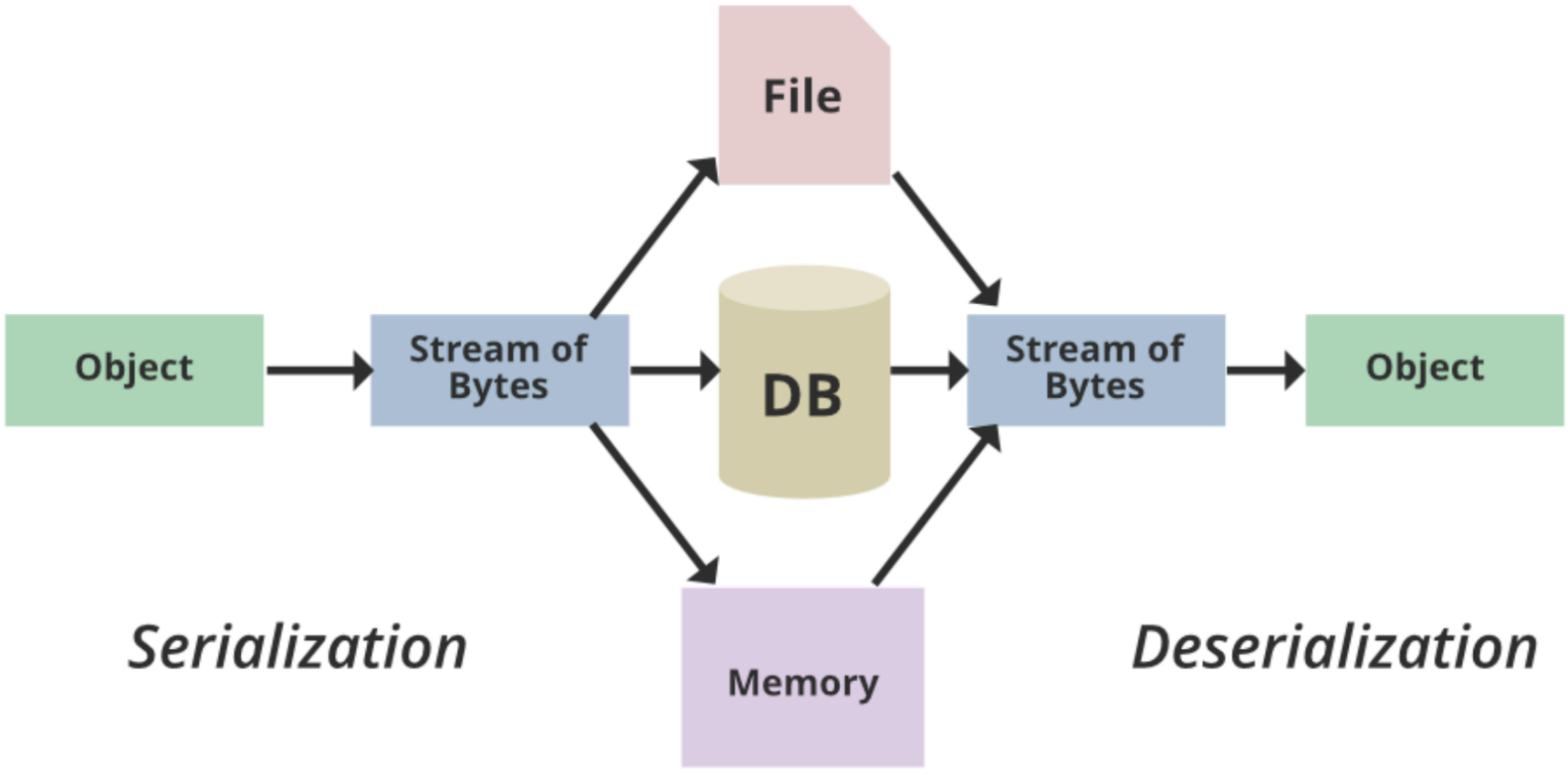
SIMS - VEŽBE 08

SERIJALIZACIJA PODATAKA

OSNOVNI KONCEPTI

- Mehanizam u kom se objekat može predstaviti kao niz bajtova koji uključuje podatke o objektu, kao i informacije o tipu objekta i tipove podataka uskladištene u objektu.
- Nakon što je serijalizovani objekat upisan u datoteku, može se pročitati iz datoteke i deserijalizovati, to jest, informacije o tipu i bajtovi koji predstavljaju objekat i njegove podatke mogu se koristiti za ponovno kreiranje objekta u memoriji.
- Ceo proces serijalizacije je nezavisan od JVM-a, što znači da se objekat može serijalisati na jednoj platformi i deserializovan na potpuno drugoj platformi

PROCES SERIJALIZACIJE



XSTREAM

- Eksterna biblioteka koja olakšava proces serijalizacije Javinih objekata u XML ili JSON format.
- Prednosti korišćenja XStream biblioteke za serijalizaciju i deserijalizaciju XML-a:
  - Pravilno konfigurisana, proizvodi veoma čist XML
  - Pruža značajne mogućnosti za prilagođavanje XML izlaza
  - Pruža podršku za kružne reference
  - U većini slučajeva korišćenja, XStream instanca je thread safe
  - Jasne poruke prilikom rukovanja izuzecima
- Sa sajta u sekciji Download je moguće skinuti najnoviju verziju biblioteke. Uključiti glavni jar u Build path i sve zavisne jar datoteke iz foldera XStream koje se mogu naći u zipovanom preuzetom fajlu
- http://x-stream.github.io/download.html
- Ukoliko koristite Maven:

```
<dependency>
    <groupId>com.thoughtworks.xstream</groupId>
    <artifactId>xstream</artifactId>
    <version>1.4.18</version>
</dependency>
```

SERIJALIZACIJA

```java
public class Customer {

    private String firstName;
    private String lastName;
    private Date dob;

    // standard constructor, setters, and getters
}
```

POJO klasa

```java
