

JPA

Beyond Essentials

by Paweł Szulc

pawel@rabbitsoftware.pl
@paulszulc

JPA

pawel@rabbitsoftware.pl
@paulszulc

JPA

pawel@rabbitsoftware.pl
@paulszulc

JPA

pawel@rabbitsoftware.pl
@paulszulc



Afterparty 18:00

pawel@rabbitsoftware.pl
@paulszulc

Hello.

pawel@rabbitsoftware.pl
@paulszulc

Hello.

Who am I?

pawel@rabbitsoftware.pl
@paulszulc

Hello.

Who am I?

Paweł Szulc

pawel@rabbitsoftware.pl
@paulszulc

Hello.

Who am I?

Paweł Szulc

Java4People 2009 – Testing EJB

pawel@rabbitsoftware.pl
@paulszulc

New since 2009



pawel@rabbitsoftware.pl
@paulszulc

But why?
why JPA?

pawel@rabbitsoftware.pl
@paulszulc

JPA

Is EVERYWHERE!

pawel@rabbitsoftware.pl
@paulszulc

JPA

Is EVERYWHERE!

No one is really talking about.

JPA's advantage: **simplicity**

pawel@rabbitsoftware.pl
@paulszulc

JPA's advantage: **simplicity**
JPA's disadvantage: **simplicity**

JPA



Entities

pawel@rabbitsoftware.pl
@paulszulc

JPA



Entities

Mapping to database

pawel@rabbitsoftware.pl
@paulszulc

JPA

Entities

Mapping to database

persist(), merge(), remove()

