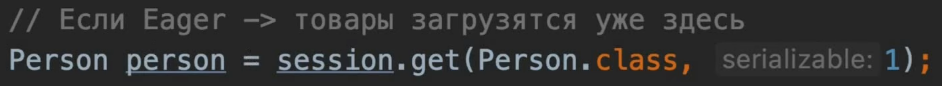
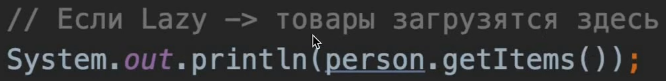
**Ленивая загрузка (lazy, eager loading)**

**Пример:** у нас есть связь один ко многим. Есть человек и у человека есть множество товаров. Этих товаров может быть большое количество. Иногда бывает такое, что нам нужно получить сущность(человека), но при этом его связанные сущности(товары) нам не нужны.

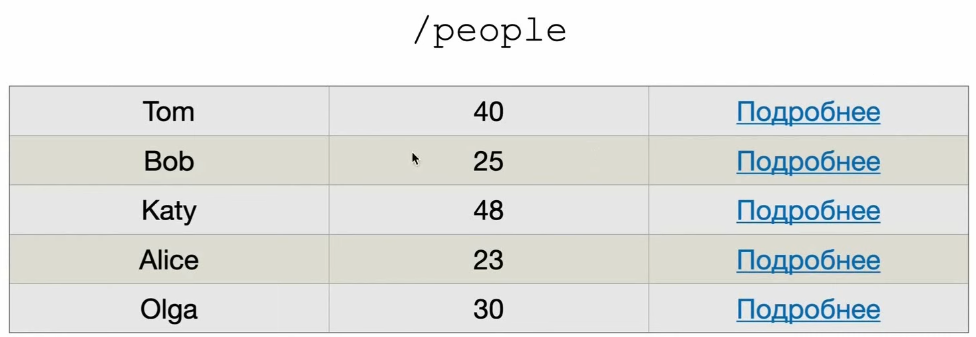
**Eager loading –** когда мы получем человека из базы данных у него уже будут назначены все его товары, тоесть **hibernate** сделал отдельный запрос к таблице с товарами и извлёк из этой таблицы все твоары которые принадлежат этому человеку.



**Lazy loading –** данные будут загружаться из базы данных только по требованию. Тоесть когда мы получим человека, то в человеке не будет товаров, но если мы эти товары потребуем, например вызовем геттер, то на этом этапе hibernate сделает запрос к таблице товаров и получит все товары которые принадлежат этому человеку.



**Пример из жизни:**



Например, у нас есть страница, и на этой странице мы просто показываем список людей из базы данных. У человека есть имя, возраст, так же есть ссылка(подробнее) для перехода на страницу конкретного человека и там посмотреть все его товары.

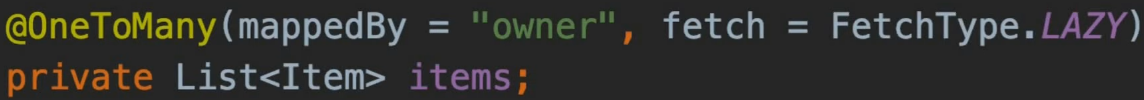
На главной странице **/people** нам не нужны товары для каждого из людей так как мы эти товары здесь не показываем. Если мы будем загружать товары для каждого человека из этого списка это будет очень долго. **НО** если мы будем использовать ленивую загрузку (lazy loading), то все эти связанные товары не будут загружаться из базы данных тоесть просто загрузяться данные по людям без товаров. А вот когда мы перейдем на страницу конкретного человека мы сможем для этого человека подгрузить нужные товары.

**Ленивая загрузка (Lazy Loading) почти всегда является хорошей практикой.**

**Пример назначения разных видов загрузки**

Связанные товары назначаются сразу (Eager Loading).