XStream xstream = new XStream();
```

...

```java
Customer customer = new Customer("John", "Doe", new Date());
String dataXml = xstream.toXML(customer);
```

Serijalizacija korišćem predefinisane konfiguracija

```xml
<com.baeldung.pojo.Customer>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <dob>1986-02-14 03:46:16.381 UTC</dob>
</com.baeldung.pojo.Customer>
```

Ime taga je puna putanja do klase

Da li želimo da prikažemo internu strakturu projekta?

ALIJASI

- Alijasi
  - Alias je ime koje želimo da koristimo za elemente umesto da koristimo podrazumevana imena (npr. naziv paketa + naziv klase)
  - Na primer, možemo da zamenimo com.baeldung.pojo.Customer sa Customer tako što ćemo registrovati alias za klasu Customer
  - Takođe možemo dodati pseudonime za svojstva klase
- Class Aliases

1. Programski
   - xstream.alias("customer", Customer.class);
2. Korišćenjem anotacija
   - @XStreamAlias(“customer")
     - Postavlja se iznad definicije klase
   - xstream.processAnnotations(Customer.class);
     - Definiše se u kodu pozivajuće metode

```
<customer>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <dob>1986-02-14 03:46:16.381 UTC</dob>
</customer>
```

ALIJASI

- Field Aliases
    1. Programski
        - xstream.aliasField("fn", Customer.class, “firstName");
    2. Korišćenjem anotacija
        - @XStreamAlias("fn")
            - Postavlja se iznad definicije atributa
- Default Aliases
    - alias("float", Float.class);
    - alias("date", Date.class);
    - alias("gregorian-calendar", Calendar.class);
    - alias("url", URL.class);
    - alias("list", List.class);
    - alias("locale", Locale.class);
    - alias("currency", Currency.class);

```
<customer>
    <fn>John</fn>
    <lastName>Doe</lastName>
    <dob>1986-02-14 03:46:16.381 UTC</dob>
</customer>
```

