

Quiz Questions

(1) Hibernate and ORM

←

Hibernate and CRUD operations

Hibernate and ORM

V

Problem

Submissions

Doubts

⚙️

✓

Hibernate and ORM

+

Score 20/20

Problem statement

Send feedback

Which of the following is true about Hibernate?

Options:

Pick one correct answer from below

☐ Hibernate is an Object Relational Mapping(ORM) for NET.

☒ Hibernate is an Object Relational Mapping(ORM) for JAVA.

☐ Both of the above.

☐ None of the above.

Solution description

Hibernate is a popular open-source ORM framework for Java that provides a way to map Java objects to relational databases and vice versa, making it easier to interact with databases and perform database operations.

(2) Hibernate and JDBC

←

Hibernate and CRUD operations

Hibernate and JDBC

V

Problem

Submissions

Doubts

⚙️

✓

Hibernate and JDBC

+

Score 20/20

Problem statement

Send feedback

How does Hibernate's handling of object-relational mapping differ from JDBC?

Options:

Pick one correct answer from below

☒ Hibernate manages the mapping automatically, while JDBC requires manual mapping.

☐ Hibernate requires manual mapping, while JDBC handles the mapping automatically.

☐ Both Hibernate and JDBC require manual mapping.

☐ Both Hibernate and JDBC manage the mapping automatically.

Solution description

Hibernate manages the mapping automatically, while JDBC requires manual mapping. Hibernate mappings establish the relationship between two database tables as attributes in your model.

(3) Feature of JDBC

←

CC

Hibernate and CRUD operations

Features of JDBC

V

Problem

Submissions

Doubts

✓ Features of JDBC

Score 20/20

Problem statement

Send feedback

What are the features that JDBC provides in a SpringBoot Application?

Options:

Pick one correct answer from below

☐ Send queries and update statements to the database.

☐ Enables a Java application to communicate with a database.

☐ Retrieve and process the results received from the database in answer to your query.

☒ All of the Above.

Solution description

JDBC (Java Database Connectivity) helps a programmer to write a Java program to connect to a database, retrieve the data from the database, and perform various operations on the data in a Java program.

(4) @Entity in Entity Layer

←

CC

Hibernate and CRUD operations

@Entity in Entity Layer

V

Problem

Submissions

Doubts

✓ @Entity in Entity Layer

Score 20/20

Problem statement

Send feedback

Imagine you are building a Spring Boot web application that allows users to save Blog posts in the application. You have created the following BlogPost class, which gets mapped to a table in the database:

```
package com.example.blogapp.entity;  
  
import javax.persistence.Entity;  
  
@Entity  
public class BlogPost {  
}  

```

What is the purpose of the @Entity annotation in the BlogPost class?

Options:

Pick one correct answer from below

☐ To mark the class as a Spring Bean.

☐ To mark the class as a Repository.

☒ To mark the class as a Hibernate Entity.

☐ To mark the class as a REST Controller.

Solution description

An entity represents a table in a relational database, and each entity instance corresponds to a row in that table. @Entity annotation defines that a class can be mapped to a table.

Submit

Ask a doubt

(5) @Table in Entity Layer

←

CC

Hibernate and CRUD operations

@Table in Entity Layer

V

Problem

Submissions

Doubts

✔ @Table in Entity Layer

Score 20/20

Problem statement

Send feedback

While creating BlogPost entity class, your teammate adds @Table annotation in your code as follows:
The Student class is as follows:

```
package com.example.blogapp.entity;
import javax.persistence.Entity;
@Entity
@Table(name="blog_post")
public class BlogPost {
}
```

Does the table name specified in the @Table annotation affect the entity class name of the BlogPost class in the code?

Options:

Pick one correct answer from below

☐ Yes, it will change the name of the entity.

☒ No, it only affects the name of the table in the database. ✔

☐ It depends on the configuration of the application.

☐ None of the above.

Solution description

No, the name specified in the @Table annotation only affects the name of the table in the database. It does not affect the name of the entity class in the code.

Submit

Ask a doubt

(6) @GeneratedValue in Entity Layer

←

CC

Classroom

Hibernate and CRUD operations

@GeneratedValue in Entity Layer

110%

+

Reset

?

