03. Labwerk Servlets

## Inleiding

Hoe je een Java Servlet kunt maken en uitvoeren met behulp van annotaties in IntelliJ IDEA:

## De vereisten van de labwerk:

Installeer en configureer de vereiste software Voordat je een Java Servlet kunt maken, moet je een paar software hebben geïnstalleerd en geconfigureerd op je computer. Dit omvat:

* Java Development Kit (JDK): Dit biedt de benodigde tools voor het schrijven, compileren en uitvoeren van Java-programma's.
* IntelliJ IDEA: Dit is een Integrated Development Environment (IDE) die een grafische interface biedt voor het ontwikkelen van Java-webapplicaties.
* Apache Tomcat: Dit is een Java Servlet-container die Java Servlets en JavaServer Pages (JSP's) kan uitvoeren op een webserver.

Zodra je deze software hebt geïnstalleerd en geconfigureerd op je computer, kun je doorgaan met het maken van je Java Servlet.

## Het project aanmaken:

Maak een Dynamic Web Project in IntelliJ IDEA Om een ​​Java Servlet in IntelliJ IDEA te maken, moet je eerst een Dynamic Web Project maken. Hier zijn de stappen om dit te doen:

1. Open IntelliJ IDEA en selecteer Bestand > Nieuw > Project.
2. In het Nieuw project dialoogvenster, selecteer Java Enterprise.
3. Kies de versie van Java EE die je wilt gebruiken en klik op Volgende.
4. Geef je project een naam en selecteer de projectlocatie.
5. Klik op Voltooien om het project te maken.

## Dynamische Web Project:

Maak een Java Servlet Met het Dynamic Web Project gemaakt, kun je nu je Java Servlet maken. Hier zijn de stappen om dit te doen:

1. Klik met de rechtermuisknop op je project en selecteer Nieuw > Servlet.
2. In de New Servlet-wizard, geef je Servlet een naam en selecteer de package waarin het moet worden gemaakt.
3. Klik op Volgende en selecteer de URL-mapping voor je Servlet. Dit bepaalt de URL die de Servlet zal triggeren.
4. Klik op Volgende en selecteer de methoden die moeten worden gegenereerd voor je Servlet. Standaard zijn de doGet() en doPost() methoden geselecteerd.
5. Klik op Voltooien om de Servlet te maken.

## Servlet Afhankelijkheden:

Voeg annotaties toe aan je Java Servlet Om annotaties te gebruiken in je Java Servlet, moet je de nodige afhankelijkheden aan je project toevoegen. Hier zijn de stappen om dit te doen:

1. Open het pom.xml-bestand in je project.
2. Voeg de volgende afhankelijkheden toe:

<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
</dependency>  
  
<dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-jsp</artifactId>  
 <version>2.3.2-b02</version>  
 <scope>provided</scope>  
</dependency>  
Copy

1. Sla het pom.xml-bestand op.

## Servlet Annotaties:

Voeg annotaties toe aan je Java Servlet Met de nodige afhankelijkheden toegevoegd, kun je nu annotaties toevoegen aan je Java Servlet klassen. Hier zijn de stappen om dit te doen:

1. Voeg de volgende importstatements toe aan de bovenkant van je Java Servlet-bestand:

import javax.servlet.annotation.WebServlet;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import java.io.IOException;  
Copy

1. Voeg de @WebServlet-annotatie toe aan je Java Servlet-klasse om de Servlet te registreren bij de webcontainer. Bijvoorbeeld:

@WebServlet(name = "MyServlet", urlPatterns = {"/my-servlet"})  
public class MyServlet extends HttpServlet {  
 // ...  
}  
Copy

1. Implementeer de gewenste methoden van je Servlet. Bijvoorbeeld, om een ​​HTML-pagina terug te sturen naar de gebruiker:

@Override  
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException {  
 response.setContentType("text/html");  
 PrintWriter out = response.getWriter();  
 out.println("<html><body>");  
 out.println("<h1>Hello, World!</h1>");  
 out.println("</body></html>");  
}  
Copy

## Het uitvoeren van de App:

Voer je Java Servlet uit Met je Java Servlet gemaakt en geconfigureerd, kun je het nu uitvoeren in IntelliJ IDEA. Hier zijn de stappen om dit te doen:

1. Klik op het pictogram van het tabblad 'Run/Debug Configurations' in de rechterbovenhoek van het IntelliJ IDEA-venster.
2. Kies Tomcat Server -> Local.
3. Geef je configuratie een naam.
4. Selecteer de Tomcat-server die je wilt gebruiken.
5. Onder het Deployment-tabblad, klik op het + -pictogram en selecteer Artifact.
6. Selecteer je webapplicatie-artifact uit de lijst.
7. Klik op Toepassen om je configuratie op te slaan.
8. Klik op de knop Run in de bovenste toolbar om de Tomcat-server te starten en je webapplicatie te implementeren.
9. Open een webbrowser en ga naar de URL "http://localhost:8080/CONTEXT\_ROOT/my-servlet", waar CONTEXT\_ROOT de naam van je webapplicatie is.
10. Je zou nu de output van je Java Servlet in je webbrowser moeten zien.