KOLEKCIJE

- Ukoliko u atribute Customer klase želimo da dodamo polje čiji je tip kolekcija

```
private List<ContactDetails> contactDetailsList;
```

- Dobijamo sledeći izlaz

```
<customer>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <dob>1986-02-14 04:14:05.874 UTC</dob>
    <contactDetailsList>
        <ContactDetails>
            <mobile>6673543265</mobile>
            <landline>0124-2460311</landline>
        </ContactDetails>
        <ContactDetails>
            <mobile>4676543565</mobile>
            <landline>0120-223312</landline>
        </ContactDetails>
    </contactDetailsList>
</customer>
```

KOLEKCIJE

- Ukoliko želimo da uklonimo<contactDetailsList> tag

1. Programski
    - xstream.addImplicitCollection(Customer.class, "contactDetailsList");
2. Korišćenjem anotacija nad atributom klase
    - @XStreamImplicit

- Dobijamo sledeći izlaz

```
<customer>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <dob>1986-02-14 04:14:20.541 UTC</dob>
    <ContactDetails>
        <mobile>6673543265</mobile>
        <landline>0124-2460311</landline>
    </ContactDetails>
    <ContactDetails>
        <mobile>4676543565</mobile>
        <landline>0120-223312</landline>
    </ContactDetails>
</customer>
```

IGNORISANJE POLJA KLASE

- Iz procesa serijalizacije moguće je izuzeti neke od atributa klase

1. Programski
    - xstream.omitField(Customer.class, “firstName");
2. Korišćenjem anotacija nad atributom klase
    - @XStreamOmitField

- Dobijamo sledeći izlaz

```
<customer>
    <lastName>Doe</lastName>
    <dob>14-02-1986</dob>
</customer>
```

DESERIJALIZACIJA

```
<com.baeldung.pojo.Customer>
    <firstName>John</firstName>
    <lastName>Doe</lastName>
    <dob>1986-02-14 03:46:16.381 UTC</dob>
</com.baeldung.pojo.Customer>
```

Serijalizovani podatak

XML se može deserijalizovati na više načina, uključujući deserijalizaciju iz datoteke, stream-a, reader-a ili string-a.

```
Customer convertedCustomer = (Customer) xstream.fromXML(customerXmlString);
Assert.assertTrue(convertedCustomer.getFirstName().equals("John"));
```

```
public class Customer {

    private String firstName;
    private String lastName;
    private Date dob;

    // standard constructor, setters, and getters
}
```

Objekat dobijen iz XML-a

JETTISON MAPPED XML DRIVER

- XStream omogućava serijalizaciju Java objekata i u JSON format
- U tu svrhu koristi se JettisonMappedXmlDriver koji se prosleđuje konstruktoru XStream klase
- Ukoliko koristite Maven

```
<dependency>
    <groupId>org.codehaus.jettison</groupId>
    <artifactId>jettison</artifactId>
    <version>1.4.1</version>
</dependency>
```

SERIJALIZACIJA U JSON

```java
public class Customer {

    private String firstName;
    private String lastName;
    private Date dob;
    private String age;
    private List<ContactDetails> contactDetailsList;

    // getters and setters
}
```

```java
xstream = new XStream(new JettisonMappedXmlDriver());
dataJson = xstream.toXML(customer);
```

```json
{
  "com.baeldung.pojo.Customer": {
    "firstName": "John",
    "lastName": "Doe",
    "dob": "1986-02-14 16:25:50.745 UTC",
    "age": 30,
    "contactDetailsList": [
      {
        "com.baeldung.pojo.ContactDetails": [
          {
            "mobile": 6673543265,
            "landline": "0124-2460311"
          },
          {
            "mobile": 4676543565,
            "landline": "0120-223312"
          }
        ]
      }
    ]
  }
```