JPA

Entities

Mapping to database

`persist()`, `merge()`, `remove()`

JPQL queries, cascading

World of JPA

Entities

Mapping to database

`persist()`, `merge()`, `remove()`

JPQL queries, cascading

`pawel@rabbitsoftware.pl`
`@paulszulc`

World of JPA



Entities

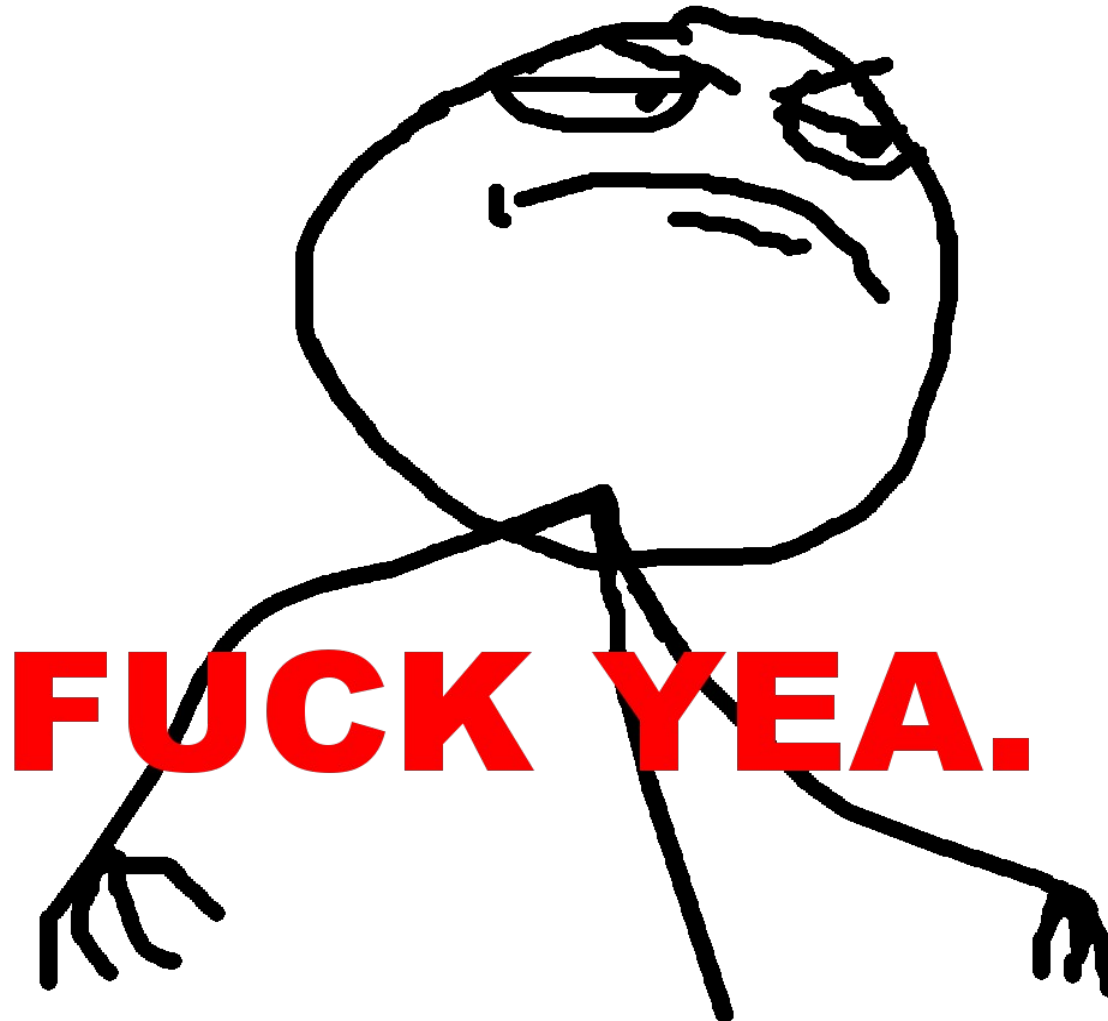
Mapping to database

`persist()`, `merge()`, `remove()`

JPQL queries, cascading

`pawel@rabbitsoftware.pl`
`@paulszulc`

World of JPA



pawel@rabbitsoftware.pl
@paulszulc

World of JPA



Entities

Mapping to database

`persist()`, `merge()`, `remove()`

JPQL queries, cascading

`pawel@rabbitsoftware.pl`
`@paulszulc`

World of JPA



Entities

Mapping to database

`persist()`, `merge()`, `remove()`

JPQL queries, cascading

`pawel@rabbitsoftware.pl`
`@paulszulc`

World of JPA



Entities

Mapping to database

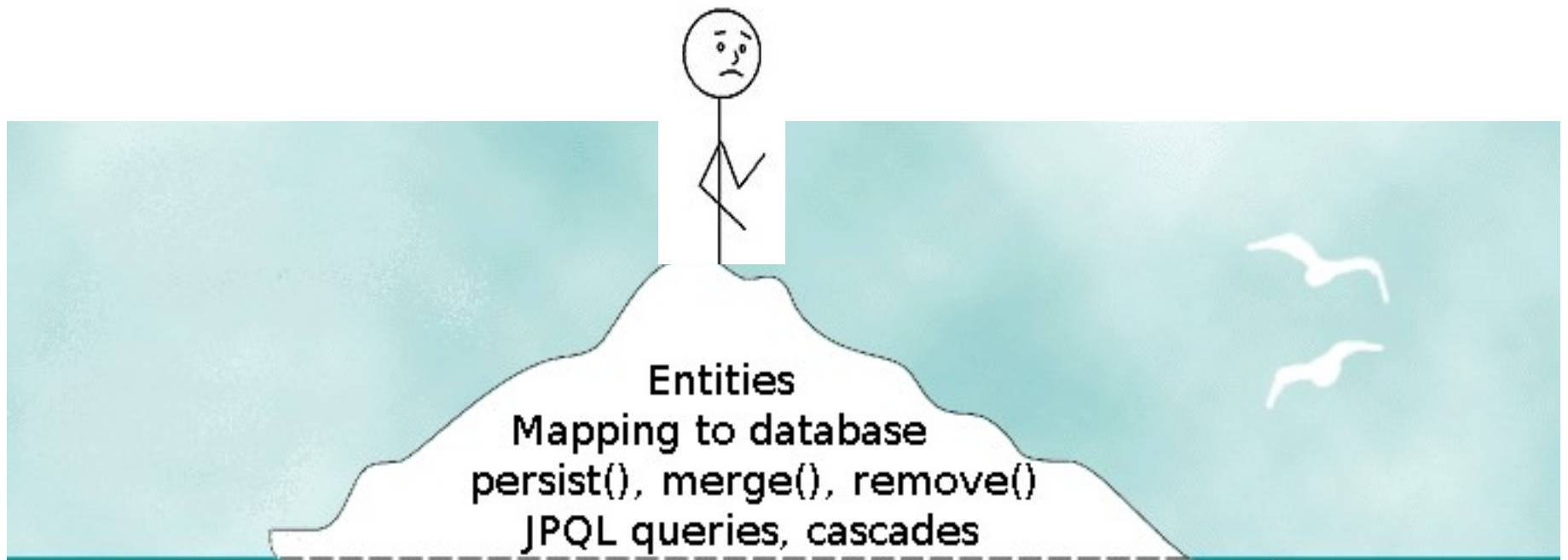
`persist()`, `merge()`, `remove()`

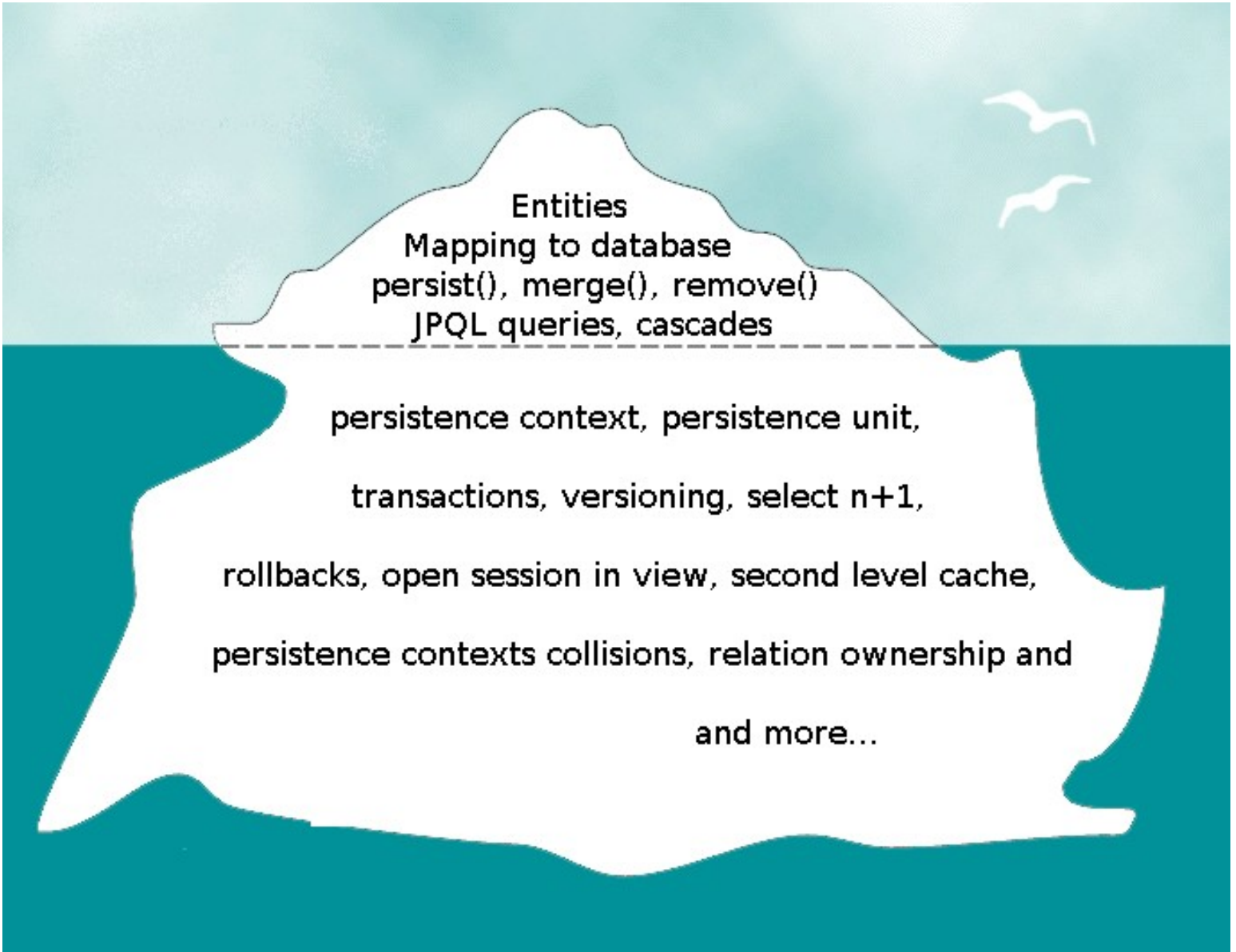
JPQL queries, cascading

pawel@rabbitsoftware.pl
@paulszulc



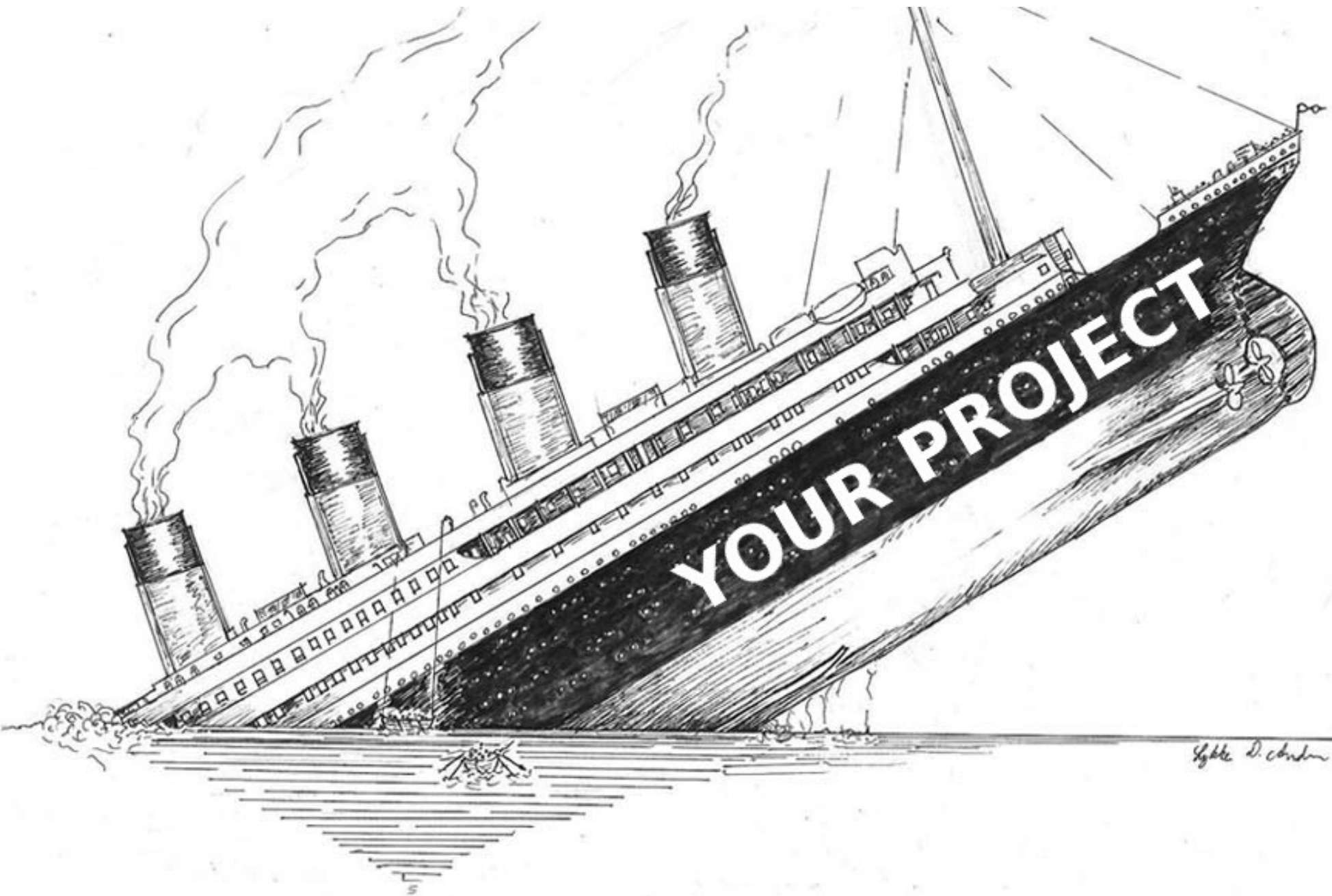
World of JPA

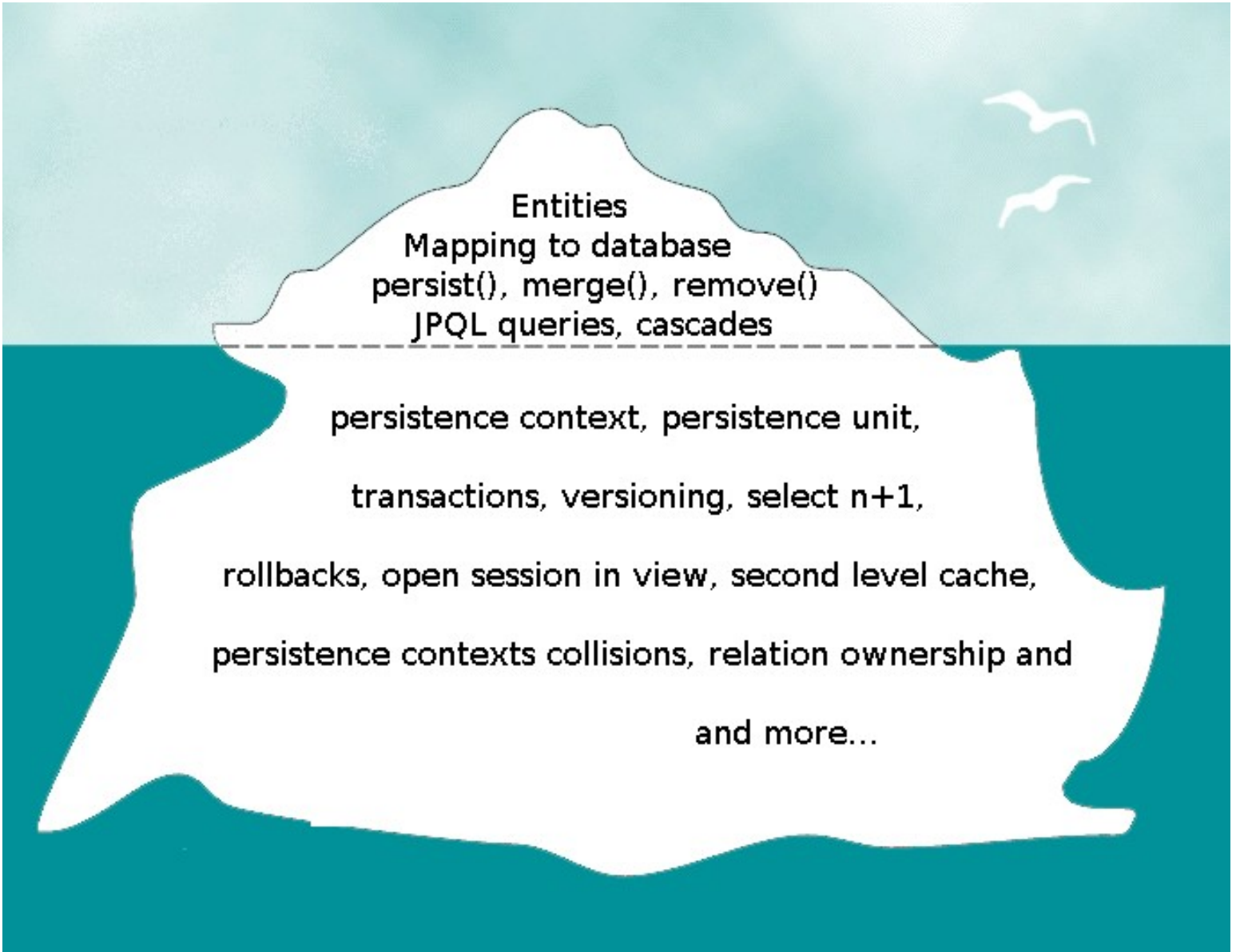


An iceberg floating in a teal sea. The tip of the iceberg is above the water line, and the much larger part is submerged. The sky is a lighter teal with two white birds flying. A dashed horizontal line separates the visible tip from the submerged part.

Entities
Mapping to database
persist(), merge(), remove()
JPQL queries, cascades

persistence context, persistence unit,
transactions, versioning, select n+1,
rollbacks, open session in view, second level cache,
persistence contexts collisions, relation ownership and
and more...



An iceberg floating in a teal sea. The tip of the iceberg is above the water line, and the much larger part is submerged. The sky is a lighter teal with two white birds flying. A dashed horizontal line separates the visible tip from the submerged part.

Entities
Mapping to database
persist(), merge(), remove()
JPQL queries, cascades

persistence context, persistence unit,
transactions, versioning, select n+1,
rollbacks, open session in view, second level cache,
persistence contexts collisions, relation ownership and
and more...