**Gefeliciteerd!** Je hebt met succes een Java Servlet gemaakt en uitgevoerd met annotaties in IntelliJ IDEA. Met deze kennis kun je nu meer geavanceerde webapplicaties ontwikkelen met behulp van Java Servlets.

## Voorbeeld Servlet:

Hier is een voorbeeld van een Java Servlet die een REST API implementeert:

@WebServlet(name = "BooksServlet", urlPatterns = {"/api/books/\*"})  
public class BooksServlet extends HttpServlet {  
 private static final Map<Integer, String> books = new HashMap<Integer, String>() {{  
 put(1, "The Catcher in the Rye");  
 put(2, "To Kill a Mockingbird");  
 put(3, "1984");  
 }};  
  
 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String pathInfo = request.getPathInfo();  
 if (pathInfo == null || pathInfo.equals("/")) {  
 // Haal alle boeken op  
 Gson gson = new Gson();  
 String json = gson.toJson(books);  
 response.setContentType("application/json");  
 PrintWriter out = response.getWriter();  
 out.println(json);  
 } else {  
 // Haal een specifiek boek op  
 int id = Integer.parseInt(pathInfo.substring(1));  
 String title = books.get(id);  
 if (title == null) {  
 response.setStatus(HttpServletResponse.SC\_NOT\_FOUND);  
 } else {  
 Gson gson = new Gson();  
 String json = gson.toJson(Collections.singletonMap("title", title));  
 response.setContentType("application/json");  
 PrintWriter out = response.getWriter();  
 out.println(json);  
 }  
 }  
 }  
  
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 // Voeg een nieuw boek toe  
 Gson gson = new Gson();  
 String body = request.getReader().lines().collect(Collectors.joining(System.lineSeparator()));  
 Map<String, String> data = gson.fromJson(body, Map.class);  
 String title = data.get("title");  
 int id = books.size() + 1;  
 books.put(id, title);  
 String location = request.getRequestURL().append("/").append(id).toString();  
 response.setHeader("Location", location);  
 response.setStatus(HttpServletResponse.SC\_CREATED);  
 }  
  
 protected void doDelete(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 // Verwijder een boek  
 String pathInfo = request.getPathInfo();  
 if (pathInfo == null || pathInfo.equals("/")) {  
 response.setStatus(HttpServletResponse.SC\_BAD\_REQUEST);  
 } else {  
 int id = Integer.parseInt(pathInfo.substring(1));  
 String title = books.remove(id);  
 if (title == null) {  
 response.setStatus(HttpServletResponse.SC\_NOT\_FOUND);  
 } else {  
 response.setStatus(HttpServletResponse.SC\_NO\_CONTENT);  
 }  
 }  
 }  
}  
  
Copy

In dit voorbeeld implementeert de BooksServlet een REST API die clients in staat stelt om een lijst van alle boeken op te halen, individuele boeken op te halen, toe te voegen of te verwijderen op basis van hun ID's. De methoden doGet, doPost en doDelete van de Servlet behandelen respectievelijk GET-, POST- en DELETE-verzoeken en gebruiken de HttpServletRequest en HttpServletResponse objecten om gegevens te lezen en te schrijven.

## Do het Zelf:

Hier is een "doe-het-zelf" opdracht voor de studenten:

1. Pas de BooksServlet aan om het bijwerken van boeken via PUT-verzoeken te ondersteunen.
2. Implementeer validatie om ervoor te zorgen dat alle vereiste velden aanwezig zijn in POST- en PUT-verzoeken en dat de boek-ID's geldig zijn.
3. Voeg foutmeldingen toe aan de API-responses als er velden ontbreken of ongeldig zijn.
4. Implementeer een nieuwe servlet genaamd AuthorsServlet die een REST API implementeert voor het beheren van auteurs.
5. Maak een nieuw Author-object aan met velden zoals naam, leeftijd en nationaliteit.
6. Implementeer endpoints voor GET-, POST-, PUT- en DELETE-verzoeken om auteurs toe te voegen, bij te werken, op te halen en te verwijderen.
7. Implementeer validatie om ervoor te zorgen dat alle vereiste velden aanwezig zijn in POST- en PUT-verzoeken en dat de auteurs-ID's geldig zijn.
8. Voeg foutmeldingen toe aan de API-responses als er velden ontbreken of ongeldig zijn.
9. Integreer de BooksServlet en de AuthorsServlet om de gegevens van de boeken en de auteurs in één API te beheren.
10. Test de API grondig met behulp van een REST-client zoals Postman of cURL om er zeker van te zijn dat het correct werkt en fouten worden afgehandeld.

