 9009:9009

  wildfly:
    build: ../../images/wildfly
    ports:
     - 9090:9090
     - 9990:9990
     - 8787:8787

db:
  build: ../../images/mysql
  command: --default-authentication-plugin=mysql_native_password
  restart: always
  ports:
   - 3306:3306
  environment:
    MYSQL_DATABASE: amt_notes
    MYSQL_ROOT_PASSWORD: bananaSplit

phpmyadmin:
  image: phpmyadmin/phpmyadmin
  environment:
   - PMA_HOST=db
  restart: always
  depends_on:
   - db
  ports:
   - 8888:80
```

Important: check the Dockerfiles in the GitHub repo to see how to configure data sources in Payara and Wildfly

Important: note that I changed the Wildfly config to listen on port 9090 instead of 8080. Arquilliian does not like port mapping onto a different port…

Defining the containers

HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

We build the .war with maven and deploy it (alternative to @Deployment)

This (default) means that tests run in the container.

If we set this to ROLLBACK, the state of the DB will be restored after the test execution. Cool!

```
@RunWith(Arquillian.class)
@MavenBuild
@DeploymentParameters(testable = true)
public class UsersDAOTest {

  @EJB
  IUsersDAO usersDao;

  /*
  @Deployment
  public static JavaArchive createDeployment() {
    return ShrinkWrap.create(JavaArchive.class, "test.jar")
      .addPackages(true, "ch.heigvd");
  }
  */

  @Test
  @Transactional(TransactionMode.COMMIT)
  public void itShouldBePossibleToCreateAUser() throws DuplicateKeyException, SQLException {
    User olivier = User.builder().username("oliechti_" +
System.currentTimeMillis()).firstName("Olivier").lastName("Liechti").build();
    usersDao.create(olivier);
  }

…
}
```