persistency context,
entity manager
and
transactions

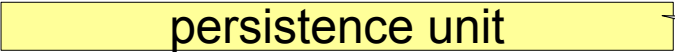
persistence unit

pawel@rabbitsoftware.pl
@paulszulc

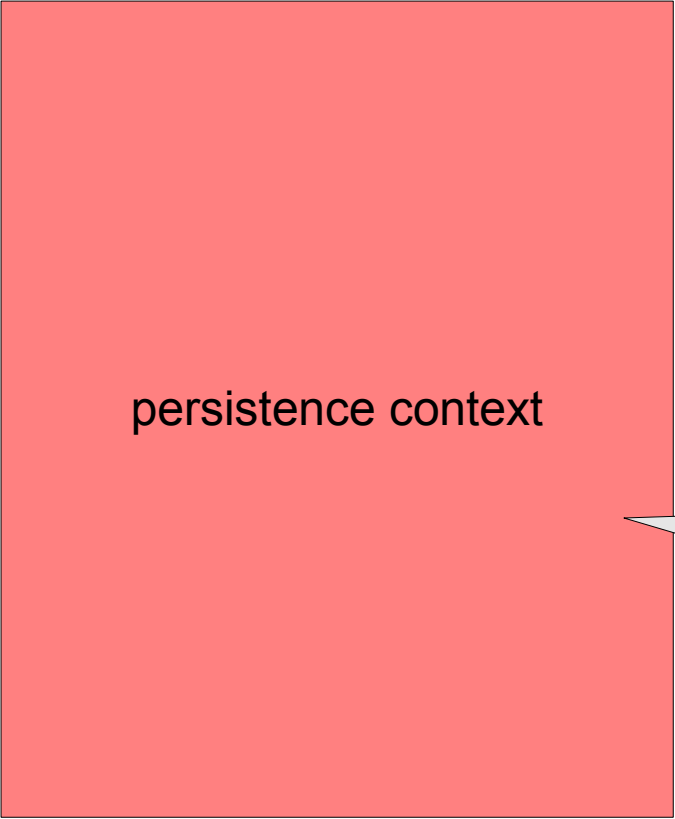
persistence unit

Named configuration

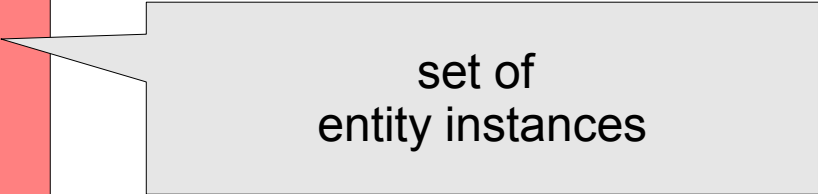
pawel@rabbitsoftware.pl
@paulszulc



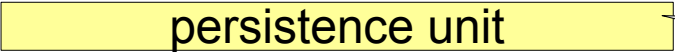
pawel@rabbitsoftware.pl
@paulszulc



persistence context



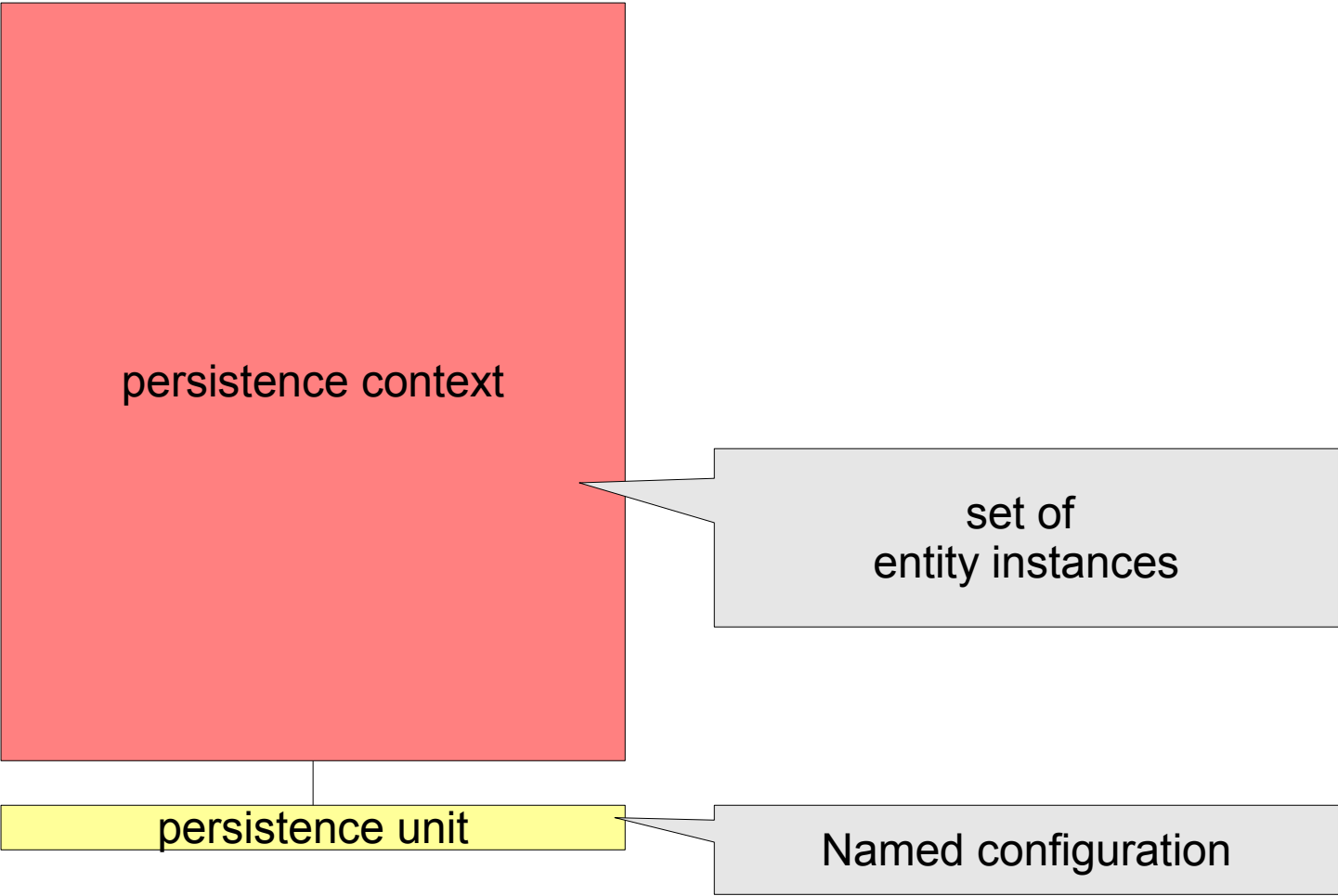
set of
entity instances



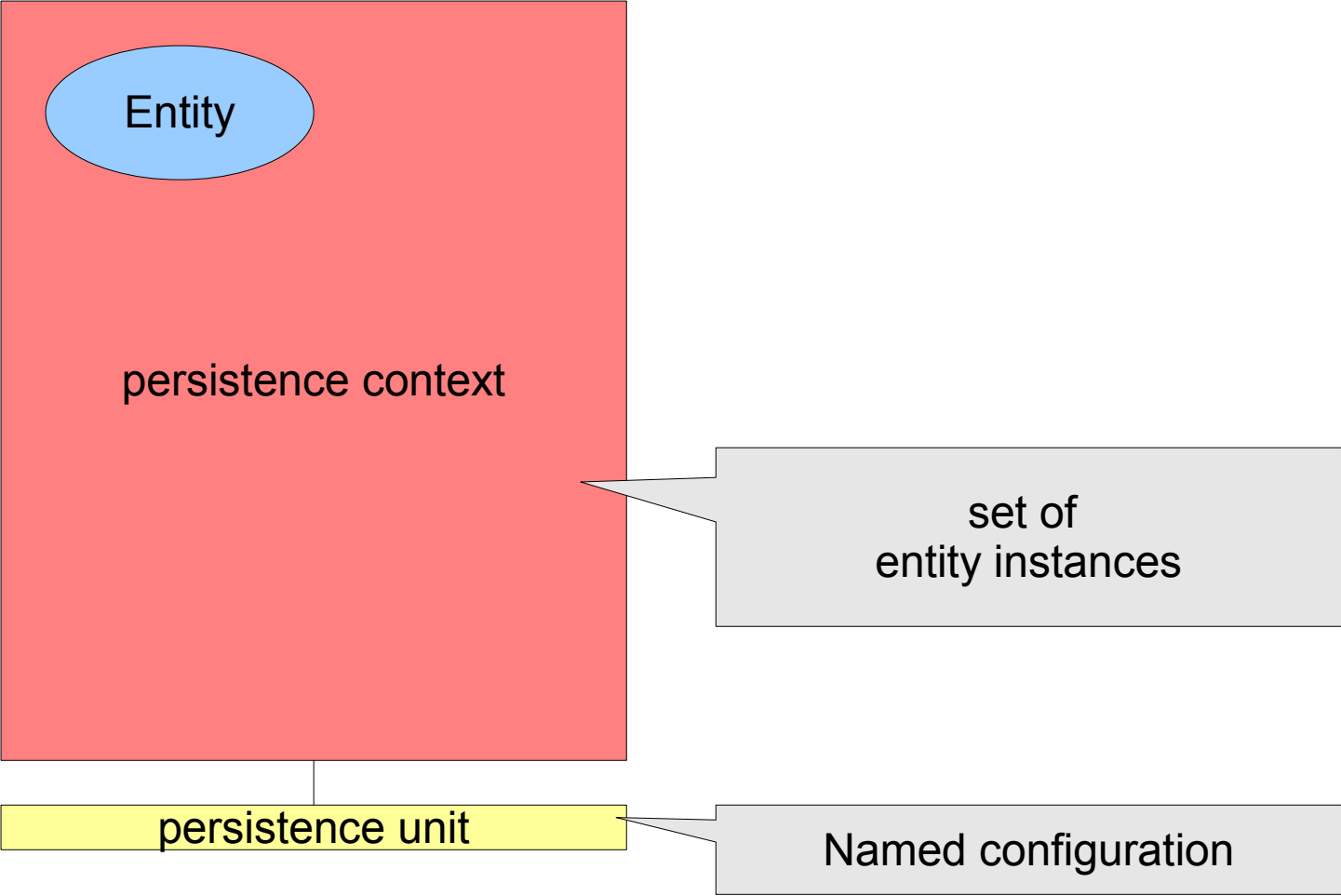
persistence unit

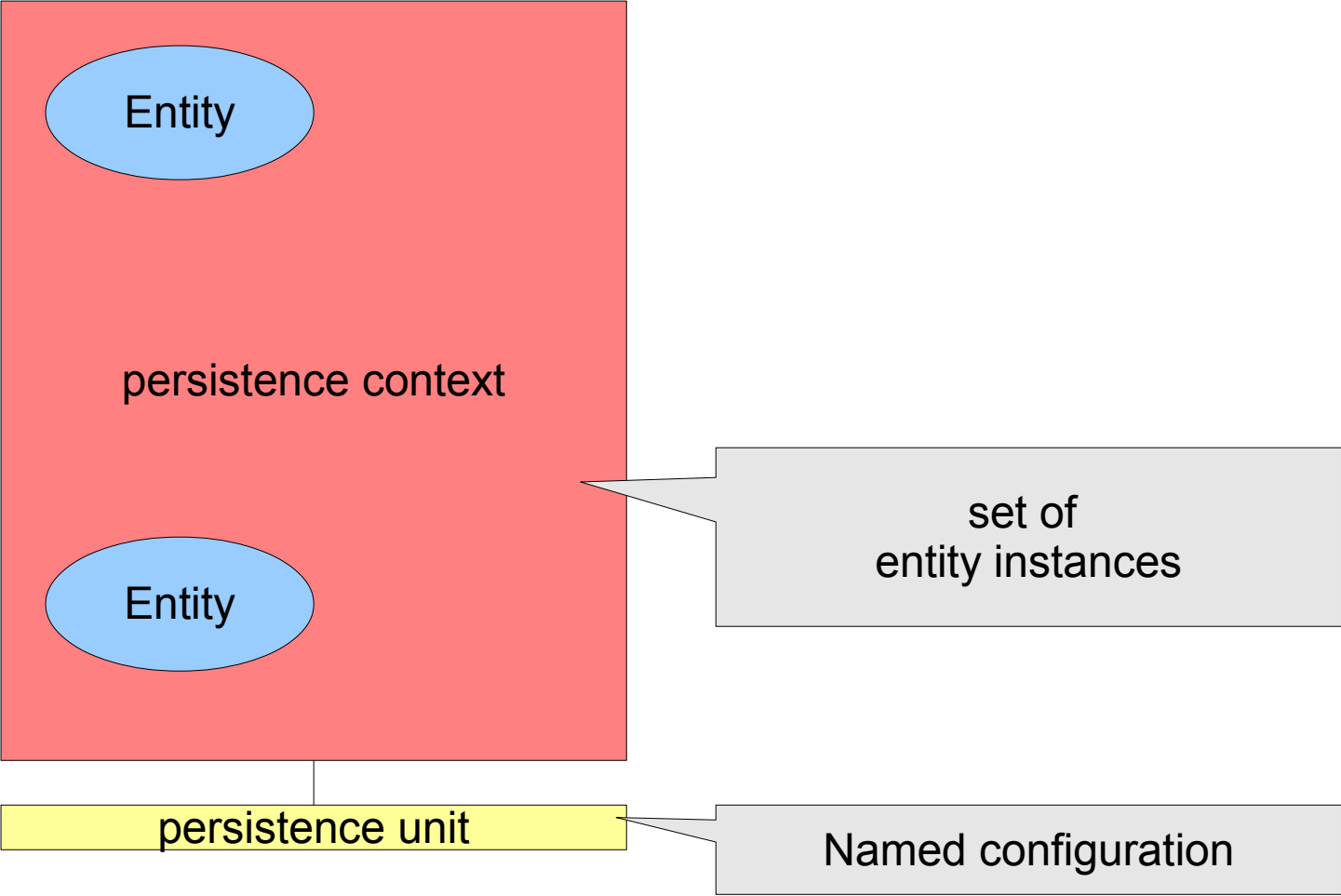


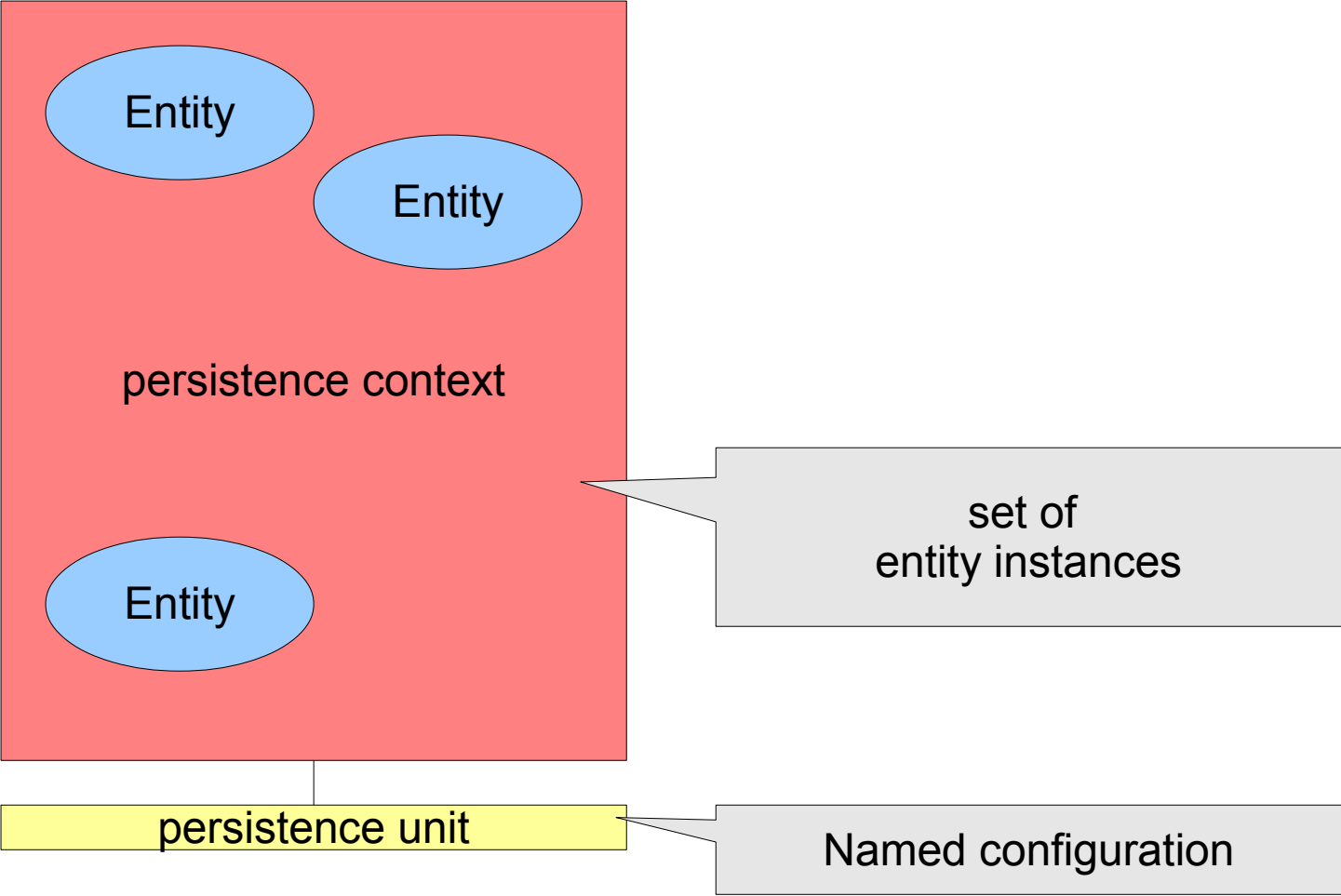
Named configuration

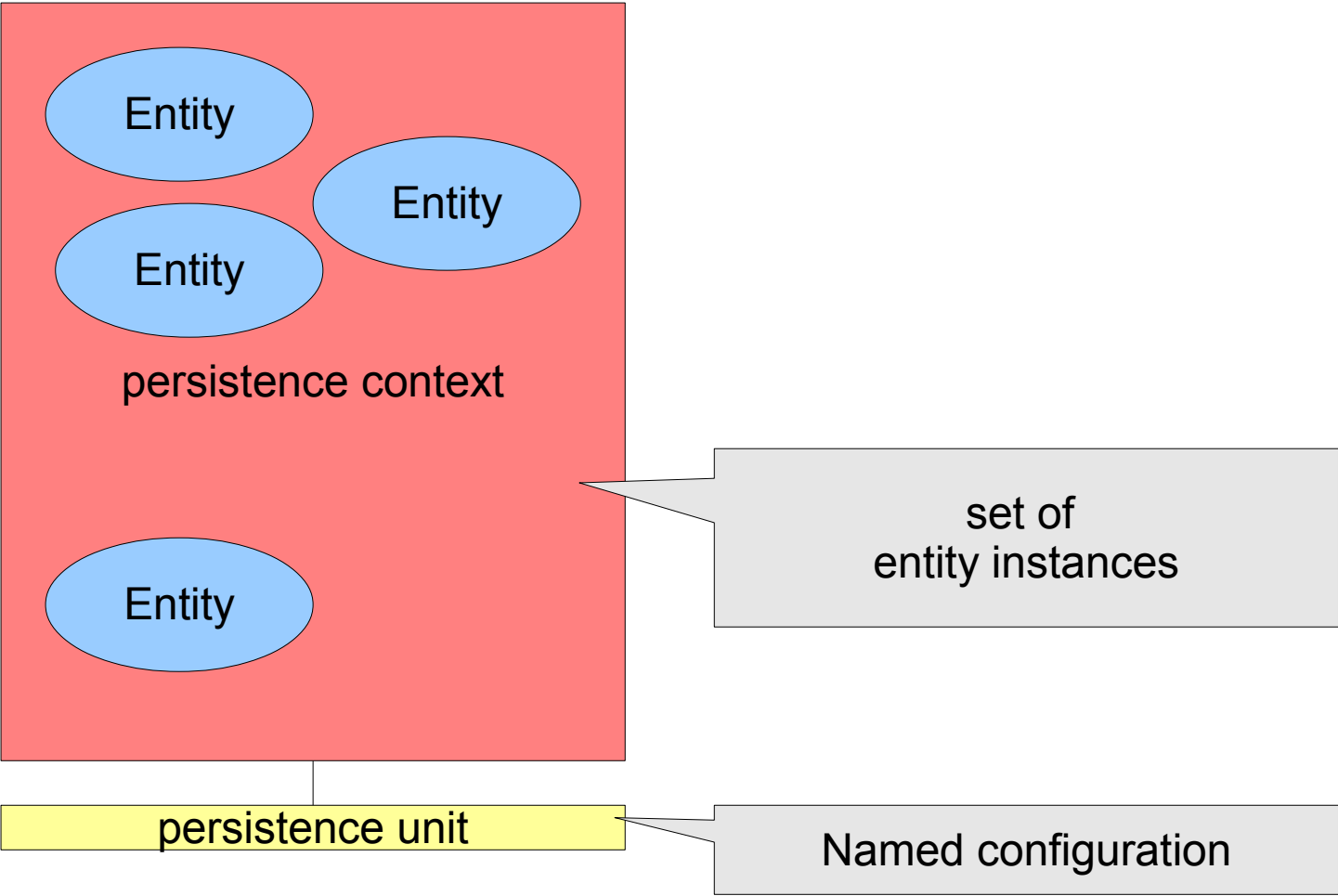


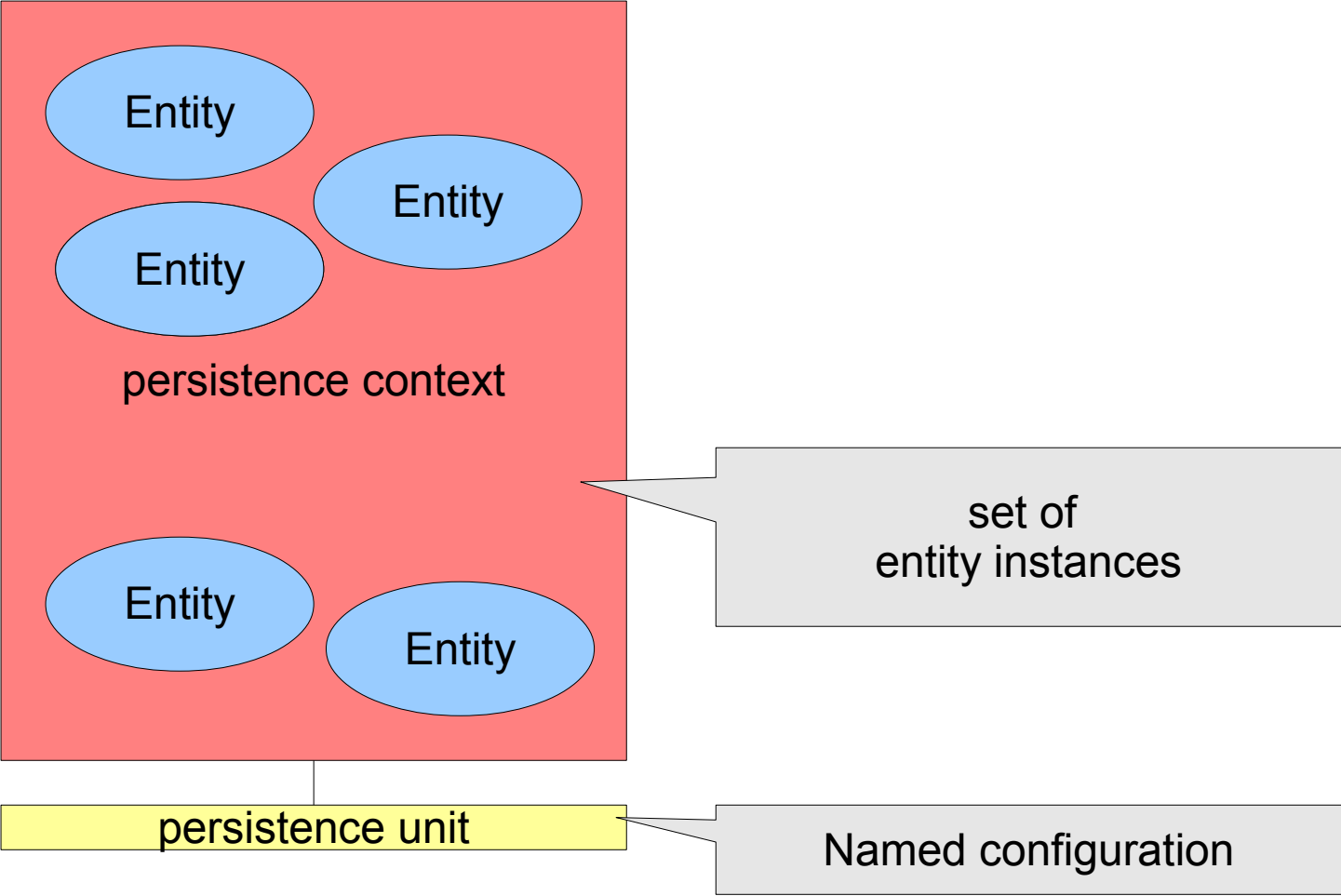
pawel@rabbitsoftware.pl
@paulszulc

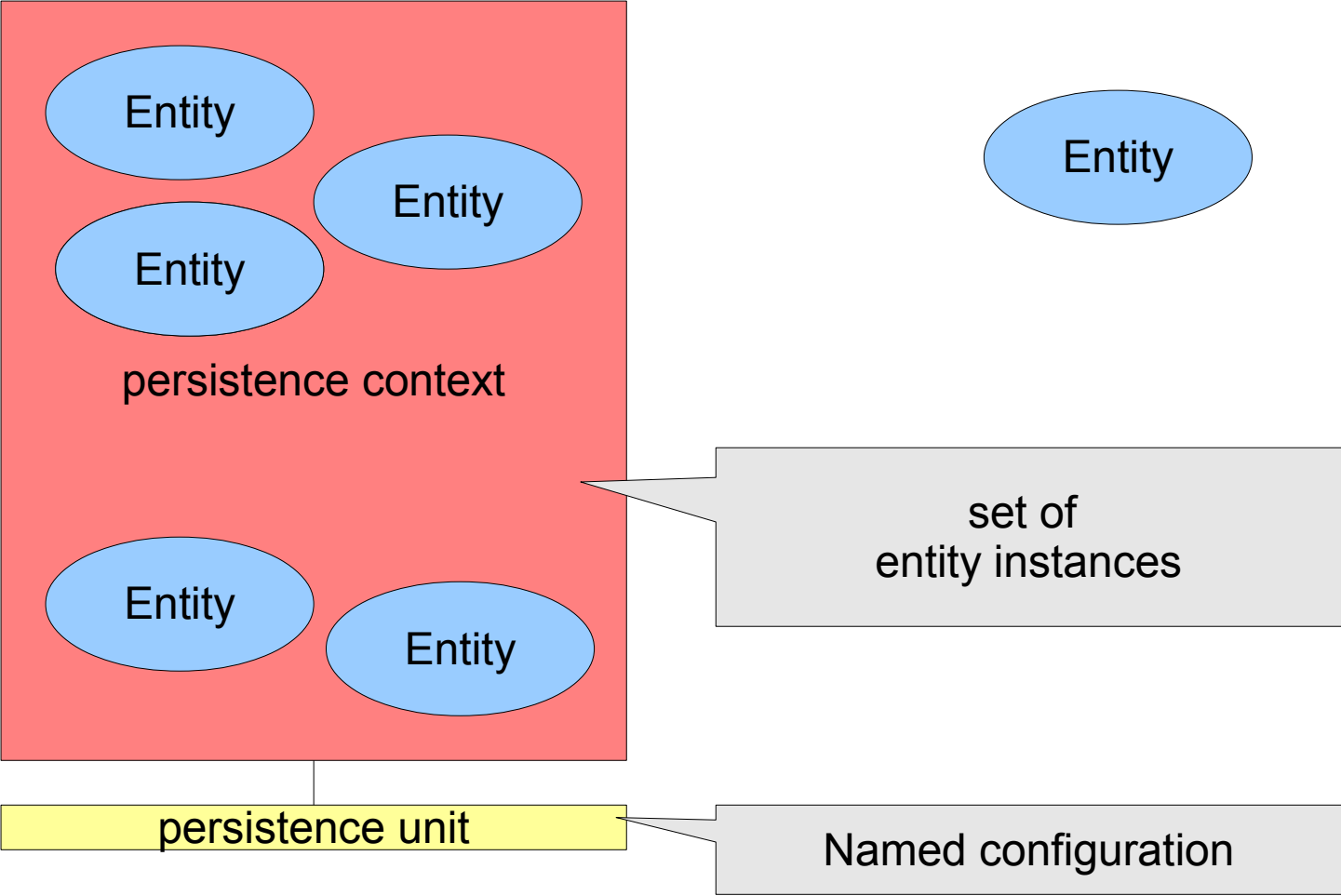


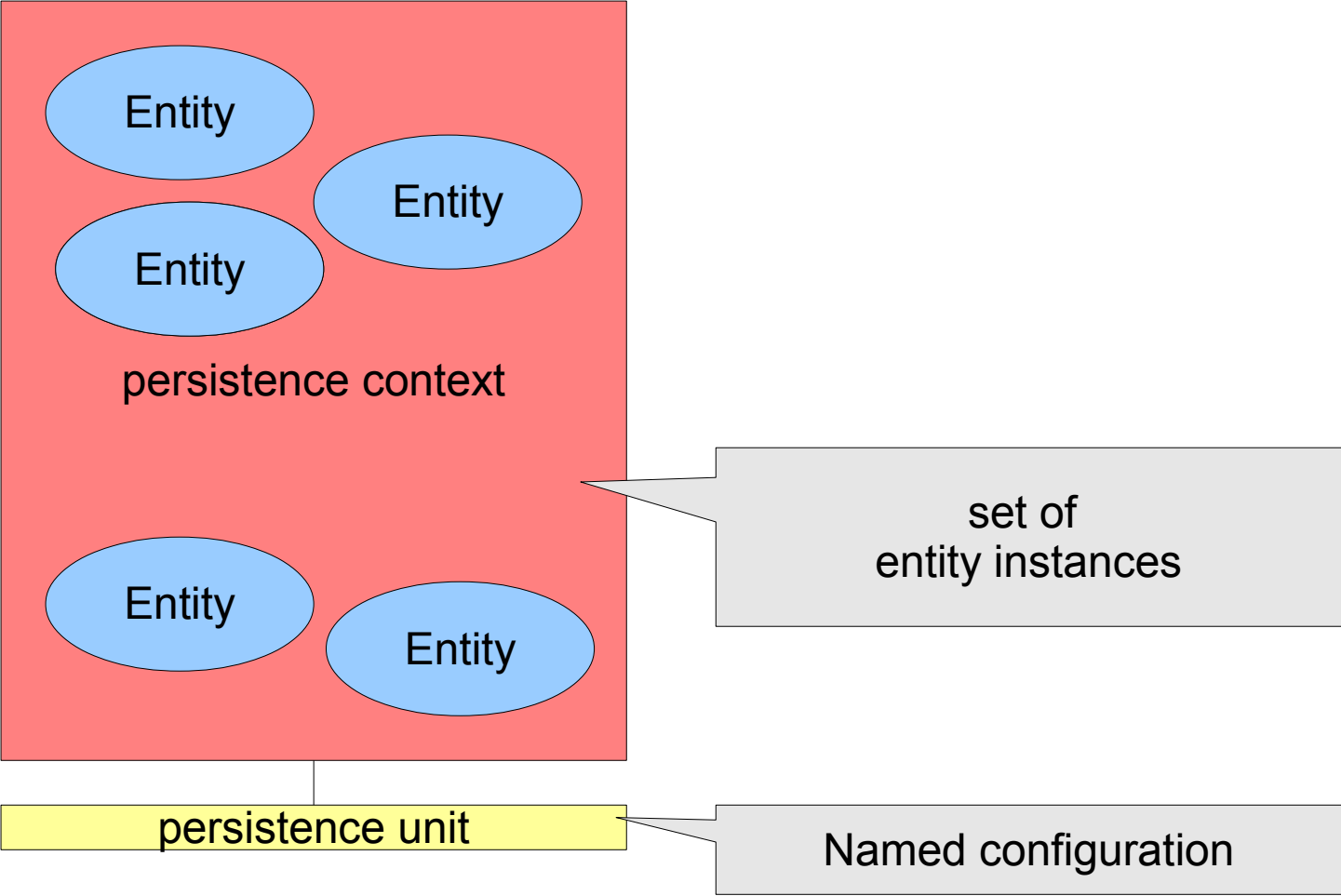




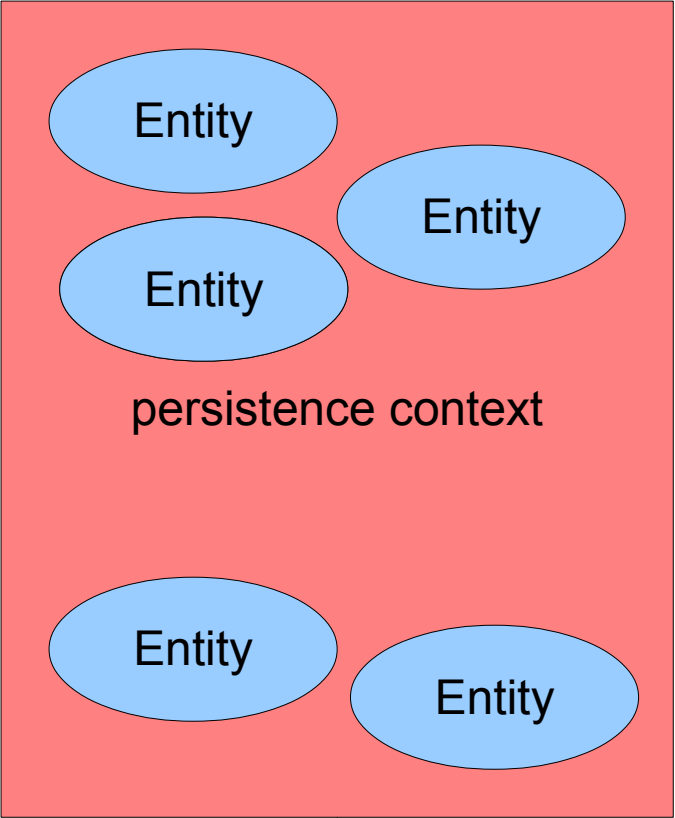






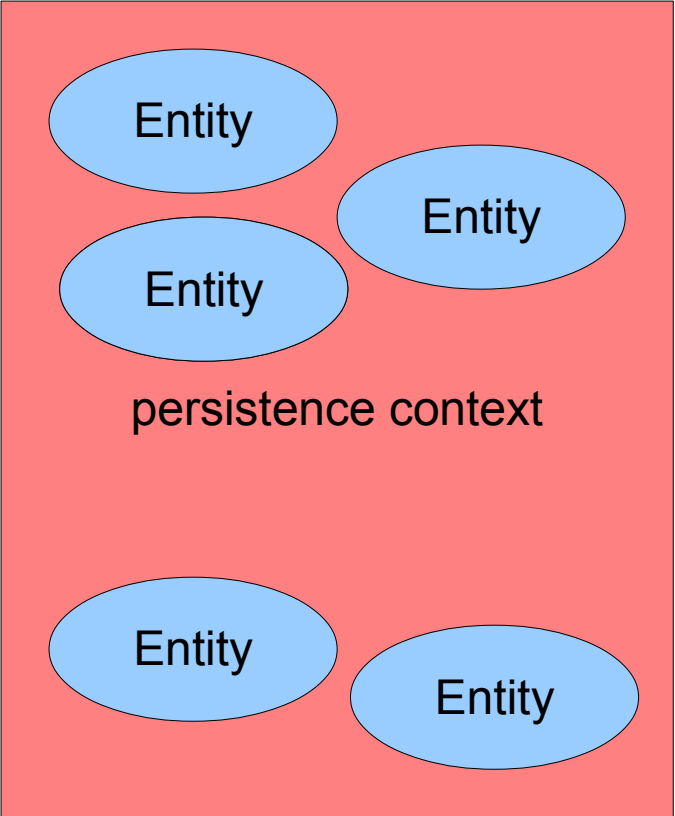


Entity manager



persistence unit

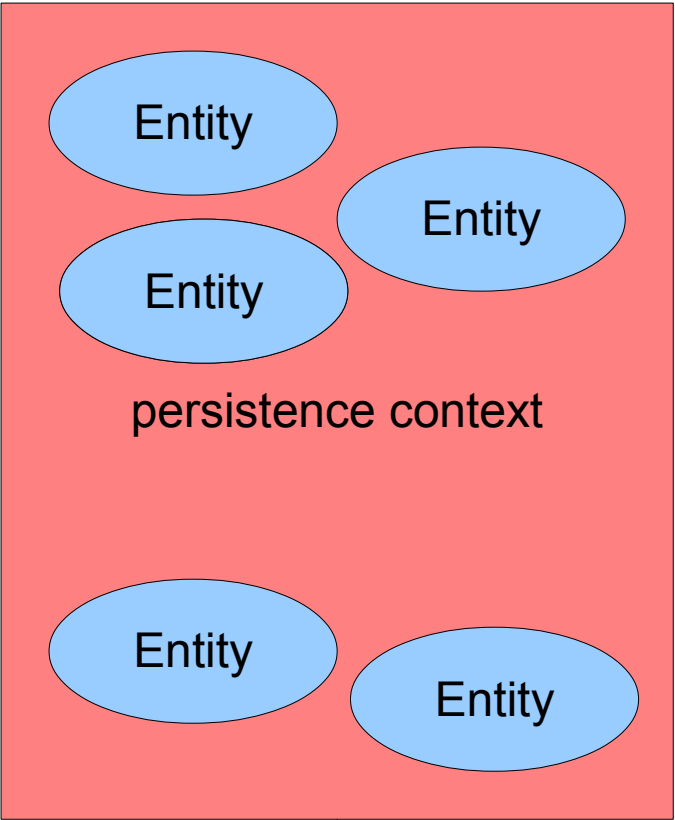
Entity manager



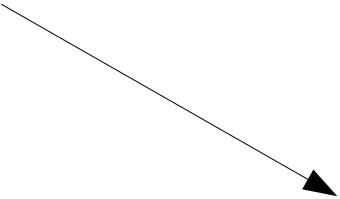
persistence unit



Entity manager



persistence unit



Entity Manager configurations

pawel@rabbitsoftware.pl
@paulszulc

Entity Manager configurations

- Container-Managed Entity Managers

Entity Manager configurations

- Container-Managed Entity Managers
- Application-Managed Entity Managers

Entity Manager configurations

- Container-Managed Entity Managers
 - Transaction-scoped
