



CS544 EA

# Hibernate

## Mapping Data Types

# Data Types

- JPA has **decent defaults** for most types
  - Java and SQL data types are not that different
  - Ints become ints, Strings become varchar(255), ...
  - You can customize things (length of varchar)
- Not all types always map correctly
  - Specifically date and time related types



# @Basic

- @Basic indicates that a property should be persisted and the default type should be used
  - JPA assumes these are there
    - (you don't have to add them)
  - Also has options for:
    - Indicating that a property is Nullable
    - Indicating if a property should be fetched lazily

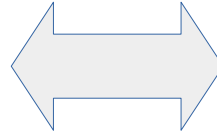
Hibernate mostly ignores this  
it doesn't make sense from an  
optimization point of view

# Exactly the same

```
package cs544.hibernate01.basic;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Integer id;
    @Basic
    private String firstName;
    @Basic
    private String lastName;
    ...
}
```



```
package cs544.hibernate01.basic;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    private String firstName;
    private String lastName;
    ...
}
```

# @Column

- **@Column** allows us to specify several optional additional values for this column
  - Name: column name can differ from property name
  - Length: for string valued properties
  - Scale and Precision for decimal columns
  - Nullable: if the column should be nullable
  - Unique: if the column values should be unique
  - Table (for secondary tables, discussed later)
  - ColumnDefinition: raw DDL to be used for this column

```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;
    @Column(name="first", length=45, nullable=false)
    private String firstName;
    @Column(name="last", length=60, nullable=true)
    private String lastName;

    ...
}
```

# Date and Time

- Date and Time related data-types always default to the SQL type: **TimeStamp**
  - Includes: `java.util.Date`, `java.sql.Date`, `java.util.Calendar`
  - But you may not always want it stored as a Timestamp!
- `java.time.*` not officially supported by JPA
  - Include Hibernate-Java8 maven artifact to fix this

See: <https://thoughts-on-java.org/hibernate-5-date-and-time/>

# @Temporal

- **@Temporal** lets you to specify a SQL data type by giving it a value of the TemporalType enum:
  - TemporalType.DATE
  - TemporalType.TIME
  - TemporalType.TIMESTAMP (default)

```
@Entity
public class Customer {

    ...

    @Temporal(TemporalType.DATE)
    private Date birthDate;

    ...

}
```



# @Transient

- JPA **automatically includes** all the instance variables of a class
  - Auto-maps them to columns of the same name
- What if you do not want to persist an variable?
  - @Transient specifies that it should not be stored

# Large Objects

- Certain things need more space in the DB
  - Images are usually stored as BLOBs
  - Large amounts of text as CLOBs
- JPA offers the **@LOB** annotation
  - Placed on text related properties makes CLOB
  - Placed on binary related properties makes BLOB

```
@Entity
public class Customer {
    @Id
    @GeneratedValue
    private Long id;

    @Column(name="first", length=45, nullable=false)
    private String firstName;

    @Column(name="last", length=60, nullable=true)
    private String lastName;

    @Temporal(TemporalType.TIMESTAMP)
    private Date birthDate;

    @Transient
    private String temp;

    @Lob
    private String biography;

    ...
}
```