Overview of the Second-Level Cache

A **second-level cache** is a local store of entity data managed by the persistence provider to improve application performance. A second-level cache helps improve performance by avoiding expensive database calls, keeping the entity data local to the application. A second-level cache is typically transparent to the application, as it is managed by the persistence provider and underlies the persistence context of an application. That is, the application reads and commits data through the normal entity manager operations without knowing about the cache.

Note - Persistence providers are not required to support a second-level cache. Portable applications should not rely on support by persistence providers for a second-level cache.

The second-level cache for a persistence unit may be configured to one of several second-level cache modes. The following cache mode settings are defined by the Java Persistence API.

Table 38-1 Cache Mode Settings for the Second-Level Cache

Cache Mode Setting	**Description**
`ALL`	All entity data is stored in the second-level cache for this persistence unit.
`NONE`	No data is cached in the persistence unit. The persistence provider must not cache any data.
`ENABLE_SELECTIVE`	Enable caching for entities that have been explicitly set with the `@Cacheable`annotation.
`DISABLE_SELECTIVE`	Enable caching for all entities except those that have been explicitly set with the`@Cacheable(false)` annotation.
`UNSPECIFIED`	The caching behavior for the persistence unit is undefined. The persistence provider’s default caching behavior will be used.

One consequence of using a second-level cache in an application is that the underlying data may have changed in the database tables, while the value in the cache has not, a circumstance called a **stale read**. Stale reads may be avoided by changing the second-level cache to one of the cache mode settings, controlling which entities may be cached (described in Controlling Whether Entities May Be Cached), or changing the cache’s retrieval or store modes (described in Setting the Cache Retrieval and Store Modes). Which strategies best avoid stale reads are application dependent.

Controlling Whether Entities May Be Cached

The `javax.persistence.Cacheable`annotation is used to specify that an entity class, and any subclasses, may be cached when using the `ENABLE_SELECTIVE` or`DISABLE_SELECTIVE` cache modes. Subclasses may override the `@Cacheable`setting by adding a `@Cacheable` annotation and changing the value.

To specify that an entity may be cached, add a`@Cacheable` annotation at the class level:

```
@Cacheable
@Entity
public class Person { ... }
```

By default, the `@Cacheable` annotation is `true`. The following example is equivalent:

```
@Cacheable(true)
@Entity
public class Person{ ... }
```

To specify that an entity must not be cached, add a `@Cacheable` annotation and set it to`false`:

```
@Cacheable(false)
@Entity
public class OrderStatus { ... }
```

When the `ENABLE_SELECTIVE` cache mode is set, the persistence provider will cache any entities that have the `@Cacheable(true)` annotation and any subclasses of that entity that have not been overridden. The persistence provider will not cache entities that have `@Cacheable(false)` or have no `@Cacheable` annotation. That is, the `ENABLE_SELECTIVE` mode will cache only entities that have been explicitly marked for the cache using the `@Cacheable` annotation.

When the `DISABLE_SELECTIVE` cache mode is set, the persistence provider will cache any entities that **do not** have the `@Cacheable(false)` annotation. Entities that do not have `@Cacheable` annotations, and entities with the `@Cacheable(true)` annotation will be cached. That is, the `DISABLE_SELECTIVE` mode will cache all entities that have not been explicitly prevented from being cached.

If the cache mode is set to `UNDEFINED`, or is left unset, the behavior of entities annotated with `@Cacheable` is undefined. If the cache mode is set to `ALL` or `NONE`, the value of the `@Cacheable` annotation is ignored by the persistence provider.