- Application-Managed Entity Managers

Entity Manager configurations

- Container-Managed Entity Managers
 - Transaction-scoped
 - Extended
- Application-Managed Entity Managers

Entity Manager configurations

- Container-Managed Entity Managers
 - Transaction-scoped
 - Extended
- Application-Managed Entity Managers

Transaction Scoped Entity Manager

pawel@rabbitsoftware.pl
@paulszulc

Transaction Scoped Entity Manager

```
public class LibraryService {  
    @PersistenceContext  
    private EntityManager em;  
  
    @Transactional  
    public void assignBook(int userId, int bookId) {  
        Book book = em.find(Book.class, bookId);  
        User user = em.find(User.class, userId);  
        user.borrowBook(book);  
    }  
}
```

Transaction Scoped Entity Manager

```
public class LibraryService {  
    @PersistenceContext  
    private EntityManager em;  
  
    @Transactional  
    public void assignBook(int userId, int bookId) {  
        Book book = em.find(Book.class, bookId);  
        User user = em.find(User.class, userId);  
        user.borrowBook(book);  
    }  
}
```

Transaction Scoped Entity Manager

```
public class LibraryService {  
    @PersistenceContext  
    private EntityManager em;  
  
    @Transactional  
    public void assignBook(int userId, int bookId) {  
        Book book = em.find(Book.class, bookId);  
        User user = em.find(User.class, userId);  
        user.borrowBook(book);  
    }  
}
```



Transaction Scoped Entity Manager

