dependency injection

Basically, instead of having your objects creating a dependency or asking a factory object to make one for them, you pass the needed dependencies in to the object externally, and you make it somebody else's problem. This "someone" is either an object further up the dependency graph, or a dependency injector that builds the dependency graph. A dependency as I'm using it here is any other object the current object needs to hold a reference to.

Dependency Injection Example

In this blog post, I will take a realistic example of having a web controller and a service. In practice, the controller would be responsible for managing requests from the web, and the service would interact with the persistence layer.

Product Class

```
 package com.springframework.domain;

public class Product {

   private String description;



   public Product(String description) {

      this.description = description;

   }



   public String getDescription() {

      return description;

   }



   public void setDescription(String description) {

      this.description = description;
   }
}
```

Interface

Java

```
package com.springframework.services;
import com.springframework.domain.Product;
import java.util.List;
public interface ProductService {
   List<Product> listProducts();
}
```

```
package com.springframework.services;
import com.springframework.domain.Product;
import java.util.List;
public interface ProductService {
   List<Product> listProducts();
}
```

implementaition

```
package com.springframework.services;
import com.springframework.domain.Product;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.List;
```

```
@Service
public class ProductServiceImpl implements ProductService {
   @Override
   public List<Product> listProducts() {
      ArrayList<Product> products = new ArrayList<Product>(2);
      products.add(new Product("Product 1 description"));
      products.add(new Product("Product 2 description"));
      return products;
   }
}
```

<u>controller</u>

```
package com.springframework.services;
import com.springframework.domain.Product;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
@Service
public class ProductServiceImpl implements ProductService {
   @Override
   public List<Product> listProducts() {
      ArrayList<Product> products = new ArrayList<Product>(2);
      products.add(new Product("Product 1 description"));
      products.add(new Product("Product 2 description"));
```

```
        return products;
    }
}
```

Example Execution Code

```
package com.springframework.services;
import com.springframework.domain.Product;
import org.springframework.stereotype.Service;
import java.util.ArrayList;
import java.util.List;
@Service
public class ProductServiceImpl implements ProductService {
    @Override
    public List<Product> listProducts() {
        ArrayList<Product> products = new ArrayList<Product>(2);
        products.add(new Product("Product 1 description"));
        products.add(new Product("Product 2 description"));
        return products;
    }
}
```