  
Связанные товары загружаются только по запросу (при вызове геттера, например)



**FetchType** по умолчанию когда явно его не указываем**:**

**@OneToMany –** Lazy

**@ManyToMany** - Lazy

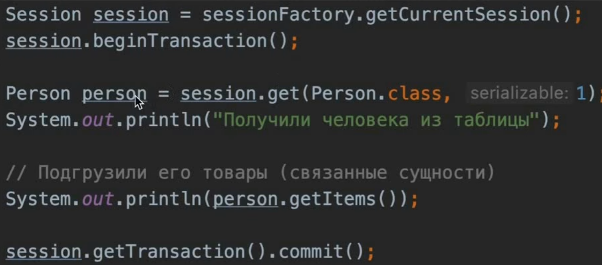
**@ManyToOne -** Eager

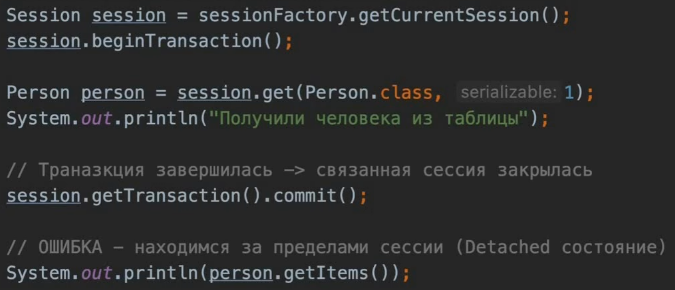
**@OneToOne –** Eager

**Вывод:** там где на конце **Many** по умолчанию будет **Lazy** загрузка.

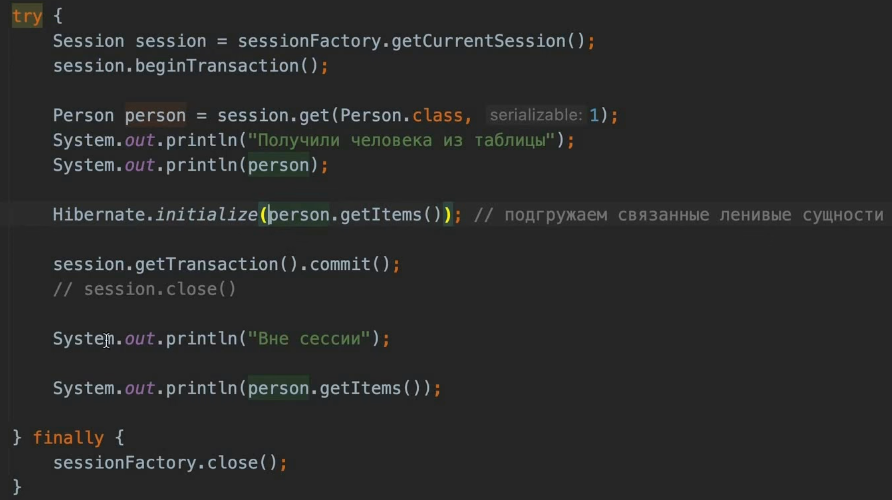
**Тонкости связанные с ленивой загрузкой**

**1.** Мы можем подгружать связанные сущности только находясь внутри транзакции (сущность должна находиться в состоянии **Persistent/Managed**).

  
  
  
  
  
  
  
  
  
  
  
  
  
**2.** За пределаеми **Hibernate** сессии (**Detached** состояние) загрузить связанные сущности не получится — будет ошибка.



**3.** Может быть такая ситуация когда нужно использовать связанные сущности за пределами сессии, чтобы это сделать нужно внутри сессии вызвать геттер и получить связанные сущности, НО если мы просто на объекте вызовем геттер никак не используя полученные сущности, то возникнет ошибка при обращении к этим связанным сущностям за пределами сессии. Ошибка будет возникать из за того, что **hibernate** оптимизурует запросы к базе данных и если те связанные сущности которые мы пытаемся получить через геттер нигде не будут использованны внутри сессии, то **hibernate** просто не будет делать запрос на получение этих сущностей. Чтобы ошибка не возникала и можно было использовать связанные сущности за пределами сессии нужно использовать метод ***Hiberante.session.initialize()****.*



**4.** Если нужно дозагрузить связанные сущности после того как мы вышли из сесии (вне транзакции), то нужно заново открыть сессию и начать новую транзакцию. НО при открытии новой сессии тот объект который мы получили из первой сессии будет в состоянии **Detached** и не будет привязан к новой сесии поэтому перед тем как получить связанные сущности из объекта который был создан в первой сессии нам нужно сначала этот объект пристегнуть к новой сессии переведя его в состояние **Persistent** с помощью метода ***session.merge()***. Этому методу мы передаем нужный объект который нужно перевести в состояние **Persistent** и то что этот метод возвращает мы должны положит опять в ту переменную которую мы передали в качестве аргумента. Тоесть этот метод возвращает тот объект который будет пристегнут к сессии.