```
@Transactional
public void assignBook(int userId, int bookId) {
    Book book = em.find(Book.class, bookId);
    User user = em.find(User.class, userId);
    user.borrowBook(book);
}
```

Transaction Scoped Entity Manager

```
@Transactional
public void assignBook(int userId, int bookId) {
    Book book = em.find(Book.class, bookId);
    User user = em.find(User.class, userId);
    user.borrowBook(book);
}
```

Entity manager

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager



container

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager

container

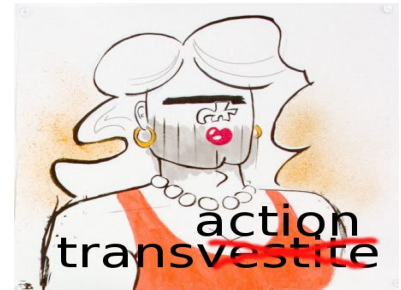


Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager

container



pawel@rabbitsoftware.pl
@paulszulc

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId) ;  
    User user = em.find(User.class, userId) ;  
    user.borrowBook(book) ;  
}
```

Entity manager

container



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId) ;  
    User user = em.find(User.class, userId) ;  
    user.borrowBook(book) ;  
}
```

Entity manager

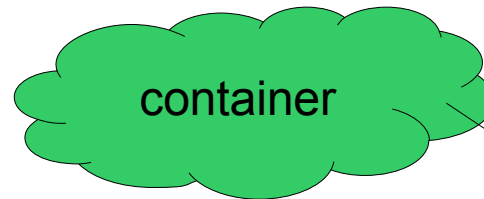
container



Transaction Scoped Entity Manager

```
@Transactional
public void assignBook(int userId, int bookId) {
    Book book = em.find(Book.class, bookId);
    User user = em.find(User.class, userId);
    user.borrowBook(book);
}
```

Entity manager



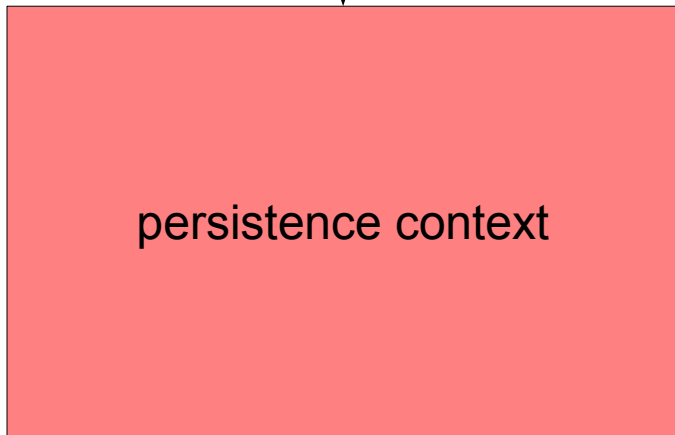
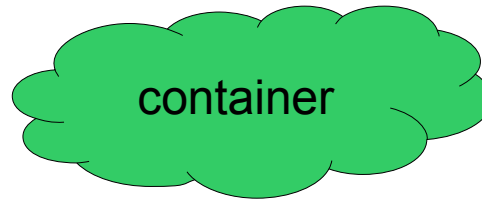
I don't have any! Pinky swear!



