

Exam : EX378

Title : Red Hat Certified Cloud-native Developer exam

Version : DEMO

1.Create a configuration property for the application name and inject it into a CDI bean.

A. See the Explanation.

Answer: A

Explanation:

1. Add to

application.properties:

app.name=CloudNativeApp

2. Create a bean:

@ApplicationScoped

```
public class AppNameService {
```

```
    @Inject
```

```
    @ConfigProperty(name = "app.name")
```

```
    String appName;
```

```
    public String getAppName() {
```

```
        return appName;
```

```
    }
```

```
}
```

3. Inject AppNameService into a REST endpoint or another bean to use it.

2.Externalize a boolean flag and use it to enable/disable a service method.

A. See the Explanation.

Answer: A

Explanation:

1. In application.properties:

feature.enabled=true

2. Inject the flag:

@ApplicationScoped

```
public class FeatureService {
```

```
    @Inject
```

```
    @ConfigProperty(name = "feature.enabled")
```

```
    boolean enabled;
```

```
    public String execute() {
```

```
        return enabled ? "Feature is ON" : "Feature is OFF";
```

```
    }
```

```
}
```

3.Use default values when a configuration property is missing.

A. See the Explanation.

Answer: A

Explanation:

@ApplicationScoped

```
public class TimeoutService {
```

```
    @Inject
```

```
    @ConfigProperty(name = "request.timeout", defaultValue = "5000")
```

```
int timeout;
public int getTimeout() {
    return timeout;
}
}
```

This allows your app to run even if request.timeout is not in the config file.

4.Inject a configuration property into a constructor using MicroProfile Config.

A. See the Explanation.

Answer: A

Explanation:

@ApplicationScoped

```
public class DatabaseService {
    private final String jdbcUrl;
    @Inject
    public DatabaseService(@ConfigProperty(name = "db.url") String jdbcUrl)
    { this.jdbcUrl = jdbcUrl;
    }
    public String getJdbcUrl() {
        return jdbcUrl;
    }
}
```

Ensure db.url is defined in application.properties.

5.Inject a property into a JAX-RS resource class.

A. See the Explanation.

Answer: A

Explanation:

@Path("/env")

```
public class EnvResource {
    @Inject
    @ConfigProperty(name = "env.type")
    String envType;
    @GET
    public String getEnv() {
        return "Environment: " + envType;
    }
}
```

Declare env.type=dev in application.properties.