Deze "doe-het-zelf" opdracht geeft de je de kans om meer te leren over het implementeren van RESTful API's met behulp van Java Servlets en om hun vaardigheden te oefenen door middel van een praktische opdracht.

## Stapsgewijze Handleiding:

Hieronder vindt u een stapsgewijze handleiding voor het maken van een nieuw project met JPA, Hibernate en Java Servlet:

1. Maak een nieuw project aan in IntelliJ IDEA.
2. Maak een nieuwe map aan genaamd src.
3. Maak een submap aan genaamd main.
4. Maak een submap aan genaamd java.
5. Maak een nieuwe Java-klasse genaamd JpaUtil. Deze klasse bevat de logica voor het opzetten van de JPA EntityManagerFactory.
6. Maak een nieuwe Java-klasse genaamd EntityManagerHelper. Deze klasse bevat de logica voor het verkrijgen van een EntityManager voor het uitvoeren van CRUD-acties.
7. Maak een nieuwe Java-klasse genaamd User. Deze klasse bevat de definitie van een gebruiker en zijn eigenschappen zoals ID, naam, e-mailadres en wachtwoord.
8. Maak een nieuwe Java-klasse genaamd UserDAO. Deze klasse bevat de logica voor het uitvoeren van CRUD-acties op de gebruikersentiteit met behulp van de EntityManager.
9. Maak een nieuwe Java-klasse genaamd HibernateUtil. Deze klasse bevat de logica voor het opzetten van de Hibernate SessionFactory.
10. Maak een nieuwe Java-klasse genaamd UserServlet. Deze klasse is verantwoordelijk voor het ontvangen van de HTTP-verzoeken en het doorgeven van de gegevens aan de UserDAO voor verwerking.
11. Implementeer de doGet-methode om alle gebruikers op te halen en terug te sturen als JSON-gegevens.
12. Implementeer de doPost-methode om een nieuwe gebruiker toe te voegen met behulp van de JSON-gegevens die zijn ontvangen van de client.
13. Implementeer de doPut-methode om een bestaande gebruiker bij te werken met behulp van de JSON-gegevens die zijn ontvangen van de client.
14. Implementeer de doDelete-methode om een gebruiker te verwijderen met behulp van de JSON-gegevens die zijn ontvangen van de client.
15. Maak een nieuwe Java-klasse genaamd UserDTO. Deze klasse bevat de definitie van de gegevensoverdrachtsobjecten die worden gebruikt om de gegevens tussen de servlet en de DAO-klasse te verzenden.
16. Implementeer de conversie van User-entiteit naar UserDTO-object en vice versa.
17. Voeg de vereiste JPA- en Hibernate-afhankelijkheden toe aan het pom.xml-bestand.
18. Configureer de JPA- en Hibernate-instellingen in het persistence.xml-bestand.
19. Voeg een nieuwe servlet-mapping toe aan het web.xml-bestand om de UserServlet te koppelen aan de /users-URI.
20. Bouw en deploy de applicatie naar een webcontainer zoals Tomcat en test deze uitvoerig met behulp van een REST-client zoals Postman of cURL om er zeker van te zijn dat het correct werkt en fouten worden afgehandeld.

Deze opdracht daagt studenten uit om een volledig functionerende webtoepassing te bouwen met behulp van JPA, Hibernate en Java Servlets. Het vereist ook dat ze bekend zijn met het implementeren van een RESTful API en het gebruik van JSON-gegevens om de gegevens tussen de client en de server over te dragen.

Naast het implementeren van de basisfunctionaliteit van het toevoegen, bijwerken, ophalen en verwijderen van gebruikers, kunnen de studenten ook extra functionaliteit toevoegen, zoals authenticatie en autorisatie, het gebruik van aangepaste query's om gegevens op te halen en andere CRUD-operaties uit te voeren op andere entiteiten.

Studenten kunnen ook worden uitgedaagd om de applicatie verder te schalen en te optimaliseren door het gebruik van caching, indexing en load balancing. Dit kan helpen om de prestaties te verbeteren en de reactietijden van de applicatie te verkorten, vooral bij het verwerken van grote hoeveelheden gegevens of wanneer er meerdere gelijktijdige verzoeken worden verwerkt.

Al met al kan deze opdracht een uitdagende en interessante manier zijn om de vaardigheden van studenten op het gebied van webontwikkeling te verbeteren en hen te laten werken aan een project dat nuttig kan zijn voor het opbouwen van een professioneel portfolio.