V

≡

Problem

Submissions

Doubts

✔ @GeneratedValue in Entity Layer

Easy • Score 20/20

Problem statement

Send feedback

You have added three columns in the BlogPost class, and now you want the id column automatically generated. What are the id generation strategies that you can implement using @GeneratedValue annotation?

```
public class BlogPost {
    @Id
    @GeneratedValue(strategy = ...) // Add Generation Strategy
    private int id;

    @Column
    private String title;

    @Column
    private String content;
}
```

Options: Pick one correct answer from below

☐ GenerationType.AUTO

☐ GenerationType.TABLE

☐ GenerationType.SEQUENCE

☒ All of the above ✔

Solution description

@Id annotation is a way to define an identifier. Identifiers in Hibernate represent the primary key of an entity. If we want to generate the primary key value automatically, we can add the @GeneratedValue annotation. Generation strategies can use four generation types:

1. AUTO: Hibernate selects the strategy based on the used dialect.
2. IDENTITY: Hibernate relies on an auto-incremented database column to generate the primary key.
3. SEQUENCE: Hibernate requests the primary key value from a database sequence.
4. TABLE: Hibernate uses a database table to simulate a sequence.

Submit

Ask a doubt

(7) Session in DAL Layer

← Classroom

Hibernate and CRUD operations
Session in DAL Layer

?

V

≡

Problem Submissions Doubts

✓ Session in DAL Layer

Easy • Score 20/20

Send feedback

Problem statement

An object that maintains the connection between a Java object application and a database is called a _____.

Options: Pick one correct answer from below

☐ Cookie

☐ Persist

☒ Session

☐ Cache

Solution description

A session is an object that maintains the connection between Java object application and database. Session also has methods for storing, retrieving, modifying, or deleting data from databases using methods like persist(), load(), get(), update(), delete(), etc.

(8) Persistent Object in DAL Layer

← Classroom

Hibernate and CRUD operations
Persistent Object in DAL Layer

?

V

Problem Submissions Doubts

✓ Persistent Object In DAL Layer

Easy • Score 20/20

Send feedback

Problem statement

Persistent objects are saved and retrieved through a Session object?

Options: Pick one correct answer from below

☒ True

☐ False

Solution description

Session is the main interface for the persistence service and acts as a factory for Transaction instances used to manage the changes made to the objects in the database. It provides methods for inserting, updating, and deleting persistent objects and querying the database.

(9) Hibernate Configuration in Spring Boot

← Classroom

Hibernate and CRUD operations
Hibernate Configuration In SpringBoot

?

V

Problem Submissions Doubts

✓ Hibernate Configuration In SpringBoot

Easy • Score 20/20

Send feedback

Problem statement

What is the purpose of a Hibernate dialect in a database configuration?

Options: Pick one correct answer from below

☐ To define the database schema.

☐ To set the password for the database.

☐ To configure the network connection to the database.

☒ To specify the type of database being used.

Solution description

To specify the type of database being used. The Hibernate dialect informs Hibernate about the specific features and SQL syntax supported by the database being used, allowing Hibernate to generate the appropriate SQL statements for that database.

(10) Flow of Components in Hibernate Framework

← Classroom

Hibernate and CRUD operations
Flow of Components in Hibernate Framework

?

Problem Submissions Doubts

✓ Flow of Components in Hibernate Framework

Easy • Score 20/20

Problem statement

Send feedback

Which of the following represents the correct flow of components in the Hibernate framework?

Options: Pick one correct answer from below

☐ Service -> Interface -> Repository -> Controller -> Database

☐ Database -> Interface -> Service -> Repository -> Controller

☒ Controller -> Service -> Interface -> Repository -> Database

☐ Repository -> Interface -> Service -> Controller -> Database

Solution description

1. The controller component receives a request from the user, passes it to the service layer for processing, and returns the response to the user.
2. The service component processes the request from the controller interacts with the repository to retrieve or persist data and returns the result to the controller.
3. The interface component defines the methods and properties the repository and service layer must implement, serving as a blueprint for these components.
4. The repository component performs operations such as retrieving, updating, and deleting data and returning the result to the service layer.
5. The database component persists the data for the application and provides a way for the repository to store and retrieve data.

(11) Annotations

← Classroom

Hibernate and CRUD operations
Annotations

?