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId) ;  
    User user = em.find(User.class, userId) ;  
    user.borrowBook(book) ;  
}
```

Entity manager



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId) ;  
    User user = em.find(User.class, userId) ;  
    user.borrowBook(book) ;  
}
```

Entity manager

container

Book

persistence context



pawel@rabbitsoftware.pl
@paulszulc

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager

container

Book

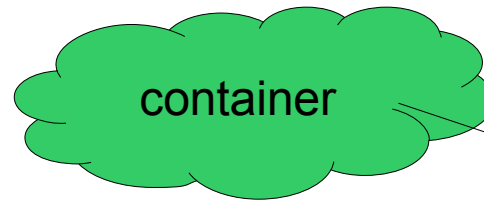
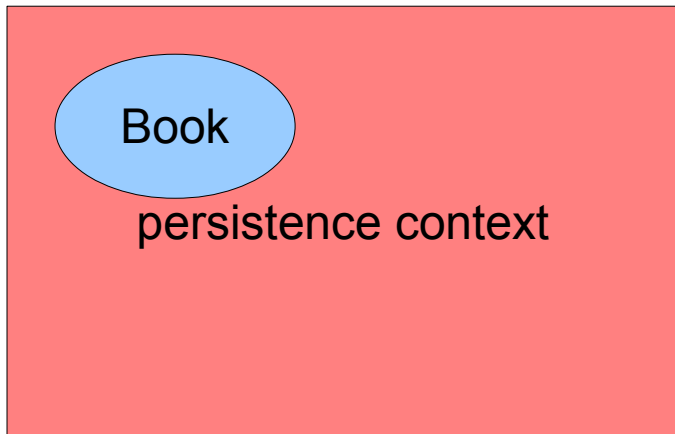
persistence context



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

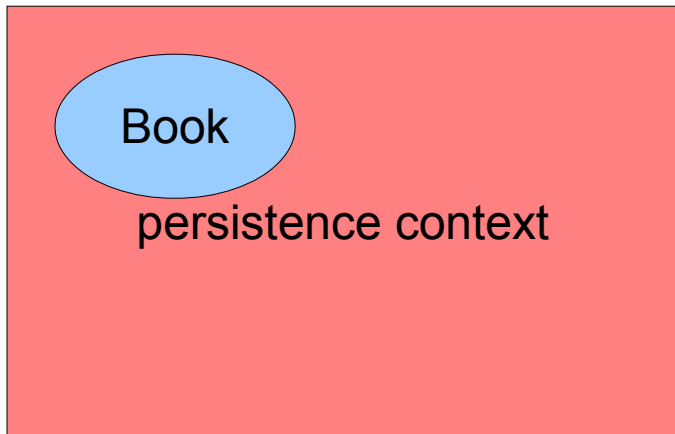
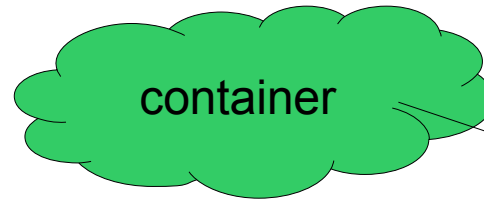
Entity manager



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

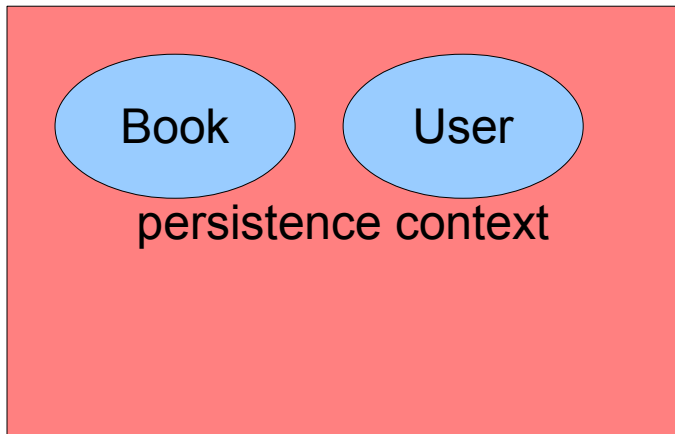
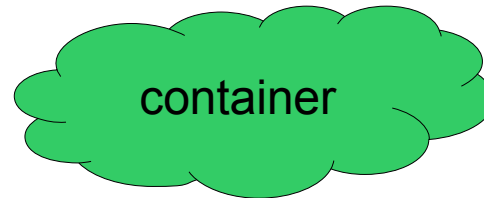
Entity manager



Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager



pawel@rabbitsoftware.pl
@paulszulc

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager

container



Book User

persistence context

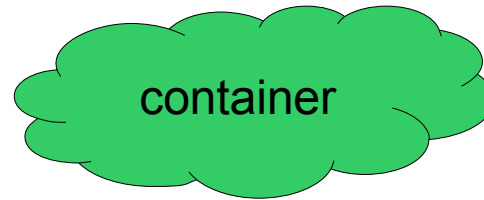


pawel@rabbitsoftware.pl
@paulszulc

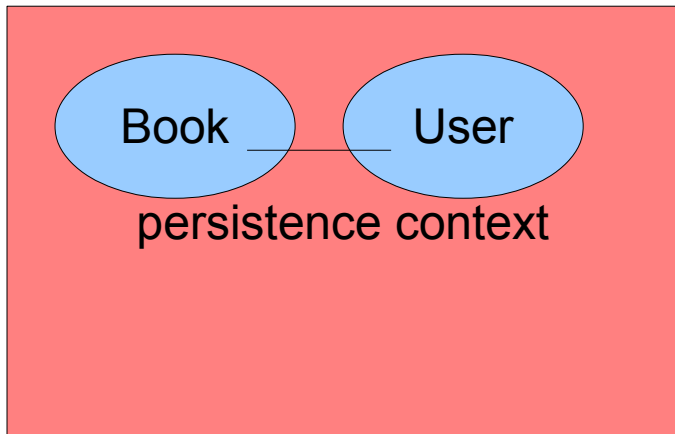
Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager



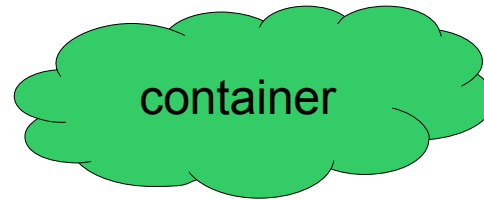
COMMIT



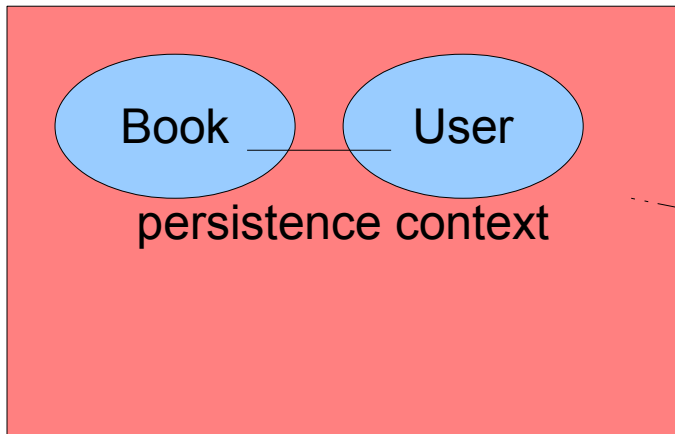
Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager



COMMIT

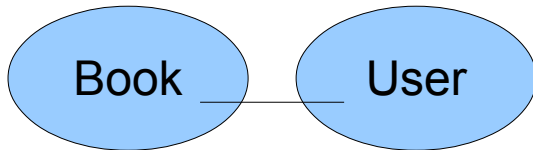
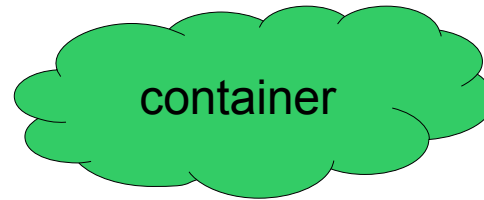


pawel@rabbitsoftware.pl
@paulszulc

Transaction Scoped Entity Manager

```
@Transactional  
public void assignBook(int userId, int bookId) {  
    Book book = em.find(Book.class, bookId);  
    User user = em.find(User.class, userId);  
    user.borrowBook(book);  
}
```

Entity manager



Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional // method from class BookService
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional // method from class BookService
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional // method from class BookService
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional // method from class BookService
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional // method from class AuditService
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional // method from class BookService
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if (em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Transaction Scoped Entity Manager

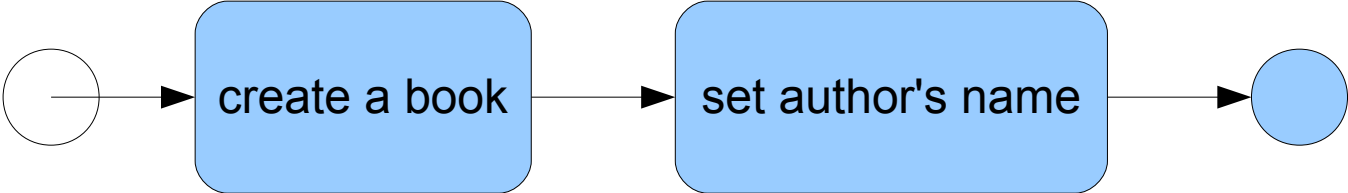
```
@Transactional(propagation = Propagation.REQUIRES_NEW)
public void logActionOnBook(int bookId, String action) {
    if(em.find(Book.class, bookId) == null) {
        throw new IllegalArgumentException("unknown book");
    }
    em.persist(new ActionLog(bookId, action));
}
```

```
@Transactional
public void createBook(Book book) {
    em.persist(book);
    auditService.logActionOnBook(book.getId(), "created");
}
```

Extended Entity Manager

pawel@rabbitsoftware.pl
@paulszulc

Extended Entity Manager



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

pawel@rabbitsoftware.pl
@paulszulc

Extended Entity Manager

@Stateful

Entity manager

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```



Extended Entity Manager

```
@Stateful
```

Entity manager

```
public class BookCreator {
```

```
    @PersistenceContext
```

```
    private EntityManager em;
```

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

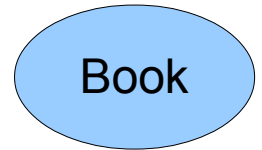
```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
@Remove public void finished() { }
```

```
}
```



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Entity manager

persistence context

Books

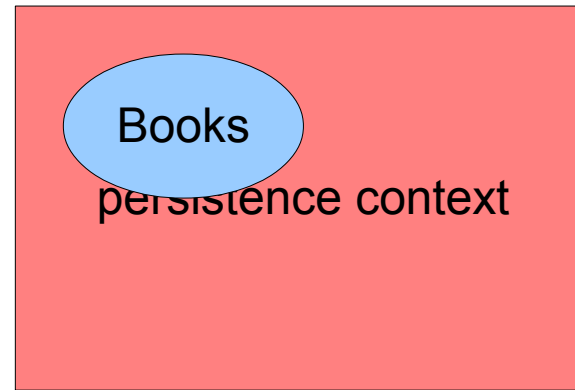


Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Entity manager

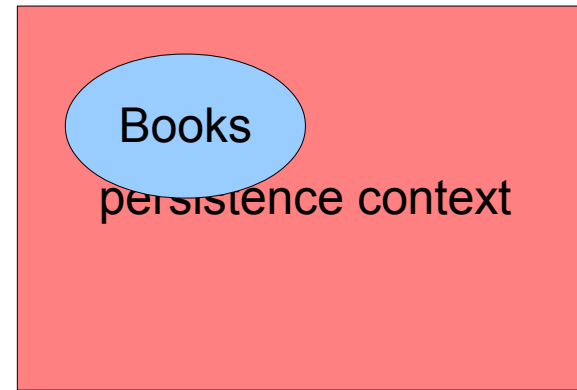


Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Entity manager

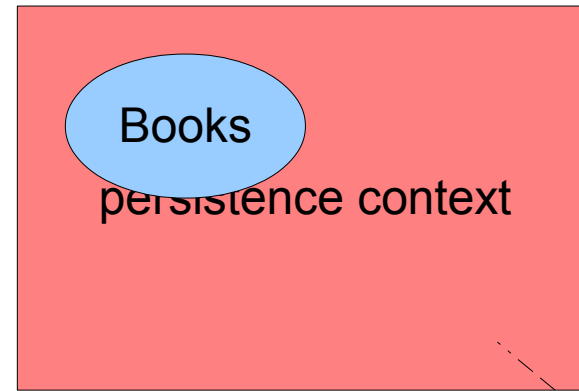


Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Entity manager



Extended Entity Manager

```
@Stateful
```

Entity manager

```
public class BookCreator {
```

```
    @PersistenceContext
```

```
    private EntityManager em;
```