## Online Bronnen:

Maak een envoudig project aan met behulp van de volgende bronnen:

1. Opzetten van je ontwikkelingsomgeving:

* Download en installeer de nieuwste versie van Intellij IDEA: <https://www.jetbrains.com/idea/>
* Download en installeer de nieuwste versie van Apache Tomcat: <https://tomcat.apache.org/download-90.cgi>
* Download en installeer de nieuwste versie van MySQL Community Server: <https://dev.mysql.com/downloads/mysql/>
* Download en installeer de nieuwste versie van MySQL Workbench: <https://dev.mysql.com/downloads/workbench/>

1. Maak een nieuw Java Servlet project in Intellij IDEA:

* Aan de slag met IntelliJ IDEA: <https://www.jetbrains.com/help/idea/getting-started.html>
* Het aanmaken van een Servlet: <https://www.jetbrains.com/help/idea/create-a-servlet.html>
* Externe bibliotheken toevoegen: <https://www.jetbrains.com/help/idea/library.html>

1. Maak een database en tabel aan met behulp van MySQL Workbench:

* Het aanmaken van een nieuwe MySQL-verbinding: <https://dev.mysql.com/doc/workbench/en/wb-getting-started-tutorial-create-connection.html>
* Het aanmaken van een nieuw schema: <https://dev.mysql.com/doc/workbench/en/wb-getting-started-tutorial-create-schema.html>
* Het aanmaken van een nieuwe tabel: <https://dev.mysql.com/doc/workbench/en/wb-getting-started-tutorial-create-table.html>

1. Maak een modelklasse met behulp van Lombok-annotaties:

* Lombok-projectpagina: <https://projectlombok.org/>
* Lombok-annotaties: <https://projectlombok.org/features/all>

1. Implementeer CRUD-operaties met behulp van JPA en Hibernate:

* Java Persistence API (JPA) tutorial: <https://www.baeldung.com/jpa-tutorial>
* Hibernate ORM tutorial: <https://www.baeldung.com/hibernate-orm-tutorial>

1. Implementeer gegevensvalidatie met behulp van Java Validation:

* Java Bean Validation-tutorial: <https://www.baeldung.com/javax-validation>
* Hibernate Validator-referentie: <https://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/>

1. Implementeer gegevens seeding met behulp van Java Faker:

* Java Faker Github Repository: <https://github.com/DiUS/java-faker>

1. Maak RESTful endpoints met behulp van Java Servlet:

* Java Servlet-tutorial: <https://www.tutorialspoint.com/servlets/index.htm>

1. Implementeer inhoudsonderhandeling met behulp van media types:

* Het gebruik van media types in RESTful webdiensten: <https://restfulapi.net/media-types/>

1. Configureer databaseverbinding en entity manager:

* Het configureren van een datasource in Intellij IDEA: <https://www.jetbrains.com/help/idea/connecting-to-a-database.html>
* JPA entity manager configureren: <https://www.baeldung.com/the-persistence-layer-with-spring-and-jpa>

1. Implementeer exception handling en error responses:

* Exception handling in Java Servlets: <https://www.javatpoint.com/exception-handling-in-servlets>
* Error handling in RESTful API: <https://restfulapi.net/error-handling-in-rest-api/>

1. Maak een DAO (Data Access Object) met CRUD-operaties voor het model met behulp van JPA:

* Java Persistence API (JPA) tutorial: <https://www.baeldung.com/jpa-tutorial>

1. Implementeer testen voor DAO-methoden met behulp van JUnit:

* JUnit tutorial: <https://www.baeldung.com/junit-5>

1. Implementeer een service-laag met CRUD-operaties die de DAO gebruikt:

* Service-laag tutorial: <https://www.baeldung.com/spring-data-jpa-named-queries>

1. Implementeer testen voor service-methoden met behulp van JUnit:

* JUnit tutorial: <https://www.baeldung.com/junit-5>

1. Implementeer een RESTful controller die de service-laag gebruikt:

* RESTful controller tutorial: <https://www.baeldung.com/spring-mvc-tutorial>

1. Implementeer testen voor controller-methoden met behulp van JUnit:

* JUnit tutorial: <https://www.baeldung.com/junit-5>

1. Implementeer authenticatie en autorisatie met behulp van Basic Authentication:

* Basic Authentication in Java Servlets: <https://www.baeldung.com/servlet-basic-authentication>
* Basic Authentication in RESTful API: <https://www.baeldung.com/spring-security-basic-authentication>

1. Implementeer logging met behulp van Log4j:

* Log4j-tutorial: <https://www.baeldung.com/log4j>

1. Implementeer caching met behulp van Ehcache:

* Ehcache-tutorial: <https://www.baeldung.com/ehcache>