V

Problem Submissions Doubts

✓ Annotations

Easy • Score 20/20

Problem statement

Send feedback

Match the following terms with their correct definition:
Match the Error codes with their respective meanings.

A. Annotation used for marking a class as a data access layer component.	1. @Service
B. Annotation used for marking elements as a signature for the Data Access Layer component.	2. @Repository
C. Annotation used for marking a class as a service layer component.	3. @RestController
D. Annotation used for creating RESTful web services.	4. @Interface

Options: Pick one correct answer from below

☒ (A-2)(B-4)(C-1)(D-3)

☐ (A-4)(B-3)(C-1)(D-2)

☐ (A-3)(B-4)(C-1)(D-2)

☐ (A-3)(B-2)(C-4)(D-1)

Solution description

1. (A-2) - The @Interface annotation is used to mark elements as a signature for the Data Access Layer component.
2. (B-4) - The @Repository annotation marks a class as a data access layer component in a Java application.
3. (C-1) - The @Service annotation marks a class as a service layer component in a Java application.
4. (D-3) - The @RestController annotation marks a class as a component for creating RESTful web services in a Java application.

(12) @PathVariable and @RequestParam in Controller Layer

Classroom

Hibernate and CRUD operations
@PathVariable and @RequestParam in Controller

90%

Reset

Problem

Submissions

Doubts

✓ @PathVariable and @RequestParam in Controller Layer

Easy • Score 20/20

Problem statement

Send feedback

You created a controller method to fetch a blog post from the database as given below.

```
***
@GetMapping
public BlogPost getBlogPostById(//Missing Parameters ) {
    return blogPostService.getBlogPostById(id);
}
```

What of the following options is correct in context to the given code?

☐ @RequestParam is used to extract values from the request body, while @PathVariable is used to extract values from the URL path.

☐ @RequestParam and @PathVariable are the same and can be used interchangeably.

☐ @RequestParam and @PathVariable extract values from different sources in the request.

☒ @RequestParam is used to extract values from the query string, while @PathVariable is used to extract values from the URL path.

Solution description

The @PathVariable annotation extracts values from the URL path and passes them to the request method as parameters. In the example, the value of the id will be extracted from the URL path, in this case, "abc".

```
***
//Mapping: http://localhost:8080/blog/abc
@GetMapping("/{id}")
public BlogPost getBlogPostById(@PathVariable int id) {
    return blogPostService.getBlogPostById(id);
}
```

The @RequestParam annotation is used to extract values from query parameters and pass them as parameters to the method handling the request. In the example, id will be removed from the query parameters of the URL, in this case, "abc".

```
***
//Mapping: http://localhost:8080/blog?id=abc
@GetMapping("/")
public BlogPost getBlogPostById(@RequestParam String id) {
    return blogPostService.getBlogPostById(id);
}
```

Submit

Ask a doubt

(13) Session Methods in DAL Layer

Classroom

Hibernate and CRUD operations
Session Methods In DAL Layer

?

V

Problem

Submissions

Doubts

✓ Session Methods In DAL Layer

Easy • Score 20/20

Problem statement

Send feedback

In the DAL layer, you have created a method to fetch a BlogPost by id from the database.

```
***
@Override
public BlogPost getById(int id) {
    //Insert the code here
    BlogPost blogPost = session.get(BlogPost.class, id);
    return blogPost;
}
```

What is the correct way to create a session in this method present in BlogPostDALImpl.class?

Options: Pick one correct answer from below

☐ Session session = entityManager.wrap(Session.class);

☒ Session session = entityManager.unwrap(Session.class);

☐ Session session = entityManager.getSession(Session.class);

☐ Session session = entityManager.openSession();

Solution description

The entityManager.unwrap method allows you to interact directly with the underlying Hibernate session and perform operations such as fetching data from the database.

(14) @Transactional in-Service Layer

← Classroom

Hibernate and CRUD operations
@Transactional In Service Layer

?

V

Problem Submissions Doubts

✓ @Transactional in Service Layer

Easy • Score 20/20

Problem statement

Send feedback

What happens when a @Transactional method throws an exception?

Options: Pick one correct answer from below

☐ The transaction is automatically committed.