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
@Remove public void finished() { }
```

```
}
```



Extended Entity Manager

```
@Stateful
```

Entity manager

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Books



Extended Entity Manager

```
@Stateful
```

Entity manager

```
public class BookCreator {
```

```
    @PersistenceContext
```

```
    private EntityManager em;
```

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

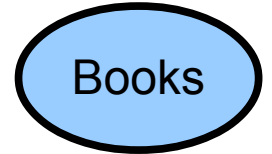
```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
@Remove public void finished() { }
```

```
}
```



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

pawel@rabbitsoftware.pl
@paulszulc

Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext (type=PersistenceContextType.EXTENDED)  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

pawel@rabbitsoftware.pl
@paulszulc

Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

persistence context

```
    }
```

```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
    @Remove public void finished() { }
```

```
}
```

pawel@rabbitsoftware.pl
@paulszulc



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

persistence context

Books

```
}
```

```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
}
```

```
@Remove public void finished() { }
```

```
}
```

pawel@rabbitsoftware.pl
@paulszulc



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

Books

persistence context

```
}
```

```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
}
```

```
@Remove public void finished() { }
```

```
}
```

pawel@rabbitsoftware.pl
@paulszulc



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {  
    @PersistenceContext(type=PersistenceContextType.EXTENDED)  
    private EntityManager em;  
    private Book book;  
    public void createBook() {  
        book = new Book();  
        em.persist(book);  
    }  
    public void setAuthorsName(String name) {  
        book.setAuthorsName(name);  
    }  
    @Remove public void finished() { }  
}
```

Entity manager

Books

persistence context



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

```
    public void setAuthorsName(String name) {
```

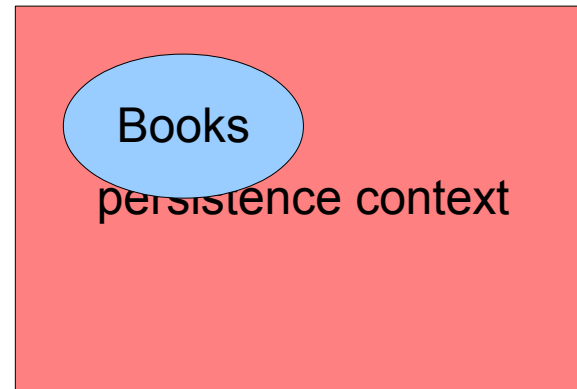
```
        book.setAuthorsName(name);
```

```
    }
```

```
    @Remove public void finished() { }
```

```
}
```

pawel@rabbitsoftware.pl
@paulszulc



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

```
    public void setAuthorsName(String name) {
```

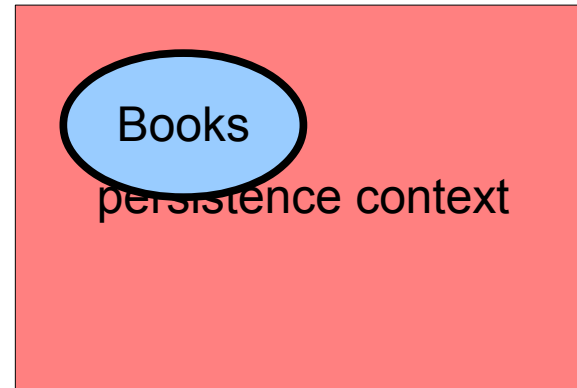
```
        book.setAuthorsName(name);
```

```
    }
```

```
    @Remove public void finished() { }
```

```
}
```

pawel@rabbitsoftware.pl
@paulszulc



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

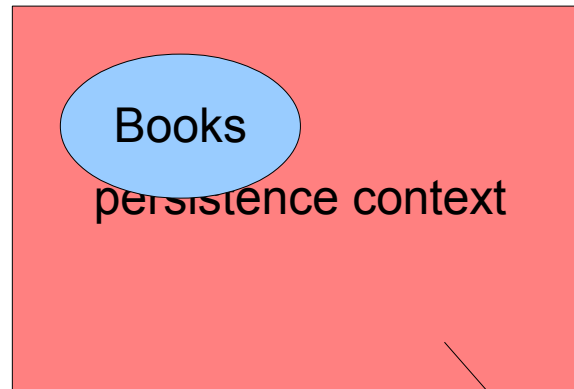
```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
    @Remove public void finished() { }
```

```
}
```



Extended Entity Manager

```
@Stateful
```

```
public class BookCreator {
```

```
    @PersistenceContext(type=PersistenceContextType.EXTENDED)
```

```
    private EntityManager em;
```

Entity manager

```
    private Book book;
```

```
    public void createBook() {
```

Books

```
        book = new Book();
```

```
        em.persist(book);
```

```
    }
```

```
    public void setAuthorsName(String name) {
```

```
        book.setAuthorsName(name);
```

```
    }
```

```
@Remove public void finished() { }
```

```
}
```


pawel@rabbitsoftware.pl
@paulszulc



persistence context collisions


pawel@rabbitsoftware.pl
@paulszulc

persistence context collisions



stateless bean

persistence context collisions



stateless bean



stateful bean

persistence context collisions



stateless bean

stateful bean

persistence context collisions

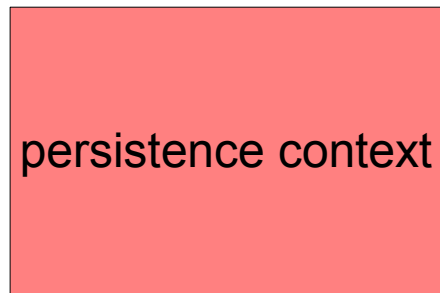
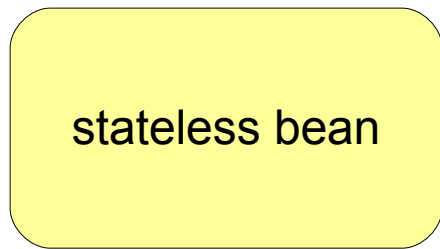


stateless bean

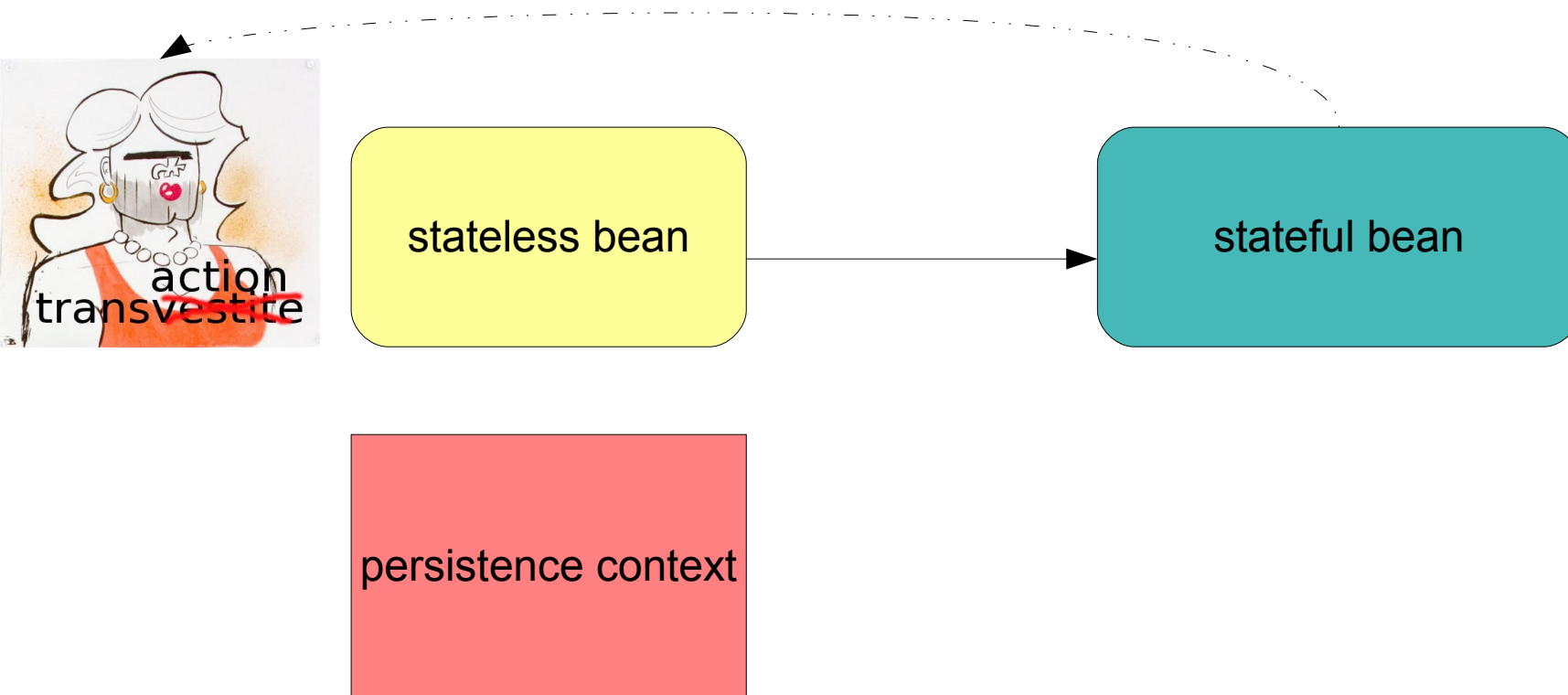
stateful bean

persistence context

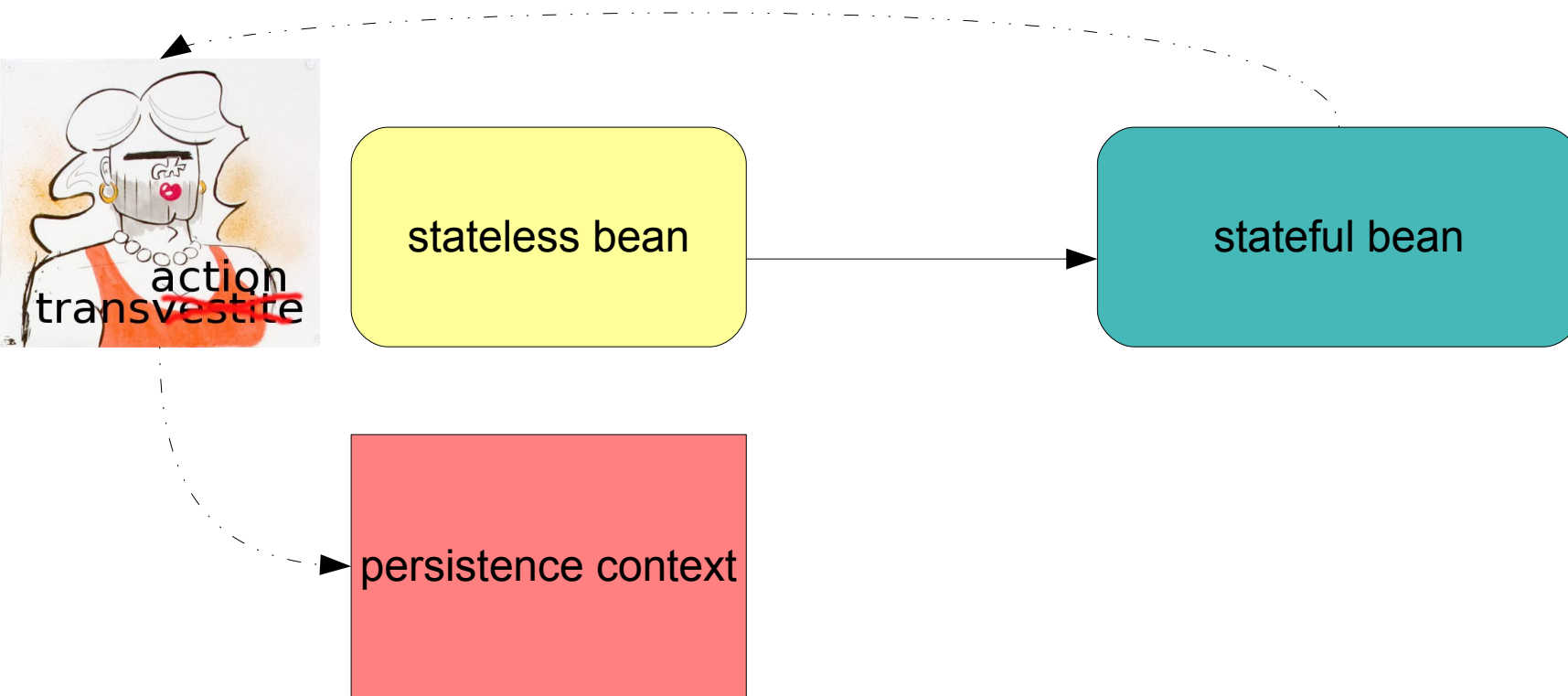
persistence context collisions



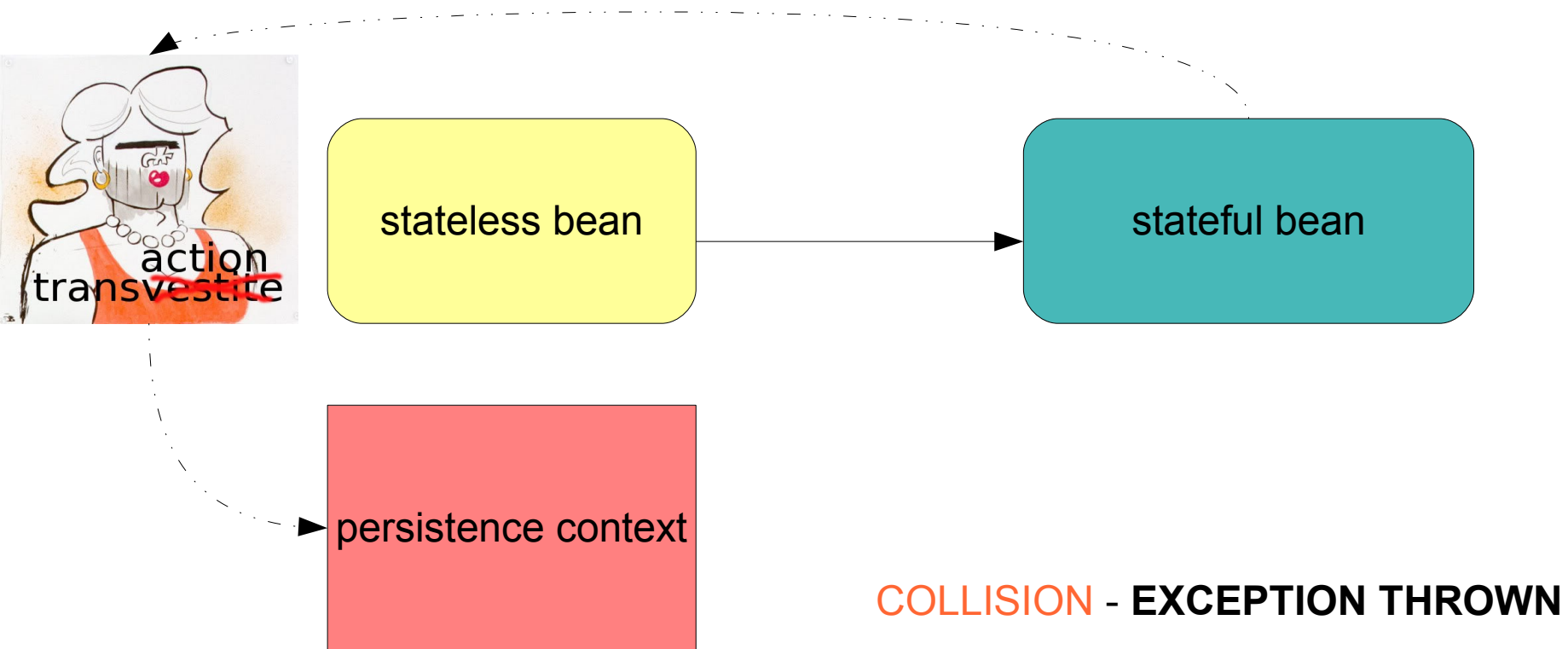
persistence context collisions



persistence context collisions



persistence context collisions



ownership of relations

pawel@rabbitsoftware.pl
@paulszulc

ownership of relations

dog owner

pawel@rabbitsoftware.pl
@paulszulc

ownership of relations



dog owner

ownership of relations



dog owner

dogs

ownership of relations



dog owner



dogs

ownership of relations



dog owner



dogs



owner has many dogs
dog has one owner

pawel@rabbitsoftware.pl
@paulszulc

ownership of relations

Id	Name
1	Paris

Id	Name	Owner_id
1	Smurkles	1
2	Puppkins	1
3	Chunky	1

ownership of relations

```
Owner paris = em.find(Owner.class, 1);
```

```
Dog szarik = new Dog("Szarik");
```

```
paris.addDog(szarik);
```

ownership of relations

```
Owner paris = em.find(Owner.class, 1);
```

```
Dog szarik = new Dog("Szarik");
```

```
paris.addDog(szarik);
```

```
public void addDog(Dog dog) {
```

```
    dogs.add(dog);
```

```
    dog.setOwner(this);
```

```
}
```

ownership of relations

```
Owner paris = em.find(Owner.class, 1);
```

```
Dog szarik = new Dog("Szarik");
```

```
szarik.setOwner(paris);
```

ownership of relations

Id	Name
1	Paris

Id	Name	Owner_id
1	Smurkles	1
2	Puppkins	1
3	Chunky	1
4	Szarik	1

dog owner

ownership of relations

```
Owner paris = em.find(Owner.class, 1);  
Dog szarik = new Dog("Szarik");  
szarik.setOwner(paris);
```

```
public class Owner {  
    @OneToMany(cascade = {CascadeType.ALL})  
    private Collection<Dog> dogs;
```

association class mapping

association class mapping

```
public class Candidate {
```

association class mapping

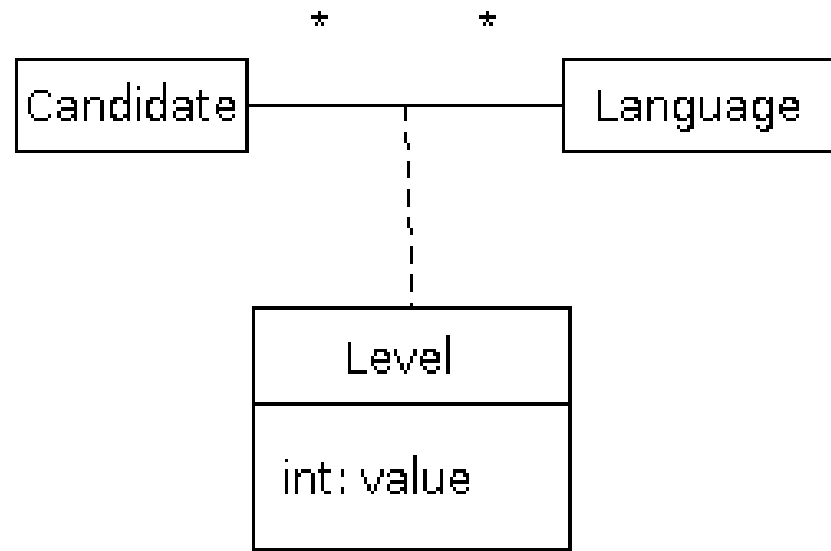
```
public class Candidate {
```

```
public class Language {
```

association class mapping

```
public class Candidate {  
    @ManyToMany  
    private Collection<Language> languages;  
  
public class Language {
```

association class mapping



association class mapping

```
public class Candidate {  
    @ManyToMany  
    private Collection<Language> languages;  
  
}
```

```
public class Language {  
  
}
```


association class mapping

```
public class Candidate {  
    @OneToMany  
    private Collection<LanguageLevel> languages;  
}
```

```
public class LanguageLevel {  
    @ManyToOne  
    private Language language;  
    private Level value;  
}
```

```
public class Language {
```

pawel@rabbitsoftware.pl
@paulszulc

association class mapping

```
public class Candidate {  
    @OneToMany  
    private Collection<LanguageLevel> languages;  
public boolean knowsLanguage(Language lang) { ?? }  
  
    public class LanguageLevel {  
        @ManyToOne  
        private Language language;  
        private Level value;  
  
    }  
  
    public class Language {
```

association class mapping

```
public class Candidate {  
    @ManyToMany  
    private Collection<Language> languages;  
  
}
```

```
public class Language {  
  
}
```

association class mapping

```
public class Candidate {
```

```
    private Map<Language, Level> languages;
```

```
public class Language {
```

association class mapping

```
public class Candidate {  
    @ElementCollection  
    private Map<Language, Level> languages;  
  
public class Language {
```

association class mapping

```
public class Candidate {  
    @ElementCollection  
    private Map<Language, Level> languages;  
public boolean knowsLanguage(Language lang) {  
    return languages.containsKey(lang);  
}
```

```
public class Language {
```

generator's allocation size

generator's allocation size

```
@Entity
public class User {
    @Id
    private Long id;
}
```

generator's allocation size

```
@Entity
```

```
public class User {
```

```
    @Id
```

```
    private Long id;
```

```
}
```

```
CREATE SEQUENCE user_seq
```

```
START WITH      100
```

```
INCREMENT BY    1
```

generator's allocation size

```
@Entity
```

```
public class User {
```

```
    @Id()
```

```
    @GeneratedValue(strategy = GenerationType.SEQUENCE,  
        generator = "user_seq_gen")
```

```
    private Long id;
```

```
}
```

```
CREATE SEQUENCE user_seq
```

```
START WITH      100
```

```
INCREMENT BY    1
```

generator's allocation size

```
@Entity
public class User {
    @Id()
    @GeneratedValue(strategy = GenerationType.SEQUENCE,
        generator = "user_seq_gen")
    @SequenceGenerator(sequenceName = "user_seq",
        name = "user_seq_gen")
    private Long id;
}
```

```
CREATE SEQUENCE user_seq
START WITH      100
INCREMENT BY   1
```

generator's allocation size

```
@Entity
```

```
public class User {
```

```
    @Id()
```

```
    @GeneratedValue(strategy = GenerationType.SEQUENCE,  
                    generator = "user_seq_gen")
```

```
    @SequenceGenerator(sequenceName = "user_seq",  
                       name = "user_seq_gen",
```

```
                       allocationSize = 1)
```

```
    private Long id;
```

```
}
```

```
CREATE SEQUENCE user_seq
```

```
START WITH      100
```

```
INCREMENT BY    1
```

select n + 1 problem

select n + 1 problem

```
List<Person> persons = repository.findAll();  
for(Person person : persons) {  
    addRow(person.getName(), person.getAddress().getCity());  
}
```

select n + 1 problem

```
List<Person> persons = repository.findAll();
for(Person person : persons) {
    addRow(person.getName(), person.getAddress().getCity());
}
```

```
select * from persons;
```

```
select * from addresses where person_id =
```

```
select * from addresses where person_id =
```

```
...
```

```
select * from addresses where person_id =
```

select n + 1 problem

```
public List<Person> findMeWithAddresses() {  
    String definition = "select p from Person JOIN FETCH p.address";  
    return em.createQuery(definition).getResultList();  
}
```

select n + 1 problem

```
List<Person> persons = repository.findAll();  
for(Person person : persons) {  
    addRow(person.getName(), person.getAddress().getCity());  
}
```

select n + 1 problem

```
List<Person> persons = personDao.findMeWithAddresses() ;  
for(Person person : persons) {  
    addRow(person.getName(), person.getAddress().getCity());  
}
```

select n + 1 problem

```
List<Person> persons = personDao.findMeWithAddresses();  
for(Person person : persons) {  
    addRow(person.getName(), person.getAddress().getCity());  
}
```

```
select * from persons JOIN addresses;
```

taking **vendor** for granted

pawel@rabbitsoftware.pl
@paulszulc

worth reading

- Pro JPA 2: Mastering the Java™ Persistence API
- <http://art-of-software.blogspot.com>
- stackoverflow.com

worth reading

- Pro JPA 2: Mastering the Java™ Persistence API
- <http://art-of-software.blogspot.com>
- stackoverflow.com

shameless self-promotion

Rabbit Software is doing JPA workshops :)

`www.rabbitsoftware.pl`

`pawel@rabbitsoftware.pl`
`@paulszulc`

thank **you** very much for listening
hoped you've enjoyed it

pawel@rabbitsoftware.pl
@paulszulc