☒ The transaction is automatically rolled back. ✓

☐ The transaction is left in an undefined state.

☐ The transaction is left in a pending state.

Solution description

When a @Transactional method throws an exception, the transaction is automatically rolled back. Any changes made during the transaction will be undone, and the database will be returned to its state before the transaction begins. This is done to ensure that the database remains in a consistent state and to prevent data corruption.

(15) Create Query in DAL Layer

In the DAL layer, to fetch all the BlogPost from the database, you created another method as follows:

```
@Override
public List<BlogPost> getAllBlogPosts() {
    Session session = entityManager.unwrap(Session.class);
    //Insert the code here
}
```

What is the correct way to fetch all BlogPost from the database using the session created?

← Classroom

Hibernate and CRUD operations
Create Query In DAL Layer

?

V

Problem Submissions Doubts

Problem statement

Send feedback

There has been a requirement to fetch all the posts created. The following additions to the controller and service are already made to implement the changes:
Controller Class:

```
***
@RestController
@RequestMapping("/blogposts")
public class BlogPostController {

    @Autowired
    private final BlogPostService blogPostService;

    @GetMapping("/getAllPosts")
    public List<BlogPost> getAllBlogPosts() {
        return blogPostService.getAllBlogPosts();
    }
}
```

Service Class:

```
***
@Service
public class BlogPostService {
    @Autowired
    private final BlogPostRepository blogPostRepository;

    public List<BlogPost> getAllBlogPosts() {
        return blogPostRepository.getAllBlogPosts();
    }
}
```

Options: Pick one correct answer from below

☐ ***
List<BlogPost> blogPost = session.createQuery("from BlogPost", BlogPost.class).getResultList();
return blogPost;

☐ ***
List<BlogPost> blogPosts = session.get("from BlogPost", BlogPost.class).getResultList();
return blogPosts;

☐ ***
List<BlogPost> blogPosts = session.fetchAll("from BlogPost", BlogPost.class).getResultList();
return blogPosts;

☒ ***
List<BlogPost> blogPosts = session.createQuery("from BlogPost", BlogPost.class).getResultList();
return blogPosts; ✓

Solution description

1. The method session.createQuery is used to retrieve all BlogPost objects from the database. The argument to createQuery is a query that specifies what data to retrieve. In this case, the query specifies retrieving all BlogPost objects from BlogPost.
2. The method then calls getResultList on the query object to execute the query and retrieve the results, which are stored in a List and returned by the method.

(16) Session Update in DAL Layer

← Classroom

Hibernate and CRUD operations
Session Update in DAL Layer

?

V

Problem Submissions Doubts

✓ Session Update In DAL Layer

Easy • Score 20/20

Send feedback

Our blog has a new feature request, and the User can update his blog details. You have the update method in BlogPostDalImpl.class as follows:

```
***
@Override
public void update(BlogPost updateBlogPost) {
    Session session = entityManager.unwrap(Session.class);
    BlogPost currentBlogPost = session.get(BlogPost.class, updateBlogPost.getId());
    currentBlogPost.setTitle(updateBlogPost.getTitle());
    currentBlogPost.setContent(updateBlogPost.getContent());
    //Insert the correct code
}
```

What is the correct code to update the entity using this session?

Options: Pick one correct answer from below

☐ session.update()

☒ session.update(currentBlogPost)

☐ session.modify()

☐ session.modify(currentBlogPost)

Solution description

In this code, "session" is a variable that represents an active session with a database, and "currentBlogPost" is a variable that represents the entity (or row) that you want to update. The "update()" method is used to change an existing entity in the database.

(17) Implementing Session Update Method

← Classroom

Hibernate and CRUD operations
Implementing Session Update Method

?

V

Problem Submissions Doubts

✓ Implementing Session Update Method

Easy • Score 20/20

Send feedback

What is the requirement for the entity passed as a parameter to the update method?

Options: Pick one correct answer from below

☐ It should be a new entity.

☐ It should be a transient object.

☒ It should be a persistent object.

☐ It should have been previously deleted from the database and then re-added.

Solution description

When you call the update method, Hibernate will synchronize the state of the persistent object with the database, updating the database record to reflect any changes made to the object. For this to work, the object must already be associated with a Hibernate session and have a corresponding database record.